UNIVERSITATEA BABEŞ-BOLYAI
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ

Sisteme mari de fişiere distribuite pe platforme Unix

Rezumat

Doctorand: Dan Cojocar
Conducător ştiinţific: Prof. Univ. Dr. Florian Mircea Boian

Cluj-Napoca
2015

Listă publicaţii:

[1] Florian Mircea Boian, Darius Vasile Bufnea, **Dan Cojocar**, Alexandru Ioan Vancea, and Adrian Sterca. "A Model for Efficient Session Object Management in Web Applications". In: *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*. Cluj–Napoca, Romania, June 2006, pp. 137–142.

[2] **Dan Cojocar**. "Hardware Fault-Tolerant File System". In: *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*. Cluj–Napoca, Romania, June 2006.

[3] Ioan Lazar and **Dan Cojocar**. "On Model-Driven Development for web Applications". In: *Studia Universitatis Babeş-Bolyai, Informatica* (2006), pp. 101–112.

[4] Boian Florian Mircea, Darius Vasile Bufnea, Claudiu Cobarzan, Alexandru Ioan Vancea, Adrian Sterca, and **Dan Cojocar**. *Sisteme de operare*. RISOPRINT, 2006.

[5] Florian Mircea Boian, Darius Vasile Bufnea, Alexandru Ioan Vancea, Adrian Sterca, **Dan Cojocar**, and Rares Florin Boian. "Some Formal Approaches for Dynamic Life Session Management". In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, June 2007, pp. 227–235.

[6] **Dan Cojocar**. "BBUFs: A new lookup mechanism based on IPV6". In: *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2008*. Timisoara, Romania, 2008, pp. 358–361. **(ISI Proceedings)**.

[7] **Dan Cojocar**. "BBUFs: Synchronization Mechanism". In: *6th International Conference of Applied Mathematics (ICAM)*. Baia Mare, Romania, 2008, pp. 363–368.

[8] Rares Florin Boian and **Dan Cojocar**. "Moving Excess Data Into External Peer-to-Peer Storage". In: *KEPT2009 Knowledge Engineering Principles and Techniques* Selected Papers (2009), pp. 358–365. **(ISI Proceedings)**.

[9] Rares Florin Boian and **Dan Cojocar**. "Moving Excess Data Into External Peer-to-Peer Storage". In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, July 2009, pp. 296–299.

[10] **Dan Cojocar**. "BBUFs: Architecture Overview". In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, July 2009, pp. 276–279.

[11] **Dan Cojocar**. "The Architecture of BBUFs". In: *KEPT2009 Knowledge Engineering Principles and Techniques* Selected Papers (2009), pp. 335–342. **(ISI Proceedings)**.

[12] **Dan Cojocar** and Florian Mircea Boian. "BBUFs: Replication Strategies". In: *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*. Cluj–Napoca, Romania, July 2009, pp. 284–287.

[13] **Dan Cojocar** and Florian Mircea Boian. "BBUFs: Replication Strategies Comparison". In: *KEPT2009 Knowledge Engineering Principles and Techniques* Selected Papers (2009), pp. 343–350. **(ISI Proceedings)**.

[14] **Dan Cojocar**. "BBUFs: Routing Protocol". In: *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*. Cluj–Napoca, Romania, June 2010, pp. 116–121.

[15] Grigoreta Sofia Cojocar and **Dan Cojocar**. "A Comparison of AOP Based Monitoring Tools". In: *Studia Universitatis Babeş-Bolyai, Informatica* (2011), pp. 65–70.

[16] **Dan Cojocar**. "Replication Location Decisions". In: *Nano, Information Technology and Reliability (NASNIT), 2011 15th North-East Asia Symposium*. Macao, China, Oct. 2011, pp. 161–165. **(indexed IEEE)**.

[17] **Dan Cojocar**. "Load Balance Queries in Decentralized Peer-to-Peer File Systems". In: *Proceedings of the National Symposium ZAC2012 (Zilele Academice Clujene, 2012)*. 2012, pp. 105–110.

[18] Gabriel Ciobanu and **Dan Cojocar**. "Expressing BBUFs lookup using the π-calculus". In: *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2014*. Timisoara, Romania, 2014. **(ISI Proceedings)**.

Cuprins

Introducere

Această teză reprezintă rezultatul cercetării efectuate de noi în domeniul sistemelor de operare, în special în domeniul sistemelor de fişiere distribuite descentralizate nestructurate. Cercetarea a început în 2005 sub îndrumarea Prof. dr. Florian Mircea Boian.

În această lucrare ne vom concentra pe prezentarea problemelor legate de operaţia de căutare în astfel de sisteme descentralizate. Mai exact încercăm să studiem problemele care apar în regăsirea datelor într-un astfel de sistem. Studiem cum se pot îmbunătăţi performanţele sistemului folosind un sistem de replicare a datelor şi de balansare a cererilor de la clienţi utilizând aceste copii. De asemenea vom prezenta şi o propunere de un nou formalism, util în a modela acţiunile unui astfel de sistem. Teza este împărţită în 4 capitole după cum urmează.

Capitolul 1, **Sisteme de fişiere distribuite**, prezintă date generale legate de sistemele distribuite peer-to-peer, modele de reprezentare a sistemelor de fişiere descentralizare, problemele generale legate de regăsirea datelor în sisteme peer-to-peer descentralizate, cauzele ce produc aceste probleme şi cum se rezolvă.

Capitolul 2, **O nouă abordare pentru sisteme peer-to-peer descentralizate nestructurate**, prezintă o nouă abordare originala pentru operatia de căutare bazata pe protocolul IPv6. Se prezintă noutăţile introduse de protocolul IPv6, redefinirea schemelor de adresare anycast şi multicast, iar apoi operaţia de căutare bazata pe IPv6 propusă. De asemenea, este prezentată arhitectura unui sistem de fişiere descentralizat nestructurat, numit BBUFs, dezvoltat folosind această abordare. În acest capitol este descrisa si o nouă metodă originală de a balansa cererile când sistemul dispune de mai multe replici.

Capitolul 3, **O nouă strategie de replicare** introduce o nouă metodă de replicare care să permită specificarea locaţiei în care să se plaseze replicile.

Capitolul 4, **Formalizare Lookup**, prezintă un model formal de căutare. Sunt prezentate câteva abordări precedente şi se propune o extindere a formalismului *Time and Space Coordination of Mobile Agents*. De asemenea se prezintă un model formal pentru mecanismul propus în capitolul 2 şi se face validarea acestuia.

Contribuţiile originale introduse de această teză se regasesc în capitolele 2, 3, 4, şi propun:

- O nouă abordare în rezolvarea problemei regăsiri datelor într-un sistem descentralizat nestructurat [Coj09, Coj08a] (Secţiunea 2.2).
- Propunerea unui nou sistem de fişiere distribuit descentralizat, bazat pe operaţia de căutare propusă [Coj09, Coj08b] (Secţiunea 2.4).
- Un nou mod de a balansa cererile clienţilor în sisteme în care se folosesc mai multe replici [Coj12, Coj10] (Secţiunea 2.3.1).
- Un nou model de a reprezenta datele interne care să permită distribuirea conţinutului unui fişier pe mai multe noduri [Coj08b] (Secţiunea 2.4.3).
- O nouă strategie de replicare care să tină cont de locaţia nodurilor, folosind diverse metrici [CB09a, Coj11] (Secţiunea 3.1).
- O propunere de formalizare a mecanismului de căutare folosind π-calcul [CC14] (Secţiunea 4.1.1).
- Un nou formalism creat prin extinderea atributului de domeniu la ambienţii mobili definiţi de formalismul: *Time and Space Coordination of Mobile Agents* (Secţiunea 4.2.2).

