Universitatea
Transilvania
din Braşov

ŞCOALA DOCTORALĂ INTERDISCIPLINARĂ

Facultatea: Matematică şi Informatică

Drd. Corina-Ştefania CHIRIAC (căs. NĂNĂU)

Cercetări în teoria aşteptării şi în reţele cu toleranţă la întârzieri

Research in Queuing Theory and Delay Tolerant Networks

REZUMAT/ABSTRACT

Conducător ştiinţific
Prof. dr. Eleonor CIUREA

BRAŞOV, 2020

Universitatea Transilvania din Braşov

D-lui (D-nei) .....

COMPONENŢA

Comisiei de doctorat

Numită prin ordinul Rectorului Universităţii Transilvania din Braşov

Nr. ..... din .....

PREŞEDINTE:	**Prof. dr. Dorina RĂDUCANU** Decanul Facultăţii de Matematică şi Informatică Universitatea Transilvania din Braşov
CONDUCĂTOR ŞTIINŢIFIC:	**Prof. dr. Eleonor CIUREA** Universitatea Transilvania din Braşov
REFERENŢI:	**Prof. dr. Ioan TOMESCU** Universitatea din Bucureşti
	Prof. dr. Cornelius CROITORU Universitatea "Alexandru Ioan Cuza" din Iaşi
	Prof. dr. Ion FLOREA Universitatea Transilvania din Braşov

Data, ora şi locul susţinerii publice a tezei de doctorat: ....., ora ....., sala .....

Eventualele aprecieri sau observaţii asupra conţinutului lucrării vor fi transmise electronic, în timp util, pe adresa corina.nanau@unitbv.ro

Totodată, vă invităm să luaţi parte la şedinţa publică de susţinere a tezei de doctorat.

Vă mulţumim.

Cuprins

Content

Introducere

Această lucrare dezbate două aspecte importante de cercetare: elemente din teoria aşteptării şi caracteristici ale reţelelor cu toleranţă la întârzieri.

Multe sisteme din viaţa reală pot fi modelate ştiinţific folosind sisteme de aşteptare. Fenomenele studiate în acest context au un caracter aleator, fiind utilizate metode de investigaţie ale statisticii matematice. Sistemele de aşteptare se pot rezolva folosind două metode: cea analitică, bazată pe calcul matematic, sau cea bazată pe algoritmi de simulare, folosită în principal pentru problemele mai complexe.

A doua cercetare se referă la reţelele cu toleranţă la întârzieri. Acestea au apărut cu mai mulţi ani în urmă, ca proiecte finanţate de guvernul Statelor Unite ale Americii, datorită necesităţii de a dezvolta o tehnologie de comunicare cu posibilitatea de a susţine întârzieri foarte mari ale transmisiei mesajelor. În primă instanţă, această necesitate a apărut datorită încercării de a comunica în spaţiu.

După anul 2000, odată cu creşterea interesului cu privire la reţelele ad-hoc şi mobile, a crescut considerabil şi numărul cercetărilor şi al conferinţelor care au avut ca subiect tema aceasta. Cel care a introdus conceptul de reţea cu toleranţă la întârzieri şi întreruperi (Delay Tolerant Network în limba engleză), prescurtat DTN, a fost Kevin Fall, în anul 2002, iar bazele arhitecturii reţelei DTN au fost puse de către V. Cerf şi K. Scott în 2007. Datorită cercetărilor ulterioare în acest domeniu, au apărut metode noi de transmitere a datelor în reţele de tip DTN, au fost adaptate la situaţii reale şi îmbunătăţite performanţele metodelor deja existente, s-au definit tot mai multe clase de mobilităţi ale nodurilor reţelei, demonstrând că acesta este un domeniu de actualitate şi în continuă expansiune.

O reţea DTN poate interconecta dispozitive mobile, telefoane inteligente, senzori etc. Ele sunt într-o continuă mişcare, iar dinamica lor va influenţa permanent topologia reţelei. Rolul dispozitivelor care formează reţeaua este acela de transmitere şi recepţionare de date.

Reţeaua DTN se modelează sub formă de graf, ale cărui noduri se cunosc, iar prezenţa sau absenţa arcelor variază în timp. Cercetările au arătat că se poate stabili un grad de predictibilitate al configuraţiei reţelei. Predictibilitatea se poate calcula în funcţie de comportamentul nodurilor, în special în funcţie de modelul de deplasare a acestora, care poate fi unul aleator sau poate respecta un anumit tipar.

Datorită considerentelor prezentate, alegerea drumului pe care îl urmează mesajele care circulă în reţea este unul dintre principalele atribuţii ale algoritmilor de rutare. Pentru a maximiza transmiterea de mesaje, trebuie ales un drum optim. Având în vedere că majoritatea algoritmilor care deservesc reţeaua DTN răspândesc multe copii ale mesajelor, trebuie alese şi strategii conexe de evitare a supraîncărcării reţelei. Aceste strategii sunt reprezentate de politicile de gestiune a capacităţii de stocare de date asociate nodurilor.

Contextele în care se pretează implementarea unei astfel de reţele sunt diverse: reţele de senzori care culeg informaţii despre mediu, reţele care urmăresc dezvoltarea animalelor sălbatice în mediul lor natural, reţele pe câmpul de luptă, reţele de gestionare a salvării de persoane în urma unui dezastru natural, reţele care interconectează zone rurale sau defavorizate, care nu au acces la electricitate sau Internet.

Având în vedere multitudinea de oportunităţi pe care o oferă reţeaua DTN, am ales spre rezolvare gestionarea salvării persoanelor afectate de un cutremur de proporţii. Pentru aceasta, am implementat un algoritm de rutare care se potriveşte paradigmei DTN şi care maximizează transmiterea de mesaje cu caracter urgent, în vederea coordonării operaţiunii de salvare a victimelor cutremurului. Pentru a demonstra eficienţa algoritmului propus, am realizat o simulare a contextului real prezentat, în care am evidenţiat rata de livrare a mesajelor oferită de noul algoritm.

Structura acestei lucrări este următoarea:

Capitolul 1, intitulat **Noţiuni fundamentale**, descrie o serie de noţiuni de bază privind reţelele cu toleranţă la întârzieri, provocările la care sunt dispuse şi criteriile de performanţă ale unei astfel de reţele. Sunt sintetizaţi unii dintre cei mai importanţi algoritmi de rutare existenţi şi se oferă de asemenea o prezentare cronologică a principalelor strategii de gestiune a bufferului. Tot în acest capitol este descris simulatorul utilizat pentru testarea funcţionării reţelei.

Capitolul 2, intitulat **Algoritmi de simulare în teoria aşteptării**, reprezintă fundamentul teoretic pe care se sprijină rezolvarea multor probleme care pot apărea într-o reţea cu toleranţă la întârzieri, deoarece fiecare nod al unei reţele DTN poate fi definit ca un sistem de aşteptare individual. În acest capitol a fost efectuată o paralelă a două sisteme de aşteptare cu revenire, în care clienţii sosesc individual şi în loturi. Cele două sisteme sunt rezolvate cu ajutorul algoritmilor de simulare, iar la finalul capitolului sunt prezentate comparativ rezultatele factorilor de eficienţă.

Capitolul 3, intitulat **Probleme în reţele cu toleranţă la întârzieri**, abordează două situaţii care pot apărea într-o reţea DTN, şi anume:

- identificarea fluxului maxim - reţeaua dinamică iniţială a fost transformată într-o reţea statică înaintea calculării fluxului maxim.
- maximizarea transmiterii de mesaje - pentru a atinge acest obiectiv, a fost dezvoltat un nou algoritm de rutare, numit **MaxDelivery**, care deţine o strategie de gestionare a spaţiului de stocare de date.

Tot în capitolul 3 sunt prezentate şi rezultatele simulărilor efectuate pentru a testa eficienţa algoritmului MaxDelivery în comparaţie cu unii dintre cei mai cunoscuţi algoritmi de rutare în DTN. La finalul capitolului este descris un context real de utilizare al algoritmului propus şi sunt prezentate rezultatele pe care acesta le obţine în cazul gestiunii unei operaţiuni de salvare în urma unui cutremur.

Capitolul 4 prezintă concluziile cercetării efectuate în această lucrare şi indică direcţiile viitoarei cercetări.

Articolele publicate sau trimise spre publicare:

1. articole publicate în reviste:

- [23] Florea, I.L. and **Năănău C.Ş.**, An algorithmic approach of retrial queuing system with one serving station. Part I: The description of the simulation algorithm, Bulletin of the Transilvania University of Braşov, 2013. (revistă indexată SCOPUS)
- [24] Florea, I.L. and **Năănău C.Ş.**, An algorithmic approach of retrial queuing system with one serving station. Part II: The implementation of the simulation algorithm, Bulletin of the Transilvania University of Braşov, 2014. (revistă indexată SCOPUS)
- [25] Florea, I.L. and **Năănău C.Ş.**, A simulation algorithm for a single server retrial queuing system with batch arrivals, Analele ştiinţifice ale universităţii Ovidius Constanţa, 2015. (revistă **cotată ISI cu factor de impact 0.638**)
- [52] **Năănău C.Ş.**, An overview of Delay Tolerant Networks and routing protocols, Bulletin of the Transilvania University of Braşov, 2018. (revistă indexată SCOPUS)
- [53] **Năănău C.Ş.**, Example of routing protocols in Delay Tolerant Networks, Bulletin of the Transilvania University of Braşov, 2019. (revistă indexată SCOPUS)
- [54] **Năănău C.Ş.**, Maximum flow in buffer-limited Delay Tolerant Networks. The static approach, Bulletin of the Transilvania University of Braşov, 2020. (revistă indexată SCOPUS)

2. articole prezentate în conferinţe internaţionale:

   - [55] **Năănău, C.Ş.**, Queuing Theory Application on DTN Buffer Management, in Proceeding of the 8th International Conference on Computers Communications and Control (ICCCC 2020), Oradea, 2020. (Categoria D)

3. articole acceptate la conferinţe internaţionale (a se vedea Anexa 1):

   - [56] **Năănău, C.Ş.**, MaxDelivery: a new approach to a DTN Buffer Management, Proceeding of 21st IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WOWMOM 2020), Cork, Ireland - articol în formă redusă, trimis la forumul dedicat doctoranzilor. (Categoria A)

4. articole trimise la reviste spre publicare:

   - [17] Deaconu, A., Udroiu, R. and **Năănău, C.Ş.**, Data or Physical Packages Delivery in Isolated, Disaster or Quarantined Areas Using DTN Based Algorithms for Unmanned Aerial Vehicles, IEEE Access, (revistă **cotată ISI cu factor de impact 3.745**, Categoria A)

* * *
*

Doresc să îi mulţumesc domnului profesor dr. Eleonor Ciurea, coordonatorul acestei teze, pentru răbdarea şi înţelegerea pe care mi le-a arătat în toată această perioadă, pentru sprijinul şi încurajarea acordate în vederea finalizării acestei lucrări. Doresc să le mulţumesc de asemenea şi membrilor comisiei de îndrumare pentru răbdarea şi implicarea dânşilor în activitatea mea ştiinţifică.

Capitolul 1

Noţiuni fundamentale

1.1 Reţele cu toleranţă la întârzieri. Privire de ansamblu

1.1.1 Informaţii generale

Reţelele cu toleranţă la întârzieri şi întreruperi, în limba engleză "Delay and Disruption Tolerant Networks", prescurtat DTN, s-au concretizat atât la nivel teoretic cât şi practic în anul 2003. Kevin Fall a lansat astfel o arhitectură suplimentară, suprapusă arhitecturii de reţea existente până în acel moment. Aceste informaţii sunt prezentate în [21].

Conform cu [81], multe medii de comunicare nu pot fi în conformitate cu ipotezele fundamentale ale reţelei Internet (conexiune continuă şi bidirecţională între sursa de date şi destinaţia acestora, întârzieri de transfer de ordinul milisecundelor, rate simetrice şi consistente de transfer al datelor, număr mic de pierderi sau coruperi de date). Aceste constrângeri sunt impuse de protocoalele de comunicare folosite de Internet, în special de către TCP/IP (Transmission Control Protocol/Internet Protocol în limba engleză), cel mai comun protocol utilizat. Aceste protocoale pleacă de la premisa că există o cale continuă pe care să o poată parcurge mesajele de la sursă la destinaţie, ceea ce nu este valabil, de exemplu, într-o reţea din spaţiul cosmic.

O reţea de tip DTN poate fi reprezentată formal ca un graf orientat, neorientat sau mixt. Având în vedere dinamica unei reţele cu toleranţă la întârzieri, graful folosit pentru reprezentare trebuie să fie unul variabil în funcţie de timp.

Reţeaua poate avea atât noduri fixe cât şi mobile. Nodurile mobile pot fi: persoane, autovehicule, sateliţi artificiali, drone etc. Nodurile fixe pot fi: antene, instituţii sau orice punct fix pe care este montat un dispozitiv care are configurat un protocol de comunicare cunoscut de toţi membrii reţelei.

Drumurile dintr-o reţea variabilă în timp se construiesc respectând constrângerea ca următorul arc, pentru a fi parcurs, trebuie să fie activ după traversarea arcului curent. Ca şi într-un graf orientat, existenţa unui drum de la nodul i la nodul j nu justifică şi existenţa unui drum de la nodul j la nodul i. Drumurile în funcţie de timp nu sunt tranzitive. Cu alte cuvinte, dacă există un drum de la nodul i la nodul j şi un alt drum de la nodul j la nodul k, nu înseamnă că există drum şi de la i la k. Acest tip de drum este el însuşi variabil în timp,

fiind valid doar pe un anumit interval.

Mai multe detalii despre reţelele temporale se găsesc în [29].

În [65] este prezentată o scurtă descriere a reţelelor premergătoare celei cu toleranţă la întârzieri, şi anume reţeaua wireless şi cea ad-hoc.

Pe lângă dispozitivele conectate prin fir, există şi o categorie aparte de dispozitive conectate wireless în Internet, precum telefoanele mobile sau alte dispozitive inteligente. Acestea oferă posibilitatea de a ne deplasa aproape oriunde şi de a avea conexiune la Internet. Reţeaua wireless este o combinaţie de dispozitive fixe şi mobile care se deconectează de obicei pentru o perioadă foarte scurtă de timp de dispozitivele fixe principale. Reţelele fără fir au de asemenea o infrastructură fixă, iar topologia lor este similară cu cea a reţelelor cu conexiune prin fir, putându-se determina cu uşurinţă căi continue între noduri.

În cadrul reţelelor fără fir există o categorie numită MANET (Mobile Ad Hoc Networks) [78], în care două dispozitive sunt conectate între ele doar dacă se găsesc în raza de acţiune a unuia faţă de celălalt. Deoarece reţelele MANET sunt alcătuite din dispozitive mobile, acestea pot avea o mobilitate mare, iar legăturile dintre ele se pot întrerupe, ieşind din raza de acţiune. Astfel, aceste legături au perioade succesive de activitate şi de inactivitate. Dispozitivele din acest tip de reţea îşi actualizează permanent căile de comunicare pe baza informaţiilor despre modificările topologice. Modificările dese ale căilor creează posibilitatea existenţei unui drum continuu între două noduri, iar comunicarea nu va avea de suferit în acest sens.

Atunci când avem în vedere comunicarea prin satelit sau când legăturile dintre dispozitive se pot întrerupe datorită unor defecţiuni sau unor limitări de resurse, reţeaua se poate fragmenta astfel încât nu se mai poate determina un drum continuu între oricare două noduri. Apare astfel imposibilitatea prezicerii timpului în care va ajunge la destinaţie un mesaj care a plecat de la nodul sursă.

Pentru a întâmpina aceste provocări, a fost dezvoltată o reţea care să extindă reţeaua MANET şi care oferă posibilitatea stocării mesajelor pentru o perioadă mai lungă de timp şi a redirecţionării acestora în momentul apariţiei unei conexiuni. Această extindere a reţelei MANET este reţeaua DTN, cea care tolerează întârzieri mari de transmitere a mesajelor. Reţeaua DTN a fost concepută pentru a fi folosită acolo unde nu există Internet sau în mediile în care sunt prezente întreruperi frecvente şi de lungă durată ale reţelei.

Ceea ce aduce nou reţeaua DTN faţă de reţeaua Internet este posibilitatea comunicării în medii caracterizate prin:

- conexiune intermitentă, care poate fi periodică sau întâmplătoare
- întârzieri lungi şi cu valori ce pot varia consistent în transmiterea mesajelor
- rate de transfer ale mesajelor cu o asimetrie destul de ridicată, care este inacceptabilă protocoalelor de comunicare din Internet
- rată a erorilor ridicată pe parcursul unei conexiuni între noduri, a căror corectare generează şi mai mult trafic.

Conexiunea intermitentă din reţeaua DTN este provocată, în afară de mobilitatea foarte mare a nodurilor, de încercarea de a conserva energia dispozitivelor reţelei, de atacurile asupra reţelei, de unele defecţiuni sau dezastre naturale.

Chiar dacă iniţial reţeaua DTN a apărut în contextul comunicărilor spaţiale, odată cu cercetările tot mai intense în domeniu, cu dezvoltarea tehnologiei şi a dispozitivelor care o folosesc, aplicaţiile reţelei de tip DTN şi-au extins din ce în ce mai mult aria. Astfel putem avea: reţele pe câmpul de luptă, reţele de senzori care pot monitoriza diverse obiective (dezvoltarea animalelor în sălbăticie, clima, mediul înconjurător etc.), reţele de comunicare subacvatică, reţele rurale sau din zone defavorizate, reţele de comunicare în caz de cataclisme naturale, reţele vehiculare etc.

Modalităţi de funcţionare

În 2007 au fost introduse elemente noi de arhitectură pentru reţeaua de tip DTN şi au fost prezentate în lucrările [13],[64].

Arhitectura unei reţele DTN implementează un mecanism de **stocare, transport şi transmitere mai departe a mesajelor** (store, cary and forward messages, în limba engleză), facilitat de protocolul de împachetare (bundle protocol, în limba engleză). Acest protocol permite stocarea şi transmiterea mai departe a unor pachete de date întregi sau a unor fragmente de pachete, făcând posibilă comunicarea între noduri, indiferent de tipurile de protocoale de nivel inferior pe care le implementează.

Pe baza acestui protocol, nodul care transmite un pachet de date va solicita nodului de contact acceptarea "custodiei" pachetului. Un custode al pachetului va trebui să îl stocheze până când un alt nod acceptă custodia sau până când expiră timpul de viaţă al pachetului. Pentru realizarea unei transmisii sigure, timpul de viaţă trebuie să fie suficient de mare.

Nodurile unei reţele DTN au posibilitatea de stocare de date pe o perioadă nelimitată, spre deosebire de ruterele din Internet care pot stoca mesajele doar câteva milisecunde în microcipurile sau bufferele cu care sunt dotate.

Provocări ale unei reţele de tip DTN

Reţelele cu toleranţă la întârzieri sunt supuse unor provocări care nu sunt prezente în cele tradiţionale. Acestea provin din necesitatea de a face faţă deconectărilor care influenţează expedierea mesajelor. În [34] apar o serie de astfel de provocări ce influenţează în mod radical funcţionarea unei astfel de reţele.

Una dintre problemele care pot apărea într-o reţea de tip DTN este adaptarea la resurse limitate, iar prin resurse vom înţelege atât resurse de nivel fizic, cât şi resurse logice. Avem astfel limitări ale spaţiului de stocare al nodurilor, ale energiei electrice, ale timpului de comunicare între noduri, ale puterii de calcul şi de procesare a datelor etc.

Una dintre principalele probleme pe care trebuie să le gestioneze o reţea DTN este **orarul de contact**. Pentru a obţine performanţe ridicate de comunicare, trebuie eficientizat timpul de aşteptare până la intrarea nodurilor în contact cu alte noduri. Într-o astfel de reţea, acest timp poate varia de la secunde până la zile întregi. O soluţie a eficientizării timpului de aşteptare poate fi aceea a stabilirii unor orare de contact, indiferent de precizia lor. O extremă poate fi considerată aceea în care orarul de contact este foarte exact, cum ar fi cazul reţelelor din spaţiu, unde deconectările sunt datorate interpunerii unor obiecte care se deplasează după un program bine stabilit. Se poate considera de asemenea o reţea DTN în care nodurile sunt montate pe autobuzele din oraş. Aceste autobuze au un orar, dar care

nu poate fi respectat cu acurateţe din cauza traficului, a defecţiunilor, a accidentelor etc. Astfel se poate ajunge la variaţii semnificative a timpilor de sosire în staţie. La cealaltă extremă sunt situate reţelele a căror noduri au un orar de deplasare complet aleatoriu, cum sunt reţelele ad-hoc.

Capacitatea legăturilor dintre noduri este o altă provocare a reţelei DTN. Aceasta se referă la cantitatea de date care poate fi schimbată între noduri şi depinde atât de tehnologia folosită pentru conectare, cât şi de durata conexiunii. Dacă volumul traficului de date din reţea este foarte mic în comparaţie cu capacitatea legăturilor, atunci poate fi rezonabilă ignorarea capacităţii legăturilor. Dacă volumul traficului creşte datorită creşterii numărului de utilizatori sau datorită schimbului de mesaje de dimensiuni foarte mari, atunci capacitatea legăturilor dintre noduri devine un factor foarte important.

Un factor care presupune o reală provocare în funcţionarea optimă a reţelei, este **spaţiul de stocare din buffer**. Aşa cum am mai precizat, nodurile care compun reţeaua DTN au ca particularitate prezenţa unui buffer în care sunt stocate mesaje pe o periodă îndelungată. Pentru a rezolva această problemă, nodurile trebuie să aibă o bună strategie de gestionare a spaţiului de stocare.

Puterea de procesare este un alt factor important în reţelele cu toleranţă la întârzieri. Dispozitivele conectate în reţea pot avea dimensiuni foarte mici şi implicit capacitate mică de procesare, iar acestea nu vor putea rula mecanisme complexe de rutare.

Consumul de energie electrică reprezintă o altă provocare de care trebuie să ţină seama reţelele cu toleranţă la întârzieri. Unele noduri au resurse limitate de energie datorită mobilităţii ridicate sau pentru că se găsesc într-o locaţie în care nu au acces cu uşurinţă la reţeaua electrică. Procesul de rutare este consumator de energie la trimiterea, primirea sau stocarea datelor. Astfel, strategiile de rutare care trimit puţine mesaje sau realizează puţine procesări sunt mai eficiente din punct de vedere energetic.

Criterii de evaluare a performanţei

Ca în orice desfăşurare a unei acţiuni, trebuie stabilite anumite criterii care să cuantifice performanţa acelei acţiuni. În mod similar, funcţionarea unei reţele de tip DTN are stabilite o serie de criterii care îi definesc performanţele. Conform cu [34], acestea sunt:

- **Rata livrării mesajelor** - este unul dintre criteriile cele mai importante de evaluare într-o reţea DTN. În general, în reţelele cu toleranţă la întârzieri, probabilitatea ca un mesaj să fie eliminat din reţea este mult mai mică decât aceea de a ajunge la destinaţie cu întârziere. Acest lucru depinde în mare măsură de topologia şi de strategia de funcţionare a reţelei. Astfel, rata livrării mesajelor este definită ca numărul de mesaje livrate corect la destinaţie în unitatea de timp prestabilită.

- **Latenţa (întârzierea de transmitere a mesajelor)** - este definită prin perioada de timp dintre momentul generării mesajului de către nodul sursă şi momentul primirii acestuia de către nodul destinaţie. Acest criteriu este important deoarece multe aplicaţii pot beneficia de un timp scurt de aşteptare, chiar dacă sunt capabile să tolereze perioade îndelungate de aşteptare. Astfel se limitează intervalul de timp în care este utilă primirea datelor.

- **Transmisia** - este acel factor care defineşte numărul de mesaje transmise în timpul

unei conexiuni dintre noduri. Unele strategii de funcţionare trimit mai multe mesaje decât altele, pentru că folosesc mai multe copii ale aceluiaşi mesaj, suprasolicitând canalul de transmitere a datelor.

În [76] apar şi **numărul de hopuri** (opriri, staţionări) parcurse de către mesaj, dar şi **rata de supraîncărcare** ca factori de eficienţă ai reţelei. Numărul de hopuri influenţează întârzierea de transmitere a mesajelor, iar supraîncărcarea se reflectă în costul mediu de transmisie al unui mesaj. Cu alte cuvinte, supraîncărcarea indică numărul de mesaje nenecesare transmise în medie pentru un mesaj care ajunge la destinaţie. Acesta este un factor care influenţează performanţele reţelelor care nu permit un consum ridicat de energie.

1.1.2 Rutarea în reţelele cu toleranţă la întârzieri

Una dintre părţile esenţiale ale telecomunicaţiilor, care este foarte importantă şi în cazul reţelelor de tip DTN, este rutarea. **Rutarea este decizia luată în procesul de identificare a drumului optim pe care trebuie să îl urmeze mesajele pentru a ajunge la destinaţie.** Fără existenţa acestui mecanism, nodurile nu ar şti ce mesaje să transmită şi cărui nod să transmită mesaje pentru a efectua operaţii optime în reţea. În DTN rutarea este importantă în mod special deoarece contactele dintre noduri sunt rare şi de scurtă durată, iar fiecare oportunitate de conectare trebuie exploatată la maxim. Din acelaşi motiv, rutarea este mai complexă în acest caz decât în reţelele tradiţionale, unde toate conexiunile se cunosc de la bun început şi este mult mai simplă stabilirea unui drum optim al mesajelor.

Scopul rutării este optimizarea ratei de livrare a mesajelor, a întârzierii de transmitere a mesajelor şi minimizarea impactului asupra resurselor reţelei.

Reţelele de tip DTN au o varietate foarte mare de aplicaţii, însă una dintre cele mai interesante este reţeaua bazată pe telefoane inteligente sau pe alte dispozitive care se găsesc asupra oamenilor. Acesta este motivul principal pentru care, în [19] se afirmă faptul că este nevoie de a dezvolta soluţii pentru reţele cu un şablon de mobilitate uman, întrucât mişcarea oamenilor şi comportamentul lor nu sunt complet aleatorii, ci sunt predictibile.

Clasificarea strategiilor de rutare

În literatura de specialitate se găsesc mai multe criterii de clasificare a strategiilor de rutare.

În [81], întâlnim strategii de rutare care iau în considerare **tipul de conexiune dintre noduri**, şi anume: conectarea oportunistă a nodurilor şi conectarea în funcţie de un orar prestabilit.

Spunem că nodurile reţelei se conectează în mod oportunist atunci când emiţătorul şi receptorul intră în contact fără un orar prestabilit. Astfel, persoanele în mişcare, vehiculele, aeronavele sau sateliţii pot intra accidental în contact şi schimba informaţii.

Conectarea pe baza unui orar este întâlnită în special în spaţiu unde totul este în mişcare, iar nodurile care pot fi conectate se deplasează pe o orbită bine definită. Acest fapt implică posibilitatea stabilirii poziţiei nodurilor la un anumit moment. Conectarea pe baza unui orar se poate întâlni de asemenea şi în viaţa cotidiană, cu precădere în cazul mijloacelor de transport în comun.

O altă clasificare importantă a protocoalelor de rutare se face în funcţie de **numărul de replici ale mesajului**, aşa cum se poate găsi în [34],[49],[52],[59],[76]. Avem astfel protocoale de rutare care se bazează pe o singură copie a mesajului, numite în literatura din limba engleză "forward-based", precum şi protocoale de rutare care se bazează pe mai multe copii ale mesajului, întâlnite în literatura de specialitate cu numele de "flood-based" (bazate pe inundarea reţelei).

Strategiile care utilizează o singură copie a mesajelor sunt mai puţine spre deosebire de cele care răspândesc în reţea un număr mare de replici. Răspândirea de copii creşte şansa ca mesajul să ajungă la destinaţie şi reduce timpul petrecut în reţea. Abordarea aceasta afectează costul transportului mesajului deoarece se va consuma lăţimea de bandă şi spaţiul din buffer. Varianta cea mai ieftină este aceea ca mesajul să se găsească într-un singur exemplar în reţea, dar când apare o situaţie neprevazută, mesajul poate fi pierdut uşor.

În secţiunea următoare vom putea observa că strategiile de rutare bazate pe inundarea reţelei pot aborda replicarea în mai multe maniere, aşa cum se constată în [34]:

- transmiterea cu două opriri - care presupune transmiterea mesajului de la sursă la primele n noduri cu care aceasta intră în contact. Aceste noduri vor transmite mesajul mai departe numai în cazul în care intră în contact direct cu destinaţia. Reţelele în care mobilitatea nodurilor este aleatoare nu vor avea succes cu această strategie, dar dacă mobilitatea este bazată pe un orar, atunci livrarea mesajelor va fi cu succes. Această strategie a fost propusă în [51] pentru cazul în care rutarea ad-hoc nu poate găsi o cale de conectare a nodurilor.

- inundarea arborescentă - se bazează pe cea precedentă, doar că nodul care a primit o copie a mesajului nu o va livra doar destinaţiei, ci şi unui nod intermediar. Astfel, nodurile la care ajung copii ale mesajului, formează un arbore.

