Report - Secondo assegnamento Ingegneria del Software

Vincenzo Arceri - VR360465
Matteo Calabria - VR363871
Pietro Musoni - VR359914
Carlo Tacchella - VR362194

Anno accademico 2013/2014

1 Specifiche del progetto

L' *Automotive System* è un esempio di sistema per connettere tramite rete wireless, automobili su un tratto stradale ad una vase centrale per il controllo delle velocità e del traffico. Il sistema è composto da tre nodi principali: i channel, la base station e le automobili. La base station invia i pacchetti, per la comunicazione, in broadcast a tutte le auto sul tratto coperto dalla rete, queste rispondono direttamente alla stazione e qualora ci fossero posti disponibili sui canali controllati dalla stazione, vengono registrate su uno di essi. I canali possono essere al massimo cinque e hanno un limite di packet rate uguale a 100 pps. Una volta raggiunto il limite di auto connesse il canale non può ospitarne altre e sarà creato un ulteriore canale fino al raggiungimento del limite consentito (cinque canali). Lo scenario è quindi il seguente:

- la *base station* gestisce il sistema di registrazione delle auto e comunica in broadcast con le auto connesse al sistema per il controllo della velocità;
- i *channel* collegano la automobili registrate con la base station e controllano il flusso di informazioni che queste inviano alla stazione;
- le *automobili* possono essere di due tipi (automatiche o manuali) e in base al loro tipo reagiscono in modi differenti ai messaggi della base station per quanto riguarda l'adeguamento dei limiti di velocità.

2 Scelte progettuali

Il diagramma delle classi riporta lo schema del progetto, mostrando graficamente le scelte implementative, le assunzioni ed i pattern utilizzati (Figura 1). Il sistema è principalmente composto da tre nodi che comunicano tra loro (auto, stazione e canali) come detto in precedenza. La stazione centrale ha un'interfaccia **FactoryChannel** per la creazione dei canali e una **Subject** per l'invio di pacchetti a tutte le auto tramite il metodo **update** dell'interfaccia **Observer** (come specificato dal omonimo design pattern).

Ogni auto è estesa da una classe **Adapter** per adattare le automobili manuali ad automibili automatiche, cambiando il metodo update che nel secondo caso, invece di mostrare semplicemente un messaggio su display, chiama automaticamente il metodo **break** che fa frenare l'auto in questione. Ogni auto viene creata da una classe **FactoryCar**. Se un'auto riesce a registrarsi su un canale, comunica tramite quest'ultimo con la base station, mentre tutti i pacchetti che partono dalla stazione vengono inviati direttamente alle auto in broadcast. I pacchetti sono definiti dalla classe **Packet**, che ne descrive i campi.

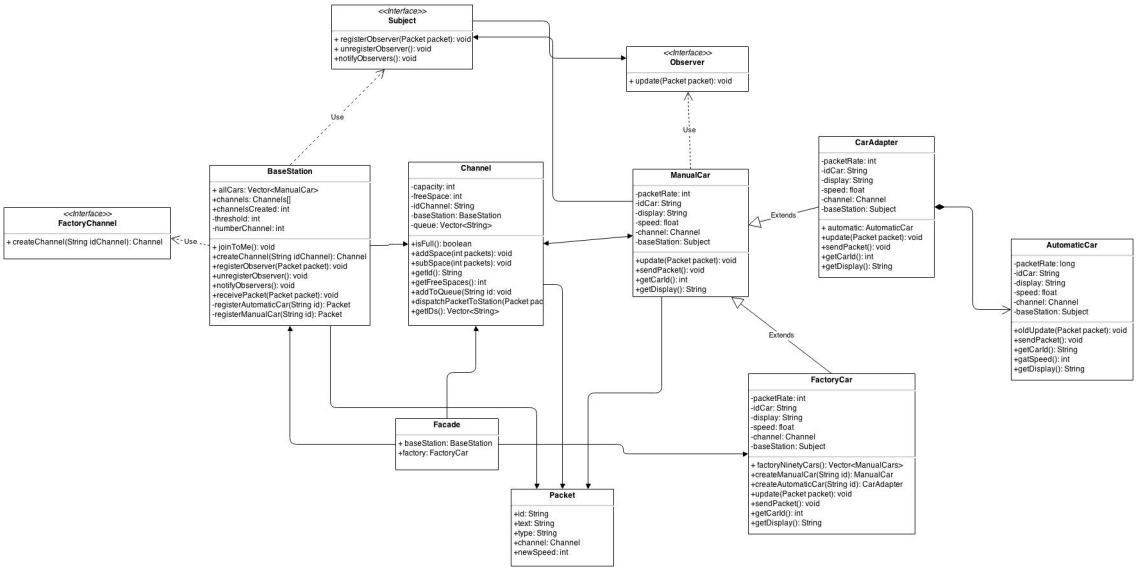

Figura 1: Diagramma delle classi

L'invio di messsaggi da parte della Base Station a una singola auto inviato tramite oggetti Packet in broadcast a tutte le auto del sistema. Tramite il campo Id l'auto a cui è rivolto il pacchetto riceve il messaggio (tramite il campo text) e agisce di conseguenza, tutte le auto con id diverso ignorano il pacchetto.