Capitolul 1

Sisteme de fişiere distribuite

1.1 Sisteme peer-to-peer

Sistemele peer-to-peer, aproape necunoscute acum zece ani, au ajuns să fie sursă majoră de trafic în Internet. Lv [LCC+02a] clasifică sistemele peer-to-peer existente, în funcţie de arhitectura lor, în următoarele categorii:

- *centralizate* - sisteme care au un nod central prin care se gestionează accesul în întreg sistemul.
- *descentralizat* - în această categorie nici un nod nu deţine controlul absolut. Această categorie se împarte în două subcategorii:
  - *structurate* - între noduri şi datele pe care acestea le reprezintă există o legatură directă, foarte strânsă.
  - *nestructurate* - sisteme la care nodurile nu sunt organizate în funcţie de informaţia pe care o gestionează.

Nu ne propunem să susţinem o anumită funcţionalitate, dar nu putem să nu remarcăm că sistemele *descentralizate* au câştigat detaşat la capitolul popularitate. Tocmai datorită descentralizării, ele pot să ofere servicii mai bune clienţilor lor, prin: timpi de răspuns mai mici şi viteze de transfer mai mari [Ora01].

1.2 Modele de reprezentare a sistemelor descentralizate

Majoritatea implementărilor descentralizate folosesc tabele de dispersie distribuite pentru reprezentarea nodurilor din reţea, şi adaugă un nivel suplimentar reţelei existente. Acest nivel suplimentar este apoi folosit pentru căutarea datelor. Un dezavantaj major al acestei abordări este că de fiecare dată când un nou nod intră în sistem sau un nod existent părăseşte sistemul, nivelul suplimentar de reţea trebuie actualizat. În continuare sunt prezentate conceptele de bază legate de tabelele de dispersie distribuite.

1.2.1 Tabele de dispersie distribuite

Conceptul de tabelă de dispersie distribuită (eng. *Distributed Hash Table - DHT*) a fost propus de Litwin, Niemat şi Shneider în 1996 în lucrarea [LNS96]. Începand cu anul 2001 au început să apară primele propuneri de tabele de dispersie distribuite care sunt împărţite pe reţele foarte mari.

O tabelă de dispersie distribuită este o tabelă de dispersie de dimensiuni foarte mari împărţită şi gestionată pe o mulţime de calculatoare de pe glob care cooperează, numite *noduri*. Mulţimea de obiecte păstrate într-o DHT este împărţită în partiţii disjuncte, fiecare nod fiind responsabil de gestiunea partiţiei alocate lui. Mulţimea de noduri care participă la implementarea unei DHT este dinamică: în orice moment un nod nou se poate alătura structurii, un nod existent poate oricând părăsi structura şi, de asemenea, pot exista noduri care nu funcţioneaza normal (eşuează).

1.2.2 Implementări pentru tabele de dispersie distribuite

Primele implementări pentru tabele de dispersie distribuite au apărut în 2001 şi pornesc de la două idei publicate în 1997:

- Hashing consistent (eng. *consistent hashing)* - este un tip special de hashing, care asigură faptul că atunci când o tabelă de dispersie este redimensionată doar

K/n chei trebuie rearanjate unde K este numărul de chei, iar n este numărul de poziţii disponibile din tabela de dispersie.

- Schema Plaxton Mesh (sau PRR) - este o proiectare care permite rutarea eficientă la nodul responsabil de un anumit obiect, folosind tabele de rutare de dimensiuni mici [PRR97].

Abordările propuse pentru construirea unei tabele de dispersie diferă, de obicei, prin modul în care unui nod i se asociază un ID şi prin construirea tabelei de rutare asociate fiecărui nod.

1.3 Sisteme de fişiere distribuite peer-to-peer

Sistemele de fişiere distribuite peer-to-peer sunt sisteme descentralizate nestructurate [Ber03], în care nodurile sistemului sunt calculatoare simple [LM09]. Toate nodurile din sistem au aceleaşi capabilităţi dar şi aceleaşi responsabilitaţi, comunicarea între noduri fiind directă. O parte importantă din funcţionalităţile care trebuie luate în considerare în implementarea unui sistem de fişiere descentralizat sunt:

- Căutarea datelor.
- Un model prin care să se asigure accesul la informaţia utilizatorului chiar dacă apar pene de reţea sau probleme hardware.
- Gestionarea cât mai eficientă a resurselor.
- Securitatea informaţiei.

1.4 Problema căutării datelor

Una dintre problemele majore intr-un sistem peer-to-peer descentralizat este problema căutării datelor, numită *operaţia de căutare sau lookup: Cum regăsim o resursă într-un sistem descentralizat foarte mare într-un mod cât mai eficient, fără a avea acces la un nod central?* [BKK+03]

Un mod simplu de a enunţa operaţia de căutare este: Un client depune un fişier în sistem, sau un nod din sistem publică un anumit fişier; la un moment dat un alt client doreşte să folosească acel fişier. Cum poate acest client să localizeze nodul pe care se află fişierul iniţial sau o replică a acestuia?

1.5 Balansarea cererilor

1.5.1 Abordări similare

Folosind diferite strategii de replicare (descrise în Secţiunea 1.6), cele mai multe sisteme folosesc un număr mare de replici, pentru a oferi o performanţă ridicată, dar şi o soluţie prin care se conferă siguranţa datelor în caz de accidente [WJH97, LSSD02, HSW94].

Abordările propuse până acuma pentru balansarea cererilor pot fi clasificate în următoarele categorii:

- *Statice* - Sistemul decide unde să se efectueze operaţia la fiecare operaţie de scriere sau depunere de conţinut.

- *Dinamice* - Sistemele sunt proiectate să facă faţă în cazul în care sunt cereri frecvente, abordarea statică întâmpinând dificultăţi în acest caz, datorită faptului ca evaluarea locaţiei se face pentru fiecare operaţie inainte de a efectua operaţia.

Abordările dinamice sunt folosite mai ales de sistemele structurate [CDG+08, SKRC10] deoarece în aceste sisteme se poate beneficia de structura oferită de sistem. Nodul central este nodul care poate să distribuie cererile. Când un client face o nouă cerere, nodul central îl va trimite la cel mai bun nod care să-i dea răspunsul cerut. Astfel toate optimizările şi deciziile se fac la acest nod central [HGM10].

1.6 Replicarea

Sistemele şi aplicaţiile peer-to-peer descentralizate şi nestructurate sunt sisteme distribuite în care nu există un nod central care să gestioneze activităţile sistemului. De asemenea datorită faptului că sunt nestructurate, deciziile se iau la nivel de nod [LCC+02b]. Astfel toate nodurile din sistem au acelaşi cuvânt decizional.

