Universitatea POLITEHNICA din București

Facultatea de Automatică și Calculatoare,
Departamentul de Calculatoare

LUCRARE DE DIPLOMĂ

Suport pentru transferul aplicaţiilor ubicue

Conducător Ştiinţific:
Ș.l. Dr. Ing. Andrei Olaru

Autor:
Claudiu-Mihai Toma

București, 2017

University POLITEHNICA of Bucharest

Faculty of Automatic Control and Computers,
Computer Science and Engineering Department

BACHELOR THESIS

Support for ubiquitous application transfer

Scientific Adviser:
Ș.l. Dr. Ing. Andrei Olaru

Author:
Claudiu-Mihai Toma

Bucharest, 2017

Abstract

O problemă comună în zilele noastre o reprezintă dificultatea de a transmite date de la dispozitivele mobile către calculatoare. Sunt necesare cabluri, configurare manuală, driver-e, utilizarea de servicii centralizate ce folosesc conexiunea la Internet sau alte inconveniente ce afectează experienţa utilizatorului.

O soluţie simplă, uşor de utilizat şi intuitivă o reprezintă o aplicaţie distribuită care realizează automat o conexiune între un dispozitiv mobil şi o staţie prin simpla direcţionare a camerei telefonului către calculatorul destinaţie, după care se începe automat transmisia datelor, utilizatorul având astfel o intervenţie minimală.

Cuprins

Abstract **ii**

1 Introducere **1**
1.1 Contextul istoric . . . 2
1.2 Contextul actual . . . 3

2 Stadiul actual al modalităţilor de a transfera date între dispozitive **4**
2.1 Protocoale peste IP . . . 4
2.1.1 FTP . . . 4
2.1.2 SCP . . . 5
2.2 Transmisie prin cablu . . . 5
2.3 Soluţii centralizate . . . 6
2.3.1 Google Drive . . . 6
2.3.2 Dropbox . . . 7
2.4 Aplicaţii . . . 7
2.4.1 Monect . . . 7
2.4.2 Send Anywhere . . . 8

3 Descrierea soluţiei propuse **10**
3.1 Descriere generală . . . 10
3.2 Arhitectura aplicaţiei . . . 11
3.3 Biblioteci utilizate . . . 11
3.3.1 OpenCV . . . 12
3.3.2 ZXing . . . 14
3.3.3 Alte biblioteci . . . 14
3.4 Agentul Android . . . 14
3.4.1 Descriere generală . . . 14
3.4.2 Detalii de implementare . . . 15
3.4.3 Formarea topologiei . . . 18
3.5 Agentul PC . . . 18
3.5.1 Descriere generală . . . 18
3.5.2 Detalii de implementare . . . 19
3.5.3 Prelucrarea imaginilor . . . 21

4 Rezultate obţinute **24**
4.1 Avantaje . . . 24
4.1.1 Aplicaţie ce nu necesita pregătiri . . . 24

4.1.2 Aplicaţie intuitivă 25
4.1.3 Aplicaţie prietenoasă cu reţeaua 26
4.2 Limitări 28
4.2.1 Timeout-uri 28
4.2.2 Hardware 28
4.2.3 Scalabilitate 29

5 Concluzii şi dezvoltări ulterioare 34

Listă de figuri

1.1 Exemplu de cod QR [9] 1

2.1 Un sumar al funcţionalităţilor Thunderbolt 3 [21]. 6
2.2 Paşii necesari pentru a putea începe transferul de fişiere în Monect (prin FTP). 8
2.3 Exemplu de cod sub formă de cifre şi QR folosite în Send Anywhere. . 9

3.1 Arhitectura aplicaţiei. 12
3.2 OpenCV detectează şi împerechează aceleaşi puncte cheie din imagini diferite [14]. 13
3.3 Distribuţia actuală pe piaţă a versiunilor sistemului de operare Android [11]. 15
3.4 Ciclul de viaţă al unei activităţi Android [12]. 16
3.5 Codul QR ce apare atunci când este detectat un agent Android. 19
3.6 Exemplu de imagine a unei imprimante prelucrată cu ImageJ. 21
3.7 Imagini utilizate pentru testa algoritmii de comparaţie. 22

4.1 Sherry apare în lista de aplicaţii ce pot distribui conţinut. 26
4.2 Camera web, facilitate prezentă la majoritatea laptop-urilor [13]. 29

5.1 Ilustrarea conceptului de detectare a calculatoarelor de către agenţii Android [10]. 35

Listă de tabele

4.1 Timpii de răspuns măsuraţi de agentul Android ce rulează pe un telefon mobil performant. Fiecare valoare reprezintă timpul măsurat de când agentul Android începe transmisia unei capturi de imagine către agenţii PC activi până când sunt recepţionate răspunsurile prelucrării imaginii respective de la agenţii PC activi ce au primit imaginea. Liniile corespund unor capturi de imagine consecutive trimise agenţilor PC activi. Fiecare coloană corespunde câte unui experiment. Ultima linie conţine timpii medii de răspuns. 30

4.2 Timpii de răspuns măsuraţi de agentul Android ce rulează pe un telefon mobil mai puţin performant. Fiecare valoare reprezintă timpul măsurat de când agentul Android începe transmisia unei capturi de imagine către agenţii PC activi până când sunt recepţionate răspunsurile prelucrării imaginii respective de la agenţii PC activi ce au primit imaginea. Liniile corespund unor capturi de imagine consecutive trimise agenţilor PC activi. Fiecare coloană corespunde câte unui experiment. Ultima linie conţine timpii medii de răspuns. 31

4.3 Timpii de răspuns şi numărul de agenţi PC care au răspuns în scenariul în care există doi agenţi Android activi şi doi agenţi PC activi. Fiecare timp de răspuns reprezintă timpul măsurat de când un agent Android începe transmisia unei capturi de imagine către agenţii PC activi până când sunt recepţionate răspunsurile prelucrării imaginii respective de la agenţii PC activi ce au primit imaginea. Deoarece interacţiunea nu are loc întotdeauna cu toţi agenţii PC, există câte o coloană care specifică pentru fiecare timp măsurat cu câţi agenţi PC s-a interacţionat şi au trimis un răspuns. Fiecare tabel corespunde câte unui experiment. . . . 33

Capitolul 1

Introducere

Aflându-ne în epoca tehnologiei, dorim ca dispozitivele noastre să comunice cu cât mai mare uşurinţă, un caz particular fiind transmisia de date precum link-uri sau imagini, de la telefoanele inteligente către staţiile de tip desktop sau laptop cât şi în sens invers.

Această lucrare tratează problema menţionată anterior, având ca scop obţinerea unei aplicaţii distribuite ce va permite utilizatorilor să transmită informaţii prin simpla direcţionare a camerei telefonului către staţia destinaţie. Vom numi această aplicaţie Sherry.

În prezent, aplicaţia poate transmite sau recepţiona conţinut sub formă de imagini sau de şiruri de caractere către sau de la staţii care se pot afla în următoarele două situaţii:

1. staţii al căror ecran este vizibil, fiind afişat un cod QR pentru identificare;
2. staţii al căror ecran nu este vizibil dar care dispun de o cameră web ce va furniza imagini folosite de aplicaţie pentru a identifica în mod unic fiecare calculator.

Un exemplu de cod QR poate fi văzut în Figura 1.1.

Figura 1.1: Exemplu de cod QR [9]

Spre deosebire de majoritatea programelor de pe piaţă care necesită a fi configurate sau au nevoie de cabluri, aici conexiunile între dispozitive se realizează automat, dinamic, fără a necesita intervenţia utilizatorilor. Singura condiţie este aceea ca toate dispozitivele să se afle în aceeaşi reţea locală.

Atât datorită constrângerilor cauzate de diferenţa mare între platforme cât şi de funcţionalitatea foarte diferită pe care o pun la dispoziţie cele două categorii de

dispozitive, vom implementa practic două aplicaţii, una pentru telefoane şi una pentru calculatoare, încercând pe cât posibil să păstrăm o simetrie a codului sursă.

1.1 Contextul istoric

Am observat cu toţii rapiditatea evoluţiei tehnologiei, fiecare an fiind mai prolific decât cel anterior. In aceste condiţii, calculatoarele şi telefoanele mobile au continuat să devină din ce în ce mai performante, mai versatile şi uşor de utilizat. Aceste dispozitive au ajuns să facă parte din viaţa noastră de zi cu zi. Ele reprezintă practic o extensie a noastră şi au devenit astfel o necesitate.

Pentru a înţelege mai bine contextul în care ne aflăm, vom analiza pe scurt rapiditatea dezvoltării atât a calculatoarelor cât şi a telefoanelor mobile inteligente.

Primul calculator electronic programabil, Colossus, a apărut in decembrie 1943, în perioada celui de-al doilea război mondial. Acesta a fost construit de Aliaţi ca urmare a eforturilor depuse în decriptarea mesajelor produse de celebra maşină Enigma. Deoarece codurile folosite pentru criptarea mesajelor erau schimbate zilnic şi decriptarea manuală nu producea niciodată rezultate, faimosul inginer Tommy Flowers a decis să construiască un sistem programabil capabil să lucreze cu o viteză mult superioară celei umane. Colossus s-a dovedit a fi un real succes, acest dispozitiv ajutând la scurtarea semnificativă a războiului, astfel, salvând numeroase vieţi omeneşti [18].

Primul calculator personal, cu semnificaţia de "operat de o singură persoană", ENIAC, a apărut in 1946 şi era capabil să rezolve o clasă mare de probleme numerice.

În 1973 IBM produce primul calculator portabil, SCAMP (Special Computer APL Machine Portable) rezolvând astfel problema mobilităţii calculatoarelor.

De aici, conform legii lui Moore, hardware-ul are o creştere exponenţială a performanţei ajungând în zilele noastre limita de frecvenţă de 4-5 GHz, frecvenţă impusă de viteza de deplasare a electronilor. Cu toate acestea, performanţele continuă să fie îmbunătăţite prin creşterea gradului de paralelizare.

O categorie aparte a calculatoarelor îl reprezintă telefoanele mobile inteligente care sunt calculatoare miniaturizate având încorporate capabilităţile necesare pentru a interacţiona cu serviciile de telefonie mobilă.

În ciuda faptului că primul concept de dispozitiv care combină elemente de telefonie şi elemente computaţionale a fost gândit de Nikola Tesla în 1909, primul telefon inteligent de succes avea să apară 98 de ani mai târziu. In 2007 Apple Inc. lansează iPhone, un dispozitiv cu ecran multi tactil având ca principal mod de interacţiune atingerea ecranului cu degetele.

Un an mai târziu, in 2008 apare primul telefon care utilizează Android, HTC Dream. Chiar dacă rata de creştere a telefoanelor Android a fost mică la început, în 2010 acestea dobândesc deja o popularitate foarte ridicată iar în 2012 acestea domină piaţa mondială a telefoanelor inteligente, în momentul de faţă deţinând peste 80 de procente.

În ziua de astăzi toate dispozitivele noastre sunt inteligente şi au infrastructura necesară pentru a putea comunica între ele. Vom analiza în capitolul următor soluţiile existente pentru a rezolva problema transferului de date între aceste dispozitive şi neajunsurile lor.

1.2 Contextul actual

Nivelul tehnologiei din prezent este unul remarcabil în ceea ce priveşte creşterea mobilităţii dispozitivelor, uşurinţa în utilizare a acestora sau dezvoltarea extraordinară a competenţelor reţelelor şi viteza conexiunii la Internet. Toate aceste realizări nu ar fi fost posibile fără echipamente performante, rezultat al unui proces evolutiv relativ scurt.

