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 piú intuitivo possibile il ruolo da loro ricoperto. Il ruolo del Publisher viene

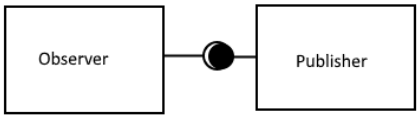

Fig. 1. Ruoli

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 struttra della rete risulta dunque essere un miscuglio approsimatamente omogeneo di Publisher ed Observer. La forza di questo sistema il fatto che la rete non preveda una vera e propria sturuttra, ma che si adatti a quelle che sono le limitazioni dell'ambiente circostante. Questa qualit del sistema lo rende flessibile e particolarmente scalabile.

IV. Implementation

Descrivi come eśtato implementato il sistema, ossia tecnologie utilizzate, linguaggi, APIs, etc. Nel caso, fornisci pseudocodice degli algoritmi piuínteressanti sviluppati nel progetto.

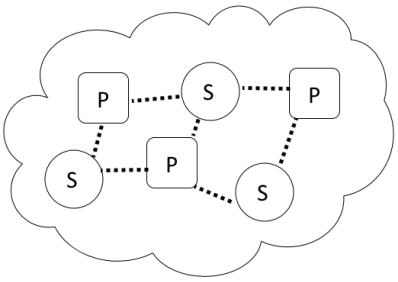

Fig. 2. Rete Omogenea

A. Ambiente di Sviluppo

Come giá descritto in precedenza, il sistema é stato completamente sviluppato per sistema operativo Android in modo da consentire la maggior portabilitá del software. Infatti il nostro obiettivo non era solo la verifica verticale delle funzionalitá 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 eśtato 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 inidividuare un pedone con il maggior preavviso possibile, ma limitando il consumo energetico del dispositivo ad egli in possesso. Per fare cioábbiamo effettuato alcuni test utilizzando il Bluetooth, la tecnologiaá 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 velocitaá fidotta. Purtroppo in entrambi casi i risulati ottenuti sono notevolmente scadenti. Infatti i dispositivi riuscivano a trovarsi solo a distanze quasi nulle (non piuú di una decina di metri). La nostra seconda scelta eú dunque ricaduta 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, video-camere 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 dio poter scambiare dati:
1) Effettuare il discovery dei dispositivi circastanti
2) Creazione di un gruppo (se questo non esiste gia') al fine di poter poi accettare connesione
3) Effettuare handshake di connessione
Come racconta lo standar 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 pressocheńullo. 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 puoínserire qualunque dato sottoforma di tabella hash ed il processo di segnalazione e di ricezione del segnale epressocheímmediato. Fatte queste considerazioni, abbiamo deciso di demolire l'applicazione di chat lasciando unicamente la parte di discovery ed effettuare svariati test in cui misuravamano la distanza minima in cui i due dispositivi riuscivano a vedersi per la prima volta. I riusultati 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 efisultato 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 dispoositivo) la callback del Discovery non segnala lo stesso dispositivi piuďi una volta se esso egiaśtato trovato anche se possiede un record differente. Questo ci ha costretti a resettare la procedura del discovery ogni T secondi in modo da poter mettere il dispositivo Observer nuovamente in ascolto anche di Pubblisher giaťrovati. 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 varro poi discussi nei paragrafici successivi nei relativi modelli. - Longitudine
- Latitudine
- Velocita
- Timestamp
- Timestamp della Posizione
- Accuratezza
- Bearing
Lato Observer invece, i dati di tutti i pedoni trovati tramite il discovery vengono salvati(eventualmente aggiornati se gia 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 dovra percorrere sottofroma di Polilinea (vettore di segmenti) ed un gestore di posizionamento che sfrutti il segnale GPS in modo da sapere in ogni istante le cordinate 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.