Din cauză că nodurile dintr-un sistem peer-to-peer descentralizat şi nestructurat sunt transiente şi pot să dispară în orice moment, aceste sisteme trebuie să se asigure că anumite conţinuturi sunt replicate pe mai multe noduri [PC04].

Menţinerea, chiar şi a unei singure copii, introduce o nouă problemă: *consistenţa datelor replicate*. Dacă statusul replicii se modifică, datorită operaţiilor de actualizare sau adăugare, toate celelalte copii necesită aceleaşi modificări. Pentru a rezolva această problemă s-a introdus un nou mecanism numit *sincronizarea datelor* [MK03].

1.7 Sisteme existente de fişiere distribuite descentralizate

Există multe încercări de a construi un sistem distribuit de fişiere. Unele dintre ele, precum Netware [MMP94], NFS [SGK+85], Andrew [HKM+88] şi Sprite [NWO88] folosesc un nod central, care este responsabil cu deservirea tuturor cererilor clienţilor. Acest model de sistem impune restricţii privind atât performanţa cât şi siguranţa datelor. Pentru a evita aceste dezavantaje s-a migrat spre implementări peer-to-peer descentralizate, precum: OceanStore [KBC+00], Ivy [MMGC02], Freenet [CSWH01], PAST [RD01b] sau CFS [DKK+01]. Fiecare având avantaje şi dezavantaje proprii.

Capitolul 2

O nouă abordare pentru sisteme peer-to-peer descentralizate nestructurate

În acest capitol propunem o nouă abordare de implementare a unui sistem peer-to-peer descentralizat nestructurat bazată pe IPv6[DH98], în care nodurile nu menţin nici un fel de informaţie suplimentară despre nodurile din sistem. În continuare prezentăm noţiunea de adresă IPv6 anycast, noua abordare propusă, iar apoi prezentăm ca şi studiu de caz Babeş Bolyai University Filesystem (BBUFs).

2.1 Adrese IPv6 anycast

Protocolul IPv6 a fost dezvoltat pentru a rezolva problema epuizării spaţiului de adrese din versiunea precedentă, IPv4 [Tra95]. Specificaţia IPv6, pe langă rezolvarea acestei probleme, propune şi alte numeroase îmbunătăţiri la conceptele definite anterior. Printre acestea apare şi redefinirea tipului de adresă anycast, ca fiind un identificator a unui set de interfeţe, de obicei de pe noduri diferite. Astfel, un pachet trimis la o adresa anycast va fi direcţionat către cea mai "apropiată" interfaţă care are această adresă.

Chi-Yuan Chang foloseşte acest mecanism de adresare la proiectarea unui ser-

viciu de rutare folosind logica fuzzy [CCC+05], pentru a determina cum să se facă balansarea cererilor.

2.2 Abordarea propusă

În această secţiune prezentăm o nouă abordarea de a dezvolta un sistem de fişiere distribuit descentralizat nestructurat folosind adresele IPv6 anycast.

Diferenţa majoră între abordarea propusă şi sistemele distribuite bazate pe Chord sau DHash este faptul că abordarea noastră nu construieşte niciun nivel de reţea suplimentar menit să gestioneze nodurile din sistem. În această abordare ne bazăm doar pe nivelul de reţea definit în IPv6. De asemenea, nodurile sunt independente unele faţă de celelalte, singurul lucru cunoscut de un nod este modul în care să gestioneze resursele proprii şi să se asocieze unui grup/subreţea care reprezintă aceste resurse.

În această abordare un grup este un set de noduri care replică acelaşi conţinut (vezi Figura 2-1). Fiind vorba de un sistem de fişiere grupurile sunt definite la nivel de directoare. Un nod poate să facă parte dintr-o multitudine de astfel de grupuri. Un grup este reprezentat de:

- O adresă IPv6 alocată din subreţeaua dedicată adreselor anycast.
- O adresă IPv6 alocată din subreţeaua de adrese multicast corespunzătoare adresei anycast. Această adresă se obţine din adresa anycast schimbând doar prefixul.

Astfel, sistem propus foloseşte trei tipuri de adrese:

- *Unicast* - folosite pentru comunicarea între clienţi şi nodurile din sistem, dar şi pentru comunicarea între nodurile din sistem.
- *Anycast* - folosite pentru mesajele produse de operaţia de căutare.
- *Multicast* - folosite doar între nodurile din sistem (pentru sincronizarea replicilor, notificări, etc.)

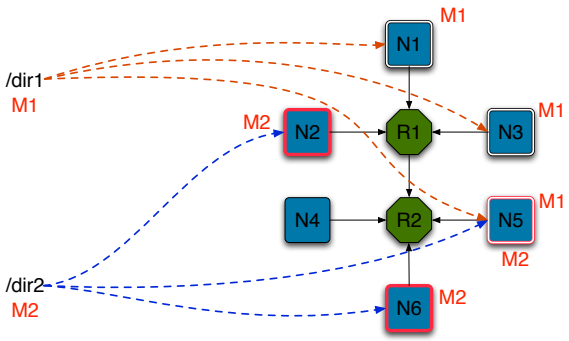

Figura 2-1: Fiecare nod face parte din grupul asociat conţinutului partajat.

Fiecare nod are cel puţin câte o adresă din fiecare grup amintit anterior (vezi Figura 2-2:

- *O adresă unicast* - aceasta va reprezenta nodul în sistemul de fişiere, fiind asociată unic nodului de către administratorul de sistem.
- *Cel puţin o adresă anycast* - va reprezenta cel puţin un director pe care acest nod îl gestionează. Această adresă este generată automat folosind un algoritm de mapare cum este cel definit în secţiunea 2.4.2.
- *Cel puţin o adresă multicast* - reprezentând grupul din care acest nod face parte.

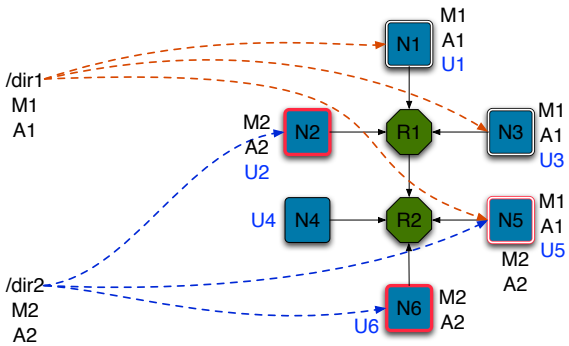

Figura 2-2: Adresele unui nod.

Folosind doar schemele de adresare din IPv6 se elimină necesitatea menţinerii unui nivel de reţea suplimentar, folosit la implementările studiate, şi se optimizează considerabil operaţia de căutare [Coj08a].

