Parallel Implementation of the OMNeT++ INET Framework for V2X Communications

Ioannis Mavromatis, Andrea Tassi, Robert J. Piechocki, and Andrew Nix
Department of Electrical and Electronic Engineering, University of Bristol, UK
Emails: {Ioan.Mavromatis, A.Tassi, R.J.Piechocki, Andy.Nix}@bristol.ac.uk

***Abstract*—The field of parallel network simulation frameworks is evolving at a great pace. That is also because of the growth of Intelligent Transportation Systems (ITS) and the necessity for cost-effective large-scale trials. In this contribution, we will focus on the INET Framework and how we re-factor its single-thread code to make it run in a multi-thread fashion. Our parallel version of the INET Framework can significantly reduce the computation time in city-scale scenarios, and it is completely transparent to the user. When tested in different configurations, our version of INET ensures a reduction in the computation time of up to 43%.**

***Index Terms*—Vehicular Communications, INET Framework, Omnet++, Full-Stack Simulations, IEEE 802.11p, CAV, DSRC.**

I. Introduction and Motivation

Recently, more and more resources are being allocated to developing of sophisticated communication frameworks for the next-generation Intelligent Transportation Systems (ITSs). As such, the issue of running cost-effective city-scale experimentation is often addressed by means of simulation frameworks [1], [2]. Unfortunately, existing simulation frameworks (i.e., ns-2, INET, etc.) execute in a single-thread fashion. Thus, in city-scale scenarios, one minute of simulation can easily result in days of computation.

Researchers in the area of Parallel Discrete Event Simulations (PDES) have tried to leverage from new high-performance computing platforms for a very long time. Unfortunately, this resulted in parallel features that require a quite laborious reconfiguration of the existing scenarios, without always assuring improvement in the simulation time [3]. For these reasons, existing simulation frameworks are mainly operated in a single-thread fashion. The necessity for parallel models, motivated us to investigate the existing frameworks and exploit ways of parallelizing them. In this work, we investigate the INET Framework [4] that is an open-source library for OMNeT++ simulation environment [3]. INET is one of the most well-known tools for vehicular simulations being a full-stack network simulation framework. In this poster, we will identify the sequential functions of INET related to the exchange of wireless packets and analyze the way they can be parallelized. Then, we will present our multi-threaded version of the aforementioned functions. Our multi-thread implementation of the INET Framework can be downloaded from `https://github.com/v2x-dev/multithread-inet`.

II. System Analysis and Proposed Solution

We identify at first the way INET interprets obstacles and propagated signals. In a vehicular scenario, buildings are the main obstacles that can be found in a city. INET refers to them as obstacles and parses them within the *PhysicalEnvironment* namespace. All the obstacles are listed in an XML file with an attribute *type*, which defines their shape. Each obstacle is represented by a set of coordinates, that are regarded as the edges of the building in the 3D space. Each building is also associated with a specific *material* that is being used to calculate the attenuation loss caused by the obstacle. The obstacle loss in INET is calculated by two different models, the *IdealObstacleLoss* and the *DielectricObstacleLoss*. The first determines either the signal as completely blocked, or not attenuated when intersected with a physical object. The latter computes the power loss based on the material properties, the shape, the position and the orientation of an obstacle.

INET treats the positions of each vehicle as a point on the simulation canvas and updates them by using the SUMO traffic generator [5]. The signal propagation is modeled as a line segment, between point A and point B. The signal attenuation is a function of: 1) the path loss model, 2) the obstacle loss model from the number of intersections calculated as mentioned before. Both models are configured by the user. What is really of interest, is the way INET calculates the attenuation due to the wall intersections. When a packet is transmitted, INET finds in a sequential and iterative manner all the intersections with the obstacles. For the given intersections, it calculates the obstacle loss using the function *visit* of the obstacles classes mentioned above. Considering the above and that IEEE 802.11p operates in broadcast mode, it is evident that the above process is computationally expensive. In fact, the computation of the attenuation loss has a computational complexity of $O(mn^2)$, where n is the vehicles and m the number of obstacle intersections. This significantly increases the simulation time in large-scale scenarios.

In order to overcome the aforementioned problem, we developed a multi-thread version of the *PhysicalEnvironment* class by modifying the function *visitObjects*. This function is responsible for parsing all the obstacles and finding the power attenuation. Also, we modified *visit* in *DielectricObstacleLoss*, to ensure flawless operation. In our version of INET, the number of threads can be dynamically

Table I
LIST OF SIMULATION PARAMETERS.

Parameter	Value	Parameter	Value
Simulation time	100 s	Carrier Frequency	5.9 GHz
TX Power	25 dBm	Channel Bandwidth	10 MHz
TX/RX Antenna Gain	9 dBi	Message Length	140 B
RX Sensitivity	−93 dBm	Pathloss Exponent	2.4
Cable/System Loss	3 dB	Distance Boundary	1000 m
Transmission Interval	0.1 s		

changed by the user when initializing a scenario.