Interesul nostru este axat pe trei mari ramuri tehnologice, ele formând infrastructura noii noastre aplicaţii. Acestea sunt:

1. calculatoarele personale;
2. telefoanele mobile (pentru scopul acestei lucrări, vom include aici şi tabletele);
3. reţeaua folosită pentru a interconecta dispozitive (care, de asemenea, oferă accesul la Internet).

Capitolul 2

Stadiul actual al modalităţilor de a transfera date între dispozitive

Am menţionat în capitolul anterior faptul că dispozitivele noastre devin treptat din ce în ce mai uşor de utilizat. Această uşurinţă în utilizare se traduce, în cele din urmă, la producerea unei cantităţi mari de date de către utilizatori, şi, implicit, în necesitatea de transfera aceste date între diverse dispozitive.

Astfel, a fost necesară dezvoltarea unor soluţii care să satisfacă o gama cât mai largă de consumatori, aceştia variind de la simpli utilizatori de telefoane mobile care doresc să transfere poze între dispozitivele personale până la companii ce doresc soluţii complexe şi sigure de backup.

Vom ilustra doar câteva categorii de soluţii propuse de-a lungul timpului, fiecare având propriile avantaje şi dezavantaje.

2.1 Protocoale peste IP

Printre primele soluţii la care n-am putea gândi sunt însăşi protocoalele de transfer de date menite să propună standarde care rezolvă cât mai multe probleme ce apar o dată cu dorinţa de a transfera date prin diverse medii. Aceste standarde sunt implementate de diverse aplicaţii care, împreună cu o interfaţă grafică, ne ajută la îndeplinirea unor sarcini simple precum distribuirea unui fişier.

Cele mai populare protocoale de transfer fişiere sunt FTP (File Transfer Protocol) şi SCP (Secure Copy Protocol).

2.1.1 FTP

FTP este un protocol standard de reţea folosit pentru a transfera fişiere intre un client şi un server prin intermediul unei reţele [17].

FTP este construit pe o arhitectură client-server şi utilizează două conexiuni separate între client şi server: una pentru control, prin intermediul căreia sunt transmise comenzi, şi una pentru date. Utilizatorii FTP se pot autentifica printr-un protocol de autentificare în text clar, în general sub formă de nume de utilizator şi parolă. De asemenea, FTP oferă utilizatorilor opţiunea de se pot conecta în mod anonim dacă serverul este configurat pentru a permite acest lucru. Pentru a proteja numele de utilizator şi parola şi a cripta datele, FTP este adesea securizat folosind SSL sau, mai nou, TLS.

Primele aplicaţii client FTP erau programe în linie de comandă dezvoltate înainte ca sistemele de operare să aibă interfaţă grafică. Aceste aplicaţii încă sunt integrate în sisteme de operare precum Windows, Linux şi Unix. Numeroşi clienţi FTP au fost dezvoltaţi de atunci pentru desktop-uri, servere, dispozitive mobile iar FTP a fost încorporat chiar în editoare de pagini web [2].

2.1.2 SCP

SCP reprezintă o modalitate de a transfera fişiere, în mod securizat, între două calculatoare. În cele mai multe cazuri, SCP referă atât protocolul cât şi programul cu acelaşi nume.

SCP este un protocol bazat pe protocolul RCP din BSD. SCP foloseşte SSH (Secure Shell) pentru a transfera date şi foloseşte acelaşi mecanism pentru autentificare, astfel asigurând autenticitatea şi confidenţialitatea datelor. Un client poate trimite fişiere către un server, având posibilitatea să includă şi atribute care pot descrie mai detaliat fişierul precum permisiunile acestuia sau data ultimei accesări. Clienţii pot de asemenea să solicite fişiere sau directoare de la un server. În mod implicit, SCP rulează peste portul TCP 22. Nu exista un RFC care să definească specificaţiile acestui protocol [5].

2.2 Transmisie prin cablu

O altă soluţie populară în zilele noastre o reprezintă utilizarea cablurilor de date. Acestea sunt folosite pentru a conecta direct dispozitivele mobile cu calculatoarele, permiţând transmisia de date printr-o conexiune serială.

Cele mai utilizate tipuri de cabluri sunt cablurile USB. Acestea au fost create în 1994 când 7 mari companii au decis să implementeze o soluţie pentru a înlocui multitudinea de conectoare existente în acea perioară. Companiile fondatoare au fost: Compaq, DEC, IBM, Intel, Microsoft, NEC şi Nortel. Astfel, a apărut un nou standard în industrie, USB. Acesta permitea rate mari de transfer, uşura configurarea software a dispozitivelor conectate şi extindea versatilitatea calculatoarelor [6].

În prezent, acestea sunt folosite, printre altele, pentru conectarea telefoanelor cu calculatoarele şi permit atât încărcarea cât şi transferul de date, un exemplu ce are performanţe extraordinare îl reprezintă Thunderbolt 3.

Thunderbolt 3 este o interfaţă hardware dezvoltată de Apple şi Intel ce permite conectarea diverselor periferice externe cu un calculator folosind USB Tip-C. O

descriere succintă a capabilităţilor acestei tehnologii poate fi văzută în Figura 2.1. Acesta combină PCI Express şi DisplayPort în două semnale seriale şi asigură alimentare în curent continuu, toate în acelaşi cablu. Se pot conecta până la 6 periferice la acelaşi port iar vitezele suportate pot atinge 40 Gbit/s.

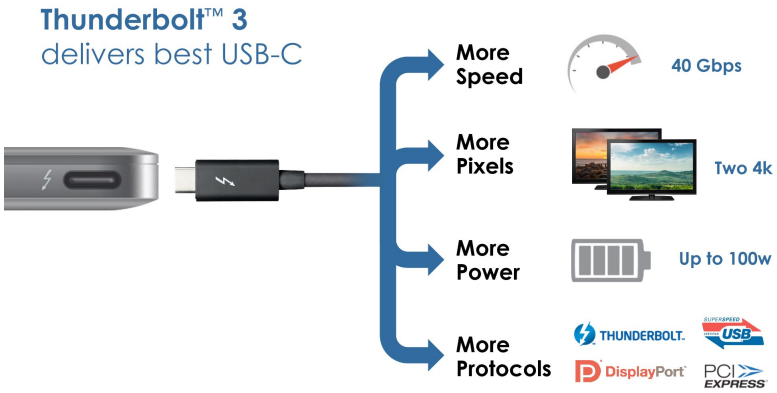


Figura 2.1: Un sumar al funcţionalităţilor Thunderbolt 3 [21].

2.3 Soluţii centralizate

Apărute în primul rând pentru a putea oferi unui număr mare de oameni acces la diverse date, soluţiile centralizate de stocare a fişierelor în cloud se bucură astăzi de o popularitate ridicată datorită, în principal, siguranţei datelor. În general, aceste soluţii oferă garanţii precum securitatea datelor, păstrarea integrităţii sau spaţii mari de stocare.

2.3.1 Google Drive

Google Drive este un mediu de stocare şi sincronizare de fişiere dezvoltat de Google. Acesta a fost lansat în 2012 şi permite utilizatorilor să îşi stocheze fişierele în cloud, să sincronizeze fişiere între diferite dispozitive şi să distribuie fişiere.

De asemenea, există aplicaţii care să ofere accesul offline la fişiere pentru Windows, macOS, Android sau iOS. Fiecare utilizator are accesul la 15 GB spaţiu de stocare gratuit [3].

Google Drive este capabil sa vizualizeze o multitudine de formate de fişiere dintre care amintim:

1. imagini (.JPEG, .PNG, .GIF);
2. videoclipuri (.MP3, .WAV);
3. fişiere text (.TXT);
4. arhive (.ZIP, .RAR);

5. fişiere Portable Document Format (.PDF);
6. fişiere Microsoft Office (.DOC, .DOCX, .PPT, .PPTX).

2.3.2 Dropbox

Dropbox este un alt exemplu de mediu de stocare online. Acesta a fost creat în 2007 de studenţii MIT Drew Houston şi Arash Ferdowsi fiind iniţial un startup ajutat de acceleratorul Y Combinator.

Utilizatorii au acces la 2 GB spaţiu de stocare gratuit şi aplicaţii care să faciliteze utilizarea serviciilor Dropbox pentru cele mai populare platforme precum Windows sau Android.

În 2010 a primit premiul Crunchie pentru cea mai bună aplicaţie pentru Internet [1].

2.4 Aplicaţii

Am văzut până acum că există numeroase modalităţi de a transfera date între dispozitivele noastre. Însă, în contextul evoluţiei accelerate a hardware-ului şi ţinând cont de faptul că dezvoltarea de aplicaţii cunoaşte o creştere substanţială, a apărut o nouă ramură care ne ajută să rezolvăm problema transferului de date: aplicaţiile pentru telefoane.

Succesul răsunător al sistemului de operare Android se datorează versatilităţii extraordinare de aplicaţii disponibile pentru această platformă. Această abundenţă de aplicaţii apare deoarece dezvoltarea lor este mult facilitată de:

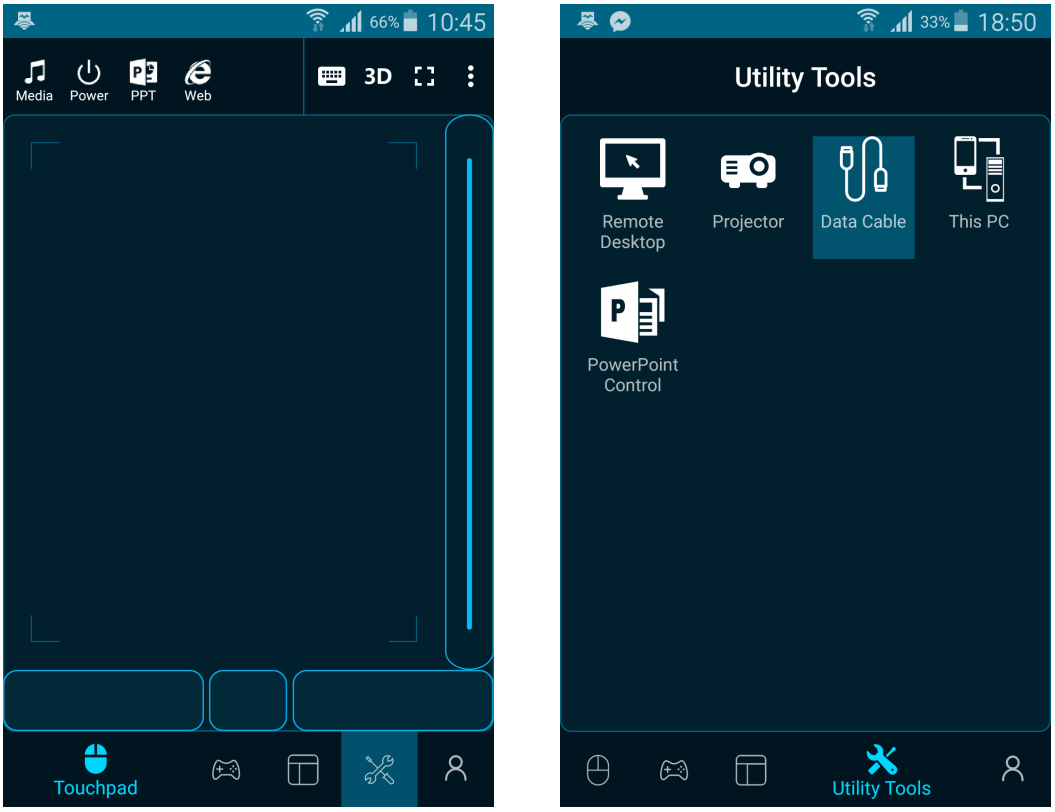
1. medii de dezvoltare performante precum Android Studio sau Eclipse;
2. o documentaţie adecvată menţinută de Google;
3. un API în continuă dezvoltare;
4. foarte multe biblioteci calitative open-source.