- inundarea *epidemică* - a fost introdusă de către Vahdat şi Becker în [79] pentru a garanta furnizarea unui numar suficient de mare de replici ale mesajului astfel încât destinaţia să primească şi ea o copie.

Există mai multe modalităţi de a limita numărul de copii ale mesajului. Una dintre cele mai simple modalităţi este restricţionarea numărului de hopuri până la destinaţie, limitând adâncimea arborelui. O rafinare a acestei modalităţi constă în limitarea numărului de copii realizate de un nod.

Protocoalele de rutare pot fi clasificate în funcţie de **cantitatea de cunoştinţe despre reţea** pe care o folosesc pentru trimiterea mesajelor. Această clasificare se întâlneşte în [32],[34].

O extremă ar fi aceea ca nodul să ia decizia de rutare fără să cunoască nimic despre reţea (în afară de nodurile cu care a intrat în contact). Un dezavantaj ar fi acela că această strategie nu poate lua o decizie optimă întrucât nu se poate adapta la condiţiile reale de reţea. O altă extremă ar fi aceea ca nodurile să cunoască întregul orar al tuturor nodurilor din reţea, caz în care programul de rutare ar lua decizii cu o acurateţe ridicată cu privire la alegerea celei mai bune căi de rutare a mesajelor.

Există şi varianta intermediară de cunoaştere a unor informaţii parţiale despre reţea, pe care să le descopere pe parcurs. Aceste informaţii se referă la caracteristicile topologiei de reţea şi la dinamica acesteia, precum şi la cerinţele de trafic.

Abordarea care necesită cele mai puţine informaţii este aceea prin care se atribuie coordonate fiecărui nod, iar pentru estimarea costului de transport al mesajelor se foloseşte o funcţie de distanţă. Coordonatele unui nod pot fi cele fizice (coordonatele GPS) aşa cum s-a studiat în [48] sau coordonate relative la spaţiul topologiei de reţea, care au fost studiate în [58].

O altă modalitate de a clasifica protocoalele de rutare este în funcţie de **abordarea socială**. Nodurile unei reţele pot avea comportament social datorită faptului că reprezintă dispozitive pe care oamenii le poartă cu ei pentru a testa funcţionarea reţelei DTN. În acest caz, deplasarea nodurilor în reţea este identică deplasării cotidiene a oamenilor. Unele protocoale de rutare utilizează aceste cunoştinţe pentru a anticipa deplasările viitoare ale nodurilor. Această abordare se întâlneşte în [15],[67].

Clasificarea strategiilor de rutare poate fi efectuată şi în funcţie de **momentul stabilirii rutei mesajelor** [32]. Într-o abordare proactivă, ruta este stabilită de către nodul sursă, care o codifică în interiorul mesajului, şi nu poate fi modificată ulterior. În abordarea reactivă, avem rutarea **per hop**, în care fiecare nod din calea mesajului stabileşte doar următorul hop, pe baza informaţiilor pe care le are din reţea.

Ultima clasificare amintită este cea **bazată pe administrarea bufferului**. Avem astfel protocoale care ţin cont de acest lucru şi protocoale care nu ţin cont. Protocoalele care nu ţin cont de buffer, fie consideră că spaţiul de stocare al nodurilor este nelimitat, ceea ce este nerealist, fie utilizează metoda FIFO pentru eliminarea sau transmiterea mai departe a mesajelor. Această abordare va fi discutată mai detaliat în capitolul 3.

Algoritmi de rutare

După cum am amintit în subparagraful precedent, protocoalele de rutare care folosesc o singură copie a mesajului sunt destul de puţine. Dintre acestea, sunt prezentate Direct Delivery [74], DTLSR (Delay Tolerant Link State Routing) [87] şi SABR (Schedule Aware Bundle Routing) [87].

Direct Delivery este cea mai simplă abordare a unui protocol de acest tip. Nodul va livra mesajul doar destinaţiei, în momentul în care intră în contact cu aceasta. Abordarea sa simplistă se dovedeşte a fi destul de utilă doar pentru cazul în care nodurile din reţea se întâlnesc des.

Protocolul **DTLSR** utilizează starea legăturilor dintre noduri. Fiecare legătură va primi o valoare care va creşte continuu până când va ajunge la un anumit maxim. Doar în momentul atingerii valorii maxime va fi eliminată legătura din graf. Dacă nodurile se reîntâlnesc atunci valoarea asociată legăturii se resetează. Se recurge la acest mijloc de a păstra încă o perioadă legăturile inactive în graf pentru a se putea calcula drumurile de rutare ale mesajelor, în speranţa că legăturile vor fi reactivate în curând.

Protocolul **SABR** este o extensie a protocolului Contact Graph Routing [27], care utilizează conexiuni între noduri pe baza unui orar, dar şi conexiuni oportuniste. Prioritatea de transmitere se va acorda mesajelor sosite cel mai recent, iar pachetele sunt rutate pe un graf cu conexiuni variabile în timp. În cazul conexiunilor oportuniste, rutarea se face către vecinii nou descoperiţi sau către vecinii care au un istoric de conexiune cu nodul în cauză.

Printre algoritmii de rutare care creează mai multe copii ale mesajelor, amintim: Epidemic [79], Spray and Wait [75], PRoPHET [44], MaxProp [11], RAPID [5]. Câteva observaţii despre aceşti algoritmi se găsesc şi în [53].

Epidemic este forma de bază a protocoalelor de rutare bazate pe inundarea reţelei. Algoritmul său presupune crearea mai multor copii ale mesajului. Acest algoritm are o întârziere de livrare scăzută, dar este destul de costisitor deoarece consumă o cantitate ridicată de resurse de reţea datorită duplicării excesive a mesajelor. Maniera de lucru, conform cu [79]: când se întâlnesc două noduri, acestea îşi împărtăşesc informaţii despre propriile buffere şi identifică pachetele care lipsesc. După aceasta îşi transferă pachetele lipsă, astfel că la sfârşitul procesului, cele două noduri au acelaşi conţinut în buffer. Acest proces se repetă de fiecare dată când două noduri intră în contact. Această abordare s-a dovedit a fi eficientă în cazul reţelelor care nu sunt foarte încărcate, dar în cazul celor încărcate, abordarea va deveni complet ineficientă.

Spray and Wait este un protocol de rutare prin inundare care necesită prezenţa unor spaţii de stocare destul de mari ataşate nodurilor. Algoritmul are două faze: una de împrăştiere a mesajelor şi una de aşteptare. Acest algoritm a apărut în [75] şi circulă în mai multe variante, în funcţie de numărul de copii ale mesajului care se răspândesc.

Varianta standard sau Vanilla este cea în care nodul sursă generează un mesaj şi îi ataşează un număr L de copii. Faza de împrăştiere a mesajelor se încheie atunci când nodul sursă rămâne cu o singură copie, iar alte $L-1$ noduri au şi ele câte o copie. În faza de aşteptare, nodurile aşteaptă oportunitatea de a se întâlni cu destinaţia mesajului.

O altă variantă este cea binară, cea în care, în momentul întâlnirii cu un alt nod care nu are în bufferul său o copie a mesajului, nodul sursă îi transmite acestuia responsabilitatea transmiterii mai departe a jumătate din numărul de copii pe care le are el în buffer. Şi tot aşa până rămâne cu un singur mesaj, moment în care nodul intră în faza de aşteptare, în care se comportă ca în cazul protocolului Direct Delivery. Nu doar nodul sursă are acest comportament, ci şi celelalte noduri care au primit o copie a mesajului împreună cu responsabilitatea împrăştierii lui în reţea.

Acest algoritm poate fi aplicat cu success în cazul reţelelor de dimensiuni mici, cu o deplasare aleatoare a nodurilor. În cazul în care mişcarea nodurilor are distribuţie uniformă este mai avantajoasă versiunea binară decât cea standard, deoarece va împrăştia mai rapid în reţea copiile mesajului.

PRoPHET este un protocol de rutare similar cu Epidemic, care face schimb de mesaje doar în cazul în care nodul întâlnit are o probabilitate mai mare de a livra la destinaţie acele mesaje. Acesta este un protocol de rutare probabilistic, denumirea sa reprezentând acronimul de la **Probabilistic Routing Protocol using History of Encounters and Transitivity** [44].

Modalitatea de calcul a *probabilităţii de livrare* a unui nod N este următoarea:

Vom nota cu P funcţia de probabilitate a livrării. Aceasta are 2 parametri: N şi D, unde N este nodul curent, iar D este un nod destinaţie. Fiecare nod va stoca valoarea lui P pentru oricare nod destinaţie D din reţea. Dacă nu se cunoaşte probabilitatea livrării mesajelor spre un anumit nod, atunci valoarea lui P va fi zero pentru acel nod. La fiecare întâlnire dintre două noduri, acestea îşi recalculează probabilităţile, bazându-se pe trei reguli:

1. Atunci când nodul N întâlneşte un nod M, probabilitatea lui M va fi crescută astfel:

$$P(N,M)_{new} = P(N,M)_{old} + (1 - P(N,M)_{old}) \cdot L_{encounter} \tag{1.1}$$

   $L_{encounter}$ fiind o constantă.

2. Se actualizează probabilitatea pentru toate nodurile destinaţie D (în afară de nodul M) astfel:
$$P(N,D)_{new} = P(N,D)_{old} \cdot Y^K \tag{1.2}$$
unde Y este o constantă de actualizare, iar K este numărul de unităţi de timp care s-au scurs de la ultima actualizare.

3. Probabilităţile sunt schimbate între N şi M, iar tranzitivitatea funcţiei de probabilitate este folosită pentru a actualiza valorile acesteia pentru destinaţiile D astfel:
$$P(N,D)_{new} = P(N,D)_{old} + (1 - P(N,D)_{old}) \cdot P(N,M) \cdot \beta \tag{1.3}$$
unde β este o constantă de scalare.

Acest algoritm este pretabil reţelelor care au noduri cu deplasare aleatoare şi întâlniri oportuniste.

Pentru îmbunătăţirea ecuaţiilor, în [9] a fost introdus un factor de îmbunătăţire α. Se poate demonstra din punct de vedere matematic faptul că pentru $\forall\alpha \in [0,1]$ şi $P \in [0,1]$ avem $P^\alpha \geq P$. Pornind de la acest fapt, în [9] este demonstrat că noul algoritm are o eficienţă mai bună decât varianta clasică a algoritmului PRoPHET.

MaxProp este un protocol de rutare care se bazează pe prioritizarea transmiterii pachetelor şi a abandonării acestora [11]. Este unul dintre primele protocoale de rutare din DTN care se ocupă şi de gestiunea spaţiului de stocare din noduri. Acesta are la bază un buffer de mesaje sortate, în funcţie de care se va decide care sunt mesajele cu prioritate la transmitere şi care sunt cele cu prioritate la ştergere.

Specificul acestui algoritm constă în obţinerea probabilităţilor de întâlnire ale nodurilor. Astfel, fiecare nod va stoca un vector cu $n-1$ elemente (unde n reprezintă numărul nodurilor din reţea). În aceşti vectori, nodurile reţin probabilităţile de a întâlni celelalte noduri. În primă fază se iniţializează toate nodurile cu probabilităţi egale cu $\frac{1}{n-1}$. În momentul în care se întâlnesc două noduri, acestea adaugă o unitate la vechea valoare a probabilităţii lor de întâlnire, după care se refac probabilităţile de întâlnire şi pentru celelalte noduri.

Avem spre exemplu reţeaua din Fig. 1.1, unde considerăm nodul 1 ca fiind nod curent. Acesta va avea un vector care conţine probabilităţile de întâlnire ale nodului 1 cu nodurile 2, 3, 4 şi 5, care vor avea iniţial valorile $\frac{1}{5-1} = 0.25$. Astfel, vectorul nodului 1 va fi de forma: [0.25, 0.25, 0.25, 0.25].

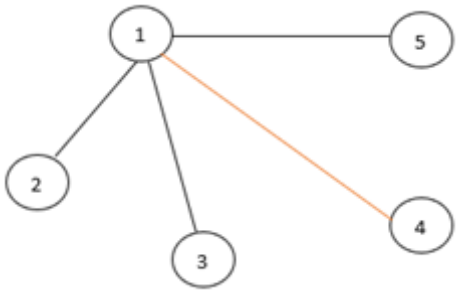


Figura 1.1: Graf cu 5 noduri, cu muchia [1,4] activă [53]

Atunci când se întâlneşte nodul 1 cu nodul 4 se va adăuga o unitate la valoarea probabilităţii întâlnirii celor două noduri, iar vectorul va ajunge astfel: [0.25, 0.25, 1.25, 0.25]. Se vor

reface probabilităţile elementelor astfel încât suma lor să fie egală cu 1, iar vectorul va avea forma aceasta: [0.125, 0.125, 0.625, 0.125]. Dupa ce îşi recalculează probabilităţile, nodurile schimbă între ele vectorii lor de probabilităţi. Ideal, toate nodurile vor conţine vectorii actualizaţi ai tuturor celorlalte noduri. Având cei n vectori, fiecare nod poate calcula calea cea mai scurtă a mesajelor către destinaţie pe baza unei parcurgeri în adâncime, unde dimensiunea drumului indică probabilitatea ca legătura să nu aibă loc.

Dimensiunea drumului va reprezenta astfel însumarea dimensiunilor muchiilor care îl compun. Un nod va face aceste calcule pentru toate mesajele pe care le stochează în buffer.

Funcţionalitatea prezentată mai sus reprezintă nucleul algoritmului MaxProp. Există şi completări ale acestui nucleu, prezentate în [87]. Un mecanism complementar ar consta în injectarea în reţea a unor mesaje de tip ACK (Acknowledgment în limba engleză), care solicită confirmarea faptului că mesajul a ajuns la destinaţie. Acest procedeu ajută la eliberarea bufferelor de copii inutile ale mesajelor.

Un alt mecanism complementar este acela de a oferi o prioritate mai mare pachetelor care au avut un număr mic de hopuri până în prezent. Pentru a gestiona bufferul, acesta este împărţit din punct de vedere logic în două părţi, delimitate de un prag în funcţie de numărul de hopuri. Mesajele care sunt peste pragul predefinit sunt sortate în ordinea descrescătoare a probabilităţii de a fi livrate, în vederea eliminării lor în caz de umplere a bufferului.

RAPID (provine din acronimul Resource Allocation Protocol for International DTN Routing) şi este un algoritm de rutare ce abordează problema alocării resurselor în cazul rutării într-o reţea de tip DTN [5]. Acesta este un algoritm dezvoltat la Universitatea din Massachusetts Amherst şi a fost lansat ca parte a proiectului DieselNet. Autorii acestui algoritm au folosit o **funcţie utilitate**, căreia i se asociază o valoare notată U_i, pentru fiecare pachet i care se transmite în reţea. Astfel, U_i este contribuţia estimativă a pachetului i la îmbunătăţirea performanţelor algoritmului. Algoritmul RAPID va replica pachetele în ordinea descrescătoare a progresului valorii utilitate. Astfel vor fi replicate mai întâi pachetele cu cea mai mare îmbunătăţire a valorii funcţiei utilitate.

Funcţionalitatea algoritmului RAPID este împărţită în trei componente principale:

1. **Un algoritm de selecţie** care identifică ce pachete vor fi replicate atunci când se poate face transfer de date între noduri, în funcţie de valorile utilităţilor pachetelor.
2. **Un algoritm de deducţie** care se foloseşte pentru a estima utilitatea unui pachet, dându-se ca metrici de rutare minimizarea întârzierii medii, minimizarea termenelor de livrare ratate şi minimizarea întârzierii maxime.
3. **Un canal de control** care propagă metadatele necesare algoritmului de la punctul 2. Sunt schimbate astfel informaţii despre pachetele din reţea în momentul conexiunilor.

1.1.3 Gestiunea spaţiului de stocare al nodurilor în reţele cu toleranţă la întârzieri

Noţiuni introductive

În DTN, selecţia nodului următor din ruta mesajului joacă un rol foarte important în performanţa rutării. Pe lângă aceasta, politicile eficiente de gestionare a spaţiului de stocare

al mesajelor au de asemenea un impact major asupra performanţei rutării.

Gestiunea spaţiului de stocare al nodurilor este un factor deosebit de important în reţelele cu toleranţă la întârzieri datorită spaţiului foarte limitat pe care îl au la dispoziţie nodurile acestor reţele, dar şi pentru că mulţi algoritmi de rutare de tip DTN împrăştie în reţea mai multe copii ale mesajelor, favorizând umplerea rapidă a spaţiilor de stocare şi congestia reţelei.

Prin gestiunea bufferului se înţelege considerarea a două politici distincte:

- politica de planificare a transmiterii mai departe a mesajelor (întâlnită în literatura de specialitate sub denumirea de **Scheduling Policy**), care determină ordinea de livrare a mesajelor către nodurile de contact;
- politica de abandonare a mesajelor (întâlnită în literatura de specialitate sub denumirea de **Dropping Policy**), care determină ordinea eliminării mesajelor din buffer în cazul în care acesta este plin şi soseşte un nou mesaj.

Atunci când se produce o conexiune între două noduri şi unul dintre ele doreşte să transmită un mesaj celuilalt, dar spaţiul de stocare este insuficient, nodul în cauză trebuie, fie să renunţe la unul sau mai multe mesaje pentru a face loc noului mesaj sosit, fie să refuze noul mesaj sosit. Abordarea acestei probleme este rezolvată de către politicile de gestionare a bufferului.

Principalul scop al gestiunii bufferului este optimizarea următoarelor metrici de performanţă ale reţelei:

- **rata livrării mesajelor** - reprezintă raportul dintre numărul mesajelor ajunse la destinaţie şi numărul mesajelor generate de către nodurile sursă
- **întârzierea de transmitere a mesajelor** - este definită ca medie a intervalelor de timp dintre trimiterea şi recepţionarea mesajelor
- **timpul petrecut în buffer de către mesaje** - se defineşte ca medie a timpilor petrecuţi în buffer de către mesajele din sistem
- **supraîncărcarea** - este definită conform următoarei formule [84]:

$$overhead = \frac{mesaje\_transmise - mesaje\_livrate}{mesaje\_livrate}$$

Chiar dacă reţelele de tip DTN sunt concepute să tolereze întârzieri mari în transmiterea mesajelor, nu trebuie neglijată totuşi întârzierea medie de transmitere. Dacă întârzierea este prea mare, există riscul ca mesajele să îşi piardă relevanţa în momentul în care ajung la destinaţie.

Un alt factor pe care ar putea să îl ia în considerare politicile de gestiune a bufferului este minimizarea consumului de resurse, cum ar fi spaţiul de stocare sau lăţimea de bandă, prin minimizarea numărului de noduri intermediare pe care le parcurge un mesaj până la destinaţie.

Multe protocoale de rutare nu au un mecanism de gestionare a spaţiului de stocare. Motivul este acela că se presupune că lăţimea de bandă are capacitate infinită şi nu apar

întreruperi în timpul transmiterii unui mesaj. Aceasta este o presupunere nerealistă conform cu funcţionalitatea reală a reţelelor DTN.

Deoarece reţelele DTN conţin dispozitive portabile şi senzori, spaţiul de stocare şi energia limitată reprezintă constrângeri importante. Toate eforturile depuse de către nod pentru a selecta cât mai eficient hopul următor pentru mesaj vor fi risipite dacă bufferul nodului selectat este plin. Astfel de scenarii pot apărea frecvent în reţele dense, cu un trafic ridicat. În literatura de specialitate există o serie de abordări care propun diferite politici de gestiune a bufferelor în scopul optimizării performanţelor unei reţele de tip DTN.

Trecerea în revistă a literaturii de specialitate

În [33], Jain şi Chawla au făcut un rezumat al politicilor de gestiune a bufferului existente până în 2012, împreună cu deficienţele şi avantajele lor, discutând argumente pro şi contra acestora. Astfel, literatura de specialitate propusă este următoarea:

În [16], Davis şi coautorii au introdus în 2001 unele dintre cele mai simple şi mai cunoscute modele de eliminare a mesajelor din buffer. Scopul acestora a fost creşterea performanţei de livrare a mesajelor într-o reţea ad-hoc cu partiţionare sporită. Modelele expuse presupun: eliminarea aleatoare a unui mesaj, numită DRA (Drop Random), abandonarea primului mesaj sosit în buffer, numită DLR (Drop Least Recently Received), eliminarea mesajelor care au stat cel mai mult în reţea, numită DOA (Drop Oldest) şi eliminarea mesajului cu probabilitatea cea mai mică de livrare, numită DLE (Drop Least Encountered).

Strategia DLR este definită de faptul că un mesaj care a petrecut mai mult timp în buffer are probabilitate ridicată ca una dintre copiile sale să fi întâlnit destinaţia.

Strategia DOA se bazează pe faptul că mesajul care stă mai mult timp în reţea are o probabilitate mai mare să ajungă la destinaţie şi de aceea este abandonat primul.

Strategia DLE sortează pachetele de date în buffer în funcţie de capacităţile estimative a două noduri de a trimite pachetul către destinaţie. Pentru a calcula capacitatea estimativă, fiecare nod va păstra o listă cu adresele vecinilor altor noduri din reţea. La fiecare interval de timp, nodul A actualizează timpul de întâlnire cu un nod C, ţinând cont de nodul vecin B, utilizând următoarea regulă:

$$M_{t+1}(A,C) = \begin{cases} \lambda M_t(A,C), & \text{dacă nu există niciun nod vecin} \\ \lambda M_t(A,C) + 1, & \text{dacă } C = B \\ \lambda M_t(A,C) + \alpha M_t(B,C), & \text{pentru oricare } C \neq B \end{cases}$$

unde $M_t(A,C)$ este valoarea de întâlnire la timpul t , $\alpha = 0.1$ este un parametru care decide ce proporţie din valoarea de întâlnire cu B trebuie adaugată la nodul A şi $\lambda = 0.95$ este rata de micşorare a valorii de întâlnire. Valoarea $M_t(A,C)$ este iniţial 0 pentru toate perechile de noduri.

O altă serie de politici de gestiune a bufferului a fost propusa de Lindgren şi Phanse în [45], în anul 2006. Politicile de abandonare de mesaje prezentate sunt:

- FIFO (First In First Out) - este abordarea cea mai simplă, care presupune abandonarea mesajelor în funcţie de ordinea în care au sosit în buffer.

- MOFO (Most Forwarded First) - presupune abandonarea mesajelor care au fost transmise de cele mai multe ori de către nodul curent.
- MOPR (Most Favourably Forwarded) - se asociază fiecărui mesaj o metrică numită predictibilitate de transmitere, notată FP (Forwarding Predictability), care este iniţializată cu 0. De fiecare dată când un mesaj este trimis unui nod, valoarea FP a acelui mesaj se măreşte. Mesajele cu valoarea cea mai mare a metricii FP vor fi şterse primele.
- SHLI (Shortest Life First) - se bazează pe conceptul de timp de viaţă al mesajului şi abandonează mai întâi mesajele cu cel mai scurt timp de viaţă rămas.
- LEPR (Least Probable First) - propune abandonarea mesajelor care au cea mai mică probabilitate de livrare.

Autorii lucrării au demonstrat că MOFO este cel mai bun mecanism de abandonare de mesaje, în scopul creşterii ratei livrării mesajelor în reţea, iar SHLI este cel mai bun mecanism pentru scăderea întârzierii medii de livrare a mesajelor. Aceasta se datorează faptului că MOFO se asigură că fiecare mesaj a fost retransmis cel puţin o dată înainte de a fi abandonat, iar SHLI permite şi abandonarea mesajelor care nu au fost transmise niciodată.

Tot în [45] au fost prezentate şi o serie de strategii de transmitere a mesajelor: GRTR, GRTRSort şi GRTRMax. Strategia GRTR vine cu o uşoară modificare a strategiei de rutare din algoritmul PRoPHET, prin care, nodul A transmite nodului B un mesaj cu destinaţia D doar dacă $P(B, D) > P(A, D)$, unde $P(i, j)$ reprezintă probabilitatea de livrare dintre două noduri i şi j. Strategia GRTRSort selectează mesajele în ordinea descrescătoare a valorii $P(B, D) - P(A, D)$ şi le transmite mai departe doar dacă $P(B, D) > P(A, D)$. Politica GRTRMax sortează mesajele în ordine descrescătoare a valorii $P(B, D)$ şi după aceea le redirecţionează dacă $P(B, D) > P(A, D)$.

Simulările prezentate în [45] au demonstrat că GRTRSort în combinaţie cu MOFO au cel mai mare procent de livrare a mesajelor.

În [60] a fost propusă o schemă de gestiune a bufferului care consideră o topologie de reţea numită Politica de Rutare Epidemică Prioritizată (PREP). Această politică sortează mesajele în buffer atât pentru transmisie cât şi pentru abandonare. PREP este compus din două mecanisme:

- o metodă care estimează costul de rutare de la un nod dat la nodul destinaţie
- un mecanism de prioritizare pentru transmiterea şi ştergerea mesajelor

Pentru a şterge mesaje din buffer, sunt selectate mesajele care au parcurs un număr de hopuri mai mare decât un prag predefinit. Mesajele care se găsesc cel mai departe de nodul destinaţie (au costul drumului de la nodul curent la nodul destinaţie cel mai ridicat) sunt şterse primele.

Pentru a trimite mesaje, un nod va calcula costul de la nodul de contact până la nodul destinaţie al mesajelor. Dacă se obţine un cost mai mic decât al său, atunci mesajul este poziţionat în secţiunea din faţa bufferului. În caz contrar, acesta este poziţionat în secţiunea de la capătul bufferului. Mesajele din capătul bufferului vor fi sortate în funcţie de timpul de viaţă rămas.

În 2008, Krifa şi coautorii au propus în [38] o politică optimă atât pentru planificarea transmiterii mesajelor cât şi pentru eliminarea acestora, pe baza cunoştinţelor globale, numită GBD (Global Knowledge Based Drop).

Algoritmul este unul distribuit, care utilizează învăţarea statistică pentru a aproxima cunoştinţele globale necesare politicii de gestiune. Astfel, GBD calculează o funcţie utilitate pentru fiecare metrică de rutare asociată fiecărui mesaj, eliminând mesajele care au cea mai mică valoare pentru funcţiile utilitate asociate metricilor. Aceste funcţii iau în calcul numărul de noduri care au avut o copie a mesajului, dar şi numărul de noduri care deţin în prezent o copie a mesajului. Timpul de întâlnire dintre noduri se consideră a fi exponenţial distribuit.

Cu toate că această politică oferă un cadru optim de lucru, se bazează pe ipoteza că mişcarea nodurilor este cunoscută în reţea, lăţimea de bandă este nelimitată şi toate mesajele au aceeaşi dimensiune. Însă o astfel de situaţie este destul de rar întâlnită într-o reţea de tip DTN.

Datorită presupunerilor nerealiste din [38], Li şi Qian au propus în [41] un nou set de politici optime pentru gestiunea bufferului, adaptate mediilor de reţea de tip DTN, numit AOBMP (Adaptive Optimal Buffer Management Policies). Aceştia au presupus că atât lăţimea de bandă cât şi capabilităţile conexiunilor dintre noduri sunt limitate, iar dimensiunile mesajelor sunt diferite.

Protocolul AOBMP constă în trei paşi:

1. iniţializarea - constă în schimbul de metadate între două noduri şi actualizarea oportunităţii de transmitere a mesajului, precum şi a parametrilor referitori la modelul de mobilitate.

2. calculul funcţiei utilitate - calculează valoarea acestei funcţii, asociată mesajelor din nodul i.

3. eliminarea mesajelor - elimină mesajele care îndeplinesc condiţiile funcţiei utilitate asociate metricii.

Metricile considerate în [41] vor fi rata medie de livrare a mesajelor şi întârzierea medie de transmisie. Mai mult decât atât, AOBMP se adaptează la modelul de mobilitate al nodurilor cu ajutorul informaţiilor schimbate prin canalul de control.

În [42] a fost propusă strategia de rutare numită N-drop care controlează congestia în cadrul rutării de tip epidemic dintr-o reţea DTN. Această strategie previne congestia bufferului prin faptul că fiecare nod ţine evidenţa numărului total de transmisii ale mesajelor. Fiecare nod va calcula un prag N care este în conformitate cu spaţiul de stocare al bufferului său. Mesajul cu numărul de transmisii mai mare sau egal cu pragul N va fi abandonat. Dacă toate mesajele din buffer au numărul de transmisii mai mic decât N, atunci va fi abandonat ultimul mesaj sosit în buffer.

Soares şi coautorii au analizat în lucrarea [72] din 2010 performanţele diverselor politici de planificare a transmiterii mesajelor şi de abandonare a acestora, precum şi îmbinarea celor două abordări, cu scopul de a evidenţia efectul gestiunii spaţiului de stocare într-o reţea vehiculară, cu toleranţă la întârzieri (Vehicular Delay Tolerant Network – VDTN). Politicile abordate au fost: FIFO, modul aleator, timpul de viaţă rămas (Remaining Lifetime - RL) şi replicarea copiilor (Replicated Copies - RCs).