2.2.1 Operaţia de căutare

În această secţiune prezentăm paşii pe care un client îi urmează pentru a căuta o anumită resursă în sistem. Presupunem că avem un sistem de fişiere reprezentat ca în Figura 2-3, unde Nx sunt noduri ($x \in \{1..3\}$), Ry sunt rutere ($y \in \{1..2\}$) iar C este clientul. Clientul C doreşte să găsească locaţia fişierului $/d1/d2/f2$. Folosind

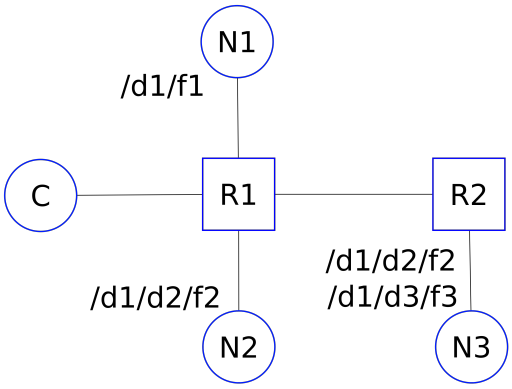

Figura 2-3: Nodurile unui sistem de fişiere distribuit.

algoritmul de mapare prezentat în Secţiunea 2.4.2 şi variabilele sistem: prefixul de reţea, adâncimea preferată şi funcţia de hash, putem să determinăm adresa anycast a nodului care deţine fişierul căutat.

2.2.2 Protocol de comunicare

În această secţiune prezentăm o parte din operaţiile cele mai importante din acest tip de sistem distribuit descentralizat nestructurat. Descriem paşii pe care îi execută sistemul în următoarele cazuri: când un nod intră în sistem, când un nod părăşeste sistemul, şi când un client caută o resursă.

2.2.3 Avantaje şi dezavantaje

În această secţiune prezentăm o parte din avantajele şi dezavantajele abordării propuse. Dintre avantaje distingem:

- Nu mai este nevoie să se întreţină un nivel secundar de reprezentare a reţelei.

- Nodurile sistemului nu trebuie să menţină registre cu informaţii cum să contacteze alte noduri sau care sunt nodurile vecine, etc.

- Pentru a determina nodul care conţine resursa dorită se foloseşte un număr mult mai mic de cereri decât în abordările anterioare [SMK+01, RFH+01, HKRZ02, RD01a] (doar în cel mai rău caz nodurile care sunt din grupul respectiv sunt contactate).

Dezavantajele abordării sunt:

- Sistemul depinde de IPv6. Din fericire în prezent toate sistemele de operare suportă IPv6.

- Un nod poate avea foarte multe adrese IPv6 asociate generând astfel tabele de rutare mari. Stabilind un număr relativ mic (20-30) pentru parametrul de adâncime, când se apelează algoritmul de mapare, se poate controla numărul de astfel de adrese folosite.

- Când apar coliziuni, sistemul foloseşte adrese multicast care pot să genereze congestie [Jac88]. Această problemă poate fi înlăturată dacă folosim un număr mai mic pentru partea de reprezentare a subreţelei şi alocăm mai mulţi biţi pentru funcţia de hash.

2.3 Balansarea cererilor

În această secţiune descriem cum se realizează balansarea cererilor pentru abordarea noastră de sisteme de fişiere peer-to-peer descentralizare nestructurate. Propunerea noastră încearcă să aducă avantajele enumerate în Secţiunea 1.5.1 şi la sistemele peer-to-peer descentralizate nestructurate. Pentru a ne asigura că noua abordare aduce aceste avantaje, trebuie să stabilim următoarele:

- Unde se stabilesc deciziile de balansare în cazul operaţiilor de depunere sau modificare de conţinut?

- Dacă la sistemele centralizate nodul central menţine şi foloseşte registrele care ajută la luarea deciziei, unde vor fi gestionate aceste registre în cadrul unui sistem nestructurat descentralizat?

2.3.1 MCP - Maximizarea utilizării puterii de calcul

Rezultatele prezentate de Dong Xuan şi echipa sa, demonstrază că se poate implementa un algoritmul de rutare şi pentru adresele de tip anycast, acesta fiind o soluţie viabilă în gestionarea cererilor către nodurile care folosesc acest tip de adrese [XJZZ00].

Folosind rezultatele obţinute de Xuan, Han Zhi-nan defineşte două metrici care pot fi folosite pentru a stabili un astfel de drum optim [ZnWLY11].

Rezultatele publicate de noi în [Coj10] definesc o nouă astfel de metrică, numită *Putere maxima de computaţie ("Maximum Computing Power" MCP)*. Această metrică ţine cont de tipul de cerere şi de cum s-a comportat sistemul până acuma în situaţii asemănătoare.

MCP este definită astfel: $MCP(n) = ri - k * no\_cores$, unde:

- n - Reprezintă nodul la care se calculează puterea de calcul pe care o are disponibilă.
- ri - Numărul de servicii care rulează pe acest nod.
- k - O constantă prin care se permite ajustarea algoritmului.
- no_cores - Numărul de nuclee disponibile pe acest nod.

Astfel, folosind această metrică putem să alegem un drum care să ne ducă la nodul care are capacitatea de a rezolva o cererea cât mai rapid.

2.3.2 AQK - Costul mediu pe tip de operaţie

În [Coj12] arătăm cum folosind aceste rezultate şi informaţiile colectate de sistem, fiecare nod poate să stabilească care este costul pentru un anumit tip de cerere.

Definiţia 1 Costul unei operaţii o este definit astfel:

$$tipOperaţie(o) * dimensiuneConţinut$$

unde:

- *costOperaţie* (OK - *Operation cost*) - reprezintă costul operaţiei efectuate o. Aici putem controla faptul că operaţiile de scriere sunt mai costisitoare decât operaţiile de citire.
- *dimensiuneConţinut* (CS - *Content Size*) - dimensiunea cererii.

Prin faptul că se asociază un astfel de cost pentru fiecare cerere primită cu timpul de procesare al cererii respective (AST - *actual time spend*), sistemul poate să stabilească care este costul pentru fiecare cerere (QK - *Query cost*). De asemenea, sistemul poate să menţină o valoare a costului mediu (AQK) pentru fiecare tip de cerere, definită în continuare:

Definiţia 2 Costul mediu AQK este definit astfel:

$$AQK = \frac{\sum_{i=1}^{N} QK(i)}{\sum_{i=1}^{N} CS(i)}$$

unde: $QK(i) = OK(i) * CS(i) * AST(i)$ reprezintă costul cererii i.

Folosind valorile MCP, un router poate să decidă care sunt nodurile libere în acest moment. Utilizând costul definit pe tip de cerere, routerul poate să aleagă un nod care are valoarea AQK cea mai mică pentru acest tip. Astfel se va alege nodul cel mai liber care poate să servească cererea în cel mai scurt timp [Coj12].

2.4 Studiu de caz: BBUFs

BBUFs (Babeş Bolyai University File System) [Coj09] este un sistem de fişiere distribuit care are caracteristici similare cu Ivy [MMGC02] şi Pastis [BPS05]. Dintre aceste caracteristici amintim:

- *Este scalabil* - datorită faptului că este un sistem descentralizat, iar un nod nu menţine nici un fel de legătură cu alte noduri.
- *Este tolerant la pene* - sistemul menţine replici pentru fiecare resursă din sistem, în locaţii diferite.
- *Performanţă sporită* - datorită modelului peer-to-peer, fiecare client se conectează direct la nodul la care se află resursa care îl interesează (fără noduri intermediare).

2.4.1 Arhitectura sistemului

Obiectul de bază din sistem este *BBUMeta* prezentat în figura 2-4 [Coj08b]. Acest obiect menţine pentru fiecare conţinut (instanţă) informaţii care să identifice conţinutul în sistem, în cadrul nodului, câte copii se doresc a fi menţinute în sistem, etc. Important de menţionat aici este faptul că acest număr reprezintă câte copii se doresc.

