Sistemi Context-Aware, a.a. 2019-2020

Proposte di Progetti

Docente: Prof. Marco Di Felice, marco.difelice3@unibo.it

February 3, 2020

Regole generali

- **COME** Il progetto può essere svolto singolarmente o in gruppi di massimo 3 unità.
- **COME** La consegna avviene attraverso la piattaforma Insegnamenti On Line (IOL) di UNIBO. Il link verrà indicato a breve.
- **QUANDO** Non sono previste deadline di consegna.
- **COSA** Occorre consegnare **i sorgenti** ed una **relazione** (strutturata nelle seguenti sezioni: Introduzione, Progettazione, Implementazione, Risultati), lingua a scelta (inglese preferibile).
- **COSA** A valle della consegna, occorre preparare una **presentazione con slides** per la discussione del progetto: durante la presentazione, verrà chiesto di effettuare una demo del progetto.
- **NOTA** Tutti i membri del gruppo devono essere presenti durante la discussione, e devono conoscere il 100% del progetto svolto.

1 Proposta 1 (Geo-fencing Platform)

Si chiede di sviluppare una piattaforma software context-aware di geo-fencing, ossia di disseminazione di messaggi di testo sulla base della posizione utente. In particolare, il sistema consente l'invio di notifiche ogni qualvolta l'utente entra all'interno di specifiche aree geografiche (geo-fences), individuate nella città di Bologna. La piattaforma include componenti di **location-awareness** ed di **activity-awareness**. Ogni geo-fence è caratterizzato da una tripla:

$$< geometry, message, activity > \quad (1)$$

- *geometry* denota il perimetro del geo-fence, sotto forma di poligono.
- *message* denota un messaggio di testo o un URL di un sito da visitare.

- $activity = \{bike, car, walk\}$ denota il tipo di mobilità utente currente.

La piattaforma è composta da un **client mobile** (app Android/IoS), un **back-end** ed un **front-end** per la visualizzazione dati lato amministratori del servizio. Nello specifico:

- Il client-mobile deve recuperare la posizione utente (GPS), modellare il dato tramite GEO-JSON, ed inviarlo al back-end. Inoltre, il client-mobile classifica il tipo di mobilità corrente, tra le tre classi precedentemente indicate. Sia il dato di posizione sia quello di mobilità vengono periodicamente inviati dal client al back-end.
- Il back-end calcola -mediante query spaziale- l'eventuale geo-fence associato alla posizione/attività utente, e resituisce il messaggio da visualizzare. Inoltre, il back-end si fa carico di salvare la traiettoria dell'utente sul database, come dato spaziale.
- Il client-mobile visualizza il messaggio sotto forma di notifica asincrona. Nel caso in cui il messaggio contenga un URL, visualizzare la risorsa Web corrispondente attraverso una WebView.
- Il front-end Web consente -lato amministratore- la visualizzazione ed analisi delle tracce di mobilità degli utenti. In particolare, devono essere previste le seguenti funzionalità:
  1. Visualizzazione delle traiettorie utenti, per ciascun tipo di mobilità;
  2. Visualizzazione dei confini dei geo-fence, per ciascun tipo di mobilità;
  3. Visualizzazione dei geo-fence, colorati sulla base dell'intensità delle attivazioni (ossia di quanti utenti sono entrati all'interno di quell'area), per ciascun tipo di mobilità;
  4. Visualizzazione dei cluster spaziali, relative alle posizioni di arrivo degli utenti, calcolate mediante l'algoritmo di K-Means;
  5. **OPZIONALE.** Suggerimento sul dove posizionare nuovi geo-fence sulla mappa in modo da massimizzare il numero atteso di attivazioni, per ciascun tipo di attività.

Il messaggio deve essere inviato nel momento in cui l'utente ACCEDE all'area del geo-fence. Non devono essere inviati messaggi ripetuti metre l'utente si muove all'interno del geo-fence.

1.1 File di Supporto

Nella pagina Web del corso, sono forniti i seguenti file di input contenenti i **dati con cui effettuare il popolamento iniziale della piattaforma**:

- file di geofences per ciascun tipo di mobilità (`geofence_car.json`, `geofence_bike.json`, `geofence_walk.json`). Ogni elemento del file contiene un geo-fence (geometria Polygon) ed una property (il messaggio da visualizzare).

