**Simulator pentru algoritmi de poziționare interioară prin tehnologia Wi-Fi**

LUCRARE DE LICENŢĂ

|  |  |  |
| --- | --- | --- |
|  | Absolvent: | **Vlad-Nicolae NIȚU** |
|  |  |  |
|  | Coordonator științific: | **Conf. Dr. Ing. Bogdan IANCU** |

**2023**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | |  | |
| DECAN, |  | | DIRECTOR DEPARTAMENT, | |
| **Prof. dr. ing. Liviu MICLEA** |  | **Prof. dr. ing. Rodica POTOLEA** | |

Absolvent: **Vlad-Nicolae NIȚU**

**Simulator de algoritmi de poziționare prin Wi-Fi**

1. **Enunțul temei:** *Acest proiect își propune să realizeze transpunerea unui proces de testare a unor algoritmi de poziționare cu ajutorul mai multor dispozitive capabile de transmitere de semnal Wi-Fi, dintr-un mediu real, într-un mediu simulat.*
2. **Conținutul lucrării:** *Cuprins, Introducere – Contextul proiectului, Obiectivele Proiectului, Studiu Bibliografic, Analiză și Fundamentare Teoretică, Proiectare de Detaliu și Implementare, Testare și Validare, Manual de Instalare și Utilizare, Concluzii, Bibliografie, Anexe*
3. **Locul documentării**: Universitatea Tehnică din Cluj-Napoca, Departamentul Calculatoare
4. **Consultanți**:
5. **Data emiterii temei:** 1 noiembrie 2022
6. **Data predării:** 6 iulie 2023

|  |  |
| --- | --- |
| Absolvent: |  |
|  |  |
| Coordonator științific: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Declarație pe propria răspundere privind**

**autenticitatea lucrării de licență**

Subsemnatul(a) **Nițu Vlad-Nicolae**, legitimat(ă) cu C.I. seria VX nr. 841062  
CNP 5000907385563, autorul lucrării SIMULATOR DE ALGORITMI DE POZIȚIONARE INTERIOARĂ PRIN TEHNOLOGIA WI-FI elaborată în vederea susținerii examenului de finalizare a studiilor de licență la Facultatea de Automatică și Calculatoare, Specializarea CALCULATOARE din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie a anului universitar 2023, declar pe propria răspundere că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

In cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

|  |  |  |
| --- | --- | --- |
| Data  04/07/2023 |  | Nume, Prenume  Nițu Vlad-Nicolae |
|  |  |  |
|  |  | Semnătura |

**Cuprins**

[Capitolul 1. Introducere 1](#_Toc139369739)

[1.1. Contextul proiectului 1](#_Toc139369740)

[1.2. Conținutul lucrării 2](#_Toc139369741)

[Capitolul 2. Obiectivele proiectului 3](#_Toc139369742)

[Capitolul 3. Studiu bibliografic 5](#_Toc139369743)

[3.1. Introducere 5](#_Toc139369744)

[3.2. Tehnologii folosite pentru localizare 5](#_Toc139369745)

[3.2.1. Wi-Fi 5](#_Toc139369746)

[3.2.2. Bluetooth 6](#_Toc139369747)

[3.2.3. ZigBee 7](#_Toc139369748)

[3.2.4. Radio Frequency Identification Device (RFID) 7](#_Toc139369749)

[3.2.5. Ultra Wideband (UWB) 8](#_Toc139369750)

[3.3. Simulatoare de rețea 8](#_Toc139369751)

[3.3.1. Network Simulator 3 (NS-3) 9](#_Toc139369752)

[3.3.2. OMNeT++ 9](#_Toc139369753)

[3.3.3. WLAN Toolbox 10](#_Toc139369754)

[3.4. Simulatoare de poziționare existente 10](#_Toc139369755)

[3.4.1. GuiNuMo 11](#_Toc139369756)

[3.4.2. Simulator for Methods of Indoor Localization (SMILe) 13](#_Toc139369757)

[3.4.3. MatLab Three-Dimensional Indoor Positioning with 802.11az Fingerprinting and Deep Learning 14](#_Toc139369758)

[Capitolul 4. Analiză și fundamentare Teoretică 16](#_Toc139369759)

[4.1. Arhitectura Software 16](#_Toc139369760)

[4.2. Șablonul Model-View-Controller (MVC) 16](#_Toc139369761)

[4.2.1. Model 17](#_Toc139369762)

[4.2.2. View 17](#_Toc139369763)

[4.2.3. Controller 17](#_Toc139369764)

[4.3. Algoritmi de poziționare 18](#_Toc139369765)

[4.3.1. Introducere 18](#_Toc139369766)

[4.3.2. Tehnici de localizare 19](#_Toc139369767)

[4.4. NS3 27](#_Toc139369768)

[4.4.1. Abstractizări cheie 27](#_Toc139369769)

[4.5. JavaFX 29](#_Toc139369770)

[4.6. Wi-Fi (IEEE 802.11) 30](#_Toc139369771)

[Capitolul 5. Proiectare de detaliu și implementare 33](#_Toc139369772)

[5.1. Arhitectura globală a sistemului 33](#_Toc139369773)

[5.2. Diagrama de utilizare 34](#_Toc139369774)

[5.3. Diagrama de secvență 38](#_Toc139369775)

[5.4. JavaFX 39](#_Toc139369776)

[5.4.1. Arhitectura 39](#_Toc139369777)

[5.4.2. Implementare 40](#_Toc139369778)

[5.5. Librăria NS3 48](#_Toc139369779)

[5.5.1. Structură 48](#_Toc139369780)

[Capitolul 6. Testare și validare 54](#_Toc139369781)

[6.1. JavaFX 54](#_Toc139369782)

[6.2. NS3 59](#_Toc139369783)

[Capitolul 7. Manual de instalare si utilizare 60](#_Toc139369784)

[7.1. Manual de instalare 60](#_Toc139369785)

[7.1.1. Descărcare și instalare JDK 60](#_Toc139369786)

[7.1.2. Descărcare CLion 60](#_Toc139369787)

[7.1.3. Descărcare WSL 60](#_Toc139369788)

[7.1.4. Descărcare ns3-3.37 61](#_Toc139369789)

[7.1.5. Descărcare Intellij 61](#_Toc139369790)

[7.1.6. Instalare ns3-3.37 61](#_Toc139369791)

[7.2. Manual de utilizare 63](#_Toc139369792)

[7.2.1. Pornire aplicația JavaFX 63](#_Toc139369793)

[7.2.2. Exemplu rulare 64](#_Toc139369794)

[Capitolul 8. Concluzii 66](#_Toc139369795)

[8.1. Contribuții personale 66](#_Toc139369796)

[8.2. Rezultate obținute 66](#_Toc139369797)

[8.3. Dezvoltări ulterioare 67](#_Toc139369798)

# Introducere

Acest proiect are ca scop principal transpunerea unui test real simulat realizat cu dispozitive de rețea reale într-un mediu simulat, cu ajutorul tehnologiilor existente pentru a reduce astfel costurile de implementare și de testare a mai mulți algoritmi de localizare într-un spațiu interior.

## Contextul proiectului

Serviciile de poziționare, încă de la apariția lor, își mențin rolul de ajutor constant în viața oamenilor pentru navigație și orientare în spațiu. Într-o lume care devine din ce în ce mai complexă și mai aglomerată, odată cu evoluția tehnologiei, precum și creșterea vitezei cu care informația se deplasează pe glob, nevoia de localizare devine din ce în ce mai importantă. Acestea oferă utilizatorilor posibilitatea de a afla informații relevante despre poziția din care le folosesc sau de a descoperi locuri de interes în împrejurimi. De-a lungul timpului, evoluția tehnologică a vizat și îmbunătățiri pe planul serviciilor de poziționare, adăugând complexitate și precizie procesului de localizare a unei entități.

Deși conceptul de poziționare geografică a existat de o bună perioadă de vreme, serviciile de poziționare au început să se dezvolte serios odată cu apariția sistemelor de navigație globale (GNSS) în anii 1970, acestea incluzând celebrele sisteme GPS (Global Positioning System) și GLONASS (Global Navigation Satellite System) care au revoluționat modul în care oamenii se orientează în perioada actuală. Aceste sisteme au fost dezvoltate inițial pentru uz militar și aviatic, dar au devenit treptat comerciale pentru utilizarea pe scară largă.

Primii utilizatori ai sistemului GPS erau de obicei entuziaștii de tehnologie sau profesioniștii care aveau nevoie de un sistem de navigație precisă pentru a-și atinge obiectivele ( de exemplu piloții sau marinarii ). Cu toate acestea, odată cu dezvoltarea acestor inovații, utilizarea sistemului GPS a început să se extindă și în viața cotidiană, datorită apariției dispozitivelor mobile, precum și a saltului imens pe care le-au făcut telefoanele mobile și smartphone-urile în ultimele două decenii.

Odată cu apariția tehnologiei fără fir și implicit a capabilităților dispozitivelor mobile de a se conecta la rețea și a accesa astfel de servicii, poziționarea a devenit un lucru simplu de realizat de oricine, cu sau fără cunoștințe de specialitate. Fără doar și poate, aceste îmbunătățiri au putut doar stârni interesul oamenilor pentru a avansa cu aceste sisteme, urmărind să înregistreze performanța reală a procesului de localizare, pe baza a diferite metrici.

## Conținutul lucrării

Lucrarea este structurată în 8 capitole. Următoarea listă va prezenta un sumar a ce se va regăsi în fiecare dintre capitole.

* **Capitolul 1 – Introducere – Contextul proiectului -**  Se va prezenta sumar scopul proiectului precum și o descriere a contextului proiectului propus.
* C**apitolul 2 –** **Obiectivele Proiectului –** Se vor prezenta obiectivele principale ale lucrării, precum și scheme ce vor facilita înțelegerea temei alese.
* **Capitolul 3 – Studiu Bibliografic -** Se vor prezenta informații relevante din domeniul proiectului obținute prin studiul rețelelor wireless, a algoritmilor de poziționare bazați pe rețelele de calculatoare existente. Oferă o imagine de ansamblu a ce se prezintă actual în domeniul curent.
* **Capitolul 4 – Analiză și Fundamentare Teoretică -**  Se va prezenta arhitectura sistemului lucrării, precum și API-ul folosit pentru realizarea aplicației dorite, utilizate pentru atingerea obiectivului lucrării.
* **Capitolul 5 –** **Proiectare de Detaliu și Implementare –** Se vor prezenta în detaliu diagramele de utilizare și de secvență, implementarea simulatorului propriu-zis, precum și evaluarea preciziei acestuia.
* **Capitolul 6 – Testare și Validare –** Se vor prezenta rezultatele testării simulatorului, precum și informații relevante cu privire la cum se situează în comparație cu sisteme reale de simulare.
* **Capitolul 7 –** **Manual de Instalare și Utilizare -**  Va cuprinde pașii principali de instalare și utilizare a simulatorului dezvoltat.
* **Capitolul 8 – Concluzii -**  Se vor prezenta aportul personal, rezultatele obținute cu aplicația dezvoltată precum și propuneri pentru posibile dezvoltări viitoare ale sistemului.

# Obiectivele proiectului

În acest capitol se prezintă tema propriu zisă a lucrării curente, precum și obiectivele pe care această lucrare își propune să le atingă.

Primul obiectiv al proiectului este să ofere un mediu simulat pentru algoritmii de poziționare existenți în industrie, pentru a putea realiza astfel de simulări fără necesitatea deținerii de instrumente care compun un astfel de test, printre acestea se enumeră multiple dispozitive wireless de tipul ”Access Point” (AP), existența unei rețele comune între aceste dispozitive, precum și un dispozitiv fără fir, mobil, care să comunice cu toate AP-urile.

Un astfel de simulator se adresează persoanelor care vor să opereze rapid și facil un test de localizare, cunoscând aria în care se va desfășura poziționarea, precum și pozițiile relative ale dispozitivelor.

Diagram

Description automatically generated

Figură 2.1 - Introducere date și afișare rezultat

Pozițiile sunt reprezentate de un set de coordonate într-un sistem cartezian, originea este setată într-un colț al încăperii în care se vrea simulat unul sau mai mulți algoritmi de poziționare.

Aplicația se va folosi pentru fiecare lansare în execuție de librăria de simulatorul de evenimente discrete în cadrul unei topologii de rețea Network Simulator 3 (NS3) pentru a trimite datele acumulate în interiorul interfeței grafice sub forma de listă de dispozitive wireless ce constituie aria în care se va desfășura simularea propriu-zisă.

Astfel, pentru a putea atinge acest obiectiv, se va implementa o aplicație ce va deservi introducerii de date relevante pentru un mediu de testare, de exemplu, pozițiile relative ale unor dispozitive wireless ce vor deservi unui test de poziționare. Pozițiile sunt reprezentate de un set de coordonate dintr-un sistem cartezian, originea este setată într-un colț al încăperii în care se vrea simulat unul dintre algoritmi. Pentru a putea rula un test cu aceste date referitoare la hardware, este nevoie de introducerea algoritmului care va servi drept șablon de rulare a testului. Aplicația va afișa grafic un chenar care va conține spațiul efectiv de lucru în care se vor introduce vizual dispozitivele menționate anterior.

Aplicația poate fi considerată în acest sens o abstractizare a simulatorului de rețea NS3, care este oarecum decent de înțeles, dificil de folosit, dar și mai dificil de modificat pentru redarea de simulări diferite de cele puse la dispoziție de către echipa de suport a simulatorului. Interfața grafică deservește acestei abstractizări prin achiziția de date simple de la un utilizator oarecare, fără să cunoască toată complexitatea sistemului în care va simula o topologie.

Al doilea obiectiv al lucrării este acela de a implementa un simulator care să se ridice la performanțele unui test real în ceea ce privește un test de poziționare. Pentru un astfel de simulator, este nevoie de o performanță ridicată a rezultatelor testelor ce vor fi rulate. Pentru a atinge acest obiectiv, proiectul își propune să implementeze identic mediul de simulare aferent algoritmilor prezentați în mod fizic.

A picture containing line, diagram, plot, parallel

Description automatically generated

Figură 2.2 - Exemplu simulare reală

Un test real este afectat de orice tip de „zgomot” ce poate apărea în timpul testelor, precum obiecte străine prin care trece semnalul dispozitivelor către obiectul mobil de poziționat sau interferențe de la alte sisteme din vecinătate. Un sistem simulat ar trebui să obțină rezultate mai bune ca și performanță din pricina mediului de lucru simulat, care este un mediu ideal, fără interferențe sau zgomot exterior testului, însă aceste elemente pot fi introduse pentru a se apropia cât mai mult de un test real.

În acest sens, aplicația dezvoltată, poate fi integrată în orele de laborator ale materiilor care fac parte din domeniul rețelisticii pentru o mai bună înțelegere a cum funcționează algoritmii implementați, precum și cum se raportează aceștia la metodele teoretice existente de măsurare a poziției unui obiect mobil.