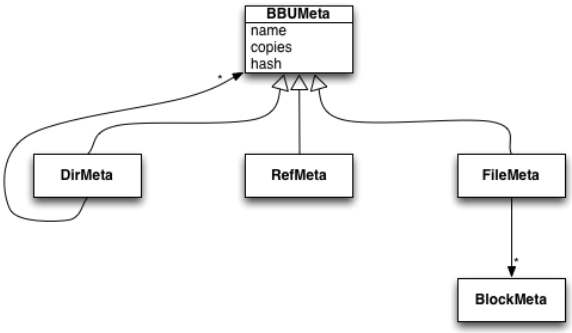

Figura 2-4: Ierarhia claselor de bază.

Altă componentă de bază din sistem este *BBUFsRepository* (vezi Figura 2-5). Aceasta rulează pe fiecare nod din sistem şi este responsabilă cu tratarea cererilor de actualizare sau de citire a conţinutului [CB09b].

Utilizând componentele definite până acum putem să încercăm să răspundem la întrebarea: *Ce se întâmplă când detectăm că avem o copie coruptă?*

Unul din scopurile sistemului BBUFs este să supravieţuiască chiar şi în situaţii când o întreagă subreţea nu mai este disponibilă. Pentru aceasta trebuie să ne asigurăm şi de modul în care replicile sunt distribuite, nu doar la felul în care putem

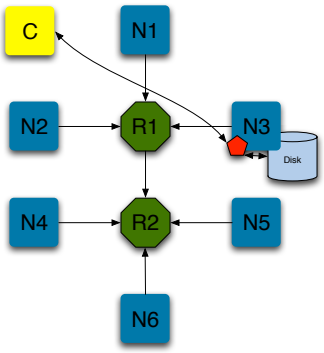

Figura 2-5: BBURepository - rezolvă cererile venite de la clienţi.

mări performanţele clienţilor şi la numărul de copii pe care trebuie să le facem. În Capitolul 3 prezentăm o abordare proprie în a atinge acest obiectiv.

2.4.2 BBUFsMapper

În această secţiune prezentăm algoritmul BBUFsMapper, folosit de BBUFs pentru a asocia un nume de director cu o adresă IPv6.

2.4.3 Replicarea în BBUFs

Sistemul de fişiere BBUFs foloseşte mai multe replici pentru a evita pierderile de date şi pentru a menţine copii cât mai apropape de clienţi, oferindu-se astfel o performanţă sporită [CB09b, Coj11].

În abordarea propusă de noi un grup este un set de noduri care replică acelasi conţinut.

Abordarea noastră încercă să răspundă şi la următoarele întrebări:

- Cum sincronizăm un director întreg?
- Unde să facem noua replică?
- Cum să detectăm dacă o anumită replică şi-a modificat conţinutul?
- Pe care nod avem cele mai recente date?

Algorithm 1 Algoritmul BBUFsMapper

Date intrare:

- *type* - tipul de adresă pe care dorim să-l generăm (anycast sau multicast)
- *dir* - numele directorului, folosind calea absolută
- *depth* - adâncimea preferată
- $prefix$ - prefixul de reţea
- *hash* - funcţia hash

Date ieşire

- *address* adresa generată

Algoritmul *BBUFsMapper* este:
Begin
if $computeDepth(dir) > depth$ **then**
    $dir \leftarrow determineParentDir(dir, depth)$ ▷ determină directorul părinte *dir* care are adâncimea *depth*.
end if
$key \leftarrow hash(dir)$ ▷ returnează valoarea hash pentru string-ul *dir*
if *type* este anycast **then**
    $address \leftarrow "FEC0:" + prefix + key$
else
    $address \leftarrow "FF08:" + prefix + key$
end if
End.

2.5 Concluzii şi direcţii noi de cercetare

În acest capitol am prezentat o abordare originală de dezvoltare a unui sistem de fişiere distribuit descentralizat nestructurat, care foloseşte noul mecanism de adresare anycast introdus în IPv6. Am descris o parte din operaţiile cele mai importante pe care sistemul distribuit trebuie să le implementeze: paşii necesari unui nod de a participa în sistem, ce se întâmpla când un nod părăseşte sistemul. De asemenea am prezentat avantajele şi dezavantajele abordării propuse.

Capitolul 3

O nouă strategie de replicare

În acest capitol prezentăm o abordare proprie în soluţionarea problemelor descrise în secţiunea 1.6, pentru care am folosit ca şi context sistemul BBUFs [Coj09]. BBUFs este un sistem peer-to-peer descentralizat, în care se foloseşte direct infrastructura definită în IPv6 [DH98]. Din faza de design s-a ales o abordare descentralizată nestructurată, unde fiecare nod este total independent de restul sistemului, iar o cerere de la un client se încearcă să se rezolve de nodul cu şansele cele mai mari să gestioneze conţinutul dorit folosindu-se mecanismul de căutare (Lookup), prezentat în Capitolul 2, algoritm de complexitate O(1) în cazul general [Coj08a]. Secţiunea 3.1 prezintă noua strategie propusa pentru localizarea unei locaţii candidat pentru o nouă replică. De asemenea sunt prezentate şi câteva concluzii şi direcţii noi de cercetare, legate de problemele abordate în acest capitol.

3.1 Replicare în funcţie de locaţie

Folosind un program, numit *BBUFsStatistics* (vezi Figura 3-1), care colectează date despre sistem, programul responsabil cu replicarea conţinutului poate uşor să determine când este nevoie de o nouă replică [CB09b].

Pentru aceasta va folosi una din strategiile următoare:

- *Replicarea bazată pe acces* - odată ce se determină că numărul de accesări a depăşit un prag critic se iniţiază procesul de creare a unei noi copii.

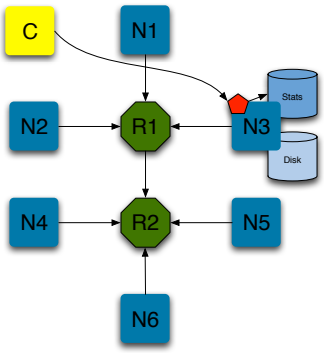

Figura 3-1: BBUStatistics - colectează date legate de cererile primite de nod.

- *Replicarea bazată pe ponderi* - sistemul ţine cont de legătura existentă între două directoare.
- *Replicarea bazată pe nivelul de încărcare a unui nod* - dacă un nod, la un moment dat, detectează că este foarte ocupat, în a servi un anumit conţinut, poate să iniţieze un proces de replicare al acestuia.

În acest capitol am prezentat o nouă strategie de replicare, *replicarea bazată pe locaţie*, propusă pentru sistemul BBUFs. Această strategie foloseşte mai multe date de intrare:

- timpul de răspuns dus-întors (rtt),
- timpul de expediere (ttl),
- atribute care să ne ajute să stabilim locaţia exactă a unui nod cum ar fi latitudinea şi longitudinea la care acesta se află (GPS).

