Combining Wireless Network Simulation and Emulation

Sam Burnett

March 21, 2008

1 Experimentation Platforms

Developing new wireless networking technologies is a challenging task in part because it is difficult to perform controlled, repeatable experiments in a wireless environment. Wireless networks are very susceptible to external factors such as signal interference, attenuation and fading, and physical obstacles. These factors often cannot be controlled by the experimenter, making it impractical or impossible to perform flexible and repeatable experiments. In order to make construction and testing of wireless systems easier, researchers have built several wireless experimentation platforms that attempt to replicate real world scenarios while allowing the researcher full control over the experiment.

One class of such experimentation platforms are *network simulators*. A network simulator reproduces a large portion of the network stack in software. For example, instead of using a physical wireless radio and its corresponding physical and data link layers, a simulator may create such a radio in software. The advantage of this method is clear: since parts of the experiment are simulated, an experiment can be run without needing to purchase all of the equipment necessary to run the same experiment in the real world. The biggest downside of simulation is that the software components are only approximations of their real world counterparts. For example, a simulated wireless network card may not behave exactly the same as a real one. Performance scaling can also become an issue, since multiple radios are simulated on one machine. Nevertheless, simulators often provide an inexpensive way to get a high-level view of a wireless network system's behavior. They are the most popular form of wireless network experimentation platform. Examples include ns-2, GloMoSim and JiST/SWANS.

A second class of experimentation platforms are *network testbeds*. A testbed takes the opposite approach of that taken by a simulator. In a testbed, no components of the network stack are modified. Instead, a testbed aims to modify the environment so that external influences on the experiment can be controlled. If the experimenter has complete control over these influences, then an experiment can be repeated. Because testbeds use real world components, they are very accurate. However, conducting experiments using a testbed can be difficult due to the cost of components and difficulty of maintaining control over the surrounding environment. In addition, testbeds tend to limit the environment they can be used in. For example, a testbed may only operate inside a small room. If the experimenter

wishes to perform an experiment in another environment, he must physically move the testbed to a new location, which is not always feasible or even possible. Thus, testbeds are used in situations where accuracy is much more important than scalability or flexibility. An example of a wireless testbed is Mobile EmuLab.

A final class of experimentation tools are *network emulators*. An emulator is similar to a simulator, except that instead of redefining many levels of the OSI model, an emulator only redefines the physical layer. This means, for example, that real radios are used. Only the links between hosts are modified, not the hosts themselves. This solution offers a nice middle ground between simulators and testbeds. Since the link is the only thing that needs to be managed, it can be emulated in hardware, allowing for greater better performance and accuracy compared to simulators. At the same time, an emulator eliminates many of the practical problems with testbeds by completely controlling external factors that may influence an experiment. Emulators still lack scalability, however, since a lot of radios may be needed to perform an experiment. An example of a wireless emulator is the CMU Wireless Emulator.

All of these experimentation platforms can be modeled as implementation modifications to the standard OSI model. Each platform replaces components of the model with new components. For example, an emulator reimplements the physical layer and a simulator typically reimplements nearly all layers of the model. Keep in mind that this modified OSI model is used for each host in the experiment. Thus, if a simulator is running a 100 node experiment then each node is an instantiation of the simulator's version of the OSI stack. This representation of experimentation platforms as modifications to the OSI model is a concept that will be used in later sections of this document.

These three popular forms of experimentation platforms all suffer from various problems. Simulators are inaccurate, testbeds are impractical and unscalable, and emulators are unscalable. However, we aim to show that by combining experimentation platforms, we can create platforms that are more accurate, scalable and practical than existing platforms.

The remainder of this thesis is organized as follows. Section 2 introduces the concept of a hybrid experimentation platform and discussed challenges with implementing them in general.

2 Hybrid Experimentation Platforms

We define a *hybrid experimentation platform* (or simply hybrid platform) as an experimentation platform created by combining two or more members of different classes of experimentation platforms. More formally, given a set of experimentation platforms, a hybrid platform is created using some combination of nodes, each of which is an instantiation of one of the OSI stack for one of the experimentation platforms. For convenience purposes, we will refer to the OSI model used by a particular node as that node's *class* (e.g. a node is of the emulator class). An example of a hybrid platform is one created by combining a network simulator with a network emulator. An experiment using this platform might consist of twenty simulated nodes and five emulated nodes (or twenty nodes of the simulator class and

five of the emulator class.)

It is important to notice that, as presented, nodes in different classes cannot communicate, since at very least their physical layer implementations are different. This problem is typically solved using a gateway that translates from one OSI stack to another. Since the goal of any wireless experimentation platform is to reproduce real world behavior as accurately and efficiently as possible, the main difficulty in combining experimentation platforms is not just maintaining communication, but doing so in a manner that is representative of real world behavior.

