

Spatial Computing Meets Realistic Mobile Wireless Problems

Bernát Wiandt*, Vilmos Simon*, András Kőkuti*, Jacob Beal†,

*Inter-University Centre for Telecommunications and Informatics

*Budapest University of Technology and Economics, Hungary

†Raytheon BBN Technologies, USA

Abstract—Controlling and leveraging the vast and ever-growing number of wireless devices around us has become a serious problem. Spatial computing offers a promising approach toward solving this problem, in the form of higher-level distributed programming abstractions. Many challenging mobile communications problems, however, can only be investigated using a realistic and detailed model of wireless communication. We thus demonstrate a new research platform, integrating the MIT Proto spatial computing suite with the OMNeT++ network simulator framework, which will enable investigation of spatial computing solutions to difficult mobile wireless problems.

I. MOTIVATION

We are surrounded by wireless devices, but it is difficult to design applications to take advantage of them. Spatial computing is an approach for developing higher-level abstractions for programming such devices, by leveraging the locality of wireless communications. The Proto [1] language offers a general approach to spatial computing, but the communication model that has been used by the MIT Proto [2] simulator is far too simplistic for many mobile *ad hoc* networking problems.

The problem is that MIT Proto defaults to a unit disc model of wireless communication. In this model two devices can communicate if and only if they are less than a certain distance apart. Real world wireless communication is a much more complicated matter. There are effects, such as, fading, interference, variable path losses, different radio output powers, etc., that strongly affect connectivity, particularly in complex environments such as urban areas. How well will the Proto approach hold up when subjected to such conditions?

The first step towards answering this question is to link MIT Proto to an accurate communication simulation. For this we used OMNeT++, a well-established network simulator framework. Our implementation runs Proto and OMNeT++ as separate processes, linked by extensions that use interprocess communication to synchronize the two simulators. We demonstrate this integration with an urban communications scenario, showing mobile devices using a spatial computing approach to create a communication channel along a path with ideal connectivity, as well as showing a tradeoff between speed and reliability of information sharing with neighbors.

II. CONNECTING PROTO AND OMNeT++

A spatial computer is any collection of devices distributed in space, such that the difficulty of communicating between devices is strongly dependent on their geometric distance. Examples include robotic swarms, wireless sensor networks,

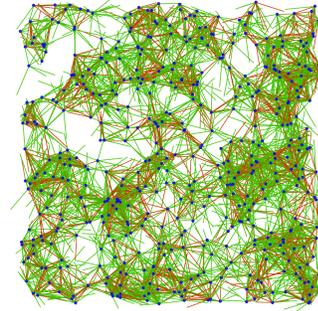


Fig. 1. Connectivity and link quality in an integrated Proto/OMNeT++ system, with reddish lines indicating poor connectivity. Lines with no nodes indicate outdated neighbor information.

and engineered biofilms. Proto is a purely functional language for programming such a spatial computer[1]—one of the few general purpose spatial languages amongst many more specialized spatial computing languages and platforms [3].

With Proto, the programmer views the system as an amorphous medium—a region of space consisting of a computational device at every point—rather than a collection of individual devices. This abstraction makes it possible to program the global behavior of the system in terms of geometry and information flow, then generate low level code for a virtual machine that runs on a variety of different platforms, including in simulation. MIT Proto [2] is an implementation of Proto, including a compiler that generates virtual machine code, and a simulator and virtual machine for testing algorithms. As noted, however, its simulator is using a unit disc model of communication, which prevents applications from being tested under realistic conditions.

OMNeT++ is a framework for building wired, wireless, on-chip or queueing network simulators [4]. OMNeT++ has a modular structure, which means that functionalities needed during simulation are grouped into individual modules and clean/standard interfaces are defined between them. Modules can have multiple gates and two modules can be connected via channels attached to the appropriate gates. Communication between modules is established by sending messages or signals through channels. This enables the reuse of available modules and makes it easier to extend the system because the dependencies between pieces of code are easily comprehensible and limited. Simulations are described by an input file, which defines the network, initializes the module instances and defines the connections between them. Castalia [5] is a framework based

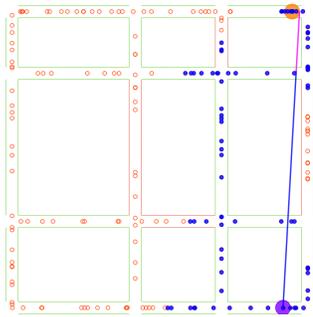


Fig. 2. Target tracking in an urban environment. Orange/red roads have higher path loss, therefore worse connectivity.

on OMNeT++ that is designed to simulate Wireless Sensor Networks and Body Area Networks, by including an advanced channel model with the capability to use user-defined path loss maps and temporal variation of the defined path loss. Castalia also supports mobility and models for various real world radio channels, modulation types, carrier sensing, clock drift and device power consumption. OMNeT++ and Castalia were thus an excellent choice for our scenario, providing realistic wireless models with a clean design for easy extensibility.

