UNIUNEA EUROPEANĂ
GUVERNUL ROMÂNIEI MINISTERUL MUNCII, FAMILIEI, PROTECȚIEI SOCIALE ȘI PERSOANELOR VÂRSTNICE AMPOSDRU
Fondul Social European POSDRU 2007-2013
Instrumente Structurale 2007-2013
MINISTERUL EDUCAȚIEI NAȚIONALE
OIPOSDRU
Universitatea POLITEHNICA din București

FONDUL SOCIAL EUROPEAN
Investește în oameni!
Programul Operațional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013
Proiect POSDRU/107/1.5/S/76909 – *Valorificarea capitalului uman din cercetare prin burse doctorale (ValueDoc)*

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI

Facultatea de Automatică și Calculatoare

Departamentul Calculatoare

Nr. Decizie Senat 225 din 27.09.2013

TEZĂ DE DOCTORAT
REZUMAT

Extinderea Capabilităților Dispozitivelor Mobile prin Transferarea Sarcinilor în Cloud

Extending the Capabilities of Mobile Devices through Cloud Offloading

Autor: As.ing. Olteanu Alexandru-Corneliu

Conducător de doctorat: Prof.dr.ing. Nicolae Țăpuș

COMISIA DE DOCTORAT

Președinte	Prof.dr.ing. Adina Florea	de la	Universitatea Politehnica București
Conducător de doctorat	Prof.dr.ing. Nicolae Țăpuș	de la	Universitatea Politehnica București
Referent	Prof.dr.ing. Victor Patriciu	de la	Academia Tehnică Militară București
Referent	Prof.dr.ing. Ion Ivan	de la	Academia de Științe Economice București
Referent	Prof.dr.ing. Valentin Cristea	de la	Universitatea Politehnica București

București, 2013

Rezultatele prezentate în acestă teză de doctorat au fost obţinute cu sprijinul financiar
al Ministerului Muncii, Familiei şi Protecţiei Sociale prin Fondul Social European,
Programul Operaţional Sectorial Dezvoltarea Resurselor Umane 2007-2013,
Contract nr. POSDRU/107/1.5/S/76909.

Cuprins

Terminalele mobile moderne, precum telefoanele inteligente și tabletele, oferă funcționalități din ce în ce mai complexe. Le utilizăm zilnic în activități ce variză de la divertisment la îndeplinirea sarcinilor profesionale. Cu toate că dispozitivele mobile dobândesc o funcționalitate și o putere de procesare tot mai mare, considerăm că va crește și rolul unei infrastructuri mai puternice, care să amplifice capacitățile portabilelor. Trecând în revistă cele mai avansate progrese tehnologice, observăm faptul că atât dezvoltatorii, cât și cercetătorii studiază modalități de a accesa, de pe terminale mobile, resurse din Cloud, din infrastructura de calcul locală, precum și din alte portabile.

Accentul activității noastre este pus pe definirea unui spațiu de explorare destinat operațiunilor de offloading și pe utilizarea acestuia în evaluarea empirică și analitică a unor astfel de mecanisme pentru dispozitive portabile. În primul rând, consultăm eforturile curente de cercetare în materie de offloading folosind dispozitive mobile și propunem un Model și o Taxonomie Generală de Offloading. De asemenea, propunem un Spațiu de Explorare care ilustrează noi mecanisme de offloading, precum și oportunitatea explorării unui spațiu de proiectare. În al doilea rând, propunem un model pentru volumul de lucru destinat aplicațiilor sociale, atât la nivel macro - numărul de utilizatori - cât și la nivel micro - operațiile pe care le comandă utilizatorii. Colectăm proiecții din mii de jocuri găzduite de platforma Facebook și din mai multe aplicații mobile native, și le folosim pentru a caracteriza și modela diverse elemente din modelul volumului de lucru. În plus, demonstrăm maniera în care modelul nostru poate fi utilizat pentru a genera volume de lucru sintetice și pentru a ajuta dezvoltatorii să înțeleagă efectele pe care modificările din aplicațiile lor le pot avea asupra volumului de lucru. În al treilea rând, desfășurăm o analiză comparativă a mecanismelor de offloading pentru diverse aplicații mobile. Investigăm Adaptarea și Offloadingul Comunicației în cazul aplicațiilor pentru terminale mobile ce utilizează extensiile hardware personalizate, precum și Adaptarea și Offloadingul Calculului pentru aplicații bazate pe bucle de procesare. Mai propunem și o analiză operațională vizând descrierea mecanismelor de offloading pentru aplicații bazate pe bucle de procesare care, potrivit modelului nostru pentru volumul de lucru, includ aplicații sociale online. În al patrulea rând, efectuăm evaluarea performanțelor pentru diverse mecanisme de offloading. Prezentăm rezultate ale evaluării empirice pe baza unei aplicații din lumea reală, ale evaluării analitice bazată pe analiza operațională propusă de noi și le comparăm pentru a le demonstra validitatea.

Offloadingul pentru terminalele mobile este și va continua să fie o temă de larg interes, dat fiind faptul că suntem înconjurați de dispozitive personale interconectate. Am găsit, în comunitatea științifică, un puternic interes pentru modelul volumului de lucru propus de noi, ceea ce ne determină să figurăm îmbunătățiri viitoare, care vor rafina modelul și îi vor permite să acopere mai multe aplicații, cu o mai mare precizie. De asemenea, considerăm că spațiul de explorare propus de noi poate continua să servească drept bază pentru diverse evaluări empirice, atât cu ajutorul aplicațiilor reale, cât și al simulărilor.

Capitolul 1.

Introducere

Dispozitivele mobile moderne, de tipul telefoanelor inteligente și tabletelor, oferă experiențe tot mai complexe pentru utilizatori. Le folosim zilnic pentru activități de la cele axate pe divertisment la cele vizând îndeplinirea sarcinilor profesionale. Pe măsură ce vânzările portabilelor depășesc vânzările de calculatoare personale, încercarea de a aduce la același nivel progresele în materie de hardware și procesare mobilă și cerințele implicate de acest avânt uriaș al fenomenului portabilelor constituie o provocare atât pentru dezvoltatori, cât și pentru cercetători. Cu toate că terminalele mobile dobândesc tot mai multă funcționalitate și putere de procesare, credem că la fel va crește și rolul unei infrastructuri mai puternice, menită să amplifice capacitățile portabilelor. Astfel, cercetarea prezentată în această teză de doctorat se axează pe tehnici destinate operațiunilor de offloading de pe terminale mobile într-o infrastructură mai puternică de tip cloud.

1.1 Contextul Offloadingului[1] în Cloud pentru Mobile

Dispozitivele mobile moderne, cum ar fi telefoanele inteligente și tabletele, oferă portabilitate, putere crescută de procesare și capacități de comunicare. Astfel, acestea devin opțiuni atractive prin care utilizatorii să interacționeze unii cu alții, prin intermediul aplicațiilor sociale și, în cadrul propriului lor mediu de activitate, prin calcul omniprezent [1]. Facebook, care a anunțat de curând creșterea numărului lunar de utilizatori activi la peste 1 miliard, raportează că mai mult de jumătate din utilizatori le accesează rețeaua de socializare de pe un terminal mobil [2]. Pe de altă parte, în paralel cu progresele tehnologice la nivel de hardware și de procesare mobilă, cresc și cerințele utilizatorilor, întrucât aceștia se așteaptă să beneficieze de aplicații bogate în conținut, precum jocurile, și de acces la cantități mari de date din servere izolate, precum streaming multimedia. Dispozitivele mobile moderne au în continuare anumite limitări raportat la cerințelor utilizatorilor, în ce privește autonomia bateriei, capacitatea de stocare a datelor și disiparea căldurii. Astfel, este firesc să luăm în considerare faptul că portabilele, în ciuda puterii lor de procesare tot mai mari, continuă să folosească o infrastructură mai puternică.

Convergența dintre procesarea mobilă și calculul de tip cloud este studiată de câțiva ani și rămâne în continuare cap de afiș datorită dinamicii relevante în procesarea mobilă și provocărilor care continuă să apară. Pe măsură ce vânzările portabilelor depășesc vânzările de calculatoare personale, numeroși producători de hardware și software concurează pe piața terminalelor mobile. Companii precum Samsung, Nokia, HTC, Motorola, Apple, Acer și Asus produc dispozitive mobile cu diferite caracteristici hardware, cu o varietate de sisteme de operare, cum ar fi Apple iOS, Google Android sau Microsoft Windows. Lipsa de omogenitate a dispozitivelor mobile, la nivel de hardware și sisteme de operare, ridică dificultăți în calea eforturilor dezvoltatorilor de aplicații de a satisface toți utilizatorii mobili, aceștia fiind nevoiți să păstreze mai multe versiuni ale aceleiași aplicații. și furnizorii de procesare în Cloud sunt interesați de ajustarea propriilor sisteme pentru a

[1] Ne referim prin "offloading", conform literaturii de specialitate, la transferarea sarcinilor în medii de procesare din exteriorul dispozitivului mobil și îl descriem pe larg în Capitolul 2.

face față puternicelor variații ale numărului de utilizatori. În acest vast ecosistem, cercetătorii descoperă aspecte deosebite ale căror efecte pornesc de la experiența în utilizare până la optimizarea costurilor pentru furnizorii de resurse.

Transferarea sarcinilor în mediul cloud prin *offloading* reprezintă una din tendințele emergente în distribuția calculului efectuat de dispozitive mobile. Dezvoltatorii și cercetătorii deopotrivă, studiază modalități de accesare de pe terminalele utilizatorilor a puterii oferite de infrastructura de tip cloud în materie de procesare și stocare. Mediul cloud este utilizat de mult timp pentru furnizarea de offloading pentru computere. Totuși, în ultima vreme, dispozitivele mobile au încurajat progresele offloadingului de calcul și comunicație, cu accent pe compromisurile între beneficiile aduse de infrastructura puternică și costurile utilizării resurselor de la distanță, compromisuri la nivel de timp și fonduri necesare.

1.2 Formularea Problemei

Viziunea noastră se referă la dispozitive mobile interconectate, parte a unei ierarhii de dispozitive de procesare, ordonate în raport cu proximitatea acestora față de utilizator. Această ierarhie, discutată detaliat în Capitolul 2, conține, în esență, dispozitive wearable, dispozitive handheld, infrastructura locală și cloud. Latența este mai redusă în apropierea utilizatorului, însă resursele sunt mai bogate cu cât se mărește distanta. Astfel, se dorește găsirea unui echilibru între compromisurile ce guvernează această ierarhie.

Convergența procesării mobile cu calculul în cloud este studiată de câțiva ani, cu rezultate în proiectarea sistemelor[3][4], planificarea sarcinilor [5][6], descoperirea resurselor [7][8], etc. Integrarea dispozitivelor mobile în rândul altor tipuri computaționale capătă multe forme, cum ar fi *mobile cloud computing*, *offloading*, delegarea, *cyber foraging* și *data staging*. Ne axăm pe offloading, ca formă de transfer a sarcinilor de lucru, cu diferite nivele de granularitate, către resurse la distanță. Au fost propuse mai multe sisteme de offloading, sub formă de middleware, cadre de lucru sau servicii. Cu toate acestea, constatăm că puține demersuri ajung la stadiul de a avea un impact semnificativ asupra sistemelor și aplicațiilor reale.

Studiul offloadingului din surse la distanță este dificil din cauza lipsei de omogenitate a dispozitivelor și aplicațiilor. Suntem însă de părere că axarea pe un tip specific de aplicație poate aduce progrese pentru dispozitivele mobile, menținându-se totodată un domeniu larg de aplicabilitate. Astfel, abordarea noastră vizează efectuarea unei explorări a domeniului de aplicații și selectarea unei familii de aplicații pe care apoi să realizăm analiza și evaluarea diverselor mecanisme de offloading.

1.3 Provocări Principale și Întrebări de Cercetare

Offloadingul pentru terminale mobile face obiectul unei teme populare de cercetare, pe măsura popularității atinse de terminalele mobile propriu-zise în rândul publicului larg. Pornind de la activitățile timpurii de cercetare axate în mod specific pe dispozitive mobile și debutate în anii '90, în ultimii câțiva ani au fost publicate numeroase materiale pe tema offloadingului pentru surse portabile. Prima noastră provocare este să acoperim acest material divers, să îl organizăm și să identificăm oportunitatea contribuției noastre.

Multe dintre aplicațiile mobile moderne sunt destinate interacțiunii dintre utilizatori, de exemplu prin platforme de socializare precum Facebook, sau distribuirii de informații, de exemplu platforme de distribuire a fotografiilor digitale precum Instagram. în plus, majoritatea platformelor mobile dispun de o piață a aplicațiilor mobile, exemple fiind Google Play sau Apple iTunes, unde utilizatorii pot, de asemenea, interacționa când se decid dacă descărca o aplicație. Astfel, folosirea aplicațiilor mobile este în mare măsură influențată de relațiile dintre utilizatori, a doua provocare cu care ne confruntăm fiind analizarea comportamentului utilizatorilor și volumului de lucru din aplicații în contextul interacțiunilor sociale dintre aceștia.

Aplicații care utilizează interacțiunea dispozitivelor mobile cu mediul cloud sunt deja disponibile pe piață. Totuși materiale recente de cercetare au identificat cloudlet, resursele de procesare locale, ca o locație țintă a offloadingului [9], subliniind compromisul dintre costurile comunicației și procesării. Inspirați de această cercetare, plasăm portabilele într-o ierarhie de sisteme computaționale folosite de oameni în prezent, ierarhie compusă, în esență, din dispozitive wearable, dispozitive handheld, cloudlet și cloud. Dat fiind ritmul în care dispozitivele inteligente devin tot mai mici și mai integrate în rutina utilizatorului, prevedem posibilitatea că în viitorul apropiat portabilele vor funcționa nu ca surse și ca destinatari ai offloadingului, venit de la dispozitive mai mici și confortabil de purtat cu sine. Cea de-a treia provocare pe care o contemplăm este să decidem cum să realizăm procesul de offloading, în contextul aplicațiilor sociale online.

Recenta popularitate a portabilelor inteligente motivează numeroși producători să creeze dispozitive pentru multe categorii de consumatori, fapt ce determină ordine de mărime de eterogenitate în caracteristicile lor hardware și software. Procesorul principal poate varia de la versiuni mono-nucleu la modele cvadri-nucleu, și chiar la arhitecturi hibride care includ un procesor principal cu două nuclee și un procesor grafic. Autonomia bateriei poate varia de la zeci la sute de ore în modul de așteptare, fiind puternic influențată de comportamentul utilizatorului. A patra provocare cu care ne confruntăm este să evaluăm beneficiile offloadingului în condițiile acestei lipse de omogenitate.

Pe măsura acestor provocări propunem următoarele întrebări de cercetare:

- Care este spațiul de explorare destinat operațiunilor de offloading de pe terminale mobile? Care mecanisme noi de offloading pot veni în avantajul terminalelor mobile?

- Cum caracterizăm și modelăm volumele de lucru pentru aplicații mobile care beneficiază de pe urma procesului de offloading, știind că interacțiunile sociale dintre utilizatori au, de regulă, efecte majore asupra volumelor de lucru realizate prin acest tip de aplicații?

- Ce tip de aplicații mobile este adecvat studiului mecanismelor de offloading? Cum măsurăm progresul pe care un melanj de mecanisme de offloading îl poate înregistra în cazul diverselor dispozitive în condițiile unor volume reale de lucru?

- Ce îmbunătățiri pot aduce diferite mecanisme de offloading pentru diverse terminale în realizarea unor proiecții reale și sintetice ale volumelor de lucru?

1.4 Obiective Principale

Prezenta teză vizează definirea unui spațiu de explorare pentru operațiuni de offloading și utilizarea sa în analiza și evaluarea empirică a diferitelor mecanisme de offloading destinate terminalelor portabile. În acest sens este necesar să structurăm uriașul volum de informații din literatura de specialitate pe tema offloadingului pentru portabile, să investigăm modul în care diverse mecanisme de offloading funcționează în combinație cu mai multe tipuri de aplicații și să alegem un domeniu aplicativ pe marginea căruia să detaliem o selecție de mecanisme de offloading extrase din spațiul nostru de explorare. Astfel, principalele obiective ale tezei includ următoarele:

- studierea și organizarea numărului uriaș de aspecte privind offloadingul pentru terminale mobile, identificarea mecanismelor noi de offloading și definirea unui spațiu de explorare
- caracterizarea și modelarea volumelor de lucru specifice unui număr cuprinzător de aplicații, luând în considerare interacțiunile sociale dintre utilizatorii mobili, care au, de regulă, un efect semnificativ asupra acelor volume de lucru
- investigarea detaliată a modului în care diferite mecanisme de offloading operează în cazul câtorva aplicații și derivarea unui suport analitic ce poate fi extins să acopere o întreagă familie de aplicații
- utilizarea proiecțiilor volumelor de lucru și a suportului analitic pentru a realiza evaluarea analitică a mecanismelor de offloading în spațiul de explorare, și validarea concluziilor prin prisma evaluării empirice a unei aplicații din lumea reală.

1.5 Structura Tezei

În această secțiune prezentăm structura tezei și o succintă trecere în revistă a manierei în care ne abordăm obiectivele și ne realizăm propria contribuție, aceasta din urmă fiind rezumată în Secțiunea 6.1.

Teza conține patru capitole ce prezintă contribuția noastră originală, fiecare capitol abordând unul din cele patru obiective ale prezentei lucrări. Figura 1.1 rezumă relația dintre cele patru capitole, în timp ce restul acestei secțiuni o detaliază.

În Capitolul 2 conspectăm demersurile curente de cercetare privind offloadingul pentru portabile și propunem o fundamentare a conceptului de offloading, un Model General și o Taxonomie destinate operațiunilor de profil, dorind să structurăm vastele posibilități de realizare a procesului de offloading. De asemenea, identificăm două direcții principale de cercetare. Mai întâi, subliniem echilibrul dintre offloading și alternativele sale, precum adaptarea, și definim trei factori majori care guvernează acest echilibru: tipul de aplicație, tipul de offloading și ajustarea procesului de offloading. În a doua fază, ne structurăm studiul pe acești trei factori, propunem un Spațiu de Explorare care conturează un spațiu de proiectare, precum și oportunitatea de a indica noutatea la nivelul mecanismelor de offloading, cum ar fi offloadingul paralel și parțial.

În Capitolul 3 rezumăm eforturile noastre de a colecta proiecțiile aplicațiilor sociale online

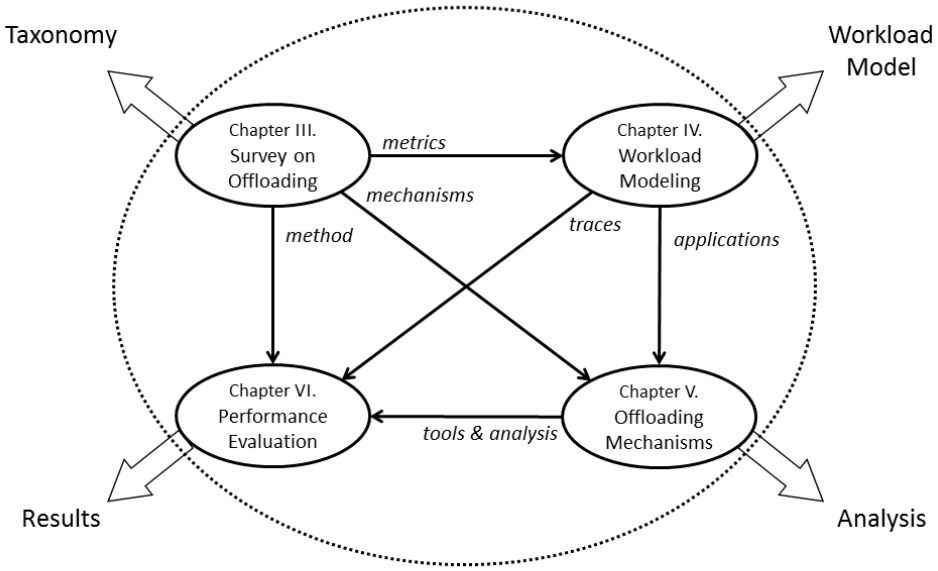


Figura 1.1: Relația dintre cele patru capitole principale ale tezei.

ce conțin informații referitoare la volumele de lucru la nivel macro (număr de utilizatori) și la nivel micro (operații), proiecții înregistrate în cazul a 16.000 de jocuri găzduite de platforma Facebook și mai multe aplicații mobile. De asemenea, propunem un model al volumului de lucru cu mai multe componente: Distribuția Popularității și Tiparul de Evoluție, pentru informația de la nivelul macro, Statistici la Nivel de Pachete și Diagrama Tranziției între Stări, pentru informații la nivel micro. Ulterior, prezentăm rezultatele obținute din caracterizarea și modelarea realizate cu ajutorul proiecțiilor aplicațiilor sociale online pe modelul volumului de lucru, demonstrând faptul că aplicațiile normale, cu o valoare unică de vârf a numărului de utilizatori, ajung la maturitate într-o etapă timpurie a duratei lor de viață, și că facilitățile de relaționare socială influențează tot mai mult experiența în utilizare a aplicațiilor online. În cele din urmă, arătăm felul în care modelul nostru pentru volumul de activitate poate fi util în prognozarea traficului și tipologiei de evoluție, precum și în generarea proiecțiilor sintetice.

Capitolul 4 prezintă eforturile noastre de investigare detaliată a unor mecanisme de offloading folosind câteva aplicații reale. Studiem Adaptarea și Offloadingul Comunicațiilor în cazul aplicațiilor pentru terminale mobile ce utilizează extensiile hardware personalizate, de tipul dispozitivelor cu senzori și rețelelor de automatizare a casei. Studiem, de asemenea, Adaptarea și Offloadingul Calculului pentru aplicații bazate pe buclă de procesare, cu accent pe procesarea video utilizând o aplicație de realitate augmentată, și pe randarea video, folosind un joc-simulator real. După aceea propunem un sistem formal pentru a facilita realizarea analizei operaționale pentru întreaga familie de aplicații bazate pe buclă de procesare, pentru mecanismele din Spațiul de Explorare definit în Capitolul 2.