# Studiu bibliografic

## Introducere

Convergențele tehnologice susțin apariția de noi aplicații. În perioada actuală, cu răspândirea rețelelor wireless și a serviciilor asociate acestora, apar noi probleme legate de mobilitatea terminalelor aferente acestor servicii. În special, dacă poziționarea este tratată pe larg și destul de precisă, poziționarea în interior, sau, mai general, în medii heterogene, la fel ca și predicția mobilității, necesită dispozitive complexe, cum ar fi senzori de busolă sau senzori de ultrasunete. Când soluția Wi-Fi este supusă comparației cu soluția extinsă de GPS, sau cea de rețea-senzorială, aceasta reiese ca fiind complexă, consumatoare de timp, și prea scumpă pentru a fi implementată eficient precum și configurată într-un mediu interior. O abordare pentru sistemul de poziționare interioară prin Wi-Fi în primul rând constă în localizarea și configurarea dispozitivelor de tip Access Point corespunzător, în al doilea rând în aplicarea tehnicilor dovedite de poziționare a unui terminal mobil, și în ultimul rând în mărirea acurateții aparatului de măsură prin metode inteligente precum verificarea coerenței rezultatelor bazate pe cunoștințe despre topologia folosită și hardware. [1]

## Tehnologii folosite pentru localizare

Numeroase tehnologii existente care au fost folosite pentru localizare interioară vor fi prezentate și discutate în această secțiune. Tehnologii de comunicație radio, cum ar fi, IEEE 802.11, Bluetooth, ZigBee, RFID și Ultra-Wideband (UWB), vor fi prezentate iar mai apoi comparate. Există și tehnologii diferite de contextul actual, cum ar fi tehnologii bazate pe lumina vizibilă sau bazate pe acustică. Există de asemenea și sisteme de localizare bazate pe tehnologii de cameră/viziune. Toate aceste sisteme care nu au legătură cu contextul actual nu vor fi discutate în cele ce urmează. [2]

### Wi-Fi

Standardul IEEE 802.11, uzual cunoscut drept ”Wi-Fi”, operează în banda Industrială, Științifică și Medicală (ISM) și este folosit primar pentru a oferi capabilități de rețea și conexiune la Internet diferitelor dispozitive în medii private, publice cât și comerciale. Inițial, Wi-Fi a avut o rază de recepție de aproximativ 100 de metri care a fost extinsă ulterior la 1 kilometru (km), în 802.11ah (optimizat în principal pentru servicii IoT). [1]

Majoritatea smartphone-urilor curente, a laptopurilor și a altor dispozitive portabile a utilizatorului au Wi-Fi activat, ceea ce face Wi-Fi un candidat ideal pentru localizarea în interior și una dintre cele mai pe larg studiate tehnologii din literatură. Ținând cont că existentele Access Point-uri (APs) Wi-Fi pot fi folosite ca și puncte de referință pentru strângere de date de semnal, sistemele de localizare uzuale (care pot obține precizie de localizare rezonabilă) pot fi construite fără nevoia de infrastructură suplimentară. [1]

Totuși, rețelele Wi-Fi existente sunt lansate normal pentru comunicare (de exemplu, pentru a maximiza cantitatea de date trimisă și acoperirea rețelei) mai degrabă decât pentru intenții de localizare și astfel algoritmi eficienți sunt necesari pentru a îmbunătății precizia lor de localizare. Mai mult de atât, interferența incontrolabilă în banda ISM a demonstrat că poate afecta precizia localizării. Tehnicile uzuale folosite (Received Signal Strength (RSS), Channel State Information (CSI), Time of Flight (ToF), Angle of Arrival (AoA) și orice combinații ale acestora) pot fi folosite pentru a oferi servicii de localizare prin Wi-Fi. Sistemele recente bazate pe localizare Wi-Fi au obținut o precizie a localizării medie de 23 de centimetri (cm). [1]

### Bluetooth

Bluetooth (sau IEEE 802.15.1) constă în specificațiile straturilor de MAC și fizic pentru conectarea diferitelor dispozitive wireless fixe sau mobile dintr-un anumit spațiu personal. Ultima versiune de Bluetooth, adică, Bluetooth Low Energy (BLE), de asemenea cunoscut drept Bluetooth Smart, poate produce o rată de date îmbunătățită de 24Mbps și o rază de acoperire de 70-100 de metri cu o eficiență energetică ridicată, în comparație cu modelele anterioare. [1]

În timp ce BLE poate fi folosit pentru diferite tehnici de localizare cum ar fi RSSI, AoA, și ToF, majoritatea soluțiile de localizare bazate pe BLE se bazează pe inputuri bazate pe RSS, din moment ce sistemele bazate pe RSS sunt mai puțin complexe. Încrederea în inputurile bazate pe RSS limitează acuratețea localizării. Chiar dacă BLE în forma sa originală poate fi folosit pentru localizare (datorită razei sale, a costului redus și a consumului de energie), 2 protocoale bazate pe BLE, adică, *iBeacons* (creat de Apple Inc.) și *Eddystone* (creat de Google Inc.) au fost recent propuse, în principiu pentru serviciile bazate pe proximitate conștientă de context. [1]

Apple a anunțat iBeacons în World Wide Developer Conference (WWDC) în 2013. Protocolul este proiectat specific pentru detecția proximității și servicii bazate pe proximitate. Protocolul permite dispozitivului cu BLE activat (cunoscut drept iBeacon sau simplu beacon) să transmită ”beacons” sau semnale la intervale periodice de timp. Mesajul beacon constă într-un identificator unic universal de 16 bytes (UUID) și de 2 bytes de valori majore și minore. Orice dispozitiv cu BLE activat, care are o aplicație proprietară să asculte aceste beacons, primește mesajele beacon și folosește RSSI pentru a estima proximitatea dintre dispozitivul iBeacon și utilizator. Bazat pe puterea RSSI, utilizator este clasificat *immediate* (<1m), *near* (1-3m), *far* (>3m) și *unknown.* [1]

A diagram of a cell phone

Description automatically generated with low confidence

Figură 3.1 – iBeacons [1]

Schematica unei arhitecturi tipice bazate pe beaconuri este prezentată în figura anterioară. După ce se primește un mesaj de la iBeacon, dispozitivul utilizator consultă un server sau cloud-ul pentru a identifica acțiunea afiliată cu beacon-ul primit. Acțiunea poate fi să trimită un cupon de reducere să fie primit de către utilizator, să deschidă o ușă sau să afișeze un conținut interactiv pe un monitor (sau un actuator) bazat pe proximitatea utilizatorului relativă la un beacon sau la altă entitate.

O constrângere fundamentală a iBeacons (impusă de către Apple) este că doar valoarea medie a RSSI este raportată la dispozitivul utilizator la fiecare secundă, chiar dacă beacon-urile sunt transmise la intervale de 50ms. Aceasta este impusă pentru a compensa variațiile în valorile instantanee la dispozitivul utilizator. Totuși, această medie RSS și raportarea întârzierii poate impune provocări semnificative localizării în timp real. Din moment ce motivul din spatele iBeacons a fost să ofere detecția proximității, a fost de asemenea folosit pentru localizare interioară. [1]

### ZigBee

ZigBee este construit pe standardul IEEE 802.15.4 care este preocupat cu straturile MAC și fizic pentru cost redus, rată mică de date și eficiență energetică a rețelelor personale. ZigBee definește nivelele superioare ale stivei protocolului și este folosit de bază în rețelele de senzori wireless (WSN). Stratul de rețea în ZigBee este responsabil pentru rutarea multihop și organizarea rețelei în vreme ce stratul aplicației este responsabil de comunicație distribuită și dezvoltare. În vreme ce ZigBee este favorabil localizării de senzori în WSN, nu este disponibil încă pentru majoritatea dispozitivelor utilizator, acest lucru arătând că nu este ideal localizării interioare. [1]

### Radio Frequency Identification Device (RFID)

RFID este în principal folosit pentru transferul și stocarea datelor folosind transmisii electromagnetice de la un transmițător la orice circuit compatibil cu frecvențe radio (RF). Un sistem RFID constă într-un cititor care comunică cu taguri RFID. Tagurile RFID emit date pe care cititorul RFID le poate citi folosind un RF predefinit și un protocol, cunoscut și de cititor și de taguri *apriori.* Sunt două tipuri de sisteme de bază RFID: [1]

* *Active RFID:* Active RFID operează pe razele de frecvență Ultra High Frequency (UHF) și pe microunde. Ele sunt conectate la o sursă locală de energie, transmițând periodic ID-ul lor și pot opera pe sute de metri de la cititorul RFID. Aceste sisteme pot fi folosite pentru localizare și urmărirea obiectelor din moment ce au o rază rezonabilă, cost scăzut și pot fi ușor imbricate într-un obiectele urmărite. Totuși, tehnologia Active RFID nu poate obține acuratețe sub un metru și nu este disponibilă pe majoritatea dispozitivelor utilizator portabile. [1]
* *Passive RFID:* Passive RFID sunt limitate în raza de comunicare (1-2m) și pot opera fără baterie. Ele sunt mai mici, mai ușoare și costă mai puțin decât cele active; ele pot opera în raze de frecvență joase, înalte sau UHF precum și microunde. Chiar dacă ele sunt folosite ca o alternativă la codurile de bare, în special când tagul nu este în LoS cu cititorul, raza lor limitată le face nepotrivite pentru localizare în interior. Ele pot fi folosite pentru serviciile bazate pe proximitate folosind soluții *brute-force*, dar acestea tot vor necesita modificări la procedurile existente folosite de Passive RFID cum ar fi transmiterea ID-ului care poate fi folosit să identifice RFID-ul. [1]

### Ultra Wideband (UWB)

În UWB, pulsurile ultra scurte cu o perioadă de timp de sub 1 nanosecundă (ns) sunt transmise peste o lățime de bandă mare (>500MHz), într-o rază de frecvență de la 3.1 la 10.6GHz, folosind un low duty cycle care rezultă un consum de energie redus. Tehnologia a fost în principal folosită pentru comunicație între sisteme pe rază scurtă, cum ar fi perifericele PC-ului, și alte aplicații interioare. UWB a fost în particular o tehnologie atractivă pentru poziționare în interior pentru că este imună la interferențele de la alte semnale (datorită tipului de semnal drastic diferit și de spectrul radio), în timp ce semnalul UWB (în special frecvențele joase incluse în raza largă a spectrului UWB) pot penetra o varietate de materiale, incluzând pereți (chiar dacă metalele și lichidele pot interfera cu semnalul UWB). Mai mult de atât, durata foarte scurtă a pulsurilor UWB le fac mai puțin sensibile la efectele multipath, permițând identificării căii principale în prezența semnalelor multipath și procurând estimări precise de ToF, care au fost recunoscute că oferă o acuratețe de 10cm la localizare. [1]

Totuși, progresul încet în dezvoltarea standardului UWB (chiar dacă UWB a fost propus inițial pentru a fi folosit în rețelele personale (PAN)), a limitat folosirea acestui standard în produsele consumator și în dispozitivele utilizator portabile. [1]

Tabel 1 - Tehnologii de poziționare [1]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tehnologie | Raza maximă | Debitul maxim | Energia consumată | Avantaje | Dezavantaje |
| IEEE 802.11 n  802.11 ac  802.11 ad | 250m exterior  35m interior  Câțiva metri | 600Mbps  1.3Gbps  4.6Mbps | Moderată  Moderată  Moderată | Disponibilitate ridicată, acuratețe mare, nu necesită hardware complex | Predispus la zgomote, are nevoie de algoritmi complexi |
| UWB | 10-20m | 460Mbps | Moderată | Imun la interferențe, oferă acuratețe ridicată | Rază mai scurtă, cost ridicat |
| RFID | 200m | 1.67Gbps | Scăzută | Consumă puțină energie, are rază largă | Acuratețe scăzută |
| Bluetooth | 100m | 24Mbps | Scăzută | Debit mare, consum mic de energie | Acuratețe scăzută, predispus la zgomot |

## Simulatoare de rețea

Simularea de rețea este fără niciun dubiu una dintre metodologiile de evaluare predominante în aria rețelelor de calculatoare. Este vast utilizată pentru dezvoltarea de noi arhitecturi de comunicare și protocoale de rețea. Așa numitele *simulatoare de rețea* permit modelarea unei rețele arbitrare de calculatoare prin specificarea comportamentului nodurilor de rețea și a canalelor de comunicație. De exemplu, pentru a investiga caracteristicile unui protocol nou de rutare, este de obicei implementat un simulator de rețea. După aceea, comportamentul de rutare poate fi ușor studiat în topologii diferite, având în vedere că topologia rețelei este nu mai mult decât un set de parametrii de simulare. Seturile de simulare a rețelei cele mai disponibile sunt bazate pe paradigme ale simulării bazate pe evenimente discrete. (DES). [3]

### Network Simulator 3 (NS-3)

NS-3 a folosit dezvoltat pentru a oferi o platformă de simulare a unei rețele extensibile și deschise pentru cercetare a rețelisticii și pentru educație. În mare, ns-3 oferă modele care explică cum pachetele de date din rețea funcționează și cum operează, și oferă un motor de simulare pentru utilizatori pentru a întreprinde experimente. Câteva motive pentru a folosi ns-3 includ efectuarea studiilor care sunt prea dificile sau imposibile de realizat în sistemele reale, pentru a studia comportamentul sistemului într-un mediu reproductibil și controlat și pentru a învăța cum funcționează rețeaua. [4]

Ca și predecesorul său, ns-3 se bazează pe C++ pentru a-și implementa modelele de simulare. Totuși, ns-3 nu mai folosește scripturi oTcl pentru a controla simularea, astfel abandonând problemele care au fost introduse din combinația C++ și oTcl în ns-2. În loc de acesta, simulările de rețea în ns-3 sunt implementate prin C++ pur, în timp ce unele părți din simulare pot fi realizate opțional folosind Python. Mai mult de atât, ns-3 integrează concepte arhitecturale și cod din GTNetS, un simulator cu caracteristici de scalabilitate bune. [3]

Multe unelte de simulare există pentru studiile de simulare a rețelei. Vor fi prezentate câteva caracteristici distincte ale ns-3 în contrast cu alte unelte: [4]

* Ns-3 este proiectat ca un set de librării care pot fi combinate împreună și de asemenea cu alte librării software externe. În timp ce alte platforme de simulare oferă utilizatorului un mediu grafic integrat de interfață utilizator în care toate taskurile sunt procesate, ns-3 este mai modular în această speță. Multipli animatori externi și unelte de vizualizare și analiză de date pot fi folosiți cu ns-3. Totuși, utilizatorii ar trebui să se aștepte să lucreze cu linia de comandă și cu uneltele de dezvoltare software C++ și/sau Python. [4]
* Ns-3 este în principal folosit pe Linux sau sisteme macOS, chiar dacă există suport și pentru sistemele BSD sau chiar pentru framework-uri Windows care pot fi construite pe cod Linux, cum ar fi Windows Subsytstem for Linux, sau Cygwin. Visual Studio nativ de pe Windows nu este prezentat ca și suportat chiar dacă un dezvoltator lucrează pentru viitor suport în acest sens. Utilizatorii Windows pot de asemenea să folosească o mașină virtuală. [4]