- file contente i tragitti degli utenti per ciascun tipo di mobilità (`path_car.json`, `path_bike.json`, `path_walk.json`).

Si assume che l'insieme dei geo-fence della piattaforma sia predeterminato in fase di set-up e non vari nel tempo.

1.2 Componenti aggiuntive

Lo sviluppo del riconoscimento del tipo di mobilità utente (activity-awareness), così come il supporto lato front-end per le tre classi di mobilità è OPZIONALE se il progetto viene svolto singolarmente: in questo caso, si può considerare solo la mobilità `walk`. Vice versa, lo sviluppo di tali componenti è OBBLIGATORIO nel caso di sviluppo di progetti di gruppo. Possono inoltre essere considerati i seguenti add-on (OPZIONALI, ma tenuti in considerazione in fase di valutazione):

- +1 punto. **Valutazione delle prestazioni**: quanto è accurato il meccanismo di notifica, rispetto alla posizione corrente dell'utente ed alla mobilità? Quanto è accurato il rilevamento della mobilità corrente? Qual è il ritardo della notifica (ossia il tempo medio che intercorre da quando l'utente accede realmente al geo-fence a quando riceve la notifica)?
- +.5 punti. Meccanismi di **ottimizzazione energetica** lato client: ogni quanto campionare il GPS ed interrogare la piattaforma?
- +.5 punti. **Privacy** ed anonimizzazione dei dati spaziali (solo per i dati raccolti dall'app, non per quelli caricati dai file).

1.3 Tecnologie da utilizzare

Vincoli sull'implementazione:

- App mobile: sistema operativo a scelta dell'utente. *L'interfaccia dell'app può essere minimale e non è oggetto di valutazione.*
- Back-end: utilizzare POSTGRES/POSTGIS e query spaziali. Linguaggio back-end a scelta dell'utente.
- Front-end: sviluppo di interfaccia Web mediante OpenLayers, oppure sviluppo di un progetto QGIS con vari livelli di visualizzazione, oppure modalità ibrida (usare plugin `qgis2web`).
- Il riconoscimento della mobilità utente (activity-awareness) può avvenire mediante funzioni di libreria (es. API Android o CMMotionActivity in ambiente iOS) oppure mediante lo sviluppo di un modello di classificazione ad-hoc (*BONUS per ottenere la lode*).

E' possibile utilizzare un qualsiasi linguaggio di programmazione o librerie a scelta dello studente.

2 Proposta 2 (Sistema di Multi-room audio)

Si chiede di sviluppare una piattaforma context-aware di multi-room audio streaming, ossia di riproduzione di file audio musicali nell'ambiente indoor in cui si trova l'utente. La piattaforma include funzionalità di **location-awareness** e di **neighbor-awareness**. Il sistema è composto da tre componenti logiche:

- dispositivo utente (smartphone o PC);
- riproduttore audio (RA): uno per ogni stanza, può essere costitituito da hardware dedicato (speaker) o da un PC;
- server di localizzazione ed adattamento dei contenuti (SLAC).

In particolare, il sistema deve essere in grado di:

- localizzare l'utente in un ambiente indoor, riconoscendo la stanza nella quale si trova;
- attivare la riproduzione dell'audio sul RA collocato nella stanza corrente dove si trova l'utente;
- gestire la migrazione del servizio, sulla base della mobilità utente.

Tramite il dispositivo Utente, l'utente seleziona il file audio che vuole riprodurre, tra quelli già presenti nello SLAC. Inoltre, ad intervalli regolari, il dispositivo campiona costantemente i dati degli Access Point Wi-Fi presenti nell'ambiente (es. SSID e potenza ricevuta), e trasferisce le feature corrispondenti allo SLAC. Lo SLAC effettua la localizzazione indoor (riconoscimento della stanza) utilizzando tecniche di triangolazione o di radio-fingerprinting, e restituisce il risultato sul dispositivo utente. Sulla base della posizione indoor corrente, lo SLAC determina il dispositivo RA che deve avviare la riproduzione, ed avvia lo streaming audio. Nel caso venga rilevato un cambio di stanza, lo SLAC provvede ad interrompere lo streaming corrente ed a re-indirizzarlo (possibilmente partendo dall'istante attuale di riproduzione) verso il nuovo RA.