Simulările efectuate au arătat că cele mai bune performanţe s-au obţinut folosind o politică ce combină politicile de transmitere şi de abandonare a mesajelor, oferind o prioritate mai mare mesajelor cu un număr mai mic de copii. Această analiză a demonstrat că într-o reţea DTN, importanţa cea mai mare trebuie oferită mesajelor nou create, deoarece acestea încă nu au fost răspândite prea mult în reţea.

Ke, Nenghai şi Bin au propus în [35] o nouă metodă de gestiune a bufferului pentru reţelele de tip DTN. Metoda aceasta calculează utilitatea unui mesaj în funcţie de cât de mare este probabilitatea livrării lui la destinaţie. Fiecărui pachet din buffer îi este asociat un cost notat cu W_i, unde i reprezintă un nod. Costul mesajului este compus din două părţi: prima parte ia în considerare posibilitatea ca nodul să ajungă la destinaţie într-un singur pas, iar a doua parte ia în considerare faptul că mesajul trebuie transmis unor noduri intermediare.

Astfel, prima parte a costului mesajului ţine cont de timpul scurs până când cele două noduri intră în zona de contact, precum şi de timpii de viaţă rămaşi ai mesajelor. Cea de-a doua parte ţine cont de informaţiile cu privire la numărul de copii ale mesajului. Pentru această informaţie, fiecare nod înregistrează numărul de copii al fiecărui mesaj m_i pe care l-a creat. La fiecare conexiune, nodurile actualizează numărul de copii ale mesajelor. Atunci când bufferul unui nod este plin, este abandonat pachetul de date cu cel mai mic cost.

În [7], Bjurefors şi coautorii au propus o politică de abandonare a mesajelor care a fost evaluată în proiectul Haggle. Acesta reprezintă o reţea cu o arhitectură centrată pe date, bazată pe un model de publicare şi subscriere. Strategiile prezentate sunt: cel mai puţin interesant mesaj (LI), cel mai interesant mesaj (MI), replicări max-max, cel mai transmis mesaj (MF), cel mai puţin transmis mesaj (LF) şi abandonarea aleatorie.

- Strategia LI abandonează mesajul de care sunt interesaţi cei mai puţini vecini. O astfel de metodă creşte rata medie a livrării mesajelor într-o reţea centrată pe date, deoarece interesul global al reţelei continuă să crească. În această situaţie, mesajele de care nu sunt interesate nodurile vor ajunge să expire şi pot fi abandonate.
- Strategia MI acţionează opus faţă de strategia LI prin abandonarea mesajelor de care sunt interesaţi cei mai mulţi vecini. Logica din spate se bazează pe reducerea numărului de copii ale mesajelor pe care le caută cele mai multe noduri.
- Strategia max-max presupune eliminarea pachetelor care au ajuns la un prag maxim de copii. Acest lucru limitează copierea excesivă a mesajelor.
- Strategia MF abandonează mesajele care au cel mai mare număr de copii. Mesajul care a fost replicat de cele mai multe ori a ajuns deja în multe noduri şi are şanse mari de a ajunge la destinaţie, putând fi abandonat primul.
- Strategia LF acţionează opus faţă de strategia MF, prin abandonarea mesajului care are numărul cel mai mic de replici.

Simulările efectuate au demonstrat că strategia MF este cea mai bună în ceea ce priveşte optimizarea ratei livrării mesajelor, a întârzierii medii de transmitere şi a supraîncărcării bufferului.

Lucrările [3],[61],[62],[63] au oferit o viziune nouă gestiunii bufferului, deoarece nu au mai luat în considerare timpul de viaţă şi numărul de copii ale mesajului, ci a fost introdus un nou parametru: dimensiunea mesajului. În [61] se prezintă situaţia în care se elimină din buffer

cel mai mare mesaj. Este sacrificat mesajul cel mai mare pentru a se evita abandonarea mai multor mesaje de dimensiuni mici. În [3], metoda T-drop presupune eliminarea mesajului a cărui dimensiune se află sub un prag T al bufferului. În [62], metoda E-drop presupune eliminarea unui mesaj cu aceeaşi dimensiune cu mesajul nou sosit, avantajul fiind acela că se elimină un singur mesaj. În [63], politica Mean-drop calculează media dimensiunii mesajelor din bufferul congestionat şi elimină acele mesaje care au dimensiunea cel puţin egală cu dimensiunea medie.

Abordarea propusă în [22] este aceea de a lua în calcul priorităţile mesajului atât pentru planificarea transmiterii acestuia, cât şi pentru eliminarea lui din buffer în caz de supraîncărcare. Se oferă în acest sens trei clase de priorităţi mesajelor: **în masă** (bulk), **normal** şi **de expediat**. În momentul apariţiei unei oportunităţi de conectare între noduri, mesajele trimise în masă au prioritatea cea mai mică, iar cele de expediat au prioritatea cea mai mare. Atunci când soseşte în nod un mesaj nou, clasificatorul de mesaje îl stochează în coada potrivită, în funcţie de clasa căreia îi aparţine acel mesaj. În momentul unei conexiuni este apelat mai întâi planificatorul de mesaje, pentru a organiza pachetele pentru redirecţionare, şi apoi eliminatorul de mesaje, care va elimina mesajul cu prioritatea mai mare dintre acelea care au priorităţile în masă şi normal. Politica propusă are grijă ca nodul să nu elimine niciodată un mesaj care a fost creat de el însuşi.

În [46] a fost propus un sistem care lucrează cu două proprietăţi principale ale mesajelor: numărul de copii din reţea şi viteza de diseminare.

Mesajele cu numărul cel mai mic de replici vor primi priorităţile cele mai mari pentru a fi livrate. Pe de altă parte, atunci când trebuie să fie eliminat un mesaj, va fi selectat acela cu prioritatea cea mai mică, cu cel mai mare număr de replici. Dacă toate mesajele din buffer au acelaşi număr de replici, atunci se va folosi ca şi condiţie suplimentară viteza de diseminare a mesajului. Viteza de diseminare se calculează ca fiind raportul dintre numărul de hopuri pe care le-a avut mesajul până în prezent şi timpul petrecut în sistem de către acesta. Mesajele cu cea mai mare viteză de diseminare sunt abandonate primele în cazul în care bufferul devine supraîncărcat.

Abordarea propusă în [46] oferă performanţe bune pentru rata livrării mesajelor şi întârzierea de livrare, evidenţiind importanţa numărului de copii din reţea al mesajelor.

Shin şi Kim au propus în [68] o politică de gestiune a bufferului bazată pe următoarele proprietăţi ale mesajelor: numărul estimativ de replici, vârsta şi timpul de viaţă al mesajului.

Primul pas important în gestionarea bufferului este acela de a cunoaşte numărul de replici din reţea ale mesajului. Pentru a estima acest număr, au fost introduse două variabile: estimarea numărului total de replici (ETRs - Estimated Total Replicas) şi replicile furnizate de către nodul curent (MF - My Forward).

În [77], Tang şi coautorii au propus în 2012 o politică de gestiune a bufferelor, care utilizează conceptul de frecvenţă medie a contactelor dintre noduri (ACF – Average Contact Frequency). Frecvenţa medie a contactelor, notată f_{ij}, este definită ca fiind numărul de întâlniri dintre nodul i şi nodul j, în unitatea de timp.

Atunci când se întâlnesc două noduri, acestea schimbă între ele acele mesaje pe care nu le deţin, bazându-se pe următorul principiu, care are loc în două faze:

- în primul rând sunt transmise mesajele destinate nodului întâlnit, în ordinea descrescătoare a timpului de creare al mesajelor;

- alte mesaje care îndeplinesc condiţia de replicare sunt transmise nodului întâlnit în ordinea descrescătoare a valorii lui ACF.

De asemenea este utilizată şi o politică de abandonare a mesajelor bazată pe numărul de copii ale mesajului, care se calculează prin numărarea hopurilor parcurse până în prezent. Astfel, mesajul cu cele mai multe replici este abandonat atunci când bufferul este plin.

1.2 Simulatorul ONE

În secţiunea aceasta este prezentată o scurtă descriere a principalului instrument de simulare utilizat în lucrare. Este prezentat astfel simulatorul ONE [36] împreună cu unele dintre funcţionalităţile sale. Se va putea observa o imagine de ansamblu asupra protocoalelor de rutare implementate de către simulator şi asupra scenariilor de lucru oferite de acesta.

1.2.1 Informaţii generale

ONE este acronimul de la Opportunistic Network Environment şi este un produs cu sursă deschisă, dezvoltat în limbajul Java, care se poate descărca şi studia conform materialelor oferite de către Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD) [89],[90]. ONE este un motor de simulare bazat pe evenimente discrete. La fiecare etapă a simulării, motorul actualizează o serie de module care descriu principalele funcţionalităţi ale sale.

Acesta este un instrument creat pentru a permite definirea şi simularea activităţii unei reţele de tip DTN. Funcţionalităţile sale principale includ posibilitatea definirii unui mediu şi a unui scenariu de simulare. Mediul de simulare este inspirat din realitate şi este reprezentat sub formă de graf. Acesta oferă posibilitatea definirii mişcării nodurilor şi a contactelor dintre ele. Scenariile de simulare pot include configurarea mesajelor şi a modalităţii lor de transmitere. Acest instrument de simulare este unul uşor de utilizat şi de configurat, utilizatorul având la dispoziţie un fişier text cu specificaţiile generale ale unei simulări.

Simulatorul ONE oferă de asemenea o hartă pe care se pot deplasa nodurile. Harta implicită este cea a oraşului Helsinki, dar utilizatorul poate încărca în aplicaţie propria hartă.

Deplasarea nodurilor utilizează modele de mişcare sintetice, aşa cum sunt cele aleatoare, sau bazate pe traiectorii inspirate din realitate, cum ar fi modelele de deplasare ale tramvaielor. Conexiunile nodurilor se bazează pe locaţia acestora, pe raza lor de acţiune şi pe capacitatea de transfer asociată acestor conexiuni. Modulul de rutare are implementate mai multe protocoale de bază, utilizate des de către reţelele de tip DTN. Mesajele se generează aleatoriu, cu ajutorul unui generator de evenimente. După ce se efectuează simularea, ONE colectează datele obţinute şi metricile de performanţă ale sistemului şi le ilustrează în fişiere de raportare, create cu ajutorul modulului de rapoarte. Datele obţinute astfel pot fi preluate şi transformate cu alte instrumente destinate generării de grafice şi rapoarte. Interfaţa grafică a simulatorului oferă o vizualizare a stării în care se găseşte simularea, incluzând locaţia nodurilor, contactele active, mesajele transportate şi timpul simulării [36].

Simulatorul oferă posibilitatea de a adăuga un nou model de deplasare pentru noduri, o nouă modalitate de rutare a mesajelor sau un raport nou, conform cu cerinţele concrete ale sistemului care se doreşte a fi simulat.

1.2.2 Funcţionalitatea şi configurarea simulatorului ONE

În această secţiune este prezentată o scurtă descriere a simulatorului şi modalitatea de configurare a acestuia, pe baza parametrilor săi. Majoritatea informaţiilor sunt conform cu prezentările făcute de către autorii care au dezvoltat această platformă de simulare, în [36],[37].

Nodurile şi funcţiile acestora

Având în vedere că reţeaua de tip DTN se poate reprezenta sub formă de graf, acelaşi lucru îl realizează şi instrumentul de faţă. Nodurile sale sunt de mai multe tipuri: pietoni, maşini, autobuze sau tramvaie. Toate categoriile de noduri au în comun acţiunea de a se comporta ca un ruter care primeşte, stochează, transportă şi transmite mai departe mesaje. Modalitatea de deplasare a acestor categorii de noduri este însă diferită, acestea putând avea şi trasee distincte. Conform cu [36], printre parametrii configurabili ai nodurilor se află: raza de acţiune, capacitatea de stocare, rata de transfer a datelor, viteza de transmisie, modalitatea de mişcare, consumul de energie şi protocolul de rutare folosit. Pentru raza de acţiune, viteza de transmisie şi rata de transfer a datelor se foloseşte o interfaţă a undelor radio, realizată prin clasa *NetworkInterface* din pachetul *core* al aplicaţiei.

Funcţionalităţile mai complexe ale nodurilor, cum ar fi deplasarea sau rutarea, sunt implementate în module separate, de unde există acces la poziţia geografică a nodurilor.

În ceea ce priveşte consumul de energie, fiecare nod are alocată o cantitate de energie care este consumată de către evenimente precum scanarea reţelei sau transmisia de date. Cantitatea de energie este refăcuta prin încărcarea la anumite locaţii [36].

Modele de mobilitate

Simulatorul ONE oferă mai multe modele de mobilitate pentru noduri. Acestea sunt constrânse de poziţionarea lor pe hartă şi de determinarea unor drumuri între două puncte ale hărţii. Avem astfel trei modele de mobilitate:

- aleatoare - specificată în clasa *RandomWaypoint*
- bazată pe hartă - specificată în clasa *MapBasedMovement*
- bazată pe drumul minim - specificată în clasa *ShortestPathMapBasedMovement*

Toate aceste clase sunt derivate din clasa abstractă *MovementModel* şi se găsesc în pachetul *movement* al aplicaţiei. În clasa *MovementModel* sunt definite viteza de deplasare, timpul de aşteptare între două drumuri şi tipul de deplasare.

Modelul aleator al deplasării nodurilor este cel mai simplu şi se bazează pe generarea unei noi coordonate unde să fie plasat nodul.

Un model mai realist este acela al deplasării pe hartă. Harta acceptată de către simulator este un fişier în format WKT (Well Known Text), care se foloseşte adesea în diverse programe care utilizează hărţi. Acest model presupune aranjarea nodurilor aleator pe hartă la începutul simulării. După aceea, nodurile se deplasează pe segmente de drum consecutive, până ce ajung la capătul drumului sau într-o intersecţie. Atunci când un nod a ajuns într-o

intersecţie, selectează aleatoriu o direcţie de parcurs, fără a se întoarce înapoi pe direcţia de unde a venit.

O variantă îmbunătăţită a celei prezentate mai sus este cea bazată pe drumul cel mai scurt. Se alege o destinaţie aleatoare pentru fiecare nod din sistem. Nodul se deplasează pe segmente consecutive de drum până la acea destinaţie, folosind algoritmul de drum minim al lui Dijkstra [18]. După ce ajunge la destinaţie, nodul aşteaptă un timp şi selectează o altă destinaţie.

Dacă se doreşte definirea unui nou model de deplasare, se poate crea o nouă clasă care va extinde *MapBasedMovement*. În această clasă, în metoda *getInitialLocation* se vor stabili poziţiile de început ale nodurilor, iar în metoda *getPath* se va stabili drumul pe care vor merge nodurile.

Rutarea

Dacă modelele de mobilitate decid unde trebuie să se deplaseze nodurile, modelele de rutare decid unde trebuie transmise mesajele. Dezvoltatorii simulatorului au ales să implementeze atât algoritmi care transmit o singură copie a mesajului în reţea, cum sunt Direct Delivery şi First Contact, cât şi algoritmi care să împrăştie mai multe copii ale mesajelor în reţea, cum sunt Epidemic, Spray and Wait, PRoPHET şi MaxProp. Aceşti algoritmi au fost prezentaţi în subcapitolul precedent.

Fiecare algoritm de rutare se găseşte în propria sa clasă. Clasele aferente algoritmilor derivă din clasa abstractă *ActiveRouter*, care la rândul ei, derivă din clasa abstractă *MessageRouter*. Aceste clase au metode ce permit: verificarea conexiunii, transferul de mesaje, abandonarea de mesaje, sortarea cozii de mesaje, verificarea posibilităţii de transfer a mesajelor etc. Fiecare algoritm de rutare suprascrie aceste metode şi le oferă implementare în funcţie de cerinţele proprii.

Dacă se doreşte crearea propriului algoritm de rutare se va crea o nouă clasă care să extindă clasa *ActiveRouter* şi care să suprascrie convenabil metoda *messageTransferred*. Dacă se doreşte implementarea unei politici de eliminare a mesajelor din buffer se va suprascrie metoda *getNextMessageToRemove*. Clasele aferente modulului de rutare se găsesc în pachetul *routing*.

Raportarea rezultatelor

Simulatorul ONE face posibilă vizualizarea în două moduri a rezultatelor simulării:

- o modalitate este cea vizuală, care permite afişarea stării sistemului la fiecare pas al simulării
- o altă modalitate este cea bazată pe fişiere text, în care sunt depuse rezultatele finale ale simulării

După cum se poate observa în Fig. 1.2, afişarea în timp real a rezultatelor prezintă: poziţionarea nodurilor pe hartă, conexiunile dintre noduri, încărcarea bufferului nodurilor, raza de acţiune a nodurilor, declanşarea evenimentelor de generare, de ştergere, de transmitere a unui mesaj etc.

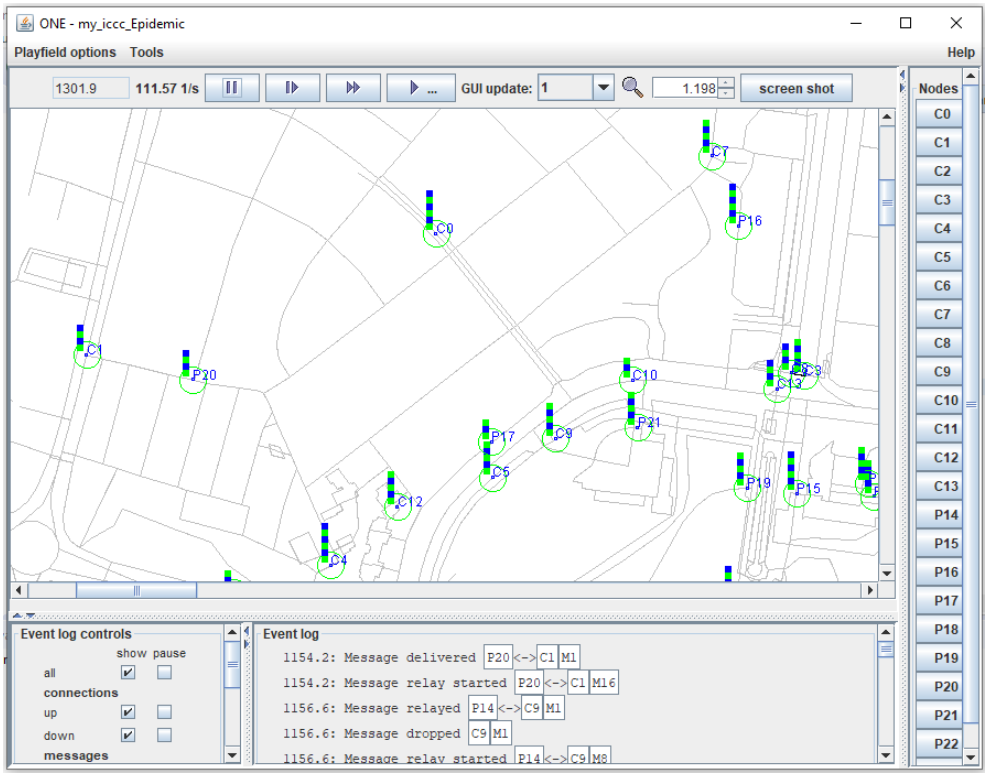


Figura 1.2: Interfaţa grafică a simulatorului ONE

Cea mai importantă modalitate de raportare este însă cea bazată pe fişiere. Clasele care generează aceste rapoarte se găsesc în pachetul *report* al aplicaţiei. Modulul care se ocupă de rapoarte permite salvarea de informaţii despre conexiuni, mesaje şi evenimente de mişcare. Atunci când se declanşează astfel de evenimente pe parcursul simulării, se apelează metodele aferente, care efectuează înregistrarea datelor în rapoarte. Fiecare raport este implementat în câte o clasă care derivă din clasa abstractă *Report*.

Dintre clasele specializate de rapoarte amintim câteva mai importante:

- *MessageStatsReport* - permite generarea unui fişier cu informaţii generale despre numărul de mesaje create, transmise, abandonate, livrate, despre rata livrării mesajelor, despre numărul mediu de hopuri parcurse de mesaje, despre întârzierea medie de livrare, despre timpul mediu petrecut în buffer de către mesaje etc.

- *DeliveredMessagesReport* - permite afişarea informaţiilor despre mesajele care au fost livrate destinaţiei: timpul de creare al mesajului, dimensiunea, numărul de hopuri parcurse, calea pe care a străbătut-o, timpul în care a ajuns la destinaţie etc.

- *MessageGraphvizReport* - afişează toate drumurile de la sursă la destinaţie, parcurse de mesaje.

- *AdjacencyGraphvizReport* - afişează toate conexiunile nodurilor.

Simulatorul include în modulul de raportare şi fişiere cu date compatibile cu Graphviz [91], aşa cum sunt *MessageGraphvizReport* şi *AdjacencyGraphvizReport*, amintite mai sus.

1.2.3 Parametrii care definesc un scenariu

Scenariul se defineşte prin modificarea caracteristicilor specificate în fişierul de configurare. Simulatorul are un fişier implicit de simulare, numit *default_settings.txt*, dar se pot crea propriile fişiere de simulare. Acestea vor conţine:

- informaţii despre scenariul simulării: un nume, timpul de simulare, intervalul de timp după care să se facă afişarea stării curente a simulării etc.
- informaţii cu privire la interfaţa de conectare: tipul interfeţei, viteza de transmisie, raza de acţiune etc.
- informaţii despre grupurile de noduri (vor exista configuraţii comune şi configuraţii specifice fiecărui grup): modelul de deplasare, algoritmul de rutare folosit, harta pe care se face deplasarea, numărul de noduri ale grupului, viteza de deplasare, timpul de viaţă al mesajelor generate de nodurile din grup etc.
- informaţii privitoare la mesaje: generatorul folosit, intervalul de timp după care se generează mesajele, dimensiunea etc.
- informaţii privitoare la modelul de deplasare: dimensiunea hărţii, intervalul de timp după care începe contorizarea rezultatelor simulării pentru raportare (se oferă un timp de încălzire, pentru a se crea legăturile dintre noduri), fişierele care conţin hărţile pe care se face simularea etc.
- informaţii despre rapoarte: numărul rapoartelor, locaţia unde să fie salvate fişierele cu rapoarte, denumirea rapoartelor care se salvează etc.
- informaţii despre interfaţa grafică: imaginea hărţii, decalajul în pixeli, scalarea, rotirea etc.

Capitolul 2

Algoritmi de simulare în teoria aşteptării

2.1 Noţiuni fundamentale în teoria aşteptării

2.1.1 Noţiuni preliminarii

În acest capitol este prezentată o bază teoretică pentru modelul de aşteptare propus pentru a defini reţeaua DTN. În domeniul teoriei aşteptării există o terminologie specifică. Vom vorbi astfel despre:

- clienţi - sunt cei care intră în sistem pentru a fi serviţi. Clienţii pot sosi pe rând sau în loturi.
- populaţie - este sursa de potenţiali clienţi, care poate fi una finită sau una infinită.
- server - este reprezentat de o staţie de servire, care va rezolva cererile clienţilor.
- coada de aşteptare - coada la care se vor aşeza clienţii în aşteptarea servirii.
- orbita - este o structură ce conţine clienţii care au găsit serverul ocupat, coada plină şi doresc să revină pentru a fi serviţi. Această structură apare în anumite sisteme de aşteptare mai complexe.
- timpii inter-sosiri - sunt timpii scurşi între două sosiri consecutive de clienţi.
- timpii de servire - sunt perioadele de timp alocate servirii clienţilor.
- disciplina aşteptării - este reprezentată de ordinea de servire a clienţilor din coadă.

Timpii inter-sosiri şi cei de servire pot avea diferite tipuri de repartiţie. Principalele tipuri de repartiţie sunt: Poisson (notată cu M de la proprietatea Markoviană a "lipsei memoriei"), k-Erlang (notată cu E_k), generală (notată cu G), deterministă (notată cu D) şi repartiţia exponenţial negativă. Repartiţia timpilor inter-sosiri este de parametru λ, cu $\lambda > 0$, iar repartiţia timpilor de servire este de parametru μ, cu $\mu > 0$. Aceşti parametri mai sunt cunoscuţi şi sub denumirea de *rata sosirilor*, respectiv *rata servirilor*. Aceste informaţii se pot regăsi în [28].

Cea mai cunoscută notaţie pentru diferitele tipuri de modele de aşteptare este cea a lui Kendall. Notaţia este de tipul: A/B/c/K/m/Z şi reprezintă:

- A - tipul de repartiţie a sosirilor
- B - tipul de repartiţie a timpului de servire
- c - numărul de staţii de servire
- K - capacitatea maximă a sistemului de aşteptare (dimensiunea maximă a cozii de aşteptare)
- m - mărimea / dimensiunea populaţiei
- Z - tipul de disciplină a cozii (politica de gestiune a cozii de aşteptare)

În general, într-un sistem de aşteptare, un client care intră în sistem şi găseşte serverul liber va fi servit imediat. Dacă noul client găseşte serverul ocupat, se va aşeza la coadă. În multe cazuri, cozile ataşate serverelor au dimensiuni limitate. Astfel că există posibilitatea ca noul client să găsească serverul ocupat şi coada plină. În acest context, el decide dacă părăseşte definitiv sistemul sau intră pe orbită (dacă sistemul are prevăzută o astfel de structură). Clienţii de pe orbită nu au acces la starea internă a sistemului, ci vor interoga periodic serverul pentru a afla dacă s-a eliberat. În cazul în care serverul s-a eliberat, clientul va fi servit, iar dacă nu este liber, clientul revine pe orbită. Această abordare este întâlnită în lucrările [23],[24],[25], ale căror rezultate aparţin parţial autoarei tezei.

Sistemele de şteptare sunt clasificate, în funcţie de numărul serverelor, în sisteme cu un singur server şi sisteme cu servere multiple. În funcţie de prezenţa sau absenţa orbitei, putem avea sisteme de aşteptare fără revenire (cele clasice, fără orbită) şi sisteme de aşteptare cu revenire.

Principalii factori de eficienţă ai unui sistem de aşteptare, conform definiţiilor şi notaţiilor din lucrarea [28], sunt:

- L - numărul mediu de clienţi din sistem
- L_q - numărul mediu de clienţi din coadă
- W - timpul mediu petrecut de clienţi în sistem
- W_q - timpul mediu petrecut de clienţi la coadă

În lucrarea [23] sunt consideraţi şi următorii factori de eficienţă:

- $MTOrb$ - timpul mediu petrecut de clienţi pe orbită
- $Clen$ - rata de inactivitate a serverului

Sistemele de aşteptare pot fi studiate analitic sau prin algoritmi de simulare. Folosind metode matematice de calcul, factorii de eficienţă ai unui sistem de aşteptare pot fi determinaţi doar pentru cazuri particulare. Prin metoda simulării, pot fi studiate sisteme mai complexe, unde formulele matematice nu se pot aplica.

Exemple de utilizare pentru sistemele de aşteptare

Exemplul 1 - Sistem de telefonie. O persoană care sună la un telefon ocupat, va reveni cu apelul cu o anumită probabilitate sau va renunţa la apel cu o probabilitate complementară. Dacă mai mulţi apelanţi ai aceluiaşi număr de telefon găsesc numărul ocupat, pot fi plasaţi pe orbită, pentru a reveni mai târziu.

Exemplul 2 - Un magazin cu o singură casă de marcat. Un client care găseşte o coadă mare la casa de marcat poate alege între a se întoarce după un anumit timp (plasare pe orbită) cu o anumită probabilitate sau va renunţa la cumpărături şi va părăsi magazinul (sistemul de aşteptare).

Exemplul 3 - Reţeaua Ethernet. În contextul metodei CSMA/CD, fiecare gazdă din reţeaua de comunicaţii poate fi considerată un client al sistemului de aşteptare, iar serverul este mediul de comunicare. Când o gazdă vrea să trimită un mesaj, poate să facă acest lucru doar atunci când mediul este liber. În caz contrar, va trebui să revină după un anumit timp. Mai multe gazde din reţea pot fi în această situaţie, plasându-se pe orbită.

Exemplul 4 - Reţeaua DTN. În contextul unei reţele cu toleranţă la întârzieri, fiecare nod poate fi considerat un sistem de aşteptare individual. Mesajele din bufferul nodului reprezintă clienţii, iar sistemul de gestiune al bufferului reprezintă serverul. În momentul conexiunii cu un nod vecin, transmiterea mesajelor către acesta semnifică servirea clienţilor, iar abandonarea mesajelor reprezintă părăsirea sistemului de aşteptare.

2.1.2 Modele de aşteptare. Abordare analitică

În literatura de specialitate se întâlnesc mai multe modele de aşteptare. Unele dintre acestea sunt potrivite sistemelor de aşteptare cu un singur server, iar altele sunt potrivite sistemelor de aşteptare cu mai multe servere. În lucrarea de faţă vor fi abordate doar modele de aşteptare pentru sisteme cu un singur server, mai exact modelul M/G/1, bazat pe cozi de priorităţi. Astfel, fiecare nod din reţeaua DTN va fi reprezentat printr-un sistem de aşteptare cu o singură staţie de servire. Se va studia aşadar comportamentul reţelei pe termen lung, pentru a putea determina starea de echilibru a acesteia.