### OMNeT++

OMNeT++ este un framework de simulare a rețelei bazat pe procesarea de evenimente discrete modular într-o manieră orientată spre obiect. Are o arhitectură generică, deci poate fi folosit (și a fost folosit) în varii probleme din acest domeniu: [5]

* Modelarea rețelelor de comunicații cu fir și fără fir
* Modelarea protocoalelor
* Modelarea rețelelor în cozi
* Modelarea de multiprocesoare și a sistemelor hardware distribuite
* Validarea arhitecturilor hardware
* Evaluarea aspectelor de performanță a sistemelor software complexe
* În general, modelarea și simularea oricărui sistem unde soluțiile bazate pe evenimente discrete sunt potrivite, și pot fi mapate pe entități care comunică prin schimbul de mesaje

OMNeT++ nu este un simulator de lucruri concrete, ci mai degrabă oferă infrastructură și unelte pentru a crea astfel de simulări. Unul dintre ingredientele fundamentale a acestei infrastructuri este componenta arhitecturală pentru modelele de simulare. Modelele sunt construite prin reutilizarea componentelor numite *module.* Modulele bine scrise sunt cu adevărat reutilizabile, și pot fi combinate în varii moduri ca blocurile de LEGO. [5]

Simulările OMNeT++ pot fi executate sub numeroase interfețe utilizator. Grafic, interfețele utilizator animate sunt nespus de folositoare pentru a demonstra și a ajuta la procesul de debugging, și interfețele utilizator bazate pe linia de comandă sunt cele mai bune pentru execuția în lot. [5]

Simulatorul precum și interfețele utilizator și uneltele sunt ultra portabile. Ele sunt testate pe majoritatea sistemelor de operare comune (Linux, Mac OS/X, Windows) și pot fi compilate ”out-of-the-box” sau după modificări triviale pe majoritatea sistemelor de operare bazate pe Unix. [5]

Ca și ns-3, OMNeT++ se bazează pe C++ pentru implementarea modulelor simple. Totuși, compoziția acestor module simple în module compuse și astfel, în setarea simulării rețelei se întâmplă în NED, limbajul descriptiv de rețea al OMNeT++. NED este randat transparent în cod C++ când simularea este compilată ca și întreg. Mai mult de atât, NED suportă specificarea a parametrilor variabili în descrierea rețelei: De exemplu, numărul de noduri într-o rețea poate fi marcat ca și dinamic și mai încolo să fie configurat la în timpul lansării în execuție. În acest caz, modulele reprezentând nodurile sunt instanțiate dinamic de către simulator în timpul execuției. Această caracteristică este o consecință directă a proiectării stricte a simulatorului într-o manieră orientată pe obiect. [3]

### WLAN Toolbox

WLAN Toolbox oferă funcții de proiectare conforme standardelor, simulare, analiză, și testare a sistemelor de comunicație LAN fără fir. Include forme de undă configurabile la nivelul fizic pentru familia de standarde IEEE 802.11. De asemenea oferă operații de transmitere, modelare de canal, și de recepție, incluzând codarea canalului, modulare, mapare a unui stream spațial, și receptoare MIMO. [6]

Toolboxul oferă referințe de proiectare care ajută la realizarea de simulări la nivel de legături și simulări multi-node la nivel de sistem. Poți genera forme de undă și customiza ateliere de testare, ori programatic ori interactiv, folosind aplicația Wireless Waveform Generator. Poți genera sau parsa cadre MAC uzuale. Poți realiza măsurători de semnal cum ar fi puterea canalului, masca spectrului, și lățimea de bandă ocupată, și să creeze ateliere de testare pentru simulări end-to-end a legăturilor de comunicație WLAN. [6]

## Simulatoare de poziționare existente

În acest capitol vor fi prezentate simulatoare de poziționare deja existente pe piață, precum și caracteristici cheie ce le definesc și protocoalele/algoritmii pe care îi folosesc. Mai simplu, modul de funcționare pentru a reda poziția. De asemenea, pentru fiecare simulator prezentat se va compara succint cu aplicația propusă în documentul curent.

### GuiNuMo

Proiectul *GuiNuMo* (prescurtat din franceză de la *Guide Numérique Mobile,* care înseamnă ghid mobil digital) este bazat pe poziționarea Wi-Fi a dispozitivelor pentru a decide ce conținut va fi transmis către clientul mobil. Proiectul acesta se focusează pe muzeele interioare; astfel, trebuie să localizeze terminalele interioare. Se folosește tehnologia Wi-Fi pentru accesibilitățile acesteia precum și pentru costul redus. Mai mult decât atât, poate de asemenea transporta conținut multimedia către dispozitivele mobile. [2]

Subiectul articolului este despre serviciul de poziționare interioară bazat pe dispozitivele Wi-Fi. Wi-Fi este folosit din pricina costului scăzut pentru investiție de a seta un sistem bazat pe WLAN. Mai mult de atât, Wi-Fi este utilizat pentru localizarea terminalelor mobile și pentru a transmite date între ele și servere media bogate în informații despre conținut. Obiectivul acestei aplicații este de a propune un sistem de poziționare interioară, a cărui acuratețe este suficientă pentru a localiza pe cineva într-o camera și rata de poziționare este îndeajuns de rapidă pentru a se conforma constrângerilor de timp real, cum ar fi predicția mișcării și managementul predictiv al transferului. Astfel, serviciul menționat este bazat pe unirea a două modele: unul este cel colector de date iar al doilea este un model de propagare a undelor radio. Experimente reale au fost rulate pentru a valida propunerea acestui serviciu și a îl compara cu alte soluții asemănătoare. Figura următoare ilustrează sistemul complet de management al mobilității, lansat într-un muzeu. [2]

A picture containing text, diagram, sketch

Description automatically generated

Figură 3.2 - Reprezentare GuiNuMo [2]

Sistemul este compus din: [2]

* Un server care hostează componentele software pentru a gestiona mobilitatea terminalului: poziționarea, învățarea, predicția, și transmisia. Conține secvențe multimedia care potrivesc itemii muzeului. Conține date despre topologia muzeului
* O rețea cablată între server și AP-uri și o rețea wireless între AP-uri și clienții mobili. [2]
* Access Points (APs) permit părților cablate ale rețelei să comunice cu dispozitivul mobil. AP-ul poate să strângă adițional date despre clienții mobili, cum ar fi puterea semnalului sau calitatea acestuia. [2]

Alegând Wi-Fi ca și tehnologie de dezvoltare a acestui sistem, într-adevăr, este mai ieftin și disponibil în orice magazin, mai mult de atât, cu capabilitățile lățimii de bandă, este în stare să transmită date mari cum ar fi secvențe video, texte, și sunete în plus față de localizarea terminalelor mobile în vecinătatea lor. [2]

GuiNuMo își propune să ofere utilizatorilor conținut media în funcție de poziția utilizatorului în interiorul muzeului. GuiNuMo este o aplicație client-server scrisă în limbajele C și C++. [2]

Sunt două versiuni principale de GuiNuMo. O versiune, procesul de măsurare este localizat pe clientul mobil. În cealaltă versiune, procesul de măsurare este localizat pe AP. [2]

Sistemul principal al GuiNuMo este localizat pe serverul de poziționare. Folosește măsurători Signal Strength (SS) pentru a localiza terminalul mobil. Serverul este scris în C++ și datele punctului de referință sunt stocate într-o bază de date PostgreSQL. Când serverul primește măsurători SS, poziționarea este obținută prin FRBHM ( FBCM (Friis-based Calibration Model) and RADAR (sistem de referință de poziționare folosind Wi-Fi) based hybrid model ). Mai întâi, serverul creează un nou fir de execuție în care măsurătorile SS sunt sparte în cupluri de tipul (AP*i*, SS*i*), unde AP*i* este al i-lea AP și SS*i* este SS-ul măsurat pentru acest AP. Odată sparte, măsurătorile sunt folosite pentru a calcula distanța euclidiană (în spațiul SS) între măsurătorile primite și cele stocate în baza de date. Rezultatul este ordonat în funcție de distanță, și primele *K* puncte de referință sunt returnate. [2]

A diagram of a computer network

Description automatically generated with low confidence

Figură 3.3 - Pași de poziționare GuiNuMo versiunile 1 și 2 [2]

Sistemul GuiNuMo este un sistem de poziționare interioară limitat în ceea ce privește obiectivele proiectului curent. GuiNuMo se folosește doar de capabilitățile puterii semnalului (SS) unui AP pentru a determina poziția relativă a unui terminal mobil raportat la toate dispozitivele din spațiul cartografiat. Astfel, sistemul prezentat se diferențiază major prin limitarea numărului de algoritmi folosiți pentru poziționare, are un caz de utilizare foarte interesant însă îngust, particular, în același timp (distribuirea de conținut multimedia către un telefon/laptop) în comparație cu alte sisteme de poziționare deja existente precum și cu proiectul ce se vrea dezvolta. [2]

### Simulator for Methods of Indoor Localization (SMILe)

În ziua de azi, multe echipe de cercetare din toată lumea lucrează la metode de localizare dedicate pentru mediile interioare. Aceste echipe evaluează de obicei ideile ce conduc la experimente în premise locale. Mai mult de atât, ei au discreție deplină în ceea ce privește echipamentele și tehnologiile utilizate în experimente, lansarea rețelei și configurația. O asemenea soluție este suficientă pentru a evalua metode individuale, însă poate obține rezultate incomparabile. În consecință, pentru a se compara metode de localizare dezvoltate cu alte soluții, toate echipele de cercetare trebuie să implementeze și să evalueze fiecare soluție cu cea dezvoltată de ei. Acest fapt este foarte impractic și consumator de timp. [7]

Metodele de localizare care se folosesc de conceptul de timp de cadru radio (timestamps) ca TDoA sau ToF de obicei necesită noduri wireless pentru a urmări secvența predefinită de comunicație. Astfel de secvențe pot utiliza un simplu unicast de un singur pas sau transmisii de cadru broadcast. Totuși, ele pot constitui mai multe forme de multicast sofisticat care includ multe noduri wireless. Pentru a avea o imagine clară asupra imaginii de ansamblu despre cum metodele de localizare particulare se comportă, cercetătorii ar trebui să execute experimente extensive pentru a acoperi multiple scenarii (lansarea nodurilor diferite, localizarea staționară și mobilă a nodurilor, cazuri de utilizare la scară largă, etc.). În vreme ce o astfel de evaluare este dezirabilă, de obicei trece peste capabilitățile echipelor. [7]

SMILe are ca scop doborârea acestor probleme și propune o soluție de simulare completă și unificată ajutând dezvoltarea și evaluarea metodelor bazate pe TDoA și ToF. Scopul este de a oferi o unealtă de simulare de o configurare ridicată și bine definită, unde varii metode de localizare pot fi evaluate în manieră reproductibilă. SMILe ajută utilizatorul să își seteze un mediu interior artificial, unde varii factori semnificativi care afectează performanța globală a localizării pot fi configurați. Acești factori includ: diferite noduri la lansare, capabilitățile radio, inacuratețile ceasurilor hardware, și multe altele. Este posibil să se definească o varietate de cazuri de testate care pot include un număr variabil de noduri staționare sau mobile într-o singură încăpere. Configurația cazurilor de test și a mediului pot fi împărțite între diferitele metode de localizare, astfel comparându-le, metodele devin ușor de înțeles și plauzibile. [7]

SMILe este împărțit logic în două entități: simulator și partea analitică. Prima parte este despre operațiunea de simulare a nodurilor wireless în mediu artificial, felul cum se mișcă și cum comunică. Simulatorul îl lasă pe utilizator să ajusteze diferite aspecte ale simulării (acuratețea ceasului hardware, modele de propagare a undelor radio, mobilitatea nodurilor, etc.). Simulatorul stochează informații despre configurația nodurilor și despre toate cadrele radio care au fost trimise sau primite de către oricare noduri wireless. Pentru fiecare nod, simulatorul stochează cel puțin poziția inițială, adresa MAC și rolul (exemplu: ancoră, releu, mobil). Informația despre cadrul radio include câmpuri precum: destinația și sursa adreselor MAC, direcția operației (transmisie, recepție) și acuratețea timestampurilor operației. [7]

A doua parte, cea analitică, oferă fundații ce vor asista utilizatorul în implementarea și evaluarea propriu-zisă a metodelor de localizare. Adițional, oferă suport pentru vizualizarea și compararea rezultatelor analizei. Datorită abstractizării părții analitice a SMILe, utilizatorul își poate implementa metodele matematice de rezolvare singuri peste ce este deja implementat. [7]

Partea de simulare este bazată pe OMNeT++ și INET. OMNeT++ este state-of-the-art și unul dintre cele mai populare simulatoare de rețea folosite în proiectele business și de învățământ. Este un framework extensibil bazat pe componente cu motorul de simulare scris în C++. Metodele SMILe de localizare se bazează pe următorii algoritmi: [7]

* TDoA fără sincronizare (SF-TDoA) – o variantă a clasicului TDoA cu necesități lejere.
* Whistle – tot o metodă bazată pe TDoA care nu necesită sincronizarea ceasurilor.
* Single-Sided Two-Way-Ranging (SS-TWR) – este o implementare simplă a metodei ToF.
* TDoA asincron (ATDoA) – o metodă care similar cu celelalte, nu necesită sincronizarea ceasurilor.

