

Titolo del progetto svolto

Nome Autore11, Nome Autore21,
1 DISI, University of Bologna, Italy

Emails: autore1@email.it, autore2@email.it, autore3@email.it

Abstract— Riassumi qui il succo del progetto, in 20-30 righe.

I. INTRODUCTION

Descrivi qui il contesto generale, i contributi del progetto, e la struttura del documento.

II. RELATED WORKS

Fornisci una breve rassegna di articoli di ricerca, software, prototipi o tecnologie che sono collegate in qualche modo al problema affrontato nel progetto. Tutti i lavori devono essere referenziati ed inseriti nella Bibliografia.

III. ARCHITECTURE

Il sistema é stato sviluppato in modo da poter girare su dispositivi Android, lasciando in questo modo la possibilit  di essere utilizzato in larga scala e senza costi aggiuntivi al prezzo del dispositivo stesso. Suddetto applicativo costituito da due tipologie di layer applicativi che potremmo denominare Publisher e Observer cos da poter rendere il pi  intuitivo possibile il ruolo da loro ricoperto. Il ruolo del Publisher viene

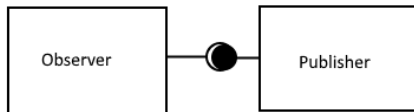


Fig. 1. Ruoli

assunto da tutti i dispositivi in possesso ai pedoni. Il loro compito principale é quello di avvisare i veicoli circostanti riguardo la propria posizione corrente. Tutti i veicoli presenti sulla strada, invece, sono Observer ed hanno l'obbligo di ascoltare in ogni istante i Publisher, elaborare i loro dati ed avvertire il conducente. La struttura della rete risulta dunque essere un miscuglio approssimativamente omogeneo di Publisher ed Observer. La forza di questo sistema il fatto che la rete non preveda una vera e propria struttura, ma che si adatti a quelle che sono le limitazioni dell'ambiente circostante. Questa qualita del sistema lo rende flessibile e particolarmente scalabile.

IV. IMPLEMENTATION

Descrivi come é stato implementato il sistema, ossia tecnologie utilizzate, linguaggi, APIs, etc. Nel caso, fornisci pseudo-codice degli algoritmi piu interessanti sviluppati nel progetto.

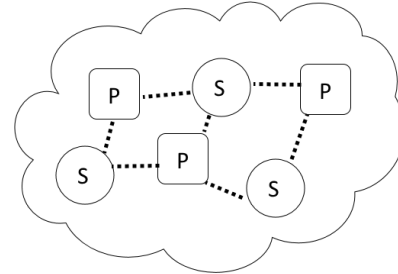


Fig. 2. Rete Omogenea

A. Ambiente di Sviluppo

Come gi  descritto in precedenza, il sistema é stato completamente sviluppato per sistema operativo Android in modo da consentire la maggior portabilit  del software. Infatti il nostro obiettivo non era solo la verifica verticale delle funzionalit  del software, ma anche il confronto orizzontale tra dispositivi che utilizzano hardware differenti. Questo ha consentito di trovare i punti in comune ma anche le discrepanze tra i diversi hardware e di generare un software flessibile in grado di adattarsi al dispositivo host. Per lo sviluppo é stato utilizzato l'IDE di default di Android ovvero Android Studio con linguaggio di programmazione Java.

B. Obiettivi

L'idea iniziale era quella di riuscire ad individuare un pedone con il maggior preavviso possibile, ma limitando il consumo energetico del dispositivo ad egli in possesso. Per fare cio  abbiamo effettuato alcuni test utilizzando il Bluetooth, la tecnologia a radio frequenza con il miglior risparmio energetico. In particolare ci siamo concentrati sull'utilizzo dei Beacon Bluetooth inviati dal dispositivo pedone ed elaborati dal veicolo. Al loro interno sono stati inseriti dati fittizi che simulassero un payload di dimensioni verosimili. Le prove effettuate includevano l'invio dei beacon sia dal veicolo in movimento che dal pedone a velocit  ridotta. Purtroppo in entrambi i casi i risultati ottenuti sono notevolmente scadenti. Infatti i dispositivi riuscivano a trovarsi solo a distanze quasi nulle (non pi  di una decina di metri). La nostra seconda scelta dunque ricadde sull'utilizzo del WiFi Direct(P2P). Nel paragrafo seguente vedremo in breve come funziona questo standard.