Principalele modele de aşteptare cu un singur server sunt:

- M/M/1 - este un sistem de aşteptare în care, atât repartiţia timpilor inter-sosiri, cât şi cea a timpilor de servire este una Markoviană, reprezentată printr-un proces Poisson, cu timpii independenţi şi identic repartizaţi exponenţial.
- M/G/1 - este un sistem de aşteptare cu repartiţia sosirilor reprezentată printr-un proces Poisson, iar servirile au o repartiţie arbitrar generală.
- G/G/1 - este un sistem de aşteptare în care, atât repartiţia timpilor inter-sosiri, cât şi cea a timpilor de servire este una generală.
- D/G/1 - este un sistem de aşteptare cu repartiţie deterministă, cu intervale constante, pentru timpii inter-sosiri şi repartiţie arbitrar generală pentru timpii de servire.

Mai multe amănunte legate de aceste modele se pot găsi în [28].

În Fig. 2.1 se poate observa imaginea de bază a unui sistem de aşteptare cu un singur server, cu repartiţia timpului inter-sosiri de parametru λ, iar repartiţia timpului servirilor de parametru μ.

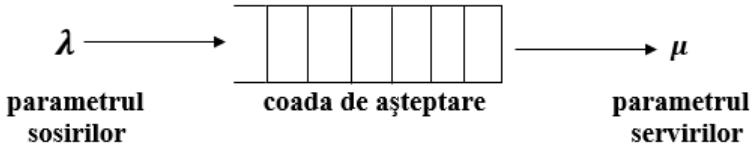


Figura 2.1: Sistem de aşteptare cu un singur server, rata sosirilor λ şi a servirilor μ

Modelul M/G/1

De cele mai multe ori, repartiţia de tip Poisson a sosirilor într-un sistem de aşteptare este una destul de realistă, dar timpii exponenţiali de servire nu sunt foarte frecvenţi în practică.

În [88] se consideră că spaţiul stărilor este $\{0, 1, 2, 3, ..., \}$, cu valori corespunzătoare numărului de clienţi din coadă (inclusiv clientul care este servit), atunci putem spune că:
- tranziţia din starea i în starea $i + 1$ reprezintă sosirea unui nou client în sistem.
- tranziţia din starea i în starea $i - 1$ reprezintă servirea unui client.

Timpii inter-sosiri vor avea distribuţie exponenţială de tip Poisson, de parametru λ, iar timpii de servire vor avea o distribuţie generală, de parametru μ. Lungimile intervalelor de timp sunt variabile aleatoare care se presupune că sunt independente statistic.

Mai multe detalii legate de acest model de aşteptare se găsesc în [2],[39],[82].

Pentru a studia analitic un model M/G/1, avem nevoie de câteva notaţii şi formule.

Se consideră următoarele notaţii:

- λ este rata medie a sosirilor
- μ este rata medie a servirilor
- $\frac{1}{\lambda}$ este media timpilor de sosire
- $\frac{1}{\mu}$ este timpul mediu de servire
- $\sigma = \frac{1}{\mu^2}$ este deviaţia standard a timpului de servire
- $\rho = \frac{\lambda}{\mu}$ este timpul mediu de funcţionare al serverului
- p_i este probabilitatea să se afle i clienţi în sistem în starea de echilibru
- p_0 este probabilitatea să nu se găsească niciun client în sistem, iar serverul să fie liber

Cu ajutorul acestor notaţii vom exprima conform cu [28] formulele de calcul aferente factorilor de eficienţă ai sistemului, şi anume:

$$p_0 = 1 - \frac{\lambda}{\mu} = 1 - \rho \tag{2.1}$$

Numărul mediu de clienţi care aşteaptă la coadă:

$$L_q = \frac{\lambda^2\sigma^2 + \rho^2}{2(1-\rho)} \tag{2.2}$$

Numărul mediu de clienţi care aşteapta în sistem:

$$L = L_q + \rho \tag{2.3}$$

Pentru a calcula timpii medii de aşteptare la coadă şi respectiv în sistem de către clienţi, avem nevoie de formulele lui Little, şi anume:

$$L_q = \lambda W_q \tag{2.4}$$

şi

$$L = \lambda W \tag{2.5}$$

Timpul mediu de aşteptare la coadă al clienţilor, în funcţie de numărul mediu de clienţi din coadă:

$$(2.4) \Rightarrow W_q = \frac{L_q}{\lambda} \tag{2.6}$$

Timpul mediu petrecut de către clienţi în sistem, în funcţie de timpul mediu petrecut la coadă:

$$(2.5) \Rightarrow W = \frac{L}{\lambda} = \frac{L_q + \rho}{\lambda} = \frac{L_q}{\lambda} + \frac{1}{\mu} = W_q + \frac{1}{\mu} \tag{2.7}$$

Modelul M/G/1 a fost extins în [6] folosind timpul rezidual de servire. Autorii au demonstrat că acest nou concept este util sistemelor care utilizează cozi de priorităţi sau sistemelor de rezervare, în care o parte din timpul de servire este ocupat cu trimiterea pachetelor de date, iar cealaltă parte cu trimiterea informaţiilor de control.

Timpul de servire rezidual pentru cel de-al i-lea client a fost notat cu R_i. Prin aceasta se înţelege faptul că dacă un client j este în curs de servire în momentul în care soseşte clientul i, R_i reprezintă timpul rămas până când se termină servirea clientului j. Dacă nu există clienţi în sistem în momentul în care soseşte clientul i, atunci valoarea lui R_i este 0.

Timpul mediu de servire se notează cu $\overline{X}$, iar timpul următoarei serviri cu $\overline{X^2}$. În aceste condiţii formula lui Pollaczek-Khinchin va deveni:

$$W_q = \frac{\lambda\overline{X^2}}{2(1-\rho)} \tag{2.8}$$

Pe baza formulei lui Pollaczek-Khinchin vom deduce timpul total de aşteptare în sistem ca fiind

$$W = \overline{X} + \frac{\lambda\overline{X^2}}{2(1-\rho)} \tag{2.9}$$

Formula (2.8) se poate rescrie bazându-se pe un concept de timp rezidual de servire.

Se vor folosi următoarele notaţii:

- W_i - timpul de aşteptare la coadă al celui de-al i-lea client.
- R_i - timpul de servire rezidual pentru cel de-al i-lea client.
- X_i - timpul de servire al celui de-al i-lea client.
- N_i - numărul de clienţi din coadă în momentul sosirii celui de-al i-lea client.

Astfel avem următoarea formulă pentru timpul de aşteptare în coadă al clientului i:

$$W_i = R_i + \sum_{j=i...N_i}^{i-1} X_j \tag{2.10}$$

Considerând limita când $i \to \infty$, obţinem următorul rezultat:

$$W_q = R + \frac{L_q}{\mu} \tag{2.11}$$

unde R este timpul rezidual mediu, definit ca

$$\lim_{i \to \infty} E(R_i)$$

unde E reprezintă funcţia de medie.

2.2 Sisteme de aşteptare cu revenire

Rezultatele prezentate în acest subcapitol aparţin parţial autoarei acestei teze, şi se regăsesc în lucrările [23],[24],[25].

Într-un sistem clasic de aşteptare, atunci când un client soseşte în sistem şi serverul este disponibil, acesta va fi servit imediat. În caz contrar, clientul va fi adăugat în coadă. Un sistem de aşteptare cu revenire extinde sistemul clasic prin noţiunea de orbită.

Sistemul de aşteptare cu revenire are două variante: una în care clienţii sosesc individual (a se vedea [23],[24]) şi alta în care clienţii sosesc în loturi de dimensiuni variabile (a se vedea [25]).

Ambele variante ale sistemului au o singură staţie de servire şi o singură coadă de aşteptare pentru clienţii care doresc să fie serviţi. Coada de aşteptare are dimensiune limitată, iar clienţii care o găsesc ocupată în momentul intrării lor în sistem, pot alege să aştepte pe orbită. Orbita este reprezentată sub forma unei cozi de clienţi cu dimensiune infinită, sortată în funcţie de timpul de revenire al acestora.

2.2.1 Descrierea sistemului

Sistemul de aşteptare propus este alcătuit din trei module principale:

- prelucrarea sosirilor în sistem
- prelucrarea servirilor
- prelucrarea revenirilor de pe orbită

Sosirile în sistemul de aşteptare

Atât în cazul sosirilor individuale, cât şi în cel al sosirilor în loturi, intervalul de timp dintre două sosiri consecutive este o variabilă aleatoare, cu distribuţie exponenţial negativă. Atunci când clienţii sosesc individual în sistem, aceştia au mai multe variante de a acţiona, în funcţie de starea în care se găseşte sistemul la acel moment, şi anume:

1. În cazul în care sistemul este în stare de "lenevire", fără clienţi în el, clientul nou sosit va fi servit imediat de către sistem.
2. În cazul în care staţia de servire este ocupată, dar există locuri libere în coada de aşteptare, clientul va intra în coadă. Sistemul va incrementa cu o unitate numărul total de clienţi şi va actualiza timpul de aşteptare în coadă.
3. În cazul în care staţia de servire este ocupată şi nu mai sunt locuri în coadă, clientul alege să părăsească sistemul sau să intre pe orbită. Pentru cazul în care intră pe orbită, acestuia i se va genera un număr maxim de reveniri şi o valoare a timpului pentru următoarea revenire de pe orbită. Sistemul va actualiza astfel timpul de aşteptare pe orbită.

În cazul în care clienţii sosesc în loturi, comportamentul lor şi cel al sistemului este similar ca în cazul variantei sosirii individuale. În acest caz însă, dacă dimensiunea lotului este mai mare decât numărul de locuri disponibile în coada de aşteptare, clienţii din lot care nu mai au loc la coadă pot opta pentru a părăsi sistemul sau pentru a intra pe orbită. Revenirile de pe orbită vor fi individuale şi în cazul sosirilor în loturi.

Probabilitatea ca un client să aleagă intrarea pe orbită sau părăsirea sistemului este o variabilă aleatoare de tip Bernoulli, în care sunt prezente două evenimente: rămânerea în sistem (pe orbită) şi părăsirea sistemului. Probabilitatea primului eveniment este notată cu p, şi în consecinţă, probabilitatea evenimentului al doilea este $1 - p$.

$$B : \begin{pmatrix} 1 & 2 \\ p & 1-p \end{pmatrix}$$

Prelucrarea servirilor în sistemul de aşteptare

În momentul începerii unui eveniment de servire, atât în cazul în care clienţii sosesc individual, cât şi în cel în care sosesc în loturi, sistemul actualizează dimensiunea cozii de aşteptare, decrementând-o cu o unitate. În momentul finalizării servirii, sistemul execută următoarele două acţiuni:

- actualizează numărul total de clienţi serviţi;
- actualizează timpul total de lucru al staţiei de servire;

Prelucrarea revenirii unui client de pe orbită

În momentul revenirii unui client de pe orbită, în sistem au loc următoarele acţiuni:

- Dacă staţia de servire este liberă, atunci clientul va fi preluat pentru a fi servit, iar sistemul va actualiza numărul de clienţi de pe orbită şi timpul de aşteptare pe orbită.
- Dacă staţia de servire este ocupată, dar există locuri în coada de aşteptare, clientul va intra în coadă, iar sistemul va actualiza numărul de clienţi, timpul de aşteptare în coadă şi pe orbita.
- Dacă staţia este ocupată, coada este plină, iar numărul de reveniri de pe orbită ale clientului nu a ajuns la zero, acesta se va întoarce pe orbită, iar sistemul va decrementa cu o unitate numărul rămas de reveniri ale sale şi îi va genera un nou timp pentru următoarea revenire de pe orbită.
- Dacă staţia este ocupată, coada este plină, iar numărul de reveniri de pe orbită ale clientului a ajuns la zero, acesta părăseşte sistemul.

Clienţii de pe orbită nu cunosc starea sistemului, aşa că sunt nevoiţi să interogheze periodic sistemul pentru a afla dacă pot fi serviţi sau dacă au loc la coadă. Aceste operaţii se execută atât în cazul în care clienţii sosesc în loturi, cât şi în cazul în care sosesc individual, deoarece servirile şi revenirile de pe orbită se desfăşoară în mod individual.

2.2.2 Detalii de simulare a sistemului

Sistemul va rula până când numărul de sosiri va atinge un prag predefinit. Rata sosirilor în sistem trebuie să fie mai mică decât rata servirilor, iar numărul total al sosirilor trebuie să fie unul foarte mare, pentru ca rezultatele sistemului să fie concludente. În acest caz, procentul clienţilor serviţi va fi unul ridicat.

Fiecare ciclu al simulării sistemului va consta în execuţia unui eveniment de sosire în sistem, de servire sau de revenire de pe orbită.

La finalul simulării, au fost analizaţi următorii factori de eficienţă ai sistemului:

- timpul mediu petrecut în coadă de către un client (notat MTwq), calculat ca raport între timpul total de aşteptare în coadă şi numărul total de clienţi serviţi ($MTwq = Twq/Tnrserv$);
- timpul mediu petrecut pe orbită de către un client (notat MTOrb), calculat ca raport între timpul total de aşteptare pe orbită şi numărul total de clienţi care au aşteptat pe orbită ($MTOrb = Torb/Tnrorb$);
- timpul mediu de servire al unui client (notat Mts), calculat ca raport între timpul total de servire al clienţilor şi numărul total de clienţi serviţi ($Mts = Tserv/Tnrserv$);
- gradul de inactivitate al sistemului (notat Clen), calculat ca raport între timpul total de inactivitate al sistemului şi timpul când s-a încheiat ultimul eveniment din sistem ($Clen = Tlen/Ltime$);
- dimensiunea medie a cozii de aşteptare (notată Mqueue), calculată ca raport între timpul total de aşteptare al clienţilor în coadă şi timpul când s-a încheiat ultimul eveniment din sistem ($Mqueue = Twq/Ltime$).

2.2.3 Prezentarea algoritmilor simulării

În primul rând, algoritmul de simulare al sistemului de aşteptare cu revenire, în care clienţii sosesc în loturi, va genera parametrii pentru variabilele aleatoare asociate: intervalului dintre două sosiri consecutive (valoarea parametrului λ), timpului de servire (valoarea parametrului μ), timpului petrecut pe orbită şi probabilităţii de a intra pe orbită. Această generare se va realiza prin apelul procedurii **Read()**. Generarea variabilelor care utilizează parametrii mai sus numiţi, dar şi a dimensiunii lotului de clienţi, se va face prin apelul procedurii **Gen(IntArriv, Stime, Bernoulli, DimBatch)**. Procedurile **InsertClientInQueue** şi **InsertClientInOrbit** efectuează inserarea clienţilor în coada de aşteptare, respectiv pe orbită.

Algoritmul 1 Algoritmul de simulare a unui sistem de aşteptare cu revenire, ai cărui clienţi sosesc în loturi [25]

```
1: procedure RetrQueuingSystOneStat(Tnra)
2:     Read();
3:     Nra ← 0;                                  ▷ numărul de sosiri în sistem
4:     nq ← 0;                                   ▷ numărul de elemente din coadă
5:     Ltime ← 0;                                ▷ timpul ultimului eveniment din sistem
6:     Ctime ← ∞;                                ▷ timpul curent al simulării
7:     Tnrserv ← 0;                              ▷ numărul de clienţi serviţi
8:     Tnrorb ← 0;                               ▷ numărul de clienţi de pe orbită
9:     Torb ← 0;                                 ▷ timpul total petrecut pe orbită
10:    Tserv ← 0;                                ▷ timpul total de servire al clienţilor
11:    Tlen ← 0;                                 ▷ timpul total de inactivitate al serverului
12:    Twq ← 0;                                  ▷ timpul total petrecut în coadă
13:    no ← 0;                                   ▷ numărul de clienţi de pe orbită
14:    To(1).Time_Rev ← ∞;                       ▷ timpul de revenire al primului client
15:    Gen(IntArriv, Stime, Bernoulli, DimBatch);
16:    Atime ← IntArriv;
17:    while Nra ≤ Tnra do
18:        if min{Atime, Ctime, To(1).Time_Rev} = Atime then
19:            Update_Arriv();
20:        else
21:            if min{Atime, Ctime, To(1).Time_Rev} = Ctime then
22:                Update_Fin_Serv();
23:            else
24:                Update_Retrial();
25:            end if
26:        end if
27:    end while
28:    MTwq ← Twq/Tnrserv;
29:    MTOrb ← Torb/Tnrorb;
30:    Mts ← Tserv/Tnrserv;
31:    Clen ← Tlen/Ltime;
32:    Mqueue ← Twq/Ltime;
33:    Write(MTwq, MTOrb, Mts, Clen, Mqueue);
34: end procedure
```

Algoritmul 2 Evenimentul de sosire a unui lot de clienţi în sistem

1: **procedure** UPDATE_ARRIV
2: **if** $Ctime = \infty$ **then** ▷ dacă staţia este liberă
3: $Tlen \leftarrow Tlen + Atime - Ltime$;
4: $Tsc \leftarrow Stime$;
5: $Ctime \leftarrow Atime + Stime$;
6: **if** $DimBatch - 1 \leq d$ **then**
7: $nc \leftarrow nc + DimBatch - 1$;
8: $InsertClientInQueue(1, DimBatch - 1)$; ▷ DimBatch-1 clienţi intră în coadă
9: $Tw \leftarrow Tw + nc \cdot (Atime - Ltime)$;
10: **else** ▷ primul client este servit, d clienţi intră în coadă, DimBatch-1-d clienţi intră pe orbită
11: $nc \leftarrow nc + d$;
12: $InsertClientInQueue(1, d)$;
13: $Tw \leftarrow Tw + nc \cdot (Atime - Ltime)$;
14: $no \leftarrow no + DimBatch - 1 - d$;
15: $InsertClientInOrbit(d + 1, DimBatch - 1)$;
16: $Torb \leftarrow Torb + no \cdot (Atime - Ltime)$;
17: **end if**
18: **else** ▷ dacă staţia este ocupată
19: **if** $DimBatch \leq d$ **then** ▷ dacă încap toţi cei DimBatch clienţi în coadă
20: $nc \leftarrow nc + DimBatch$;
21: $Tw \leftarrow Tw + nc \cdot (Atime - Ltime)$;
22: $InsertClientInQueue(1, DimBatch)$;
23: **end if**
24: ▷ dacă nu încape tot lotul în coadă, DimBatch-d clienţi se trimit pe orbită
25: **if** $DimBatch > d \& d > 0$ **then**
26: $nc \leftarrow nc + d$;
27: $InsertClientInQueue(1, d)$;
28: $Tw \leftarrow Tw + nc \cdot (Atime - Ltime)$;
29: $no \leftarrow no + DimBatch - d$;
30: $InsertClientInOrbit(d + 1, DimBatch)$;
31: $Torb \leftarrow Torb + no \cdot (Atime - Ltime)$;
32: **end if**
33: **if** $DimBatch > d \& d = 0$ **then** ▷ toţi cei DimBatch clienţi se trimit pe orbită
34: $no \leftarrow no + DimBatch$;
35: $InsertClientInOrbit(1, DimBatch)$;
36: $Torb \leftarrow Torb + no \cdot (Atime - Ltime)$;
37: **end if**
38: **end if**
39: $Ltime \leftarrow Atime$;
40: $Gen(IntArriv, Stime)$;
41: $Atime \leftarrow IntArriv$;
42: $Nra \leftarrow Nra + 1$;
43: **end procedure**

Algoritmul 3 Evenimentul de servire a unui client

```
1: procedure UPDATE_FIN_SERV
2:     Tserv ← Tserv + Tsc;
3:     Tnrserv + +;
4:     Torb ← Torb + no · (Ctime − Ltime);
5:     Tw ← Tw + nc · (Ctime − Ltime);
6:     Ltime ← Ctime;
7:     if nc > 1 then
8:         Tsc ←Ts(1);
9:         Ctime ← Ctime + Ts(1);
10:        RemoveClientFromSystem();
11:        nc − −;
12:    else
13:        Ctime ← ∞;
14:    end if
15: end procedure
```

Algoritmul 4 Evenimentul de revenire a unui client de pe orbită

```
1: procedure UPDATE_RETRIAL
2:     Torb ← Torb + no · (To(1).Time_Rev − Ltime);
3:     if Ctime = ∞ then
4:         Tlen ← Tlen + To(1).Time_Rev − Ltime;
5:         Ltime ← To(1).Time_Rev;
6:         Ctime ← To(1).Time_Rev + To(1).Time_Serv;
7:         Tsc ← To(1).Time_Serv;
8:         RemoveFromOrbit(1);
9:         no − −;
10:        Tnrorb + +;
11:    else
12:        if d > 0 then
13:            nc + +;
14:            Twq ← Twq + nc · (To(1).Time_Rev − Ltime);
15:        else
16:            To(1).Rev_Ram − −;
17:            if To(1).Rev_Ram > 0 then
18:                To(1).Time_Rev ← Gen(IntRet);
19:                InsertClientInOrbit();
20:            else
21:                RemoveClientFromSystem();
22:                no − −;
23:            end if
24:        end if
25:    end if
26: end procedure
```

Teorema 1. *Acţiunea care prelucrează sosirile tuturor clienţilor în sistem, are complexitatea* $O(mb \cdot Tnra \cdot (max\{no \cdot DimBatch, nqmax\}))$.

Teorema 2. *Acţiunea care prelucrează încheierea servirii tuturor clienţilor din sistem, are complexitatea* $O(mb \cdot Tnra \cdot nqmax)$.

Teorema 3. *Acţiunea care prelucrează revenirea tuturor clienţilor de pe orbită, are complexitatea* $O(mb \cdot Tnra \cdot no \cdot NoMaxRet)$.

Teorema 4. *Complexitatea algoritmului de simulare RetrQueuingSystOneStat este* $O(mb \cdot Tnra \cdot (max\{no \cdot DimBatch, nqmax\} + nqmax + no \cdot NoMaxRet))$.

2.2.4 Validitatea algoritmului de simulare

Pentru testarea algoritmului prezentat au fost simulate 30000 de sosiri ($Nra = 30000$). Intervalul de timp între două sosiri în sistem, timpul de servire şi intervalul de timp între două reveniri ale unui client sunt variabile cu distribuţie exponenţial negativă, de parametri: λ, μ şi ν.

Cele patru scenarii prezentate în [25] sunt:

Cazul 1. Vom considera că $\lambda = 1$, $\mu = 2$, $nqmax = \infty$ (dimensiunea cozii este infinită), $DimBatch = 1$ şi B: $\begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ (probabilitatea ca un client să intre pe orbită este 0). Acest caz este similar cu un sistem de aşteptare fără revenire, în care coada de aşteptare este infinită.

În tabelul de mai jos sunt prezentate în mod comparativ rezultatele factorilor de eficienţă ai sistemului, rezultate obţinute prin simularea sistemului propus şi a unui sistem de aşteptare clasic, fără revenire.

Factor Eficienţă	**Sistem clasic**	**Sistem propus**
Timpul mediu de aşteptare în coadă	1.00185	0.99388
Dimensiunea medie a cozii	0.50223	0.49857
Timpul mediu de servire	0.99547	0.99103
Gradul de utilizare al sistemului	0.49629	0.49778

Tabela 2.1: Analiza sistemelor de aşteptare cu şi fără revenire [25]

Se poate observa din Tab. 2.1 că valorile obţinute sunt aproximativ egale.

Cazul 2. Vom considera că $\lambda = 1$, $\mu = 2$, $nqmax = 0$ (dimensiunea cozii este 0), $DimBatch = 1$ iar B: $\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \end{pmatrix}$ (probabilitatea ca un client să intre pe orbită este egală cu probabilitatea să părăsească sistemul). Acest caz este similar cu un sistem de aşteptare cu revenire, în care clienţii sosesc individual, iar rezultatele simulării sunt asemenea celor din [23].

Cazul 3. Vom considera că $\lambda = 1$, $\mu = 2$, $\nu = 1$, $nqmax = 0$ (dimensiunea cozii este 0), $DimBatch > 1$, B: $\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \end{pmatrix}$ (probabilitatea ca un client să intre pe orbită este egală cu probabilitatea să părăsească sistemul). Acest caz a fost studiat din punct de vedere analitic

în [2],[20],[82]. Rezultatele obţinute analitic şi cele obţinute prin simulare sunt prezentate în Tab. 2.2, şi se poate observa că valorile sunt aproximativ egale.

Factor Eficienţă	**Rezultate obţinute analitic**	**Rezultate obţinute prin simulare**
Timpul mediu de aşteptare în coadă	1.00185	0.99388
Dimensiunea medie a cozii	0.49725	0.49857
Timpul mediu de servire	1.00742	0.99103
Gradul de utilizare al sistemului	0.50149	0.49778
Timpul mediu petrecut pe orbită	1.51001	1.49751

Tabela 2.2: Prezentarea rezultatelor analitice şi a celor obţinute prin simulare [25]

Cazul 4. Vom considera că $\lambda = 1$, $\mu = 2$, $\nu = 1$, $nqmax > 0$, $DimBatch > 1$, B: $\begin{pmatrix} 1 & 2 \\ 0.5 & 0.5 \end{pmatrix}$. Acest caz nu a fost studiat din punct de vedere analitic, ci sunt prezentate în paralel situaţia în care dimensiunea maximă a cozii este de 100 şi cea în care dimensiunea maximă a cozii este de 200. Rezultatele obţinute sunt prezentate în Tab. 2.3. Se poate observa că pentru valori mai mari ale cozii, scade timpul mediu de aşteptare pe orbită.

Factor Eficienţă	**Rezultatele pentru nqmax = 100**	**Rezultatele pentru nqmax = 200**
Timpul mediu de aşteptare în coadă	1.10185	0.99388
Dimensiunea medie a cozii	0.49725	0.49857
Timpul mediu de servire	1.00742	0.99103
Gradul de utilizare al sistemului	0.50149	0.49778
Timpul mediu petrecut pe orbită	1.23158	0.74609

Tabela 2.3: Analiza rezultatelor pentru variaţia dimensiunii cozii [25]

2.2.5 Integrarea sistemului de aşteptare într-o reţea cu toleranţă la întârzieri

Capitolul de faţă prezintă o serie de noţiuni fundamentale pentru sistemele de aşteptare, care stau la baza modelelor propuse în următorul capitol. De asemenea, sunt prezentate câteva comparaţii ale rezultatelor analitice şi ale celor obţinute prin simularea sistemului de aşteptare, care se poate integra într-o reţea de tip DTN, în care fiecare nod este un sistem de aşteptare individual. Atunci când avem o reţea cu toleranţă la întârzieri, cu mai multe surse de transmitere de date şi mai multe posibile destinaţii ale datelor, fiecare nod al reţelei poate fi considerat un centru independent de servicii, care are rolul de staţie de servire. Serviciile oferite de noduri se îndreaptă către mesajele pe care le au în buffer, şi constau în redirecţionarea corectă a mesajelor către următorul hop din drumul spre destinaţie. Sistemul de aşteptare utilizat pentru descrierea nodurilor este unul cu o singură staţie de servire. După terminarea servirii, mesajul se va deplasa către un alt centru de servire (un alt nod), pe baza

unui algoritm de rutare, intrând iarăşi într-un proces similar. Procesul se va finaliza în momentul în care mesajul ajunge la destinaţie, părăsind sistemul de aşteptare.

În capitolul următor, cel în care este descris algoritmul de rutare propus în această teză de doctorat, se poate observa prezenţa cozilor de priorităţi, similare celor din modelul de aşteptare M/G/1. Prin optimizarea modelului matematic se poate ajunge şi la o optimizare a factorilor de eficienţă ai reţelei DTN modelate.

Modelul de aşteptare prezentat (care este unul de tip M/G/1) se îmbină cu natura dinamică a reţelei cu toleranţă la întârzieri şi cu modalitatea de a transmite mesaje în această reţea, prin protocoale de rutare adecvate, cu număr multiplu de copii ale mesajelor. Tocmai această modalitate de răspândire a mesajelor, cu număr mare de copii, conduce la necesitatea prezenţei unei strategii de gestiune a bufferului nodurilor reţelei (cozilor de aşteptare ale modelului reţelei). Strategia de gestiune a bufferului pentru metoda de rutare propusă are ca scop major maximizarea ratei de livrare a mesajelor, utilizând un model de aşteptare care oferă o rată ridicată de servire a clienţilor, aşa cum se poate observa în simularea sistemului prezentat anterior.

Capitolul 3

Probleme în reţele cu toleranţă la întârzieri

În capitolul de faţă sunt propuse spre rezolvare două probleme importante într-o reţea de tip DTN:

1. Transformarea unei reţele DTN dinamice, cu buffer limitat, într-o reţea statică, în care se identifică un flux maxim. Această cercetare aparţine autoarei şi se regăseşte în lucrarea [54].

2. Realizarea unui nou protocol de rutare, cu scopul de a îmbunătăţi rata de livrare a mesajelor şi de a reduce supraîncărcarea reţelei. Astfel am propus protocolul de rutare numit **MaxDelivery**. Rezultatele acestea aparţin integral autoarei şi se regăsesc în [55] şi [56].