În Capitolul 5 reluăm mecanismele ce alcătuiesc Spațiul de Explorare din Capitolul 2 pentru a realiza evaluarea performanțelor. Efectuăm evaluarea empirică folosind un *testbed* pe care l-am conceput și implementat pe baza unei aplicații din lumea reală, precum

și evaluarea analitică, pe baza sistemului formal propus în Capitolul 4 și a proiecțiilor volumului de lucru din Capitolul 3. Prezentăm rezultatele și le comparăm în vederea validării.

Capitolul 6 încheie teza menționând modul în care această lucrare se înscrie în rândul demersurilor curente pe tema activităților de offloading pentru terminale mobile. Rezumăm principalele contribuții ale activității noastre, cum ar fi definirea unui spațiu de explorare pentru mecanismele de offloading, modelarea volumelor de lucru pentru mii de aplicații, implementarea si investigarea detaliată a modului în care multiple mecanisme de offloading funcționează în câteva aplicații, felul în care această funcționalitate poate fi generalizată pentru o întreagă familie de aplicații prin intermediul analizei operaționale, și validarea prin comparație a rezultatelor analitice cu cele empirice. În cele din urmă, descriem direcțiile viitoare deschise de această activitate de cercetare. Modelul volumului de lucru poate fi îmbunătățit pentru a obține o acuratețe sporită și a acoperi un număr mai mare de aplicații. Poate fi conceput un model analitic complex, bazat pe rețele de cozi, destinat unei mai bune formalizări a modului în care sarcinile sunt îndeplinite într-un sistem de offloading. Nu în ultimul rând, predicțiile obținute din modelul volumului de lucru pot fi extinse cu posibile rezultate reflectate în planificarea și asigurarea capacității de lucru în infrastructuri de tip cloud.

Capitolul 2.

Aspecte ale Offloadingului pentru Mobile

Dispozitivele mobile moderne, de tipul telefoanelor inteligente și tabletelor, oferă portabilitate, putere computațională sporită și capacități de comunicare. Printre caracteristicile acestora amintim: conectivitate, capacități de procesare, capacități senzoriale, omniprezență, lipsă de omogenitate și autonomie limitată a bateriei. Autonomia limitată a bateriei și capacitățile reduse de procesare, cel puțin relativ la cerințele utilizatorului, reprezintă caracteristicile care stârnesc cel mai mult interesul în cercetarea conceptului de offloading. Conectivitatea sprijină procesul de offloading, în timp ce lipsa omogenității implică mai multe dificultăți.

Date fiind caracteristicile terminalelor mobile, acestea reprezintă o opțiune atractivă de interacționare între utilizatori, prin intermediul aplicațiilor sociale și, în cadrul propriului mediu de activitate, prin intermediul automatizării casei. Popularitatea dispozitivelor mobile poate fi observată în multe feluri. Facebook, care a anunțat de curând creșterea numărului lunar de utilizatori activi la peste 1 miliard, raportează că mai mult de jumătate din utilizatori le accesează rețeaua de socializare de pe un terminal mobil [10].

Oamenii folosesc dispozitive mobile zilnic în activități care variază de la divertisment la îndeplinirea sarcinilor profesionale. Aplicațiile mobile acoperă un vast domeniu operațional, fiind dezvoltate pentru scopuri diferite, cum ar fi jocuri, streaming multimedia, călătorii, comunicații, etc. Multe dintre aceste tipuri de aplicații au la bază conectivitatea și datele stocate în servere izolate. De asemenea, multe din acestea utilizează puterea computațională ridicată a dispozitivelor mobile. În cadrul acestui generos spațiu aplicativ există mai multe tipuri de aplicații care ar beneficia de pe urma procesului de *offloading* (de transfer al sarcinilor în cloud):

- aplicații cu conținut computațional intensiv (de ex. șahul)
- aplicații care se bazează pe date provenite de la server (de ex. recunoașterea obiectelor)
- aplicații cu procesare de tip randare (de ex. procesare de imagini)
- aplicații care interacționează deja cu mediul cloud (de ex. m-commerce)

Cu toate că funcționalitatea și puterea computațională a dispozitivelor mobile avansează, considerăm că va crește și rolul unei infrastructuri mai puternice, care să amplifice capacitățile portabilelor, date fiind limitările de performanță ale bateriei și disipării căldurii, cererile tot mai complexe ale utilizatorilor și interacțiunile sociale dintre aceștia. După cum se subliniază în [11], convergența procesării mobile cu serviciile bazate pe medii cloud este una din modalitățile cele mai sigure prin care calculul în cloud va evolua în viitorul apropiat.

Aplicațiile care utilizează interacțiunea dispozitivelor mobile cu mediul cloud sunt deja disponibile pe piață. Totuși materiale recente de cercetare au identificat conceptul de cloudlet, adică resursele disponibile local, ca o locație-țintă a offloadingului [9], subliniind compromisul dintre costurile comunicațiilor și cele ale procesării. Inspirați de această cercetare, plasăm portabilele într-o ierarhie de sisteme computaționale folosite de oameni

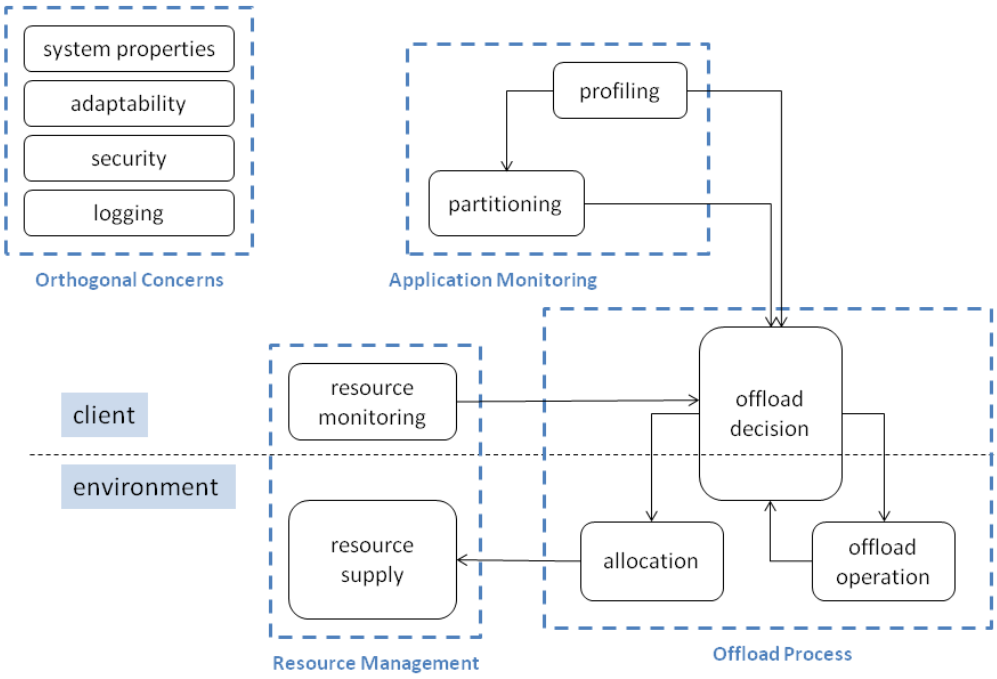


Figura 2.1: Modelul General de Offloading.

în prezent, ierarhie compusă, în esență, din dispozitive wearable, dispozitive handheld, cloudlet (infrastructura de calcul locală) și cloud.

Trecem în revistă activitățile curente de cercetare vizând offloadingul pentru terminale mobile. Propunem un Model General de Offloading și o Taxonomie aferentă aspectelor de Offloading. De asemenea, identificăm direcții de cercetare. Subliniem echilibrul dintre offloading și adaptare, ca alternativă sau procedură complementară offloadingului. În plus, scoatem în evidență oportunitatea unor noi mecanisme de offloading, precum offloadingul parțial și paralel, precum și importanța explorării spațiului de proiectare.

2.1 Modelul General de Offloading

În această secțiune prezentăm un model general de offloading, ce descrie un sistem generic de offloading. Realizăm un tablou de ansamblu al fiecărei componente cheie implicate în offloading și vom detalia funcționalitatea lor pe baza unei taxonomii în Secțiunea 2.2.

Împărțim componentele unui sistem de offloading în două categorii: componente ale clientului – dispozitivul mobil, și componente ale mediului de activitate – fie el un cloud, un *cloudlet* (bazat pe infrastructura locală), un dispozitiv similar al altui utilizator sau un mediu hibrid. Componentele sunt descrise în Figura 3.1 și detaliate în restul acestei secțiuni.

Multe dintre actualele activități de cercetare se axează pe înțelegerea aprofundată a aplicației ce urmează să facă obiectul offloadingului. Prin urmare, componenta de *Monitorizare a Aplicației* din sistemul de offloading reprezintă elementul cheie pentru obținerea unui proces benefic de offloading. Componenta de Profilare are capacitatea de a deter-

mina modul în care aplicația funcționează prin intermediul a diverse mecanisme, cum ar fi analiza statică sau cea dinamică. Informațiile obținute de la componenta de Profilare pot fi utilizate de componenta de Partiționare, care urmărește să dividă aplicația în componente cu granularitate predefinită și să le identifice pe acelea ce pot fi transferate.

Există și necesitatea de a evalua resursele, atât cele locale cât și cele izolate, disponibile pentru operarea aplicației. Astfel, componenta de *Gestiune a Resurselor* acoperă atât categoria aferentă clientului, cât și cea specifică mediului de activitate. În categoria componentelor clientului, componenta de Monitorizare a Resurselor evaluează parametrii precum nivelul bateriei, sarcina procesorului central, calitatea conexiunii fără fir și așa mai departe. În categoria componentelor din mediu, componenta Furnizării de Resurse gestionează resursele externe care pot fi folosite în offloading, prin intermediul mecanismelor de genul planificării și asigurării resurselor. Descoperirea resurselor este utilă în abordări oportuniste, unde dispozitivul mobil încearcă să găsească ținte disponibile de offloading în mediul de funcționare.

Asigurarea resurselor reprezintă o abordare proactivă, utilizată frecvent în medii cloud, unde resursele sunt create dinamic pentru ajustarea necesităților computaționale. *Procesul de Offloading* trebuie să fie unul iterativ, date fiind mobilitatea și natura schimbătoare a condițiilor de execuție. Spre exemplu, clientul mobil poate comuta de pe WiFi pe 3G, sau poate atinge un nivel critic al bateriei, cu consecințe asupra procesului de offloading. Componenta de Decidere a Transferului de Sarcini primește informații de la componentele de Monitorizare a Aplicațiilor și de Gestiune al Resurselor în scopul evaluării necesităților și condițiilor curente de offloading, precum și de la reiterările precedente al Operațiunii de Offloading pentru a evalua beneficiile și deficiențele acelor reiterări. Componenta de Decidere a Transferului de Sarcini poate alege:

- conținutul pentru Offloading, dintre seturile de partiții oferite de componenta de Partiționare
- momentul offloadingului, întrucât uneori operațiunea de Offloading nu își are rostul
- locația offloadingului, și poate instrui componenta de Alocare să utilizeze ținta adecvată de offloading.

Pe lângă procesele de offloading de bază, activitățile de cercetare se adresează, de asemenea, unei serii de *Aspecte Ortogonale*. Unele abordări se axează pe adaptabilitate: un joc poate dezactiva animațiile dacă sistemul de offloading nu reușește să mențină o frecvență satisfăcătoare a cadrelor redate, sau dispozitivul poate comuta între rețelele de comunicație prin procedeul de *handover* . Offloadingul poate releva și aspecte de securitate prin transferul către resurse izolate ce aparțin, de regulă, terților. Jurnalizarea poate fi utilizată în scopuri de evidență, taxare și monitorizare a întregului proces.

2.2 Taxonomia Offloadingului pentru Dispozitive Mobile

În această Secțiune prezentăm o nouă taxonomie destinată aspectelor de offloading, taxonomie structurată pe modelul prezentat în Secțiunea 2.1. Modelul identifică patru domenii majore de teme specifice procesului de offloading: Monitorizarea Aplicațiilor, Gestiunea Resurselor, Procesul de Offloading și Aspecte Ortogonale. Fiecare dintre aceste domenii

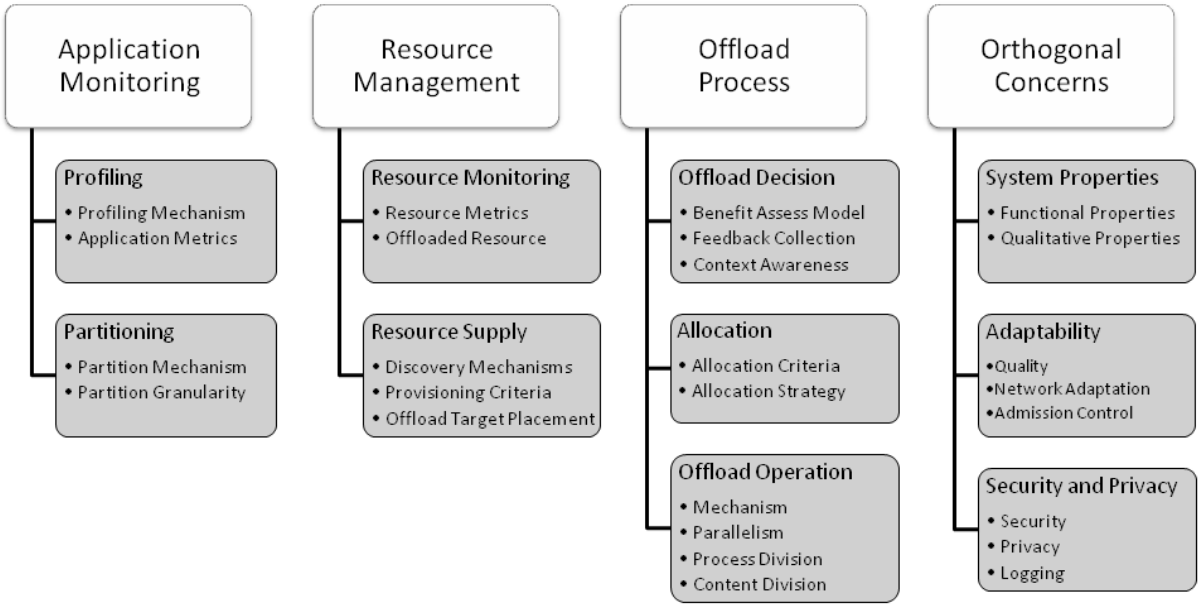


Figura 2.2: Taxonomia noastră pentru preocupări de offloading.

include o serie de puncte cheie de interes, fiecare cu mai multe subpuncte. Figura 2.2 prezintă taxonomia noastră pe trei nivele. O prezentăm pe scurt în această secțiune și facem referire la teza completă pentru detalirea sa.

A. Monitorizarea Aplicațiilor

Numeroase activități de cercetare pe tema offloadingului pentru terminale mobile se bazează pe doar câteva aplicații, simplificând astfel domeniul Monitorizării Aplicației. Totuși, există unele cercetări care urmăresc identificarea unei soluții automatizate, care ar funcționa în cazul multor tipuri de aplicații, și ar necesita, astfel, să folosească mecanisme complexe.

Majoritatea cercetătorilor [12][13][3] utilizează o combinație de analiză statică și dinamică. În plus, unele lucrări [14][15][4] apelează la profilarea online, adică monitorizarea comportamentului aplicației în timpul offloadingului. Partiționarea poate utiliza o serie de algoritmi, cum ar fi cei de clustering. Majoritatea activităților de cercetare ce vizează profilarea aplicațiilor pentru offloading măsoară, pentru fiecare componentă în parte, sarcina procesorului central și alți parametri specifici de performanță, cum ar fi memoria utilizată, consumul de energie și timpul de funcționare. Unele lucrări [16][17][18] apelează și la metrici statistice, precum numărul de invocări înregistrate de o componentă și cantitatea de date pe care aceasta o comunică. Pentru partiționare, unele soluții se bazează pe adnotarea codului de către dezvoltator, în timp ce altele se străduiesc să distingă componentele printr-o abordare automată.

B. Gestionarea Resurselor În literatura de specialitate studiată se poate remarca faptul că abordările oportuniste de offloading se bazează frecvent pe colocație, întrucât dispozitivele mobile vor utiliza puterea computațională disponibilă în vecinătatea acestora. De poate, de asemenea, menționa faptul căputține lucrări abordează problematica furnizării resurselor, în timp ce mulți cercetători preferă să definească static resursele folosite în experimente.

Metricile Resurselor reflectă metricile aplicațiilor prezentate la A.1.2. și sunt utilizate în procesul de luare a deciziei de offloading. Majoritatea aplicațiilor monitorizează sarcina pe procesor [16][14][19] și nivelul bateriei dispozitivului [19][20][21]. La fel de multe măsoară și latența de rețea [20], utilizarea lățimii de bandă sau timpul de răspuns [22][16]. Huerta [14] și Yan [23] mai iau în calcul și costurile monetare ale utilizării resurselor de la distanță. De asemnea, sunt folosite diferite mecanisme de descoperire a resurselor. Zhang et al [20] subliniază două scenarii care necesită resurse masive în locații limitate, mai exact ajutorarea în caz de dezastre naturale și găsirea copiilor dispăruți. În lucrarea prezentată de Marin et al [21], o componentă denumită Receptor Cloud acționează ca agent intermediar în luarea deciziei privind resursa din cloud care trebuie folosită.

C. Procesul de Offloading

În cadrul Procesului de Offloading ne referim, printre altele, la aspecte ce țin de evaluarea avantajelor, alocarea pe resurse și mecanismul de offloading propriu-zis. Pentru evaluarea avantajelor, majoritatea cercetătorilor iau în calcul performanța dispozitivului mobil cu și fără offloading, exprimată ca timpi de funcționare. Alții optimizează și consumul de energie, precum [24][25][20], sau costurile monetare, precum [22][23][20]. Satyanarayanan[9] și Verbelen[13] fac, de asemenea, referire la calitatea rezultatului după offloading, cum ar fi o rezoluție mai bună a imaginilor.

Ferber et al. [22] investighează pierderile de timp la alocare și procesare folosind resurse la distanț a, în cazul soluției Amazon EC2. Alți cercetători adoptă diverse alte criterii de alocare, precum resursa izolată cel mai puțin ocupată [22][26], sau locația datelor ce urmează a fi procesate [24][16][17].

Există o serie de mecanisme prin care se poate face offloading. Lagerspetz et al [25] descriu procesul de offloading ca execuție la distanță, folosind un serviciu din cloud. Zhang et al [20] propun un model aplicativ pe baza componentelor portabile ale aplicațiilor, denumite weblets. În [9], autorii folosesc mutarea și sinteza dinamică de mașini virtuale pentru a reproduce activitatea. Divizarea procesării și a datelor, precum și paralelismul, sunt încă subiecte care nu au fost tratate explicit în evaluarea diferitelor mecanisme de offloading.

D. Aspecte Ortogonale

Studierea lucrărilor care tratează Aspectele Ortogonale în momentul transferului de sarcini relevă faptul că foarte puține activități de cercetare abordează toate tipurile de aspecte ortogonale, însă multe din acestea abordează cel puțin un aspect.

Referitor la Adaptabilitate, în [27], autorii descriu schimbarea automată a rețelei de comunicație, pe baza conștientizării locației, încărcarii rețelei și a predicțiilor de acțiune ale utilizatorilor. Ca un alt exemplu, în [28] autorii propun o tehnică de adaptare a randării, cu accent pe experiența în utilizare.

Kumar et al [24] discută mai multe aspecte de confidențialitate legate de utilizarea resurselor de tip cloud, făcând referire la accesarea datelor de către furnizori terți și identificarea locației. Eom et al [12] propun utilizarea conceptului SocialVPN, o tehnică de tunneling prin intermediul unor Rețele virtuale Private, ce vine cu două avantaje: securitate sporită și virtualizare. Mai recent, GCM, prevede jurnalizarea în scopuri de păstrare a evidenței și de taxare.

2.3 Direcții de Cercetare

Offloadingul pentru terminale mobile este un subiect de mare interes. Are o prezență crescută în publicațiile analizate de utilizatori și în evenimentele din ultimii ani, într-o manieră destul de similară cu popularitatea tot mai crescută a dispozitivelor mobile în rândul publicului larg.

De asemenea, din ce în ce mai vizibile sunt și progresele aplicațiilor mobile derivate din procesul de offloading. Spre exemplu, tehnica de offloading a comunicațiilor prezentată de Kemp [29] este implementată comercial și de furnizori importanți de servicii mobile, după cum se menționează în [30]. Tot în sfera pieții, aplicații precum Shazam folosesc procesarea de la distanță ca modalitate standard de funcționare. În ciuda celor de mai sus, numeroase activități de cercetare se confruntă în continuare cu o serie de dificultăți care le întârzie implementarea în folosul publicului larg. Identificăm, în restul acestei secțiuni, direcții de cercetare pentru offloadingul destinat terminalelor mobile cu potențial semnificativ de explorare viitoare.

Monitorizarea Aplicațiilor: mare parte din cercetare se concentrează pe modalități tot mai mult automatizate de partiționare a aplicațiilor la nivel operațional. Considerăm că monitorizarea aplicațiilor poate fi înțeleasă și din perspectiva volumului de lucru, oferind rezultatele precum predicțiile de încărcare, acestea permițând, la rândul lor, o mai bună asigurare a resurselor și operarea unor sisteme de offloading mai inteligente.

Gestiunea Resurselor: descoperirea și asigurarea resurselor sunt subiecte prea puțin abordate. Multe dintre abordări fac referire la offloadingul oportunist, căutarea oricăror resurse, cyber foraging și așa mai departe. Cu toate acestea, furnizorii de resurse, având experiență în descoperirea resurselor și asigurarea acestora în medii cloud, sunt în mod cert interesați să găsească modalități de furnizare eficientă a propriilor resurse pe piața de software pentru portabile, piață ce numără sute de milioane de utilizatori.

Procesul de Offloading: se pot efectua cercetări pe tema offloadingului parțial și paralel, și se poate exploata regiunea de interes. În plus, un sistem experimental poate fi mai bine ajustat pentru aplicații din viața reală: experimente cu clienți puțini, cu utilizarea laptopului și nu a portabilelor.

Aspecte Ortogonale: adaptarea face des pereche bună cu offloadingul și uneori reprezintă chiar o alternativă superioară. Obținerea unui echilibru ideal între offloading și adaptare poate reprezenta o abordare nouă pentru optimizarea performanței.