Both Proto and OMNeT++ are discrete event simulators. This means that there is an event queue that is filled with scheduled events during the simulation. In both simulators, each event is atomic and takes zero simulation time, while time is (generally) independent of real time. One way to connect two discrete event simulators is a continuous communication between the two, meaning that if an event occurs in one simulator the other has to be notified of that event. This implies continuous synchronization between the two simulators and a complex implementation. When interdependent events do not come too quickly, however, this can be loosened by letting the first simulator run for a short and predetermined amount of simulated time, stopping it, then sending all relevant events that happened during this time to the other simulator. Fortunately, Proto assumes a two-phase computation that allows us to choose this second approach, which is both easier to implement and has higher performance due to less synchronization.

III. THE URBAN TRACKING SIMULATION SCENARIO

To demonstrate the enhanced capabilities gained from integrating these two systems together, we have implemented an urban target tracking scenario. A simpler version of this demo is already present in the Proto distribution. The nodes here are communicating using radio links, via broadcast messages. After the simulation is started, the user selects a device to be tracked and a base station where the position of the tracked device should be sent. If the base station has a connection to the target (meaning they can communicate through some intermediate set of devices), then a vector is drawn between the two.

We wanted to try this scenario in an urban setting as it is a constrained environment compared to the standard Proto simulations. To implement this, a city mobility module for Castalia had to be designed along with a map file that describes the city layout. To model the communication conditions accurately, we then generated a path loss map from the city

map. The general idea was that communication should be possible along roads, but near impossible through buildings that cover the area between roads. In the generation of the path loss map we have also defined areas along roads where path loss is larger (e.g., due to interfering obstructions), making it harder or impossible for devices to communicate with each other. Figure 2 shows tracking running in this scenario, using the wireless communication model implemented in Castalia to accurately simulate the effects of wireless communication. Castalia can also allow us to investigate different settings of the radio hardware or different path loss configurations for this scenario.

IV. CONCLUSIONS

The urban setting and the non-uniform path loss map transform the behavior of the target tracking scenario. The target can still be tracked by the base station if any chain of devices that reliably exchange information can be found between them. Rather than following a simple shortest path, however, the chain follows a path that takes network conditions and device connectivity into account.

The connectivity model also provides an example of the self-adaptation challenges that can be investigated with the Proto/OMNeT++ system. Device connectivity can be severely obstructed by setting the message sending frequency in Proto either too low or too high. Slow message sending means that there is a better chance, that a sent message will arrive (little or no interference caused by other devices), but the information will be out of date, which can combine with device mobility to create an inability to form a channel between the base station and the target. On the other hand, sending messages at a higher frequency means that the neighbor information is now fresh, channels can be formed quickly, but since the wireless channel is saturated, devices experience high interference hence cannot communicate effectively with each other. Finding the optimum rate is an example of the self-adaptation problems that can be investigated with this system.

ACKNOWLEDGEMENT

The publication was supported by the TAMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

REFERENCES

- [1] J. Beal and J. Bachrach, "Infrastructure for engineered emergence in sensor/actuator networks," *IEEE Intelligent Systems*, vol. 21, pp. 10–19, March/April 2006.
- [2] "MIT Proto," software available at <http://proto.bbn.com/>, Retrieved June 22nd, 2012.
- [3] J. Beal, S. Dulman, K. Usbeck, M. Viroli, and N. Correll, "Organizing the aggregate: Languages for spatial computing," *CoRR*, vol. abs/1202.5509, 2012.
- [4] G. Pongor, "Omnnet: Objective modular network testbed," in *MASCOTS '93: Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*. San Diego, CA, USA: The Society for Computer Simulation, International, 1993, pp. 323–326.
- [5] A. Boulis, "Castalia: revealing pitfalls in designing distributed algorithms in wsn," in *Proceedings of the 5th international conference on Embedded networked sensor systems*, ser. SenSys '07, 2007, pp. 407–408. [Online]. Available: <http://castalia.research.nicta.com.au/index.php/en/>