2.1 Key Challenges in Building Hybrid Platforms

When designing real world network systems, the goal is to create a reliable, efficient means of transporting information. This same goal holds true when building experimentation platforms. However, there is an additional goal as well: to accurately reproduce the behavior of the real world system. When designing hybrid platforms, achieving this goal becomes harder since we often mesh systems that were not originally intended to work together. Combined with the fact that we seek a hybrid platform that is more accurate than its component platforms, designing hybrid platforms becomes a very difficult task.

The make designing such systems easier, it is useful to keep the key requirements in mind:

Shared Communications Medium In wireless networking, all nodes share the same communications medium. Thus, communication between two nodes can be significantly affected by a third party node or by external other interference. It is important that this property be maintained when modeling a wireless environment using an experimentation platform. In a hybrid platform, this means that if nodes A and B are communicating, a third node C node could have an effect on this communication, regardless of which class A, B and C belong to.

Accurate Propagation of Experiment Metadata In order to accurately translate between two experimentation platforms, it may be necessary to exchange information about the experiment itself. Such data includes the location of the nodes in the experiment and experiment event information. For example, if the experimenter wishes to start an experiment, it is important that every node executes this request at the same time. Depending on how this is accomplished, this may require synchronization of clocks between all nodes. Note that this is of exceptional difficulty when multiple nodes are running on the same processor (as usually happens with a simulator), since no two of these nodes can run at the same time.

Time Consistency Events that occur in the real world should take the same amount of time when they occur in a hybrid platform. For example, given an emulated node B and a simulated node C, if a node A equidistant from B and C sends a message it should reach both B and C are the same time. This requires that messages are processed very quickly when moving between experimentation platforms. Generally, this requirement is very difficult to achieve.

Verification As with any experimentation platform, it is important to evaluate how well a hybrid platform approximates the real world behavior of a wireless system. This is a particularly hard problem with hybrid systems, especially since our goal to create a platform that is more accurate that stand-alone experiment platforms.

2.2 Existing Hybrid Platforms

A few hybrid experimentation platforms already exist.

GrooveNet GrooveNet is a hybrid networked vehicle simulator that combines simulated vehicles with a testbed consisting of real cars on the road and allows them to communicate. It is useful for designing Ad Hoc vehicular network protocols. Because GrooveNet's driving real cars is rather impractical, a previous effort was made to combine GrooveNet with a wireless emulator. However, it became apparent as this project progressed that GrooveNet was never designed to support arbitrary experimentation platforms and would have required significant reimplementation to support this.

TWINE TWINE is a hybrid experimentation platform that combines wireless and wired networks. As such, it avoids numerous issues encountered with wireless networks.

2.3 Constructing a Generic Hybrid Platform

Construction of an actual hybrid experimentation platform depends highly upon the platforms that are being combined. However, there are ideas and methods that are common to all hybrid platforms. We discuss these commonalities here.

As mentioned in section 2, hybrid platform nodes require some form of gateway service to communicate with nodes of other classes. Ideally, this gateway would reside in separate process or on a separate processor or machine and would behave similarly to a gateway on the Internet. In other words, if a gateway needed to translate between two platforms at a level m of the OSI stack then the gateway would contain instances layers 1 through m from both platforms and use these levels in the translation between the two platforms. In practice, however, it is often desirable to run the gateway as part of the nodes themselves, for performance reasons. Thus, each node would contain several OSI stacks: one for communicating with nodes of the same class, and others for communication with nodes of other classes. In other cases, the gateway must use all seven layers of the OSI stack in the translation process, for implementation reasons. *Insert some diagrams here.*

Discuss in as much details possible the issues in section 2.1.

3 Combining Simulation and Emulation

As discussed in section 1, simulators are scalable yet inaccurate, while emulators are accurate yet unscalable. Thus, if the goal is to create a scalable, accurate hybrid platform, it makes sense to combine an emulator and a simulator and hopefully borrow aspects of both.

In addition, both simulators and emulators are more flexible environments than testbeds, making them better candidates for modification for a hybrid platform.

This section will give a detailed account of how to create a hybrid platform using a generic simulator and emulator, specifically addressing the issues introduced in section 2.1.

4 Combining the CMU Wireless Emulator and JiST/SWANS

This section will introduce the CMU Wireless Emulator and JiST/SWANS and explain why we chose to combine them. We will also give a detailed account of how this combination was done, again addressing the issues in section 2.1.

5 Validation of the Approach

Discuss techniques for validating the correctness of hybrid experimentation platforms.

6 Experimentation and Results

Discuss the experiments run and results we obtained.

7 Further Work

Discuss the shortcomings of the work, further experiments that should be run, and ideas that are implemented better in other projects.

8 Conclusion

Discuss the impact of this work for wireless systems design and verification.

References