Adaptare versus Offloading: o Alegere Echilibrată

Taxonomia noastră identifică la punctul D.2. capacitatea de adaptare a sistemului ca aspect ortogonal cheie. Adaptarea poate fi realizată la nivelul calității rezultatelor, accesului la rețea sau controlului accpetării sarcinilor în sistem. Când offloadingul nu aduce avantaje, în funcție de aplicație, sistemul poate realiza o adaptare a calității. De exemplu, într-un joc poate fi preferată diminuarea calității grafice sau într-o aplicație de procesare video se poate recomanda doar procesarea anumitor cadre.

Adaptarea poate fi utilizată în mod complementar sau ca înlocuitor pentru offloading. Pentru a determina care tehnică este mai adecvată într-o mulțime vastă de situații, luăm

în considerare factorii cu cea mai semnificativă influență asupra avantajelor offloadingului:

- *Tipul de Resursă:* după cum este descrisă în taxonomia noastră la punctul B.1.1., comunicarea, calculul și conținutul pot fi în întregime transferate. Adaptarea comunicării se poate realiza și în sisteme de tip client-server, limitând numărul de căutări efectuate de client. Adaptarea calculului poate fi efectuată, de exemplu, într-un joc prin scăderea calității grafice. Investigăm adaptarea și offloadingul comunicației și calculului în secțiunea 4.

- *Tipul de Aplicație:* am identificat deja mai multe tipuri de aplicații care ar putea beneficia de pe urma adaptării. Acestea sunt orientate cu precădere spre procesarea imaginilor și grafică, dar și spre transferul multor mesaje. Pe de altă parte, aplicații care ar fi avantajate de offloading sunt, de obicei, cele care prezintă deja un anumit tip de funcționalitate distribuită, cu volum computațional ridicat sau care depind de datele disponibile de la distanță. Investigăm aplicații care respectă cele două criterii în Capitolul 4.

- *Ajustarea Procesului:* după cum am arătat în taxonomia noastră, la punctul C.3., procesul aplicațiilor mobile poate fi ajustat pentru offloading, exploatând caracterul parțial și paralelismul. Adaptarea exploatează inerent caracterul parțial al procesului, permițând funcționarea anumitor porțiuni din aplicație, însă paralelismul nu se acplică în acest caz. Astfel, în aceste condiții, offloadingul demonstrează o complexitate mai mare decât adaptarea.

În Capitolul 4, efectuăm o explorare a domeniului aplicativ, unde investigăm offloadingul comunicației și calculului în contextul mai multor aplicații. Unele elemente ale celui de-al treilea factor, ajustarea procesului, sunt asociați și caracterului inerent al adaptării și offloadingului. Totuși, pentru a acoperi aspecte multiple ale celui de-al treilea factor, definim un Spațiu de Explorare ce va fi utilizat în Capitolul 5, în Evaluarea Experimentală a Sistemului Integrat de Offloading propus de noi.

Spațiul de Explorare

Pe baza înțelegerii noastre a actualelor activități de cercetare în domeniul offloadingului pentru portabile, propunem un spațiu de explorare care vizează subiecte ce deschid calea unor oportunități interesante de cercetare. Figura 2.3 prezintă un spațiu de explorare cu cinci dimensiuni:

- *Componentă*: folosind Granularitatea Partiției de Aplicație, aplicația poate fi transferată într-o măsură mai mică sau mai mare, de la o singură componentă la majoritatea acestora, lăsând date client minime în dispozitivul mobil. Ca exemple pentru a doua situație menționăm diverșii clienți VNC pentru dispozitive mobile și tehnologii de streaming pentru jocuri.

- *Intermitență*: conform descrierii Diviziunii Procesului în taxonomia noastră, procesul poate fi transferat parțial în manieră spațială sau temporală. Pentru a acoperi aspectul temporal, ne vom referi la offloadingul intermitent.

- *Divizarea Datelor*: conform descrierii Diviziunii Datelor în taxonomia noastră, datele pot fi transferate integral sau parțial, fie la nivel de obiect de interes, fie la cel

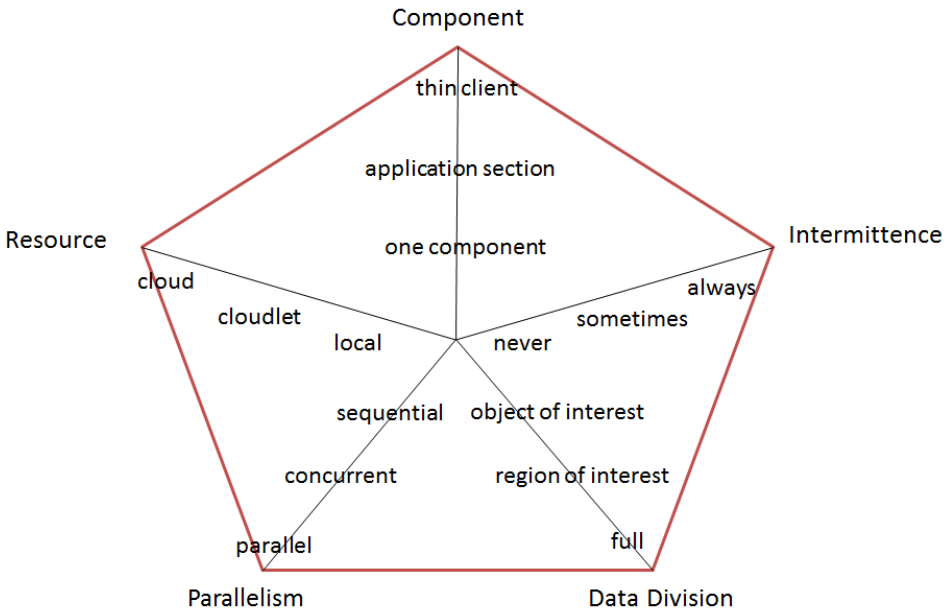


Figura 2.3: Spațiul de Explorare al offloadingului pentru mobile

de regiune de interes

- *Paralelism*: conform descrierii Paralelismului din taxonomia noastră, offloadingul poate fi realizat în paralel pentru anumite sarcini. De asemenea, offloadingul simultan poate fi interesant pentru studii ca metodă de maximizare a performanței cu reducerea costurilor
- *Amplasarea Resurselor*: După cum am descris Amplasarea Resurselor în taxonomia noastră, folosirea de resurse din mediile cloud, cloudlet (infrastructura locală), sau chiar ale altor dispozitive din vecinătate, relevă câteva compromisuri interesante.

Combinații diferite ale celor cinci dimensiuni pot produce rezultate interesante. Încă nu a fost realizată o comparație a diverselor nivele de transferare a componentelor în contextul diverselor resurse disponibile. De asemenea unele metode de ajustare a offloadingului pot dovedi că încă mai pot fi relevate noutăți ale procesului de offloading.

Din datele pe care le deținem, niciun sistem de Offloading nu a fost testat în condițiile unor volume de lucru reale produse de zeci de milioane de utilizatori. Astfel, offloadingul secvențial către resurse izolate, deja util întrucât acele resurse sunt furnizate de infrastructura puternică de tip cloud, a fost până acum suficient. Intenționăm să investigăm offloadingul paralel ca metodă superioară de Offloading sub presiunea volumelor de lucru, după cum o demonstrează cele mai populare aplicații mobile din ziua de azi. De asemenea, offloadingul simultan poate fi exploatat și în condiții experimentale.

Offloadingul procesării parțiale și cel al datelor parțiale au fost studiate, dar cu rezultate limitate. Credem că axarea pe un domeniu specific de aplicație poate furniza o cunoaștere mai aprofundată și o înțelegere adecvată a offloadingului parțial. De exemplu, în cazul jocurilor, am putea investiga nivelul de parțialitate al regiunii de interes aplicând domeniul conceptual de interes în cadrul experimentelor de offloading, am putea investiga offloadingul spațial parțial prin divizarea unei hărți și am putea investiga offloadingul temporal parţial procesând izolat doar unele cadre.

Capitolul 3.

Modelarea Volumului de Lucru pentru Aplicații Sociale Online

Credem că offloadingul poate fi benefic pentru aplicațiile care (1) deja utilizează o formă oarecare de comunicație, și (2) își pun în practică funcționalitatea reiterând o buclă de execuție. Luând în calcul și popularitatea de care se bucură diverse tipuri de aplicații, am decis să investigăm în continuare Aplicațiile Sociale Online, aplicații dedicate utilizatorilor interconectați social și dezvoltate în scopuri diverse cum ar fi jocuri, streaming multimedia, călătorii, comunicare etc.

Aplicațiile sociale online sunt în prezent implementate prin utilizarea unor tehnologii variate:

- Web 2.0: exemple sunt aplicațiile găzduite de platforma Facebook și dezvoltate de către un întreg ecosistem de companii; astfel de aplicații transmit către server mesaje cu acțiunile utilizatorului ce urmează a fi realizate; aceste aplicații fac legătura cu milioane de utilizatori numai de pe terminale mobile [10]
- Simulări Strâns Cuplate: de exemplu, jocurile de simulare multiplayer, precum OpenTTD, care determină o procesare intensă în dispozitiv și comunică, de regulă, numai mesaje de comandă; numai jocul OpenTTD are peste 100.000 de utilizatori pe platforma Android
- Streaming: exemple sunt platformele de aplicații sociale online și de streaming video, precum OnLive, care determină procesare masivă în server trimițând fluxul video și audio către dispozitiv; în 2012 numai OnLive a raportat un număr de 1,5 milioane de utilizatori activi [31]

Aplicații din toate cele trei categorii tehnologice dispun de procesare în buclă, unele etape fiind realizate la distanță ca urmare a caracteristicilor online și sociale specifice. În plus, relațiile sociale dintre utilizatori determină acumularea unor volume de lucru consistente care se pot dovedi pentru toate dispozitivele de calcul implicate.

După cum s-a explicat în Secțiunea 2.3, considerăm că monitorizarea aplicațiilor poate fi înțeleasă și din perspectiva volumului de lucru, oferind rezultatele precum predicțiile de volum de lucru, acestea permițând, la rândul lor, o mai bună asigurare a resurselor și operarea unor sisteme de offloading mai inteligente. Astfel, în acest capitol ne concentrăm pe modelarea volumului de lucru la nivelul aplicațiilor sociale online și vom folosi rezultatele prezentate aici în efectuarea evaluării performanțelor mai multor tehnici de offloading, prezentate în Capitolul 5.

3.1 Colectarea Datelor

Pentru acest studiu am colectat mai multe seturi de date de mari dimensiuni provenite din aplicații sociale online, după cum ilustrează Tabelul 3.1. Obiectivul este de a deține seturi de date care acoperă intervale îndelungate de timp și un număr mare de utilizatori,

precum și provenite de la aplicații ce rulează cu tehnologii diferite. Procesul de colectare a datelor este, totuși, dificil deoarece dezvoltatorii de aplicații, în general, nu doresc să distribuie informații de utilizare legate de produsele proprii. Asemenea informații pot avea efecte negative asupra unor elemente precum cota de piață sau opinia utilizatorilor. Dispunem, astfel, de o colecție extrem de variată de seturi de date, dintre care numai unele au fost adecvate caracterizării și modelării prezentate în Secțiunea 3.3.

Tabela 3.1: Privire de ansamblu a seturilor de date.

Dataset	Type	total # of applications	total # of users	Duration
facebook-1	users count operations	630 18	10M 2	3 years 60 sessions
facebook-2	users count	16,800	10M	3 years
open-ttd	operations	1	10 (AI)	100 sessions
tune-up	users count operations	500	15,000	9 months 1000 sessions

Seturile de date *facebook-1* și *facebook-2* corespund aplicațiilor online, în timp ce seturile *open-ttd* și *tune-up* corespund unor simulări strâns cuplate. Pentru fiecare dintre seturile de date am trecut printr-un întreg proces de colectare a datelor, care constă în extragere, stocare și sanitizare.

Colectăm două tipuri de seturi de date: contorizare număr de utilizatori și operațiuni. Seturile de date privind contorizarea numărului de utilizatori înregistrează numărul de utilizatori activi. Folosim în activitatea noastră denumirile date de platforma Facebook propriilor indicatori de utilizare: utilizatori activi pe zi (DAU) și utilizatori activi pe lună (MAU). Seturile de date ale operațiunilor constau în operațiuni declanșate de utilizatori, cu efecte asupra sarcinii de calcul și comunicației. Cele două tipuri de seturi de date corespund celor două secțiuni principale ale modelului nostru pentru volumul de lucru, Modelul Macro și, respectiv, Modelul Micro, după cum acestea sunt prezentate în Secțiunea 3.2.

3.2 Modelul Volumului de Lucru

Pentru a facilita înțelegerea sarcinilor din aplicațiile sociale online, am creat un model de evoluție a volumului de lucru (vezi Figura 3.1). Elementul central al modelului nostru este evoluția, fie pe termen scurt (la nivelul unei sesiuni de lucru, adică până la câteva ore, cu o rezoluție de ordinul milisecundelor), fie pe termen lung (la nivelul duratei de existență a unei aplicații, ajungând astfel până la câțiva ani, cu o rezoluție de o zi). Pentru a trata cele două intervale de timp și grade de acuratețe total diferite, propunem componente separate de model:

- o componentă de macro-evoluție pentru intervalul îndelungat, care definește numărul de utilizatori activi înregistrat de aplicații, constând în două elemente: distribuția popularității - care descrie modul în care utilizatorii sunt distribuiți în cadrul unei mulțimi de aplicații - și tiparul de evoluție - care descrie gradul de variație a numărului de utilizatori pe parcursul duratei de viață al unei aplicații
- o componentă de micro-evoluție pentru intervalul scurt, care definește operațiunile declanșate de utilizatori, împreună cu sarcina lor aferentă, constând în două ele-

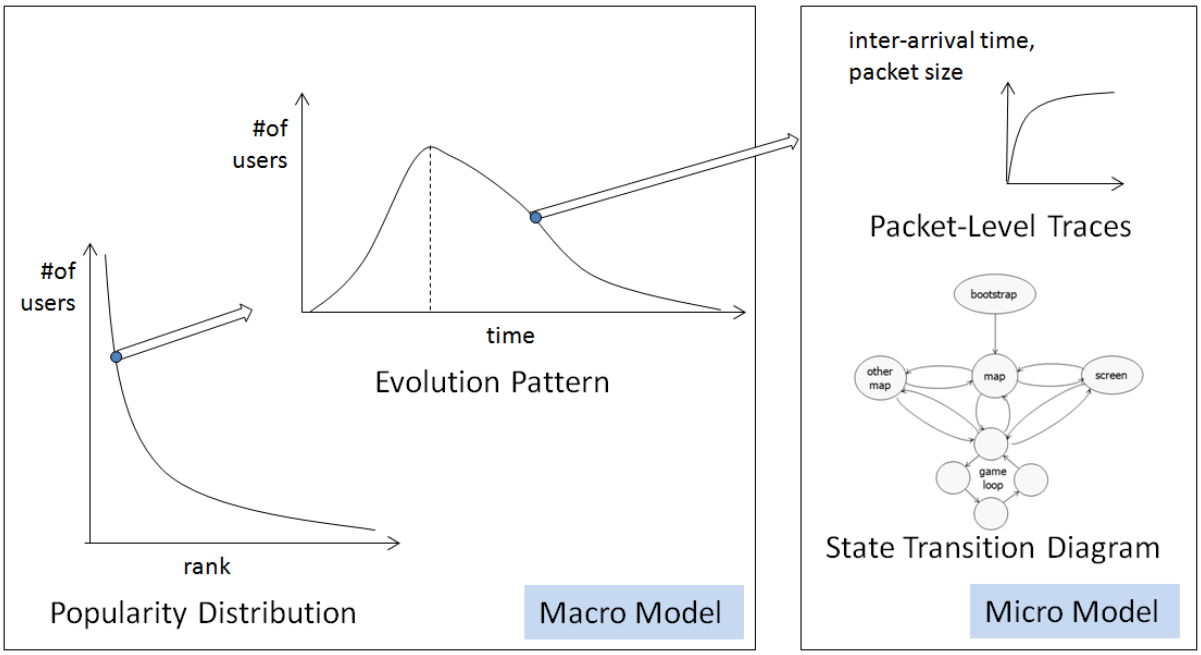


Figura 3.1: Modelul volumului de lucru.

mente: statisticile generale de rețea - cu detalii aprofundate privind traficul în rețea și timpii de procesare - și digrama tranziției între stări - care segmentează sarcinile în stări diferite ale aplicațiilor.

Detaliem modelul nostru în restul acestei secțiuni și prezentăm rezultatele modelării și caracterizării în Secțiunea 3.3, folosind seturile de date descrise în Secțiunea 3.1.

Distribuția de Popularitate

Pentru a cuantifica popularitatea unei aplicații într-o mulțime de aplicații similare, investigăm relația dintre numărul maxim de utilizatori DAU și nivelul ierarhizat al aplicației. Ne așteptăm ca această relație să urmeze un principiu de tip Pareto, unde aplicațiile din vârful ierarhiei însumează mult mai mulți utilizatori, chiar și în comparație cu cele clasate imediat după acestea.

Realizăm modelarea distribuțiilor empirice cu ajutorul a diferite distribuții de referință: Exponențială, Weibull, Pareto, Lognormal, și Gama. Utilizăm metoda estimării maximei probabilități (MLE) pentru a estima parametrii de distribuție, precum și testul Kolmogorov-Smirnov (KS) pentru a evalua indicele de potrivire (GOF).

Tiparul de Evoluție

Am identificat mai multe tipologii majore de evoluție, fiecare din acestea fiind guvernată de tipare parametrizate de evoluție. Dezvoltăm un model pentru fiecare tipar, precum și un clasificator algoritmic care mapează o aplicație dată în cadrul tipologiei sale evolutive. Validăm modelul nostru în Secțiunea 3.3.

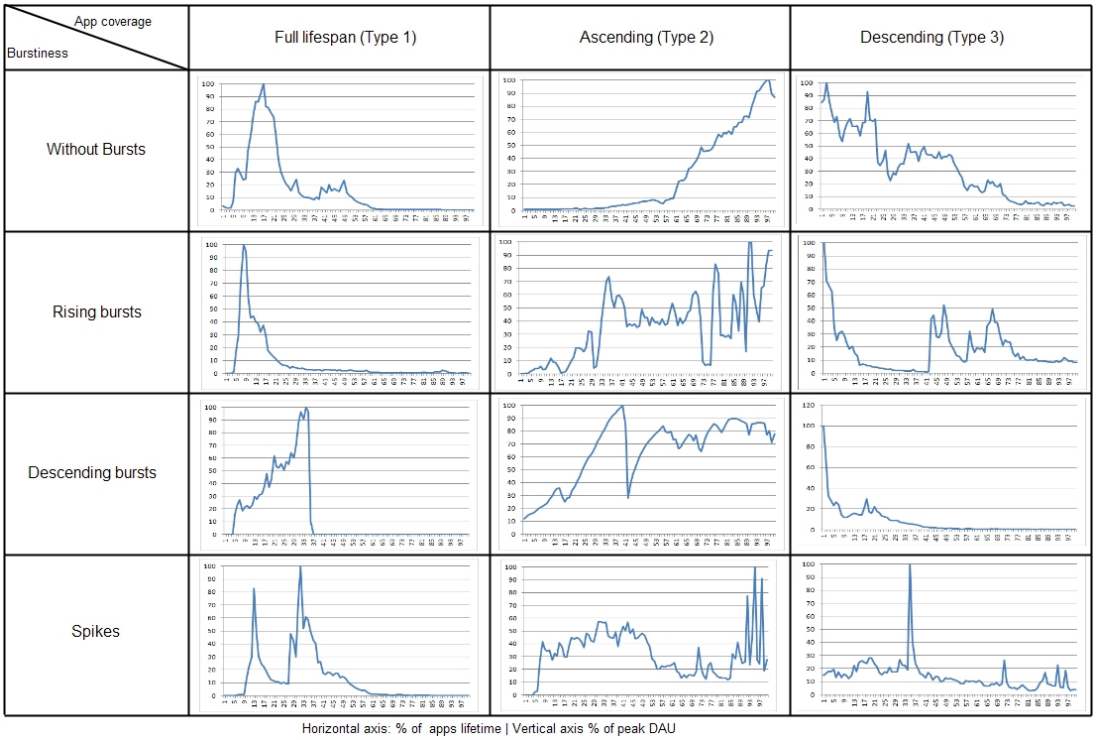


Figura 3.2: Clase și tipologii de evoluție.

Având caracter ortogonal pentru tipologiile de evoluție, însă fiind importante pentru clasificatorul algoritmic, identificăm în plus și trei tipologii de aplicații pentru a ilustra acoperirea aplicațiilor - dacă datele de observație capturează întreaga durată de viață a aplicației (Acoperire de Tipul 1), un interval ascendent (Tipul 2), sau un interval descendent (Tipul 3).

Identificăm următoarele cinci tipologii majore de evoluție (vezi Figura 3.2): staționară, cu vârfuri dar fără variații bruște (de obicei cu un singur vârf), cu variații bruște ascendente, cu variații bruște descendente, și cu vârfuri multiple (cu spike-uri). Detaliem aceste clase de aplicații în teză.

Proiecții la Nivel de Pachete

Proiecțiile la Nivel de Pachete permit modelarea încărcării de pe rețea cauzată de aplicațiile sociale online, sub forma dimensiunii pachetelor și a timpilor dintre sosiri.

Mesajele transmise de aceste aplicații pot varia ca dimensiune cu ordine de mărime, de la mesaje de control fără conținut sau doar cu câțiva octeți de conținut, până la fișiere multimedia de câțiva MB. Mesajele mici pot cauza o încărcare semnificativă, întrucât fiecare pachet conține antete de la diferite nivele din stiva de comunicație. Mesajele mari de asemenea trebuie împărțite în cadre de date, la fiecare din acestea adăugându-se antete. Timpul dintre sosiri complementează dimensiunea pachetelor, pentru că și el are o influență semnificativă asupra consumului de energie. Odată ce a fost transmis un pachet, modulul wireless va rămâne activ o perioadă de timp consumând energie fără să transfere date. Astfel, mesaje mici și dese pot duce la un consum de energie chiar mai mare decât mesaje mari și puține.

