Enabling Multiple Controllable Radios in OMNeT++ Nodes

Ólafur Helgason and Sylvia T. Kouyoumdjieva

Laboratory for Communication Networks
KTH, Royal Inst. of Tech.
100 44 Stockholm, Sweden
{olafurr, stkou}@kth.se

ABSTRACT

This work describes our implementation of a framework that allows mobile nodes in OMNeT++ simulations to be equipped with multiple radio subsystems that can be dynamically suspended and woken up. Our framework enables the simulation of wireless architectures that exploit radio hierarchies for power-efficient neighbor and service discovery and connection setup. The design is implemented as an extension of the MiXiM framework and is maintained as a MiXiM branch.

Categories and Subject Descriptors

I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Simulation, performance

Keywords

Wireless Networks, Opportunistic Networking, OMNeT++

1. INTRODUCTION

Nowadays mobile devices are commonly equipped with multiple radios and as an example, a modern smartphone usually has cellular, Bluetooth and 802.11 radios. These radios have different characteristics and capabilities in terms of communication range, bitrate, infrastructure dependence, energy consumption and more.

Recently there has been a growing interest in dynamically exploiting the different radios when performing cross-layer optimizations in wireless networks. As one example of this we have energy aware applications where the radio used to transmit or receive a message is selected based on context (application semantics, location, etc) or resource availability (i.e. remaining battery capacity). Another related example is vertical handoffs for dynamically changing the connectivity type of a device to support mobility. As devices become easier to program, it is likely that applications that can actively take advantage of the different radio capabilities become both more popular and more important.

In this work we describe our design and implementation of a multi radio simulation framework for OMNeT++ [10]. Our implementation extends the MiXiM [6] framework for mobile wireless networks and enables nodes in MiXiM to be equipped with multiple radios where each radio can be dynamically toggled between on/sleep/off modes.

In section 2 we discuss related work and further motivate our work. Section 3 describes our design and implementation of a framework that enables OMNeT++ nodes to have multiple controllable radio interfaces. In section 4 we present, as a proof of concept, some results where we use our framework for evaluating the energy savings of a dual radio system in a simple opportunistic content distribution scenario. We conclude our work in section 5.

OMNeT++ 2011 March 21, Barcelona, Spain.

2. BACKGROUND AND RELATED WORK

OMNeT++ [10] is a general purpose, modular event-based simulation tool, widely used for simulating communication networks. The MiXiM [6] framework for OMNeT++ provides support for both fixed and mobile wireless networks. In this work we present an extension to the MiXiM framework, which allows mobile nodes to have multiple controllable radio interfaces.

Although recent smartphones are powerful devices with advanced networking and multimedia capabilities, battery capacity is still a scarce resource. In particular, the wireless LAN interface is responsible for a large fraction of the energy consumption [9]. In [4], Feeney and Nilsson measure the energy consumption of an 802.11 interface in ad-hoc mode and provide a detailed energy profile. They show that the interface consumes significant energy in idle mode and that the ad-hoc mode is not as power efficient as the infrastructure mode. This is mainly because the infrastructure mode has a more efficient power saving mode since it can rely on the access point to synchronize and buffer data for nodes that are sleeping in the power saving mode. A recent implementation of an opportunistic content distribution system [12] further confirms this and shows that when the 802.11 interface is turned on, the battery life is reduced to only 25% of what it is with the interface turned off. This is despite the fact that no application data is being transmitted or received via the interface. Once the interface is turned on, it consumes relatively high power regardless of being in a *transmit* or *receive* state since waiting to catch a signal to decode in *idle* state consumes almost as much power as in when transmitting or

receiveing. This suggests that reducing or eliminating the *idle* energy cost of the 802.11 interface may be a promising strategy to reduce the overall energy consumption and prolong battery life.