În multe cazuri, mai ales în cadrul reţelelor cu toleranţă la întârzieri, este mai important să se livreze un număr cât mai mare de mesaje decât să se livreze mesajele într-un timp cât mai scurt, cu toate că nici aceast criteriu de performanţă nu este unul care trebuie neglijat.

3.1 Fluxul maxim în reţele cu toleranţă la întârzieri. Abordarea statică

Într-o reţea de tip DTN nu există garanţia existenţei unui drum permanent între două noduri, fapt care se datorează topologiei reţelei şi caracteristicilor variabile în timp ale arcelor. În acest caz, pentru a face faţă evoluţiei sistemului în timp, este necesară modelarea reţelei DTN sub formă de reţea dinamică. Atunci când timpul este considerat o variabilă discretă, această problemă se poate rezolva prin construcţia unei reţele statice extinse, echivalentă celei dinamice [1],[26]. Această secţiune propune o abordare statică pentru studiul fluxului maxim într-o reţea de tip DTN, în care nodurile au capacitate limitată de stocare.

Punctul de start al acestei transformări îl reprezintă reţeaua variabilă în timp din lucrarea [85], publicată de Zhang, numită TAG (Time Aggregated Graph în limba engleză). În urma transformării acestui graf, reţeaua statică obţinută va avea mai multe noduri şi arce decât cea originară, permiţând arcelor să fie modelate ca serii de timp [43],[85].

Deoarece modelul dinamic nu necesită replicarea întregului graf pentru fiecare interval de timp, algoritmii pentru această reţea sunt mai eficienţi decât cei pentru reţeaua statică extinsă. Cu toate acestea, în multe cazuri este de preferat şi mai uşor de abordat o reţea statică decât una dinamică.

3.1.1 Fluxul maxim în reţele statice şi reţele dinamice

Noţiunile şi rezultatele din acest subparagraf sunt preluate din [1],[12],[14],[26],[69].

Se consideră un graf $G = (N, A)$ conex, antisimetric şi fără cicluri. Mulţimea nodurilor grafului este $N = \{1, ..., i, ..., j, ..., n\}$, iar mulţimea arcelor este $A = \{a_1, ..., a_k, ..., a_m\}$, cu $a_k = (i, j), i, j \in N$. Fie $S = (N, A, u)$ o reţea statică, cu funcţia capacitate definită ca $u : A \to \mathbb{N}$, $\mathbb{N}$ reprezentând mulţimea numerelor naturale. Vom considera nodul sursă 1 şi nodul stoc n.

Pentru o pereche dată de submulţimi X şi Y ale mulţimii N de noduri ale reţelei S, vom avea notaţia:

$$(X, Y) = \{(i, j)|(i, j) \in A, i \in X, j \in Y\}$$

Pentru o funcţie dată f, definită pe mulţimea de arce A, facem următoarea notaţie:

$$f(X, Y) = \sum_{(X,Y)} f(i, j)$$

Dacă $X = \{i\}$ sau $Y = \{j\}$, atunci se pot folosi notaţiile (i, Y) şi (X, j).

Fluxul este definit ca o funcţie $f : A \to \mathbb{N}$, care satisface următoarele condiţii:

$$f(i, N) - f(N, i) = \begin{cases} v & \text{dacă } i = 1 \\ 0 & \text{dacă } i \neq 1, n \\ -v & \text{dacă } i = n \end{cases} \tag{3.1a}$$

$$0 \leq f(i, j) \leq u(i, j), (i, j) \in A \tag{3.1b}$$

pentru $\forall v \geq 0$, unde v este valoarea fluxului f.

Problema fluxului maxim constă în determinarea unui flux f pentru care valoarea v este maximă.

Mulţi algoritmi utilizaţi pentru probleme de flux maxim se bazează pe conceptul de drum de mărire a fluxului.

Se consideră un flux f şi se defineşte reţeaua reziduală $R = (N, \tilde{A}, r)$, unde N este mulţimea de noduri şi $\tilde{A}, r$ sunt definite prin formulele (3.2),(3.3).

$$\tilde{A} = \tilde{A}^+ \cup \tilde{A}^- \tag{3.2a}$$

$$\tilde{A}^+ = \{(i, j)|(i, j) \in A \text{ şi } f(i, j) < u(i, j)\} \tag{3.2b}$$

$$\tilde{A}^- = \{(j, i)|(i, j) \in A \text{ şi } f(i, j) > 0\} \tag{3.2c}$$

Capacitatea reziduală $r : \tilde{A} \to \mathbb{N}$ în raport cu fluxul f este următoarea:

$$r(i, j) = \begin{cases} u(i, j) - f(i, j) & \text{pentru } (i, j) \in \tilde{A}^+ \\ f(j, i) & \text{pentru } (j, i) \in \tilde{A}^- \end{cases} \tag{3.3}$$

Un drum de mărire a fluxului relativ la f este un drum P în reţeaua R, de la nodul 1 la nodul n.

Teorema 5. *[1] Un flux f^* este flux maxim dacă şi numai dacă reţeaua reziduală R nu conţine drumuri de mărire a fluxului.*

Se consideră funcţia distanţă $d : N \to \mathbb{N}$ în reţeaua reziduală R. Vom spune că funcţia distanţă este validă în cazul în care satisface următoarele două condiţii:

$$d(n) = 0 \tag{3.4a}$$

$$d(i) \leq d(j) + 1, i \in N, (i, j) \in \tilde{A} \tag{3.4b}$$

Etichetele distanţă au următoarele două proprietăţi [1]:

Proprietatea 1. Dacă etichetele distanţă sunt valide, atunci eticheta distanţă $d(i)$ este o limită inferioară a lungimii drumului minim de la nodul i la nodul n în reţeaua reziduală R.

Proprietatea 2. Dacă $d(1) \geq n$, atunci reţeaua reziduală R nu conţine niciun drum de la nodul sursă la nodul stoc.

Vom spune că etichetele distanţă sunt exacte pentru fiecare nod i dacă $d(i)$ este egal cu lungimea drumului minim de la nodul i la nodul stoc n, în reţeaua reziduală R. De asemenea vom considera că un arc (i, j) din reţeaua reziduală este admisibil dacă satisface condiţia $d(i) = d(j) + 1$. Celelalte arce, care nu satisfac această condiţie, sunt inadmisibile. Un drum de la nodul 1 la nodul n, care este alcătuit în întregime din arce admisibile, este considerat un drum admisibil. Un drum admisibil are următoarea proprietate [1]:

Proprietatea 3. Un drum admisibil de la nodul sursă 1 la nodul stoc n este un drum minim de mărire a fluxului.

Se poate determina un drum minim de mărire a fluxului de la nodul sursă 1 la nodul stoc n, în reţeaua reziduală R, prin executarea unei parcurgeri inverse în lăţime în reţeaua R, începând de la nodul n, cu $d(n) = 0$.

Algoritmul care determină drumul minim de mărire a fluxului este prezentat în Alg. 8. Acesta apelează trei proceduri, prezentate în Alg. 5, Alg. 6 şi Alg. 7 [1].

Algoritmul 5 Procedura de înaintare.

```
1: procedure ÎNAINTARE(i)
2:     fie (i, j) un arc admisibil în Ã
3:     pred(j) ← i
4:     i ← j
5: end procedure
```

Algoritmul 6 Procedura de retragere.

```
1: procedure RETRAGERE(i)
2:     d(i) ← min{d(j) + 1|(i, j) ∈ Ã}
3:     if i ≠ s then
4:         i ← pred(i)
5:     end if
6: end procedure
```

Algoritmul 7 Procedura de mărire.

1: **procedure** MĂRIRE
2: folosind indicele predecesorului, se identifică un drum P de mărire a fluxului, de la nodul 1
la nodul n
3: $mr \leftarrow min\{r(i,j)|(i,j) \in P\}$
4: se adaugă mr unităţi de flux de-a lungul drumului P
5: **end procedure**

Algoritmul 8 Algoritmul drumului minim de mărire a fluxului.

1: $f \leftarrow 0$
2: se obţin etichetele distanţă exacte $d(i)$
3: $i \leftarrow 1$
4: **while** $d(1) < n$ **do**
5: **if** i are un arc admisibil **then**
6: Înaintare(i)
7: **if** $i = n$ **then**
8: execută Mărire() şi iniţializează $i = 1$
9: **end if**
10: **else**
11: Retragere(i)
12: **end if**
13: **end while**

Următoarele două teoreme caracterizează algoritmul de determinare a drumului minim de mărire a fluxului [1].

Teorema 6. *Algoritmul de determinare a drumului minim de mărire a fluxului calculează în mod corect un flux maxim în reţeaua* $S = (N, A, u)$.

Teorema 7. *Algoritmul de determinare a drumului minim de mărire a fluxului are complexitatea* $O(n^2m)$.

Modelele de reţele statice au multiple posibilităţi de aplicare, însă, în anumite cazuri, timpul joacă un rol esenţial. Aceste situaţii necesită a fi modelate sub forma unor reţele dinamice.

Problema identificării fluxului maxim într-o reţea dinamică de forma $D = (N, A, u, h)$, cu funcţia timp definită ca $h : A \to \mathbb{N}$, este mult mai complexă decât problema identificării fluxului maxim în reţeaua statică $S = (N, A, u)$. Se poate însă rezolva această problemă complexă prin transformarea reţelei D într-o reţea statică extinsă, echivalentă reţelei D, pe care o notăm cu $S_e = (N_e, A_e, u_e)$. Avem în acest caz o abordare statică a unei probleme dinamice.

Fie $H = \{0, 1, ..., T\}$ mulţimea perioadelor de timp. Avem următoarele definiţii de mulţimi şi funcţii, utilizate pentru transformarea reţelei: $N_e^{'} = \{i_t|i \in N, t \in H\}, A_e^{'} = \{(i_t, j_\theta)|(i,j) \in A; t, \theta \in H\}, u_e^{'}(i_t, j_\theta) = u(i,j), (i_t, j_\theta) \in A_e^{'}$.

Pe baza acestora se obţine reţeaua statică extinsă $S_e^{'} = (N_e^{'}, A_e^{'}, u_e^{'})$, cu mai multe noduri sursă, $1_1, ..., 1_T$, şi mai multe noduri stoc, $n_1, ...n_T$.

În continuare vom reduce problema cu surse şi stocuri multiple din reţeaua extinsă $S_e^{'} = (N_e^{'}, A_e^{'}, u_e^{'})$ la o problemă cu un singur nod sursă şi un singur nod stoc, introducând un nod sursă suplimentar, notat cu 0 şi un nod stoc suplimentar, notat cu $n+1$. Astfel se obţine reţeaua extinsă $S_e = (N_e, A_e, u_e)$, cu $N_e = N_e^{'} \cup \{0, n+1\}, A_e = A_e^{'} \cup \{(0, 1_t)|t \in H\} \cup \{(n_t, n+1)|t \in H\}, u_e(0, 1_t) = u_e(n_t, n+1) = \infty, t \in H$.

Mai multe detalii despre studiul reţelelor dinamice se găsesc în [1],[12],[14],[26],[69].

3.1.2 Fluxul maxim în reţelele cu toleranţă la întârzieri şi buffer cu dimensiune limitată. Abordarea statică

Această abordare aparţine autoarei tezei [54].

Reţeaua de tip DTN este un caz special de reţea dinamică [43],[85]. Se consideră un interval de timp $H = [t_0, T)$, unde t_0 şi T reprezintă timpul de start şi, respectiv, timpul final. Intervalul H este împărţit în q intervale de timp mai mici, notate $\tau_k = [t_{k-1}, t_k)$, cu $k = 1, 2, ..., q$.

Fie $D = (N, A, H, u_\tau, c_\tau, b_\tau)$ reţeaua dinamică având mulţimea de noduri $N = \{1, ..., n\}$, mulţimea de arce $A = \{a_1, ..., a_m\}$, intervalul de timp $H = [t_0, T)$, limita superioară (capacitatea) $u_\tau(i, j) = (u_{\tau_1}(i, j), ..., u_{\tau_q}(i, j)), (i, j) \in A$, seria de transfer $c_\tau = (c_{\tau_1,\tau_2}(i), ..., c_{\tau_{q-1},\tau_q}(i)), i \in N$, unde $c_{\tau_{k-1},\tau_k}(i)$ reprezintă datele transferate între intervalele de timp τ_{k-1} şi τ_k în nodul i. Valorile iniţiale ale lui $c_\tau(i)$, pentru oricare $i \in N$, sunt egale cu 0, iar spaţiile de stocare ale nodurilor sunt notate cu $b_\tau(i) = (b_{\tau_1,\tau_2}(i), ..., b_{\tau_{q-1},\tau_q}(i))$, cu $i \in N - \{1, n\}, b_{\tau_{k-1},\tau_k}(1) = b_{\tau_{k-1},\tau_k}(n) = \infty, k = 1, ..., q$.

Problema fluxului maxim pentru o reţea de tip DTN în care nodurile au buffer cu dimensiune limitată se rezumă la a trimite cât mai multe unităţi de flux de la nodul sursă 1 la nodul stoc n, fără a încălca valoarea capacităţii arcelor şi constrângerile nodurilor. În această lucrare, rezolvarea acestei probleme are o abordare statică.

Pentru problema fluxului maxim într-o reţea de tip DTN în care nodurile au buffer cu dimensiune limitată se foloseşte o reţea extinsă $S_e = (N_e, A_e, u_e)$, care prezintă următoarele modificări faţă de cea prezentată anterior: $N_e^{'} = \{i_k|i \in N, k = 1, ..., q\}, A_e^{'} = \{(i_k, j_k)|(i, j) \in A, k = 1, ..., q\} \cup \{(i_k, i_{k+1})|k = 1, ..., q-1\}, u_e(i_k, j_k) = u_{\tau_k}(i, j), k = 1, ..., q, u_e(i_k, i_{k+1}) = b_{\tau_k,\tau_{k+1}}(i, j), k = 1, ..., q-1$.

Pentru identificarea drumului minim de mărire a fluxului în reţeaua statică nou construită, se va folosi Alg. 8, prezentat anterior. Fluxul în reţeaua $S_e = (N_e, A_e, u_e)$ este echivalent cu un flux maxim din reţeaua dinamică $D = (N, A, H, u_\tau, c_\tau, b_\tau)$. Reţeaua reziduală $R_e = (N_e, \tilde{A}_e, r_e)$ din abordarea statică a reţelei DTN este construită similar cu reţeaua reziduală clasică $R = (N, \tilde{A}, r)$, descrisă mai sus.

În continuare sunt prezentate următoarele două teoreme:

Teorema 8. *Algoritmul de determinare a drumului minim de mărire a fluxului calculează în mod corect un flux maxim în reţeaua statică extinsă $S_e = (N_e, A_e, u_e)$.*

Teorema 9. *Algoritmul de determinare a drumului minim de mărire a fluxului, aplicat reţelei statice extinse $S_e = (N_e, A_e, u_e)$, are complexitatea $O(T^3 n^2 (n+m))$.*

3.1.3 Exemplu de calcul al fluxului maxim

Vom porni de la exemplul din lucrarea [85], unde s-a obţinut un flux maxim într-o abordare dinamică. Acest exemplu se va relua, pentru a obţine un flux maxim cu abordare statică.

Graful suport este cel prezentat în Fig. 3.1, unde nodul sursă este 1 şi nodul stoc este 4, iar intervalul de timp dat este $H = [t_0, T) = [0, 5)$.

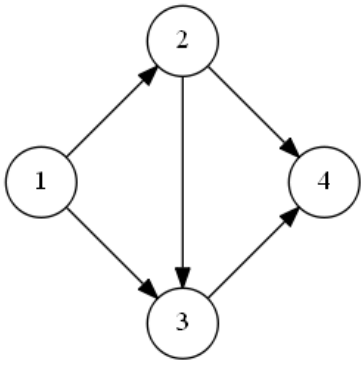


Figura 3.1: Graful suport G=(N,A)(din [85])

În acest caz, intervalul H este împărţit în cinci intervale de timp mai mici: $\tau_1 = [0, 1), \tau_2 = [1, 2), \tau_3 = [2, 3), \tau_4 = [3, 4), \tau_5 = [4, 5)$. Vom presupune că fluxul maxim iniţial are valoarea 0.

Valorile capacităţilor pentru $u_{\tau_k}(i, j)$, cu $k = 1, 2, 3, 4, 5$ şi $(i, j) \in A$, sunt prezentate în Tab. 3.1

(i,j) \ $u_\tau(i,j)$	$u_{\tau_1}(i,j)$	$u_{\tau_2}(i,j)$	$u_{\tau_3}(i,j)$	$u_{\tau_4}(i,j)$	$u_{\tau_5}(i,j)$
(1,2)	7	0	6	0	0
(1,3)	0	3	2	0	0
(2,3)	6	0	1	0	0
(2,4)	0	7	0	4	2
(3,4)	1	0	0	5	1

Tabela 3.1: Valorile capacităţilor $u_{\tau_k}(i, j), k = 1, 2, 3, 4, 5$ şi $(i, j) \in A$

Valorile bufferelor nodurilor sunt $b_\tau(i) = (b_{\tau_1,\tau_2}, b_{\tau_2,\tau_3}, b_{\tau_3,\tau_4}, b_{\tau_4,\tau_5}), i \in N$ şi sunt prezentate în Tab. 3.2.

i \ $b_\tau(i)$	$b_{\tau_1,\tau_2}(i)$	$b_{\tau_2,\tau_3}(i)$	$b_{\tau_3,\tau_4}(i)$	$b_{\tau_4,\tau_5}(i)$
1	∞	∞	∞	∞
2	5	5	5	5
3	5	5	5	5
4	∞	∞	∞	∞

Tabela 3.2: Valorile bufferelor $b_\tau(i), i \in N$

În reţeaua extinsă este folosit algoritmul de determinare a drumului minim de mărire a fluxului pentru a identifica drumul $P_1 = (0, 1_1, 2_1, 3_1, 4_1, 5)$, cu valoarea fluxului $f(P_1) = 1$. După ce se aplică mărirea de flux, se obţine reţeaua reziduală $R_e = (N_e, \tilde{A}_e, r_e)$. Ulterior se determină succesiv toate drumurile de mărire a fluxului:

- $P_2 = (0, 1_1, 2_1, 2_2, 4_2, 5)$ cu valoarea fluxului $f(P_2) = 5$
- $P_3 = (0, 1_3, 2_3, 2_4, 4_4, 5)$ cu valoarea fluxului $f(P_3) = 4$
- $P_4 = (0, 1_3, 3_3, 3_4, 4_4, 5)$ cu valoarea fluxului $f(P_4) = 2$
- $P_5 = (0, 1_2, 3_2, 3_3, 3_4, 4_4, 5)$ cu valoarea fluxului $f(P_5) = 1$
- $P_6 = (0, 1_3, 2_3, 2_4, 2_5, 4_5, 5)$ cu valoarea fluxului $f(P_6) = 1$
- $P_7 = (0, 1_3, 2_3, 3_3, 3_4, 4_4, 5)$ cu valoarea fluxului $f(P_7) = 1$
- $P_8 = (0, 1_1, 2_1, 3_1, 3_2, 3_3, 3_4, 4_5, 5)$ cu valoarea fluxului $f(P_8) = 1$

În reţeaua reziduală $R_e = (N_e, \tilde{A}_e, r_e)$ nu mai există niciun drum de mărire a fluxului, şi conform Teoremei 5 rezultă că am ajuns la un flux maxim. Valoarea fluxului maxim obţinut este $v_e = f(P_1) + f(P_2) + f(P_3) + f(P_4) + f(P_5) + f(P_6) + f(P_7) + f(P_8) = 1 + 5 + 4 + 2 + 1 + 1 + 1 + 1 = 16$.

3.2 MaxDelivery: Cercetări preliminarii

Dacă în secţiunea precedentă s-a realizat maximizarea fluxului într-o reţea DTN, printr-o abordare statică, în secţiunea curentă se va realiza maximizarea transmiterii de mesaje într-o reţea DTN, printr-o politică de gestiune a bufferului, bazată pe un algoritm de abandonare a mesajelor în cazul supraîncărcării spaţiului de stocare al nodurilor reţelei.

Rezultatele originale din această secţiune aparţin autoarei acestei lucrări şi sunt prezentate în lucrarea [55].

Din studiul diferitelor politici de gestiune a bufferului, se poate observa că au fost identificate mai multe criterii care pot influenţa abandonarea mesajelor şi programarea acestora pentru retransmitere. Aceste criterii sunt:

- Timpul de viaţă rămas al mesajului (remaining TTL) - cât timp a rămas până la expirarea mesajului.
- Numărul de hopuri ale mesajului până în prezent - prin câte noduri intermediare a trecut mesajul de la nodul sursă până la nodul curent.
- Numărul de replici ale mesajului - numărul de noduri din reţea care deţin o copie a mesajului la un moment dat.
- Dimensiunea mesajului
- Vârsta mesajului - timpul scurs de când a fost creat mesajul.

- Costul livrării mesajului - unii algoritmi de rutare folosesc predictibilitatea livrării mesajului la destinaţie pentru a defini costul pe arce.
- Distanţa până la destinaţie - cât are de parcurs mesajul din bufferul curent până ajunge în bufferul nodului destinaţie.

3.2.1 O nouă politică de abandonare a mesajelor

Crearea algoritmului MaxDelivery a plecat de la ideea introducerii unei politici de gestiune a bufferului pentru strategiile de rutare care nu aveau implementată o astfel de abordare. Am propus în acest scop, în [55], o modalitate de a elibera bufferul congestionat al nodurilor pentru protocoalele de rutare Epidemic şi PRoPHET.

Obiectivul politicilor de gestiune a spaţiului de stocare al nodurilor în reţelele de tip DTN este acela de a prioritiza mesajele în coada de aşteptare. În cazul abordării din [55], prioritizarea se face printr-o strategie nouă de ştergere a mesajelor.

Majoritatea politicilor de abandonare a mesajelor folosesc o singură caracteristică a mesajului sau a reţelei, aspect care s-a putut observa în secţiunea în care au fost prezentate strategiile de gestiune a bufferului în DTN.

Am putut observa algoritmi care se bazează pe:

- timpul de viaţă al mesajului: DOA(Drop Oldest), DY(Drop Youngest), SHLI(Shortest Life First)
- momentul sosirii în buffer: FIFO, LIFO, DLR(Drop Least Recently Received)
- numărul de transmiteri ale mesajului: MOFO(Most Forwarded), LF(Least Forwarded), max-max
- dimensiunea mesajului: DL(Drop Largest), E-drop, T-drop, Mean-drop
- frecvenţa întâlnirii dintre noduri: DLE(Drop Least Encountered), MOPR(Most Favourably Forwarded), LEPR(Least Probable First), DLK(Drop Less Known)

Pentru a putea detecta în mod echitabil mesajul care va fi şters, am ţinut cont de mai multe proprietăţi ale mesajului sau ale reţelei. Am ales astfel numărul de transmiteri ale mesajului pentru a asigura faptul că acesta este transmis cel puţin o dată, timpul de viaţă al mesajului şi numărul de noduri parcurse.

Îmbinând aceste criterii, am creat astfel o funcţie utilitate care se asociază fiecărui mesaj din reţea, definită prin formula de mai jos:

$$f_{drop}(x) = \frac{TTL}{Init\_TTL}(HC + MF)$$

Notaţiile din formulă reflectă în primul rând caracteristici ale mesajului, având următoarele semnificaţii:

- TTL (Time To Live) - reprezintă timpul de viaţă al mesajului, care este în continuă descreştere, până la zero. Atunci când valoarea lui TTL ajunge egală cu zero, mesajul va fi şters din buffer.

- Init_TTL - reprezintă timpul de viaţă iniţial, cel asociat la crearea mesajului.
- HC (Hop Count) - reprezintă numărul de hopuri (noduri de reţea) pe care le-a străbătut mesajul până la nodul curent. Atunci când un nod sursă generează un mesaj, acesta va avea asociată o proprietate HC cu valoarea zero. Atunci când mesajul se transmite unui alt nod, valoarea lui HC va fi incrementată cu o unitate.
- MF (My Forwarding) - reprezintă numărul de copii ale mesajului pe care le-a transmis nodul curent către nodurile cu care a intrat în contact.

Raportul $\frac{TTL}{Init\_TTL}$ favorizează ştergerea mesajelor nou create, în contrast cu metoda SHLI, care favorizează ştergerea mesajelor cu cel mai scurt timp de viaţă rămas. Rezultatul acestui raport este echilibrat de către numărul de transmisii ale mesajului, calculat prin însumarea numărului de transmisii efectuate de nodul curent şi a numărului de hopuri pe care le-a parcurs mesajul până în prezent.

Prin abordarea propusa în [55] nu se doreşte nici favorizarea ştergerii mesajelor care sunt mai noi în sistem, dar nici a mesajelor care au călătorit mai mult în reţea.

Astfel, nodul va sorta în buffer mesajele în ordinea crescătoare a funcţiei utilitate.

Conform acestor criterii, în cazul în care bufferul nodului se umple şi soseşte un mesaj nou de la un nod de contact, nodul curent va elimina din buffer mesajul cu valoarea cea mai mare a funcţiei utilitate $f_{drop}(x)$, descrisă mai sus. Această eliminare se va efectua succesiv, până când mesajul nou sosit va avea loc în buffer. Se presupune că mesajele cu valoarea cea mai mare a utilităţii au cea mai mică şansă de a ajunge la destinaţie.

3.2.2 Validarea funcţiei utilitate

În [55] au fost prezentate câteva scenarii care să demonstreze validitatea funcţiei utilitate create.

Se notează astfel $HC + MF$ cu TCV (Transmission Count Value), această valoare reprezentând numărul transmisiilor mesajului. Timpul de viaţă iniţial asociat mesajelor se consideră că este constant, în ideea în care fiecărui mesaj din sistem i se va asocia la creare aceeaşi valoare a timpului de viaţă.

În cazul congestionării bufferului, vom considera două mesaje, M_1 şi M_2, care ar purea fi şterse. Acestea au asociate funcţiile utilitate $f_{drop}(M_1)$, respectiv $f_{drop}(M_2)$, cu definiţiile următoare:

$$f_{drop}(M_1) = \frac{TTL_1}{Init\_TTL} TCV_1$$

$$f_{drop}(M_2) = \frac{TTL_2}{Init\_TTL} TCV_2$$

În contextul de faţă, se consideră următoarele 3 cazuri:

Cazul 1. $TTL_1 > TTL_2$ şi $TCV_1 < TCV_2$

Aici apar două posibilităţi:

1.1) $TTL_1 \cdot TCV_1 > TTL_2 \cdot TCV_2 \Rightarrow f_{drop}(M_1) > f_{drop}(M_2)$

Acesta este cazul în care M_1 va fi mesajul ales pentru ştergere. Presupunând că $TTL_1 > TTL_2$, atunci M_1 are un timp de viaţă mai mare şi nodul curent îl va şterge, lăsând responsabilitatea transmiterii acestuia altor noduri.

1.2) $TTL_1 \cdot TCV_1 < TTL_2 \cdot TCV_2 \Rightarrow f_{drop}(M_1) < f_{drop}(M_2)$

Acesta este cazul în care M_2 va fi mesajul ales pentru ştergere. Deoarece $TTL_1 > TTL_2$, atunci înseamnă că mesajul M_2 a fost deja transmis la suficient de multe noduri şi există multe copii ale sale în reţea. În concluzie, ştergerea mesajului M_2 nu va afecta prea mult şansa lui de livrare la destinaţie.

Cazul 2. $TTL_1 = TTL_2$ şi $TCV_1 < TCV_2 \Rightarrow TTL_1 \cdot TCV_1 < TTL_2 \cdot TCV_2 \Rightarrow f_{drop}(M_1) < f_{drop}(M_2)$

În acest caz va fi şters mesajul M_2. Deoarece $TCV_1 < TCV_2$, calculele devin aceleaşi ca şi în cazul 1.2).

Cazul 3. $TTL_1 < TTL_2$ şi $TCV_1 = TCV_2 \Rightarrow f_{drop}(M_1) < f_{drop}(M_2)$

Deoarece $TTL_1 < TTL_2$, mesajul şters va fi M_2. Motivaţia este similară cu cea pentru cazul 1.1).

Algoritmul care utilizează funcţia utilitate

Testarea eficienţei funcţiei utilitate prezentate mai sus a fost efectuată cu ajutorul simulatorului ONE. Algoritmul care descrie metoda din simulator, cea care foloseşte funcţia propusă, este cel prezentat în Alg. 9

Algoritmul 9 Metoda care eliberează spaţiul din buffer

```
1: procedure makeRoomForNewMessage(buffer, msg)
2:     buffer.sort                          ▷ sortarea sa face după funcţia utilitate
3:     while buffer.size < msg.size do
4:         removeBufferTop;                 ▷ se şterge mesajul cu utilitatea cea mai mare
5:     end while
6: end procedure
```

3.2.3 Rezultate obţinute prin simulare

Aplicaţia propusă în [55] compară metricile de performanţă ale algoritmilor Epidemic şi PRoPHET în varianta lor clasică şi în varianta actualizată cu politica de abandonare a mesajelor, pe baza funcţiei utilitate $f_{drop}(x)$.