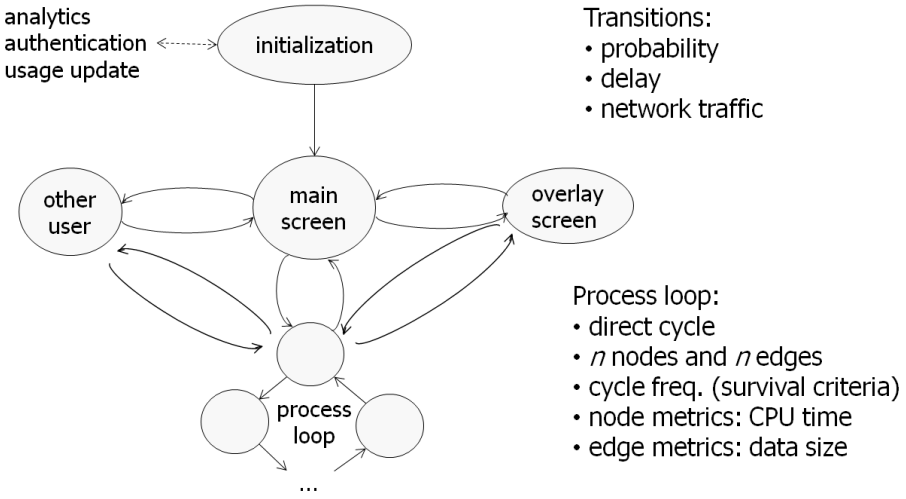


Figura 3.3: Diagrama Tranziției între Stări

Diagrama Tranziției între Stări

Aplicațiile sociale online, văzute ca mașini de stări, pot fi reprezentate cu ajutorul unui graf (vezi Figura 3.3). Mai întâi trec printr-o stare de inițializare, în care realizează operațiuni de autentificare și descărcare de conținut. Apoi trec într-o stare principală, când utilizatorul vede o hartă sau o altă formă de ecran principal. Din această stare, fie la intervale fixe de timp, fie la comanda utilizatorului, aplicația iterează o buclă de procesare. În acest timp, utilizatorul poate dechide un ecran secundar, sau poate realiza o interacțiune socială cu alți utilizatori, timp în care aplicația, de regulă, continuă sa itereze aceeași buclă de procesare.

Fiecare nod din Diagrama Tranziției între Stări reprezintă o etapă de procesare, descrisă de timpul și de resursele folosite în cadrul său. Tranzițiile reprezintă dependențele de date între etape succesive sub forma dimensiunii datelor transferate dintr-o etapă în alta.

3.3 Caracterizare și Modelare

În această secțiune prezentăm rezultatele empirice aferente modelului nostru pentru volumul de lucru. Caracterizăm și modelăm fiecare element, folosind seturile de date introduse în secțiunea 3.1.

Distribuția de Popularitate

Analizăm valorile maxime observate ale DAU și MAU ca parametri ai popularității aplicațiilor. Pentru a explica tranzițiile sezoniere și a putea urmări în continuare rezultatele, analizăm trimestrial informațiile pe care le deținem referitoare la acești parametri. Caracterizăm și modelăm cele unsprezece seturi de date trimestriale rezultate din datele colectate.

Pentru fiecare trimestru din seturile noastre de date selectăm numărul maxim DAU pentru fiecare dintre aplicații, apoi sortăm valorile maxime în ordine descrescătoare pentru a

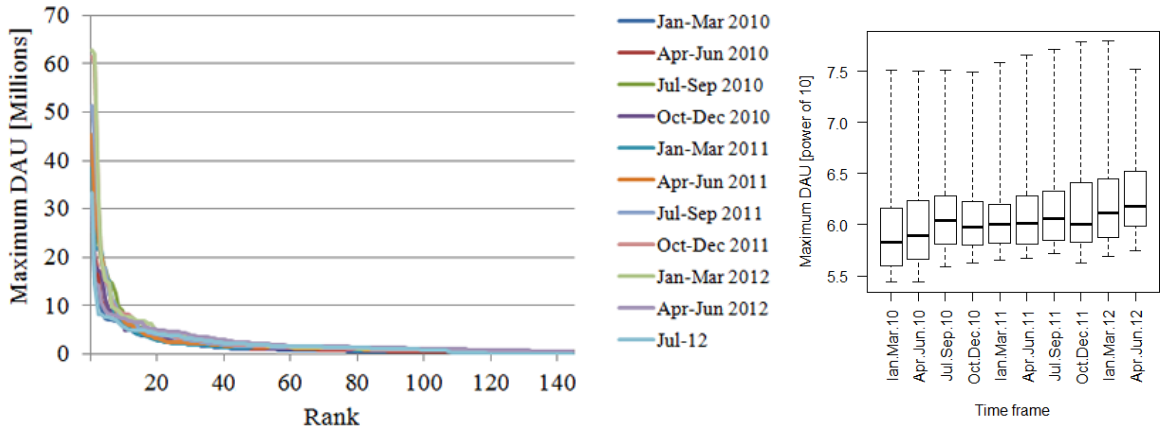


Figura 3.4: Distribuția de Popularitate la nivel de trimestre.

obține o ierarhie. În Figura 3.4 prezentăm distribuțiile corespunzătoare (în partea stângă) și o diagramă Box-and-Whiskers care oferă o vizualizare mai exactă a principalelor caracteristici statistice ale distribuțiilor (în partea dreaptă).

Fiecare curbă trimestrială prezintă același fenomen: aplicațiile clasate pe primele locuri (5% din totalul de aplicații) atrag mult mai mulți utilizatori decât toate celelalte aplicații însumate la un loc. Mai exact, primele 20 de aplicații au, individual, peste 5.000.000 de utilizatori activi zilnici.

Efectuăm potrivirea pentru toate cele unsprezece trimestre cu distribuții de referință, folosind testul Kolmogorov-Smirnov (testul K-S), ce cuantifică distanța între funcția de distribuție empirică a unui eșantion și funcția de distribuție cumulativă de referință. Alegem cinci distribuții de referință, Exponențială, Weibull, Pareto, Log-Normală și Gama, care acoperă majoritatea cazurilor întâlnite în literatura de specialitate. Pentru fiecare distribuție de referință, testul K-S furnizează parametrii care asigură cea mai bună potrivire, valorile p care indică nivelul de importanță al rezultatului, și valorile D care cuantifică distanța dintre distribuția noastră empirică și cea mai bună potrivire. Parametrii obținuți în cazul fiecărei distribuții, precum și valorile p și D pot fi regăsite în teză.

Tiparul de Evoluție

Analizăm evoluția valorilor observate în cazul utilizatorilor sub forma standard DAU și MAU. Similar abordării din subsecțiunea anterioară, prezentăm mai întâi caracterizarea, și apoi rezultatele modelării.

Selectăm toate aplicațiile care ating un vârf de cel puțin 100 de utilizatori DAU. În ciuda faptului că acest eșantion limitează numărul aplicațiilor la 5.723, considerăm, pe baza rezultatelor obținute în cadrul caracterizării popularității aplicațiilor, că toate celelalte aplicații nu înregistrează un impact semnificativ asupra populației generale de utilizatori. Clasificarea manuală a mii de aplicații este, desigur, irealizabilă. Acest aspect ne-a determinat să dezvoltăm un clasificator algoritmic.

Pentru a cuantifica prezența variațiilor bruște, clasificatorul furnizează valori mari din prima derivată a seriei temporale. Pentru a analiza durata de viață, clasificatorul compară valoarea inițială, valoarea finală și un segment de valori de la mijlocul intervalului de timp.

Tabelul 3.2 prezintă rezultatele obținute, exprimate ca număr al aplicațiilor prezente în fiecare din cele 5 clase.

Tiparul cu un singur vârf se manifestă în mai puțin de 25% din toate cazurile, fapt ce face ca majoritatea aplicațiilor să manifeste tipare neobișnuite, acestea fiind, prin urmare, cazuri de dificile pentru asigurarea resurselor.

Tabela 3.2: Rezultatele clasificării ce arată câte aplicații se încadrează în fiecare tipologie (stânga) și clasă (dreapta).

	Full	Ascending	Descending
Without bursts	460	32	871
Rising bursts	73	127	198
Descending bursts	66	51	1075
Spikes	596	240	1004
% total	25%	9.50%	66.70%
Constant: 319 (5.57%)	Other behaviours : 611 (19.7%)		

	Count	Percent
Cls-1	319	5.57%
Cls-2	1363	23.82%
Cls-3	398	6.95%
Cls-4	1192	20.83%
Cls-5	2451	42.83%

Suntem interesați și de determinarea oricăror componente sezoniere si ciclice în evoluția numărului de utilizatori. Inspirați de mai multe lucrări pe teme de statistică, precum [32], parcurgem studiul autocorelării utilizatorilor DAU ca funcție în timp prin trei etape:

1. îndepărtarea componentei-tendință prin diferențierea funcției
2. calcularea unei corelograme (o descriere a funcției de autocorelare) având o întârziere maximă egală cu lungimea vectorului de observații
3. valori de test exterioare benzii de eroare, $\pm\frac{2}{\sqrt{N}}$, cu un număr N de observații, sau utilizarea testului Ljung-Box pentru a determina orice valori aberante/atipice din corelogramă

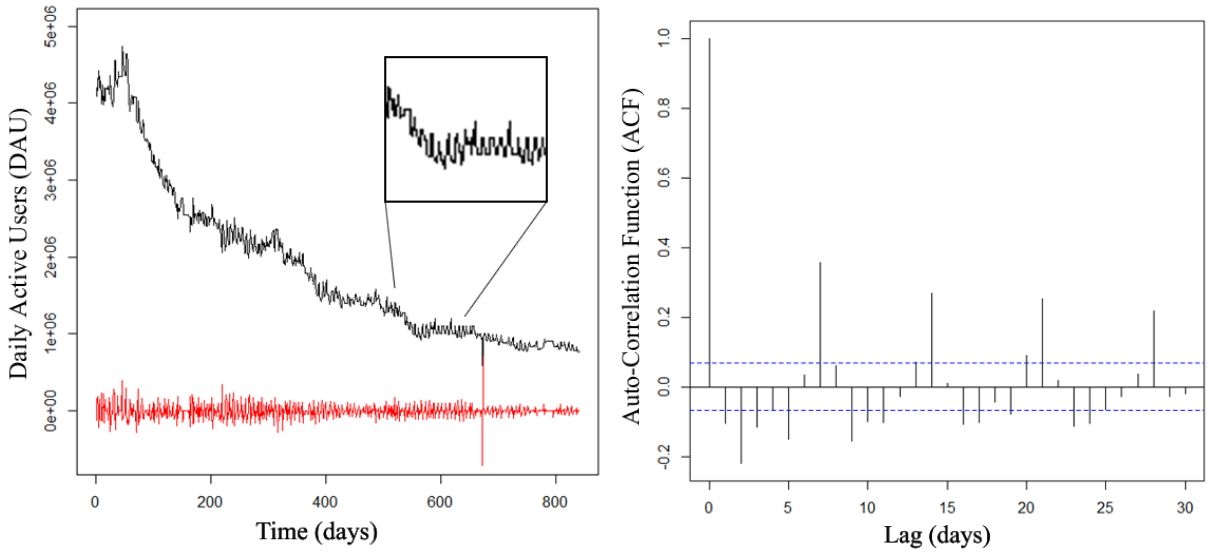


Figura 3.5: Corelograma (dreapta) arată funcționarea algoritmului de detectare a sezonalității pentru o aplicație (stânga)

Figura 3.5 indică modul în care algoritmul de detectare a periodicității funcționează în cazul unei aplicații tipice. Se poate reține modul în care funcția de autocorelare

depășește pragul cu valori multiple de 7, indicând un tipar sezonier săptămânal. Majoritatea aplicațiilor pe care le-am găsit, în special jocurile, prezintă un astfel de tipar.

Proiecții la Nivel de Pachete

Figura 3.6 arată distribuția dimensiunilor de pachete (stânga) și a timpilor dintre sosiri (dreapta) pentru câteva dintre aplicațiile din seturile noastre de date. Este important de remarcat ca FarmVille și MagicLands sunt aplicații bazate pe tehnologii web, dezvoltate de companii diferite, iar OpenTTD este o simulare strâns cuplată. Cu toate acestea, există unele similarități.

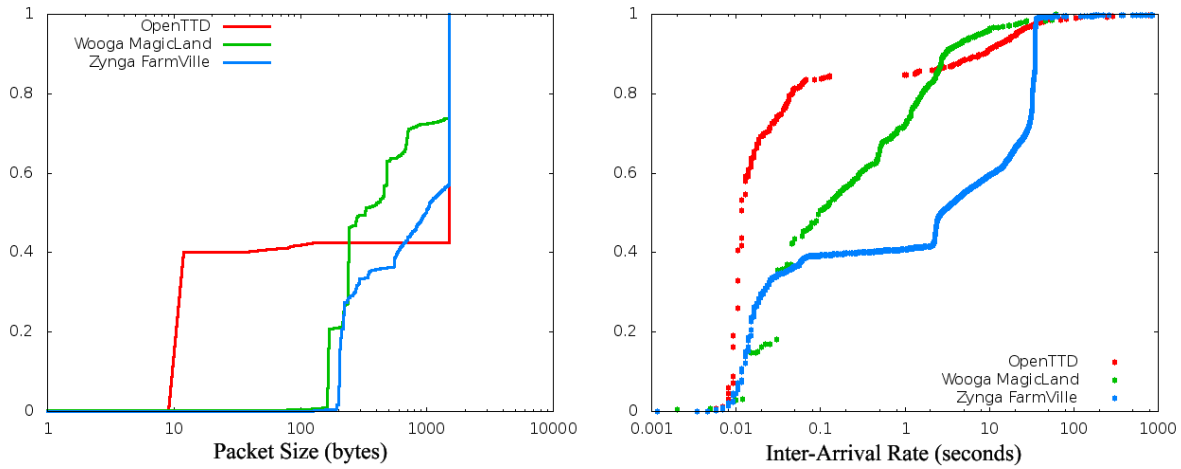


Figura 3.6: Dimensiunea pachetelor (stânga) și timpii dintre sosiri (dreapta) pentru câteva din aplicațiile din seturile noastre de date.

Atât dimensiunea pachetelor, cât și timpii dintre sosiri sunt reprezentate pe scală logaritmică, ceea ce arată că ambele diferă în valoare cu ordine de mărime. Timpul dintre sosiri are o prezență accentuată la aproximativ 2s și o dimensiune a pachetelor de 200 bytes, ceea ce indică o posibilă consecință a mesajelor de actualizare a stării. Aceeași creștere se manifestă în cazul OpenTTD la dimensiuni mai mici, de aproximativ 10 bytes. Dimensiunea fișierelor multimedia din aplicațiile sociale online este foarte mică, între 1KB și 1MB, cu o medie la mai puțin de 100KB. Obiecte de tip Flash reprezintă cantitativ, în medie, două treimi din conținutul multimedia transferat.

Pentru a modela dimensiunea pachetelor și timpii dintre sosiri, efectuăm potrivirea distribuțiilor empirice cu distribuții de referință, folosind testul Kolmogorov-Smirnov (testul K-S), ce cuantifică o distanță între funcția de distribuție empirică a unui eșantion și funcția de distribuție cumulativă a unei distribuții de referință. Alegem cinci distribuții de referință, Exponențială, Weibull, Pareto, Log-Normală și Gama, care acoperă majoritatea cazurilor întâlnite în literatura de specialitate. Pentru fiecare distribuție de referință, testul K-S furnizează parametrii care asigură cea mai bună potrivire, valorile p care indică nivelul de importanță al rezultatului, și valorile D care cuantifică distanța dintre distribuția noastră empirică și cea mai bună potrivire.

Parametrii obținuți în cazul fiecărei distribuții sunt rezumați în Tabelul 3.3, iar valorile p și D pot fi regăsite în teză.

Tabela 3.3: Parametrii pentru distribuțiile de referință (E–exponențial, W–Weibull, Ln–log-normal, și G–gamma) care oferă cea mai bună potrivire pentru Proiecțiile la Nivel de Pachete.

Model element	Best fit parameter value	Min	Median	Mean	Max
Packet inter-arrival time [s]	Ln($\mu_l = 0.28$, $\sigma = 1.95$)	0.01	1.84	-	-
Packet size, split [B]	W($k_w = 520.71$, $\lambda = 1.46$)	0	242	466	MTU
Packet size, un-split [B]	Ln($\mu_l = 7.37$, $\sigma = 1.95$)	0	988	12.5K	1.04M
Game object size, Flash only [B]	W($k_w = 80,459.72$, $\lambda = 0.81$)	1,004	44.9K	93.5K	1.04M
Game object traffic, Flash:All [%]	-	4.3	78	64	95

Diagrama Tranziției între Stări

Analizăm datele colectate pentru aplicația OpenTTD, folosind un laptop Sony Vaio 4-core @2.3GHz cu Ubuntu 10.04 ca server și o tabletă Asus TF101 2-core @1GHz cu Android 4.0.3 drept client. Pentru fiecare din cele 5 etape din bucla de procesare raportăm rezultate statistice in Tabela 3.4.

Tabela 3.4: Statistici cu privire la timpul de rulare mediu ($\overline{T}$), timpul de rulare maxim (T_{max}) și cantitatea de date rezultate (Q) pentru fiecare dintre cele cinci etape din bucla de procesare.

	tablet			laptop		
	$\overline{T}$ [ms]	T_{max} [ms]	Q [bytes]	$\overline{T}$ [ms]	T_{max} [ms]	Q [bytes]
human input	1.2	58	40	0.7	48	40
AI input	48.2	973	40	4.3	209	40
synchronization	188.5	2446	120	101.9	4576	120
simulation	9.3	211	1068576	2.6	283	1068576
rendering	5.7	726	4096000	13.7	135	5206720

Timpii citiți de pe tabletă sunt mai mari decât cei de pe laptop pentru aproape toate etapele. Randarea este singura etapă care ia mai puțin timp, probabil datorită rezoluției mai mici și a procesorului grafic dedicat de pe tabletă. Este interesant de observat că timpul de procesare pentru rularea jucătorilor de tip AI diferă cu un ordin de mărime. Simularea, deși mai consumatoare pe tabletă, nu prezintă o diferență atât de mare, probabil pentru că este deja optimizată prin împărțirea proceării peste mai multe etape.

Modelul evoluției volumului de lucru pentru aplicații vine cu utilizări variate. Pentru o companie care dezvoltă un OSG similar cu altele, modelul nostru oferă, în ansamblu, instrumentele de predicție a cerințelor de capacitate ale rețelei, pe baza unor ipoteze privind ierarhizarea relativă a aplicațiilor sociale online vs. oferta concurenței. Modelul nostru este, de asemenea, util companiilor care dezvoltă aplicații noi sau în situații când nu sunt disponibile date referitoare la competitori: în astfel de scenarii, capacitatea necesară a rețelei poate fi bazată pe ținte stabilite de dezvoltator cu privire la creșterea inițială și variațiile bruște ale aplicației sociale online. Ca un al treilea exemplu general, dacă este asociat unui generator de pachete cu nivel scăzut, modelul nostru asigură instrumentele ajustării arhitecturilor OSG existente și conceperii unor arhitecturi noi, prin generarea resurselor de intrare necesare simulărilor.

Capitolul 4.

Analiza Mecanismelor de Offloading

Terminale mobile sunt folosite zilnic în activități ce variază de la divertisment la îndeplinirea sarcinilor profesionale. Aplicațiile mobile acoperă un domeniu aplicativ vast, fiind dezvoltate în scopuri variate, cum ar fi jocuri, streaming multimedia, călătorii, comunicații etc. Multe din aceste tipuri de aplicații se bazează pe conectivitate și pe date stocate la distanță. De asemenea, multe din acestea folosesc din plin puterea computațională a dispozitivelor mobile. în cadrul acestui spațiu generos pentru aplicații identificăm mai multe tipuri de aplicații care ar avea de câștigat prin practicarea offloadingului: aplicații care au o componentă computațională importantă, precum șahul; aplicații care se bazează pe date furnizate de server, cum ar fi Shazam, programul care recunoaște piese muzicale comparând mostre cu o bază de date vastă stocată în cloud; aplicații cu procesare deosebită, bazată pe o buclă de procesare, precum aplicațiile destinate procesării imaginilor.

În acest capitol investigăm Adaptarea și Offloadingul Comunicației în cazul aplicațiilor pentru terminale mobile ce utilizează extensiile hardware personalizate, de tipul dispozitivelor cu senzori și rețelelor de automatizare a casei. Studiem, de asemenea, Adaptarea și Offloadingul Calculului pentru aplicații bazate pe buclă de procesare. În cele din urmă, propunem o analiză operațională pentru a reprezenta matematic toate mecanismele de offloading identificate în Secțiunea 2.3, pentru oricare aplicație de bazată pe buclă de procesare.

4.1 Adaptarea comunicației: Algoritm de Interogare Adaptiv

Am dezvoltat un sistem, denumit PollutionTrack (Depistarea Poluării), constând într-un telefon inteligent și un dispozitiv integrat destinat măsurării calității aerului [33]. Comunicația dintre telefonul mobil și dispozitivul cu senzori trebuie să fie eficientă din punct de vedere energetic pentru a avea un impact minim asupra ciclurilor de reîncărcare ale telefonului. Abordarea noastră pentru această provocare constă în reducerea consumului energetic al dispozitivului folosind un algoritm adaptiv de căutare ce minimizează consumul energetic necesar transferurilor de date. Algoritmul modifică timpul de actualizare a datelor de la sensor pe baza locației utilizatorului și a ultimelor valori citite. Această abordare este potrivită pentru aplicații conștiente de locație, reducând comunicația atunci când aceasta nu este necesară.

Intervalul de timp dintre interogări se calculează dinamic cu formula: $T_u = T_b(c_{ld} + c_{dd})$, unde: T_u: intervalul de interogare ajustat, T_b: intervalul de interogare de referință, c_{ld}: coeficientul dependent de locație, c_{dd}: coeficientul dependent de date.