Astfel, dezvoltatorii sunt încurajaţi să scrie şi să publice aplicaţii a căror popularitate poate creşte rapid, în funcţie de calitatea acestora. Vom menţiona două aplicaţii ce pot fi utilizate pentru transferul de date: Monect şi Send Anywhere.

2.4.1 Monect

Monect este o aplicaţie gratuită ce beneficiază de o multitudine de funcţionalităţi. Iniţial, aplicaţia a fost concepută pentru a permite utilizatorilor să îşi controleze calculatoarele folosind exclusiv un telefon mobil. Acest lucru presupune implementarea unor mecanisme precum emularea mişcării cursorului, emularea tastaturii sau chiar transmisia ecranului calculatorului către dispozitivul mobil.

Deoarece popularitatea sa a crescut rapid, dezvoltatorii au decis să extindă opţiunile aplicaţiei adăugând posibilitatea utilizatorilor de a juca uşor diverse categorii de jocuri



(a) Deschiderea paginii de utilităţi.

(b) Selectarea opţiunii de cablu de date.

Figura 2.2: Paşii necesari pentru a putea începe transferul de fişiere în Monect (prin FTP).

şi chiar transferul de fişiere dintre dispozitive. Dezavantajul acestei abordări este dat de însăşi multitudinea de posibilităţi oferite utilizatorilor noi care pot avea dificultăţi în găsirea funcţionalităţilor pe care aceştia şi le doresc.

Funcţionalitatea de transfer de fişiere poate fi accesată precum în Figura 2.2. Transferul efectiv este realizat doar de pe calculator ceea ce face ca utilizarea acestei opţiuni sa fie neplăcută. Monect poate folosi coduri QR pentru identificarea calculatorului cu care dorim să ne conectăm, dar afişarea acestuia nu este realizată automat [15].

2.4.2 Send Anywhere

Send Anywhere este o aplicaţie de distribuit conţinut care este optimizată pentru transferul între dispozitive Android. Poate distribui inclusiv conţinut sub formă de şir de caractere, însă acesta este mai întâi încapsulat într-un fişier text. Cu toate că Send Anywhere are clienţi pentru diverse sisteme de operare desktop, aceştia presupun interacţiunea cu utilizatorul pentru recepţie.

Send Anywhere, precum Monect, poate folosi coduri QR pentru identificarea dispozitivelor între ele, însă această funcţionalitate este implementată doar pentru

transferul între telefoanele mobile; se poate folosi şi un cod format din cifre care are o valabilitate finită precum în Figura 2.3 [7].

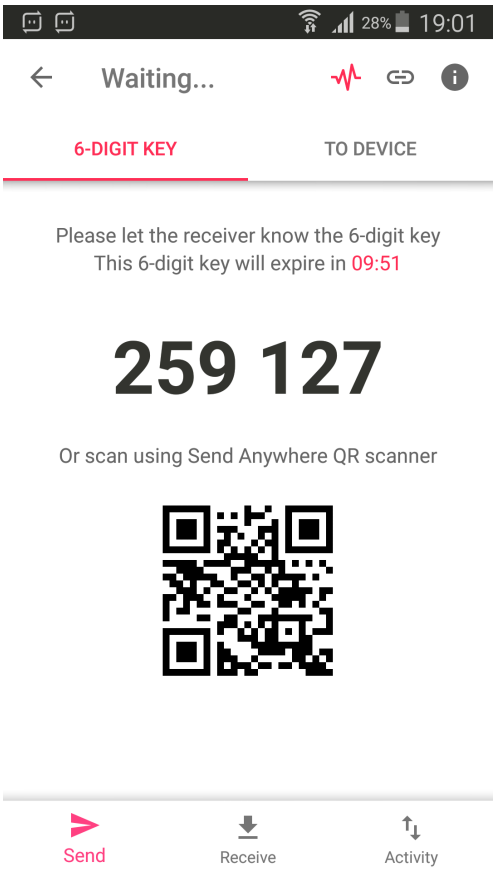


Figura 2.3: Exemplu de cod sub formă de cifre şi QR folosite în Send Anywhere.

Capitolul 3

Descrierea soluţiei propuse

În acest capitol ne vom axa pe de descrierea soluţiei alese. Pentru început, vom prezenta într-un mod general aplicaţia rezultat al acestei lucrări de licenţă cât şi modul ei de utilizare. Urmează să descriem bibliotecile folosite în cadrul aplicaţiei. Nu în ultimul rând, se va descrie în detaliu fiecare componentă a aplicaţiei.

3.1 Descriere generală

După cum am văzut până acum, marea majoritate a soluţiilor deja existente au diverse inconveniente: cabluri pentru interconectarea dispozitivelor, configurare manuală pentru a stabili conexiunea între dispozitive, driver-e specifice fiecărei versiuni de sistem de operare sau utilizarea de servicii centralizate ce folosesc conexiunea la Internet.

Ne propunem să găsim o soluţie simplă, care nu necesită cunoştinţe avansate din partea utilizatorului, să fie intuitivă şi, foarte important, să nu depindă de neajunsurile soluţiilor existente deja.

Soluţia găsita de noi este Sherry, o aplicaţie distribuită ce reprezintă în fapt o platformă multi-agent ai cărei agenţi sunt slab cuplaţi prin intermediul reţelei locale.

Aplicaţia va fi folosită astfel pentru a transmite conţinut de la telefon către calculator:

1. Se alege conţinutului care se doreşte să fie distribuit.
2. Se selectează Sherry din lista de aplicaţii capabile să distribuie conţinut.
3. Se orientează camera telefonului înspre calculatorul destinaţie, fie că se află cu faţa sau cu spatele.
4. Se aşteaptă identificarea calculatorului şi trimiterea conţinutului.

Pentru a transfera date în sens invers, de la calculator la telefon, utilizatorul va parcurge următorii paşi:

1. Se transmite un şir de caractere (care poate reprezenta atât conţinut sub formă de şir de caractere cât şi calea către un fişier) către intrarea standard a agentului pe PC.
2. Se porneşte aplicaţia Sherry pe telefon.
3. Se orientează camera telefonului înspre calculatorul destinaţie, fie că se află cu faţa sau cu spatele.
4. Se aşteaptă identificarea calculatorului şi trimiterea conţinutului.

Deoarece dorim să transferăm conţinut de la telefoanele mobile către calculatoare (şi invers), este evident faptul că avem nevoie de programe care să ruleze pe fiecare dispozitiv în parte. Pentru a ne uşura munca şi a face o administrare simplă a acestor programe, ele vor fi împărţite în două categorii: agenţi pentru Android şi agenţi pentru PC. Aceşti agenţi vor fi capabili să se identifice între intre ei şi să comunice unii cu alţii în mod automat, fără configurări din partea utilizatorului.

Un factor important de care se ţine cont este vizibilitatea ecranului calculatorului destinaţie. Astfel, Sherry va funcţiona atât când utilizatorul se aftă în faţa ecranului cât şi în spatele acestuia.

Raza de acţiune a aplicaţiei o reprezinta reţeaua locală, astfel, nu trebuie să ne facem griji pentru securitatea sau confidenţialitatea pachetelor.

În ciuda eforturilor noastre de a păstra o similitudine cât mai mare între aplicaţii, au existat factori majori care nu ne-au permis acest lucru. Cei mai importanţi sunt:

1. diferenţele foarte mari dintre platformele pe care cele două tipuri de agenţi rulează;
2. funcţionalităţile diferite pe care cele două tipuri de agenţi trebuie să le implementeze;
3. ciclii de viaţă ai celor doi agenţi sunt foarte diferiţi.

3.2 Arhitectura aplicaţiei

O imagine de ansamblu a arhitecturii este reprezentată în Figura 3.1. Aici putem vedea faptul că pot exista numeroşi agenţi din ambele categorii.

Fiecare agent Android va trimite un broadcast UDP către toţi agenţii PC aflaţi în reţeaua locală. Aceste mesaje vor fi folosite pentru afişarea codului QR şi stabilirea unei conexiuni TCP pentru comparaţia de imagini. Transferul de conţinut este efectuat în funcţie de orientarea camerei agenţilor Android.

3.3 Biblioteci utilizate

În realizarea acestui proiect au fost folosite o multitudine de biblioteci open-source care să faciliteze o dezvoltare rapidă, cu cât mai puţine probleme. Scopul precis al fiecăreia va fi menţionat după o scurtă descriere a acesteia.

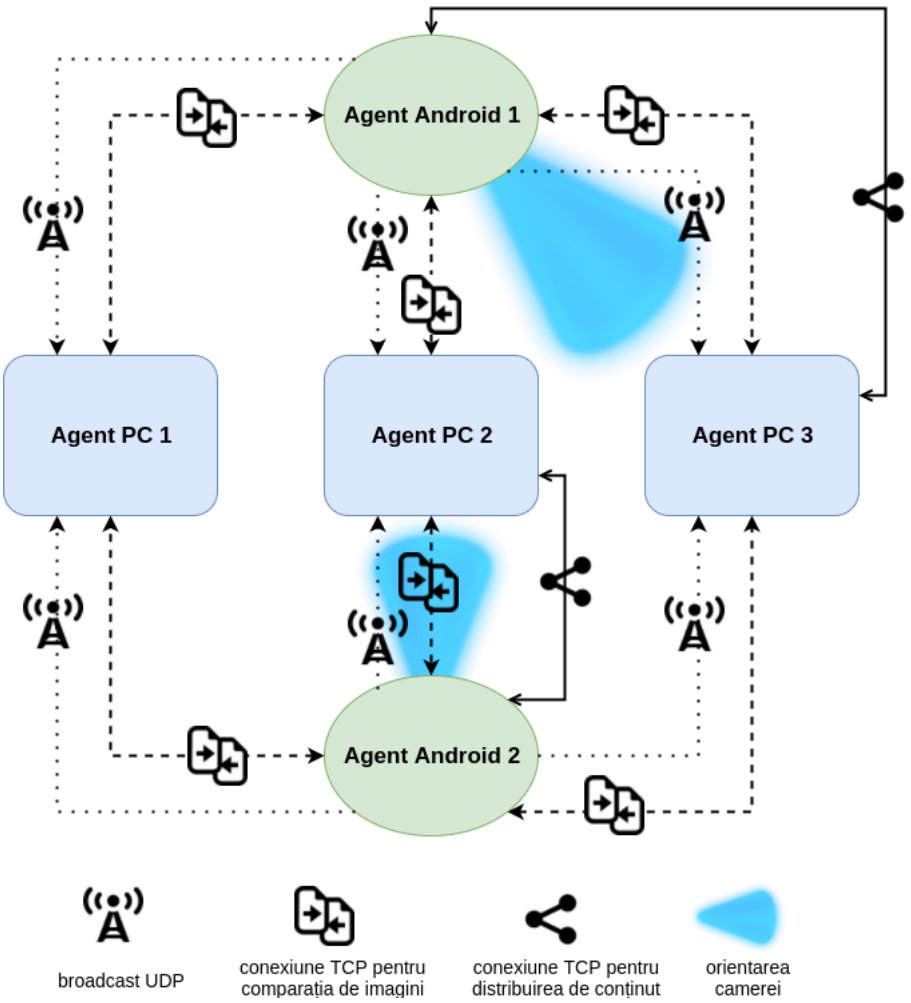


Figura 3.1: Arhitectura aplicaţiei.