C. WiFi Direct e Problemi riscontrati

Wi-Fi Direct (nato con il nome Wi-Fi P2P - Wi-Fi Peer-to-Peer) uno standard della Wi-Fi Alliance, ormai comune in molti prodotti, come smartphone, tablet, stampanti, videocamere digitali, console di gioco e scanner, che consente a due o pi dispositivi certificati di connettersi tra loro con un collegamento Wi Fi diretto senza aver bisogno di un router wireless o di un hotspot. In particolare questa tecnologia prevede 3 fasi prima di poter scambiare dati:

- 1) Effettuare il discovery dei dispositivi circostanti
- 2) Creazione di un gruppo (se questo non esiste già) al fine di poter poi accettare connessione
- 3) Effettuare handshake di connessione

Come racconta lo standard IEEE, il WiFi P2P garantisce una riuscita della connessione all'interno del range dei 13 secondi. Purtroppo, per il nostro applicativo queste latenze sarebbero state enormi ed avrebbero reso il sistema inutilizzabile. Abbiamo dunque deciso di sviluppare un'applicazione di chat base, che ci aiutasse a stimare il reale tempo di riuscita della connessione tra due dispositivi. I test ci hanno riportato quanto segue:

- Tempo di connessione 3/4,
- Una volta stabilita la connessione il tempo di risposta ad un messaggio era pressoché nullo. I 3-4 di connessione non rispecchiavano in tutto per tutto le nostre esigenze in quanto, non certi del range del segnale WiFi, $3s * (50km/h / 3.6) = 40m$ di range sarebbero stati persi in partenza. Non potendo consentire la connessione tra i dispositivi, ci rimaneva unicamente la fase di Discovery da sfruttare. All'interno del record di Discovery, infatti, vi si può inserire qualunque dato sotto forma di tabella hash ed il processo di segnalazione e di ricezione del segnale è pressoché immediato. Fatte queste considerazioni, abbiamo deciso di demolire l'applicazione di chat lasciando unicamente la parte di discovery ed effettuare svariati test in cui misuravamo la distanza minima in cui i due dispositivi riuscivano a vedersi per la prima volta. I risultati sono stati eccezionali. Infatti i valori in metri ottenuti sono stati da un centinaio di metri in campo aperto a 45-50 metri in curva con traiettoria in linea d'aria offuscata da ostacoli. Questo principio è risultato per noi essere le fondamenta vere e proprie del sistema stesso. Sfortunatamente le librerie di android non funzionano molto bene per quanto riguarda il publishing ed il discovery nel WiFi Direct. Infatti se si vuol pubblicare i propri dati sulla rete bisogna anche effettuare un'operazione continua di Discovery, altrimenti nessuno dato viene spedito. Inoltre (considerando che il Discovery chiama una callback ogni volta che trova un dispositivo) la callback del Discovery non segnala lo stesso dispositivo più di una volta se esso è già stato trovato anche se possiede un record differente. Questo ci ha costretti a resettare la procedura del discovery ogni 10 secondi in modo da poter mettere il dispositivo Observer nuovamente in ascolto anche di Publisher già trovati. Per valori di T inferiori ai 7-8 secondi si rischia di mandare in crash il driver del modulo WiFi che, non so se per problemi interni o per verifiche di funzionamento del sistema operativo,

viene disattivato bloccando qualsiasi forma di ricezione e/o trasmissione. L'applicativo lato publisher fornisce all'interno del proprio record di service discovery i seguenti dati che varranno poi discussi nei paragrafi successivi nei relativi modelli. -

- Longitudine
- Latitudine
- Velocità
- Timestamp
- Timestamp della Posizione
- Accuratezza
- Bearing

Lato Observer invece, i dati di tutti i pedoni trovati tramite il discovery vengono salvati (eventualmente aggiornati se già trovati in precedenza), processati ed alla fine del loro ciclo vita eliminati.

D. Modello di Previsione

Questo modello viene implementato lato Observer e si occupa di elaborare i dati ricevuti dai diversi Publisher ed incrociarli con quelli del veicolo che rappresenta. Un modello di previsione prevede come input il tragitto che il veicolo dovrà percorrere sotto forma di Polilinea (vettore di segmenti) ed un gestore di posizionamento che sfrutti il segnale GPS in modo da sapere in ogni istante le coordinate attuali del veicolo. Dati questi input, per ogni Publisher trovato stima se in un tratto della Polilinea di input il veicolo potrebbe andare a collidere con il Publisher. Nel caso lo segnala sulla mappa. La stima di collisione viene effettuata utilizzando il seguente algoritmo.