Coeficientul dependent de locație se calculează pe baza locației curente și a celei precedente:

$$c_{ld}^{i+1} = \begin{cases} \alpha c_{ld}^{i} & \text{if } loc^{i+1} = loc^{i} \\ \frac{1}{\alpha} c_{ld}^{i} & \text{if } loc^{i+1} != loc^{i} \end{cases} \tag{4.1}$$

Pentru a calcula coeficientul dependent de date, păstrăm o coadă cu ultimele n valori citite, pe baza cărora determinăm cât de repede se schimbă valorile de poluare. Calculăm rata $r = \sum \frac{v-D_i}{v}$, unde v este valoarea curentă și D este un vector cu cele n valori:

$$c_{dd} = \gamma \frac{\lambda}{\sum \frac{v-D_i}{v}} \tag{4.2}$$

Studiem trei strategii de interogare: Permanent Connection (*Permanent*), Fixed Time Query Interval (*Fixed*) și Adaptive Query (*Adaptive*). În prima, telefonul stabilește o conexiune permanent cu dispozitivul cu senzori, ceea ce este solicitant pentru CPU și pentru modulul Bluetooth. Pentru Fixed Time Interval Query, procesorul și modulul de Bluetooth sunt puse în starea conectat, în care consumul de energie este cel mai mare doar când se fac interogări. Strategia Adaptive Query folosește algoritmul nostru pentru a selecta un interval.

Pentru a estima consumul de energie pentru cele trei strategii, folosim PowerTutor [34], precum și acest model simplu, în care introducem timpii măsurați de noi:

$$\mathrm{E}_{total} = \mathrm{P}_{conn} * \mathrm{T}_{conn} + \mathrm{P}_{trans} * \mathrm{T}_{trans} + \mathrm{P}_{on} * \mathrm{T}_{on}$$

Măsuratorile efectuate de Perrucci [35] arată că un modul Bluetooth va consuma fără conexiune $P_{on} = 15mW$, cu conexiune $P_{conn} = 65mW$ și în transmisie $P_{trans} = 432mW$.

La testare, telefonul solicită date de la dispozitivul cu senzori la 15 secunde, ce răspunde cu un singur mesaj de 21 de octeți. Figura 4.1 arată variația consumului de energie pentru cele trei strategii pe parcursul a 300 de secunde de test. Tabela 4.1 arată puterea totală estimată pentru procesor și modulul Bluetooth.

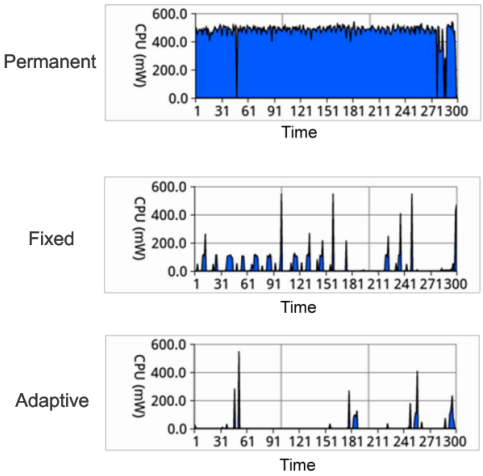


Figura 4.1: Consumul de energie pentru CPU

Tabela 4.1: Consumul estimat de energie pentru CPU și WiFi pentru cele trei strategii de interogare, în timpul testelor de 300 de secunde.

	CPU Energy (mJ)	**Data transfer (bytes)**	**Comm. Energy (mJ)**
Permanent	136,133	21 x 20 = 420	28,740
Fixed	5,465	21 x 20 = 420	14,180
Adaptive	1,310	21 x 4 = 84	6,228

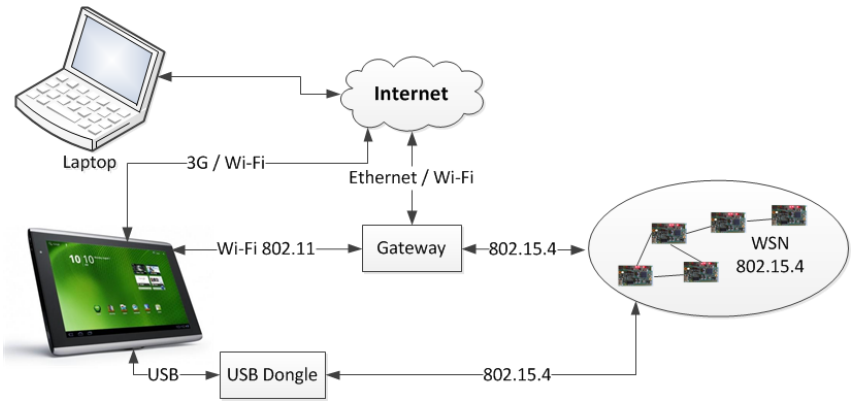
Se poate observa ca Algoritmul Adaptiv este aproape de 5 ori mai efficient decât Fixed Time Query Interval și de 100 de ori mai efficient decât Permanent Connection. Această eficiență este justificată pentru că în cele 300 de secunde, Fixed Time Query Interval a făcut 20 de interogări, în timp ce Algoritmul Adaptiv a făcut doar 4.

4.2 Offloading de Comunicație: extensii hardware

Dispozitivele mobile pot comunica cu o rețea de automatizare a casei prin intermediul unui portal de Internet, însă nu pot comunica direct cu dispozitive din rețea, care implementează, de obicei, protocoale de comunicație cu consum energetic redus, precum ZigBee, bazat pe standardul IEEE 802.15.4.

Propunem sistemul din Figura 4.2, ce constă din: o rețea ZigBee cu diverse dispozitive de automatizare a casei, un portal (gateway) aflat la marginea rețelei fără fir, ce oferă o interfață bi-direcțională între rețeaua de dispozitive și Internet, și unul sau mai multe dispozitive mobile cu rol de client ce pot accesa dispozitivele din rețeaua ZigBee atât prin Internet folosind portalul, cât și prin ZigBee, folosind un dongle USB echipat cu un modul de comunicație corespunzător.

Figura 4.2: Arhitectura sistemului de automatizare a casei



Ca parte a colaborării noastre cu Universitatea de Științe Aplicate din Dresda, am folosit o extensie hardware (dongle) de comunicație dezvoltată de ZigPos GmbH, ce se conectează la dispozitivul mobil prin USB, conform specificațiilor noastre. Pentru testare, proiectăm și implementăm o aplicație Android cu trei straturi de abstractizare, care să unifice canale diferite de comunicație: *Data Provisioning Layer*: oferă o abstractizare asupra metodelor de comunicație, fie locale prin dongle, fie la distanța prin portal; *Access Layer*: abstractizează toate metodele de comunicație locale într-un canal considerat relativ sigur, și pe toate cele la distanță într-un canal ce are nevoie de măsuri suplimentare de securitate;*Communication Channel Layer*: abstractizează folosirea a mai multe module de comunicație, precum cele prin serială, USB, audio jack, Bluetooth, WiFi, etc.

Testăm aplicația pe două dispozitive de client ce rulează Android 4.0, o tabletă Acer Iconia A500 și o tabletă Asus Transformer TF101. De asemenea, folosim un model de rețea de automatizare a casei compus dintr-un portal (gateway) PC și două dispozitive ZigBee, concepute de ZigPos, care simulează corpuri de iluminat convenționale dintr-o locuință. Testăm mai multe cazuri de utilizare:

comandă de comutare: în acest caz utilizatorul emite o comandă simplă de comutare a luminilor;imagine encodată: sistemul permite utilizatorilor să încarce o fotografie pentru dispozitivul din rețea;istoricul valorilor: un istoric detaliat într-un format compact JSON;

Estimăm consumul energetic în aceste trei cazuri de utilizare folosind WiFi. Cu toate că nu am putut o fișă de specificații tehnice pentru modulul WiFi al nici uneia dintre cele

două tablete, estimăm datele prezentate în continuare la o tensiune de 3V și un curent de 270,40 mA. Folosim interfața API Android internă pentru a măsura valorile dimensiunilor reale atinse de transferuri, viteza de conexiune și valoarea curentă pentru modulul WiFi în starea activă. Valorile raportate în Tabelul 4.2 reprezintă mediile unui număr de cinci măsurări.

Tabela 4.2: Estimarea consumului de energie pentru cele trei cazuri de utilizare.

	size	link speed	time	energy	energy/byte
	[bytes]	[Mbit/s]	[ms]	[uJ]	[uJ]
toggle command	48	26	0.003	2.47	0.051
encoded image	57,296	39	0.177	143.62	0.003
history of values	2,468,676	52	5.713	4,634.06	0.002

Observând raportul energie/byte din Tabelul 4.2, concluzionăm că modulul WiFi consumă multă energie suplimentară pentru transferuri mici, precum cele din Cazul de Utilizare 1, caz a cărui frecvență ne așteptăm să fie cea mai ridicată într-o implementare reală. Pentru transferuri de acest gen, folosirea protocolului ZigBee se poate dovedi eficientă din punct de vedere energetic.

4.3 Adaptarea Calculului: aplicații de procesare video

Un simplu telefon inteligent cu cameră video poate fi utilizat pentru a păși într-o lume virtuală sau augmentată. Spre exemplu, într-un muzeu, utilizatorii care privesc hărți prin intermediul telefoanelor mobile pot vedea animații suprapuse peste acestea care să redea evenimentul istoric. Am dezvoltat o aplicație de realitate augmentată pentru a studia etapele prin care informația trebuie să treacă pentru a fi afișată utilizatorului. Am ales coduri QR ca marker pentru detectarea reperelor, întrucât acestea beneficiază deja de o serie de implementări pentru diverse platforme mobile.

Aplicația client rulează într-un ciclu de procesare (vezi Figura 4.3): achiziție imagini, citire coduri QR, detectare a suprafeței, achiziție conținut și transformare conținut. Acesta corespunde Diagramei de Tranziție între Stări din modelul nostru pentru volumul de lucru prezentat în Secțiunea 3.2.

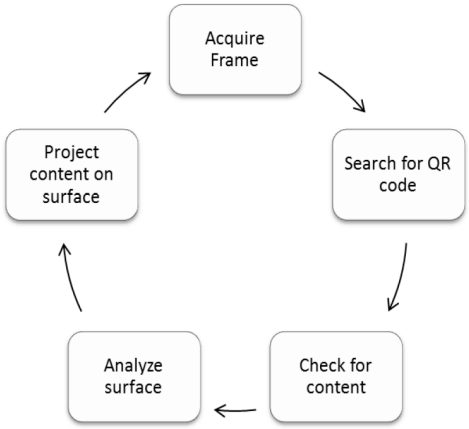


Figura 4.3: Bucla de procesare a aplicației realitate augmentată

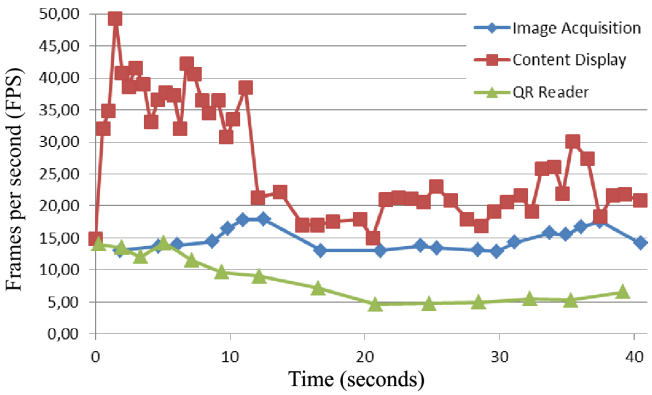


Figura 4.4: Rezultate empirice cu privire la numărul de cadre procesate de fiecare modul.

Aplicația este un exemplu de adaptare a calculului, pentru că fiecare modul va prelua de la modulul precedent doar atâtea cadre câte poate procesa, renunțând la celelalte.

Evaluăm performanța, exprimată în cadre pe secundă (FPS), în diverse condiții. Figura 4.4 prezintă rezultatele unui test care urmărește să compare performanța în diverse etape, cu și fără existența unui cod QR. Putem observa numărul cadrelor procesate în fiecare secundă de către cele mai solicitante module (achiziție imagini, cititorul de coduri de bare QR și afișarea conținutului), precum și, la secunda 12, impactul produs de poziționarea unui cod de bare în fața camerei pe care aplicația trebuie să îl proceseze.

În Tabelul 4.3 rezumăm rezultatele obținute pe patru dispozitive cu caracteristici de hardware diferite. Numărul de FPS corespunde modulului de proiectare conținut, modulul final din bucla de procesare, reprezentând viteza afișată utilizatorului pentru imaginile noi transformate.

Tabela 4.3: Numărul de cadre procesate în funcție de dispozitiv (stânga) și conținut (dreapta)

Device	Cores	Resolution	FPS			Content Type	FPS
			Low	Medium	High		
SE Xperia	1 core @1GHz	320x480	23	18	13	Image (png)	23-27
Samsung GS2	2 cores @1.2GHz	540x960	34	29	24	Animation (gif)	20-27
HTC One S	2 cores @1.7GHz	480x800	40	35	30	Video (mp4)	15-20
Acer A510	4 cores @1.3GHz	1280x800	17	15	13		

Tot în Tabelul 4.3 prezentăm performanțele obținute pentru diverse tipuri de conținut proiectat. Gestionarea animației este foarte rapidă și se limitează la manipularea unei serii de imagini simple, putând să atingă performanțe similare cu randarea imaginilor.

Totuși, randarea conținutului video necesită o putere mare de procesare deoarece folosim librăria ffmpeg pentru a extrage cadre individuale cu marcaje temporale specifice, fapt ce adaugă un exces de consum semnificativ. Când o imagine trebuie să fie proiectată (modulul de imagine este pregătit să extragă o altă fotografie), căutarea în fișierul video se va face de la ultima poziție la noul marcaj temporal.

4.4 Offloadingul Calculului: simulare și randare video

În această secțiune investigăm offloadingul aplicat pe OpenTTD, un joc popular open-source. În modul multiplayer, OpenTTD suportă în prezent până la 255 de utilizatori simultani pe aceeași hartă, unul dintre aceștia având datoria de a găzdui serverul. O altă metodă populară de utilizare a jocului este modul singleplayer, împotriva unui număr de jucători controlați de algoritmi de inteligență artificială (jucători IA). Jucătorii IA sunt dezvoltați de comunitate și disponibili gratuit pentru toți utilizatorii.

OpenTTD este o aplicație ce funcționează iterând o buclă de procesare (vezi Figura 4.5): captura inputului de la utilizator, rularea jucătorilor IA, sincronizarea mesajelor pe server (doar în jocurile multiplayer), simulare și randare. Astfel, este conceptual similară cu aplicația de realitate augmentată prezentată în Secțiunea 4.3 și poate fi modelată cu Diagrama Tranziției între Stări prezentată în cadrul modelului nostru pentru volume de lucru în Secțiunea 3.2.

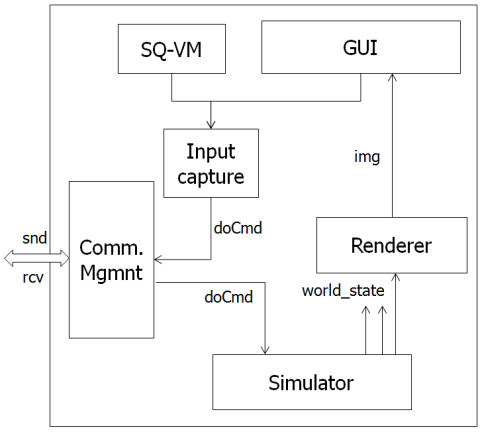


Figura 4.5: Bucla de procesare în OpenTTD

Pentru implementare, începem modificarea codului sursă al versiunii OpenTTD 1.3.0, lansată în aprilie 2013, una dintre cele mai recente versiuni de OpenTTD. Implementăm offloadingul printr-un joc multiplayer în care clientul rulează pe dispozitivul mobil iar serverul rulează jucătorii AI/boții. În acest sens efectuăm câteva modificări în versiunea pentru comunitate.

Modificăm jocul astfel încât jucătorii IA să poată rula în modul multiplayer, această caracteristică fiind de obicei dezactivată. Pentru a efectua experimentele, este necesar să le repetăm de câteva ori și să ne asigurăm că toate manifestă un comportament identic. Astfel, înlocuim jucătorul uman cu unul IA ce rulează pe dispozitiv, iar implementarea noastră îi urmărește mișcările pe ecran. Începerea unui joc cu același jucător IA și același scenariu va recrea la fiecare rulare aceleași tipare de utilizare și aceleași operațiuni.

Pentru a experimenta cu dispozitive mobile reale, folosim portarea SDL pentru Android scrisă de Pelya. SDL este, în esență, librăria grafică folosită de OpenTTD. Prin portarea SDL pentru Android, dezvoltatorul a deschis drumul portării oricăror jocuri care utilizează SDL. Am reușit astfel să folosim codul C++ modificat de noi, care este compilat cu Android NDK și apoi conectat la restul proiectului SDL Android.

Folosim în experimente, în rol de client, un Asus Eee Pad Transformer TF101 care dispune de un procesor central nVidia Tegra 2 1GHz dual-core și rulează Android 4.0.3. De asemenea, folosim ca server un laptop Sony Vaio, cu procesor Intel Core i3 2.27 GHz quad-core, care rulează Linux Ubuntu 10.04. Cele două dispozitive sunt conectate prin WiFi și se află în aceeași rețea locală.

În scenariul pe care îl investigăm un jucător uman desfășoară un meci împotriva unui număr de jucători AI. Am început cu 16 jucători AI, însă rularea jocului în rețeaua locală cu atât de mulți jucători AI se dovedește imposibil de randat pe dispozitivul nostru. Așadar, pentru a colecta date pentru o rulare de probă, am scăzut în mod repetat numărul de jucători AI/boți până când jocul a devenit utilizabil pe dispozitivul nostru. Ne-am oprit la un număr de 4 jucători AI, pe care i-am selectat dintre cei mai populari din comunitate, mai exact OtviAI, AIAI, ChooChoo și Chopper.

Figura 4.6 indică încărcarea procesorului și timpul de joc înregistrate în timpul unei sesiuni de joc de 10 minute, folosind versiunea noastră de OpenTTD.

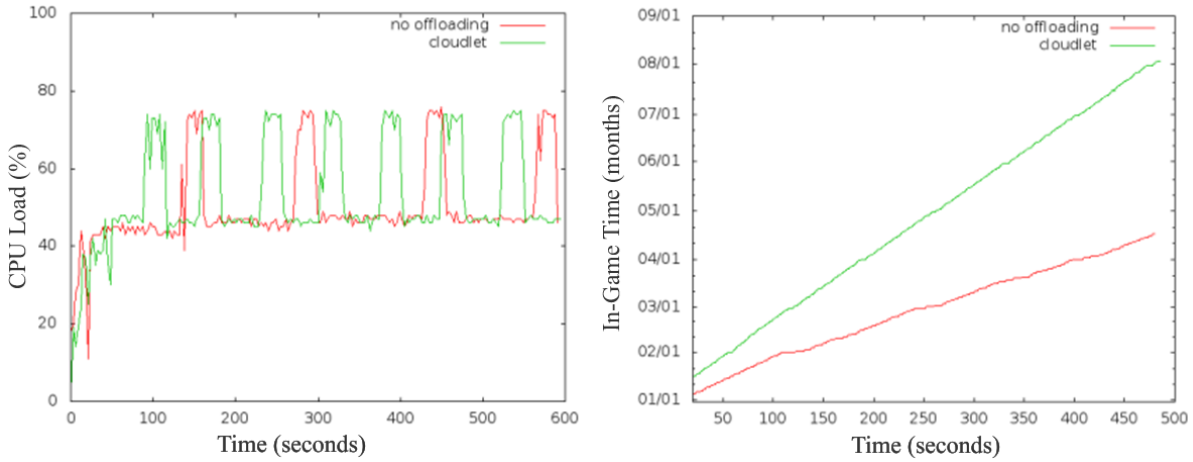


Figura 4.6: Comparison of local running of AIs (red) with offloaded running of AIs (green)

Graficul din partea stângă indică o încărcare a procesorului de 40-50% în majoritatea situațiilor, corespunzătoare utilizării intense a unuia din cele două nuclee. Spike-urile sunt declanșate de salvările automate care, conform setării noastre, se petrec o dată pe lună contorizată în joc. Salvările automate sunt realizate pe un fir separat, fapt ce explică de ce încărcarea procesorului depășește pragul de 50%. Graficul din partea dreaptă indică modul în care timpul din joc avansează în raport cu timpul real. Axa verticală descrie timpul din joc măsurat în luni, în timp ce axa orizontală descrie timpul real scurs în secunde.

Ambele grafice indică faptul că, în lipsa offloadingului, jocul devine mai lent pentru a compensa lipsa puterii de procesare a dispozitivului client. După cum o ilustrează valorile din graficul din partea dreaptă, precum și numărul de spike-uri afișat în graficul din partea stângă, într-o sesiune de joc de 10 minute versiunea transferată acoperă aproximativ 8 luni contorizate în joc, în timp ce versiunea locală acoperă doar 4 luni contorizate în joc.

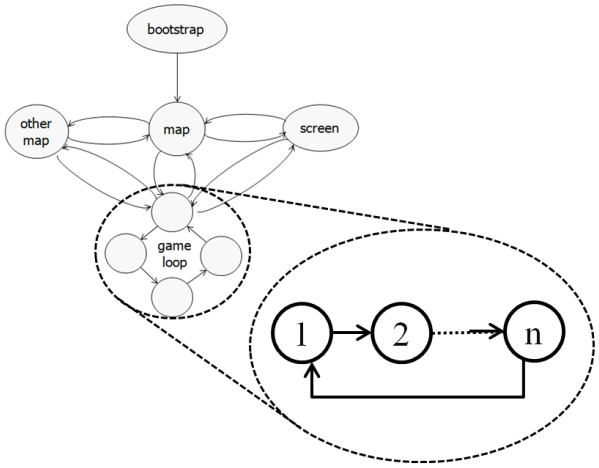
4.5 Analiza Operațională pentru Mecanisme de Offloading

Continuăm investigația noastră în domeniul aplicațiilor cu buclă de procesare și, în această secțiune, propunem o analiză operațională structurată pe mecanismele de offloading identificate în cadrul Spațiului de Explorare din Secțiunea 2.3.