2.1 Componenti aggiuntive

- +1 punto. **Valutazione delle prestazioni**: quanto è accurato il meccanismo di localizzazione indoor? quanto dura l'intervallo necessario per il processo di localizzazione?
- +1 punto. **Comunicazione D2D**: rimuovere la componente SLAC, implementando la localizzazione solo a livello di dispositivo utente. Il dispositivo utente ed i vari RA devono far parte di un gruppo Wi-Fi Direct.

2.2 Tecnologie da utilizzare

Vincoli sull'implementazione:

- App mobile: sistema operativo a scelta dell'utente. *L'interfaccia dell'app può essere minimale e non è oggetto di valutazione.*

E' possibile utilizzare un qualsiasi linguaggio di programmazione o librerie a scelta dello studente.

3 Proposta 3 (Privacy-enabled Location Based Service)

Si vuole realizzare una piattaforma Location-Based Service (LBS) che fornisca informazioni spaziali relativi alla presenza di parcheggi auto prossimi all'utente, **con meccanismi di gestione della privacy**. In particolare, il LBS riceve una coordinata GPS (esatta, o un'area di riferimento), e restituisce il Punto di Interesse (PoI) più prossimo alla posizione fornita in input. Il focus del progetto è sull'implementazione di meccanismi di privacy del dato spaziale, per ridurre il rischio legato al tracciamento della traiettoria dell'utente da parte del LBS. Nello specifico, si prevedono tre componenti:

- **Dispositivo utente** (possibilmente uno smartphone): recupera la posizione GPS e la invia attraverso un canale cifrato ad un server trusted, insieme all' id dell'utente. Riceve dal LBS le informazioni relative al parcheggio più vicino, visualizzandolo sulla mappa.
- **Server trusted**: implementa meccanismi di location-privacy (vedi sotto) ed anonimizza la richiesta dell'utente, sostituendo l'id con uno psuedonimo. Si assume la disponibilità di un numero finito di pseudonimi, pari a $K < 5$. La nuova richiesta è inoltrata al LBS.
- **Location Based Service (LBS)**: riceve una richiesta $< id, GPS >$ e resitituisce la posizione del parcheggio più vicino a quella fornita in input. Il LBS memorizza su DB interno le richieste ricevute sulla piattaforma, ed in base ad esse prova a individuare la reale posizione dell'utente (o meglio, l'associazione pseudonomo-posizione), ad esempio mediante clustering spaziale. Ovviamente, si assume che il LBS non conosca la tecnica di location privacy adottata dal surver trusted. Il LBS dispone anche di una dashboard Web attraverso la quale si visualizzano le posizioni GPS ricevute dall'utente.

Deve essere prevista una fase di valutazione delle prestazioni, considerando tre metriche:

- **Accuratezza**, calcolato come percentuale di casi in cui il parcheggio restituito dal LBS coincide con quello effettivamente più vicino alla posizione REALE dell'utente.

- **Overhead Distanza**, calcolata come differenza media tra la posizione del parcheggio più vicino alla posizione REALE dell'utente e quello restituito all'utente.
- **Privacy Distanza**, calcolata come differenza media tra il valore reale dell'utente, e quella stimata dal LBS.

Oltre al dispositivo reale, è utile (sia per l'implementazione del server trusted, sia per l'analisi di prestazioni) prevedere la presenza di dispositivi simulati, che si muovono all'interno dello scenario.

4 Tecnologie da utilizzare

Vincoli sull'implementazione:

- E' possibile implementare un qualsiasi meccanismo di location privacy (o anche più di uno) tra quelli visti a lezione: dummy updates, spatial cloaking, mix-zones, etc
- I dati dei parcheggi sono forniti sulla pagina Web del corso (`parcheggi.txt`), e devono essere salvati su un database POSTGIS; il calcolo del POI deve essere effettuato mediante una query spaziale.
- La dashboard Web deve essere sviluppata con OpenLayers.
- App mobile: sistema operativo a scelta dell'utente. *L'interfaccia dell'app può essere minimale e non è oggetto di valutazione.*

E' possibile utilizzare un qualsiasi linguaggio di programmazione o librerie a scelta dello studente.