Caracteristicile generale ale sistemului sunt:

- Timpul total al simulării: 12h
- Dimensiunea hărţii: 4500m x 3400m
- Modelul de mobilitate: cel care foloseşte drumul cel mai scurt între două puncte de pe hartă

- Tehnica de gestiune a bufferului: bazată pe funcţia utilitate descrisă anterior
- Protocoalele de rutare: Epidemic, PRoPHET
- Viteza de transmisie a datelor: 250kbps
- Raza de acţiune bluetooth:10m
- Dimensiunea mesajelor: 500kB - 1MB
- Intervalul de creare a mesajelor: 25s - 30s
- TTL: 1h - 3h - 5h - 7h - 9h

In completarea acestora avem si caracteristici specifice categoriilor de noduri, prezentate în Tab. 3.3.

Pietoni	
Parametru	**Valoare**
Numărul de noduri	80
Modelul de mobilitate	Cel care foloseşte drumul cel mai scurt între două puncte de pe hartă
Dimensiunea bufferului	5MB - 6MB - 7MB - 8MB - 9MB
Timpul de aşteptare între deplasări	0s - 120s
Viteza	2km/h - 5km/h
Maşini	
Parametru	**Valoare**
Numărul de noduri	40
Modelul de mobilitate	Cel care foloseşte drumul cel mai scurt între două puncte de pe hartă
Dimensiunea bufferului	5MB - 6MB - 7MB - 8MB - 9MB
Timpul de aşteptare între deplasări	0s - 120s
Viteza	10km/h - 50km/h
Tramvaie	
Parametru	**Valoare**
Numărul de noduri	6
Modelul de mobilitate	Cel care foloseşte ruta predefinită de deplasare
Dimensiunea bufferului	50MB
Timpul de aşteptare între deplasări	10s - 30s
Viteza	25km/h - 35km/h

Tabela 3.3: Configuraţia nodurilor reţelei

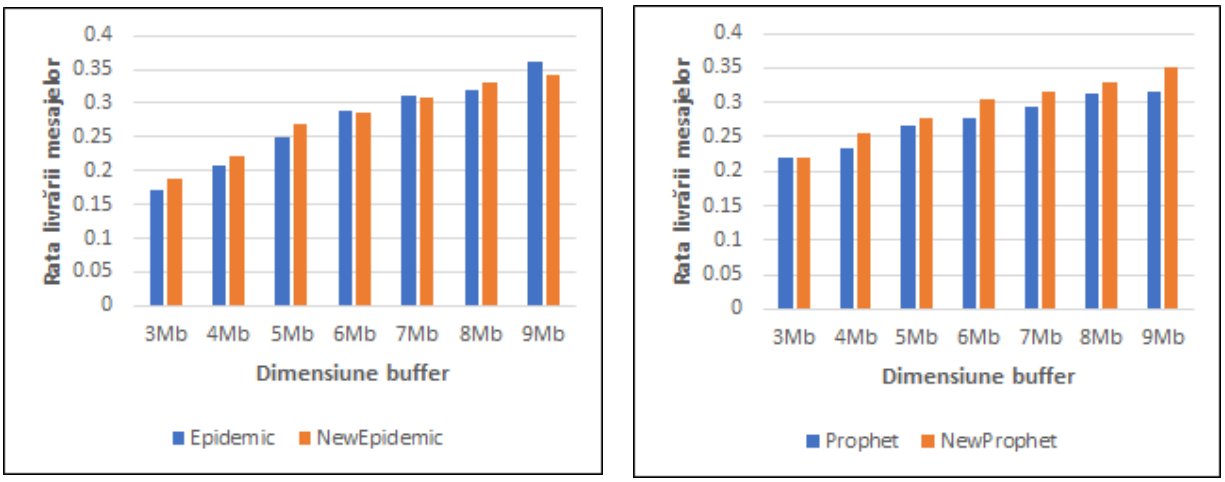
Scenariul de simulare ales este cel de bază, oferit de către simulator. Acesta conţine trei tipuri de noduri: pietoni, maşini şi tramvaie. Nodurile se deplasează pe harta oraşului

Helsinki. Metricile de performanţă urmărite sunt: rata livrării mesajelor şi timpul mediu de staţionare în buffer al mesajelor.

Aşa cum se poate observa din tabelele menţionate, simulările au fost făcute variind doi dintre parametrii sistemului: timpul de viaţă al mesajului şi dimensiunea bufferului. Modelul de mobilitate folosit de pietoni şi maşini este drumul cel mai scurt, iar tramvaiele folosesc modelul de deplasare bazat pe un orar predefinit.

Rezultatele obţinute prin variaţia dimensiunii bufferului sunt prezentate în Fig. 3.2. Se poate observa că politica de gestiune a bufferului adăugată celor doi algoritmi aduce o îmbunătăţire a ratei de livrare a mesajelor, în special algoritmului PRoPHET (a se vedea Fig. 3.2b).

În ceea ce priveşte algoritmul Epidemic, rezultatele obţinute sunt similare atât pentru cazul clasic de abandonare a mesajelor (cel bazat pe algoritmul FIFO), cât şi pentru noua abordare propusă. Există totuşi, în noua variantă, o uşoară apreciere a valorii atunci când dimensiunea bufferului este sub 6MB şi o uşoară depreciere atunci când bufferul are dimensiunea de 9MB (a se vedea Fig. 3.2a).



(a) Epidemic (b) PRoPHET

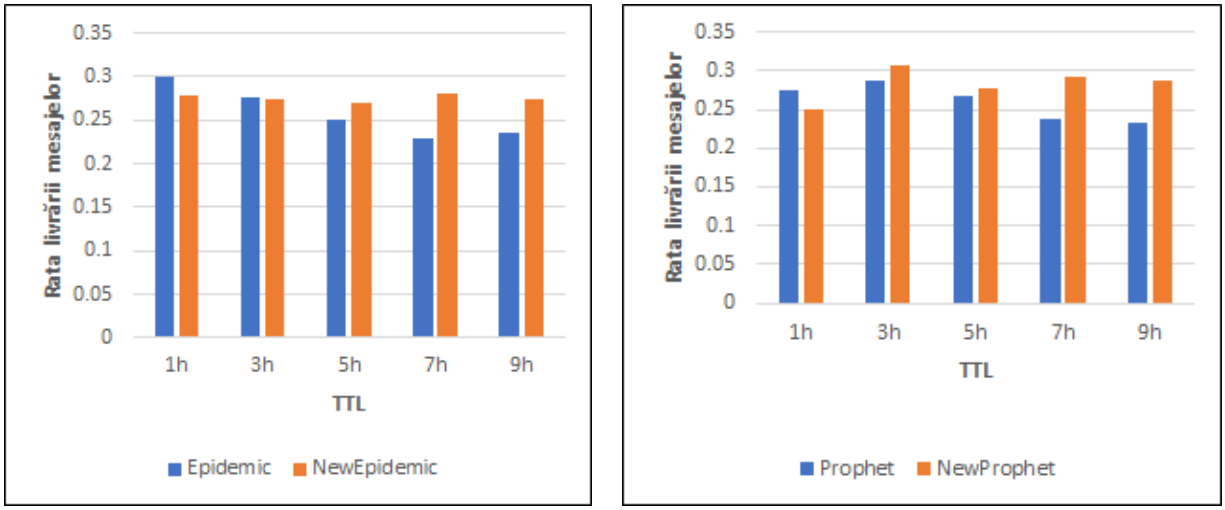
Figura 3.2: Eficienţa ratei de livrare a mesajelor, variind dimensiunea bufferului (din [55])

Rezultatele care se obţin atunci când se variază valoarea TTL sunt prezentate în Fig. 3.3. Se poate observa din figură că atunci când timpul de viaţă al mesajului este mai mare de 3 ore, noua politică de abandonare îmbunătăţeşte rata de livrare a mesajelor atât pentru algoritmul Epidemic (a se vedea Fig. 3.3a), cât şi pentru PRoPHET (a se vedea Fig. 3.3b).

În Fig. 3.4 şi Fig. 3.5, se poate observa că noua modalitate de gestiune a bufferului oferă valori ale timpului petrecut în buffer mult mai mici decât variantele standard ale algoritmilor Epidemic şi PRoPHET.

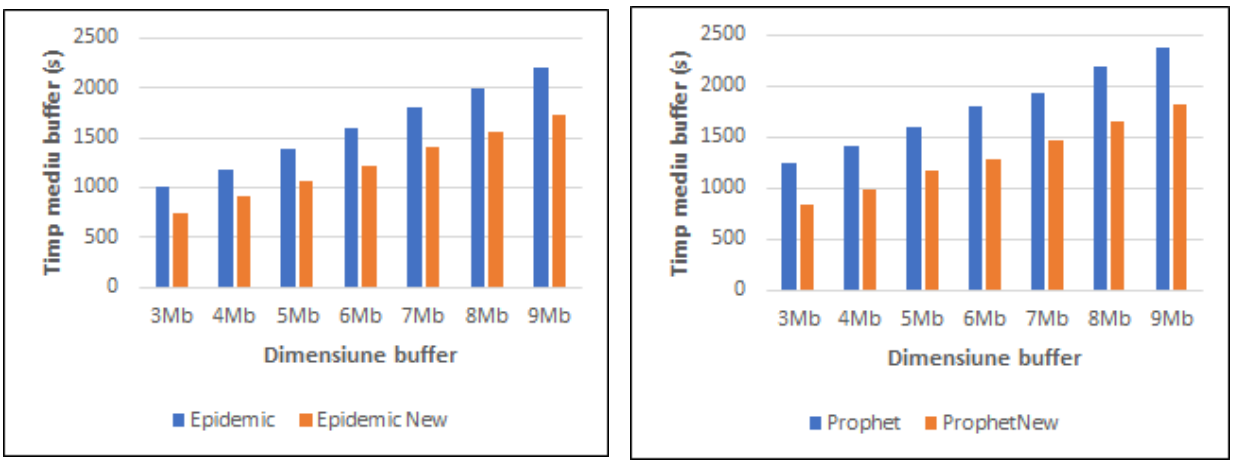
Scopul politicii de gestiune propuse, bazată pe ştergerea de mesaje în cazul congestionării bufferului, este acela de a creşte numărul de mesaje livrate şi de a micşora simultan numărul de copii ale mesajelor care se vor replica în reţea pentru a garanta ajungerea acestora la destinaţie.

Într-o reţea cu toleranţă la întârzieri este mai importantă garanţia faptului că mesajele vor ajunge la destinaţie decât livrarea lor într-un interval scurt de timp. Astfel, pentru moment nu am urmărit şi scăderea timpului de livrare al mesajelor.



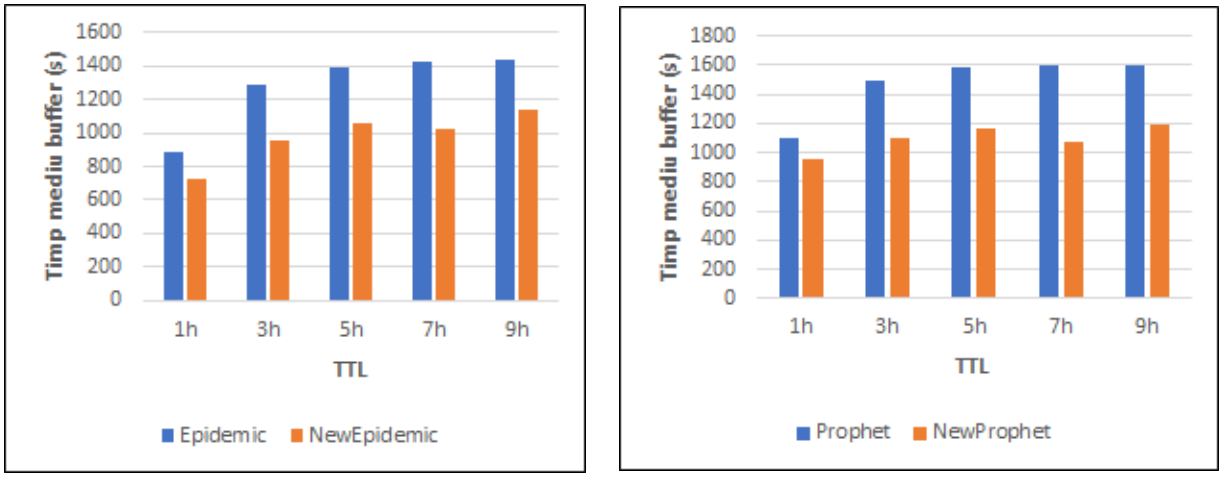
(a) Epidemic (b) PRoPHET

Figura 3.3: Eficienţa ratei de livrare, variind timpul de viaţă al mesajelor (din [55])



(a) Epidemic (b) PRoPHET

Figura 3.4: Timpul mediu petrecut în buffer, variind dimensiunea bufferului (din [55])



(a) Epidemic (b) PRoPHET

Figura 3.5: Timpul mediu petrecut în buffer, variind timpul de viaţă al mesajelor (din [55])

3.3 Descrierea algoritmului MaxDelivery

Rezultatele originale din această secţiune aparţin autoarei acestei lucrări şi se regăsesc în lucrarea [56], acceptată pentru publicare.

Conform cu specificaţiile din secţiunea precedentă, algoritmul propus are ca obiectiv îmbunătăţirea unor criterii de performanţă ale unui sistem reprezentat printr-o reţea de tip DTN. În continuare putem observa câteva dezavantaje ale algoritmilor clasici de rutare:

- Epidemic - nu este eficient în reţele dense, în care circulă un număr mare de mesaje.
- Spray and Wait - necesită capacitate mare de stocare datorită faptului că păstrează mesajele în buffer mult timp.
- MaxProp - are nevoie de un buffer de dimensiune mare şi este consumator de energie datorită prelucrărilor complexe pe care le efectuează.

Scopul algoritmului MaxDelivery este acela de a îndeplini următoarele cerinţe: livrarea unui număr cât mai mare de mesaje la destinaţie prin optimizarea selectării mesajului cu cele mai mari şanse de livrare, evitarea rutelor şi a nodurilor congestionate.

Pentru îndeplinirea obiectivelor, algoritmul propus se bazează pe următoarele acţiuni:

- descoperirea vecinilor pentru a cunoaşte informaţii despre reţea
- maximizarea trimiterii de mesaje la destinaţie
- eliminarea mesajelor care nu mai pot ajunge la destinaţie, pentru a oferi şansa altor mesaje să fie livrate
- curăţarea periodică a bufferului

Fiecare nod al reţelei va menţine o listă a nodurilor cu care a intrat în contact. În momentul conexiunii, acestea îşi vor transmite gradul de ocupare al bufferului la acel moment, vor actualiza lista mesajelor livrate în vederea ştergerii copiilor acestora din buffer şi abia după aceea vor începe transferul de mesaje.

În cele ce urmează vor fi detaliate părţile componente ale algoritmului MaxDelivery.

3.3.1 Metoda de transmitere a mesajelor

Pentru a creşte numărul de mesaje livrate, fiecare nod are bufferul împărţit în trei cozi de priorităţi, pregătite pentru transmiterea mesajelor.

Prima coadă, notată DQ, are prioritate 0 la transmitere şi conţine mesajele care au ca destinaţie nodul de contact. Mesajele sunt sortate în funcţie de dimensiune şi se trimit prima dată mesajele mai mici, asigurând un număr mai mare de transmiteri.

Cea de-a doua coadă, notată NQ, cu prioritate 1 la transmitere, conţine mesajele care au ca destinaţie un vecin al nodului de contact. Mesajele din această coadă sunt sortate de asemenea după dimensiune.

Cea de-a treia coadă, notată OQ, conţine restul mesajelor din bufferul nodului, sortate după o funcţie utilitate notată $f_{fwd}(x)$, cu următoarea definiţie:

$$f_{fwd}(x) = \begin{cases} HC + MF, & \text{dacă } HC + MF < TN/2 \\ msgSize, & \text{în caz contrar} \end{cases}$$

HC reprezintă numărul de hopuri, MF reprezintă numărul de transmisii locale ale mesajului, TN (Total Neighbors în limba engleză) reprezintă numărul total de vecini ai nodului, iar $msgSize$ reprezintă dimensiunea mesajului.

Semnificaţia funcţiei $f_{fwd}(x)$: sunt favorizate mesajele care au fost transmise la mai puţin de jumătate dintre vecini, după care sunt favorizate mesajele în funcţie de dimensiunea lor. Valoarea $TN/2$ reprezintă un prag utilizat pentru a evita inundarea reţelei cu mesaje redundante.

Sortarea se face descrescător, astfel încât să fie transmise cu prioritate mesajele care au ajuns la cei mai puţini vecini, în ordinea crescătoare a dimensiunii lor.

Pentru a evita congestionarea reţelei, un nod va trimite mesaje doar vecinilor care au cel mult 90% din capacitatea de stocare ocupată.

Algoritmul transmiterii mesajelor

Algoritmul folosit pentru prioritizarea mesajelor în vederea transmiterii la nodul de legătură este descris în Alg. 10.

Algoritmul 10 Metoda pentru transmiterea de mesaje

```
1: procedure FWD
2:     DQ ← ∅                          ▷ mesajele destinate nodului de legătură
3:     NQ ← ∅                  ▷ mesajele destinate unui vecin al nodului de legătură
4:     OQ ← ∅                                  ▷ celelalte mesaje din buffer
5:     FILL_DEST(DQ);
6:     FILL_NEIGH(NQ);
7:     FILL_OTHER(OQ);
8:     SET_TRANSMISSION_RANGE(DQ, NQ, OQ);
9: end procedure
```

Alg. 10 face apelul a patru proceduri:

- procedura $FILL\_DEST(DQ)$ populează coada DQ, cea care are cea mai mare prioritate;
- procedura $FILL\_NEIGH(NQ)$ populează coada NQ, cea care are prioritate medie;
- procedura $FILL\_OTHER(OQ)$ populează coada OQ, cea care are cea mai mică prioritate;
- procedura $SET\_TRANSMISSION\_RANGE(DQ, NQ, OQ)$ ordonează cele trei cozi după prioritate.

Algoritmul 11 Popularea cozii de mesaje care au ca destinaţie nodul de legătură

```
1: procedure FILL_DEST(DQ)
2:     for msg ∈ allMessages do
3:         if msg.Dest = contactNode then
4:             DQ ← DQ + {msg};
5:         end if
6:     end for
7:     SORT_BY_SIZE(DQ);
8: end procedure
```

Algoritmii 11, 12 şi 13 reprezintă procedurile de populare a cozilor de priorităţi.

Algoritmul **FILL_DEST(DQ)** extrage din toate mesajele care se găsesc în bufferul nodului curent, notat cu $allMessages$ în algoritm, pe acelea care au ca nod destinaţie nodul actual de legătură al nodului curent. Mesajele extrase sunt inserate în coada DQ, după care, aceasta se sortează crescător în funcţie de dimensiunea mesajelor.

Teorema 10. *Algoritmul FILL_DEST(DQ) calculează în mod corect mulţimea mesajelor cu proprietatea că au ca destinatar nodul de legătură.*

Teorema 11. *Complexitatea algoritmului FILL_DEST(DQ) este* $O(Max\{m, m_1 \cdot \log m_1\})$*, unde* m_1 *este dimensiunea cozii DQ.*

Algoritmul 12 Popularea cozii de mesaje care au ca destinaţie un vecin al nodului de legătură

```
1: procedure FILL_NEIGH(NQ)
2:     for msg ∈ allMessages do
3:         for n ∈ contactNode.Neighbours do
4:             if msg.Dest = n then
5:                 NQ ← NQ + {msg};
6:             end if
7:         end for
8:     end for
9:     SORT_BY_SIZE(NQ);
10: end procedure
```

Algoritmul **FILL_NEIGH(NQ)** extrage dintre toate mesajele care se găsesc în bufferul nodului curent, notat cu $allMessages$ în algoritm, pe acelea care au ca nod destinaţie un vecin al nodul de legătură. Mesajele extrase sunt inserate în coada NQ, după care, aceasta se sortează crescător în funcţie de dimensiunea mesajelor, ca şi în cazul lui DQ.

Teorema 12. *Algoritmul FILL_NEIGH(NQ) calculează în mod corect mulţimea mesajelor cu proprietatea că au ca destinatar un vecin al nodului de legătură.*

Teorema 13. *Complexitatea algoritmului FILL_NEIGH(NQ) este* $O(Max\{m{\cdot}nv, m_2{\cdot}\log m_2\})$*, unde* m_2 *este dimensiunea cozii NQ.*

Algoritmul **FILL_OTHER(OQ)** preia din buffer celelalte mesaje, care nu s-au încadrat în cozile de priorităţi DQ şi NQ, şi le inserează în OQ. Se consideră vectorul *util* care are acelaşi

Algoritmul 13 Popularea cozii cu mesajele din buffer care nu s-au încadrat în cozile DQ şi NQ

```
1: procedure FILL_OTHER(OQ)
2:     OQ ← allMessages − DQ − NQ;
3:     util ← ∅;
4:     for msg ∈ OQ do
5:         util(msg) ← const · msg.Size;
6:     end for
7:     for msg ∈ OQ do
8:         //HC− numărul de hopuri
9:         //MF− numărul de copii ale mesajului răspândite de nodul curent
10:        //TN− numărul total de vecini
11:        if msg.HC + msg.MF < TN/2 then
12:            util(msg) ← msg.HC + msg.MF;
13:        end if
14:    end for
15:    SORT_BY_UTILITY(OQ);
16: end procedure
```

număr de elemente ca şi coada OQ, în care se inserează valorile funcţiei utilitate. Astfel, fiecărui mesaj din OQ îi va corespunde o valoare din vectorul *util*.

Vom defini o constantă, notată *const* în algoritm, care trebuie să aibă o valoare suficient de mare astfel încât, după ce se înmulţeşte cu dimensiunea mesajului, să nu fie depăşită de numărul de transmiteri ale mesajului.

Coada OQ este împărţită în două zone care au ca separator un prag definit de jumătate din numărul de vecini ai nodului curent.

Teorema 14. *Algoritmul FILL_OTHER(OQ) populează în mod corect coada OQ.*

Teorema 15. *Complexitatea algoritmului FILL_OTHER(OQ) este* $O(m_3 \cdot \log m_3)$*, unde* m_3 *este dimensiunea cozii OQ.*

Teorema 16. *Algoritmul FWD este corect definit în raport cu utilitatea propusă cozilor de priorităţi.*

Teorema 17. *Complexitatea algoritmului FWD este* $O(Max\{m \cdot nv, M \cdot \log M\})$*, cu* $M = max\{m_i | i = 1, 2, 3\}$.

3.3.2 Abandonarea mesajelor şi curăţarea bufferului

Metoda de abandonare a mesajelor

Formula funcţiei utilitate $f_{drop}(x)$, prezentată în secţiunea precedentă, a fost ulterior îmbunătăţită. Raportul $\frac{TTL}{Init\_TTL}$ a fost înlocuit cu timpul primirii mesajului. Astfel, funcţia devine:

$$f_{drop}(x) = RT \cdot (HC + MF)$$

Notaţiile pentru HC şi MF sunt aceleaşi ca mai sus, iar RT (Receive Time în limba engleză) reprezintă timpul când a fost primit mesajul de către nodul curent.

Pornind de la protocoalele de rutare bazate pe un comportament social, putem afirma că este important ca un nod să ia decizii şi pe baza informaţiilor despre reţea, nu doar pe baza celor referitoare la mesaje. Acest fapt a condus la introducerea timpului de sosire al mesajului în buffer în definiţia funcţiei $f_{drop}(x)$. Valoarea RT reprezintă o informaţie locală a mesajului, raportată la nodul curent, spre deosebire de TTL, care este o informaţie legată strict de mesaj. Noua abordare a îmbunătăţit considerabil procentul de livrare al mesajelor. Împreună cu celelalte strategii implementate de gestiune a bufferului, metoda de abandonare a mesajelor a crescut procentul de livrare de la 25-35% la 70-80%.

În timpul conexiunii dintre două noduri A şi B, dacă nodul B îi trimite un mesaj nodului A, al cărui buffer este plin sau nu are suficient spaţiu pentru a putea stoca acel mesaj, se efectuează următoarele operaţii:

1. se sortează mesajele din buffer în ordine crescătoare, după funcţia utilitate
2. dacă ultimele mesaje au aceeaşi valoare a funcţiei utilitate, atunci se sortează crescător după dimensiune

Conform acestei metode, se vor abandona mesajele cu utilitatea cea mai mare şi de dimensiunea cea mai mare. Astfel se încearcă abandonarea a cât mai puţine mesaje, care să aibă mai multe copii transmise deja în reţea.

Metoda de curăţare periodică a bufferului

Bufferul este curăţat în două faze:

1. Mesajele care au fost livrate la destinaţie sunt şterse în permanenţă, ori de câte ori se face livrarea.
2. Dacă spaţiul ocupat din buffer este cel puţin egal cu 90% din capacitate, atunci se şterg şi mesajele create de către nodul curent, cu condiţia ca acestea să fi fost transmise la cel puţin jumătate dintre nodurile vecine.

Cea de-a doua etapă de verificare se desfăşoară în momentul fiecărei oportunităţi de conectare a nodului.

3.3.3 Rezultate obţinute prin simulare

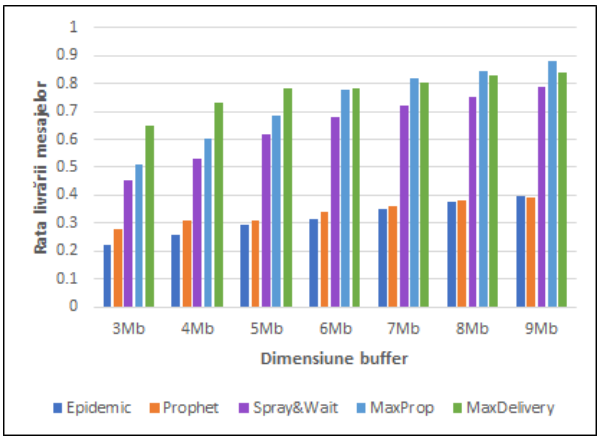
Pentru a evidenţia eficienţa algoritmului propus, rezultatele factorilor de performanţă ai acestuia au fost comparate cu cele ale algoritmilor implementaţi în simulatorul ONE. De data aceasta au fost făcute comparaţii cu rezultatele oferite de către toţi cei patru algoritmi de bază ai rutării în reţelele de tip DTN, care sunt deja implementaţi în aplicaţia simulatorului.

Mediul în care s-a realizat simularea actuală este similar celui în care s-au realizat simulările cercetărilor premergătoare, cu menţiunea că a fost mărită diferenţa dintre cel mai mic şi cel mai mare mesaj posibil. De data aceasta, dimensiunile mesajelor generate sunt în intervalul 250kB - 1MB. A fost aleasă această majorare a gradului de granularitate al mesajelor datorită faptului că poate evidenţia mai bine ştergerea mesajelor din buffer.

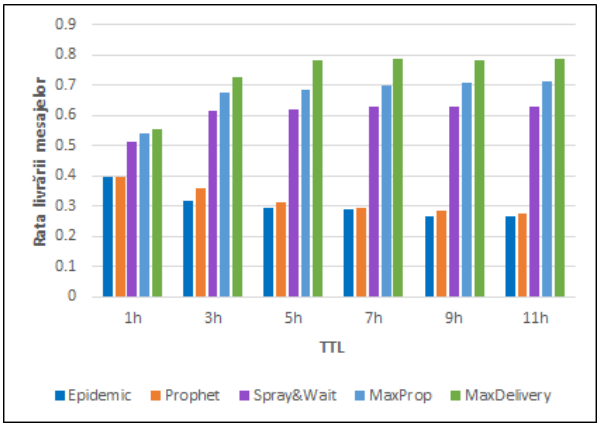
Rezultatele sistemului au fost testate tot prin variaţia timpului de viaţă al mesajelor şi prin variaţia spaţiului de stocare asociat nodurilor.

Rata de livrare a mesajelor

Graficele cu rezultatele ratei de livrare a mesajelor sunt prezentate în Fig. 3.6.