După cum am menționat în Capitolul 4, ne concentrăm asupra aplicațiilor în care o mare parte a funcționalității este asigurată prin iterarea unei bucle de execuție. Toate Aplicațiile Sociale Online, pentru care am realizat un Model al Volumului de Lucru în Capitolul 3, dispun de o astfel de buclă, după cum se poate observa în Modelul Micro din Secțiunea 3.3. În plus, în cazul Aplicațiilor Sociale Online, indiferent de tehnologia utilizată (Simulări Strâns Cuplate, Web 2.0 sau Streaming) o porțiune din funcționalitate se produce deja la distanță, acesta reprezentând unul din criteriile în funcție de care le-am ales pentru a ne desfășura studiile.

Modelăm bucla principală a aplicației printr-un graf ce conține un ciclu, cu etape numerotate de la 1 la n, după cum ilustrează Figura 4.7. Un exemplu de buclă de acest gen este

oferit în Figura 4.5, care descrie bucla principală de procesare a OpenTTD, care constă în etape precum captarea datelor de intrare, sincronizarea în server, simulare și randare. Pe baza acestei bucle, și derivate din parametrii și modelele de evaluare a avantajelor descrise în taxonomia noastră (Secțiunea 2.2), expunem trei criterii principale de evaluare a avantajelor offloadingului: performanța exprimată ca timp (Ecuația 4.3), energie (Ecuația 4.4) și cost financiar (Ecuația 4.5), unde: T - timpul necesar efectuării unui calculul sau unui transfer de date, E - energia consumată de calcul sau de un transfer de date, C - costul monetar ce urmează a fi plătit pentru un calcul sau un transfer de date, p/r - procesare locală/la distanță, Q - cantitatea de date ce urmează a fi transferată, $B(s,d)$ - lățimea de bandă în contextul transferului de date din sursa s către destinația d.



$$T = \sum_{i=1}^{n} T^i_{p|r} + \sum_{i=1}^{n} \frac{Q^i}{B(s,d)} \tag{4.3}$$

$$E = \sum_{i=1}^{n} E^i_{p|r} + \sum_{i=1}^{n} E_t(Q^i, B(s,d)) \tag{4.4}$$

$$C = \sum_{i=1}^{n} C^i_{p|r} + \sum_{i=1}^{n} C_t(Q^i, B(s,d)) \tag{4.5}$$

Figura 4.7: The main loop as a cycle graph.

În această secțiune realizăm unele simplificări pentru a păstra formulele concise, sacrificând într-o mică măsură precizia. Mai întâi considerăm viteza de descărcare și viteza de încărcare la valori apropiate:

$$B(local, ladistanță) = B(ladistanță, local) = B \tag{4.6}$$

De asemenea, luăm în calcul faptul că transferul datelor între etapele locale, precum și transferul datelor între etapele la distanță, este mult mai rapid decât transferul datelor de la o etaă locală la una la distanță și viceversa, și îl catalogăm, din acest motiv, infinit:

$$B(local, local) \to \infty, B(ladistanță, ladistanță) \to \infty \tag{4.7}$$

În cele din urmă, în demersul estimării consumului energetic, energia consumată de resursele izolate poate fi exclusă din perspectiva dispozitivului client. În mod similar, nu luăm în considerare costurile monetare necesare efectuării locale a calculelor în dispozitiv:

$$E_r \approx 0, C_p \approx 0 \tag{4.8}$$

Pe parcursul acestei secțiuni, vom detalia formula pentru performanță pentru mai multe mecanisme de offloading. Formulele pentru energie și cost se pot detalia analog.

Offloading pentru diferite componente

Modelăm aplicații care au deja o formă de procesare la distanță. Simbolizăm acest lucru figurând una din etape ca fiind procesată în exteriorul dispozitivului (vezi Figura 4.8 a). Ca exemplu, OpenTTD, jocul-simulare prezentat în secțiunea 4.4, are etapa de sincronizare a mesajelor în multiplayer pe server.

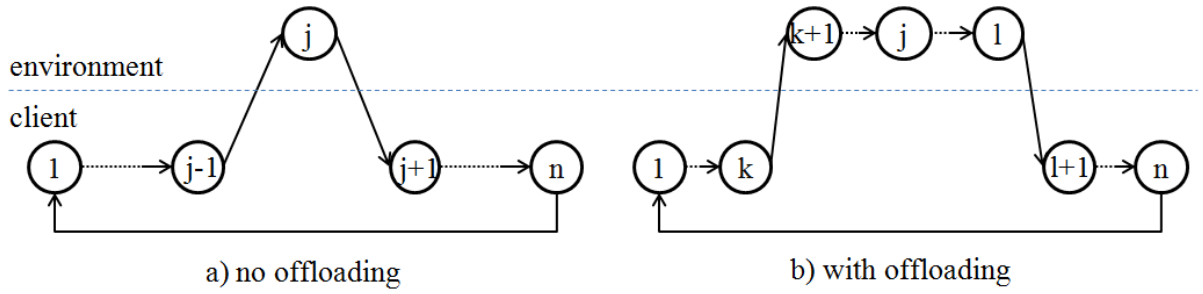


Figura 4.8: The graph model for varying offloaded components.

Prin operațiunea de offloading ne vom referi la mutarea procesării a mai multe etape în exteriorul dispozitivului. De exemplu, la OpenTTD, pe lângă sincronizare, ne putem propune să procesăm și etapa de simulare pe server. Vom nota prin k etapa unde transferăm procesarea la distanță și prin l etapa unde procesarea se întoarce pe dispozitivul mobil (vezi Figura 4.8). Exprimăm această idee în ecuația pentru timp care devine:

$$T(k,l) = \sum_{i=1}^{k} T_p^i + \sum_{i=k+1}^{l} T_r^i + \sum_{i=l+1}^{n} T_p^i + \frac{Q^k + Q^l}{B} \tag{4.9}$$

În general, procesul de offloading este benefic dacă procesarea la distanță față de cea locală aduce un beneficiu mai mare decât costul trimiterii datelor la distanță:

$$\frac{Q^k + Q^l}{B} < \sum_{i=k+1}^{l} (T_p^i - T_r^i) \tag{4.10}$$

Astfel, decizia de offloading devine problema de optimizare exprimată de:

$$T_{min} = \min_{k,l} \left\{ \sum_{i=1}^{k} T_p^i + \sum_{i=k+1}^{l} T_r^i + \sum_{i=l+1}^{n} T_p^i + \frac{Q^k + Q^l}{B} \right\}, 1 < k < j < l < n \tag{4.11}$$

Putem exprima logica pentru energie și cost financiar analog cu cea pentru durata buclei de procesare.

Offloading Intermitent

Ne referim prin offloading intermitent la acea situație în care, în executarea iterativă a unei bucle de procesare, vom apela la offloading pentru o anumită secțiune doar în anumite

iterații. Vom descrie această situație matematic folosind probabilități. Notăm cu $P(l)$ probabilitatea ca etapa $j+1$ să se desfășoare local (vezi Figura 4.9).

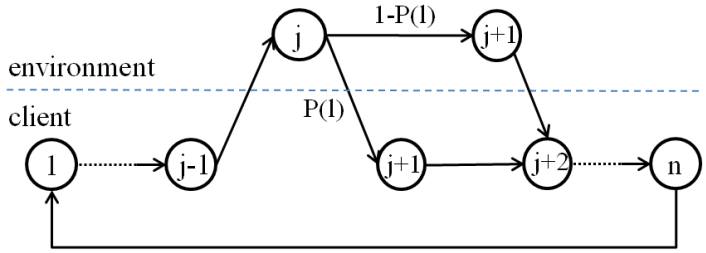


Figura 4.9: The graph model for partial offloading process.

Astfel, problema de optimizare devine găsirea acelui $P(l)$ care minimizează timpul total de parcurgere a iterației.

$$T_{min} = \min_{P(l)} \left\{ \sum_{i=1}^{j-1} T_p^i + T_r^j + P(l)T_p^{j+1} + (1-P(l))T_r^{j+1} + \right.$$
$$\left. + \sum_{i=j+2}^{n} T_p^i + \frac{Q^{j-1} + P(l)Q^j + (1-P(l))Q^{j+1}}{B} \right\} \tag{4.12}$$

Offloading cu Date Parțiale

Când apelăm la offloading cu date parțiale, cel puțin o etapă din bucla de procesare se va desfășura în același timp și local, și la distanță, pe subseturi diferite de date. Spre exemplu, în OpenTTD, jumătate de hartă poate fi procesată pe client, și cealaltă jumătate pe server. O modalitate mai potrivită de a împărți datele este folosirea noțiunii de regiune de interes, care, spre exemplu în jocuri, poate fi zona din hartă pe care o vede utilizatorul. Noi reprezentăm zona de date care se procesează local ca fiind fracția R din totalul de date (vezi Figura 4.10).

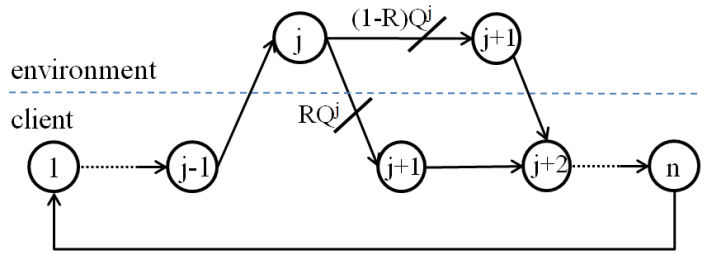


Figura 4.10: The graph model for partial offloading data.

Problema de optimizare devine găsirea acelui R pentru care timpul pe iterație devine

minim:

$$T_{min} = \min_{R} \left\{ \sum_{i=1}^{j-1} T_p^i + T_r^j + \frac{T_p^{j+1}}{R} + \frac{T_r^{j+1}}{1-R} + \sum_{i=j+2}^{n} T_p^i + \frac{1}{B}(Q^{j-1} + RQ^j + (1-R)Q^{j+1}) \right\} \tag{4.13}$$

Offloading Paralel

Offloadingul poate fi făcut în paralel, utilizând mai multe resurse la distanță în același timp. Spre exemplu, în OpenTTD, jucatorii controlați prin algoritmi de inteligență artificială iau deciziile individual, fără niciun fel de comunicație, luând în calcul doar starea curentă a hărții. Prin urmare pot fi procesați la distanță separat (vezi Figura 4.11).

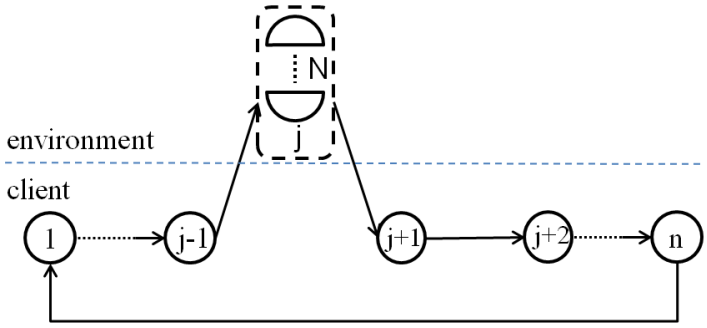


Figura 4.11: The graph model for offloading parallelism.

Un aspect cheie în decizia de a paraleliza procesul de offloading este măsura în care etapa de procesat are o secțiune serială, ce nu poate fi paralelizată. Exprimăm acest lucru cu ajutorul legii lui Amdahl:

$$T_{r||}^j = T_r^j(S + \frac{1-S}{N}) \tag{4.14}$$

Folosim această formă a legii lui Amdahl în ecuația 4.3, luând totodată în calcul faptul că paralelizarea implică trimiterea datelor la N resurse:

$$T_{min} = \min_{N} \left\{ \sum_{i=1}^{j-1} T_p^i + T_r^j(S + \frac{1-S}{N}) + \sum_{i=j+1}^{n} T_p^i + N\frac{Q^{j-1} + Q^j}{B} \right\} \tag{4.15}$$

Așa cum arată formula, offloadingul în paralel poate fi benefic pentru aplicații cu o zonă serială S cât mai mică, sau dacă cele N resurse nu vor avea nevoie de toate datele, scutind astfel multiplicarea dimensiunii comunicației.

Capitolul 5.

Evaluarea Performanței Mecanismelor de Offloading

Date fiind rezultate din toate celelalte capitole ale acestei teze, efectuăm în acest capitol evaluarea experimentală și analitică a mai multor mecanisme de offloading. Selectăm mecanismele noastre de offloading din Spațiul de Explorare definit în Capitolul 2. Pentru evaluarea empirică ne extindem experimentele în direcția simulatorului OpenTTD, prezentat în Capitolul 4, iar pentru evaluarea analitică utilizăm analiza operațională descrisă în Capitolul 4, precum și proiecțiile volumului de lucru prezentate în Capitolul 3.

Analizarea consumului de energie este o temă de cercetare interesantă, după cum o demonstrează diferite proiecte disponibile în literatura de specialitate. Caroll și Heiser [36] au conceput o serie de sisteme de referință pentru a asocia costurile energetice cu modulele dintr-un sistem mobil. Zhang et al. [34] descriu un modul software pentru a caracteriza consumul de energie pe diverse module hardware. Am folosit instrumente de acest gen pentru e estima consumul energetic al diverselor componente.

În activitatea noastră aplicăm tehnici de Analiză a Performanței Sistemelor Calcul care sunt bine definite în domeniu, oferind instrumente extrem de utile sub forma modelării analitice. Kleinrock [37] furnizează o introducere foarte detaliată în modelarea sistemelor stocastice de flux pe baza teoriei cozilor. Jain [38] oferă o trimitere amănunțită la principiile inginerești fundamentale și practice de evaluare a sistemelor computerizate iar King [39] dezvoltă aceste principiile pe teme de comunicație.

5.1 Evaluarea Empirică

În evaluarea noastră empirică continuăm activitatea prezentată în Secțiunea 4.4, unde detaliem un experiment de offloading aplicat pe OpenTTD, un joc open-source popular. Concepem și implementăm un *testbed* (platformă pentru realizarea de teste și experimente) bazat pe simulatorul OpenTTD, cu ajutorul căruia realizăm o explorare a spațiului de concepție pentru mecanismele de offloading, fundamentată pe Spațiul de Explorare propus în Secțiunea 2.3.

Configurarea Experimentelor

Pentru a efectua evaluarea empirică a diferitelor mecanisme de offloading, concepem și implementăm un *testbed* bazat pe OpenTTD, un joc open-source popular. OpenTTD reprezintă versiunea open-source pentru Transport Tycoon Deluxe, un joc-simulator de afaceri dezvoltat de Chris Sawyer în 1994. Fiind o aplicație open-source, OpenTTD are o comunitate de dezvoltatori care actualizează în permanență jocul și publică toate sursele într-o arhivă. În plus, alți dezvoltatori portează jocul pe diferite platforme. Spre exemplu, portarea pe Android realizată de Pelya are sute de mii de utilizatori activi.

Alegem OpenTTD pentru că este o aplicație reală, cu sute de mii de utilizatori, ce se încadrează în categoria aplicațiilor online sociale, implementată ca simulare strâns cuplată, pentru care avem un model al volumului de lucru (vezi Capitolul 3). Funcționalitatea sa de bază constă în iterarea unei bucle de procesare descrisă în Secțiunea 4.4, ceea ce înseamnă că o putem analiza prin metoda propusă în Secțiunea 4.5.

Implementăm testbedul operând mai multe schimbări la nivelul versiunii pentru comunitate. Pentru a efectua experimentele, este necesar să le repetăm de câteva ori și să ne asigurăm că toate manifestă un comportament identic. Astfel, înlocuim jucătorul uman cu unul AI/un bot ce rulează pe dispozitiv iar implementarea noastră îi urmărește mișcările pe ecran. Astfel, începerea unui joc cu același AI și același scenariu va recrea la fiecare rulare aceleași tipare de utilizare și aceleași operațiuni. De asemenea, adaptăm procedura de pornire pentru a putea iniția jocul cu ușurință atât pe dispozitivul mobil, cât și pe server prin intermediul scripturilor. Când se inițiază aplicațiile pentru platforma Android, nu este posibil să transmitem codului nativ argumente de linie de comandă, fiind astfel necesar să implementăm unele setări suplimentare de configurare în fișierul inițial de configurare.

În continuare, implementăm instrumentația în toate componentele arhitecturii pentru a colecta diverși parametri, după cum se ilustrează în Figura 5.1:

- pe dispozitiv, în OpenTTD: măsurăm parametrii specifici ai jocului precum cadrele pe secundă și timpul contorizat în joc, dar și statisticile specifice componentelor, cu referire la timpul de procesare și cantitatea de date
- pe dispozitiv, la nivel de aplicație: rulăm mai multe aplicații de profilare cum ar fi vprof și VTune, și instrumente de profilare precum top și iftop pentru Android, în vederea colectării unor parametri cum sunt încărcarea procesorului, încărcarea memoriei și încărcarea rețelei
- pe dispozitiv, în nucleu: pot fi plasate ancore și solicitări de sistem pentru redirecționarea la nivele mai înalte a informațiilor captate de la contoarele de hardware instalate pe dispozitivul fizic, informații precum stările C; am investigat stările C pentru a obține o mai bună înțelegere a încărcărilor procesorului [40], însă nu le detaliem în acest testbed
- la nivel de rețea, programe cunoscute drept detectoare de pachete, cum sunt Fiddler și Wireshark, captează pachete și facilitează realizarea statisticilor, cum ar fi cele pentru pachete trimise și primite, octeți trimiși și primiți, durata sesiunii, timpi de sosire și dimensiunea datelor de intrare și a celor de ieșire
- la nivelul serverului

Sistemul nu ia o decizie dinamică, ci este instruit ce teste să ruleze de către observator, întrucât suntem interesați de comparația a diferite mecanisme.

Calibrare: efecte ale parametrilor

Evaluăm influența pe care o au diverși parametrii asupra performanței, parametrii precum calitatea grafică și dimensiunea scenariului. Calitatea grafică are o mare influență

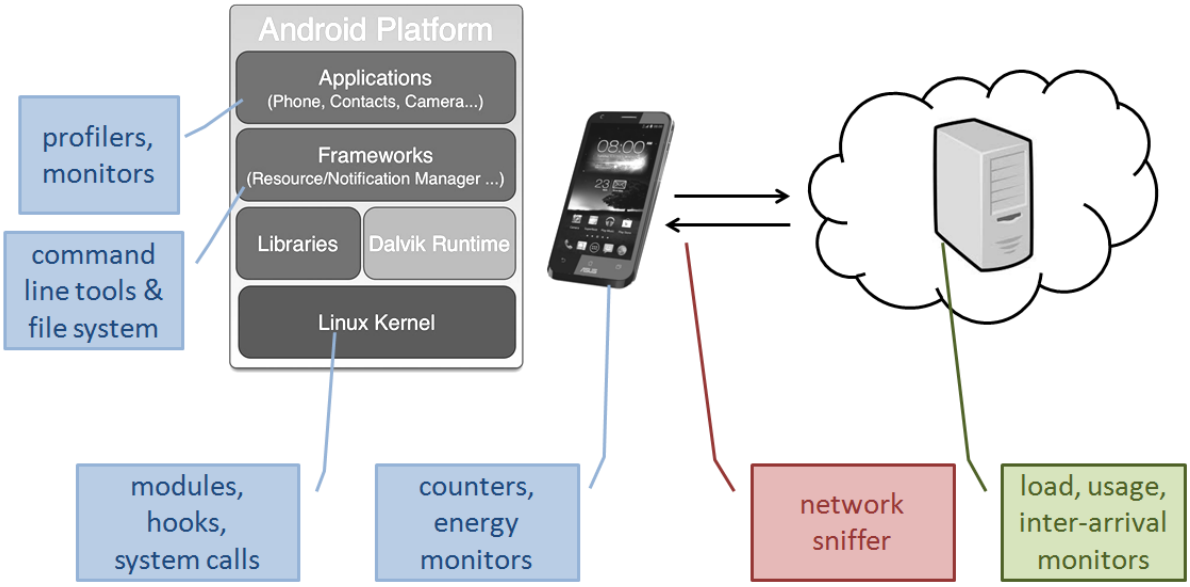


Figura 5.1: Instrumentarea în platforma noastră experimentală.

îndeosebi asupra etapei de randare. Definim calitatea grafică ca fiind scăzută sau ridicată în funcție de numărul de biți folosiți pentru a reprezenta culorile (8-bit sau 32-bit). Animațiile (pornite sau oprite) se referă la mișcarile diferitelor artefacte pe ecran, ce necesită procesare suplimentară. Detaliile (pornite sau oprite) se referă la complexitatea desenului de pe fiecare căsuță din hartă.

Dimensiunea scenariului are o mare influență asupra simulării, care funcționează iterând prin reprezentarea lumii din joc, precum și asupra timpului de decizie al jucătorilor cu inteligență artificială, pentru că și aceștia trebuie să treacă prin întreaga reprezentare a lumii din joc pentru a lua o decizie. Selectăm trei scenarii de diferite mărimi din comunitate: Europa este o hartă mare și densă (1024x1024 de căsuțe, 10.000 de resurse), Antarctica este o hartă mare și rară (1024x1024 de căsuțe, 100 de resurse) și 2 Mountains este o hartă mică și densă (128x128 de căsuțe, 100 de resurse). Folosim drept client o tabletă Asus TF101 2-core @1GHz cu Android 4.0.3 și drept server un laptop Sony Vaio 4-core @2.3GHz cu Ubuntu 10.04 (vezi 5.1). Un jucător cu inteligență artificială rulează pe client pentru a simula activitatea unui jucător uman.

Comparăm timpul necesar pentru completarea unei iterații pe client când variem calitatea grafică, respectiv dimensiunea scenariului (vezi Figura 5.2).

Când folosim grafica de calitate scăzută, 90% din iterații se încheie în mai puțin de 30ms, în timp ce folosind grafica de calitate ridicată, 90% din iterații se încheie în sub 60 ms. Dimensiunea hărții are o influență semnificativă, întrucat pentru harta mică 90% din iterații durează sub 30 de secunde, interval ce acoperă doar 70% din iterații pentru hărțile mari. în schimb, nu pare a fi of diferență semnificativă între hărțile cu aceeași dimensiune și densitate diferită.