Capitolul 4

Formalizare Lookup

4.1 Π-calcul

Metodele formale sunt folosite cu succes pentru a exprima şi verifica proprietăţile diverselor protocoalelor de comunicare. Astfel, un proces de verificare este compus din următorii doi paşi:

1. Definirea protocolului, care poate rezulta într-un model.

2. Verificarea proprietăţilor protocolului.

În Secţiunea 4.1.1 prezentăm formalizarea mecanismului de căutare pentru a descrie comportamentul şi protocolul utilizat, folosind π-calul [CC14]. Se va fi utiliza Mobility Workbench ca şi utilitar de specificare şi validare a modelului [VM94].

4.1.1 Detalii lookup

Definirea protocolului folosind π-calcul

Definim suprocesul *Lookup(memory, network, request, response, failed, valid, address, read, console, exists, missing, rule, check, message)* astfel:

$$Lookup(memory, ...) \stackrel{def}{=} memory(inputParam).$$
$$\overline{network}\langle request\rangle.Medium(...)$$
$$network(response).ProcessResponse(...)$$

Subprocesul *Medium* este un proces mai complex deoarece poate să comunice atât cu clientul cât şi cu nodurile sistem, noduri care deservesc cererile clienţilor.

$$Medium(...) \stackrel{def}{=} memory1(message).\overline{verifier}\langle rule1\rangle.$$
$$check1(rule1).([rule1 = exists1]Router(network1, ...)+$$
$$[rule1 = missing1](memory2(message).\overline{verifier}\langle rule2\rangle.$$
$$check2(rule2).([rule2 = exists2]Router(network2, ...)+$$
$$[rule2 = missing2]Medium(...))))$$

Node este subprocesul final care primeşte o cerere, prin canalul de reţea, de la procesul *Medium*. *Node(network, analyze, memory, screen, unicast, message, response, result, valid, failed)* este definit astfel:

$$Node(network, ...) \stackrel{def}{=} network(message).\overline{analyze}\langle message\rangle.$$
$$analyze(response).([response = failed]FailedReply(network, ...)$$
$$+ [response = valid]SuccessReply(network, ...))$$

Astfel putem să exprimăm interacţiunea dintre componentele sistemului. Dar de asemenea putem să observăm că în acest model nu s-a ţinut cont de dimensiunea *timp*. Folosind doar π-calcul nu putem să exprimăm ce se întâmplă când un canal este întrerupt sau când un subproces nu mai răspunde la timp. Folosind extensia de

timp propusă de Berger în [BH03] putem să rezolvăm şi această problemă, putând astfel să definim comportamentul sistemului şi în astfel de situaţii.

4.1.2 Timed π-calculus

Redefinire cu cronometre

Pentru că ne propunem să formalizăm operaţia de căutare vom ignora partea de cronometre de la majoritatea operaţiilor şi o să ne concentrăm doar la momentul când se trimite cererea iniţială şi când se citeşte răspunsul, pe partea de client. Cronometre asemănătoare sunt utilizate şi la subprocesele *Router*, *ForwardNode*, *SuccessReply*, etc. Astfel cronometrele folosite sunt:

- $SO\_RCVTIMEO$ de către subprocesul *ProcessResponse*, deoarece se foloseşte protocolul User Datagram Protocol (UDP) pentru a trimite cererea, dar şi pentru a aştepta după un răspuns.
- $TIME\_WAIT$ de subprocesul *Connect*, deoarece odată stabilită adresa unicast se foloseşte Transmission Control Protocol (TCP), pentru a avea o conexiune pe care să se facă transferul efectiv de date [MHA02].

Astfel, subprocesele *ProcessResponse* şi *Connect*, din Secţiunea 4.1.1, se redefinesc astfel:

$$
\begin{aligned}
&ProcessResponse(network, ...) \stackrel{def}{=} memory(response).\\
&\quad([response = failed]timer^{SO\_RCVTIMEO}\\
&\quad(0, memory(failedresponse).FailedResponse(...))\\
&\quad + [result = valid]memory(successresponse).\\
&\quad SuccessfulResponse(memory, ...))\\
&Connect(memory, ...) \stackrel{def}{=} memory(request).\overline{network}\langle request\rangle.\\
&\quad network(result).\overline{memory}\langle result\rangle.memory(result).\\
&\quad([result = failed]timer^{TIME\_WAIT}\\
&\quad(0, FailedResponse(memory, ...))+\\
&\quad[result = valid]Connect(memory, ...))
\end{aligned}
$$

4.1.3 Validarea modelului

Folosind Mobility Workbench (MWB), un utilitar de modelare, verificare şi analizare a sistemelor mobile descrise cu π-calcul [VM94], putem să definim şi să validăm modelul propus în Secţiunea 4.1.1.

Folosind operaţia de echivalenţă operaţională (bisimulation) putem de asemenea să comparăm doi agenţi. În figura 4-1 putem să vedem că *Lookup*1 cu *Lookup*2, $Node1$ cu $Node2$ şi $Router1$ cu $Router2$ sunt echivalenţi.

4.2 Ambienţi mobili

Ambienţii mobili au fost propuşi de Cardelli şi Gordon în 1999 din nevoia de a exprima acţiunea de deplasare a unui proces sau dispozitiv la diferite nivele administrative [CGG99]. Formalismul diferă de altele, cum ar fi π-calcul [Mil99] prin faptul că aici s-a pus accentul pe deplasarea procesului şi nu pe comunicarea dintre procese, cum am văzut în Secţiunea 4.1.1.

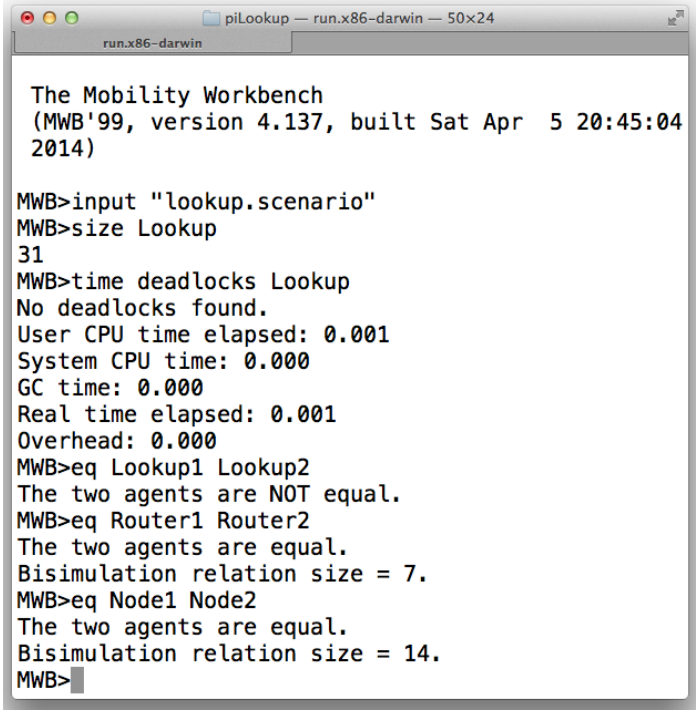

Figura 4-1: Verificarea mecanismului

În această secţiune ne propunem să extindem operaţiile de compoziţie şi selecţie, care au fost definite în [CGG99] ca operaţii asupra proceselor, pentru a putea fi folosite asupra atributului de *domeniu*. De asemenea prezentăm cum se poate folosi acest nou formalism pentru a defini operaţiile de anycast, folosite în serviciile de nume.