3.3.1 OpenCV

OpenCV (Open Source Computer Vision) este o bibliotecă open-source al cărei scop central îl reprezintă prelucrarea în timp real al imaginilor. A fost iniţial dezvoltată de Intel apoi menţinută o perioada de Willow Garage. În prezent Itseez are în grijă dezvoltarea bibliotecii.

Prima versiune alpha OpenCV a fost lansată publicului larg la Conferinţa IEEE privind viziunea calculatoarelor şi recunoaşterea modelelor în anul 2000.

O a doua lansare majoră a OpenCV a avut loc în 2009, versiune ce aduce schimbări majore în interfaţa C++ a acesteia, principalele fiind creşterea uşurinţei în utilizare, adăugarea de noi funcţionalităţi şi îmbunatăţirea performanţelor celor existente (în special pe sistemele multiprocesor).

Această bibliotecă are numeroase aplicaţii din diverse domenii. Printre acestea se numără:

1. estimarea mişcării camerelor într-un spaţiu 3D;
2. sisteme de recunoaştere facială;
3. recunoaştere de gesturi;
4. interacţiunea om-calculator;
5. robotică mobilă;
6. înţelegerea mişcării;
7. identificarea de obiecte;
8. segmentare şi recunoaştere;
9. urmărirea mişcării;
10. realitate augmentată [4].

Una dintre funcţionalităţile utile nouă o reprezintă capacitatea e de a extrage puncte cheie dintr-o imagine. Un punct cheie reprezintă o zonă a imaginii pe care OpenCV este relativ sigur că o va identifica chiar dacă imaginea a fost rotită, a fost făcută de la o altă distanţă sau de la un unghi uşor diferit [8]. Putem vedea în Figura 3.2 cum OpenCV detectează aceleaşi puncte cheie în cele doua imagini.

Figura 3.2: OpenCV detectează şi împerechează aceleaşi puncte cheie din imagini diferite [14].

Vom utiliza această bibliotecă pentru a face posibil transferul de conţinut când ecranul calculatorului nu este vizibil, mai exact, când ne aflam în spatele acestuia.

Acest lucru este posibil utilizând funcţionalitatea de extragere de puncte cheie asupra imaginilor de la camera web a calculatorului şi de la camera telefonului, urmând să

comparăm cele două mulţimi de puncte cheie şi să constatăm câte elemente au acestea în comun.

3.3.2 ZXing

ZXing (Zebra Crossing) este o bibliotecă open-source ce aparţine Google, capabilă să scaneze şi să creeze coduri de bare multiformat 1D/2D. Biblioteca este implementată în Java şi are porturi disponibile în diverse alte limbaje de programare [16].

Această bibliotecă va fi folosita pentru a genera coduri QR la nivelul calculatorului şi pentru a scana aceste coduri la nivelul telefoanelor. Codurile vor reprezenta adrese IP care vor fi folosite ca destinaţie (sau sursă) a conţinutului.

3.3.3 Alte biblioteci

În cadrul acestei lucrări au fost utilizate şi alte biblioteci. Dintre acestea menţionăm:

1. webcam-capture de la Sarxos, utilizată pentru a obţine capturi de imagini de la camera web a calculatoarelor;
2. guava de la Google, utilizată pentru prelucrarea de fişiere;
3. ImageJ, utilizată pentru detecţia marginilor în imagini.

3.4 Agentul Android

Agentul Android reprezintă o instanţă a aplicaţiei Sherry ce rulează pe un dispozitiv mobil ce are sistemul de operare Android.

Soluţia este implementată în Android SDK API 16 pentru a acoperi peste 98% din dispozitivele cu sistemul de operare Android, conform distribuţiei din Figura 3.3 pusă la dispoziţie de Google, dispozitive care reprezintă peste 80% din piaţa telefoanelor mobile din ziua de astăzi, piaţă în continuă creştere. Un alt motiv pentru care a fost aleasă platforma Android îl reprezintă comunitatea uriaşă de dezvoltatori şi ajutorul nepreţuit al oamenilor cu experienţă oferit dezvoltatorilor începători care doresc să îşi concretizeze ideile şi să le pună la dispoziţie consumatorilor.

3.4.1 Descriere generală

Acest agent are două moduri de funcţionare, receptor şi transmiţător, pentru a oferi posibilitatea transmisiei bidirecţionale de date.

După ce aplicaţia este pornită, utilizatorul va vedea pe ecran ceea ce vede camera dispozitivului, pentru a uşura încadrarea unui calculator în vizorul acesteia.

Utilizatorul îşi va orienta camera telefonului spre un calculator şi va aştepta până când transferul se va face cu succes.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.8%
4.1.x	Jelly Bean	16	3.1%
4.2.x		17	4.4%
4.3		18	1.3%
4.4	KitKat	19	18.1%
5.0	Lollipop	21	8.2%
5.1		22	22.6%
6.0	Marshmallow	23	31.2%
7.0	Nougat	24	8.9%
7.1		25	0.6%

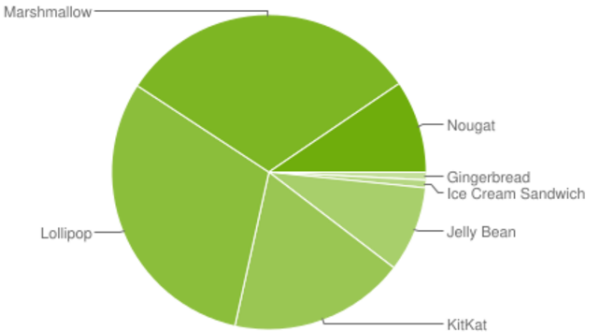


Figura 3.3: Distribuţia actuală pe piaţă a versiunilor sistemului de operare Android [11].

3.4.2 Detalii de implementare

Pentru început vom descrie conceptul de intenţie din Android, facilitate folosită şi de noi pentru a porni agentul Android în modul de transmisie.

În Android, pentru a spori securitatea, fiecare aplicaţie este pornită în propria sa maşină virtuală, Dalvik pentru versiuni API mai mici de 19 (inclusiv) şi ART (Android Runtime) pentru versiuni API mai mari de 21 (inclusiv). Cu această ocazie, toate comunicaţiile inter proces standard au fost eliminate (ex: pipe-uri).

Cu toate acestea, comunicaţia inter proces este esenţială într-un sistem inteligent, eficient şi flexibil. Astfel Android introduce conceptul de intenţie, un obiect ce are rolul unui mesaj prin intermediul căruia o componentă a unei aplicaţii poate solicita o acţiune din partea altei componente ale aceleiaşi aplicaţii sau din partea unei componente ce face parte din altă aplicaţie. Răspunsul este asincron si este livrat prin apelarea unei metode de callback a componentei ce a lansat intenţia [20].

Android este un mediu puternic asincron, acest lucru reflectându-se şi prin modalitatea de a crea aplicaţii. Fiecare aplicaţie conţine cel puţin o activitate, componentă centrală a unei aplicaţii a cărei scop principal îl reprezintă interacţiunea cu utilizatorul şi punerea în funcţiune a firelor de execuţie computaţional intensive.

Activităţile nu au puncte unice de intrare (nu există funcţia main). Acestea au o multitudine de puncte de intrare prin intermediul cărora se face administrarea ciclului

de viaţă al aplicaţiei, fiind apelate când diverse acţiuni sunt realizate de utilizator (ex: apăsarea tastei "înapoi", stingerea sau aprinderea ecranului, deschiderea altei aplicaţii).

In Figura 3.4 putem observa o versiune simplificată a ciclului de viaţă a unei activităţi Android. Toate acestea sunt realizate deoarece dispozitivele mobile au o resursă care trebuie utilizată cu grijă şi în cantităţi moderate: bateria. De exemplu, o aplicaţie bine scrisă îşi opreşte aproape complet activitatea când ecranul telefonului este stins sau când pur şi simplu aceasta nu are prim planul.

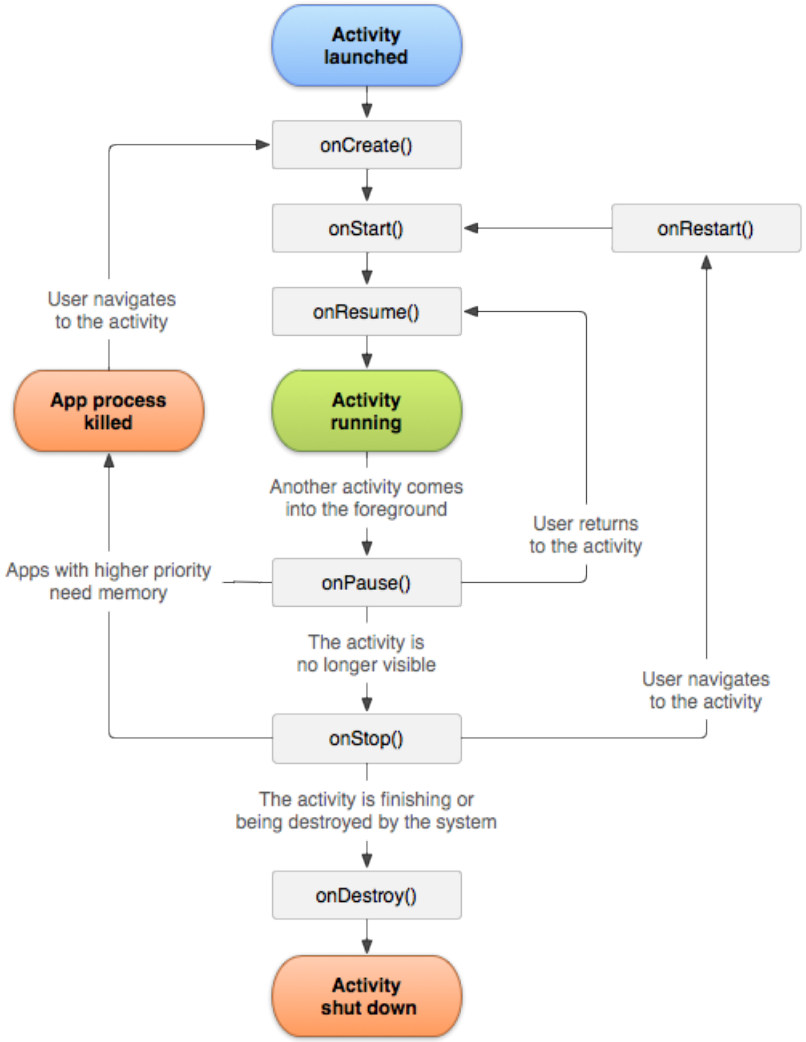


Figura 3.4: Ciclul de viaţă al unei activităţi Android [12].

Când un agent Android este pornit, primul lucru pe care îl face este să analizeze intenţia datorită căreia a fost lansat în execuţie. Dacă această intenţie nu există sau nu are detalii suplimentare, deducem faptul că aplicaţia a fost pornită manual de utilizator. În acestă situaţie Sherry porneşte în modul de recepţie.

Dacă, în schimb, au fost găsite detalii în cadrul intenţiei precum un şir de caractere sau un URI către o imagine, agentul va crea conţinutul ce va fi redirecţionat către un calculator şi va intra în modul de transmisie până când conţinutul este trimis cu succes

sau până când utilizatorul iese din aplicaţie.

În prezent, sunt suportate două tipuri de conţinut:

1. şir de caractere care, în cele mai multe cazuri, va fi un link;
2. imagine aflată fie în memoria telefonului fie pe Internet.