The main goal of our work is therefore to enable simulation scenarios for exploring how nodes can utilize different radios to reduce the energy consumption of a mobile device. In particular, we are interested in evaluating the possible energy savings that can be achieved if nodes would be equipped with a low-power, low-bitrate radio for performing neighbor and service discovery. With such an approach, the high-power 802.11 radio could be suspended and only turned on for downloading when a feasible contact is found. We expect that much of the idle cost of the radio could be reduced.

For capturing energy consumption, our OMNeT++ extensions rely on the energy framework [5] which is a part of MiXiM. This framework contains among others, a detailed energy model of 802.11. When simulating mobile wireless networks, it is also important to have a mobility model that realistically captures the space in which mobility occurs [11]. Therefore we have updated our opportunistic networking extensions for OMNeT++ [13] as part of this work (which was previously based on the deprecated Mobility Framework). In addition to supporting multiple radios per host, our extensions therefore also provide support for dynamic node creation/destruction for simulating open systems and importing of externally generated mobility traces. Our code is maintained as a branch of the main MiXiM development branch and it is available at `https://github.com/olafur/mixim`.

ns-2 [2] is a discrete event simulator targeted at the networking research community. In [3] the authors provide guidelines on how multiple radio interfaces can be implemented in ns-2. The Cognitive Radio Cognitive Network CRCN [1] simulator is a network level simulator based on ns-2 that allows nodes to have a reconfigurable multi-radio multi-channel physical layer. Unlike our design, which devotes a separate network interface card for each radio, the CRCN simulator associates a single network interface card with different radio channels.

Some hierarchical radio systems have previously been proposed in the literature. The *Wake on Wireless* system [9] was the first to propose a dual-radio system in the context of mobile handheld devices. In [8], Pering et al. further show, by prototype measurements, that significant energy can be saved by using a low-power Chipcon CC100 radio or Bluetooth for discovering 802.11 access points. Motivated by this, CoolSpots [7] proposed a dual-radio Bluetooth/802.11 architecture to switch between radio interfaces with respect to traffic intensity. These systems have mainly been evaluated by small scale experiments on proof-of-concept prototypes. Our extensions to OMNeT++ and MiXiM facilitate the evaluation of these types of systems on a larger scale for different sets of application and mobility scenarios.

3. DESIGN AND IMPLEMENTATION

In this section we describe our design and implementation for supporting multiple controllable radios per node in OMNeT++. Our implementation is based on, and extends, the MiXiM framework and it is non-intrusive and does not break pre-existing code. In the following overview, we describe our design and implementation from an example scenario where nodes are equipped with a dual radio system. This is mainly

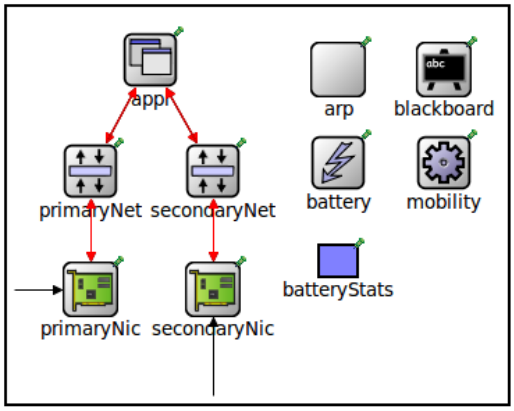

Figure 1: An example dual radio node.

for simplifying the presentation and we point out that our design is general and not just limited to two radios per node. We thus assume that each node is equipped with a primary low power radio which is used for transmitting neighbor and service discovery beacons while a secondary high-power radio is only used for downloading data after discovery has been performed by the primary radio. The high power radio can thus be suspended when not used for downloading or uploading.

3.1 Design Overview

Figure 1 shows the main modules comprising a mobile node equipped with two different radio subsystems: a primary low-power, low-bitrate network interface card (NIC) (`primaryNic`) and a secondary high-power, high-bitrate NIC (`secondaryNic`). In our design, the two radios co-exist at the node without the knowledge of each other. When running a simulation one would typically have a global `connectionManager` module for each radio type that manages the connections for each interface type (i.e. two `connectionManagers` if all nodes have identical dual radios).