SMILe se apropie de obiectivele proiectului propus în linii mari, însă este de asemenea limitat de către numărul mic de algoritmi principali pe care îi folosește (TDoA și ToF). [7]

### MatLab Three-Dimensional Indoor Positioning with 802.11az Fingerprinting and Deep Learning

Acest sistem va antrena o rețea neuronală de convoluție (CNN) pentru poziționare de precizie înaltă prin folosirea datelor generate prin tehnologia IEEE 802.11az. Folosind CNN-ul antrenat, se poate prezice poziția precisă a multiple stații (STAs) sau camera în care aceste STAs sunt localizate, bazându-ne pe fingerprinting. Sistemul obține setul de date folosit pentru antrenare și validare a CNN-ului prin două etape. Informații despre canal sunt obținute prin urmărirea razelor pentru un scenariu 3D de spațiu interior apoi conexiunile 802.11az sunt simulate pentru a genera informații de stare ale canalului (CSI) reale. [6]

Standardul Wi-Fi 802.11az, sau cum este uzual cunoscut drept viitoare generație de poziționare (NGP), oferă caracteristici ale stratului fizic care permite o gamă îmbunătățită și poziționare folosind tehnici clasice. [6]

Tehnicile clasice se bazează pe condițiile line-of-sight (LoS) pentru a extrage eficient informații temporale, cum ar fi ToA, sau informații spațiale, cum ar fi AoA, de la un semnal multipath pentru a calcula distanța sau bătaia dintre dispozitive. Când bătaia dintre minim 3 dispozitive poate fi măsurată, trilaterarea poate fi folosită pentru a calcula poziția estimată. [6]

Tehnicile de fingerprinting și deep learning pot fi folosite în sistemele de poziționare Wi-Fi pentru a obține acurateți sub un metru chiar și în medii multipath fără LoS. Un fingerprint conține de obicei informații ale stării canalului, cum ar fi RSSI sau un estimat al canalului al semnalului primit, măsurat la o anumită poziție din mediu. [6]

În timpul fazei de antrenare a rețelei, sistemul creează o bază de date prin prelevarea de probe de fingerprints ale canalului la multiple locații cunoscute din mediu. Rețeaua estimează poziția utilizatorului bazată pe un semnal primit la o locație necunoscută folosindu-se de baza de date ca și referință. [6]

A diagram of a room with tables and chairs

Description automatically generated with low confidence

Figură 3.4 - Exemplu script MatLab [6]

# Analiză și fundamentare Teoretică

## Arhitectura Software

Este deja prea comun pentru dezvoltatori să se apuce să codeze o aplicație fără o arhitectură formală în minte. Fără o arhitectură clară și bine definită, majoritatea dezvoltatorilor și arhitecții vor recurge la șabloanele arhitecturale pe nivele standard de facto (de asemenea denumite și *n-tier arhitecture*), creând straturi implicite prin separarea modulelor codului în pachete. Din păcate, ce de obicei rezultă din această practică este o colecție de module de cod neorganizate, cărora le lipsește un rol clar, responsabilități, și relațiile între fiecare. Acesta este de asemenea cunoscut drept anti-șablonul arhitectural *marea bilă de noroi*. [8]

Aplicațiile cărora le lipsește o arhitectură formală sunt de obicei cuplate strâns, fragile, dificil de schimbat, și fără o viziune clară sau direcție. Ca și rezultat, este foarte dificil să se determine caracteristicile arhitecturale ale aplicației fără a înțelege pe deplin cum funcționează fiecare componentă și modul din sistem. Întrebări elementare despre lansare și mentenanță sunt greu de răspuns: Arhitectura poate fi scalată? Care sunt caracteristicile de performanță ale aplicației? Cât de ușor răspunde aplicația la schimbare? Cât de receptivă este arhitectura? [8]

Șabloanele arhitecturale ajută la definirea caracteristicilor elementare și a comportamentului unei aplicații. De exemplu, unele șabloane arhitecturale se împrumută pe însele natural cu scopul de a realiza aplicații cu un nivel de scalabilitate ridicat, în timp ce alte șabloane arhitecturale se împrumută spre realizarea de aplicații cu un nivel de agilitate ridicat. Cunoscând caracteristicile, punctele forte, punctele slabe a fiecărui șablon arhitectural este necesar pentru a alege unul care să atingă nevoile și scopurile de business în particular. [8]

Ca și arhitect, trebuie să justifici întotdeauna deciziile arhitecturale, în particular atunci când vine vorba de alegerea unei strategii sau a unui șablon anume. [8]

## Șablonul Model-View-Controller (MVC)

Model-View-Controller (MVC) este un șablon architectural care separă aplicația în 3 componente logice principale: **modelul**, **view-ul,** și **controller-ul.** Fiecare dintre aceste componente sunt construite să se ocupe de aspecte de dezvoltare specifice ale unei aplicații. MVC este unul dintre cele mai frecvent utilizate framework-uri de dezvoltare web din standardele industriei pentru a crea proiecte extensibile și scalabile. [9]

Componentele MVC sunt următoarele:

A diagram of a model

Description automatically generated with medium confidence

Figură 4.1 - Componentele MVC [9]

### Model

Componenta Model corespunde întregii logici din spatele datelor cu care utilizatorul lucrează. Aceasta poate reprezenta ori datele care sunt tranferate între componentele View și Controller sau oricare date referitoare la business logic-ul aplicației. De exemplu, un obiect de tipul Customer va obține informațiile despre un client din baza de date, o manipulează și o updatează înapoi în baza de date sau o folosește pentru a randa date. [9]

### View

Componenta View este folosită pentru tot ce înseamnă interfața utilizator (UI) pentru aplicație. De exemplu, view-ul Customer va include componente UI cum ar fi căsuțe de text, meniuri de tip dropdown, etc. cu care utilizatorul final interacționează. [9]

### Controller

Controller-ele acționează drept interfață dintre componentele Model și View pentru a procesa toată logica de business și request-urile care vin, pentru a manipula date folosind componenta Model și pentru a interacționa cu clasele de View pentru a randa output-ul final. [9]

În domeniul Web, o aplicație MVC funcționează după următoarea diagramă:

A picture containing text, screenshot, font, businesscard

Description automatically generated

Figură 4.2 - Flow-ul execuției MVC [9]

## Algoritmi de poziționare

### Introducere

Localizarea în spații interioare a asistat la o creștere a interesului populației pentru acest serviciu, datorită unui larg spectru de servicii potențiale ce pot fi oferite cu ajutorul Internet of Things (**IoT**) și a conectivității omniprezente. Diferite tehnici, tehnologii wireless și mecanisme au fost propuse în literatură pentru a oferi servicii de localizare în spații interioare pentru a îmbunătății serviciile furnizate utilizatorilor. Cu toate acestea, există o lipsă a unui formular de actualitate care să incorporeze câteva dintre recentele sisteme precise și sigure de poziționare. Această lucrare va oferi detalii despre diferite tehnici de poziționare în încăperi, precum ”angle of arrival” (AoA), ”time of flight” (ToF), ”return time of flight” (RTOF), și received signal strength (RSS). În industrie există multe tehnologii care redau un astfel de serviciu, precum Wi-Fi, dispozitive de identificare a frecvenței radio (RFID), ultra wideband (UWB), Bluetooth. Proiectul actual se va focusa pe tehnologia Wi-Fi pentru serviciul de localizare interioară. [2]

Având în vedere poziționarea exterioară care este vast tratată și realizată de sistemele GPS, poziționarea interioară este la momentul actual sub dezvoltare. Pe de-o parte, se prezintă modelele majore ce țin de localizarea interioară și tehnicile actuale, și, pe de altă parte, sistemul de poziționare propriu-zis bazat pe tehnologia Wi-Fi. [2]

Este mai puțin de un deceniu de când proliferarea de masă a telefoanelor inteligente și a dispozitivelor de tip ”wearables” cu capabilități de conexiune wireless au făcut localizarea și urmărirea acestor device-uri sinonimă cu localizarea și urmărirea utilizatorilor care le folosesc și a inițiat o gamă largă de aplicații și servicii asemănătoare. Localizarea utilizatorului sau a device-ului are aplicații numeroase în domenii precum sănătate, industrie, managementul dezastrelor, managementul clădirilor, supraveghere și multe altele. Pot profita de pe urma acestor servicii și sisteme populare cum ar fi IoT, arhitecturile smart ( cum ar fi orașele smart, clădirile smart, rețele de energie smart șamd..). [1]

### Tehnici de localizare

În această secțiune se vor prezenta câteva din tehnicile existente de poziționare actual folosite în industrie.

Acestea sunt următoarele: [1]

1. *Received Signal Strength Indicator (RSSI)*

Soluția referitoare la puterea semnalului primit este cea mai simplă și cea mai larg utilizată rezolvare a localizării interioare. RSS-ul semnifică puterea semnalului actuală primită de către receptor, fiind măsurată de obicei în decibeli-miliwați (dBm) sau miliwați (mW). RSS poate fi utilizat pentru estimarea distanței dintre un transmițător (Tx) și un receptor (Rx); Cu cât mai mare valoarea RSS cu atât mai mică distanța dintre Tx si Rx. Distanța absolută poate fi estimată folosind un număr oarecare de modele de propagare a semnalului diferite având în vedere faptul că puterea de transmisie sau puterea la un punct de referință este cunoscută. RSSI (care este deseori confundat cu RSS) este de fapt indicatorul RSS, o măsurătoare relativă care are unități arbitrare și este majoritar proiectată de fiecare comerciant în parte. De exemplu, chipsetul Wi-Fi Atheros folosește valori RSSI între 0 și 60, în timp ce Cisco folosește o scală de la 0 la 100. Folosind RSSI și un model simplu de propagare a pierderilor, distanța „d” dintre Tx și Rx poate fi estimată la valoarea: RSSI = -10*n*lg(d)+A, unde *n* este exponentul pierderilor de pe traiectorie ( care variază între 2 în spații exterioare și 4 în spații interioare) și A care este valoarea RSSI la o distanță de referință de la receptor. [1]

A picture containing sketch, drawing, circle, diagram

Description automatically generated

Figură 4.3 - Exemplu RSSI [1]

1. *Channel State Information (CSI)*

În multe sisteme wireless, cum ar fi IEEE 802.11 și UWB, coerența lățimii de bandă a unui canal wireless este mai mică decât lățimea de bandă a unui semnal care face frecvența canalului selectivă (de exemplu frecvențe diferite exercită amplitudini și comportamente de fază diferite). Mai mult de atât, în multiple antene de transceiver, răspunsurile frecvenței canalului pentru fiecare pereche de antene poate varia semnificativ ( depinzând de distanța antenei și semnalul lungimii de undă). Atâta vreme cât RSS a fost larg folosit datorită simplității și necesității puținelor componente hardware, oferă numai o estimare a mediei amplitudinilor peste întreaga lățime de bandă a semnalului și a semnalului acumulat peste toate antenele. Acestea fac RSS susceptibil la efecte de ”multipath” și la interferențe și cauzează variațiuni mari în timp. [1]

Pe de altă parte, Channel Impulse Response (CIR) sau perechea sa Fourier, Channel Frequency Response (CFR), care este de obicei livrată către straturile superioare ca și informație despre starea canalului (CSI), are granularitate mai ridicată decât a RSS deoarece poate captura și amplitudinea și răspunsurile de fază a canalului în frecvențe diferite și între perechi de antene separate de transmisie-recepție. În general, CSI-ul este o cantitate complexă și poate fi scrisă într-o formă polară de forma: *H*(f) = |*H*(f)|ej<*H*(f), unde |*H*(f*i*)| este răspunsul amplitudinii (sau magnitudinea) și <*H*(fi) este răspunsul fazei frecvenței f*i* a canalului. În ziua de azi, multe tehnici ale IEEE 802.11 pot oferi măsurători de canal sub nivelul purtătoarelor pentru sisteme Orthogonal Frequency Division Multiplexing (OFDM) care pot fi traduse în informații multipath mai bogate, măsurători mai stabile și localizare mai precisă. [1]

1. *Fingerprinting / Analiză de scenă*

Tehnicile de localizare bazate pe analiza de scenă necesită de obicei un formular despre mediul în care vor fi rulate pentru a obține ”fingerprints” sau caracteristici ale mediului unde localizarea va fi folosită. Inițial, diferite măsurători de RSSI sunt colectate în timpul unor sesiuni offline de teste. Odată ce sistemul este lansat, măsurătorile online ( care sunt obținute în timp real ) sunt comparate cu cele offline pentru a estima poziția utilizatorului. De obicei aceste caracteristici sunt colectate sub formă de RSSI sau CSI. Există o sumedenie de algoritmi disponibili care pot fi folosiți pentru a potrivi măsurători offline cu măsurători online, unele dintre ele fiind: [1]

1. *Metode probabilistice*

Metodele probabilistice se bazează pe verosimilitatea unui utilizator aflat în poziția ”x” având în vedere că valorile RSSI obținute în măsurători online converg spre ”y”. Să presupunem că setul locațiilor candidate L este L={*L1*,*L2*,…,*Lm*}. Pentru orice vector de valoare *O* a unei măsurători online RSSI, locația utilizatorului/dispozitivului va fi *Lj* dacă: [1]

[1]

Această ecuație arată că un utilizator va fi clasificat în locația *Lj* dacă verosimilitatea lui este mai mare decât a oricărei alte locații. Dacă *P*(*Lj*) = *P*(*Lk*) pentru *j,k* = 1,2,3,…,*m*, atunci folosind teorema lui Bayes, putem obține probabilitatea verosimilității observației vectorului semnal *O* având în vedere că utilizatorul este în poziția *Lj* ca fiind *P*(*O*|*Lj*). Matematic, utilizatorul ar fi clasificat în poziția *Lj* dacă:

[1]

Așa cum a fost explicat mai sus, metodele de fingerprinting folosesc online RSSI și măsurători CSI pentru a mapa poziția utilizatorului/dispozitivului la un chenar discret; fiecare punct din acest chenar corespunde poziției în spațiu unde măsurătoarea offline care corespunde celei online (fingerprints) a fost obținută. Așadar, fingerprinting ca și metodă oferă estimări discrete mai degrabă decât continue a locației utilizatorului/dispozitivului. De asemenea, este de menționat faptul că tehnicile de fingerprinting și de analiză a scenei pot oferi estimări precise ale locației, având în vedere că depinde de măsurători online cât și offline la diferite momente de timp, însă sunt susceptibile la schimbări ale mediului cu timpul. [1]

1. *Rețele neuronale artificiale (ANN)*

Rețelele neuronale artificiale sunt folosite la o sumedenie de clasificări și scenarii de prezicere. Pentru localizare, rețeaua neuronală trebuie să fie antrenată folosind valori RSSI și coordonatele corespunzătoare care sunt obținute în timpul fazei offline. Odată ce ANN a fost antrenat, poate atunci să fie folosit pentru a obține locația utilizatorului bazată pe măsurătorile RSSI online. Rețeaua Perceptronului Multi-Layer (MLP) cu un singur nod ascuns este cel mai des utilizat ANN pentru localizare. În localizarea bazată pe MLP, un vector dat ca și input a măsurătorilor RSSI este înmulțit cu ponderilor de input și adăugat într-un strat bias de strat de input, având în vedere că biasul este selectat. Rezultatul obținut este mai apoi pus într-o funcție de transfer a stratului ascuns. Produsul rezultatului funcției de transfer și a ponderilor antrenate a stratului ascuns este adăugat la biasul stratului ascuns (dacă biasul este ales). Rezultatul obținut este estimarea locației utilizatorului. [1]

1. *K-Nearest-Neighbour (kNN)*

Algoritmii kNN se bazează pe măsurătorile RSSI online ca să obțină cele mai apropiate k potriviri (pe baza măsurătorilor offline de RSSI stocate în baza de date) a locațiilor cunoscute folosind Root Mean Square Error (RMSE). Cele mai apropiate potriviri sunt apoi aproximate pentru a obține estimarea locației dispozitivului/utilizatorului. Un kNN ponderat este folosit dacă distanțele sunt luate ca și ponderi în spațiul semnalelor, altfel un kNN neponderat este folosit. [1]

1. *Support-Vector Machines (SVM)*

SVM este o soluție atractivă pentru clasificarea datelor dar și pentru regresie. SVM este în principal folosit pentru machine learning (ML) și analiză statistică și are precizie ridicată. [1]

1. *Angle of Arrival (AoA)*

Soluțiile bazate pe AoA folosesc un arrayuri de antene (pe partea receptorului) pentru a estima unghiul la care semnalul transmis împunge receptorul prin exploatarea și calcularea diferenței de timp la sosire a elementelor individuale ale arrayului de antene. Principalul avantaj al AoA este că locația dispozitivului/utilizatorului pot fi estimate cu o valoare la fel de mică ca și 2 monitoare într-un spațiu bidimensional, sau trei monitoare într-un spațiu tridimensional. Chiar dacă AoA poate oferi estimări precise când distanța dintre transmițător și receptor este mică, necesită hardware mai complex și calibrarea atentă în comparație cu tehnicile RSS, în timp ce acuratețea sa se deteriorează odată cu creșterea în distanța dintre Tx și Rx unde o eroare cât de măruntă în calcularea unghiului de sosire duce la o eroare colosală în estimarea locației propriu-zisă. Mai mult de atât, din cauza efectelor de multipath în spații interioare, AoA din punct de vedere al Line of Sight (LoS) este de obicei greu de obținut. [1]