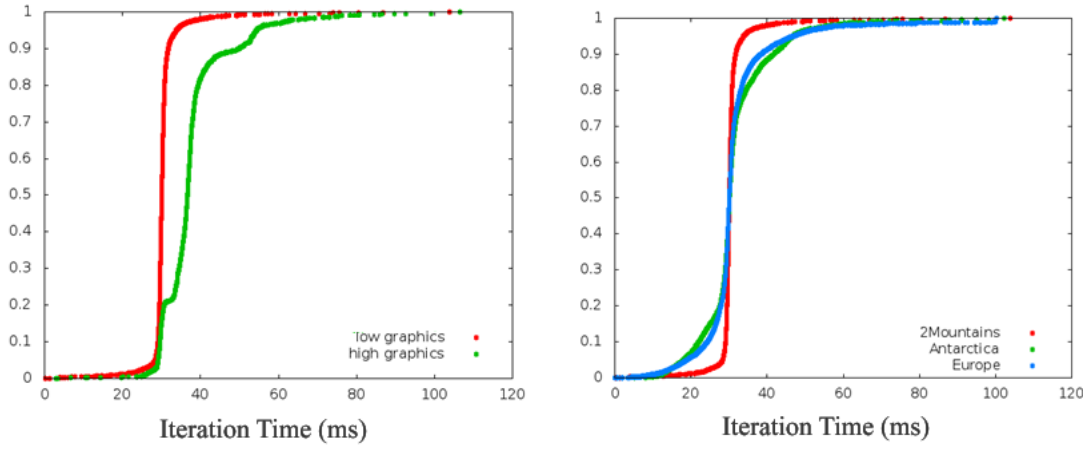


Figura 5.2: Calibrare: efectele parametrilor

Variația componentelor transferate

În acest experiment explorăm beneficiul adus de offloadingul jucătorilor cu inteligență artificială. Reprezentăm aportul acestora sub forma unei etape din bucla de procesare (vezi Secțiunea 4.4). Profilarea aplicației arată că aceștia pot solicita o bună parte din capacitatea de calcul, întrucât trebuie să determine rute prin algoritmi complecși, precum A*, mai ales atunci când reprezentarea lumii din joc este mare.

Prin acest experiment, studiem unul dintre cele patru mecanisme de offloading identificate în Spațiul de Explorare (vezi Figura 5.3). Practic, comparăm varianta fără offloading, în care 4 jucători artificial rulează pe client, cu cea în care o componentă este transferată, mai exact cei 4 jucători artificial rulează pe server. Repetăm experimentul de câteva ori, folosind o tabletă Asus Transformer TF101, însă clientul are probleme în a rula și este scos din joc de către server cu un mesaj de eroare care indică faptul că este prea încet. Este o practică comună în jocuri ca serverul să refuze clienți prea înceți, pentru a asigura corectitudinea jocului.

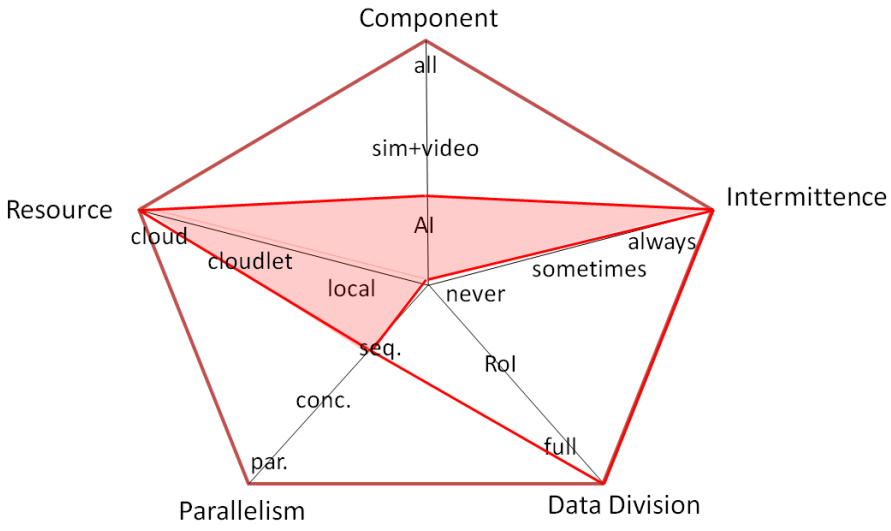


Figura 5.3: Acoperirea Spațiului de Explorare prin variația componentelor transferate

Figura 5.4 arată, pentru primele 12 minute al experimentului, două dintre metricile pe care le colectăm. Timpul per iterație (iteration time) este o metrică de performanță, ce arată cât durează parcurgerea unei iterații din bucla de procesare. Timpul din joc (in-game time) este o metrică ce se referă la calitatea experienței utilizatorului, care arată cât de mult timp a trecut în joc relativ la timpul din lumea reală. Această trecere a timpului are efecte mari asupra calității jocului, pentru că, dacă timpul trece prea încet, utilizatorul va avea sentimentul că jocul întârzie și va trebui să joace mai mult timp decât un utilizator pe laptop pentru a se bucura de aceleași rezultate.

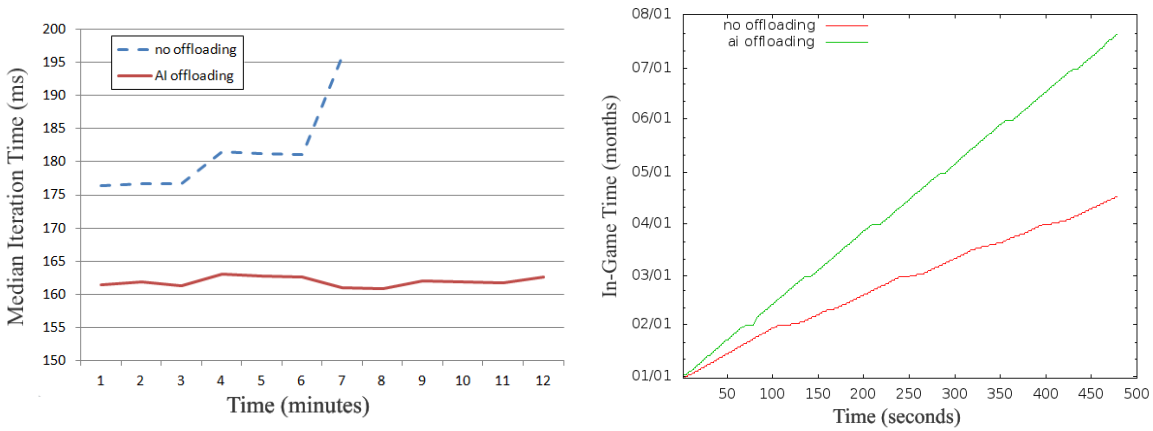


Figura 5.4: Rezultate privind performanța și experiența utilizatorului pentru variația componentelor transferate

Figura 5.4 arată cum, în primele 470 de secunde ale experimentului (cât au rulat ambele dispozitive) pentru varianta fără offloading au trecut aproximativ 3 luni, iar pentru varianta cu offloading au trecut aproape 7 luni. Graficul timpului per iterație arată valoarea mediană agregată la nivel de minut, care crește în mod repetat până este eliminat din joc.

Offloading Paralel

În acest experiment exploatăm faptul că rularea a mai mulți jucători artificiali este o sarcină ușor de paralelizat, întrucât fiecare jucător ia decizia următoarei mutări independent de ceilalți. Repetăm experimentul trecut, în care clientul mobil este doar un spectator, și rulăm cei 4 jucători artificiali pe server. Pentru comparație, rulăm cei 4 jucători artificiali pe mașina ce rulează serverul, în aplicații client separate. Mașina având un procesor cu 4 nuclee, va rula cele 4 procese în paralel.

Astfel, explorăm un alt mecanism de offloading din Spațiul de Explorare (vezi Figura 5.5).

Prezentăm rezultatul obținut pentru aceleași două metrici ca în experimentul precedent (vezi Figura 5.5). Timpul per iterație (iteration time) este o metrică de performanță, ce arată cât durează parcurgerea unei iterații din bucla de procesare. Timpul din joc (in-game time) este o metrică ce se referă la calitatea experienței utilizatorului, care arată cât de mult timp a trecut în joc relativ la timpul din lumea reală.

Timpul din joc arată o diferență semnificativă. În versiunea serială, pe parcursul celor 10 minute de timp real, se scurg 201 zile în joc, cu 11 zile mai puțin față de versiunea paralelă. Graficul timpului per iterație arată valoarea mediană agregată la nivel de minut, care nu

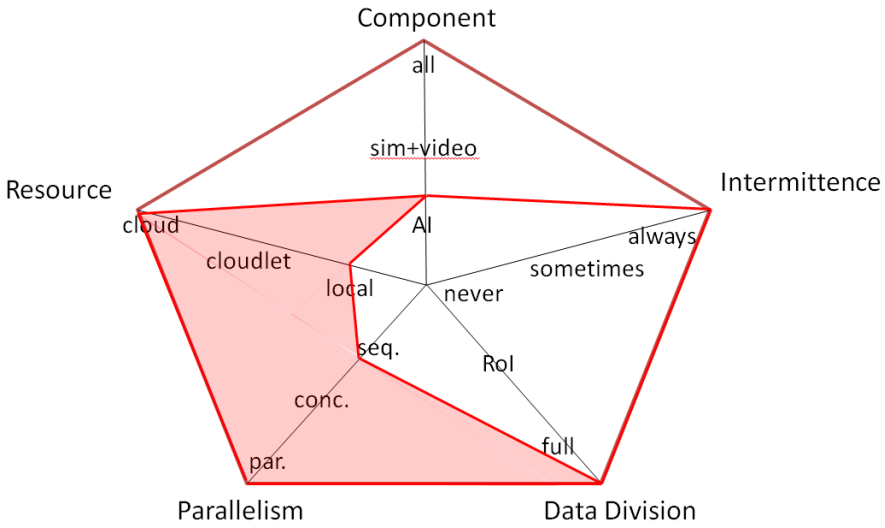


Figura 5.5: Acoperirea Spațiului de Explorare prin offloading paralel

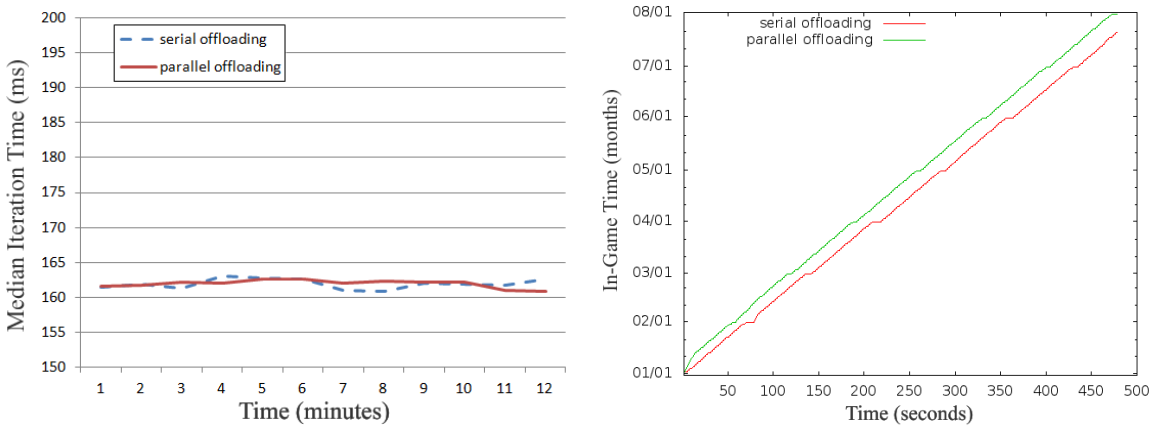


Figura 5.6: Rezultate privind performanța și experiența utilizatorului pentru offloading paralel

manifestă o diferență semnificativă, rămânând tot timpul în zona 160-165 ms. Acest lucru se datorează probabil felului în care am implementat paralelismul, prin utilizarea întregului client pentru fiecare dintre jucătorii artificiali, ceea ce face ca secțiunea de cod paralel să fie nesemnificativ de mică în comparație cu cea serială.

5.2 Evaluarea Analitică

În această secțiune folosim analiza operațională descrisă în Capitolul 4 și proiecțiile privind volumul de lucru prezentate în Capitolul 3 pentru a evalua două mecanisme de offloading folosind OpenTTD, aceeași aplicație folosită în evaluarea empirică din Secțiunea 5.1.

Parametrii Analizei

Conform Diagramei Tranziției între stări, un element al modelului pentru volumul de lucru propus de noi în Secțiunea 3.2, modelăm bucla de procesare a aplicației folosind un graf ciclic (vezi Figura 4.8). Pentru OpenTTD considerăm o buclă de procesare cu 5 etape: capturarea inputului utilizator, luarea unei decizii de către jucătorii artificiali, sincronizarea comenzilor, simularea și randarea. Astfel, graful are $n = 5$ etape.

Arătăm în caracterizarea și modelarea volumului de lucru al OpenTTD din Secțiunea 3.3 că timpul de procesare pentru Etapa 2, decizia jucătorilor artificiali, diferă între tabletă și laptop cu un ordin de mărime (în medie 48.2 ms pe tabletă, 4.3 ms pe laptop), în timp ce dimensiunea datelor de ieșire rămâne aproximativ la fel (circa 40 octeți). Pe de altă parte, procesarea Etapei 4, de simulare, nu diferă așa mult (în medie 9.3 ms pe tabletă și 2.6 ms pe laptop) și vine cu un cost suplimentar în transferarea datelor de ieșire (aproximativ 1MB, în loc de câțiva zeci de octeți).

Un alt parametru cheie este viteza rețelei. În experimentele noastre, folosim conexiuni WiFi, ce au o viteză nominală de până la 54 Mbps. însă, pot fi și alte servicii care folosesc rețeaua, iar viteza poate depinde de calitatea semnalului. Bazându-ne pe observațiile din evaluarea empirică, considerăm o viteză medie de $8Mbps$, adică $B = 1048.57B/ms$.

Variația componentelor transferate

Comparăm varianta fără offloading, cu cea în care se face offloading pentru rularea algoritmilor de decizie pentru jucătorii artificiali, folosind ecuația 4.9. Sumarizăm în Tabelul 5.1 valorile pe care le folosim, extrase din modelul volumului de lucru (vezi Secțiunea 3.3).

Tabela 5.1: Date folosite în evaluarea analitică

Stage name and number	**Local Processing**	**Remote Processing**	**Quantity of Data**
	T_p [ms]	T_r [ms]	Q [byte]
Human Input Collection (1)	1.2	0.7	40
AI Input Collection (2)	48.2	4.3	40
Command Synchronisation (3)	188.5	101.9	120
Simulation (4)	9.3	2.6	1,068,576
Rendering (5)	5.7	13.7	4,096,000

Bandwidth
B [bytes/ms]
1048.57

În cazul de bază, fără offloading, $k = j = l = 3$, indicând ca singura etapă ce se desfașoară la distanță sincronizarea datelor pe server:

$$T(3,3) = T_p^1 + T_p^2 + T_r^3 + T_p^4 + T_p^5 + \frac{Q_2 + Q_3}{B} = 166.45 \tag{5.1}$$

Similar, când se face offloading pentru Etapa 2, jucătorii artificiali, $k = 2$, $j = l = 3$ și formula devine:

$$T(2,3) = T_p^1 + T_r^2 + T_r^3 + T_p^4 + T_p^5 + \frac{Q_1 + Q_3}{B} = 122.55 \tag{5.2}$$

Timpul mai mic de procesare remote pentru Etapa 2 și dimensiunea comparabilă a datelor de ieșire, fac ca offloadingul pentru Etapa 2 să fie o opțiune bună.

Dacă considerăm Etapa 2' pentru a descrie rularea a 4 jucători artificiali în loc de 1, timpul de rulare va deveni:

$$T(3,3) = T_p^1 + T_p^{2'} + T_r^3 + T_p^4 + T_p^5 + \frac{Q_2 + Q_3}{B} = 311.05 \tag{5.3}$$

iar offloadingul Etapei 2' devine:

$$T(2',3) = T_p^1 + T_r^{2'} + T_r^3 + T_p^4 + T_p^5 + \frac{Q_1 + Q_3}{B} = 135.45 \tag{5.4}$$

Astfel, dacă rulăm local 4 jucători artificiali, pragul de procesare de 200ms este depășit și clientul va fi rejectat din joc. În acest caz, offloadingul este necesar, pentru că aduce un timp de procesare de sub 200ms.

Offloading Paralel

Am văzut că offloadingul a $N = 4$ jucători artificiali este benefic. Dorim să comparăm în ce măsură offloadingul în paralel poate aduce o imbunătățire față de cel serial. Aplicăm legea lui Amdahl, exprimată sub forma Ecuației 4.14 și presupunem ca pe acest task ușor de paralelizat porțiunea serială este $S = 0$ și obținem: $T_{r||}^j = \frac{T_r^j}{4}$).

Astfel, timpul de procesare pentru o iterație devine:

$$T(2'_{||},3) = T_p^1 + \frac{T_r^{2'}}{4} + T_r^3 + T_p^4 + T_p^5 + 4\frac{Q_1 + Q_3}{B} = 119.78 \tag{5.5}$$

Calculăm beneficiul relativ RB folosind Ecuațiile 5.3, 5.4 și 5.5 sub forma:

$$RB = \frac{T(3,3) - T(2',3)}{T(2',3) - T(2'_{||},3)} = 8.9\% \tag{5.6}$$

În acest caz, offloadingul paralel aduce un beneficiu adițional de 8.9% față de varianta serială.

5.3 Comparație

În această secțiune comparăm rezultatele obținute prin experimente și cele obținute prin analiză operațională pentru a valida rezultatele pe care le obținem pentru două mecanisme: offloading cu variația componentei și offloading paralel. Tabelul 5.2 rezumă datele obținute:

- *Cazul de Bază 1: Fără rețea* – rulăm OpenTTD doar pe dispozitivul mobil, care trebuie să proceseze 4 jucători artificiali

- *Cazul de Bază 2: Fără offloading* – rulăm OpenTTD în multiplayer, cu un client pe mobil și serverul pe laptop; clientul trimite comenzi pentru sincronizare, dar toate celelalte componente se desfășoară
- *Experiment 1: offloading serial al jucătorilor artificiali* - rulăm OpenTTD în multiplayer, pe mobil un client, iar pe laptop serverul procesează cei 4 jucători artificiali serial
- *Experiment 2: offloading paralel al jucătorilor artificiali* - rulăm OpenTTD în multiplayer mode, pe mobil un client, iar pe laptop serverul procesează cei 4 jucători artificiali în paralel

Tabela 5.2: Comparație între evaluarea empirică și analitică

Scenario	Average iteration time		Total data in 8 min	
	T [ms]		Q [bytes]	
	empirical	analytical	empirical	analytical
No network	>200	265.64	64,211	0
No offloading	>200	311.05	1,307,300	768,000
AI serial offloading	155.7	135.45	1,463,240	768,000
AI parallel offloading	154.41	119.78	1,473,267	768,000

Se poate observa că rularea a 4 jucători artificiali este prohibitivă pentru clienții mobili fără offloading, lucru confirmat de ambele metode. Din această cauză însă, nu am putut evalua empiric performanța acestui caz până la capăt. Offloading-ul serial și paralel nu par să difere prea mult. Evaluarea analitică estimează o îmbunătățire de 9% în privința timpului de procesare, care nu a putut fi confirmată empiric. Totuși rezultatele empirice arată o îmbunătățire de aproximativ 5% în privința scurgerii timpului în joc.

În privința comunicației, rezultatele empirice arată transferuri mai mari decât cele analitice, cauzate atât de transferuri ale altor aplicații, cât și de alte transferuri pe care OpenTTD le poate face și nu le-am putut lua în considerare în analiza operațională. Totuși, rezultatele celor două metode sunt proporționale.

Considerãm cã analiza operaționalã este un instrument promițător. Deși unele rezultate diferã semnificativ faţã de cele empirice, ideile ce guvernează mecanismele de offloading sunt confirmate de ambele seturi de rezultate.

Capitolul 6.

Concluzii

Teza se axează pe definirea unui spațiu de explorare destinat operațiunilor de offloading și pe utilizarea acestuia în evaluarea empirică și analitică a unor astfel de mecanisme pentru dispozitive portabile. În acest sens am structurat, în Capitolul 2 informațiile bogate din literatura de specialitate pe tema offloadingului pentru terminale mobile și propunem o Taxonomie și un Spațiu de Explorare pentru mecanismul de offloading. Mai departe, în Capitolul 3 propunem un model al volumului de lucru pentru aplicații sociale online, pe care îl folosim pentru a caracteriza și modela proiecțiile pe care le-am colectat pentru mii de aplicații de pe platforma Facebook și mai multe aplicații mobile native. În Capitolul 4 investigăm în detaliu modul în care diferitele mecanisme de offloading operează în cazul mai multor tipuri de aplicații și propunem un sistem formal ce poate fi utilizat pentru a efectua analiza operațională a mecanismelor de offloading din Spațiul de Explorare, și care poate fi implementată pentru orice aplicație care funcționează reiterând o buclă de procesare. În cele din urmă, în Capitolul 5 validăm sistemul nostru formal comparând rezultatele evaluării analitice cu rezultatele evaluării empirice. Considerăm că axarea pe un tip specific de aplicație poate aduce progrese în offloadingul pentru terminale mobile, păstrând, în același timp, posibilități bogate de aplicabilitate.

Ideea implementării acestor cercetări în mod specific pentru aplicații sociale online este nouă. Din perspectiva tehnologiei utilizate pentru implementarea aplicațiilor, abordarea noastră acoperă atât aplicațiile online, care transmit mesaje cu acțiunile utilizatorilor ce urmează a fi efectuate în server, cât și simulări strâns cuplate, care solicită intens capacitățile de procesare ale dispozitivului și comunică, de obicei, doar mesaje de comandă. Prin intermediul modelului nostru pentru volumul de lucru (Secțiunea 3.2) și analizei noastre operaționale (Secțiunea 4.5), generalizăm ambele tipuri de tehnologii și facem referire la orice tip de aplicație de tip buclă. Realizăm o explorare inovatoare a spațiului de concepție pe baza spațiului de explorare propus în Secțiunea 3.3.2, cu ajutorul evaluării empirice și analitice.