(a) Variaţia dimensiunii bufferului



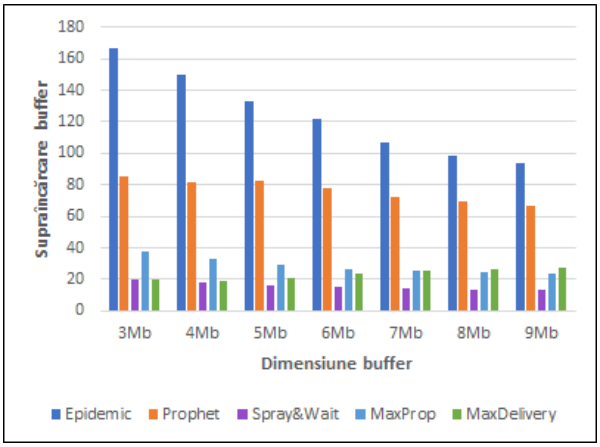
(b) Variaţia timpului de viaţă al mesajelor

Figura 3.6: Eficienţa ratei de livrare a mesajelor

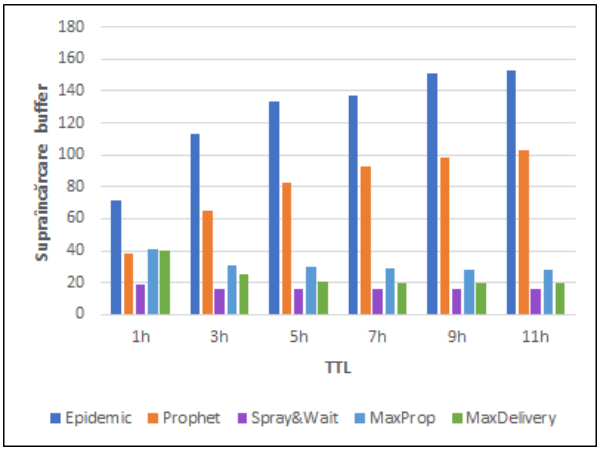
Din Fig. 3.6a reiese că pentru dimensiuni ale bufferului mai mici decât 5MB, MaxDelivery are o rată de livrare mult superioară celorlalţi algoritmi. Pentru un buffer de 6MB rezultatul este similar cu cel produs de către MaxProp, iar pentru dimensiuni ale bufferului peste 7MB se observă o uşoară depreciere a rezultatelor algoritmului MaxDelivery faţă de MaxProp. Algoritmul propus obţine valori mai bune faţă de rezultatele produse de către algoritmii de rutare Epidemic, PRoPHET şi Spray and Wait, indiferent de dimensiunea considerată a spaţiului de stocare. Din Fig. 3.6b reiese că MaxDelivery obţine cele mai bune valori ale ratei de livrare a mesajelor, indiferent de timpul de viaţă alocat mesajelor.

Supraîncărcarea reţelei

În Fig. 3.7 sunt prezentate rezultatele ratei de supraîncărcare a reţelei, cu graficele aferente acestor valori.



(a) Variaţia dimensiunii bufferului



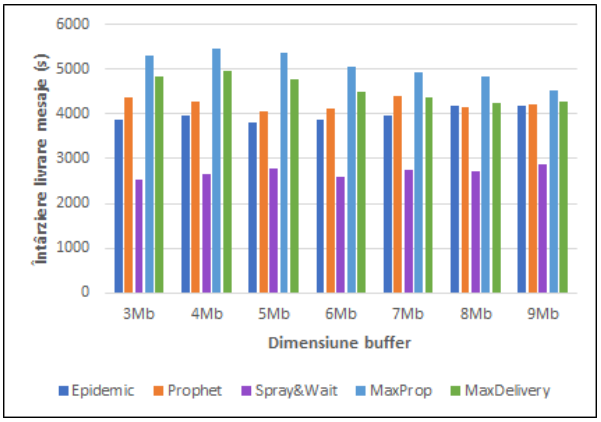
(b) Variaţia timpului de viaţă al mesajelor

Figura 3.7: Eficienţa gradului de supraîncărcare al reţelei

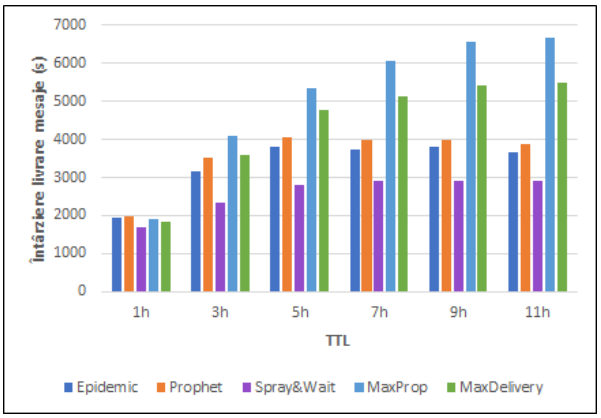
Din imaginile Fig. 3.7a şi Fig. 3.7b se observă că algoritmii Epidemic şi PRoPHET obţin rezultate foarte slabe, datorită faptului că trimit în reţea multe copii ale mesajelor. Algoritmii Spray and Wait, MaxProp şi MaxDelivery obţin rezultate mult mai bune, iar algoritmul de rutare propus este depăşit uşor din punctul de vedere al eficienţei acestei metrici de performanţă de către algoritmul Spray and Wait, care are o limitare a numărului de copii ale mesajelor.

Timpul de livrare al mesajelor

Rezultatele simulării pentru timpii de livrare ai mesajelor, măsuraţi în secunde, se observă în Fig. 3.8. Din reprezentările acestea reiese faptul că algoritmii cu o complexitate mai ridicată, care implementează şi politici de gestiune a bufferului, livrează mesajele la destinaţie cu o întârziere mai mare. Însă, dintre algoritmii cu complexitate ridicată, algoritmul propus oferă rezultate mai eficiente (MaxProp versus MaxDelivery).



(a) Variaţia dimensiunii bufferului

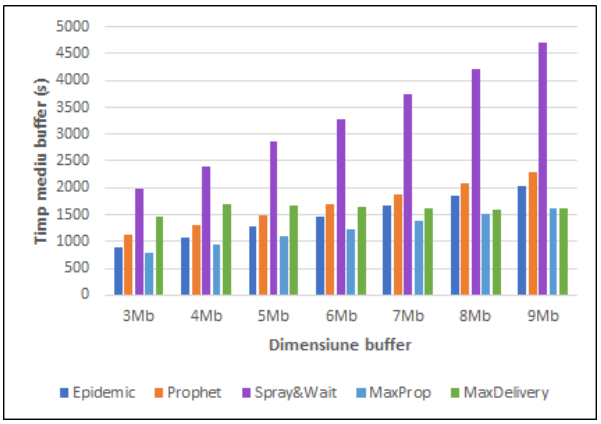


(b) Variaţia timpului de viaţă al mesajelor

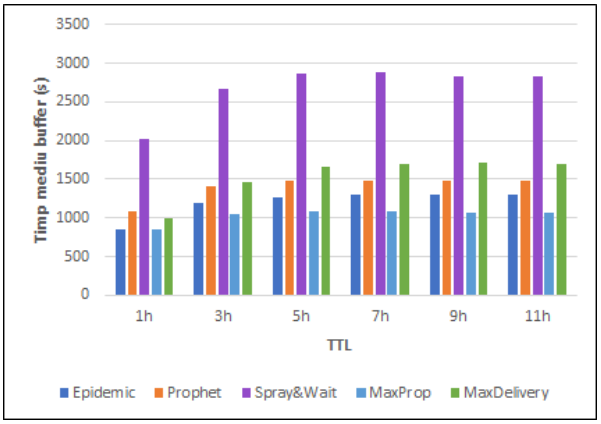
Figura 3.8: Eficienţa timpului de livrare al mesajelor

Timpul de staţionare în buffer al mesajelor

Valorile rezultate în urma simulărilor, măsurate în secunde, sunt prezentate în Fig. 3.9.



(a) Variaţia dimensiunii bufferului



(b) Variaţia timpului de viaţă al mesajelor

Figura 3.9: Eficienţa timpului mediu petrecut de mesaje în buffer

În Fig. 3.9a se observă că pentru algoritmul propus, valoarea medie a timpului petrecut în buffer de către mesaje are o uşoară descreştere odată cu creşterea dimensiunii bufferului de la 4MB la 9MB, spre deosebire de ceilalţi algoritmi care înregistrează o creştere a valorilor rezultate. Pentru valori mari ale bufferului, MaxDelivery ajunge să aiba cea mai bună valoare.

În Fig. 3.9b se observă că pentru algoritmul propus, valoarea medie a timpului petrecut în buffer de către mesaje înregistrează o uşoară creştere odată cu creşterea timpului de viaţă al mesajelor.

3.4 Mesaje cu diferite priorităţi în contextul algoritmului MaxDelivery

Într-un scenariu din viaţa reală, nu toate mesajele care se transmit într-o reţea au acelaşi grad de interes. În funcţie de contextul în care este implementată reţeaua, există posibilitatea generării de mesaje care au prioritate mare şi trebuie transmise urgent, precum şi a generării de mesaje cu o prioritate medie sau chiar scăzută, nefiind o problemă transmiterea lor cu întârziere.

Scenarii de acest gen sunt cele în care se implementează reţeaua DTN în caz de cutremur, de incendiu, de erupţie a unui vulcan etc. În aceste situaţii se doreşte transmiterea cu prioritate a mesajelor care indică punctele de localizare a unor victime sau solicitarea unei echipe de salvare.

Într-un astfel de context, obiectivul este acela de a maximiza rata de transfer a mesajelor cu prioritate ridicată şi de a reduce timpul de livrare al acestora.

Pentru ca protocolul de rutare să poată gestiona un astfel de comportament, am realizat o variantă a algoritmului MaxDelivery. Mesajele au fost împărţite pe două niveluri de prioritizare: mesaje cu prioritate ridicată şi mesaje cu prioritate scăzută. Noua variantă a algoritmului MaxDelivery urmăreşte eficientizarea criteriilor de performanţă ale reţelei pentru mesajele cu prioritate ridicată, în detrimentul celor cu prioritate scăzută. Modul în care se generează priorităţile mesajelor nu prezintă obiectul acestui studiu.

Logica algoritmului se modifică, în aceste condiţii, atât pentru strategia de transmitere a mesajelor, cât şi pentru strategia de abandonare a mesajelor în caz de supraîncărcare a bufferului.

Modalitatea de transmitere a mesajelor s-a modificat astfel: Atunci când apare o oportunitate de conectare, noul algoritm va selecta mesajul pe care îl va transmite conform unei noi funcţii utilitate, cu următoarea formulă:

$$f_{fwd}(x) = \begin{cases} (HC + MF) \cdot coef_{fwd}, & \text{dacă } HC + MF < TN/2 \\ msgSize \cdot coef_{fwd}, & \text{în caz contrar} \end{cases}$$

unde $coef_{fwd}$ reprezintă coeficientul de prioritate al mesajului la transmitere.

Modalitatea de abandonare a mesajelor s-a modificat astfel: Atunci când sosesc mesaje în buffer, iar acesta este plin, algoritmul selectează mesajul pe care îl va şterge

conform unei noi funcţii utilitate, cu următoarea formulă:

$$f_{drop}(x) = RC \cdot (HC + MF) \cdot coef_{drop}$$

unde $coef_{drop}$ reprezintă coeficientul de prioritate al mesajului pentru abandonare.

Modificând astfel funcţia utilitate, atât transmiterea cât şi ştergerea mesajelor depind şi de tipul de prioritate al mesajului. Cu toate că pentru exemplificare au fost considerate două categorii de priorităţi ale mesajelor, algoritmul se poate adapta pentru mesaje cu orice grad de prioritizare, în funcţie de contextul situaţiei reale în care se utilizează reţeaua. Coeficienţii de prioritate vor trebui modificaţi convenabil în acest caz.

3.4.1 Simulări în contextul prezenţei mesajelor cu priorităţi

În urma unei analize preliminarii, valorile de simulare pentru $coef_{fwd}$ şi $coef_{drop}$ sunt:

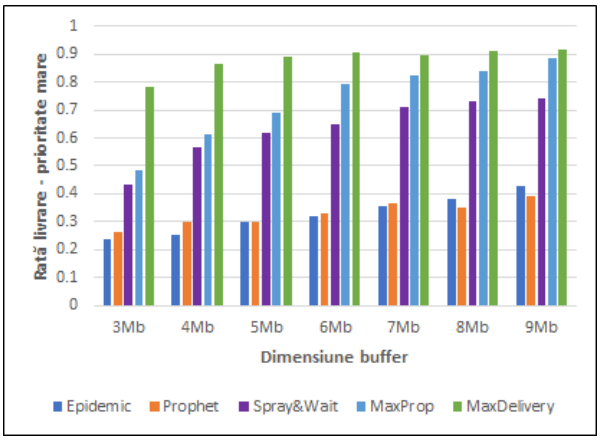
$$coef_{fwd}(msg) = \begin{cases} 5, & \text{dacă msg are prioritate ridicată} \\ 1, & \text{dacă msg are prioritate scăzută} \end{cases}$$

$$coef_{drop}(msg) = \begin{cases} 1, & \text{dacă msg are prioritate ridicată} \\ 5, & \text{dacă msg are prioritate scăzută} \end{cases}$$

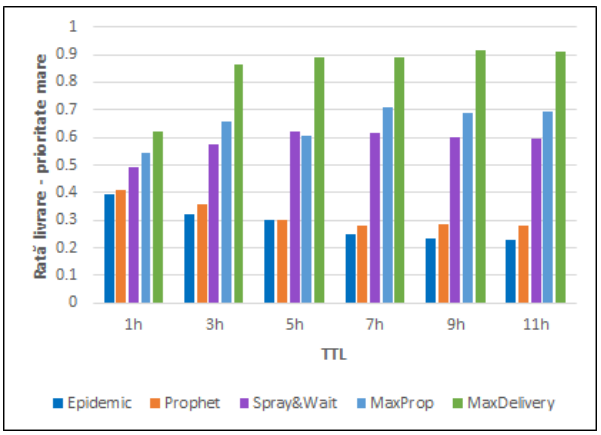
După adăugarea celor două categorii de priorităţi ale mesajelor, numărul de mesaje cu prioritate mare care au fost generate este aproximativ egal cu cel al mesajelor de prioritate mică, cu diferenţe între 1 şi 60 de mesaje, la un total de 1500. De asemenea, datorită modificărilor asupra algoritmului de rutare propus, valorile globale pentru timpul de livrare al mesajelor s-a îmbunătăţit.

Varianta propusă pentru algoritmul MaxDelivery reuşeşte să obţină un procent mult mai mare de livrare al mesajelor cu prioritate ridicată faţă de ceilalţi algoritmi implementaţi în simulatorul ONE.

Eficienţa ratei de livrare a mesajelor cu prioritate mare, cele care prezintă interes în acest context, sunt detaliate în Fig. 3.10.



(a) Variaţia dimensiunii bufferului

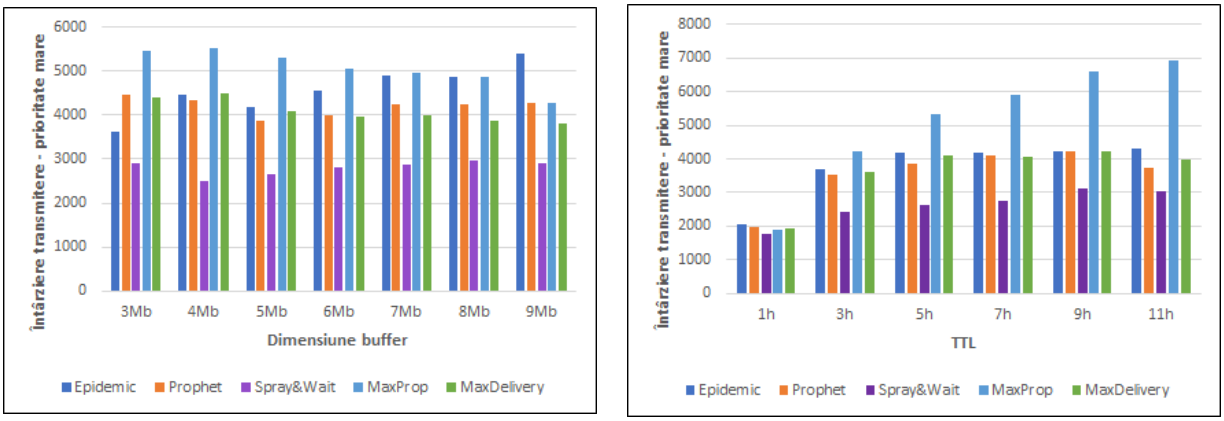


(b) Variaţia timpului de viaţă al mesajelor

Figura 3.10: Eficienţa ratei de livrare a mesajelor cu prioritate ridicată

În Fig. 3.10b, rezultatele obţinute de algoritmul propus sunt cu aproximativ 20% mai bune decât cele oferite de algoritmul MaxProp, care are cele mai bune valori dintre algoritmii oferiţi de simulator.

Timpul de livrare al mesajelor s-a îmbunătăţit, nu numai la nivel global, ci şi în cazul mesajelor cu prioritate mare. Rezultatele pentru această metrică de performanţă a reţelei se pot observa în imaginile din Fig. 3.11.



(a) Variaţia dimensiunii bufferului

(b) Variaţia timpului de viaţă al mesajelor

Figura 3.11: Eficienţa timpului de livrare al mesajelor cu prioritate ridicată

3.5 Scenariul concret de aplicare al algoritmului MaxDelivery

Conform unui raport al Organizaţiei Naţiunilor Unite (ONU) publicat în 2018, dezastrele naturale, cum ar fi inundaţiile, cutremurele, uraganele, aduc pierderi de până la 300 miliarde de dolari anual, iar numărul de persoane afectate de către acestea în perioada 1998-2017, a fost detaliat în lucrarea [80].

Dezastrele naturale au fost clasificate, din punctul de vedere al fenomenului care le declanşează, în dezastre hidrologice, meteorologice, climatologice şi geografice. Un dezastru are loc atunci când un fenomen natural afectează comunităţi vulnerabile de persoane.

În Tab. 3.4 putem observa datele statistice culese de către ONU în urma studiului impactului acestor fenomene asupra umanităţii, între anii 1998 şi 2017.

În ultimii ani, unele dintre cele mai violente dezastre apărute au fost cutremurul din Indonezia, urmat de tsunami (2004), cutremurul din Haiti (2010) şi ciclonul Nargis din Myanmar (2008). La această listă putem adăuga alte evenimente petrecute în perioada 2018-2020, cum ar fi incendiile de vegetaţie din Grecia (2018) şi din Australia (începutul anului 2020), pe fondul valului de căldură din ultimii ani, precum şi pandemia de coronavirus din anul în curs.

Din datele furnizate în [80] se poate observa că fenomenele naturale care produc cele mai mari pierderi de vieţi omeneşti sunt cutremurele, urmate de furtuni şi temperaturi extreme. De cele mai multe ori, în astfel de situaţii, infrastructura de telecomunicaţii existentă este afectată serios.

Fenomen	**Populaţie afectată**	**Decese**
Inundaţii	2 miliarde (45%)	≈142.000 (11%)
Secetă	1.5 miliarde (33%)	≈21.500 (2%)
Furtuni\Uragane	726 milioane (16%)	≈32.000 (17%)
Cutremure	125 milioane (3%)	≈750.000 (56%)
Temperaturi extreme	97 milioane (2%)	≈166.000 (13%)
Alunecări de teren	4.8 milioane (≈0.1%)	≈18.000 (1%)
Incendii şi erupţii vulcanice	6.2 milioane (≈0.1%)	≈2.400 (0.2%)

Tabela 3.4: Impactul dezastrelor naturale asupra populaţiei

Într-o situaţie de urgenţă care apare după un cutremur, este nevoie de coordonare şi comunicare pentru a putea ajuta eficient victimele apărute. Pentru a suplini reţeaua de comunicare tradiţională, afectată de cutremur, este necesară implementarea unei reţele DTN, deoarece nu necesită o conexiune continuă între toate nodurile reţelei.

Provocări post-dezastru, în cazul unui cutremur:

- identificarea zonelor afectate şi delimitarea acestora
- evacuarea răniţilor şi a victimelor şi acordarea primului ajutor
- posibilităţile limitate de acces, datorate blocării drumurilor
- dificultăţile în comunicare, datorate reţelelor avariate (telecomunicaţii, energie electrică etc.)
- coordonarea operaţiunilor de transport al victimelor la spital sau la centrele de adăpost

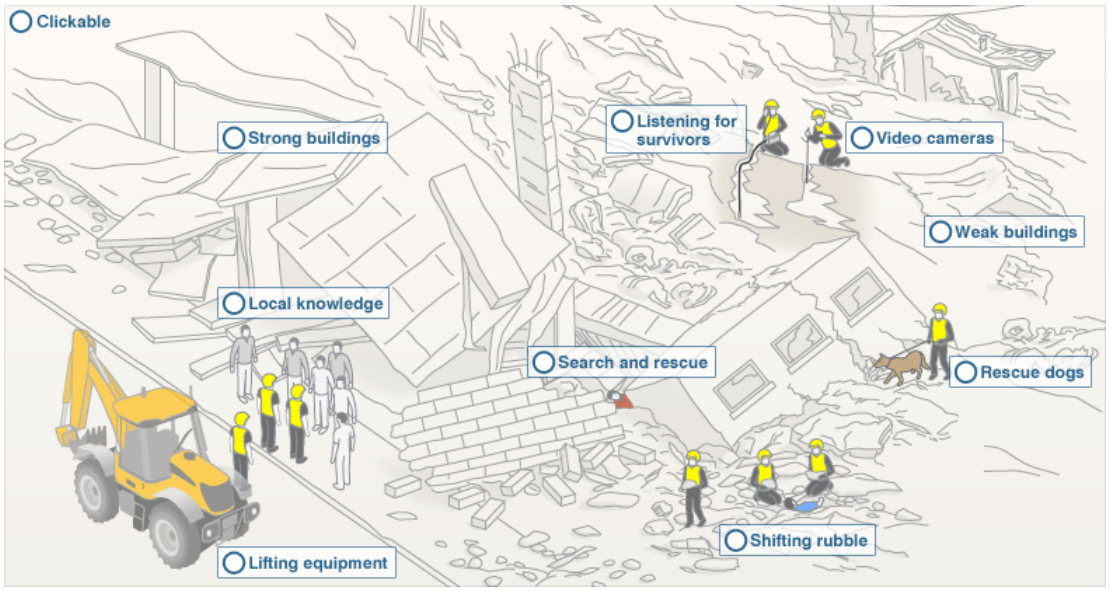


Figura 3.12: Mobilizare în caz de cutremur (din [92])

În reţeaua DTN utilizată în aceste condiţii, se găsesc mai multe tipuri de actori: victime, salvatori, spitale, ambulanţe, adăposturi de siguranţă şi un centru de comandă. Aceste tipuri de actori vor reprezenta nodurile reţelei DTN care simulează contextul propus. Spitalele, adăposturile şi centrul de comandă sunt noduri fixe, iar salvatorii (căutătorii de victime) şi ambulanţele sunt noduri mobile.

Astfel, salvatorii se deplasează cu o viteză medie între 1 km/h şi 5 km/h, iar timpul de staţionare în cazul găsirii unei victime este până la 15 minute. Modelul de mobilitate al acestora presupune identificarea drumului celui mai scurt între locaţia curentă şi locaţia unde se doreşte deplasarea.

Ambulanţele au o viteză medie de deplasare între 30 km/h şi 65 km/h, cu un timp de staţionare de 2 până la 5 minute pentru urcarea / coborârea victimei. La începutul simulării, acestea au punctul de start lângă un spital, iar ulterior se vor deplasa între un spital şi o poziţie de pe hartă, corespunzătoare unei victime.

Dispozitivele care alcătuiesc reţeaua sunt distribuite pe ambulanţe, în spitale şi salvatorilor care caută victimele şi vor folosi protocolul de rutare MaxDelivery, propus în această lucrare.

Deoarece într-o situaţie de cutremur este necesară trierea mesajelor, vom clasifica mesajele astfel:

1. mesaje cu prioritate ridicată - acele mesaje venite din partea salvatorilor, care solicită intervenţia unei ambulanţe, mesaje care transmit coordonatele de localizare a unei victime, mesaje care transmit numărul de locuri disponibile într-un anumit spital sau într-un anumit adăpost de siguranţă etc.

2. mesaje cu prioritate redusă - mesaje care transmit coordonatele de localizare a unei persoane decedate (se presupune că importanţa localizării acesteia este mai mică decât cea a localizării unei persoane aflate încă în viaţă), mesaje cu informaţii statistice solicitate de centrul de comandă: numărul de persoane decedate, numărul de răniţi etc.

Frecvenţa cu care se generează mesajele diferă în funcţie de categoria de actori implicaţi în sistem. Din grupul căutătorilor de victime se va genera în medie câte un mesaj pe minut, ambulanţele vor genera în medie câte un mesaj la 1-2 minute, iar nodurile fixe vor genera câte un mesaj la 30-60 minute.

Pentru verificarea ratei de transfer a mesajelor s-au efectuat simulări, variind numărul de noduri mobile ale sistemului. Am presupus că în zona afectată pot opera 5 spitale, 10 pucte de adăpost al populaţiei şi un centru de comandă. Au fost făcute astfel 6 simulări a câte 24 de ore, în care numărul de ambulanţe şi de salvatori au variat astfel:

1. 5 ambulanţe şi 40 de salvatori

2. 10 ambulanţe şi 50 de salvatori

3. 15 ambulanţe şi 60 de salvatori

4. 20 ambulanţe şi 80 de salvatori

5. 25 ambulanţe şi 100 de salvatori

6. 30 ambulanţe şi 120 de salvatori

Harta utilizată pentru efectuarea simulărilor este tot harta oraşului Helsinki, propusă în simulatorul ONE, timpul de viaţă al mesajelor a fost considerat de 5 ore, iar dimensiunea bufferului de 5MB.

Au fost obţinute rezultate cu un procent foarte mare de mesaje cu prioritate ridicată care ajung să fie livrate destinatiei, în acest context. Valorile obţinute ajung pana la 96%.

Odată cu creşterea numărului de salvatori şi de ambulanţe, creşte şi procentul de livrare al mesajelor. Acest fapt se datorează mobilităţii ridicate şi întâlnirilor frecvente dintre noduri.

Reţeaua descrisă se poate extinde cu un sistem de drone care să ajute la răspândirea mesajelor. Numărul şi deplasarea acestora, pentru optimizarea acoperirii suprafeţei monitorizate, va fi considerată conform celor descrise în lucrarea [17], ale cărei rezultate aparţin parţial autoarei acestei teze. Pentru acoperirea eficientă a unei suprafeţe afectată de cutremur, izolată sau carantinată, se va considera o reţea echidistantă de triunghiuri echilaterale grupate în hexagoane regulate, iar deplasarea dronelor se face pe laturile triunghiurilor echilaterale, în sensul acelor de ceasornic. Pentru alimentarea cu energie electrică se vor utiliza puncte de schimbare a bateriei, un astfel de punct fiind comun la 3 drone, similar cu cele prezentate în [17]. Pentru o trimitere eficientă de mesaje, subreţeaua de drone va calcula drumul cel mai scurt al mesajelor la destinaţie pe baza unui algoritm derivat din algoritmul lui Dijkstra. Varianta propusă pentru algoritmul lui Dijkstra este una dependentă de timp, care ia în calcul momentul sosirii şi momentul plecării dronei într-un / dintr-un nod fix şi timpul de zbor între două noduri fixe. De asemenea, algoritmul propus pentru funcţionarea dronelor utilizează o coadă de priorităţi care ţine mesajele sortate în funcţie de distanţa parcursă până în prezent.

O altă variantă, care ar putea funcţiona cu succes şi cu rezultate optime de livrare a mesajelor ar putea fi aceea în care atât nodurile de la sol, cât şi extensia formată din drone, utilizează algoritmul MaxDelivery pentru rutarea pachetelor de date.

Capitolul 4

Concluzii şi cercetări viitoare

4.1 Concluzii

În lucrarea de faţă am abordat două situaţii care se pot întâlni într-o reţea cu toleranţă la întârzieri:

- problema de identificare a unui flux maxim
- problema de maximizare a procentului de transmitere a mesajelor

La baza problemelor expuse anterior se află mecanismul de gestiune a bufferului, care eficientizează fluxul de date din interiorul nodului, dar şi din întreaga reţea. Politica gestionării spaţiului de stocare are ca suport teoretic un sistem de aşteptare. Astfel, fiecare nod al unei reţele DTN se poate reprezenta ca un sistem de aşteptare individual. În capitolul 2 au fost prezentate două situaţii a unor sisteme de aşteptare cu revenire: cea în care clienţii sosesc individual şi cea în care sosirile se fac în loturi.

Rezolvarea celor două probleme este realizată în capitolul 3.

În secţiunea 3.1 este prezentată o abordare statică pentru identificarea fluxului maxim, pornind de la o reţea cu toleranţă la întârzieri, reprezentată sub forma unei reţele dinamice.

În secţiunea 3.2 sunt descrise cercetările preliminarii care au condus la realizarea noului algoritm de rutare, MaxDelivery. Acestea sunt bazate pe o nouă politică de abandonare a mesajelor în cazul congestionării bufferului.

În secţiunea 3.3 este descrisă modalitatea de funcţionare a algoritmului MaxDelivery, scoţând în evidenţă politica de gestiune a bufferului pe care o implementează. Au fost prezentate de asemenea rezultatele simulărilor efectuate pentru a testa eficienţa algoritmului în comparaţie cu unii dintre cei mai cunoscuţi algoritmi de rutare în DTN.

Secţiunea 3.4 prezintă o variantă pentru algoritmul propus, care tratează problema optimizării transmiterii mesajelor care au prioritate ridicată.

Secţiunea 3.5 prezintă scenariul concret al unei intervenţii post-cutremur, în care se poate aplica algoritmul MaxDelivery.