Lista tipurilor de conţinut suportat poate fi uşor extinsă deoarece, în cele din urmă, ceea ce va fi transmis este un sir de byţi. Pentru scopul acestei lucrări sunt suficiente aceste tipuri de conţinut deoarece sunt cele mai populare şi se încadrează în cele două tipare de dimensiuni importante:

1. şir de caractere, dimensiuni mici, nu necesită prelucrări suplimentare, transfer rapid;
2. imagine, dimensiuni mari, necesită prelucrări suplimentare, transfer mai lent.

O dată creat conţinutul, Sherry va crea şi iniţializa două obiecte ce se vor ocupa de captarea, prelucrarea şi afişarea cadrelor de la cameră respectiv trimiterea acestor cadre către calculatoare şi centralizarea răspunsurilor calculatoarelor.

Vom începe prin a descrie componenta ce face administrarea camerei numită CameraTimer. Aceasta, la rândul ei, conţine mai multe obiecte. Le vom menţiona pe măsură ce avem nevoie de ele. Pentru început, CameraTimer conţine o referinţă către camera din spate a telefonului utilizată pentru captarea propriu-zisă a cadrelor. Pe lângă aceasta mai avem un obiect de tipul ScheduledExecutorService utilizat pentru a livra periodic activităţii principale o captură de cameră. Nu în ultimul rând, avem un obiect de tipul Preview care schimbă setările camerei, prelucrează imaginea de tip YuvImage venită direct de la cameră într-un format Bitmap şi taie din marginile acesteia pentru a avea acelaşi raport lăţime/înălţime cu cel al ecranului dispozitivului mobil.

Cea de-a doua componentă foarte importantă o reprezinta obiectul de tipul PhonePictureStream. Aceasta are mai multe funcţionalităţi, dintre care amintim: menţinerea unei topologii temporare, transmisia de imagini prelucrate de la cameră către toate calculatoarele identificate şi recepţionarea răspunsurilor de la acestea, răspunsuri care semnifică un grad de asemănare între ceea ce vede telefonul şi ceea ce vede fiecare calculator din topologie. Vom vedea în subsecţiunea următoare cum are loc formarea topologiei.

Utilizând cele două obiecte descrise anterior, Sherry poate determina dacă există calculatoare a căror imagine seamănă suficient de mult cu propria captură de imagine de la cameră. Dintre acestea, se alege calculatorul care va specifica cea mai mare asemănare, fiind ales astfel ca destinaţie (dacă agentul Android este în modul de transmisie) sau, după caz, ca sursă (dacă agentul Android este în modul de recepţie) a conţinutului. Toate aceste operaţii se vor repeta până când transferul de conţinut cu un calculator are loc cu succes sau până când utilizatorul decide să părăsească aplicaţia.

Toate operaţiile computaţional intensive sau care necesită interacţiunea cu placa de reţea se fac pe fire de execuţie dedicate, astfel nu este suprasolicitat firul de execuţie al interfeţei grafice şi utilizatorul va avea o experienţă plăcută.

3.4.3 Formarea topologiei

Pentru a putea transmite imagini de la cameră către calculatoare, este necesară stabilirea unei topologii temporare. Conexiunile se vor realiza pe baza protocolului TCP pentru a asigura integritatea imaginilor. De asemenea, dorim să refolosim o conexiune deja existentă pentru a evita overhead-ul creării unei noi conexiuni.

În momentul în care o instanţă de Sherry este creată pe un telefon, aceasta trimite periodic mesaje broadcast UDP cu scopul de a-şi face cunoscută adresa IP.

Este mult mai eficient ca telefonul să emită aceste pachete deoarece, spre deosebire de calculator, dispozitivul mobil va avea aplicaţia Sherry pornită doar pe durata identificării destinaţiei şi a realizării transferului de conţinut. Astfel, reţeaua locală va fi populată un timp minim cu aceste mesaje de broadcast.

Calculatoarele ce vor recepţiona astfel de mesaje vor încerca să iniţieze o conexiune TCP către adresa IP identificată în pachetul UDP menţionat anterior, la un port prestabilit.

Administrarea reţelei are loc la executarea operaţiilor de trimitere şi recepţionare de date. Când una din aceste operaţii eşuează, conexiunea asociată este eliminată din lista de conexiuni active.

3.5 Agentul PC

Agentul PC reprezintă o instanţă a aplicaţiei Sherry ce rulează pe un calculator ce are un sistem de operare Windows sau Linux.

Un limbaj des folosit pentru aplicaţii portabile este Java, unul dintre limbajele care ajută la ruperea barierei create artificial de diversitatea platformelor. Este folosit în implementarea agentului ce rulează pe calculatoare, incluzând biblioteci open-source Java precum webcam-capture (de la SarXos) sau ZXing.

3.5.1 Descriere generală

După ce aplicaţia PC este pornită, utilizatorul va vedea pe ecran ceea ce vede camera dispozitivului mobil, pentru a uşura încadrarea unui calculator în vizorul acesteia. Aceasta este o facilitate implementată pentru a uşura procesul de testare si poate fi înlăturată fără dificultăţi.

Utilizatorul îşi va orienta camera telefonului spre un calculator şi va aştepta până când transferul se va face cu succes.

Acest agent, spre deosebire de cel Android, funcţionează concomitent şi ca receptor şi ca transmiţător. Pe lângă asta, activităţile computaţional intensive sunt desfăşurate doar când este detectat un agent Android. Toate acestea favorizează rularea continuă a aplicaţiei fără a consuma bateria laptop-urilor şi fără a consuma procesor, păstrând avantajul faptului că nu trebuie repornit de câte ori este nevoie de el.

Când agentul PC primeşte mesaje de broadcast de la cel puţin un agent Android, acesta afişează în colţul ecranului un cod QR care semnifică propria adresa IP în reţeaua locală precum în Figura 3.5. Acest cod va fi vizibil cât timp sunt detectaţi agenţi Android care vor să transfere conţinut. După ce se pierd conexiunile cu toţi agenţii Android, codul QR va dispărea, dar agentul PC va funcţiona în continuare.

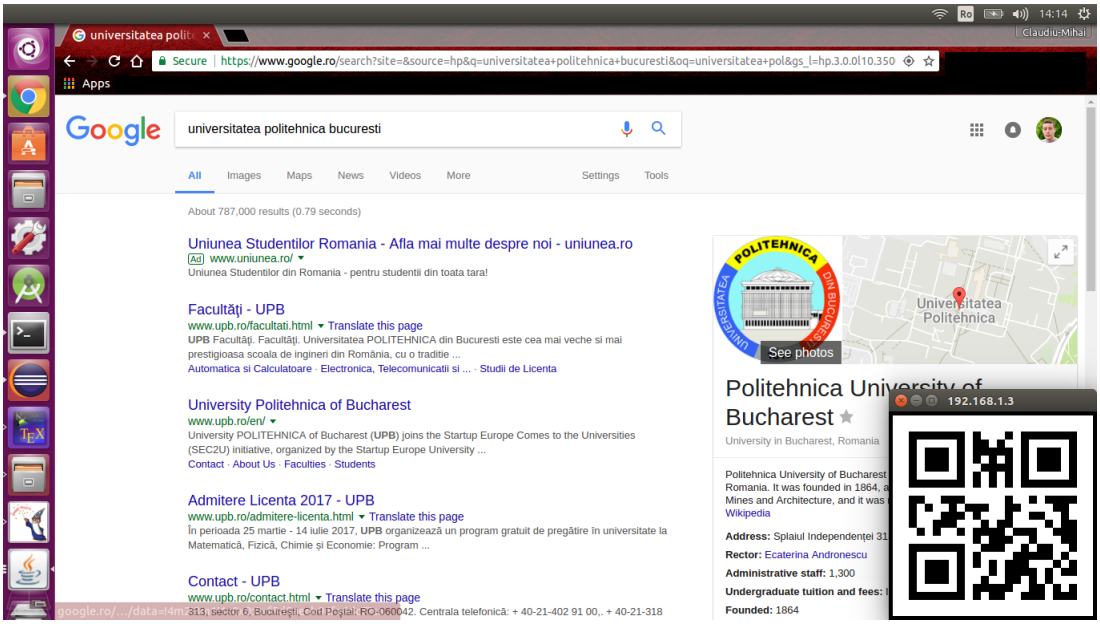


Figura 3.5: Codul QR ce apare atunci când este detectat un agent Android.

3.5.2 Detalii de implementare

În cazul agenţilor PC, clasele care realizează principalele sarcini sunt BroadcastBeaconReceiver şi ContentTransferServer.

Obiectul de tipul BroadcastBeaconReceiver va avea sarcina de a recepţiona mesajele de broadcast venite de la agenţii Android, va crea conexiuni temporare cu aceştia, va recepţiona fotografii şi va întoarce rezultatul prelucrării imaginii respective.

Operaţiile ce implică schimbul efectiv de date între agenţi sunt preluate de PhonePictureBeaconAction a cărui metode actionSuccess şi actionFailure sunt apelate când un mesaj de broadcast a fost recepţionat cu succes respectiv nu s-a recepţionat niciun mesaj de broadcast în limita de timp impusă. Acest obiect se ocupă de asemenea de afişarea sau ascunderea ferestrei ce conţine codul QR care encodează propria sa adresă IP. În continuare, vom analiza cele două metode ale PhonePictureBeacon pentru a înţelege mai bine cum are loc administrarea conexiunilor cu agenţii Android.

Pentru început vom descrie metoda actionSuccess, metodă ce primeşte ca parametru adresa IP a agentului Android al cărui mesaj de broadcast a fost recepţionat cu succes. Dacă instanţa curentă de PhonePictureBeaconAction nu conţine informaţii privind o conexiune cu adresa IP destinaţie primită ca parametru, aceasta este creată şi este

reţinută sub forma unei perechi de tip cheie-valoare, unde cheia este adresa IP iar valoarea este un obiect de tip BufferedImageReceiver care, la rândul său, implementează funcţionalitate de a recepţiona o imagine de la un anumit agent Android şi să trimită acestuia un întreg, răspuns al imaginii recepţionate. După ce se verifică existenţa unei conexiuni cu un telefon şi crearea acesteia dacă este necesar, se încearcă recepţionarea unei imagini de la agentul Android. În cazul în care această operaţie eşuează, conexiunea este înlăturată. În caz contrar, se solicită o captură a camerei web, se compară cele două imagini iar răspunsul este trimis telefonului pentru a decide cu ce agent PC va efectua transferul de conţinut.

Exista fireşte situaţii în care nu este recepţionat niciun mesaj de broadcast. Deoarece aceste mesaje sunt trimise cu o frecvenţă mult mai mare decât frecvenţa recepţionării acestora (cel puţin un ordin de mărime) vom presupune că absenţa broadcast-urilor înseamnă însăşi absenţa unor agenţi Android care doresc afişarea codului QR şi prelucrarea imaginilor de la cameră. În această situaţie este apelată metoda actionFailure a PhonePictureBeaconAction care, la rândul ei, va invalida toate conexiunile de prelucrare de imagini existente şi va ascunde fereastra ce conţine codul QR.

Complet independent, vom avea două obiecte de tipul ContentTransferServer, unul pentru transmisie şi unul pentru recepţie, cele două rulând pe porturi diferite. Acestea vor fi folosite în bucla principală de rulare a agentului. Periodic, se efectuează următoarele operaţii, în această ordine:

1. se verifică existenţa datelor de intrare de la intrarea standard; dacă acestea există, se citesc şi se interpretează;
2. dacă există conţinut care se doreşte a fi distribuit, se încearcă trimiterea acestuia; starea aplicaţiei este actualizată în funcţie de rezultatul acestei operaţii;
3. se încearcă recepţionarea de conţinut.