Our design does not require modifications to any protocol layers outside of the NIC which allows for flexibility in choosing where to implement the logic for controlling the NICs. On the one hand, this logic can be implemented in the application (as in Figure 1). On the other hand it could be placed in a lower layer, and therefore the application would not necessarily have to know that multiple radios are used.

Just as in MiXiM, a NIC is a compound module that consists of a physical layer submodule and a submodule that implements the multiple access protocol (mac). One of the main features of our design is that we have extended a NIC to be *controllable* in the sense that it can be suspended or woken up by sending control instructions to it.

Each NIC can be in one of the following three states: `TURNED_ON`, `SLEEPING` or `TURNED_OFF`. A `TURNED_ON` NIC has full functionality, as currently implemented in MiXiM. A NIC in `TURNED_OFF` or `SLEEPING` mode is not active for transmission or reception of packets, but can be turned on or awaken when requested by upper layers. The main difference between the `TURNED_OFF` and `SLEEPING` states is that they can be configured with different energy consumptions and that there can be different latency for turning on versus waking up.

Apart from the radio and networking related sub-modules,

```
class IControllable {
public:
  enum Controls {TURN_ON, SLEEP, WAKE_UP, TURN_OFF};
  enum Status {TURNED_ON, SLEEPING, TURNED_OFF};

  virtual bool isOn();
  virtual bool isSleeping();
  virtual bool isOff();

protected:
  virtual bool turnOn() = 0;
  virtual bool sleep() = 0;
  virtual bool wakeUp() = 0;
  virtual bool turnOff() = 0;
};
```

Listing 1: Abstract interface for a controllable module.

```
class PhyLayerControl
   : public PhyLayerBattery, public IControllable
{
public:
  virtual void initialize(int stage);
  virtual void finish();
  virtual void receiveBBItem(int category, const
       BBItem *details, int scopeModuleId);

protected:
  virtual void handleUpperCtrlMessage(cMessage* msg);

  virtual bool turnOn();
  virtual bool turnOff();
  virtual bool sleep();
  virtual bool wakeUp();
};
```

Listing 2: Definition of a controllable physical layer module.

each node also has a `blackboard`, a `battery` and a `mobility` module. The blackboard serves as a notification mechanism for modules to signal internal state changes to other modules. It is for example used by MiXiM modules such as the mobility module to notify changes in position and the battery module to signal host failure due to battery depletion. With our extensions, a NIC can be controlled by external modules via the blackboard and a NIC also publishes any changes in its on/sleep/off state to the blackboard. We will now describe in more detail our implementation and extensions to individual components.

3.2 Controllable NIC

In our design, we extend a NIC to include a `NicController` submodule in addition to the standard mac- and physical-layer modules. Moreover, all the submodules of a NIC are extended to implement the `IControllable` abstract interface shown in Listing 1. The `NicController` has the following main responsibilities: 1) Receive control commands to the NIC from external modules (via the blackboard). 2) Ensure that the NIC submodules are suspended and awakened in correct order. 3) Simulate the delay when waking up or turning on a NIC and 4) publish state changes of the NIC on the blackboard. An external module thus controls a NIC by publishing one of the `Controls` defined in `IControllable` on a special control category on the Blackboard. Since the `NicController` is subscribed to this category it receives the control messages from the publishing modules. When the `NicController` receives a control message, it forwards it to the mac and physical layers of the NIC. Since all the submodules of a controllable NIC implement the `IControllable` interface, they invoke the appropriate protected member function that handles the control.

When both the mac and physical-layer modules of a NIC have changed their state, the `NicController` publishes the new NIC `IControllable::Status` on the blackboard. The blackboard thus allows us to inform all interested modules simultaneously about a state change, and then each module can decide whether and how to treat this information.