A diagram of a waveform

Description automatically generated with medium confidence

Figură 4.4 - Exemplu AoA [1]

1. *Time of Flight (ToF)*

Time of Flight sau Time of Arrival (ToA) se folosește de timpul de propagare al semnalului pentru a calcula distanța dintre transmițătorul Tx și receptorul Rx. Valoarea ToF multiplicată cu viteza luminii oferă distanța fizică dintre Tx și Rx. În următoarea figură, ToF de la 3 noduri diferite de referință este folosit pentru a estima distanțele dintre aceste noduri și dispozitiv. Geometria simplă poate fi folosită pentru a calcula locația dispozitivului luând în considerare AP-urile. [1]

ToF necesită sincronizare strictă între transmițători și receptori și, în multe cazuri, timestamp-uri sunt transmise cu semnalul (depinzând de protocolul de comunicare care procesează). Factorii cheie care afectează precizia estimării ToF sunt lățimea de bandă a semnalului și rata de sampling. Rata mică de sampling (în timp) reduce rezoluția ToF de când semnalul poate ajunge între intervale de sample-uri. Tehnicile de super-rezoluție a frecvenței domeniului sunt de obicei folosite pentru a obține ToF cu rezoluție ridicată de la răspunsul frecvenței canalului. În medii interioare multipath, cu cât este mai largă lățimea de bandă, cu atât mai ridicată rezoluția estimării ToF. Chiar dacă lățime de bandă largă și tehnicile de super-rezoluție pot îmbunătății performanța ToF, ele tot nu pot elimina erorile de localizare semnificative când calea directă a LoS dintre transmițător și receptor nu este disponibilă. Așa ceva se întâmplă când obstacole deviază semnalul emis, care mai apoi traversează printr-o cale mai lungă cauzând astfel o creștere a timpului necesar a semnalului să se propage de la Tx la Rx. Fie *t*1 timpul când Tx-ul *i* trimite un mesaj către Rx-ul *j* care primește mesajul respectiv la timpul *t*2 unde *t*2 = *t*1 + *tp* (*tp* este timpul necesar ca semnalul să traverseze de la Tx la Rx). Deci distanța dintre *i* și *j* poate fi calculată cu ecuația: [1]

unde este viteza semnalului [1]

1. *Time Difference of Arrival (TDoA)*

Time Difference of Arrival (TDoA) se folosește de diferențele de timp al propagării semnalelor de la diferitele transmițătoare, măsurate la receptor. Tehnica asta este diferită de ToF, unde valoarea absolută a timpului de propagare este folosit. Măsurătorile TDoA ( *TD(i,j)* – de la transmițătorii *i* și *j*) sunt convertite în valori de distanțe fizice , unde *c* este viteza luminii. Receptorul este acum localizat pe hiperboloidul dat de ecuația: [1]

unde sunt coordonatele transmițătorului/nodului de referință *i* și (*x,y,z*) sunt coordonatele receptorului/utilizatorului. Este necesar TDoA de la cel puțin 3 transmițători pentru a calcula locația exacta a receptorului rezultând intersecția a 3 (sau mai multe) hiperboloide. Sistemul de ecuații hiperboloide poate fi rezolvat fie prin regresie liniară sau prin liniarizarea ecuației folosind extinderea seriilor Taylor. Următoarea figură arată cum 3 noduri de referință diferite pot fi folosite pentru a obține locații 2D a oricărui obiect. Figura arată hiperbolele formate ca și rezultat a măsurătorilor obținute de la nodurile de referință pentru a obține poziția utilizatorului (punctul negru). [1]

A picture containing drawing, clock, line, sketch

Description automatically generated

Figură 4.5 - Exemplu TDoA [1]

Precizia estimării TDoA ține (similar ca la tehnicile ToF) de lățimea de bandă a semnalului, rata de sampling a receptorului și de existența a LoS între transmițător și receptor. Sincronizare strictă este de asemenea necesară, dar, diferit de ToF unde sincronizarea este necesară între Tx și Rx, în cazul TDoA doar sincronizarea între transmițători este importantă. [1]

1. *Return Time of Flight (RToF)*

RToF măsoară drumul în inel (transmițător-receptor-transmițător) al timpului de propagare al semnalului pentru a estima distanța dintre Tx și Rx. Mecanismele de gamă ToF și RToF sunt similare; la primirea de semnal de la un transmițător, receptorul răspunde înapoi transmițătorului, care mai apoi calculează totalul ToF de drum în inel. Principalul beneficiu al RToF este că o sincronizarea a ceasului relativ moderată între Tx și Rx este necesară, în comparație cu ToF. Chiar și așa, precizia estimării RToF este afectată de aceiași factori ca ToF (adică, rata de sampling, lățimea de bandă) care în acest caz sunt mai severe din moment ce semnalul este transmis și recepționat de două ori. O altă problemă semnificativă cu sistemele bazate pe RToF este răspunsul întârziat al receptorului care depinde mult de overheadurile protocolului folosit precum și de electronica din spatele receptorului. Protocolul folosit poate fi neglijat dacă timpul de propagare dintre transmițător și receptor este mare comparativ cu timpul de răspuns, totuși întârzierea nu poate fi ignorată în sistemele cu bătaie scurtă cum ar fi cele de localizare interioară. Fie *t*1 momentul de timp când Tx-ul *i* trimite un mesaj la Rx-ul *j* care primește mesajul la momentul de timp *t*2 unde , la momentul de timp *t*3, transmite un semnal înapoi la *i* care primește la momentul *t*4. Deci distanța dintre *i* și *j* poate fi calculat folosind ecuația. [1]

1. *Phase of Arrival* (PoA)

Soluții bazate pe PoA se folosesc faze sau diferența de faze a semnalului dat de purtătoare pentru a estima distanța între transmițător și receptor. O presupunere uzuală pentru determinarea semnalului de fază pe partea receptorului este că semnalul transmis de la nodul ancoră sau dispozitivul utilizatorului sunt pur forme sinusoidale având aceiași frecvență și zero defazaj. Sunt o sumedenie de tehnici disponibile de estimare a razei de acțiune sau a distanței dintre Tx și Rx folosind PoA. O tehnică este de a se presupune că există o întârziere finită de tranzit *Di* între Tx și Rx, care poate fi exprimată ca și o fracție a lungimii de undă a semnalului. În figura următoare, incidentul semnalează sosirea semnalelor cu o diferență de fază la antene diferite în arrayul de antene, care poate fi folosită pentru a obține locația utilizatorului. [1]

A picture containing sketch, diagram, line, drawing

Description automatically generated

Figură 4.6 - Exemplu PoA [1]

Problema cu soluția PoA este că are nevoie de LoS pentru precizie ridicată, care nu prea este cazul în spații interioare. [1]

Următorul tabel afișează un sumar a tot ce am relatat mai sus despre tehnicile de localizare în spații interioare și arată avantajele și dezavantajele fiecăruia. [1]

Tabel 2 - Comparație algoritmi [1]

|  |  |  |
| --- | --- | --- |
| Soluție | Avantaje | Dezavantaje |
| RSSI | Ușor de implementat, eficient din punct de vedere al costului, poate fi folosit pentru o sumedenie de tehnologii | Predispus la multipath fading și la zgomot, precizie scăzută a localizării, poate necesita fingerprinting |
| CSI | Mai robust la zgomot și la multipath | Nu este atât de ușor de implementat |
| AoA | Poate să ofere o precizie ridicată a localizării, nu necesită fingerprinting | Poate avea nevoie de mai multe antene și hardware complex, performanța scade odată cu distanța dintre transmițător și receptor. Are nevoie de algoritmi complexi. |
| ToF | Oferă precizie ridicată, nu are nevoie de fingerprinting | Necesită sincronizare de timp între transmițători și receptori, poate avea nevoie de timestamps și antene multiple. LoS este obligatoriu pentru performanță |
| TDoA | Nu necesită fingerprinting, nu necesită sincronizare de ceasuri între dispozitiv și noduri de referință | Are nevoie de sincronizare între nodurile de referință, poate avea nevoie de timestamps, are nevoie de o lățime de bandă mai mare |
| RToF | Nu necesită fingerprinting, poate oferi precizie ridicată | Are nevoie de ceasuri sincronizate, întârzierea la procesare poate afecta performanța pentru măsurători cu bătaie scurtă |
| PoA | Poate fi folosit împreună cu RSS, ToA, TDoA pentru a îmbunătății performanța per total | Performanță degradantă în lipsa LoS |
| Fingerprinting | Destul de ușor de folosit | Noi fingerprints sunt necesare chiar și dacă variația de spațiu este minoră. |

## NS3

Primul lucru care trebuie făcut înainte să intrăm adânc în ns3 sau să ne apucăm să scriem cod este de a explica câteva dintre conceptele de bază și abstractizările din sistem. Multe dintre acestea vor părea transparent de evidente pentru unii, însă se recomandă să se aloce timpul necesar pentru citirea următoarelor concepte pentru a putea începe cu o fundație stabilă. [4]

### Abstractizări cheie

În următoarele exemple se vor regăsi termeni comuni folosiți în rețelistică, dar au un înțeles specific în ns-3. [4]

* **Node**

În jargonul internetului, un dispozitiv de calcul care se conectează la o rețea este numit drept *host* sau uneori *end system.* Deoarece ns-3 este un simulator de rețea, nefiind totuși un simulator de *Internet*, intenționat nu se folosește termenul de host din moment ce este strâns asociat cu Internetul și protocoalele acestuia. În schimb, se folosește un termen mai generic de asemenea folosit și de alte simulatoare ce izvorăsc din teoria grafurilor – termenul de *node.* [4]

În ns-3 abstractizarea dispozitivului comun de calcul se numește node. Această abstractizare este reprezentată în C++ de clasa **Node**. Clasa **Node** aduce metode pentru managementul reprezentărilor dispozitivelor de calcul în simulări. [4]

O analogie corectă pentru termenul de Node este un calculator căruia i se va adăuga funcționalitate. Se pot adăuga lucruri precum aplicații, stive de protocol sau carduri de periferice cu driverele asociate acestora pentru a lăsa calculatorul să execute diferite lucruri folositoare. Se folosește același model de bază și în ns-3. [4]

* **Application**

Tipic, software-ul de pe calculatoare este divizat în două clase mari. Software de sistem care organizează varii resurse ale calculatorului precum memoria, ciclurile procesorului, diskul, rețeaua, etc., conform unui model de calcul. Software-ul de sistem de obicei nu folosește acele resurse pentru a completa task-uri care sunt benefice direct utilizatorului. Un utilizator va rula de obicei o aplicație care obține și folosește resursele controlate de către software-ul de sistem pentru a realiza un țel. [4]

De obicei, linia de separare între software-ul de sistem și cel de aplicație este realizată la schimbul nivelului de privilegii care se întâmplă în trap-urile sistemului de operare. În ns-3 nu există un concept real al sistemelor de operare și în special niciun concept al nivelelor de privilegii sau al apelurilor de sistem. Există, totuși, ideea de aplicație. La fel cum aplicațiile software rulează pe un calculator pentru a îndeplini taskuri în ”lumea reală”, aplicațiile ns-3 rulează pe noduri ns-3 pentru a îndeplini simulări în lumea simulată. [4]

În ns-3 abstractizarea de bază pentru un program utilizator care generează o anume activitate pentru a fi simulată este aplicația. Această abstractizare este reprezentată în C++ prin clasa **Application.** Clasa **Application** aduce metode de management al reprezentării versiunii ns-3 a aplicațiilor de la nivelul utilizator care vor să fie simulate. Dezvoltatorii sunt așteptați să specializeze clasa Application în sensul programării orientate pe obiect pentru a crea noi aplicații. [4]

* **Channel**

În lumea reală, cineva poate conecta un calculator la o rețea. Adesea, mediul peste care aceste date se transmit în aceste rețele se numesc canale (channels). Când se conectează un cablu Ethernet la socket-ul din perete, se conectează calculatorul la un canal de comuncație Ethernet. În lumea simulată ns-3, cineva poate conecta un Node la un obiect care reprezintă un canal de comunicație. Aici, abstractizarea de bază a unei subrețele de comuncație este numită channel și este reprezentată în C++ prin clasa **Channel**. [4]

Clasa **Channel** aduce metode de management al obiectelor subrețelei de comunicație și a conectării nodurlor la ele. Canalele pot de asemenea să fie specializate de către dezvoltatori în sensul programării orientate obiect. Specializarea unui canal poate modela ceva simplu precum un fir. Canalul specializat poate modela de asemenea ceva complicat precum un switch mare de Ethernet, sau un spațiu tridimensional plin de obstrucții în cazul rețelelor wireless. [4]

* **Net Device**

Se obișnuia ca dacă se voia să se conecteze un calculator la o rețea, trebuia să se cumpere un anumit cablu de rețea și un dispozitiv hardware numit (în terminologia PC) un *card periferic* care avea nevoie să fie instalat pe un calculator. Dacă cardul periferic implementa unele funcții de rețea, se numeau Network Interface Cards, sau *NICs*. Astăzi majoritatea calculatoarelor vin cu interfață hardware de rețea din fabrică și utilizatorii nu mai văd aceste blocuri arhitecturale. [4]

Un NIC nu va funcționa fără un driver software care să controleze hardware-ul. În Unix (sau Linux), o poesă a hardware-ului periferic este clasificat ca fiind un dispozitiv. Dispozitivele sunt controlate folosind drivere de dispozitive, și dispozitivele de rețea (NICs) sunt controlate folosind drivere de dispozitive de rețea cunoscute colectiv drept *net devices*. În Unix și Linux se face referință după nume precum *eth0.* [4]

În ns-3 abstractizarea *net device* acoperă și driverele de software și hardware-ul simulat. Un net device este ”instalat” într-un Node pentru a putea lăsa Node-ul să comunice cu alte Node-uri în simulare prin intermediul Channel-urilor. Ca și într-un calculator real, un Node poate fi conectat la mai mult de un Channel prin multiple NetDevices. [4]

Abstractizarea net device în ns-3 este reprezentată în C++ prin clasa **NetDevice**. Această clasă aduce metode de management al conexiunilor dintre Node-uri și obiectele de tip Channel; și pot fi specializate de către dezvoltatori în sensul programării orientate obiect. [4]

* **Topology Helpers**

Într-o rețea reală, se vor găsi calculatoare gazdă (host) cu NIC-uri adăugate (sau din fabrică). În ns-3 se vor găsi Node-uri cu NetDevices atașate. Într-o rețea simulată de dimensiuni mari vor trebui aranjate multe conexiuni între Node-uri, NetDevice-uri și Channel-uri. [4]