Interpretarea datelor de la intrarea standard constă în verificarea existentei unui fişier care are numele identic cu cel primit la intrare. În situaţia în care acesta există, conţinutul lui este citit şi se încercă distribuirea acestuia. Se poate însă ca numele primit la intrare să nu corespundă unui fişier. În această situaţie, intrarea este interpretată ca un simplu şir de caractere şi se încercă transmisia acestuia. Distribuirea unui anumit conţinut se poate termina în două situaţii: conţinutul a fost trimis cu succes sau a fost primit un alt nume de fişier sau şir de caractere de la intrare, caz în care vechi-ul conţinut nu va mai fi transmis.

Deoarece nu putem fi niciodată siguri când un agent Android va alege agentul PC curent pentru a efectua un transfer de conţinut, operaţiile de transmisie şi recepţie sunt încercate periodic pe toată durata de viaţă a agentului. Doar un agent Android va iniţia conexiuni, astfel componentele de tip server se vor afla de partea agenţilor PC.

De ce agenţii Android vor decide partenerul de conversaţie? Dispozitivele mobile vor fi privilegiate din acest punct de vedere deoarece, spre deosebire de calculatoare, acestea trebuie să utilizeze cât mai puţină baterie cu putinţă. Astfel, imediat ce un agent Android decide că este timpul să efectueze un transfer de conţinut, va face acest lucru, nu va aştepta ca un calculator să-l aleagă.

3.5.3 Prelucrarea imaginilor

În prezent doar agentii PC se ocupă de prelucrarea imaginilor. Această prelucrare constă într-un algoritm euristic ce întoarce o valoare care simbolizează gradul de asemănare al imaginilor. Cele două imagini sunt:

1. o imagine de la camera web a agentului PC;
2. o imagine de la camera foto a unui agent Android.

Iniţial, algoritmul euristic ce compara cele două imagini consta în trei etape:

1. redimensionare: imaginile erau redimensionate pentru ca ambele sa aibă aceleaşi dimensiuni;
2. detecţia marginilor: folosind biblioteca ImageJ imaginea era trecută printr-un proces de detectare a marginilor pentru a simplifica astfel cât mai mult imaginile, imagini care conţin acum, procentual, cât mai multă informaţie relevantă; imaginile rezultate arătau precum cea din Figura 3.6;
3. comparaţie la nivel de pixel: toţi pixelii erau comparaţi la nivel de componentă RGB şi era întors procentul de deosebire al pixelilor.

Figura 3.6: Exemplu de imagine a unei imprimante prelucrată cu ImageJ.

Procentul de deosebire al pixelilor era calculat după formulele:

$$procentaj = \frac{diferenta}{numar\_pixeli * 255} \tag{3.1}$$

$$diferenta = \sum_{x=0}^{latime-1} \sum_{y=0}^{inaltime-1} \sum_{c=\{R,G,B\}} abs\_pixel\_val(y, x, c) \tag{3.2}$$

$$abs\_pixel\_val(y, x, c) = |imagine_{webcam}[y][x][c] - imagine_{telefon}[y][x][c]| \tag{3.3}$$

$$numar\_pixeli = latime * inaltime * 3 \tag{3.4}$$

După efectuarea unor teste am constatat faptul că în marea majoritate a cazurilor, în ciuda faptului că imaginile erau asemănătoare, procentajul de deosebire obţinut de algoritmul de mai sus era foarte mare. Imaginile erau raportate ca fiind aproape complet diferite.

Este clar că această abordare nu este benefică aplicaţiei noastre.

O a doua abordare, una care a produs rezultate foarte bune, presupune folosirea de funcţionalităţi ale bibliotecii OpenCV. Pentru a compara mai uşor noul algoritm cu cel vechi, îl vom descrie în continuare:

1. identificarea punctelor cheie: folosind biblioteca OpenCV imaginea era trecută printr-un proces de identificare a punctelor cheie pentru a extrage astfel informaţia esenţială din imagini;
2. comparaţie la nivel de puncte cheie: cele două mulţimi de puncte cheie (aferente celor două imagini comparate de algoritm) erau comparate între ele iar algoritmul întorcea numărul de puncte cheie comune.

Este important sa întoarcem numărul de puncte cheie comune şi nu procentul de puncte cheie comune deoarece vom considera mai relevantă o comparaţie care are 50 de puncte comune din 200 (25%) decât o comparaţie care are 3 puncte comune din 6 (50%).

Folosind imaginile din Figura 3.7 obţinem următoarele rezultate:

1. prima abordare, comparaţia la nivel de pixel, ne spune că cele două imagini se aseamănă în proporţie de 13%, însă, pentru o persoană, ele sunt foarte asemănătoare;
2. a doua abordare, cea folosind OpenCV, ne spune că pot fi identificate 95 de puncte cheie comune aflate la distanţă mai mică de 45; un rezultat foarte bun.

Figura 3.7: Imagini utilizate pentru testa algoritmii de comparaţie.

De remarcat faptul că am utilizat mai devreme noţiunea de distanţă dintre două puncte cheie. Această reprezintă o valoare euristică folosită pentru a descrie pur numeric gradul de diferenţiere dintre două puncte cheie.

Capitolul 4

Rezultate obţinute

În acest capitol vom analiza rezultatele obţinute în cardul acestei lucrări. Vom vedea ce avantaje avem datorită abordării alese, ce dezavantaje şi limitări apar, dar şi sugestii de a rezolva câteva probleme.

4.1 Avantaje

Pentru început, vom analiza punctele tari ale aplicaţiei rezultat precum gradul redus de dificultate în utilizare sau traficul redus generat.

4.1.1 Aplicaţie ce nu necesita pregătiri

Principalul nostru scop în realizarea acestui proiect a fost implementarea unei aplicaţii capabile să distribuie conţinut între un calculator şi un telefon mobil fără a fi nevoiţi să apelăm la unul din următoarele impedimente:

1. cabluri;
2. configurare manuală;
3. driver-e;
4. utilizarea conexiunii la Internet.

Vom analiza pe rând toate aceste probleme, de ce dorim sa evităm utilizarea lor şi ce alternative am preferat.

Deşi pare cea mai comodă soluţie, utilizarea unui cablu de date nu este benefică telefonului. Acesta încarcă bateria fără ca noi sa ne dorim acest lucru şi uzează portul fizic utilizat pentru a ne încarcă telefonul. Uzând aceste componente telefonul poate deveni nefolosibil şi poate necesita reparaţii ceea ce înseamnă timp pierdut pentru utilizator şi se poate ajunge în situaţia extremă în care utilizatorul este nevoit să îşi procure un alt telefon. Pentru a evita toate acestea, noi am decis să utilizăm reţeaua locală, reţea la care toate dispozitivele noastre sunt deja conectate şi care reprezintă un mediu sigur şi rapid pentru transferul de conţinut.

Un alt mare obstacol pe care utilizatorii obişnuiţi îl întâlnesc adesea este configurarea manuală a dispozitivelor, configurare care solicită cunoştinţe de reţelistică pe care marea majoritate a oamenilor nu le deţin, ajungând din nou la timpul pierdut de utilizator, de această dată necesar pentru a înţelege ceea ce are de făcut şi cum îşi poate configura dispozitivele pentru a putea comunica. Aici apare reticenţa utilizatorilor care de cele mai multe ori aleg să nu folosească astfel de produse. Aplicaţia noastră nu solicită configurare manuală deoarece toate conexiunile se realizează automat, fără intervenţia utilizatorului.

Deoarece avem acces la o mare diversitate de sisteme de operare şi la numeroase versiuni ale acestora o problemă din ce în ce mai comună o reprezintă existenţa driver-elor compatibile cu respectivul sistem de operare pe care utilizatorul îl deţine. Noi nu depindem de utilizarea vreunui driver şi astfel nu trebuie să avem această grijă.

Nu în ultimul rând, există categoria soluţiilor care apelează la conexiunea la Internet pentru a realiza transferul de date. Utilizarea acestora este inconvenientă deoarece conţinutul pe care îl distribuim este accesibil unui număr mare de atacatori şi astfel intimitatea datelor poate fi compromisă. Utilizând reţeaua locală ne expunem mult mai puţin şi reducem riscurile pe care le presupune folosirea Internetului.

Am reuşit astfel sa evităm principalele inconveniente existente momentan pe piaţă şi, după cum vom vedea în continuare, aplicaţia noastră prezintă şi alte beneficii.

4.1.2 Aplicaţie intuitivă

Un mare avantaj al Sherry este faptul că am reuşit să obţinem o aplicaţie intuitivă. Acesta se datorează unui cumul de factori descrişi în continuare.

Agenţii pentru calculator nu trebuie porniţi manual mai mult decât o dată. Dacă sistemul de operare este configurat ca la pornirea acestuia să creeze şi un agent PC, utilizatorul nu mai trebuie să aibă în grijă această problemă. Agentul PC va funcţiona fără probleme şi fără să necesite mult procesor. Pe lângă cele menţionate anterior, acest tip de agent nu are nevoie de intervenţia utilizatorului pentru a recepţiona conţinut.

Pornirea agentului Android este realizată în mod natural atât în modul de transmisie cât şi în modul de recepţie. Când un utilizator doreşte să distribuie conţinut, va alege Sherry din lista de aplicaţii capabile să realizeze această sarcină precum în Figura 4.1, urmând apoi să îndrepte camera telefonului către calculatorul destinaţie. Această direcţionare este intuitivă deoarece se aseamănă mult cu utilizarea unei telecomenzi, dispozitiv folosit des de fiecare dintre noi, fie că aparţine unui televizor sau unei maşini. Însă, acesta nu trebuie orientat întocmai precum o telecomandă, ci precum o cameră de fotografiat. Acest lucru este foarte bine indicat de faptul că pe ecranul utilizatorului va apărea ceea ce vede camera dispozitivului mobil.

Pentru a intra în modul recepţie, utilizatorul trebuie doar să pornească aplicaţia a cărei iconiţă se află pe ecranul telefonului urmând apoi acelaşi procedeu descris mai sus pentru a recepţiona conţinut distribuit de agenţi PC.

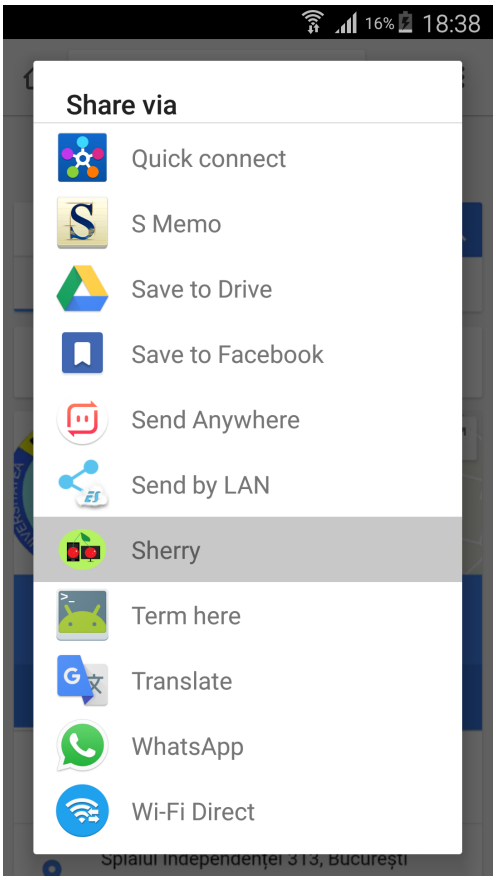


Figura 4.1: Sherry apare în lista de aplicaţii ce pot distribui conţinut.