4.2.1 Concepte de bază

Sintaxa formală cMa

În Tabelul 4.1 este descrisă sintaxa ambienţilor mobili coordonaţi (cMA) [Cio09].

4.2.2 Comportamentul ambienţilor mobili legat de domeniu

În această secţiune prezentăm schimbările pe care le sugerăm ambienţilor mobili coordonaţi, modelul cMA [Cio09]. Prin încărcarea atributului de domeniu, pentru a putea defini mai bine componenta locală a unui ambient, putem să încapsulăm şi acţiunea de selecţie. În acestă secţiune ne propunem să studiem comportamentul mecanisme-

Tabela 4.1: Sintaxa cMA

n,m,p		nume ambient	P,Q	::=	procese
C	::=	capabilităţi		0	lipsă activitate
	in n	poate intra în n		C.P	mişcare
	out n	poate ieşi din n		$(n^{\Delta t}_{(l,h,d)}[P], Q)$	ambient
	open n	îl poate deschide pe n		P|Q	compoziţie
	go y	migrare la y		P+Q	selecţie
				$M^{\Delta t}.(P, Q)$	deplasare
				(vn)P	restricţie
				*P	replicare

lor anycast şi multicast prezentat mai sus. Deoarece diferenţa între aceste mecanisme este dată de comportamentul acestora la nivelul domeniului destinaţie, propunem să extindem acest atribut prin adăugarea operaţiilor de selectie şi compoziţie, având astfel un mecanism de formalizare a celor două mecanisme.

Extinderea atributului de domeniu la ambieţii mobili

Pentru a reprezenta un grup de destinaţii propunem următoarea notaţie: d^n_m, unde:

- m - reprezintă numărul total de destinaţii disponibile.
- n - reprezintă numărul de destinaţii pe care le alegem.

Folosind această notaţie putem să definim schemele de rutare prezentate anterior, astfel:

- *grup unicast* = $d^1_1 \equiv d_1$ - cererea clientului va fi transmisă doar unui singur nod, nod care reprezintă tot grupul.
- *grup anycast* = $d^1_m \equiv d^1$ - cererea este trimisă doar unui singur nod din grup, chiar dacă în grup avem m noduri.
- *grup multicast* = $d^n_m \equiv d^n$ - cererea se trimite la n noduri dintr-un total de m noduri de reţea. n reprezintă numărul total de noduri din acest grup multicast.

- *grup broadcast* $= d_m^m \equiv d_m$ - asemănător cu operaţia de multicast, doar că n este egal cu m.

Putem să definim un ambient cu un domeniu unicast astfel:

$$(n_{(l,h,d_1)}^{\Delta t}[P], Q) \equiv (n_{(l,h,d_i)}^{\Delta t}[P], Q)$$

unde i este unul din subdomenii.

Asemănător ambientul cu domeniul anycast are următoarea formă:

$$(n_{(l,h,d^1)}^{\Delta t}[P], Q) \equiv (n_{(l,h,d_1+d_2+...+d_m)}^{\Delta t}[P], Q)$$

folosind operaţia de selecţie la atributul domeniu.

Ambienţii pentru multicast şi broadcast fiind foarte asemănători îi prezentăm împreună, diferenţa fiind doar la partea de domeniu:

$$(n_{(l,h,d^n)}^{\Delta t}[P], Q) \equiv (n_{(l,h,d_1|d_2|...|d_n)}^{\Delta t}[P], Q)$$

Diferenţa faţă de ambientul de anycast este că se foloseşte operaţia de compoziţie.

În acest capitol am prezentat două modalităţi de a formaliza procesul de căutare (*lookup*) întâlnit într-un sistem peer-to-peer. De asemenea am prezentat o propunere de îmbunătăţire a ambienţilor mobili prin extinderea atributului care reprezintă domeniul acestuia. Astfel se pot exprima mai uşor operaţiile întâlnite frecvent în sistemele peer-to-peer, gen: broadcast şi multicast.

Specificarea şi verificarea formală a protocoalelor de reţea este şi va fi tot timpul un subiect important de cercetare deoarece ne permite să validăm şi să confirmăm modelele utilizate în practică sau modele noi.

Concluzii

Scopul acestei teze de doctorat este de a evidenţia faptul că sistemele de fişiere distribuite descentralizate nestructurate reprezintă un domeniu important de cercetare şi dezvoltare din cadrul sistemelor de operare. Sistemele de fişiere sunt folosite de mai bine de 40 de ani, în toată această perioadă devenind din ce în ce mai mari şi mai complexe. Tendinţa ultimului deceniu este de a migra datele utilizatorilor în cloud. Cloud-ul fiind un ansamblu de servicii care înglobează şi sistemul de fişiere. Acest sistem este de obicei împrăştiat pe mai multe continente pentru a putea deservi cât mai eficient utilizatorii săi.

În această teză sunt prezentate principalele rezultate din domeniul sistemelor de fişiere distribuite descentralizate, contextul în care acestea au apărut, şi noutăţile pe care le introduc. Teza introduce noi abordări, utile în astfel de sisteme, un nou mod de căutare a datelor bazat pe mecanismul anycast din IPv6, o nouă modalitate de a balansa cererile clienţilor când sistemul dispune de replici de date, şi o nouă propunere la modul cum se stabileşte locaţia unei replici. O altă direcţie pe care am investigat-o în această teză o reprezintă modul de formalizare a mecanismelor propuse. Astfel propunem extinderea formalismului Timed Coordinated Mobile Ambients pentru a putea formaliza mecanismele de tip anycast, multicast şi broadcast.

Pentru fiecare abordare propusă am sugerat şi posibile implementări, creând astfel, noi direcţii de dezvoltare în domeniul sistemelor de fişiere descentralizate nestructurate.

Ca şi direcţii noi de cercetare ne propunem să extindem modelele formale sugerate, să îmbunătăţim abordările propuse, şi să venim cu noi abordări care să asigure dezvoltarea domeniului pentru a face faţă cerinţelor clienţilor.

Bibliografie

[Ber03] J. E. Berkes. Decentralized peer-to-peer network architecture: Gnutella and freenet, 2003.

[BH03] Martin Berger and Kohei Honda. The two-phase commitment protocol in an extended π-calculus. *Electronic Notes in Theoretical Computer Science*, 39(1):21–46, 2003.

[BKK+03] Hari Balakrishnan, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Looking up data in p2p systems. *Commun. ACM*, 46(2):43–48, 2003.

[BPS05] Jean-Michel Busca, Fabio Picconi, and Pierre Sens. Pastis: A highly-scalable multi-user peer-to-peer file system. In Jos C. Cunha and Pedro D. Medeiros, editors, *Euro-Par*, volume 3648 of *Lecture Notes in Computer Science*, pages 1173–1182. Springer, 2005.

[CB09a] Dan Cojocar and Florian Mircea Boian. BBUFs: Replication strategies. In *Proceedings of Knowledge Engineering: Principles and Techniques Conference (KEPT)*, pages 284–287, Cluj–Napoca, Romania, July 2009.

[CB09b] Dan Cojocar and Florian Mircea Boian. BBUFs: Replication Strategies Comparison. *KEPT2009 Knowledge Engineering Principles and Techniques*, Selected Papers:343–350, 2009.