De când conectarea NetDevice-urilor la Node-uri, NetDevice-urilor la Channel-uri, asignarea de adrese IP, etc., sunt taskuri atât de comune în ns-3, se aduc spre folosire ce ns-3 numește drept *topology helpers* pentru a face acest lucru cât mai ușor cu putință. De exemplu, poate lua o mulțime de pași fundamentali în ns-3 pentru a crea un NetDevice, pentru a adăuga o adresă MAC, a instala acel NetDevice pe un Node, de a configura acelui nod stiva de protocol, și mai apoi de a conecta acel NetDevice la un Channel. Chiar mai multe operații sunt necesare pentru a conecta mai multe dispozitive la Channel-uri multi-punct și apoi de a conecta rețele individuale împreună în rețele interconectate. Se oferă obiecte de topology helpers care combină toate aceste operații distincte într-un model ușor pentru conveniență. [4]

## JavaFX

JavaFX este un framework bazat pe Java de tipul open-source folosit pentru dezvoltarea de aplicații client mai bogate. Este comparabil cu alte framework-uri de pe piață precum Adobe AIR și Microsoft Blazer. JavaFx este de asemenea succesorul Swing în spațiul dezvoltării tehnologiei de interfețe grafice utilizator (GUI) în platforma Java. Principalele caracteristici JavaFX sunt: [10]

* JavaFX este scris în Java, ceea ce este avantajos din punct de vedere al folosirii de caracteristici Java precum multithreading, generics sau lambda expressions. Se pot folosi orice IDE-uri care suportă Java, precum NetBeans sau Eclipse pentru a compila, a rula, a debuga și a împacheta aplicații Java. [10]
* JavaFX suportă data binding prin librăriile proprii. [10]
* JavaFX oferă un set bogat de suport multimedia precum redarea de conținut audio sau video. Acesta profită de codec-urile de pe platformă. [10]
* JavaFX permite imbricarea conținutului web în aplicație. [10]

În spatele API-ului JavaFX se află numeroare componente ce profită de librăriile native Java și de software-ul și hardware-ul disponibil. [10]

A picture containing text, screenshot, line, font

Description automatically generated

Figură 4.7 - Componentele platformei JavaFX [10]

GUI-ul din JavaFX este construit ca și un graf de scenă (*scene graph*). Un graf de scenă este o colecție de elemente vizuale, numite noduri, aranjate într-o manieră ierarhică. Un graf de scenă este construită folosind API-ul public al JavaFX. Nodurile dintr-un graf de scenă pot trata inputul utilizatorului precum și gesturile utilizatorului. Acestea pot avea efecte, transformări și stări. Tipurile de noduri într-un graf de scenă includ controale UI simple cum ar fi butoane, câmpuri de text, forme bidimensionale (2D) și tridimensionale (3D), imagini, media (audio și video), conținut web și grafice. [10]

*Prism* este un pipeline grafic hardware-accelerated folosit pentru randarea grafului de scenă. Daca randarea hardware-accelerated nu este disponibilă pe platformă, Java 2D este folosit ca și opțiune de rezervă pentru randare. De exemplu, înainte de Java 2D, va încerca să folosească DirectX pe Windows și OpenGL pe Mac, Linux, și platformele imbricate. [10]

*Glass Windowing Toolkit* oferă grafică și servicii de ferestre cum ar fi ferestre și timer care se folosesc de sistemul nativ de operare. Toolkit-ul este de asemenea responsabil pentru managementul cozii de evenimente. În JavaFX, cozile de evenimente sunt conduse de un singur thread la nivelul sistemului de operare numit JavaFX Application Thread. Toate evenimentele inputului utilizator sunt dispecerate pe firul de execuție al aplicației JavaFX. Acesta are nevoie ca graful de scenă live să fie modificat doar pe firul principal de execuție. [10]

*Media Engine* este responsabil de oferirea de suport media în JavaFX, de exemplu, redarea de audio sau video. Profită de disponibilitatea codec-urilor de pe platformă. Acesta folosește un thread separat de procesare a frame-urilor media și folosește thread-ul principal pentru a sincroniza frame-urile cu graful de scenă. Acesta este bazat pe *GStreamer*, care este un framework multimedia de tip open-source. [10]

*Web Engine* este responsabil de procesarea conținutului web (HTML) imbricat în graful de scenă. Prism este responsabil de randarea conținutului web. Motorul web este bazat pe *WebKit,* care este un motor de căutare web de tip open-source. HTML5, Cascading Style Sheets (CSS), JavaScript, și Document Object Model (DOM) sunt suportate pe această platformă. [10]

*Quantum toolkit* este o abstractizare peste componentele de nivel scăzut ale Prism, și a motoarelor Glass, media și web. Facilitează de asemenea coordonarea între componentele de nivel scăzut. [10]

## Wi-Fi (IEEE 802.11)

Standardul IEEE 802.11 (Wi-Fi) este standardul dominant pentru Wireless Local Area Networks (WLAN). În 1997, IEEE a adoptat standardul 802.11 ca primul standard WLAN. Acesta relaționează cu stratul Medium Access Control (MAC) și cu stratul Physical (PHY). Figura următoare arată standardele celor 2 straturi recent menționate. [11]

A screenshot of a computer

Description automatically generated with low confidence

Figură 4.8 - Standardele MAC și PHY în 802.11 [11]

* **Stratul Physical**

Există doar un strat MAC, însă diferite standarde au fost propuse pentru stratul fizic. Standardul definește diferite metode de modulație: Infraroșu, Direct sequence spread spectrum (DSSS), Frequency-Hopping spread spectrum (FHSS), Orthogonal Frequency Division Multiplexing (OFDM), și de asemenea definește trei tehnologii diferite ale stratului fizic: IEEE 802.11a, IEEE 802.11b și IEEE 802.11g. [11]

IEEE 802.11a operează pe banda de 5GHz. Teoretic, oferă o rată de date de 54 Mbps, dar mai realist atinge un debit undeva între 20 Mbps până la 25 Mbps în condiții de trafic normal. IEEE 802.11b operează pe banda de 2.4GHz și are o rată de date maximă teoretică de până la 11 Mbps cu o medie a debitului undeva între 4 Mbps până la 6 Mbps. IEEE 802.11g oferă rate de la 6 Mbps la 54 Mbps. Folosește modulația OFDM. [11]

* **Arhitecturile de bază**

Arhitectura IEEE 802.11 este formată din multiple componente de bază. Unele dintre componentele importante sunt următoarele: [11]

* *Station* (*STA*): Orice dispozitiv care conține un MAC conform cu IEEE 802.11 și o interfață a stratului fizic (PHY) la mediul wireless (WM).
* *Access Point* (*AP*): Orice entitate care are o funcționalitate de stație și oferă acces la servicii de distribuție prin mediul wireless (WM) pentru stațiile asociate.
* *Basic Service Set* (*BSS*): BSS-ul este un bloc de construcție a WLAN-ului IEEE 802.11. BSS-ul este format dintr-un grup de orice număr de stații.
* *Basic Service Area* (*BSA*): Aria conceptuală din interiorul căreia, membrii BSS pot comunica.

A picture containing text, screenshot, circle, number

Description automatically generated

Figură 4.9 - BSS al WLAN-ului IEEE 802.11 [11]

* **Topologiile de rețea IEEE 802.11**

Topologia rețelei WLAN-ului IEEE 802.11 are blocul de bază de construcție BSS. Există două tipuri de topologii LAN pentru LAN-urile wireless, un *Independent Basic Service Set* (IBSS) și un *Infrastructure Basic Service Set*. [11]

IBSS-ul este cel mai uzual tip de LAN IEEE 802.11. Stațiile sunt conectate fiecare cu fiecare prin mediul wireless pe baza peer-to-peer. Cu toate astea, toate stațiile din IBSS s-ar putea să nu poată comunica între ele. Acest mod de operare este posibil când stațiile IEEE 802.11 pot comunica direct. Din cauza acestui tip de LAN IEEE 802.11 fiind de obicei format fără pre-planificare, pentru atâta vreme cât LAN-ul este necesar, tipul acesta de operație este deseori referit ca și o rețea de tip *ad-hoc.* [11]

Un Infrastructure Basic Service Set este un BSS cu componenta centrală Access Point (AP) care performează funcții de releu. Toate stațiile BSS comunică prin AP. Unele avantaje sunt că este mai simplu din moment ce stațiile nu trebuie să mențină relații de vecinătate, poate îmbunătății raza BSA-ului și se poate utiliza ceva putere pentru a salva mecanismele pentru stații. O stație (STA) se poate asocia cu un singur AP la un anumit moment de timp și este săvârșită prin procesul numit *asociere (association).* AP-ul poate fi atașat la un Ethernet cablat sau wireless și poate oferi conexiune la internet nodurilor implicate. [11]

A close-up of a computer

Description automatically generated with low confidence

Figură 4.10 - IBSS și Infrastructure BSS [11]

# Proiectare de detaliu și implementare

## Arhitectura globală a sistemului

A picture containing text, diagram, line, receipt

Description automatically generated

Figură 5.1- Arhitectura globală a sistemului

Module:

**JavaFX –** cuprinde tot ce înseamnă interfața grafică a aplicației dezvoltate, se folosește de programarea orientată obiect de a procesa datele oferite de către utilizator, ulterior le împachetează într-o formă ușor de transmis și citit de către NS3 (scrierea într-un fișier sub un format bine definit).

**NS3 –** cuprinde fișierele de tip ”header” și cele de tip ”source” în care sunt implementate capabilitățile librăriei de a putea simula evenimente discrete în domeniul rețelisticii. De asemenea, primește datele din fișierul anterior menționat din JavaFX, procesează datele și pornește simularea propriu-zisă a algoritmului dorit. În final, scrie înapoi datele obținute într-un fișier din care JavaFX poate citi.

## Diagrama de utilizare

A diagram of a person

Description automatically generated with low confidence

Figură 5.2 - Cazurile de utilizare

**Caz de utilizare:** Selectează algoritmul dorit

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația trebuie să fie pornită

**Postcondiții:**

**Scenariu succes:**

* Algoritmul dorit de simulat este selectat în cadrul meniului

**Scenariu eroare:**

* Algoritmul nu este setat, aplicația nu poate rula fără un algoritm setat

**Pași:**

1. Utilizatorul APASĂ pe meniul de tip dropdown intitulat ”Select an algorithm”.
2. Utilizatorul selectează unul dintre algoritmii disponibili.
3. Algoritmul este setat.

**Caz de utilizare:** Introduce AP în sistem

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația trebuie să fie pornită

**Postcondiții:**

**Scenariu succes:**

* Lista cu AP-uri este updatată.
* Noul AP poate fi văzut în tabel
* Căsuțele de input pentru AP vor fi golite pentru introducerea unui nou AP

**Scenariu eroare:**

* Un AP există deja în spațiul dorit
* Căsuțele inputului devin roșii în semn de eroare

**Pași:**

1. Utilizatorul introduce coordonata X.
2. Utilizatorul introduce coordonata Y.
3. Utilizatorul apasă pe butonul ”Add AP”.
4. AP-ul este adăugat cu succes în listă.

**Caz de utilizare:** Editează un AP din sistem

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația este pornită.
* Lista de AP-uri nu este goală.

**Postcondiții:**

**Scenariu succes:**

* AP-ul a fost editat cu succes.
* AP-ul editat se poate observa în listă.

**Scenariu eroare:**

* AP-ul respectiv există deja în cazul setării unui ID existent

**Pași:**

1. Utilizatorul apasă pe unul din AP-urile definite în listă.
2. Aplicația transcrie informațiile AP-ului în căsuțele aferente.
3. Utilizatorul modifică datele AP-ului.
4. Utilizatorul apasă pe butonul ”Edit AP”.
5. AP-ul este editat cu succes și plasat în listă sub noua formă.

**Caz de utilizare:** Șterge un AP din sistem

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația este pornită.
* Lista de AP-uri nu este goală

**Postcondiții:**

**Scenariu succes:**

* AP-ul este șters cu succes.
* AP-ul șters nu mai apare în listă.

**Scenariu eroare: -**

**Pași:**

1. Utilizatorul apasă pe unul din AP-urile definite în listă.
2. Aplicația transcrie informațiile AP-ului în căsuțele aferente.
3. Utilizatorul apasă pe butonul ”Delete AP”.
4. AP-ul este șters cu succes și scos din listă.

**Caz de utilizare:** Desenează chenar al simulării

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația este pornită.
* Datele despre chenar sunt introduse în căsuțele speciale
* Datele despre diviziune sunt introduse în căsuțele speciale

**Postcondiții:**

**Scenariu succes:**

* Chenarul va fi desenat pe ecran.
* Chenarul va desena AP-urile din listă în funcție de poziția lor.
* Aplicația se va extinde pentru a permite continuarea execuției.

**Scenariu eroare:**

* Chenarul nu va fi desenat.
* Căsuțele speciale vor fi colorate cu roșu în semn de eroare

**Pași:**

1. Utilizatorul introduce valori relevante pentru căsuțele speciale ale desenării chenarului
2. Utilizatorul apasă pe butonul ”Draw Grid”.
3. Chenarul poate fi vizionat pe ecran.
4. Aplicația și-a extins dimensiunile pentru permiterea continuării.

**Caz de utilizare:** Șterge chenarul curent

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația este pornită.
* Un chenar a fost desenat înainte.

**Postcondiții:**

**Scenariu succes:**

* Aplicația revine la forma inițială.
* Chenarul creat înainte dispare.

**Scenariu eroare: -**

**Pași:**

1. Utilizatorul a creat recent un chenar.
2. Utilizatorul apasă pe butonul ”Clear Grid”.
3. Chenarul dispare din aplicație.
4. Aplicația revine la forma inițială.

**Caz de utilizare:** Pornește simularea

**Actor principal:** Utilizator

**Precondiții:**

* Aplicația trebuie să fie pornită
* Un chenar trebuie să existe.
* Lista de AP-uri trebuie să fie definită.
* Un algoritm trebuie să fie selectat.

**Postcondiții:**

**Scenariu succes:**

* Aplicația începe rularea programului NS3
* Un mesaj de așteptare se va afișa în timpul rulării.
* Outputul din NS3 va fi afișat după procesare în chenarul special de output.

**Scenariu eroare:**

* Nu există algoritm selectat, meniul va deveni roșu în semn de eroare.
* Nu există destule AP-uri în sistem, aplicația va afișa o eroare.

**Pași:**

1. Utilizatorul apasă pe butonul ”Compute”.
2. Utilizatorul așteaptă execuția aplicației.
3. Utilizatorul observă outputul în căsuța specială de output.

## Diagrama de secvență

A picture containing text, diagram, parallel, number

Description automatically generated

Figură 5.3 - Diagrama de secvență pentru un scenariu complet

Diagrama de secvență surprinde interacțiunea utilizatorului cu sistemul dezvoltat, pentru a săvârși procesele necesare îndeplinirii cazurilor de utilizare anterior definite.

Utilizatorul va configura sistemul astfel încât simularea ce se va realiza să fie cea dorită, introducând algoritmul dorit ce se vrea a fi simulat, date corespunzătoare în ceea ce privește poziția relativă a dispozitivelor wireless de tip AP, cât și a definirii spațiului în care tuturor acestora le vor fi simulate capabilitățile wireless.