4.1.3 Aplicaţie prietenoasă cu reţeaua

Pentru a înţelege mai bine impactul pe care soluţia propusă de noi îl are asupra reţelei, vom analiza următoarele două scenarii:

1. Ce se întâmpla atunci când un agent PC nu găseşte niciun agent Android.
2. Ce face fiecare agent din momentul în care doreşte să transmită conţinut până când termina de transmis.

Primul scenariu descrie practic starea în care reţeaua se va afla în marea majoritate a timpului, exceptând fireşte situaţia absenţei ambelor tipuri de agenţi deoarece utilizatorii vor dori să distribuie conţinut doar ocazional. Vom presupune, pentru a avea în vedere situaţia cea mai defavorabilă, că agentul PC va încerca să distribuie conţinut. După cum am menţionat în Capitolul 3, acţiunile pe care agentul PC le va realiza în ciclu sunt citirea intrării standard, încercarea de a transmite conţinut şi încercarea de a recepţiona conţinut.

În mod firesc, citirea intrării standard nu va afecta în niciun fel starea reţelei. Încercările de a transmite şi recepţiona conţinut se vor traduce în apeluri ale funcţiei accept din clasa ServerSocket, care, la rândul lor, vor suspenda firul curent de execuţie până la atingerea unui timeout sau până când se vor primi întreruperi de la placa de reţea,

întreruperi ce nu vor fi generate în absenţa unor agenţi Android care să încerce sa se conecteze la un calculator.

Am văzut astfel că în acest scenariu nu se va genera trafic în reţeaua locală iar, pe lângă asta, agentul PC nu consumă procesor deoarece îşi petrece majoritatea timpului în aşteptarea operaţiilor accept.

Cel de-al doilea scenariu reprezintă situaţia în care agenţii Sherry vor genera cel mai mult trafic. Întreg traficul generat se va încadra în următoarele categorii:

1. broadcast-ul de telefoanele mobile;
2. imaginile transmise de telefoane către calculatoare pentru a putea fi comparate cu capturi de cameră web;
3. răspunsul calculatoarelor ca urmare a comparaţiei de imagini;
4. transmisia efectivă a conţinutului.

Mesajele de broadcast sunt esenţiale pentru execuţia aplicaţiei noastre. Aceste mesaje nu conţin informaţii ce se doresc a fi transmise, scopul lor fiind acela de a anunţa agenţii PC de prezenta unor agenţi Android care foarte probabil vor trimite capturi de imagine pentru a fi prelucrate. Un alt rol important servit de mesajele de broadcast este acela de a face cunoscută adresa IP a fiecărui nou telefon ce doreşte să transfere conţinut. Datorită dimensiunilor reduse ale pachetelor UDP de broadcast şi a faptului că acestea nu necesită retransmisia la intervale mici de timp (este suficient ca intervalul de retransmisie să fie de 100 ms), broadcast-ul nu va polua reţeaua locală.

Cele mai mari pachete transmise prin intermediul reţelei de Sherry vor fi imaginile capturate de camerele dispozitivelor mobile. Pentru a nu supraîncărca reţeaua, dar şi pentru a consuma cât mai puţine resurse (încercam în mod special sa protejam bateria), noi am ales ca după ce o aceeaşi imagine este transmisă tuturor calculatoarelor detectate se va aştepta răspunsul acestora. După ce toate calculatoarele au prelucrat imaginea şi au anunţat rezultatul obţinut, agentul Android va hotărâ dacă va repeta procesul.

Deoarece din momentul în care este efectuată captura până în momentul în care au răspuns toţi agenţii PC poate dura între 1 şi 3 secunde, imaginile vor fi transmise relativ rar. În mod implicit, şi răspunsurile acestora ce constau adesea în simple numere vor fi transmise rar. Astfel, nici acest gen de trafic nu este problematic.

Pentru a proteja şi mai mult reţeaua, înainte ca fiecare imagine să fie transmisă, aceasta este scanată pentru a detecta prezenţa unui cod QR. În situaţia în care s-a putut identifica un asemenea cod, procesul de a transmite capturi de cameră este întrerupt şi se încearcă efectuarea unui transfer de conţinut cu adresa identificată în cod. În cazul în care transferul eşuează, întregul proces este reluat până când transferul de conţinut este efectuat cu succes sau utilizatorul a închis aplicaţia Sherry pe telefonul său mobil.

Fiindcă scopul acestei lucrări de licenţă este obţinerea unei aplicaţii capabile să distribuie conţinut între calculatoare şi telefoane iar reţeaua este unicul nostru mediu de comunicare, overhead-ul în reţea cauzat de transferul conţinutului necesită atenţie specială. Cu toate acestea, dimensiunea conţinutului este în general redusă, în marea majoritate chiar sub 5 MB, iar tehnici de compresie a datelor nu sunt necesare în

această situaţie. Astfel, transferul efectiv al conţinutului nu va provoca niciun fel de problemă.

Există, fireşte, scenarii extreme sau nefavorabile care creează iluzia faptului că Sherry are un impact negativ asupra reţelei precum încercarea unui agent Android de a transmite date într-o reţea în care nu se află niciun agent PC sau când aceştia se află la o distanţă fizică prea mare unul de celălalt pentru a avea loc identificarea destinaţiei. Dar având în vedere cele menţionate în această secţiune putem concluziona că Sherry nu va afecta aproape niciodată performanţele reţelei locale.

4.2 Limitări

Deoarece nu există nicio soluţie fără compromis, Sherry are câteva limitări ce apar datorită abordărilor alese pentru a ataca diverse probleme care s-au dorit a fi rezolvate.

4.2.1 Timeout-uri

Deoarece majoritatea operaţiilor pe reţea sunt blocante există riscul ca agenţii să rămână suspendaţi în aşteptarea realizării unei conexiuni sau în aşteptarea realizării unui transfer de date. Astfel, au fost setate timeout-uri tuturor socketţilor utilizaţi pentru a evita astfel de comportamente.

Cu toate că aceste timeout-uri ne ajută să evităm apeluri blocante definitive, pot crea la rândul lor alte probleme dacă nu sunt ajustate corespunzător. De exemplu, timeout-urile prea mari introduc întârzieri în cadrul aplicaţiei iar identificarea destinaţiei va dura foarte mult. La polul opus, timeout-urile prea mici introduc riscul ca anumite operaţii să eşueze deoarece agenţii nu găsesc momente de timp în care sa fie sincronizaţi. Aceste timeout-uri pot afecta performanţa aplicaţiei până în punctul în care aceasta nu mai este funcţională.

În prezent, timeout-urile sunt statice, alese pur empiric, ceea ce înseamna că în anumite situaţii nu reprezintă parametrii optimi.

4.2.2 Hardware

Singurul inconvenient de tip hardware pe care Sherry îl are constă în faptul că utilizatorul este constrâns de deţinerea unei camere web HD. Aceasta este folosită pentru a capta imagini ce vor fi folosite în algoritmul de comparaţie pentru imagini, algoritm ce ajută la identificarea celui mai potrivit calculator pentru a transfera conţinut.

Deoarece funcţionalitatea de a identifica un calculator care stă cu spatele este importantă pentru noi, camera web HD este un obiect indispensabil. De asemenea, calitatea ridicată a capturilor este necesară pentru a putea localiza cât mai bine punctele cheie.

Cu toate că marea majoritate a calculatoarelor de tip laptop dispun de o cameră web HD încorporată precum în Figura 4.2, vor exista utilizatori desktop ce nu vor putea beneficia de aplicaţia noastră datorită lipsei acesteia.

Figura 4.2: Camera web, facilitate prezentă la majoritatea laptop-urilor [13].

4.2.3 Scalabilitate

O altă limitare importantă constă în lipsa scalabilităţii. Vom prezenta în continuare o analiză a scalabilităţii bazată pe timpul necesar comparaţiei de imagini. Astfel va fi măsurat timpul de când un agent Android începe să trimită o captură de cameră tuturor agenţilor PC până când agentul Android primeşte toate rezultatele comparaţiilor.

Datorită diferitelor limitări tehnice întâmpinate, analiza scalabilităţii a fost efectuată cu două telefoane mobile şi două laptop-uri. Însă, chiar dacă infrastructura de testare este redusă, vom observa câteva rezultate interesante.

Pentru început, vom descrie într-un mod abstract (simplist) dispozitivele utilizate în efectuarea testelor:

1. telefon mobil performant;
2. telefon mobil cu performanţe slabe;
3. laptop cu sistem de operare Ubuntu;
4. laptop cu sistem de operare Windows;
5. router wireless de viteză mică (aproximativ 54 Mbit/s).

Pentru uşurinţa exprimării, vom referi aceste dispozitive prin nume sugestive oferite agenţilor ce rulează pe acestea: agent Android performant şi neperformant, agent PC Ubuntu şi Windows.

Având la dispoziţie aceste dispozitive vom începe prin a analiza timpul de răspuns în scenariile în care avem un singur agent Android. Rezultatele au fost sintetizate în Tabelul 4.1 şi în Tabelul 4.2.

Timp de răspuns (ms) cu agentul PC Ubuntu activ	Timp de răspuns (ms) cu agentul PC Windows activ	Timp de răspuns (ms) cu agentul PC Ubuntu activ şi cu agentul PC Windows activ
829	1868	1535
681	1602	1480
771	1530	1925
817	1589	1620
873	1498	1629
731	1632	2048
629	1595	1485
940	1572	1505
1083	1575	1480
738	1551	1569
653	1538	1664
877	1523	1734
688	1444	1625
692	1779	1709
849	1447	1546
739	1982	1483
740	1683	1568
754	1541	1663
710	1627	1634
668	1622	1641
773.1	1609.9	1627.15

Tabela 4.1: Timpii de răspuns măsuraţi de agentul Android ce rulează pe un telefon mobil performant. Fiecare valoare reprezintă timpul măsurat de când agentul Android începe transmisia unei capturi de imagine către agenţii PC activi până când sunt recepţionate răspunsurile prelucrării imaginii respective de la agenţii PC activi ce au primit imaginea. Liniile corespund unor capturi de imagine consecutive trimise agenţilor PC activi. Fiecare coloană corespunde câte unui experiment. Ultima linie conţine timpii medii de răspuns.

Timp de răspuns (ms) cu agentul PC Ubuntu activ	Timp de răspuns (ms) cu agentul PC Windows activ	Timp de răspuns (ms) cu agentul PC Ubuntu activ şi cu agentul PC Windows activ
1384	1651	1256
988	1652	1418
1442	1573	1705
972	1665	1557
772	1440	1214
794	1118	1299
1200	1352	1278
769	1373	1191
1046	1306	1347
671	1518	1265
1154	1228	1376
648	1099	1590
771	1339	1181
868	1575	1463
1587	1523	1429
830	1144	1492
656	1532	1438
890	1567	1255
803	1407	1513
750	1283	1347
949.75	1417.25	1380.7

Tabela 4.2: Timpii de răspuns măsuraţi de agentul Android ce rulează pe un telefon mobil mai puţin performant. Fiecare valoare reprezintă timpul măsurat de când agentul Android începe transmisia unei capturi de imagine către agenţii PC activi până când sunt recepţionate răspunsurile prelucrării imaginii respective de la agenţii PC activi ce au primit imaginea. Liniile corespund unor capturi de imagine consecutive trimise agenţilor PC activi. Fiecare coloană corespunde câte unui experiment. Ultima linie conţine timpii medii de răspuns.

După cum putem vedea, prima concluzie evidentă este faptul că agentul PC Ubuntu funcţionează mai bine decât cel Windows. Pot fi multe cauze pentru acest comportament, precum rularea unor sisteme de securitate suplimentare în cadrul Windows sau eficienţa mai mare a utilizării componentelor hardware de către Ubuntu.