[CC14] Gabriel Ciobanu and Dan Cojocar. Expressing BBUFs lookup using the π-calculus. In *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2014*, 2014.

[CCC+05] Chi-Yuan Chang, Wei-Ming Chen, Han-Chieh Chao, T. G. Tsuei, and Hong Bin Liu. Ip layer load balance using fuzzy logic under ipv6 anycast mechanism. *Int. J. Netw. Manag.*, 15(5):311–319, 2005.

[CDG+08] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2):4:1–4:26, June 2008.

[CGG99] Luca Cardelli, Andrew D. Gordon, and Giorgio Ghelli. Mobility types for mobile ambients. In *ICAL '99: Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 230–239, London, UK, 1999. Springer-Verlag.

[Cio09] Gabriel Ciobanu. Coordinated mobile agents. In *Proceedings of the Romanian Academy, Series A, Volume 10, Number 1*, 2009.

[Coj08a] Dan Cojocar. BBUFs: A new lookup mechanism based on IPV6. In *Workshosp on Global Computing Models and Technologies, co-located with SYNASC 2008*, pages 358–361, 2008.

[Coj08b] Dan Cojocar. BBUFs: Synchronization mechanism. In *6th International Conference of Applied Mathematics (ICAM)*, pages 363–368, 2008.

[Coj09] Dan Cojocar. The Architecture of BBUFs. *KEPT2009 Knowledge Engineering Principles and Techniques*, Selected Papers:335–342, 2009.

[Coj10] Dan Cojocar. Bbufs: Routing protocol. In *Proceedings of the Symposium Colocviul Academic Clujean de Informatica*, pages 116–121, Cluj–Napoca, Romania, June 2010.

[Coj11] Dan Cojocar. Replication location decisions. In *Nano, Information Technology and Reliability (NASNIT), 2011 15th North-East Asia Symposium*, pages 161–165, Macao, China, October 2011.

[Coj12] Dan Cojocar. Load balance queries in decentralized peer-to-peer file systems. In *Proceedings of the National Symposium ZAC2012 (Zilele Academice Clujene, 2012)*, pages 105–110, 2012.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46–63, 2001.

[DH98] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFCs 5095, 5722, 5871, 6437, 6564, 6935, 6946.

[DKK+01] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with cfs. In *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, pages 202–215, New York, NY, USA, 2001. ACM.

[HGM10] Keith B Hall, Scott Gilpin, and Gideon Mann. Mapreduce / bigtable for distributed optimization. *Neural Information Processing Systems Workshop on Leaning on Cores Clusters and Clouds*, 1(1):1–7, 2010.

[HKM+88] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Trans. Comput. Syst.*, 6(1):51–81, 1988.

[HKRZ02] Kirsten Hildrum, John D. Kubiatowicz, Satish Rao, and Ben Y. Zhao. Distributed object location in a dynamic network. In *SPAA '02: Proceedings of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, pages 41–52, New York, NY, USA, 2002. ACM.

[HSW94] Yixiu Huang, Prasad Sistla, and Ouri Wolfson. Data replication for mobile computers. *SIGMOD Rec.*, 23(2):13–24, May 1994.

[Jac88] Van Jacobson. Congestion avoidance and control. In *ACM SIGCOMM Computer Communication Review*, volume 18, pages 314–329. ACM, 1988.

[KBC+00] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Not.*, 35(11):190–201, 2000.

[LCC+02a] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, ICS '02, pages 84–95, New York, NY, USA, 2002. ACM.

[LCC+02b] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, pages 84–95. ACM, 2002.

[LM09] Avinash Lakshman and Prashant Malik. Cassandra: structured storage system on a p2p network. In *Proceedings of the 28th ACM symposium on Principles of distributed computing*, pages 5–5. ACM, 2009.

[LNS96] W Litwin, M A Neimat, and D A Schneider. LH* - A scalable, distributed data structure. *ACM Transactions on Database Systems*, 21(4):480–525, 1996.

[LSSD02] Houda Lamehamedi, Boleslaw Szymanski, Zujun Shentu, and Ewa Deelman. Data replication strategies in grid environments. In *in Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02*, pages 378–383. Press, 2002.

[MHA02] Ravi K Madduri, Cynthia S Hood, and William E Allcock. Reliable file transfer in grid environments. In *Local Computer Networks, 2002.*

Proceedings. LCN 2002. 27th Annual IEEE Conference on, pages 737–738. IEEE, 2002.

[Mil99] Robin Milner. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, New York, NY, USA, 1999.

[MK03] Robert W Moss and Peter Korger. Methods and structure for read data synchronization with minimal latency, November 11 2003. US Patent 6,646,929.

[MMGC02] Athicha Muthitacharoen, Robert Morris, Thomer M. Gil, and Benjie Chen. Ivy: a read/write peer-to-peer file system. *SIGOPS Oper. Syst. Rev.*, 36(SI):31–44, 2002.

[MMP94] Drew Major, Greg Minshall, and Kyle Powell. An overview of the netware operating system. In *WTEC'94: Proceedings of the USENIX Winter 1994 Technical Conference*, pages 27–27, Berkeley, CA, USA, 1994. USENIX Association.

[NWO88] Michael N. Nelson, Brent B. Welch, and John K. Ousterhout. Caching in the Sprite network file system. *ACM Transactions on Computer Systems*, 6(1):134–154, 1988.

[Ora01] Andrew Oram. *Peer-to-peer: harnessing the benefits of a disruptive technology*. " O'Reilly Media, Inc.", 2001.

[PC04] Franjo Plavec and Tomasz Czajkowski. Distributed File Replication System based on FreePastry DHT. Technical report, University of Toronto, Ontario, Canada, 2004.

[PRR97] C. Greg Plaxton, Rajmohan Rajaraman, and Andrea W. Richa. Accessing nearby copies of replicated objects in a distributed environment. In *ACM Symposium on Parallel Algorithms and Architectures*, pages 311–320, 1997.

[RD01a] Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[RD01b] Antony Rowstron and Peter Druschel. Storage management and caching in past, a large-scale, persistent peer-to-peer storage utility. *SIGOPS Oper. Syst. Rev.*, 35:188–201, October 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM.

[SGK+85] Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and implementation of the Sun Network Filesystem. In *Proc. Summer 1985 USENIX Conf.*, pages 119–130, Portland OR (USA), 1985.

[SKRC10] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, pages 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[SMK+01] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[Tra95] P Traina. Bgp-4 protocol analysis. 1995.

[VM94] Björn Victor and Faron Moller. The Mobility Workbench — a tool for the π-calculus. In David Dill, editor, *CAV'94: Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 428–440. Springer-Verlag, 1994.

[WJH97] Ouri Wolfson, Sushil Jajodia, and Yixiu Huang. An adaptive data replication algorithm. *ACM Trans. Database Syst.*, 22(2):255–314, June 1997.

[XJZZ00] Dong Xuan, Weijia Jia, Wei Zhao, and Hongwen Zhu. A routing protocol for anycast messages. *Parallel and Distributed Systems, IEEE Transactions on*, 11(6):571–588, 2000.

[ZnWLY11] Han Zhi-nan, Yan Wei, Zhang Li, and Wang Yue. Design and implementation of an anycast efficient qos routing on ospfv3. 2011.