MainView (fiind resursa principală de interfață grafică din proiectul JavaFX creat) va strânge toate aceste date ca input utilizator. La apăsarea butonului de execuție, toate datele vor fi trimise la următorul nivel.

MainController este clasa ce se ocupă de tot ce se întâmplă în interiorul clasei de MainView. Acesta primește datele din View și le procesează, iar mai apoi le trimite sub forma unui fișier către următorul strat al aplicației.

NS3 primește referința la fișierul în care stratul anterior a plasat datele relevante simulării, făcând rost de date, le trimite mai departe la clasa specializată din interiorul său de a realiza simularea propriu-zisă.

Algorithms Simulator este clasa din interiorul NS3 abilitată să despacheteze datele din interiorul fișierului, de a crea date interne relevante cu ajutorul lor (poziții relative în spațiul simulat al dispozitivelor, precum și algoritmul pe care în urmărește a fi simulat). Clasa trimite înapoi o referință spre fișierul în care s-au salvat datele relevante simulării.

## JavaFX

**JavaFX** este un framework ce permite dezvoltarea de aplicații Java ce dispun de o interfață grafică modulară, ușor de dezvoltat, ce se folosește de standardul Extensible Markup Language (XML) pentru a defini diferitele componente aflate în aplicația dezvoltată. Interfața grafică utilizator (GUI) creată conține diferite elemente de interacționare, precum butoane, căsuțe de text, meniu de tip dropdown, precum și elemente grafice în care se pot observa schimbări ale stării sistemului, precum o listă a dispozitivelor aflate în sistem sau un chenar desenabil în care apar locațiile dispozitivelor relative.

Colectarea datelor este realizată prin conceptul de Controller al JavaFX care se ocupă de tot ce ține de resursele grafice definite prin XML. Acest Controller captează datele, fiind atașat la o resursă grafică XML. Astfel toate elementele grafice sunt definite în prealabil pentru a putea interacționa cu Controller-ul aferent prin adnotarea ”@FXML”.

### Arhitectura

A picture containing diagram, plan, technical drawing, schematic

Description automatically generated

Figură 5.4 - Arhitectura JavaFX de tip MVC

Arhitectura părții aplicației de JavaFX este construită pe baza șablonului arhitectural Model-View-Controller (MVC).

Componentele arhitecturii sunt următoarele:

* **Model –** în care sunt definite clasele de model ale aplicației, care stabilesc pozițiile relative ale unui dispozitiv, spațiul în care se desfășoară simularea precum și AP-urile folosite, care se folosesc de celelalte 2 modele.
* **View –** care prezintă interfața grafică a aplicației, prin intermediul a două scene complementare. Una principală și alta care va fi lipită pe cea principală.
* **Controller –** Controllerul se ocupă de achiziția de date din interfață precum și de procesarea acesteia și începerea rulării simulării propriu-zise.

### Implementare

A picture containing text, diagram, plan, technical drawing

Description automatically generated

Figură 5.5 - Diagramă implementare JavaFX

Clase organizate pe componente sunt următoarele:

**Model:**

* Clasa **Point** – definește un set de coordonate X,Y drept parametrii pentru a putea înregistra pozițiile relative ale unui dispozitiv din simulare. Se folosește de clasa ”Lombok” pentru a genera automat metode de Getter și Setter în Java, precum și de a genera un constructor cu toți parametrii prezenți, toate acestea prin adnotările: @Getter, @Setter și @AllArgsConstructor. Clasa implementează metode de setare a locației în funcție de mai multe tipuri de date oferite pentru coordonatele unui punct. De asemenea, există și o metodă ”toString()” modificată pentru a afișa corect conținutul unui obiect de tip Point.

A screenshot of a computer program

Description automatically generated with medium confidence

Figură 5.6 - Clasa Point - cod sursă

* Clasa **Device** – definește un obiect de tipul AP sau de tipul STA ce va fi stocat într-o listă din interfața grafică a aplicației. Această clasă se folosește de un string pentru a conține un număr de identificare pentru un AP și un obiect de tipul Point pentru a stoca poziția relativă din interiorul spațiului de simulare. Pentru a defini un obiect de tipul STA, se va introduce ID-ul ”0”. Clasa implementează o metodă ”toString()” modificată pentru a afișa mai frumos un obiect de tipul AP sau de tipul STA.

A screen shot of a computer program

Description automatically generated with low confidence

Figură 5.7 - Clasa Device - cod sursă

**View:**

* Clasa **view.fxml** – conține cod de tip FXML care se încadrează în standardul FXML. Acest cod este compus din taguri ierarhice de HBox-uri, VBox-uri, butoane, label-uri, etc. Această clasă este folosită pentru a modela interfața grafică principală, în care sunt introduse selecția algoritmului dorit pentru a fi simulat, lista de AP-uri ce va fi trimisă spre NS3, precum și detalii referitoare la un chenar de desenare, în care se vor observa AP-urile din sistem.

A screenshot of a computer

Description automatically generated

Figură 5.8 - Interfață grafică clasa view.fxml

A picture containing text, screenshot

Description automatically generated

Figură 5.9 - Clasa view.fxml - început cod sursă

* Clasa **compunePanel.fxml** – conține a doua parte a interfeței grafice ce va fi lipită la cea principală în urma desenării chenarului grafic unde vor fi expuse pozițiile dispozitivelor wireless. Clasa este simplă, conține doar un buton care va porni simularea propriu-zisă (deci această posibilitate nu există dacă utilizatorul nu a executat toți pașii corecți până în punctul curent).

**A screenshot of a computer

Description automatically generated with low confidence**

Figură 5.10 - Interfață grafică computePanel.fxml

**A screenshot of a computer program

Description automatically generated with medium confidence**

Figură 5.11 - computePanel.fxml - cod sursă

**Controller:**

* Clasa **MainController** – este controllerul principal al aplicației. În interiorul acestei clase se întâmplă majoritatea logicii din spatele implementării. Clasa se ocupă de tot ce înseamnă input utilizator, îl primește de la clasa la care este conectat direct (view.fxml) și îl procesează în prealabil. Schimbări de stare, introducere de text, introducere de date eronate, introducere de date corecte, toate acestea sunt procesare și afișate înapoi spre utilizator de către clasa curentă. Această clasă definește o mulțime de taguri FXML ce țin de elementele grafice din care este construită interfața.

A screen shot of a computer program

Description automatically generated with medium confidence

Figură 5.12 - Clasa MainController - exemplu cod sursă

* Clasa **ComputeController –** este responsabilă de începerea execuției simulării propriu-zise. Are un ultim mecanism de verificare în care verifică dacă algoritmul este setat pentru a fi simulat. De asemenea, este clasa care controlează toată interacțiunea utilizatorului cu interfața auxiliară definită în clasa computePanel.fxml. Clasa ComputeController este creată în interiorul clasei MainController când utilizatorul desenează cu succes chenarul grafic care conține lista de dispozitive wireless ce vor fi trimise spre a fi simulate în librăria NS3. Dacă utilizatorul nu a rulat aplicația până în punctul curent, aplicația nu va afișa butonul final de ”Compute”, deci nu va putea începe simularea. Interfața auxiliară este afișată doar dacă utilizatorul parcurge toată aplicația fără eroare până în punctul curent.

A screen shot of a computer program

Description automatically generated with low confidence

Figură 5.13 - Clasa ComputeController - exemplu cod sursă

* Clasa **AreaCanvas** – Clasa curentă este responsabilă de trasarea corectă a chenarului grafic ce va conține lista de dispozitive wireless ce urmează a fi simulate. Chenarul necesită 4 valori, lungimea și lățimea spațiului în care vor fi plasate dispozitivele, precum și lungimea și lățimea unei diviziuni, pentru observarea mai bună a poziției față de origine, precum și numerotarea cadranelor mici pentru a reda poziția în cazul în care se dorește acest lucru.

**A screenshot of a computer program

Description automatically generated with medium confidence**

Figură 5.14 - Clasa AreaCanvas - exemplu cod sursă

* Clasa **Writer** - Clasa curentă este responsabilă de trimiterea datelor din interiorul interfeței grafice, împachetate de către MainController prin metoda definită din interiorul clasei curente. Clasa are un singur scop, scrierea datelor într-un fișier ce va fi pasat către librăria NS3. Metoda ”writeToFile” trimite lista de dispozitive din MainController și numele algoritmului în fișierul cu numele ”filename”. Dacă acesta nu există, aplicația va crea un nou fișier cu numele dorit.

A screenshot of a computer program

Description automatically generated with medium confidence

Figură 5.15 - Exemplu output clasa Writer

A screenshot of a computer program

Description automatically generated with medium confidence

Figură 5.16 - Clasa Writer - cod sursă

## Librăria NS3

### Structură

Librăria NS3 dispune de mai multe module implementate de către echipa din spate pentru a ușura treaba dezvoltatorilor în construirea și simularea de topologii de rețea dorite. Librăria a împachetat logica din spate prin clase de tipul Helper. Aplicația curentă folosește o mulțime de module în care sunt incluse numeroase clase de tip helper ce servesc spre realizarea topologiei dorite. În cazul curent, topologia dorită este de forma Wi-Fi infrastructură, în care mai multe dispozitive de tipul AP sunt definite într-un spațiu, ele fiind denumite ”ancore”, din cauza poziției fixe pe care o au. În mijlocul spațiului se află un dispozitiv mobil cu capabilități de conexiune Wi-Fi care se va asocia pe rând cu toate AP-urile definite, primind câte un pachet de la fiecare și măsurând datele relevante fiecărui algoritm dorit pentru simulare.

S-a dezvoltat o clasă numită ”alg1.cc” în care se va realiza simularea propriu-zisă a topologiei create cu ajutorul listei de dispozitive.

Astfel, fișierul sursă este format din următoarele componente importante de cod ns3:

A screen shot of a computer program

Description automatically generated with low confidence

Figură 5.17 - cod ns3 - achiziția de date

Primele două rânduri permit aplicației să printeze detalii despre metadata dintr-un pachet, precum și permiterea verificării metadatei unui pachet.

Următoarele rânduri stabilesc parametrii aplicației ce trebuiesc a fi oferiți în linia de comandă, pentru a putea stabili numărul corect de dispozitive AP precum și a algoritmului ce trebuie a fi simulat. Toate acestea se pot observa în figura 5.17

A picture containing text, screenshot, software

Description automatically generated

Figură 5.18 - metoda parseFile()

Metoda parseFile() parsează toate datele din fișierul primit ca și parametru ce conține numele algoritmului, poziția clientului, numărul de AP-uri, precum și pozițiile relative ale acestora. Metoda setează valorile corespunzătoare variabilelor de algoritm și de număr de dispozitive în interiorul metodei, prin transmiterea de referință la variabilele respective, astfel făcându-se legătura corect la locul din memorie unde se află aceste variabile. Această metodă se poate observa în figura 5.18

A black screen with white text

Description automatically generated with low confidence

Figură 5.19 - Definirea de noduri

Snippet-ul de cod din figura 5.19 evidențiază definirea nodurilor ce vor fi folosite în sistem pentru topologie. Se definesc toate nodurilor prin clasa NodeContainer. Mai întâi se definește nodul de stație (STA), mai apoi nodurile de AP-uri. Definind stația prima, aceasta va avea ID-ul unic global în sistem 0.

A screen shot of a computer

Description automatically generated with low confidence

Figură 5.20 - Setarea Wi-fi

În figura 5.20 se folosește clasa Helper necesară pentru a seta complexitatea aferentă unui sistem de conexiune Wi-Fi. Prin metodele acesteia, se definește standardul ce va fi folosit din Wi-Fi (în cazul de față, standardul 802.11a), precum și tipul de manager al stațiilor remote. (AarfWifiManager).

A picture containing text, screenshot, font

Description automatically generated

Figură 5.21 - Stratul PHY

Pentru a putea comunica cu alte dispozitive, orice dispozitiv trebuie să aibe definit un strat fizic, precum și un canal de comunicație. Astfel, clasele folosite de YansWifiPhyHelper și YansWifiChannelHelper ajută la definirea acestor concepte importante fără de care nu se poate realiza comunicarea dorită. YANS vine de la ”Yet Another Network Simulator” care este modelul simplificat de helper, pentru o interacțiune mai ușoară și mai rapidă. YansErrorRateModel este un model de transmisie trunchiată a pachetelor de date.

A picture containing text, screenshot, software

Description automatically generated

Figură 5.22 - Setare strat MAC

Fiecărui dispozitiv trebuie să îi fie atribuit și un nivel MAC. Pentru asta există clasa WifiMacHelper. Prin intermediul acestei clase, se poate seta tipul de MAC dorit pentru fiecare dispozitiv Wi-Fi. De exemplu, sunt două tipuri diferite de MAC pentru AP și pentru STA. Pentru AP (prin tipul setat de ApWifiMac), trebuie definit un SSID (Service Set ID), BeaconGeneration care va fi abilitatea dispozitivului de a trimite semnale beacon, precum și conceptul de beacon jitter.

Pentru dispozitivul STA (prin tipul setat de StaWifiMac), se setează SSID-ul la care va încerca conexiunea, precum și conceptul de Active Probing, ce trimite cereri de asociere activ.

După setarea stratului MAC, toată această complexitate se instalează pe nodurile existente din sistem. Instalarea constă în setarea ambelor straturi inferioare (PHY și MAC) pe toate dispozitivele disponibile.

A screen shot of a computer

Description automatically generated with medium confidence

Figură 5.23 - Poziționarea dispozitivelor

Pentru a putea înregistra informații relevante referitoare la timpul de propagare al unui pachet de la un dispozitiv la altul sau nivelul de semnal înregistrat, nivelul de zgomot șamd. trebuie ca dispozitivele să fie distanțate unele de celelalte. Într-un mediu ideal, aceste calcule ar fi mult mai mici, iar într-un sistem simulat oricum vor fi mai mici decât valorile reale ale acelorași calcule. Prin clasa MobilityHelper, ns3 permite setarea unor poziții relative ale dispozitivelor implicate în sistem. Astfel, cu lista de AP-uri preluată din fișier, acestea vor fi poziționate în sistem conform pozițiilor lor definite în aplicația JavaFX. După adăugarea AP-urilor, se va adăuga și dispozitivul STA. Toate dispozitivele sunt setate cu poziții fixe, inclusiv dispozitivul STA.

A picture containing text, screenshot, font

Description automatically generated

Figură 5.24 - Setarea stării dispozitivelor

Toate dispozitivele AP sunt setate pe modul OFF (puterea de transmisie este setată la 0, astfel practic, dispozitivul este dezactivat). Pentru a putea comunica, un STA trebuie să fie asociat cu un singur AP, unul și numai unul, la un moment de timp. Astfel, pentru fiecare AP definit în sistem, se activează dispozitivul (se setează puterea de transmisie la ceva mai mare decât 0), se trimite un pachet la un interval de timp definit, iar după se dezactivează la loc dispozitivul, făcând loc următorului dispozitiv să se asocieze, la un moment din viitor.