La classe Facade crea la base station e il factory car per inizializzare il sistema, il facade viene creato invece dal main di Simulator che implementa anche la parte grafica composta da due finestre. Una tabella presenta la velocità, il display e l'id di tutte le automobili registrate ai canali mentre l'altra finestra segnala lo stato di ogni canale scrivendo la percentuale di occupazione. Dato che ogni canale riesce a reggere al massimo un packet rate di 100 pps, se si registra un auto manuale vengono occupati 5 pps, se se ne registra una automatica se ne occupano 10 (come da specifiche). Se un'auto si deregistra vengono liberati i rispettivi pps dal flusso del canale permettendo così la registrazione di nuove vetture al canale. Quando un'auto invia la velocità alla base station, sul display viene scritto che l'auto in questione sta aspettando una risposta.

3 Design pattern utilizzati

I pattern implementati richiesti dalle specifiche sono: Facade e Adapter. Il primo è utilizzato per visualizzare lo stato di tutti i nodi, usati per l'interfaccia con

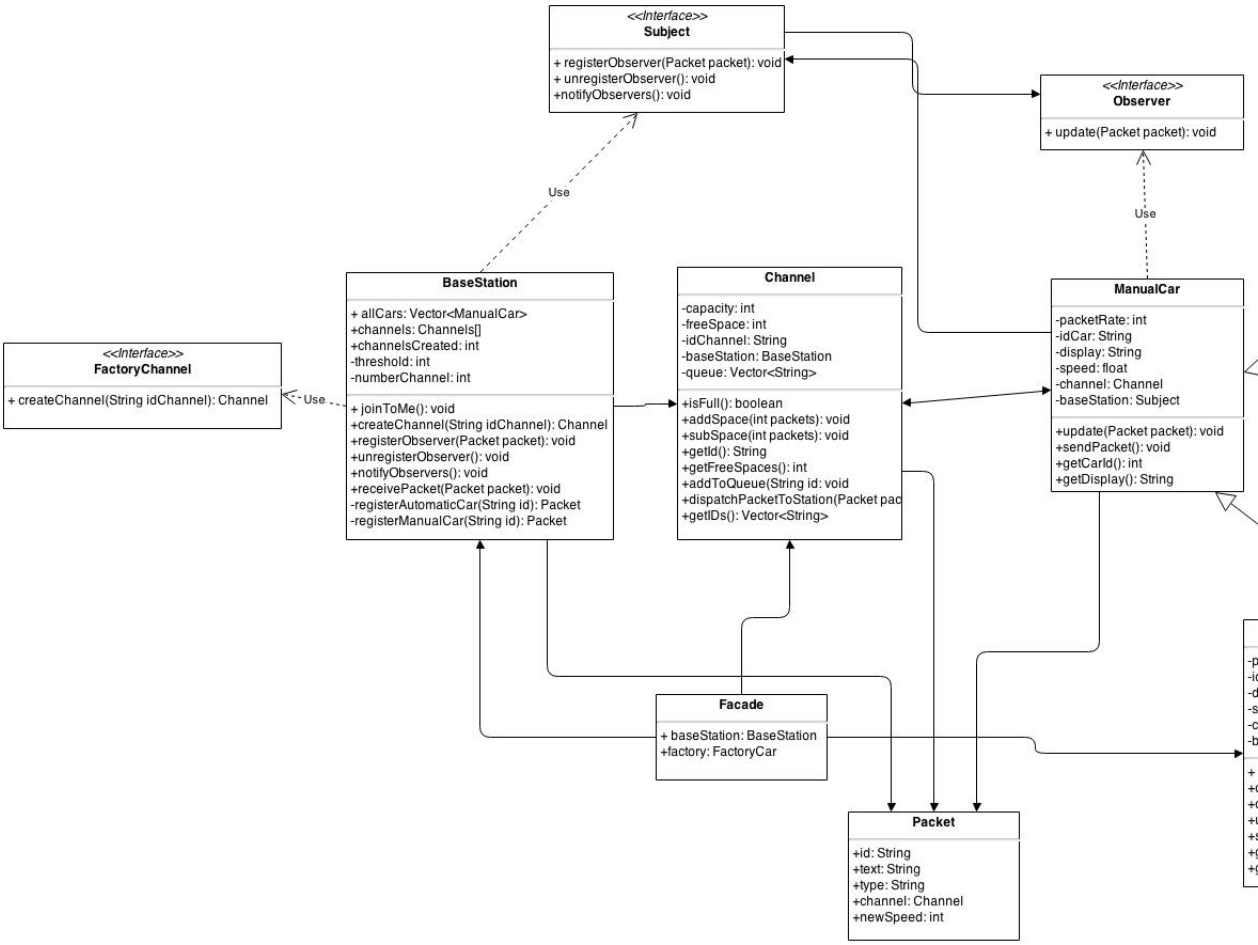

Figura 2: Screenshot dell'applicazione

l'utente. Il pattern Adapter viene utilizzato per adattare le auto automatiche in auto manuali, poiché differiscono solamente nel metodo update (come visto in precedenza, per le automatiche si chiama il metodo break in caso di messaggio di rallentamento mentre le altre stampano solo un messaggio sul display).

I design pattern che abbiamo deciso di implementare sono: Observer e Factory. Il modello Observer permette la comunicazione diretta tra la base station e le automobili, inizialmente in entrambe le direzioni per gestire la registrazione delle auto sul canale, in seguito a una direzione ovvero dalla base station alle auto per l'invio in broadcast dei pacchetti di comunicazione (canale pieno, rallenta etc..).

Il primo scenario in cui viene applicato il pattern factory è per creare le automobili che possono essere manuali o automatiche. Il secondo scenario in cui è utilizzato è per la creazione dei canali da parte della stazione centrale.