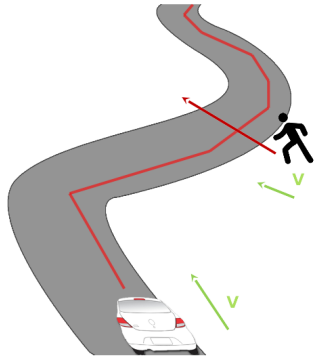

Fig. 3. Modello di Previsione

Il concetto di fondo eće l'uomo collideraćon il veicolo solo se la sua traiettoria e la sua velocitaťpartendo da un punto A lo porteranno in un punto B tale per cui il segmento AB interseziona uno dei tratti della Polilinea realitiva al tragitto del veicolo. La seguente funzione verifica se due segmenti si intersezionano.

```
/**
* 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. 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 ećonsiderato 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 orgine il centro della terra).
2) Calcolare T, il punto sulla retta AB piuv́icono al punto C utilizzando i tre seguenti prodotti vettoriali:

a) G = A x B

b) F = C x G

c) T = G x F

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

Proseguento con l'analisi della funzione *willHumanIntersectVehicle*, il primo ciclo for, partendo dal segmento corrente, cerca quali sono i segmenti di polilinea la cui distanza tra segmento e posizione del veicolo risulta essere inferiore al range del WiFi, definito dalla costante WIFI_MAX_RANGE. Questo perchev́i presume che il veicolo non possa ricevere un segnale piuv́distante del normale range di trasmissione. Una volta terminata questa verifica, avremo un estremo superiore ed un estremo inferiore che limitano il numero di indici dei segmenti sui quali poi verraéseguito il controllo di intersezione con il segmente pedone. Il secondo ciclo for esegue per ogni segmento compreso nell intervallo prima definito il controllo di intersezione con il segmento del pedone. Questo controllo viene eseguito dalla funzione *doIntersect* alla quale vengono passati i due punti del segmento veicolo, la posizione del pedone descritta all'interno del record di Discovery e la stima della posizione del pedone tra J secondi, dove J eív tempo impiegato dal veicolo a percorre il range del WiFi. La nuova posione viene calcolata utilizzando un moto rettilineo uniforme dove il tempo eúguale J + timestamp corrent - timestamp_pos ottenuto dal service discovery del pedone. Timestamp_pos eív'istante in cui eśtata calcolata la posizione del pedone prima di essere inviata. *doIntersect* eív'unica funzione ad utilizzare calcoli euclidei invece che sferici. Questo perchev́a superfice di una sfera per porzioni molto piccole, puoéssere considerata una superficie piana. Il controllo di intersezione viene sviluppato utilizzato il concetto di orientazione tra punti. Due segmenti (p1,q1) e (p2,q2) si intersezionano se e solo se una delle seguenti condizione eśverificata.
1. Caso generale:
(p1, q1, p2) e (p1, q1, q2) hanno orientazioni differenti && (p2, q2, p1) e (p2, q2, q1) hanno orientazioni differenti.
2. Caso speciale:
(p1, q1, p2), (p1, q1, q2), (p2, q2, p1), e (p2, q2, q1) sono collineari && le proiezioni in x di (p1, q1) e (p2, q2) si intersezionano && le proiezioni in y di (p1, q1) e (p2, q2) si intersezionano.
L'insieme di tutti questi algoritmi danno vita al modello di previsione.

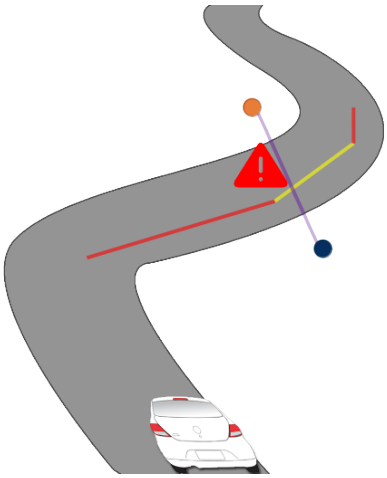

Fig. 4. Intersione tra segmenti

V. Performance evaluation

Sfortunatamente effettuare delle prove con questo sistema non risulta per niente semplice. Questo perchéśarebbe necessario scegliere un'area urbuana con un alto numero di pedoni e veicoli per intravedere tutti i minimi particolari e poter dunque costruire modelli realistici. Detto questo, i nostri test sono stati effettuati in aree rurali (per non intralciare il traffico), con perodiversi ostacoli tra trasmettitore e ricevente. Infatti gli scenarii da noi scelti ed analizzati sono i seguenti.

	Sole		Pioggia	
	$\perp$	//	$\perp$	//
Linea diretta	1	2	3	4
Curva/Ostacoli	9	10	11	12

Velocitá del veicolo $\sim$ *50Km/h*

VI. Conclusioni

Conclusioni, possibili sviluppi futuri e limitazioni del progetto realizzato

References

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