A picture containing text, software, screenshot

Description automatically generated

Figură 5.25 - Trace source-ul urmărit

Pentru a putea urmări pachetele ce sunt primite de către dispozitivul STA, trebuie urmărit unul dintre straturile în care se produce comunicare. Astfel, am ales să ascult stratul PHY, prin ruta de urmărire ”MonitorSnifferRx” prin care pot primi diferite detalii privind comunicația dintre un AP și un STA.

În ultima parte sunt definite valorile de simulare ale sistemului.

A screen shot of a computer program

Description automatically generated with low confidence

Figură 5.26 - Crearea de timestamp

Metoda din figura 5.26 este importantă deoarece aceasta creează pachetul propriu-zis ce va fi trimis de la un AP la STA-ul definit. Pentru a putea obține valori dorite, pachetul are nevoie de momentul de timp din simulare din care a fost trimis.

# Testare și validare

Aplicația constă în două module mari, partea de JavaFX în care este dezvoltată interfața grafică și partea de NS3 în care este dezvoltat un program ce va simula algoritmii setați din partea anterioară.

## JavaFX

Partea de JavaFX urmărește interacțiunea utilizatorului cu sistemul, nepermițând avansarea în rulare dacă datele introduse de utilizator nu sunt validate. Astfel, de-a lungul navigării, căsuțele de text pot să își schimbe culoarea în roșu în caz de eroare la achiziția de date, sau mesaje de eroare pot apărea pe interfața grafică semnificând nevoia schimbării inputului oferit.

Aplicația pleacă din starea inițială următoare:

A screenshot of a computer

Description automatically generated

Figură 6.1 - Stare inițială GUI

Utilizatorul poate selecta unul dintre algoritmii disponibili din meniul de dropdown, poate adăuga dispozitive de tipul AP într-o listă ce va fi vizualizată atât în chenarul de deasupra butoanelor cât și în chenarul grafic ce urmează a fi desenat. Pentru a putea avansa în aplicație, utilizatorul trebuie să introducă valori relevante pentru dimensiunea spațiului de simulare. Dacă utilizatorul completează cu caractere ilegale sau nu completează deloc, aplicația va încerca să valideze inputul oferit, răspunzând astfel:

A screenshot of a computer

Description automatically generated

Figură 6.2 - Eroare input grid

În schimb, dacă utilizatorul va introduce valori corecte, aplicația va mări scena de lucru, desenând chenarul grafic în care vor fi vizualizate dispozitivele definite anterior prin butoanele aferente. Utilizatorul va ști că inputul oferit este validat prin colorarea verde a căsuțelor de text pe care le-a completat, precum și prin mărirea scenei aplicației, permițându-i utilizatorului să avanseze în rulare. Aplicația astfel va permite începerea execuției simulării propriu-zise prin lipirea unei scene auxiliare ce conține un spațiu de text, în care vor fi tipărite diferite mesaje de înștiințare, precum numărul dispozitivelor și coordonatele lor, un mesaj de așteptare pentru începerea rulării, precum și outputul primit în urma rulării. Scena va arăta astfel:

A screenshot of a computer

Description automatically generated with medium confidence

Figură 6.3 - Valori corecte grid

Dacă în schimb, utilizatorul a ajuns până în punctul de față fără să seteze un algoritm, utilizatorul va fi întâmpinat cu următorul mesaj de eroare pe ecran:

A screenshot of a computer

Description automatically generated

Figură 6.4 - Eroare algoritm

Aplicația îi va reaminti utilizatorului să aleagă unul dintre algoritmii definiți în aplicație, fără a alege unul anume, aplicația nu poate rula, astfel, starea sistemului neputând fi schimbată până la alegerea unui algoritm.

După alegerea unui algoritm, dacă utilizatorul uită să introducă dispozitive wireless în sistem, sistemul va afișa pe ecran următoarea eroare:

A screenshot of a computer

Description automatically generated

Figură 6.5 - Eroare listă

Astfel, utilizatorului îi va fi reamintit să introducă dispozitive pentru a putea rula simularea algoritmului ales anterior.

Un exemplu de validare completă a programului este următorul:

A screenshot of a game

Description automatically generated with medium confidence

Figură 6.6 - Exemplu validare completă

Se poate verifica totul prin verificarea fișierului de output al aplicației JavaFX în folderul curent. Exemplu fișier output:

A screenshot of a computer

Description automatically generated

Figură 6.7 - Exemplu output

## NS3

Partea de NS3 primește valorile din fișierul de output creat anterior. Fișierul este creat după ce este validat complet, astfel, singura validare care mai trebuie făcută în aplicația NS3 este validarea existenței acestui fișier la rulare. În caz că nu există sau nu poate fi deschis, aplicația se oprește și afișează un mesaj de eroare.

A picture containing text, screenshot, font

Description automatically generated

Figură 6.8 - Verificare fișier

Aplicația simulează execuția unui algoritm de poziționare într-o topologie wireless creată cu ajutorul mai multor dispozitive AP și a unui dispozitiv mobil care se conectează la ele. Un mesaj de output al simulării este redat de următoarea figură:

A screenshot of a computer program

Description automatically generated with medium confidence

Figură 6.9 - Exemplu algoritm ToF

# Manual de instalare si utilizare

## Manual de instalare

### Descărcare și instalare JDK

Link descărcare: https://download.oracle.com/java/20/latest/jdk-20\_windows-x64\_bin.exe

### Descărcare CLion

CLion oferă posibilitatea instalării librăriei ns3 prin CMake, care se ocupă de construirea și de optimizarea librăriei pentru a fi utilizată. Fără CLion, ar trebui să se schimbe sistemul de operare pe Linux, sau să se dezvolte alte posibilități pe alte IDE-uri existente.

Link descărcare CLion: <https://www.jetbrains.com/clion/>

### Descărcare WSL

Pentru a putea instala cum trebuie librăria ns3, pentru a scăpa de necesitatea unui dual-boot sau a unei imagini de Ubuntu într-o mașină virtuală, instalarea se poate realiza direct pe Windows prin WSL.

WSL trebuie să instaleze un distribution de linux pentru a putea rula, recomandarea este Ubuntu 22.04.

După instalare, urmați pașii următori pentru instalarea de dependințe în linux (vor fi folosite mai târziu din CLion) la deschiderea WSL.

Link tutorial instalare WSL:

<https://www.youtube.com/watch?v=FQ6ahcJOVz0&t=9s>

1. **sudo apt-get update**
2. **sudo apt-get upgrade**
3. **sudo apt-get install gcc g++ python python3 python3-dev git**

Link descărcare:

<https://www.microsoft.com/store/productId/9P9TQF7MRM4R>

### Descărcare ns3-3.37

A screenshot of a computer program

Description automatically generated with low confidence

Figură 7.1 - Pași instalare ns3

Pe site-ul principal ns-3 se află toți pașii necesari de descărcare a versiunii dorite de ns3. În cadrul aplicației curente se folosește 3.37.

### Descărcare Intellij

Orice IDE pentru dezvoltarea aplicațiilor java poate fi folosit, însă acesta a fost ales pentru simplitate. Folosit pentru rularea aplicației principale.

Link descărcare: https://www.jetbrains.com/idea/

### Instalare ns3-3.37

Pentru a instala ns3 cum trebuie, avem nevoie de setările corecte în interiorul IDE-ului folosit. Pentru IDE-ul curent (CLion) trebuie respectați următorii pași:

1. Intrare la CLion Settings -> Build, Execution, Deployment -> Toolchain
2. Setare Toolchain nou cu WSL recent instalat
3. Toate utilitarele ar trebui să fie detectate automat din WSL.
4. Apply și OK

A screenshot of a computer

Description automatically generated

1. Figură 7.2 - Toolchain
2. Tabul **CMake**
3. Pe butonul + se adaugă profilele de build dorite (Debug, Release, RelWithDebInfo,etc.).
4. În CMake options:
5. -DCMAKE\_BUILD\_TYPE=Debug -DNS3\_EXAMPLES=ON -DNS3\_TESTS=ON
6. Apply și OK

A screenshot of a computer

Description automatically generated with medium confidence

Figură 7.3 - CMake

1. Selectarea din fișiere a oricărui fișier, după apăsarea pe butonul Build pentru a construi artefactele aplicației.

A screenshot of a computer program

Description automatically generated with medium confidence

Figură 7.4 - Build

## Manual de utilizare

### Pornire aplicația JavaFX

Se deschide folderul ”Simulator” în IDE-ul descărcat. Se rulează aplicația de pe butonul de ”Start”.

A screenshot of a computer program

Description automatically generated with medium confidence

Figură 7.5 - Pornire aplicație

### Exemplu rulare

A screenshot of a game

Description automatically generated with medium confidence

Figură 7.6 - Exemplu rulare

La introducerea datelor necesare (introducere dispozitive, dimensiuni chenar, alegerea algoritmului), pe apăsarea butonului ”Compute”, aplicația va începe rularea simulării din NS3.

La finalul rulării, outputul va fi afișat în chenarul de sub butonul ”Compute”.

# Concluzii

## Contribuții personale

Ca și contribuții personale, au existat mai multe concepte care au trebuit să fie asimilate, dezvoltate și implementate pentru a putea ajunge la un produs final, astfel, contribuțiile personale pot fi identificate prin:

Studierea problemei în detaliu a posibilităților de implementare în cadrul librăriei C++ de simulare a evenimentelor discrete din cadrul unei rețele de calculatoare. Au existat mai multe astfel de librării/aplicații care serveau acestui scop, însă NS3 s-a încadrat cel mai bine pentru simplitate, suportul oferit de către comunitate precum și structura codului destul de ușor de asimilat.

Implementarea unei topologii de rețea wireless în cadrul simulatorului ales, dovedindu-se a fi destul de provocator, în ceea ce privește respectarea standardelor și a protocoalelor de internet fără fir. Aici se numără limitările infrastructurii unei rețele wireless concepută din mai multe dispozitive de tipul Access Point.

Dezvoltarea unei interfețe grafice care să susțină introducerea informațiilor necesare pentru a putea procesa un astfel de algoritm, pentru o interacțiune cât mai facilă și fără prea multe cunoștințe din partea oricărui utilizator doritor să descopere cum operează un algoritm de poziționare într-o rețea Wi-Fi.

## Rezultate obținute

În librăria NS3 au fost implementate cu succes mai multe metode în cadrul aceluiași fișier sursă ce deservește scopului de a simula execuția unor algoritmi de poziționare. Algoritmii sunt sincronizați pentru a opera cu un singur dispozitiv mobil între care se măsoară metrici diferite ale comunicației. Pentru a avea o viziune clară asupra cât de bine funcționează acești algoritmi, rezultatele acestora sunt comparate cu cazurile ideale de comunicație. Cazurile ideale nu țin cont de niciun fel de obstrucție în calea propagării semnalului (precum interferențe, multipath, etc.). Librăria NS3 oferă diferite modele deja implementate de pierderi ale propagării semnalului, de întârzieri, modele de recepție a semnalului, modele de implementare a infrastructurii unei topologii Wi-Fi, etc.

Aplicația dezvoltată poate fi considerată ca un punct de plecare în dezvoltarea unui instrument didactic pentru simularea și măsurarea algoritmilor de poziționare într-un spațiu fictiv, simulat, al căror rezultate pot fi comparate cu rezultatele reale, manuale, ale aceluiași test, în cadrul laboratorului de rețele de calculatoare.

De asemenea, aplicația dezvoltată poate fi folosită de către orice utilizator, prin nivelul ridicat de interactivitate cu interfața grafică. Utilizatorul este ghidat ușor prin sistemul dezvoltat, prin intermediul validărilor multiple de-a lungul cazului principal de utilizare. Acest fapt aduce un nivel ridicat de abstractizare pe parcursul simulării de algoritmi de localizare, aceștia fiind destul de complexi de implementat în limbajele uzuale de programare. Aplicația dezvoltată este în linii mari o ”interfață” a librăriei NS3, în care se introduc date simple, despre spațiu si localizarea dispozitivelor dorite, returnând în urma executării măsurătorile aferente algoritmului selectat.

## Dezvoltări ulterioare

Pentru aplicația curentă, există numeroase dezvoltări ulterioare ce pot fi implementate odată cu timpul. Cele mai notabile dintre acestea ar putea fi măsurarea și afișarea propriu-zisă a poziționării unui dispozitiv mobil în spațiul declarat de testare, testându-se astfel cât de bine localizează algoritmii implementați dispozitivul implicat în testare, analiza și dezvoltarea unor modele proprii de pierderi de semnal sau introducerea de obstrucții fizice în calea propagării semnalului, precum obiecte de mobilier, pereți, materiale, etc.

În cadrul aplicației curente, sunt dezvoltați trei algoritmi de poziționare care pot fi testați. Există numeroși alți algoritmi ce pot fi dezvoltați și implementați în cadrul aceleiași aplicații, însă sunt algoritmi mai complexi, care necesită mai mult timp de dezvoltare și rafinare pentru a putea fi testați.

De asemenea, pot fi introduse elemente de selectare a modelelor de propagare, a modelelor de pierdere, a opțiunilor de setare a dispozitivelor în cadrul unei topologii etc., lucruri care pot spori complexitatea simulării, dar totodată și complexitatea problemei și a interfeței.

O ultimă dezvoltare ulterioară a proiectului poate fi încărcarea acestor metode de analiză a algoritmilor pe dispozitive reale de comunicație fără fir pentru a testa comportamentul unei topologii reale, precum și poziționarea unui dispozitiv real într-un spațiul real.

Bibliografie

|  |  |
| --- | --- |
| [1] | F. Zafari, A. Gkelias, Kin K. Leung, „A survey of Indoor Localization Systems and Technologies,” în *IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL 21, NO.3, THIRD QUARTER*, 2019. |
| [2] | F. Lassabe, P. Canalda, P. Chattonay, F. Spies, „Indoor Wi-Fi positioning techniques and systems,” în *Institut TELECOM and Springer-Verlag*, France, 2009. |
| [3] | E. Weingartner, H. vom Lehn, K. Wehrle, „A performance comparison of recent network simulators,” în *IEEE International Conference on Communications*, 2009. |
| [4] | „Network Simulator 3 Documentation,” [Interactiv]. Available: https://www.nsnam.org/docs/. |
| [5] | „OMNeT++ Documentation,” [Interactiv]. Available: https://doc.omnetpp.org/omnetpp/manual/#sec:introduction:what-is-omnetpp. |
| [6] | „MathLab WLAN Toolbox,” [Interactiv]. Available: https://www.mathworks.com/help/wlan/index.html?s\_tid=CRUX\_lftnav. |
| [7] | T. Jankowski, M. Nikodem, „SMILe - Simulator for Methods of Indoor Localization,” în *International Conference on Indoor Positioning and Indoor Navigation*, Nantes, France, 2018. |
| [8] | Mark Richards, Software Architecture Patterns, Sebastopol, CA: O'Reilly Media, 2015. |
| [9] | „MVC Framework,” Tutorials Point, 2017. [Interactiv]. Available: www.tutorialspoint.com. |
| [10] | Kishori Sharan, Peter Spath, Learn JavaFX 17. Building User Experience and Interfaces with Java, Apress, 2022. |
| [11] | Z. Mammeri, *Introduction to IEEE 802.11 standards,* Toulouse: Institut de Recherche en Informatique de Toulouse. |