Un alt aspect interesant al acestor date îl reprezintă diferenţa minoră dintre timpii medii în situaţia în care există doar agentul PC Windows şi situaţia în care sunt activi ambii agenţi PC. Acest rezultat nu reprezintă un semn al scalabilităţii, ci ne oferă o validare a faptului că transferul efectiv al imaginilor prin reţea are o durată mult mai mică decât durata necesară prelucrării imaginilor şi transmisiei răspunsurilor.

Nu în ultimul rând, putem vedea faptul că, în cazul agentului Android neperformant, obţinem timpi mai buni pentru Windows şi mai slabi pentru Ubuntu faţă de agentul Android performant. Reducerea timpilor în cazul Windows este cel mai probabil datorată calităţii reduse a imaginilor puse la dispoziţie de agentul neperformant având o camera foto mai slabă. Cât despre timpii mai mari în cazul interacţiunii cu agentul PC Ubuntu, aceştia erau de aşteptat datorită hardware-ului şi software-ului inferioare prezente pe agentul Android neperformant.

Din aceste date nu putem deduce încă absenţa scalabilităţii, ci doar alte proprietăţi ale Sherry. Problemele apar însă când în reţea există mai mult de un agent Android la un moment dat. Am efectuat teste în scenariul în care sunt prezenţi toţi cei patru agenţi şi am sintetizat rezultatele în Tabelul 4.3.

Având la dispoziţie aceste rezultate putem vedea clar faptul că Sherry nu scalează din punct de vedere al numărului agenţilor Android. Acest lucru este reflectat de faptul că agenţii Android vor nu putea interacţiona întotdeauna cu toţi agenţii PC. Pe lângă asta, se vede că agentul Android performant "acaparează" comunicarea cu calculatoarele, agentul Android neperformant comunicând cu agenţii PC doar în proporţie de 38% din timpul petrecut in reţea.

Timp de răspuns (ms)	Număr agenţi care au răspuns
1015	2
997	2
1140	2
955	2
947	2
971	2
1025	2
2041	2
968	2
1039	2
937	2
1057	2
1041	2
3117	1
893	1
920	2
1022	2
1119	2
1003	2
893	2
1068	2
911	2
1037	2
892	2
912	2
3106	1
970	1
638	1
1067	1
554	1
1141.83	1.76

(a) Timpii măsuraţi pentru agentul Android ce rulează pe un telefon mobil performant.

Timp de răspuns (ms)	Număr agenţi care au răspuns
186	0
1219	1
1390	1
1205	2
1413	2
3358	1
957	1
1640	1
3233	0
131	0
234	0
1253	1
3097	0
55	0
159	0
159	0
56	0
103	0
1317	1
1363	1
3159	1
836	1
750	1
668	1
1343	2
3668	1
1252	1
1369	1
1324	1
1464	1
1278.7	0.76

(b) Timpii măsuraţi pentru agentul Android ce rulează pe un telefon mobil neperformant.

Tabela 4.3: Timpii de răspuns şi numărul de agenţi PC care au răspuns în scenariul în care există doi agenţi Android activi şi doi agenţi PC activi. Fiecare timp de răspuns reprezintă timpul măsurat de când un agent Android începe transmisia unei capturi de imagine către agenţii PC activi până când sunt recepţionate răspunsurile prelucrării imaginii respective de la agenţii PC activi ce au primit imaginea. Deoarece interacţiunea nu are loc întotdeauna cu toţi agenţii PC, există câte o coloană care specifică pentru fiecare timp măsurat cu câţi agenţi PC s-a interacţionat şi au trimis un răspuns. Fiecare tabel corespunde câte unui experiment.

Capitolul 5

Concluzii şi dezvoltări ulterioare

În urma procesului de dezvoltare am reuşit să obţinem o aplicaţie, Sherry, capabilă să transmită conţinut sub formă de şir de caractere sau imagini atât de la un telefon către un calculator cât şi în sens invers. Au fost alese aceste tipuri de conţinut deoarece mesajele text, link-urile şi fotografiile reprezintă cele mai răspândite tipuri de conţinut. Un alt motiv pentru care cele două tipuri de conţinut sunt suficiente pentru această lucrare îl reprezintă uşurinţa în a extinde funcţionalitatea curentă astfel:

1. orice tip de text (mesaj telefonic, link, o replică dintr-o conversaţie) poate extinde distribuirea curentă de text pentru a putea manifesta un anumit comportament la recepţie precum copierea în clipboard sau deschiderea unui browser;
2. orice tip de fişier (videoclip, fişier text, fişier PDF) poate fi distribuit prin extinderea modalităţii curente de a transfera imagini deoarece acestea sunt transmise sub formă de titlu, sau numele fişierului, şi conţinutul acestuia ca şir de byţi;

Sherry este o aplicaţie intuitivă, uşor de utilizat, care consumă un minim de resurse pentru a-şi îndeplini sarcinile. Uşurinţa în utilizare este dictată de intervenţia minimală a utilizatorului care nu trebuie să cunoască numele destinaţiei şi nici detalii tehnice precum adresarea IP sau utilizarea de porturi. Acesta trebuie doar să îndrepte, în mod natural, telefonul spre staţia cu care doreşte să efectueze un transfer de conţinut.

Cu toate că Sherry este o aplicaţie eficientă din punct de vedere al consumului de resurse, aceasta poate ajunge să nu mai funcţioneze corespunzător în prezenţa unui număr mare de agenţi.

Având la dispoziţie aceasta soluţie ne putem gândi la dezvoltări ulterioare ale acesteia pentru a îmbunătăţi performanţele şi a extinde funcţionalitatea fără a renunţa la principalele noastre avantaje. Printre acestea putem număra:

1. îmbunatăţirea scalabilităţii;
2. implementarea opţiunii de a distribui mai multe fişiere simultan;
3. suportarea unui număr mai mare de tipuri de conţinut;
4. îmbunatăţirea agentului PC pentru a putea interacţiona mai uşor cu acesta;

5. detectarea vizuală a calculatoarelor de către agenţii Android şi evidenţierea acestora precum în Figura 5.1 [19];
6. actualizarea dinamică de timeout-uri în funcţie de performanţele/încărcarea reţelei.

(a) Calculator nedetectat. (b) Calculator detectat.

Figura 5.1: Ilustrarea conceptului de detectare a calculatoarelor de către agenţii Android [10].

Îmbunatăţirea scalabilităţii este un factor ce merită luat în considerare în cazul în care se doreşte folosirea Sherry într-un mediu industrial în care vor fi foarte mulţi agenţi prezenţi în cadrul aceleiaşi reţele locale. Un exemplu de scenariu în cadrul căruia scalabilitatea joacă un rol important poate fi următorul: ne aflăm în cadrul unei companii al cărei principal domeniu de activitate îl reprezinta ştirile online iar angajaţii nu se vor afla constant la un anumit calculator. Acest scenariu descrie un mediu foarte dinamic în care interacţiunea între angajaţi este foarte importantă. Distribuirea rapidă şi eficientă de text şi imagini ne este de folos iar capacitatea de a distribui acest conţinut către/de la staţiile al căror nume nu este cunoscut este esenţială.

De asemenea, ne putem afla în situaţia în care dorim să transferăm toate fotografiile făcute în ultima noastră vacanţă din telefon în calculator pentru a elibera memoria dispozitivului mobil. În asemenea situaţii este utilă opţiunea de a distribui oricâte fişiere din momentul în care s-a stabilit perechea de agenţi ce se va angaja în efectuarea transferului de date.

În mod firesc, utilizatorii îşi vor dori sa distribuie numeroase tipuri de conţinut precum videoclipuri, fişiere PDF sau fişiere din suita Microsoft Office. Astfel, această extensie se poate dovedi utilă în numeroase ocazii.

Pentru moment, agentul PC nu dispune de o interfaţă grafică foarte prietenoasă cu marea gamă de utilizatori. Când aceştia doresc să transfere conţinut aflat pe un calculator este necesară transmiterea căii acestuia către intrarea standard a agentului, lucru care poate nu este familiar marii majorităţi a utilizatorilor. O variantă mai intuitivă poate consta în adăugarea unei opţiuni în meniul apărut prin accesarea unui fişier folosind click-ul dreapta al mouse-ului sau utilizarea paradigmei "drag and drop".

Nu în ultimul rând, pot fi aduse îmbunătăţiri ale performanţei prin adaptarea agenţilor la performanţele reţelei sau schimbarea algoritmului de comparaţie între imagini.

Bibliografie

[1] Dropbox (service). https://en.wikipedia.org/wiki/Dropbox_(service), Iunie 2017. [Online; accesat 17-Iunie-2017].

[2] File Transfer Protocol. https://en.wikipedia.org/wiki/File_Transfer_Protocol, Mai 2017. [Online; accesat 17-Iunie-2017].

[3] Google Drive. https://en.wikipedia.org/wiki/Google_Drive, Iunie 2017. [Online; accesat 17-Iunie-2017].

[4] OpenCV. https://en.wikipedia.org/wiki/OpenCV, Mai 2017. [Online; accesat 17-Iunie-2017].

[5] Secure copy. https://en.wikipedia.org/wiki/Secure_copy, Aprilie 2017. [Online; accesat 17-Iunie-2017].

[6] USB. https://en.wikipedia.org/wiki/USB, Iunie 2017. [Online; accesat 17-Iunie-2017].

[7] Send Anywhere. How to Transfer Files. https://send-anywhere.com/product/how-to-transfer-files, Iunie 2017. [Online; accesat 17-Iunie-2017].

[8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[9] Clikbrix. 10 mistakes to avoid when using QR codes. https://clikbrix.wordpress.com/tag/qr-code/, Februarie 2012. [Online; accesat 10-Iunie-2017].

[10] Alejandro Escamilla. Laptop on office desk. http://www.freepik.com/free-photo/laptop-on-office-desk_755831.htm, Mai 2013. [Online; accesat 17-Iunie-2017].

[11] Google. Dashboards. https://developer.android.com/about/dashboards/index.html, Iunie 2017. [Online; accesat 10-Iunie-2017].

[12] Google. The Activity Lifecycle. https://developer.android.com/guide/components/activities/activity-lifecycle.html, Martie 2017. [Online; accesat 10-Iunie-2017].

[13] HP. HP Notebook PCs - Fixing Webcam Problems (Windows Vista). https://support.hp.com/rs-en/document/c01560721, Iunie 2017. [Online; accesat 13-Iunie-2017].

[14] A. Huaman. Feature Matching with FLANN. http://docs.opencv.org/3.0-beta/doc/tutorials/features2d/feature_flann_matcher/feature_flann_matcher.html, Noiembrie 2014. [Online; accesat 10-Iunie-2017].

[15] Monect. ABOUT. http://www.monect.com/about/, Iunie 2017. [Online; accesat 17-Iunie-2017].

[16] Sean Owen. https://github.com/zxing/zxing, Iunie 2017. [Online; accesat 17-Iunie-2017].

[17] Jon Postel and Joyce Reynolds. File transfer protocol. 1985.

[18] Brian Randell. Colossus: Godfather of the computer. In *The Origins of Digital Computers*, pages 349–354. Springer, 1982.

[19] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[20] Lars Vogel. Android development tutorial. *Based on Android*, 2013.

[21] Jason Ziller. Thunderbolt^{TM} 3 – The USB-C That Does It All. https://thunderbolttechnology.net/blog/thunderbolt-3-usb-c-does-it-all, Iunie 2015. [Online; accesat 11-Iunie-2017].