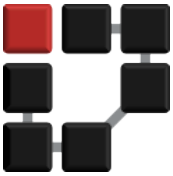
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT



Übung zur Vorlesung Fahrzeugkommunikation

Moritz Gütlein

Bilden Sie für diese Aufgabe **Zweier**-Gruppen.

Aufgabe 5: Netzwerksimulation mit 802.11p/WAVE (Abgabe 08.07. / 10.07.2019)

1. Für das realistischere Simulieren der unteren Layer werden Sie das von uns gepatchte MiXiM-Framework für OMNeT++ verwenden, welches Teil des Veins-Frameworks ist. Laden Sie hierzu das Framework von der Übungswebseite herunter.
   Der veins-Ordner wird jetzt in OMNeT++ über *File - Import - General - Existing Projects into Workspace* hinzugefügt. Nach erfolgreicher Kompilierung soll das 80211p-Beispiel als Test gestartet werden: In diesem Beispiel bewegen sich 5 Knoten mit konstanter Geschwindigkeit und senden periodisch Nachrichten.

   Spätestens in dieser Aufgabe lohnt es sich definitiv mit dem eingebauten Debugger zu arbeiten ;-). Zum Debuggen kann es hilfreich sein, die Simulation ohne graphische Oberfläche zu starten (`Cmdenv`).

2. In dieser Aufgabe soll das 80211p-Example verwendet werden (*examples/80211p*). Die Beschreibung eines Knotens findet sich in `veins/src/nodes/Wifi.ned`. Es soll jetzt ein Nachbarschaftserkennungsprotokoll programmiert werden, welches über die 80211p-WLAN-Schnittstelle funktioniert. Der momentane Applayer ist eine *SimpleWaveApp*, dessen Implementierung Sie in `veins/modules/application/ieee80211p` finden. In der vorliegenden Konfiguration sendet jeder Knoten 10 mal pro Sekunde eine Nachricht. Das Protokoll soll dahingehend geändert werden, dass Knoten in regelmäßigen Intervallen von einer Sekunde (aber zu einem unterschiedlichen Startzeitpunkt, um Kollisionen zu vermeiden) eine Anfrage an alle Nachbarn senden. Erhält ein Knoten eine Anfrage, soll er eine zufällige Zeit abwarten (max. 10 ms), bis er auf die Anfrage antwortet. Überlegen Sie welche Nachrichtentypen (BSM?) für welche Schritte geeignet sind.
   Verändern Sie die *SimpleWaveApp* und implementieren Sie dort das Protokoll, indem Knoten auf dem Control Channel Daten hin- und hersenden. Prüfen Sie, welche Funktionen die *SimpleWaveApp* und auch die Oberklasse (*BaseWaveApplLayer*) bereits implementiert. Der MAC-Layer akzeptiert Nachrichten vom Typ *BaseFrame1609_4*. Das Format des *BaseFrame1609_4* darf angepasst und vererbt werden, wenn nötig. Zeichnen Sie Statistiken über das Delay vom Versenden der Nachricht bis zum Erhalten der Antwort auf. Achten Sie darauf, dass nur Latenzen von Nachrichten aufgezeichnet werden, die tatsächlich Antworten auf selbst gesendete Pakete sind. Erweitern Sie das Nachrichtenformat um eventuell benötigte Felder.

3. Erhöhen Sie als nächstes die Anzahl der Knoten auf 100 und vergrößern Sie den Playground. Die Knoten sollen sich zufällig bewegen. Verwenden Sie hierfür, wie schon im Beispiel, das *LinearMobility* Modul und parametrieren Sie die Startposition `omnetpp.ini` mit `-1`. Geschwindigkeiten können mit Zufallszahlen aus einer Verteilung parametriert werden.

4. Parametrieren Sie die Funkmodule so, dass Knoten eine genügend hohe Sendereichweite haben, aber kein vollständiger Graph entsteht. Stellen Sie hier die Sensitivität

und/oder die Sendeleistung der Knoten in der *omnetpp.ini* entsprechend ein. (Hinweis zum Debugging: Im `calcInterfDist()` des *ConnectionManagers* wird die maximale Interferenzdistanz laut FreeSpace-Model berechnet.)

5. Programmieren Sie jetzt, dass Position und Geschwindigkeit (wird als Vektor zurückgegeben) versendet werden. Dazu müssen Sie im App-Layer über Signale auf das Mobility-Modul zugreifen (siehe *BaseWaveApplLayer*). Jeder Knoten soll diese Nachricht periodisch jede Sekunde versenden. Fügen Sie einen kleinen zufälligen Offset hinzu um Kollisionen auf dem Kanal zu verringern. Knoten, die Nachrichten empfangen, sollen eine Nachbarschaftstabelle mit den Daten der empfangenen Nachrichten verwalten. Knoten, von denen mehr als 5 Sekunden keine Nachricht empfangen wurde, sollen aus der Liste entfernt werden. Das Antworten auf Nachbarschaftsanfragen kann jetzt auskommentiert werden.

6. Eine Simulation von etwa 200 Sekunden soll ausgewertet werden. Wie viele Nachbarn hatte ein Knoten im Durchschnitt? Nutzen Sie dafür die `cOutVector`-Klasse und die `record`-Methode. Am Ende der Simulation soll jeder Knoten außerdem einmalig protokollieren, wie viele Nachrichten er empfangen hat. Nutzen Sie dafür wieder `recordScalar`. Die Ergebnisse sollen erneut visualisiert werden.

7. Diskutieren Sie die Eignung dieser Netzwerksimulation für die Bewertung von Kommunikation zwischen Fahrzeugen untereinander und danach mit dem Betreuer.