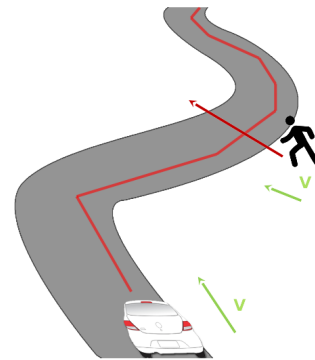


Fig. 3. Modello di Previsione

Il concetto di fondo è che l'uomo collida con il veicolo solo se la sua traiettoria e la sua velocità partendo da un punto A lo porteranno in un punto B tale per cui il segmento AB interseca uno dei tratti della Polilinea realistica al tragitto del veicolo. La seguente funzione verifica se due segmenti si intersecano.

```
/**
 * Check if the human will intersect the
 * vehicle
 * @param h human service Object
 * @return True -> Intersection
 */
```

```

public boolean
willHumanIntersectVehicle(HumanService h)
{
    LatLng loc = getCurrentPosition();
    int start = getCurrentPolylineSegment();
    int end = start;
    // Search indexes on which check
    intersection
    for(int i = start; i < route.size()-1; i++)
    {
        double d =
            getPointSegmentProjectionDistance(
                route.get(i),
                route.get(i+1),
                loc);
        if(d < WIFI_MAX_RANGE)
        {
            end = i+1;
        }else{
            break;
        }
    }
    if(end == start)
    {
        // All segments are inside 200 meters
        range
        end = route.size()-1;
    }

    // Intersection check
    for(int i = start; i < end; i++)
    {
        if(doIntersect(
            route.get(i), // P1 of the vehicle
            route.get(i+1), // P2 of the vehicle
            new LatLng(h.latitude, h.longitude),
            // P1 human
            h.getHumanPositionIn(
                (int) Math.ceil(WIFI_MAX_RANGE /
                    getCurrentSpeed())) // P2 of the
            human
        ))
        {
            return true;
        }
    }
    return false; // No intersections
}

```

Prima di tutto all'interno di questa funzione viene richiesta la posizione corrente del veicolo ed il segmento di polilinea all'interno del quale esso si trova. —Ad esclusione del calcolo della distanza tra due coordinate che viene effettuato in geometria sferica, i le restati operazioni geometriche vengono svolte in geometria Euclidea in quanto la superficie di una sfera per porzioni molto piccole, può essere considerata una superficie piana.— Per trovare il segmento di polilinea corrente (funzione *getCurrentPolylineSegment*), viene effettuato un calcolo trigonometrico della distanza tra la posizione del veicolo ed ogni segmento. Successivamente il tratto con la distanza punto-segmento minore è considerato quello corrente. La distanza punto-segmento in geometria sferica viene calcolata con il seguente algoritmo. Definiamo

con AB il segmento in questione e con C la posizione del veicolo.

- 1) Convertire le coordinate terrestri in coordinate Cartesiane (utilizzando come origine il centro della terra).
- 2) Calcolare T, il punto sulla retta AB più vicino al punto C utilizzando i tre seguenti prodotti vettoriali:

a) $G = A \times B$

b) $F = C \times G$

c) $T = G \times F$

- 3) Normalizzare T a moltiplicarlo per il raggio della terra.
- 4) Riconvertire il valore in latitudine e longitudine.
- 5) Verificare che il punto si trovi esattamente sul segmento AB, nel caso sovrascriverlo con l'estremo del segmento più vicino.
- 6) Utilizzare la formula della distanza tra due punti in geometria sferica per trovare la distanza punto-segmento.

Il primo ciclo for cerca quali sono i segmenti di polilinea il cui punto più vicino del segmento rispetto alla mia posizione risulta essere ad una distanza d inferiore al range del WiFi, definito dalla costante WIFI_MAX_RANGE

V. PERFORMANCE EVALUATION

Illustra qui i risultati sperimentali (simulazioni o esperimenti) che catturano le prestazioni del sistema realizzato. Chiarisci quali sono gli indici di stima e come sono calcolati. Inserisci un breve commento per ogni grafico.

VI. CONCLUSIONI

Conclusioni, possibili sviluppi futuri e limitazioni del progetto realizzato

REFERENCES

- [1] Lista Autori Titolo Lavoro Nome Rivista o Convegno, pagine, anno pubblicazione.