Activitatea noastră pe marginea modelării volumului de lucru a fost primită cu mare interes de comunitatea științifică, după cum o dovedesc articolele publicate pe această temă [41][42], și Premiul pentru Cea Mai Bună Lucrare pe care l-am primit cu ocazia celei de-a 13-a ediții a IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2013) desfășurat în Olanda. Spațiul de explorare este cuprinzător și versatil, date fiind numeroasele contribuții oferite până la această dată și nenumăratele oportunități de cercetare viitoare.

6.1 Contribuții Personale

Teza include multiple contribuții teoretice și practice în domeniul cercetării offloadingului pentru portabile. Principalele contribuții ale tezei sunt caracterizarea și modelarea volumelor de lucru ale aplicațiilor sociale online și explorarea spațiului de concepție pe care o realizăm pe baza evaluării empirice și analitice a diverselor mecanisme de offloading.

Redate mai detaliat, contribuțiile acestei teze sunt:

- propunem un Spațiu de Explorare (Secțiunea 2.3) ce relevă mecanisme noi de offloading, precum offloadingul paralel și parțial în aplicații bazate pe buclă de procesare, dar și oportunitatea unei explorări a spațiului de proiectare;

- colectăm proiecții ale aplicațiilor sociale online (Secțiunea 3.1) care conțin informații referitoare la volumele de lucru la nivel macro (număr de utilizatori) și la nivel micro (operațiuni ale utilizatorilor) pentru 16.000 de jocuri de pe platforma Facebook și mai multe aplicații mobile native;

- propunem un model al volumului de lucru (Secțiunea 3.2) cu mai multe componente: Distribuția Popularității și Tiparul de Evoluție, pentru informații la nivel macro, Proiecții la Nivel de Pachete și Diagrama Tranziției între Stări, pentru informații la nivel micro;

- efectuăm caracterizarea și modelarea (Secțiunea 3.3) folosind proiecțiile aplicațiilor sociale online pe modelul volumului de lucru, relevând faptul că aplicațiile obișnuite, cu un singur vârf al numărului de utilizatori, ating maturitatea timpuriu în durata lor de viață, iar caracteristicile de relaționare socială influențează tot mai mult experiența utilizatorilor în sfera aplicațiilor online;

- ilustrăm modul în care modelul nostru pentru volumul de lucru poate fi folosit pentru a anticipa traficul și tipologia de evoluție, precum și a genera proiecții sintetice;

- propunem un Algoritm de Interogare Adaptiv pentru aplicații orientate pe locație, ca mecanism de adaptare a comunicației, și arătăm practic ce beneficii poate aduce acest algoritm în contextul unei aplicații reale, de monitorizarea calității aerului (Secțiunea 4.1);

- propunem o arhitectură cu mai multe nivele de abstractizare ce permite dispozitivelor mobile să se conecteze la o rețea de control a casei inteligente printr-o extensie USB, ca mecanism de offloading a comunicației (Secțiunea 4.2);

- proiectăm și implementăm o aplicație de realitate augmentată, ce suprapune clipuri multimedia peste imaginea unui cod de bare QR prin funcționarea într-o buclă de procesare, și experimentăm cu mecanisme de adaptare a calculului pentru a îi îmbunătăți funcționarea (Secțiunea 4.3);

- modificăm un joc popular open-source, ce funcționează iterând o buclă de procesare, și îl folosim pentru a experimenta cu offloadingul calculului (Secțiunea 4.4);

- propunem un sistem formal (Secțiunea 4.5) destinat analizei operaționale a mecanismelor de offloading din Spațiul de Explorare, sistem ce poate fi folosit în cazul oricărei aplicații care rulează prin reiterarea unei bucle de procesare;

- proiectăm și implementăm un *testbed* bazat pe o aplicație reală și îl folosim pentru a desfășura evaluarea empirică pentru unele din mecanismele de offloading din Spațiul de Explorare (Secțiunea 5.1); comparăm rezultatele obținute astfel cu cele obținute prin evaluare analitică;

6.2 Direcții Viitoare

Offloadingul pentru terminale mobile reprezintă un subiect generos. Identificăm aici unele dintre direcțiile în care se pot aduce contribuții, în special axate pe modelarea volumului de lucru pentru aplicații sociale online și pe mecanisme de offloading destinate aplicațiilor de tip buclă.

Modelul volumului de lucru poate fi îmbunătățit pentru o mai bună acuratețe și pentru a include un număr mai mare de aplicații. În primul rând propunem studierea Tiparului de Evoluție folosind analiza seriilor temporale (Time-Series Analysis). Potrivit acesteia, studiem evoluția tiparelor înregistrate de trei componente (tendința, componenta sezonieră sau ciclică și componenta neregulată). Componenta neregulată acoperă variațiile bruște afișate de unele aplicații pe parcursul evoluției acestora. Eliminarea componentei neregulate și a celei sezoniere asigură o componentă mai fluidă a tendinței, care poate fi aproximată prin rezultate mai bune folosind fie modelul liniar, fie distribuții mai complexe [43]. În al doilea rând, contemplăm deja elemente noi din modelul volumului de lucru care pot include în ecuație acțiunile utilizatorului. În al treilea rând, continuăm să colectăm date și credem că utilizarea modelului pentru volumul de lucru pe seturi de date și mai bogate îi va spori acuratețea.

Intenționăm să propunem un model analitic bazat pe teoria cozilor [37] pentru a corespunde într-o mai bună măsură fluxului de procesare din sistemele de offloading bazate pe arhitecturi client-server. Modelul analitic poate oferi într-o manieră facilă și exactă mai mulți parametri ai unui sistem de offloading, cum ar fi disponibilitatea și fiabilitatea. De asemenea, intenționăm să ne extindem interesul asupra încărcării din infrastructura de calcul cu posibile rezultate în domeniul planificării și asigurării capacității. Ambele direcții pot servi drept bază pentru o mai amplă evaluare analitică a sistemelor de offloading și pot fi validate prin intermediul evaluări empirice.

Pe măsură ce aplicațiile mobile și utilizatorii mobili evoluează, eforturile de cercetare în sfera offloadingului pentru terminale mobile trebuie să evolueze în egală măsură. De exemplu, până acum câțiva ani execuția *multi-threading*, cu toate că era disponibilă la nivel de programare, nu a avut un impact efectiv asupra performanței deoarece majoritatea procesoarelor mobile aveau un singur nucleu. În ultimii ani dispozitivele mobile au devenit la fel de sofisticate precum calculatoarele personale, cu procesoare multinucleu și cu procesor grafic. Prin urmare, cercetările actuale în domeniul offloadingului trebuie să ia în calcul execuția multi-threading și paralelismul în dispozitiv când se execută procesul de offloading. De asemenea, se anticipează că în viitorul apropiat dispozitive de calcul chiar mai mici și mai puțin capabile, de tipul obiectelor wearable, vor deveni din ce în ce mai populare, obligându-ne să considerăm terminalele mobile nu o sursă, ci o țintă pentru procesul de offloading.

Lista Publicațiilor

Premii și Nominalizări

1. **Best Doctoral Symposium Paper** at CCGrid 2013, May 2013, The Netherlands, for the paper: 'Extending the capabilities of mobile devices for online social applications through cloud offloading'
2. **Research Grant** from the Romanian American Foundation to continue the Adaptive Query Algorithm presented in Section 4.1 during Nov 2013 - May 2014

Articole în Conference Proceedings

1. A. Gherghina, **A. C. Olteanu**, and N. Ţăpuş. A marker-based augmented reality system for mobile devices. In *RoEduNet*, 2013. Available: http://bit.ly/XXPLYV
2. A. Făsui, **A. C. Olteanu**, and N. Ţăpuş. Fault tolerant surveillance system based on a network of mobile devices. In *RoEduNet*, 2013. Available: http://bit.ly/WCOr0K
3. V. Zamfirache, **A. C. Olteanu**, and N. Ţăpuş. Collaborative learning assistant for android. In *RoEduNet*, 2013. Available: http://bit.ly/11cQ9u3
4. D. O. Rizea, D. Ş. Tudose, **A. C. Olteanu**, and N. Ţăpuş. Adaptive query algorithm for location oriented applications. In *RoEduNet*, 2013. Available: http://bit.ly/WGXRbg
5. **A.C. Olteanu**, G. D. Oprina, N. Ţãpuş, and S. Zeisberg. Enabling mobile devices for home automation using zigbee. In *CSCS*, pages 189–195. IEEE, 2013
6. **A. C. Olteanu**, A. Iosup, and N. Ţãpuş. Towards a workload model for online social applications: Icpe 2013 work-in-progress paper. In *ICPE*, pages 319–322. ACM, 2013
7. **A. C. Olteanu**, N. Ţăpuş, and A. Iosup. Extending the capabilities of mobile devices for online social applications through cloud offloading. In *CCGRID*, pages 160–163, 2013
8. M.A. Popescu, O. Sluşanschi, and **A.C. Olteanu**. New bounds of a measure in information theory. In *Dezvoltare durabilã în condiţii de instabilitate economicã*, 2013
9. S. Chiricescu, **A.C. Olteanu**, and N. Ţăpuş. Interacţiunea cu dispozitive mobile pe baza detecţiei mişcãrilor feţei. In *RoCHI*, 2013. In publication. Available:
10. A. Gherghina, **A. C. Olteanu**, and N. Ţăpuş. Measuring performance in application offloading for mobile devices. In *ICSCS*, 2013. In publication. Available:
11. A. Făsui, **A. C. Olteanu**, and N. Ţăpuş. A survey regarding grid and distributed computing using mobile devices. In *ICSCS*, 2013. In publication. Available:

12. V. Zamfirache, A. Eftenoiu, P. Iosif, **A. C. Olteanu**, and N. Ţăpuş. Extending the moodle course management system for mobile devices. In *ICSCS*, 2013. In publication. Available:
13. **A. C. Olteanu**, D. S. Tudose, and N. Ţăpuş. Energy-efficient user interaction with an off-grid building. In *ICSCS*, 2013. In publication. Available:
14. R. Prejbeanu, **A. C. Olteanu**, and N. Ţăpuş. Real time collaboration in cloud for tune-up android. In *RoEduNet (Fall)*, 2013. In publication. Available:

Articole de Jurnal

1. **A. C. Olteanu**, A. Iosup, N. Ţăpuş, and F. Kuipers. A workload evolution model for online social games. *Internet Computing*. submitted
2. **A. C. Olteanu** and N. Ţăpuş. Offloading for mobile devices: A survey. *UPB Scientific Bulletin*. submitted

Prezentări la evenimente ştiinţifice

1. **A. C. Olteanu**, D. S. Tudose, A. Voinescu, and N. Ţăpuş. Complete hardware and software solution for energy economy. In *WERC*, 2012
2. G. Oprina, **A.C. Olteanu**, D. S. Tudose, and N. Ţăpuş. Enabling mobile devices with medical diagnosis capabilities. In *WPA*, 2013
3. D. Rizea, D. S. Tudose, **A. C. Olteanu**, and N. Ţăpuş. Mobile application for pollution data. In *WPA*, 2013
4. **A. C. Olteanu**, R. I. Chioibasu, and N. Ţăpuş. Design of m-health solutions for psychology practitioners. In *WPA*, 2013‘

Rapoarte de Cercetare

1. **A. C. Olteanu**, A. Iosup, and N. Ţăpuş. Towards a workload model for online social applications: Extended report. Tech.Rep. PDS-2013-003, TU Delft, January 2013
2. **A. C. Olteanu**, A. Iosup, and N. Ţăpuş. Cloud offloading for mobile computing: A survey. Scientific report, UPB, 2013
3. **A. C. Olteanu**, A. Iosup, and N. Ţăpuş. Workload modeling for online social applications on mobile devices. Scientific report, UPB, 2013

Bibliografie

[1] Mark Weiser, Rich Gold, and John Seely Brown. The origins of ubiquitous computing research at parc in the late 1980s. *IBM systems journal*, 38(4):693–696, 1999.

[2] CNN. Facebook reaches one billion users, 2012. online, last access: Sept 2013, available: http://money.cnn.com/2012/10/04/technology/facebook-billion-users.

[3] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: elastic execution between mobile device and cloud. In *Proceedings of the sixth conference on Computer systems*, pages 301–314. ACM, 2011.

[4] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Aiolos: Middleware for improving mobile application performance through cyber foraging. *Journal of Systems and Software*, 85(11):2629–2639, 2012.

[5] P. Ghosh, N. Roy, and S.K. Das. Mobility-aware efficient job scheduling in mobile grids. In *CCGrid*. IEEE, 2007.

[6] K.A. Hummel and G. Jelleschitz. A robust decentralized job scheduling approach for mobile peers in ad-hoc grids. In *CCGrid*. IEEE, 2007.

[7] D. Bruneo, M. Scarpa, A. Zaia, and A. Puliafito. Communication paradigms for mobile grid users. In *CCGrid*. IEEE, 2003.

[8] A.T.A. Gomes, A. Ziviani, L.S. Lima, and M. Endler. Dichotomy: A resource discovery and scheduling protocol for multihop ad hoc mobile grids. In *CCGrid*. IEEE, 2003.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE Perv. Comp.*, 2009.

[10] Facebook. Facebook reaches one billion users, 2012. online, last access: May 2013, available: https://s3.amazonaws.com/OneBillionFB/Facebook+1+Billion+Stats.docx.

[11] 10 ways cloud computing will change in 2013. online, last access: Sept 2013, available: http://www.zdnet.com/10-ways-cloud-computing-will-change-in-2013-7000008364/.

[12] Heungsik Eom, Pierre St Juste, Renato Figueiredo, Omesh Tickoo, Ramesh Illikkal, and Ravishankar Iyer. Snarf: a social networking-inspired accelerator remoting framework. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, MCC '12, pages 29–34. ACM, 2012.

[13] Tim Verbelen, Pieter Simoens, Filip De Turck, and Bart Dhoedt. Cloudlets: Bringing the cloud to the mobile user. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 29–36. ACM, 2012.

[14] Gonzalo Huerta-Canepa and Dongman Lee. A virtual cloud computing provider for mobile devices. In *Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond*, MCS '10, pages 6:1–6:5, New York, NY, USA, 2010. ACM.

[15] Eduardo Cuervo, Aruna Balasubramanian, Dae-ki Cho, Alec Wolman, Stefan Saroiu, Ranveer Chandra, and Paramvir Bahl. Maui: making smartphones last longer with code offload. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 49–62. ACM, 2010.

[16] Mohammed Anowarul Hassan and Songqing Chen. Mobile mapreduce: Minimizing response time of computing intensive mobile applications. In *MobiCASE*, pages 41–59, 2011.

[17] S. Ou, K. Yang, and J. Zhang. An effective offloading middleware for pervasive services on mobile devices. *Pervasive and Mobile Computing*, 3(4):362–385, 2007.

[18] Xiaohui Gu, Klara Nahrstedt, Alan Messer, Ira Greenberg, and Dejan Milojicic. Adaptive offloading for pervasive computing. *Pervasive Computing, IEEE*, 3(3):66–73, 2004.

[19] Rajesh Krishna Balan, Darren Gergle, Mahadev Satyanarayanan, and James Herbsleb. Simplifying cyber foraging for mobile devices. In *Proceedings of the 5th international conference on Mobile systems, applications and services*, MobiSys '07, pages 272–285. ACM, 2007.

[20] Xinwen Zhang, Sangoh Jeong, Anugeetha Kunjithapatham, and Simon Gibbs. Towards an elastic application model for augmenting computing capabilities of mobile platforms. In *Mobile wireless middleware, operating systems, and applications*, pages 161–174. Springer, 2010.

[21] Radu-Corneliu Marin and Ciprian Dobre. Reaching for the clouds: contextually enhancing smartphones for energy efficiency. 2013.

[22] M. Ferber, T. Rauber, M.H.C. Torres, and T. Holvoet. Resource allocation for cloud-assisted mobile applications. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 400 –407, june 2012.

[23] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, pages 77–90. ACM, 2010.

[24] Karthik Kumar and Yung-Hsiang Lu. Cloud computing for mobile users: Can offloading computation save energy? *IEEE Computer*, 43(4):51–56, 2010.

[25] E. Lagerspetz and S. Tarkoma. Mobile search and the cloud: The benefits of offloading. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 117 –122, march 2011.

[26] Mahadev Satyanarayanan. Mobile computing: the next decade. *ACM SIGMOBILE Mobile Computing and Communications Review*, 15(2):2–10, 2011.

[27] Andreas Klein, Christian Mannweiler, Joerg Schneider, and Hans D Schotten. Access schemes for mobile cloud computing. In *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, pages 387–392. IEEE, 2010.

[28] Shaoxuan Wang and Sujit Dey. Rendering adaptation to address communication and computation constraints in cloud mobile gaming. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[29] R. Kemp, N. Palmer, T. Kielmann, and H. Bal. Energy efficient information monitoring applications on smartphones through communication offloading. *Mobicase*, 2012.

[30] Google, Inc. *Android Cloud to Device Messaging Framework*, 2012.

[31] Onlive has over 1.5 million active users. online, last access: Sept 2013, available: http://www.computerandvideogames.com/363989/onlive-has-over-15-million-active-users/.

[32] StatSoft. How to identify patterns in time series data: Time series analysis.

[33] D. O. Rizea, D. Ş. Tudose, **A. C. Olteanu**, and N. Ţăpuş. Adaptive query algorithm for location oriented applications. In *RoEduNet*, 2013. Available: http://bit.ly/WGXRbg.

[34] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010.

[35] Gian Paolo Perrucci, Frank HP Fitzek, and Jörg Widmer. Survey on energy consumption entities on the smartphone platform. In *Vehicular Technology Conference (VTC Spring), 2011 IEEE 73rd*, pages 1–6. IEEE, 2011.

[36] Aaron Carroll and Gernot Heiser. An analysis of power consumption in a smartphone. In *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, pages 21–21, 2010.

[37] Leonard Kleinrock. Queueing systems. volume 1: Theory. 1975.

[38] Raj Jain. *The art of computer systems performance analysis*, volume 182. John Wiley & Sons Chichester, 1991.

[39] Peter JB King. *Computer and communication systems performance modelling*. Prentice Hall International (UK) Ltd., 1990.

[40] A. Gherghina, **A. C. Olteanu**, and N. Ţăpuş. Measuring performance in application offloading for mobile devices. In *ICSCS*, 2013. In publication. Available: .

[41] **A. C. Olteanu**, A. Iosup, and N. Ţãpuş. Towards a workload model for online social applications: Icpe 2013 work-in-progress paper. In *ICPE*, pages 319–322. ACM, 2013.

[42] **A. C. Olteanu**, A. Iosup, N. Ţăpuş, and F. Kuipers. A workload evolution model for online social games. *Internet Computing*. submitted.

[43] B. Zhang, A. Iosup, J. Pouwelse, and D. Epema. Identifying, analyzing, and modeling flashcrowds in bittorrent. In *P2P*, pages 240–249. IEEE, 2011.

[44] A. Gherghina, **A. C. Olteanu**, and N. Ţăpuş. A marker-based augmented reality system for mobile devices. In *RoEduNet*, 2013. Available: http://bit.ly/XXPLYV.

[45] A. Făsui, **A. C. Olteanu**, and N. Ţăpuş. Fault tolerant surveillance system based on a network of mobile devices. In *RoEduNet*, 2013. Available: http://bit.ly/WCOr0K.

[46] V. Zamfirache, **A. C. Olteanu**, and N. Ţăpuş. Collaborative learning assistant for android. In *RoEduNet*, 2013. Available: http://bit.ly/11cQ9u3.

[47] **A.C. Olteanu**, G. D. Oprina, N. Ţãpuş, and S. Zeisberg. Enabling mobile devices for home automation using zigbee. In *CSCS*, pages 189–195. IEEE, 2013.

[48] **A. C. Olteanu**, N. Ţăpuş, and A. Iosup. Extending the capabilities of mobile devices for online social applications through cloud offloading. In *CCGRID*, pages 160–163, 2013.

[49] M.A. Popescu, O. Sluşanschi, and **A.C. Olteanu**. New bounds of a measure in information theory. In *Dezvoltare durabilã în condiţii de instabilitate economicã*, 2013.

[50] S. Chiricescu, **A.C. Olteanu**, and N. Ţăpuş. Interacţiunea cu dispozitive mobile pe baza detecţiei mişcărilor feţei. In *RoCHI*, 2013. In publication. Available: .

[51] A. Făsui, **A. C. Olteanu**, and N. Ţăpuş. A survey regarding grid and distributed computing using mobile devices. In *ICSCS*, 2013. In publication. Available: .

[52] V. Zamfirache, A. Eftenoiu, P. Iosif, **A. C. Olteanu**, and N. Ţăpuş. Extending the moodle course management system for mobile devices. In *ICSCS*, 2013. In publication. Available: .

[53] **A. C. Olteanu**, D. S. Tudose, and N. Ţăpuş. Energy-efficient user interaction with an off-grid building. In *ICSCS*, 2013. In publication. Available: .

[54] R. Prejbeanu, **A. C. Olteanu**, and N. Ţăpuş. Real time collaboration in cloud for tune-up android. In *RoEduNet (Fall)*, 2013. In publication. Available: .

[55] **A. C. Olteanu** and N. Ţăpuş. Offloading for mobile devices: A survey. *UPB Scientific Bulletin*. submitted.

[56] **A. C. Olteanu**, D. S. Tudose, A. Voinescu, and N. Ţăpuş. Complete hardware and software solution for energy economy. In *WERC*, 2012.

[57] G. Oprina, **A.C. Olteanu**, D. S. Tudose, and N. Ţăpuş. Enabling mobile devices with medical diagnosis capabilities. In *WPA*, 2013.

[58] D. Rizea, D. S. Tudose, **A. C. Olteanu**, and N. Ţăpuş. Mobile application for pollution data. In *WPA*, 2013.

[59] **A. C. Olteanu**, R. I. Chioibasu, and N. Ţăpuş. Design of m-health solutions for psychology practitioners. In *WPA*, 2013.

[60] **A. C. Olteanu**, A. Iosup, and N. Ţăpuş. Towards a workload model for online social applications: Extended report. Tech.Rep. PDS-2013-003, TU Delft, January 2013.

[61] **A. C. Olteanu**, A. Iosup, and N. Ţăpuş. Cloud offloading for mobile computing: A survey. Scientific report, UPB, 2013.

[62] **A. C. Olteanu**, A. Iosup, and N. Ţăpuş. Workload modeling for online social applications on mobile devices. Scientific report, UPB, 2013.