Rezultatele originale care aparţin în totalitate sau parţial autoarei, şi care stau la baza realizării acestei teze, sunt următoarele:

1. o privire de ansamblu asupra reţelelor cu toleranţă la întârzieri şi o scurtă trecere în revistă a protocoalelor de rutare, completate de o serie de remarci personale, a fost publicată în lucrarea: **Nănău C.Ş.**, An overview of Delay Tolerant Networks and routing protocols, Bulletin of the Transilvania University of Braşov, Vol. 11(60), no. 2, pp. 279-284, 2018.

2. exemplificarea celor mai cunoscute protocoale de rutare a fost publicată în lucrarea: **Nănău C.Ş.**, Example of routing protocols in Delay Tolerant Networks, Bulletin of the Transilvania University of Braşov, Vol. 12(61), no. 1, pp. 145-156, 2019.

3. o abordare algoritmică a sistemelor de aşteptare cu revenire, care au o singură staţie de servire, iar clienţii sosesc individual în sistem, a fost publicată în lucrarea: Florea, I.L. şi **Nănău C.Ş.**, An algorithmic approach of retrial queuing system with one serving station. Part I: The description of the simulation algorithm, Bulletin of the Transilvania University of Braşov, Vol. 6(55), no. 2, pp. 95-106, 2013. Aceasta este una dintre lucrările care stau la baza capitolului 2 al tezei.

4. implementarea algoritmilor descrişi în lucrarea precedentă, împreună cu prezentarea factorilor de eficienţă ai sistemului, atât din punct de vedere analitic, cât şi pe baza simulărilor, se regăsesc în lucrarea: Florea, I.L. şi **Nănău C.Ş.**, An algorithmic approach of retrial queuing system with one serving station. Part II: The implementation of the simulation algorithm, Bulletin of the Transilvania University of Braşov, Vol. 7(56), no. 2, pp. 183-192, 2014.

5. implementarea unui sistem de aşteptare cu revenire, care are o singură staţie de servire, iar clienţii sosesc în loturi în sistem, a fost publicată în lucrarea: Florea, I.L. şi **Nănău C.Ş.**, A simulation algorithm for a single server retrial queuing system with batch arrivals, Analele ştiinţifice ale universităţii Ovidius Constanţa, Vol. 23, no. 1, pp. 83-98, 2015 - **revistă indexată ISI, cu Factor de impact 0.638 şi SRI 0.232**. Această abordare este evidenţiată în capitolul 2 al tezei.

6. cercetările preliminarii care au condus la dezvoltarea algoritmului MaxDelivery sunt prezentate în lucrarea: **Nănău, C.Ş.**, Queuing Theory Application on DTN Buffer Management, in Proceeding of the 8th International Conference on Computers Communications and Control (ICCCC 2020), Oradea, 2020. Rezultatele din această lucrare se regăsesc în secţiunea 3.2.

7. calculul fluxului maxim într-o reţea de tip DTN, recurgând la o abordare statică, stă la baza secţiunii 3.1 şi a fost publicat în lucrarea: **Nănău C.Ş.**, Maximum flow in buffer-limited Delay Tolerant Networks. The static approach, Bulletin of the Transilvania University of Braşov, Vol. 14(63), no. 1, 2020.

Rezultate originale acceptate la conferinţe internaţionale:

1. descrierea funcţionalităţii algoritmului MaxDelivery şi a politicii de gestiune a bufferului pe care o implementează acesta, care se regăseşte în secţiunea 3.3, este prezentată în lucrarea: **Nănău, C.Ş.**, MaxDelivery: a new approach to a DTN buffer management, trimisă pentru recenzie la conferinţa 21st IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WOWMOM 2020), Cork, Ireland.

Rezultate originale trimise spre publicare:

1. determinarea drumului minim parcurs de mesaje într-o reţea DTN, reprezentată de un sistem echidistant de triunghiuri echilaterale grupate în hexagoane regulate şi respectiv, pătrate, ale cărei noduri sunt reprezentate de drone, este descrisă în lucrarea: Deaconu, A., Udroiu, R. and **Năănău, C.Ş.**, Data or Physical Packages Delivery in Isolated, Disaster or Quarantined Areas Using DTN Based Algorithms for Unmanned Aerial Vehicles, trimisă pentru recenzie la revista IEEE Access, **cotată ISI, cu Factor de impact 3.745 şi SRI 0.642**. Rezultatele obţinute sunt utilizate în secţiunea 3.5, unde este prezentat exemplul de aplicaţie al reţelei DTN.

4.2 Cercetări viitoare

Una dintre direcţiile viitoare de cercetare este aceea de a îmbunătăţi eficienţa transmiterii mesajelor în algoritmul de rutare propus, prin completarea sa cu un algoritm de flux maxim. Astfel, oportunităţile de conectare dintre noduri ar putea fi folosite într-un mod optim pentru livrarea mesajelor.

O altă problemă de actualitate în domeniul reţelelor ad-hoc, este cea referitoare la reţeaua de drone, cunoscută sub denumirea de FANET (Flying ad-hoc Network în limba engleză). Acest concept este unul foarte recent, studiat tot mai mult în ultimii ani.

Posibilităţi de aplicare a unei reţele de tip FANET:

- monitorizarea suprafeţelor cultivate şi a pădurilor
- operaţiuni de căutare şi salvare de persoane
- livrare de produse
- monitorizarea traficului
- observarea zonelor post dezastru

Schimbarea frecventă a topologiei reţelei, mediul diferit de aplicare, factorii climatici şi consumul de energie sunt factori care transformă FANET într-o provocare. Pentru a obţine performanţele dorite, este necesar ca protocoalele de rutare implementate să ţină cont de aceste aspecte.

Datorită faptului că obiectivul unei reţele FANET este acela de a descoperi rute optime de parcurgere, în cercetările viitoare voi studia aplicarea paradigmei DTN în reţelele FANET.

Bibliografie

[1] Ahuja, R., Magnanti, T. and Orlin, J., **Network Flows. Theory, algorithms and applications**, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[2] Arrar, N.K., Djellab, N.V. and Baillon J.B., **On the asymptotic behaviour of M/G/1 retrial queues with batch arrivals and impatience phenomenon**, Mathematical and Computer Modelling, Vol. 55, Issues 3-4, pp. 654-665, 2012, doi: 10.1016/j.mcm.2011.08.039.

[3] Ayub, Q. and Rashid, S., **T-drop: an optimal buffer management policy to improve QOS in DTN routing protocols**, Journal of Computing, Vol. 2, no. 10, pp. 46-50, 2010.

[4] Ayub, Q., Ngadi, A., Rashid, S. and Habib, H.A., **Priority Queue Based Reactive Buffer Management Policy for Delay Tolerant Network under City Based Environments**, PloS One, Vol. 13, no. 2, 2018, doi: 10.1371/journal.pone.0191580.

[5] Balasubramanian, A., Levine, B.N. and Venkataramani, A., **DTN routing as a resource allocation problem**, ACM SIGCOMM, Vol. 37, no. 4, pp. 373-384, 2007, doi: 10.1145/1282380.1282422.

[6] Bertsekas, D.P. and Gallager, R., **Data Networks**, Cap.3, Prentice Hall, 1991, ISBN 0132009161.

[7] Bjurefors, F., Gunningberg, P., Rohner, C. and Tavakoli, S., **Congestion avoidance in a data-centric opportunistic network**, Proceedings of the ACM SIGCOMM workshop on Information-centric networking, Toronto, pp. 32-37, 2011, doi: 10.1145/2018584.2018594.

[8] Boldrini, C., Conti, M. and Passarella, A., **Less is More: Long Paths do not Help the Convergence of Social-Oblivious Forwarding in Opportunistic Networks**, Proceedings of the ACM/SIGMOBILE MobiOpp, pp. 13-20, 2012, doi: 10.1145/2159576.2159582.

[9] Boudguig, M. and Abdali, A., **New DTN Routing Algorithm**, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 2, no. 3, pp. 82-87, 2013.

[10] Bulut, E. and Szymanski, B.K., **Friendship Based Routing in Delay Tolerant Mobile Social Networks**, IEEE Global Telecommunications Conference GLOBECOM, 2010, doi: 10.1109/GLOCOM.2010.5683082.

[11] Burgess, J., Gallagher, B., Jensen, D. and Levine, B.N., **MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks**, IEEE INFOCOM, pp. 1-11, 2006, doi: 10.1109/INFOCOM.2006.228.

[12] Cai, X., Sha, D. and Wong, C., **Time-varying Network Optimization**, Springer, 2007.

[13] Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K. and Weiss, H., **Delay-Tolerant Networking Architecture**, Network Working Group, `https://tools.ietf.org/html/rfc4838`, April 2007.

[14] Ciurea, E., **Counterexamples in maximal dynamic flows**, Annual Congress of the American Romanian Academy of Arts and Sciences, Libertas Mathematica, Vol. XVII, pp. 77-97, 1997.

[15] Daly, E.M. and Haahr, M., **Social network analysis for routing in disconnected delay-tolerant MANETs**, MobiHoc '07: Proceeding of the 8th ACM international symposium on Mobile ad hoc networking and computing, pp. 32-40, 2007, doi: 10.1145/1288107.1288113.

[16] Davis, J.A., Fagg, A.H. and Levine, B.N., **Wearable computers as packet transport mechanisms in highly partitioned ad-hoc networks**, Proceedings of Fifth International Symposium on Wearable Computers, Zurich, pp. 141-148, 2001, doi: 10.1109/ISWC.2001.962117.

[17] Deaconu, A., Udroiu, R. and ***Nănău, C.Ş.*, Data or Physical Packages Delivery in Isolated, Disaster or Quarantined Areas Using DTN Based Algorithms for Unmanned Aerial Vehicles**, IEEE Access, 2020, trimisă spre publicare.

[18] Dijkstra, E.W., **A note on two problems in connexion with graphs**, Numerische Mathematik, Vol. 1, no. 1, pp. 269-271, 1959.

[19] Dziekonski, A.M. and Schoeneich, R.O., **DTN Routing Algorithm for Networks with Nodes Social Behavior**, International Journal of Computers Communications & Control, Vol. 11, no. 4, pp. 457-471, 2016, doi: 10.15837/ijccc.2016.4.1454.

[20] Falin, G.I., **A single-server batch arrival queue with returning customers**, European Journal of Operational Research, Vol. 201, no. 3, pp. 786-790, 2010, doi: 10.1016/j.ejor.2009.03.033.

[21] Fall, K., **A Delay-Tolerant Network Architecture for Challenged Internets**, SIGCOMM, Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 27-34, 2003, doi: 10.1145/863955.863960.

[22] Fathima, G. and Wahidabanu, R.S.D., **Buffer management for preferential delivery in opportunistic Delay Tolerant Networks**, International Journal of Wireless & Mobile Networks, Vol. 3, no. 5, pp. 15-28, 2011, doi: 10.5121/ijwmn.2011.3502.

[23] Florea, I.L. and ***Nănău C.Ş.*, An algorithmic approach of retrial queuing system with one serving station. Part I: The description of the simulation algorithm**, Bulletin of the Transilvania University of Braşov, Vol. 6(55), no. 2, pp. 95-106, 2013.

[24] Florea, I.L. and ***Nănău C.Ş.***, **An algorithmic approach of retrial queuing system with one serving station. Part II: The implementation of the simulation algorithm**, Bulletin of the Transilvania University of Braşov, Vol. 7(56), no. 2, pp. 183-192, 2014.

[25] Florea, I.L. and ***Nănău C.Ş.***, **A simulation algorithm for a single server retrial queuing system with batch arrivals**, Analele ştiinţifice ale universităţii Ovidius Constanţa, Vol. 23, no. 1, pp. 83-98, 2015, doi: 10.1515/auom-2015-0007.

[26] Ford, L.R. and Fulkerson, D.R., **Flows in Networks**, Princeton University Press, Princeton New Jersey, 1962.

[27] Giuseppe, A., Bezirgiannidis, N., Birrane, E., Bisio, I., Burleigh, S., Caini, C., Feldmann, M., Marchese, M., Segui, J. and Suzuki, K., **Contact Graph Routing in DTN Space Networks: Overview, Enhancements and Performance**, IEEE Communications Magazine, Vol. 53, no. 3, pp. 38–46, 2015, doi: 10.1109/MCOM.2015.7060480.

[28] Gorunescu, F. and Prodan, A., **Modelare stochastică şi simulare**, Editura Albastră, Cluj-Napoca, ISBN 973-650-023-3, 2001.

[29] Holme, P. and Saramäki, J., **Temporal Networks**, Physics Reports, Vol. 519, no. 3, pp. 97-125, 2012.

[30] Hui, P. and Crowcroft, J., **How small labels create big improvements**, Procedeengs of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07), pp. 65-70, 2007, doi: 10.1109/PERCOMW.2007.55.

[31] Hui, P., Crowcroft, J. and Yoneki, E., **BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks**, Mobile Computing, IEEE Transactions, Vol. 10, no. 11, pp. 1576-1589, 2011, doi: 10.1109/TMC.2010.246.

[32] Jain, S., Fall, K. and Patra, R., **Routing in a Delay Tolerant Network**, SIGCOMM '04, Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications, pp. 145-158, 2004, doi: 10.1145/1015467.1015484.

[33] Jain, S. and Chawla, M., **Survey of Buffer Management Policies for Delay Tolerant Networks**, The Journal of Engineering, Vol. 2014, Issue 3, pp. 117-123, 2014, doi: 10.1049/joe.2014.0067.

[34] Jones, E.P.C., **Practical Routing in Delay-Tolerant Networks**, Masters Thesis, Waterloo, Ontario, Canada, 2006.

[35] Ke, M., Nenghai, Y. and Bin L., **A new packet dropping policy in Delay Tolerant Network**, IEEE 12th International Conference on Communication Technology, Nanjing, pp. 377-380, 2010, doi: 10.1109/ICCT.2010.5689151.

[36] Keranen, A., Ott, J. and Karkkainen, T., **The ONE simulator for DTN protocol evaluation**, Proceedings of the 2nd International Conference on Simulation Tools and Techniques (SimuTools 2009), Rome, Italy, article no. 55, pp. 1-10, 2009, doi: 10.4108/ICST.SIMUTOOLS2009.5674.

[37] Keranen, A., **Opportunistic Network Environment Simulator**, Special Assignment report, Helsinki University of Technology, Department of Communications and Networking, May 2008.

[38] Krifa, A., Barakat, C., Spyropoulos, T., **Optimal buffer management policies for delay tolerant networks**, IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2008), pp. 1-9, 2008, doi: 10.1109/SAHCN.2008.40.

[39] Krishna, K.B. and Arivudainambi, D., **The M/G/1 retrial queue with Bernoulli schedules and general retrial times**, Computers and Mathematics with Applications, Vol. 43, Issues 1-2, pp. 15-30, 2002, doi: 10.1016/S0898-1221(01)00267-X.

[40] Lee, K., Hong, S., Kim, S.J., Rhee, I. and Chong, S., **SLAW: Self-Similar Least-Action Human Walk**, IEEE, Vol. 20, no. 2, pp. 515-529, 2012, doi: 10.1109/TNET.2011.2172984.

[41] Li, Y. and Qian, M., **Adaptive optimal buffer management policies for realistic DTN**, IEEE Global Telecommunication Conference, GLOBECOM 2009, USA, pp. 1-5, 2009, doi: 10.1109/GLOCOM.2009.5426161.

[42] Li, Y., Zhao, L., Liu, Z. and Liu, Q., **N-drop congestion control strategy under epidemic routing in DTN**, Proceedings of the International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly (IWCMC 2009), Leipzig, Germany, pp. 457-460, 2009, doi: 10.1145/1582379.1582479.

[43] Li, H., Zhang, T., Zhang, Y., Wang, K. and Li, J., **A maximum flow algorithm based on storage time aggregated graph for Delay Tolerant Networks**, Ad Hoc Networks, Vol. 59, pp. 63-70, 2017, doi: 10.1016/j.adhoc.2017.01.006.

[44] Lindgren, A., Doria, A. and Schelen, O., **Probabilistic routing in intermittently connected networks**, SIGMOBILE Mobile Computing and Communication, 2003, doi: 10.1145/961268.961272.

[45] Lindgren, A. and Phanse, K.S., **Evaluation of queuing policies and forwarding strategies for routing in intermittently connected networks**, Proceedings of International Conference on Communication Systems Software & Middleware, pp. 1-10, 2006, doi: 10.1109/COMSWA.2006.1665196.

[46] Liu, Y., Wang, J., Zhang, S. and Zhou, H., **A buffer management scheme based on message transmission status in Delay Tolerant Networks**, IEEE Global Telecommunications Conference (GLOBECOM 2011), pp. 1-5, 2011, doi: 10.1109/GLOCOM.2011.6134084.

[47] Lo, S.C., Tsai, C.C. and Lai Y.H., **Quota-Control Routing in Delay-Tolerant Networks**, Ad Hoc Networks, Vol. 25(B), pp. 393-405, 2015, doi: 10.1016/j.adhoc.2014.07.029.

[48] Mauve, M., Widmer, A. and Hartenstein, H., **A survey on position-based routing in mobile ad hoc networks**, IEEE Network, Vol. 15, no. 6, pp. 30–39, 2001, doi: 10.1109/65.967595.

[49] Mehta, N. and Shah, M., **Performance of Efficient Routing Protocol in Delay Tolerant Network: A Comparative Survey**, International Journal of Future Generation Communication and Networking, Vol. 7, no. 1, pp. 151-158, 2014, doi: 10.14257/ijfgcn.2014.7.1.15.

[50] Mei, A., Morabito G., Santi, P. and Stefa, J., **Social-aware stateless forwarding in pocket switched networks**, in Proceedings of the IEEE INFOCOM, pp. 251-255, 2011, doi: 10.1109/INFCOM.2011.5935076.

[51] Nain, D., Petigara, N. and Balakrishnan, H., **Integrated routing and storage for messaging applications in mobile ad hoc networks**, Mobile Networks and Applications (MONET), Vol. 9, no. 6, pp. 595–604, 2004, doi: 10.1023/B:MONE.0000042498.60917.e8.

[52] ***Năn̆au C.Ş.*, An overview of Delay Tolerant Networks and routing protocols**, Bulletin of the Transilvania University of Braşov, Vol. 11(60), no. 2, pp. 279-284, 2018.

[53] ***Năn̆au C.Ş.*, Example of routing protocols in Delay Tolerant Networks**, Bulletin of the Transilvania University of Braşov, Vol. 12(61), no. 1, pp. 145-156, 2019.

[54] ***Năn̆au C.Ş.*, Maximum flow in buffer-limited Delay Tolerant Networks. The static approach**, Bulletin of the Transilvania University of Braşov, Vol. 13(62), no. 1, pp. 363-372, 2020.

[55] ***Năn̆au, C.Ş.*, Queuing Theory Application on DTN Buffer Management**, in Proceeding of the 8th International Conference on Computers Communications and Control (ICCCC 2020), Oradea, 2020.

[56] ***Năn̆au, C.Ş.*, MaxDelivery: a new approach to a DTN Buffer Management**, Proceeding of 21ST IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WOWMOM 2020), Cork, Ireland, acceptată pentru susţinere.

[57] Nelson, S.C., Bakht, M. and Kravets, R., **Encounter-Based Routing in DTNs**, Proceeding of IEEE INFOCOM 2009 Conference, Rio de Janeiro, pp. 846-854, 2009, doi: 10.1109/INFCOM.2009.5061994.

[58] Ng, T.S.E. and Zhang, H., **Predicting internet network distance with coordinates based approaches**, Proceedings of IEEE INFOCOM, Vol. 1, pp. 170–179, 2002, doi: 10.1109/INFCOM.2002.1019258.

[59] Patel, Y. and Patel, T., **A Survey on DTN Routing Protocols**, International Journal of Engineering Development and Research (IJEDR), Vol. 3, no. 4, 2015.

[60] Ramanathan, R., Hansen, R., Basu, P., Rosales-Hain, R. and Krishnan, R., **Prioritized epidemic routing for opportunistic networks**, Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking, pp. 62-66, 2007, doi: 10.1145/1247694.1247707.

[61] Rashid, S. and Ayub, Q., **Effective buffer management policy DLA for DTN routing protocols under congestion**, International Journal of Computer and Network Security, Vol. 2, no. 9, pp.118-121, 2010.

[62] Rashid, S., Ayub, Q., Soperi, M.M.Z. and Abdullah, A.H., **E-Drop: an effective drop buffer management policy for DTN routing protocols**, International Journal of Computer Applications, Vol. 13, no. 7, pp. 8-13, 2011, doi: 10.5120/1795-2486.

[63] Rashid S., Abdullah A.H., Soperi, M.M.Z. and Ayub, Q., **Mean drop an effectual buffer management policy for Delay Tolerant Network**, European Journal Of Scientific Research, Vol. 70, no. 3, pp. 396-407, 2012.

[64] Scott, K. and Burleigh, S., **Bundle Protocol Specification**, Network Working Group, `https://tools.ietf.org/html/rfc5050`, November 2007.

[65] Sehgal, R., **Modeling and Analysis of Delay Tolerant Networks**, Dissertation Thesis, 2016.

[66] Settawatcharawanit, T., Yamada, S., Haque, E. and Rojviboonchai, K., **Message dropping policy in congested social Delay Tolerant Networks**, The 10th International Joint Conference on Computer Science and Software Engineering (JCSSE), pp. 116-120, 2013, doi: 10.1109/JCSSE.2013.6567330.

[67] Shah, M., Gondaliya, N. and Barad, A., **An Improved SimBet Routing Algorithm for Human Mobility based DTN**, Kalpa Publications in Computing, Vol. 2, pp. 166-176, 2017, doi: 10.29007/c28h.

[68] Shin, K. and Kim, S., **Enhanced buffer management policy that utilizes message properties for Delay Tolerant Networks**, IET Communications, Vol. 5, no. 6, pp. 753-759, 2011, doi: 10.1049/iet-com.2010.0422.

[69] Skutella, M.G., **An Introduction to Network Flows Over Time**, Research Trends in Combinatorial Optimization, pp. 451-482, 2008, doi: 10.1007/978-3-540-76796-1_21.

[70] Small, T and Haas, Z.J., **Resource and performance tradeoffs in delay-tolerant wireless networks**, Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05), pp. 260–267, 2005, doi: 10.1145/1080139.1080144.

[71] Sobin, C.C., **An Efficent Buffer Management Policy for DTN**, Procedia Computer Science, Vol. 93, pp. 309-314, 2016, doi: 10.1016/j.procs.2016.07.215.

[72] Soares, V.N.G.J., Farahmand, S.F. and Rodrigues, J.J.P.C., **Performance analysis of scheduling and dropping policies in vehicular delay-tolerant networks**, International Journal on Advances in Internet Technology - IARIA, Vol. 3, no. 1, pp. 137-145, 2010.

[73] Souza, C., Mota, E., Galvao, L., Manzoni, P. and Cano, J.C., **Drop Less Known strategy for buffer management in DTN Nodes**, Proceedings of the Latin America Networking Conference on LANC 2014, Article no. 6, pp. 1-7, 2014, doi: 10.1145/2684083.2684089.

[74] Spyropoulos, T., Psounis, K. and Raghavendra, C.S., **Single-copy routing in intermittently connected mobile networks**, Proceedings of Sensor and Ad Hoc Communications and Networks SECON, pp. 235-244, 2004, doi: 10.1109/SAHCN.2004.1381922.

[75] Spyropoulos, T., Psounis, K. and Raghavendra, C.S. , **Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks**, Proceedings of the ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05), pp. 252–259, 2005, doi: 10.1145/1080139.1080143.

[76] Stewart, M.F., **CASPaR: Congestion Avoidance Shortest Path Routing for Delay Tolerant Networks**, Louisiana State University, Agricultural and Mechanical College, The Department of Computer Science, 2015.

[77] Tang, L., Chai, Y., Li, Y. and Weng, B., **Buffer management policies in opportunistic networks**, Journal of Computational Information Systems, Vol. 8, no. 12, pp. 5149-5159, 2012.

[78] Toh, C.K., **Ad Hoc Mobile Wireless Networks**, Prentice Hall Publishers, 2002, ISBN 978-0-13-007817-9

[79] Vahdat, A. and Becker, D., **Epidemic routing for partially-connected ad hoc networks**, Handbook of Systemic Autoimmune Diseases, Duke University, Tech. Rep. CS-2000-06, July 2000.

[80] Wallemacq, P. and House, R., **Economic losses, poverty & disasters: 1998-2017**, United Nations Office for Disaster Risk Reduction, Technical Report, 2018, doi: 10.13140/RG.2.2.35610.08643.

[81] Warthman, F., **Delay-and Disruption-Tolerant Networks (DTNs)**, Warthman Associates, Version 3.2, September 14, 2015.

[82] Yamamuro K., **The queue length in an M/G/1 batch arrival retrial queue**, Queueing Systems, Vol. 40, pp. 187-205, 2012, doi: 10.1007/s11134-011-9268-4.

[83] Yang, L., Chen, S. and Wei, D., **A Buffer Management Strategy based on Message Drop History in DTN Satellite Network**, Advances in Computer Science Research, 2nd International Conference on Mechatronics Engineering and Information Technology (ICMEIT 2017), Vol. 70, pp. 562-568, 2017, doi: 10.2991/icmeit-17.2017.123.

[84] Zhang, Z., **Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges**, IEEE Communications Surveys & Tutorials, Vol. 8, no. 1, pp. 24-37, 2007, doi: 10.1109/COMST.2006.323440

[85] Zhang, T., Deng, S., Li, H., Hou, R. and Zhang, H., **A maximum flow algorithm for buffer-limited Delay Tolerant Networks**, Journal of Communications and Information Networks, Vol. 2, No. 3, pp. 52-60, 2017, doi: 10.1007/s41650-017-0016-8.

[86] Zhu, Y., Xu, B., Shi, X. and Wang, Y., **A Survey of Social-Based Routing in Delay Tolerant Networks: Positive and Negative Social Effects**, IEEE Communications Surveys and Tutorials, Vol. 15, no. 1, pp. 387-401, 2013, doi: 10.1109/SURV.2012.032612.00004.

[87] `https://en.wikipedia.org/wiki/Routing_in_delay-tolerant_networking`

[88] `https://en.wikipedia.org/wiki/M/G/1_queue`

[89] `https://akeranen.github.io/the-one/`

[90] https://crawdad.cs.dartmouth.edu

[91] https://www.graphviz.org/

[92] http://news.bbc.co.uk/2/hi/americas/8459653.stm

Scurt rezumat al tezei

Teza de doctorat, intitulată **Cercetări în teoria aşteptării şi în reţele cu toleranţă la întârzieri**, are ca obiectiv principal identificarea unui algoritm care să maximizeze rata de livrare a mesajelor într-o reţea de tip DTN. Pentru atingerea obiectivului, au existat o serie de cercetări preliminarii. Una dintre cercetările premergătoare algoritmului propus se referă la teoria aşteptării, prin identificarea şi implementarea unor sisteme de aşteptare cu revenire, descrise şi analizate în capitolul 2 al tezei. Cea de-a doua cercetare premergătoare constă în identificarea unei politici de abandonare optimă a mesajelor dintr-un buffer congestionat, descrisă în capitolul al treilea al tezei. Tot în acest capitol am transformat o reţea dinamică de tip DTN într-o reţea statică, în care am identificat fluxul maxim. Toate aceste studii au culminat cu realizarea algoritmului de rutare MaxDelivery, care utilizează o politică de gestiune a bufferului, pentru a optimiza procentul de mesaje livrate la destinaţie. A fost prezentată de asemenea şi o variantă a acestui algoritm, care lucrează cu mesaje de diferite priorităţi, maximizând rata de transmitere a mesajelor cu prioritate mare. Simulările care au evidenţiat performanţele algoritmului MaxDelivery au fost realizate cu simulatorul ONE şi au fost făcute comparaţii cu unii dintre cei mai cunoscuţi algoritmi de rutare din DTN. Rezultatele cuprinse în această teză pot fi îmbunătăţite şi extinse pentru a fi aplicate în contexte reale variate.

* * *
*

The main objective of the PhD thesis, entitled **Research in Queuing Theory and Delay Tolerant Networks**, is the identification of an algorithm that maximizes the delivery rate of the messages in a DTN network. In order to achieve this goal, there was a set of preliminary researches. One of the first research preceding the proposed algorithm refers to the queuing theory, identifying and implementing single server retrial queuing systems, described and analyzed in chapter 2 of the thesis. The second preliminary research consists in identifying an optimal dropping policy of messages from a congested buffer, described in the third chapter of the thesis. Also in this chapter we transform a dynamic DTN network into a static network, in which we identify the maximum flow. All these studies culminated with the development of the MaxDelivery routing algorithm, which uses a buffer management policy to optimize the percentage of messages delivered to the destination. A variant of this algorithm was also presented, which works with messages with different priorities, maximizing the transmission rate of high priority messages. The simulations that highlight the performance of the MaxDelivery algorithm were performed with the ONE simulator. The results were compared with those of the best known routing algorithms in DTN. The researches contained in this thesis can be improved and extended to be applied in various real contexts.