In city-scale scenario (namely, maps greater than ≥ 2 km), vehicles are not expected to communicate from one side of the city to the other. Despite this, INET always computes the intersection map between each pair of vehicles and their signal attenuation, regardless of the distance between them. In order to speed up the execution time even further, we integrate the notion of the transmission radius in the system. As such, we introduced the *distanceBoundary* user parameter, within the *ScalarAnalogueModel* class, under the *RadioMedium* namespace. For all the exchanged packets, we find the distance between the two communicating vehicles, and if it is greater than the given boundary, we regard the packet as non-deliverable. By that, we can avoid unnecessary calculations in the *PhysicalEnvironment* model. Of course, the above improvement can be implemented into other analog models as well (e.g., *DimensionalAnalogModel*) but we chose the scalar one as a proof of concept.

III. PERFORMANCE EVALUATION AND DISCUSSION

We evaluate the performance of our implementation with two large-scale scenarios in a grid-like fashion. The simulation parameters are as shown in Table I. At first, we evaluate the execution time as a function of the number of vehicles for a map of size 2 km^2. Then, we present the execution time as a function of the map size. We consider six map sizes $\{800, 1100, 1400, 1700, 2000, 2300\}$ m^2 and 100 vehicles for each scenario. All scenarios have roads equally spread horizontally and vertically every 100 m, without traffic lights at the intersections. Each road is 2-lanes wide. Within each road square, we generated buildings with sides of 950 m that act as obstacles in our scenarios. Finally, for all scenarios, we generated the vehicle traffic by using SUMO [5].

Fig. 1 shows the execution time measured as a function of the number of vehicles. Any multi-threaded execution generates some computational overhead in order to create the threads and handle the content-switch. To that extent, we compared two different number of threads (4 and 10). Overall, by parallelizing the functions mentioned above, our version of INET ensures reduced computation times. When four threads are used, we observe an improvement of up to 30%. When ten threads are utilized, the improvement ranges between 10% and 12%. The increased overhead of the number of threads in the second case is the reason for this difference in the performance. We observed that the optimum number of threads is scenario dependent and is related to the average number of intersections between the communicating vehicles (one thread per intersection).

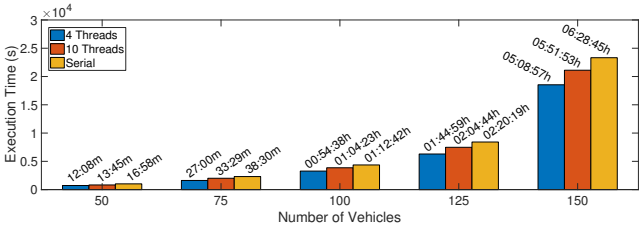

Figure 1. The execution time, measured as a function of the number of vehicles for the parallel and the sequential implementation.

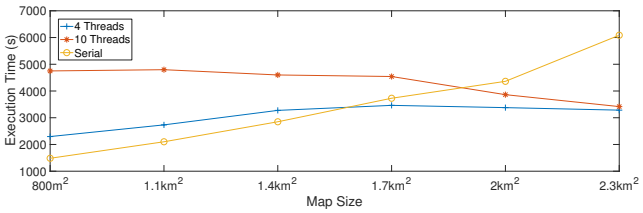

Figure 2. The execution time, measured as a function of the map size for the parallel and the sequential implementation.

Fig 2 shows the execution time required, as a function of the map size, for 100 vehicles. The synthetic maps we generated have a relatively small number of obstacles compared to a real city. We observe that for small maps (from 800 m^2 to 1.4 km^2) the sequential INET achieves slightly better performance compared to the parallel one. This is caused by the multi-thread overhead. However, for larger maps, we observe that our multi-thread version of INET outperforms the sequential one by reducing the computational time of 43%, for a 2.3 km^2 map. Again, when we increase the number of threads, the increased overhead leads to increased simulation time compared to the 4-thread scenario, but still manages to outperform the sequential execution. Finally what we observe is that for maps ≥ 1.7 km^2, as the size of the map increases, the execution time decreases. This is because of the distance boundary that we introduced. For a fixed number of vehicles that are equally spread on the surface of the map, the distance between them will be greater when the size of the map is increased.

IV. CONCLUSIONS

In this work, we investigated the bottleneck of the INET Framework and proposed an optimized multi-thread refactoring of its code. With our solution, the computation time can be decreased by up to 43% compared to the single-thread version. Our multi-threaded implementation ensures the seamless integration with the existing simulation scenarios of a user and is easily configurable to speedup the simulation time when required.

REFERENCES

[1] I. Mavromatis, A. Tassi, R. J. Piechocki, and A. Nix, "Agile Calibration Process of Full-Stack Simulation Frameworks for V2X Communications," in *Proc. of IEEE VNC 2017*, Nov. 2017.

[2] I. Mavromatis, A. Tassi, G. Rigazzi, R. J. Piechocki, and A. Nix, "Multi-Radio 5G Architecture for Connected and Autonomous Vehicles: Application and Design Insights," *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, vol. 4, no. 13, 3 2018.

[3] A. Varga and R. Hornig, "An Overview of the OMNeT++ Simulation Environment," in *Proc. of ICST*, Mar. 2008.

[4] "INET Framework," https://inet.omnetpp.org/, 2018.

[5] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *Int. J. On Adv. in Syst. and Measurements*, vol. 5, no. 3-4, Dec. 2012.