3.3 Extending MAC and PHY

When a new node is created in MiXiM, the `initialize` methods of all the node submodules are invoked. In MiXiM, a NIC does not have any on/sleep/off states and it is initialized in a state that corresponds to our `TURNED_ON` state. In our framework a NIC can be in any state when a node is created (it is configurable) and therefore we need to override the default MiXiM initialization of `mac` and `phy` modules.

In order not to break any existing code we create new `mac` and `phy` modules that extend the corresponding MiXiM modules. In these new modules we defer the standard initialization by overriding the `initialize` function of the parent class. In the new `initialize` function we now only read configuration parameters but defer the radio-state initialization which is instead executed when a `Control` message is received.

Listing 2 shows how we extend the physical layer of the MiXiM Energy Framework (`PhyLayerBattery`) with a new implementation. The new module only needs to override the `initialize/finish` functions and the control handler of the original physical layer (`handleUpperControlMessage`), all other functions can be left as-is. In addition, the new physical layer implements the `IControllable` interface (as described before) and the state change routines are now invoked from the overridden control handler.

When a `NicController` receives a `TURN_ON` message it is passed to the mac layer after some simulated delay. For the CSMA based mac implementations in MiXiM (i.e. Mac80211, CSMA and CSMA802154) the mac cannot start sensing until the physical layer has been turned on. Therefore the control message is passed down directly to the `phy` which in turn invokes `turnOn`. This method registers the NIC with the global `connectionManager` module for that particular radio type and initializes the radio which starts drawing a current from the battery. When the `phy` has turned on it sends a `TURNED_ON` message up to the `mac` which can now start sensing the channel. Then the mac sends the `TURNED_ON` message up to the `NicController` which publishes the message on the blackboard.

A `TURN_OFF` control message will result in the inverse behavior. First the `mac` empties all send and receive buffers. Then the `phy` unregisters from the `connectionManager`, stops drawing current and finally a state update is published by the `NicController` when the `TURNED_OFF` message has been passed from the `phy` to the `mac` to the `NicController`.

The handling of `WAKE_UP` and `SLEEP` messages is similar. The main difference is that the delay in waking up is different from that of turning on and that the current drawn in `SLEEP` state is different from that in `TURNED_OFF` state.

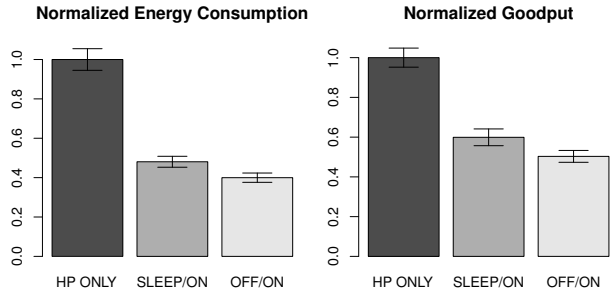

Figure 2: Normalized mean energy consumption (left) and mean goodput (right) for a simple dual-radio content distribution scenario.

4. EVALUATION

To evaluate and demonstrate the usefulness of our framework we have simulated the following scenario. We consider a simple opportunistic content distribution application where nodes share small content items with each other. Each node is equipped with a low-power (LP) radio that is always turned on and periodically broadcasts beacons. When a node receives a beacon from a neighboring device that it has not shared a content item with, it fires up its high-power (HP) radio for transferring data. When the data transfer is done, the HP radio is suspended if no other feasible node is in range. Our evaluation uses the MiXiM 802.11 NIC for the high power radio and the 802.15.4 NIC for the low-power radio and we assume a fixed communication range of 50 m. In OFF state the HP radio draws no energy but it takes 2 seconds to turn it on. In SLEEP mode it draws 10% of the energy it does in ON state but it takes only 500 ms to turn it on. We use mobility traces from Legion studio, a pedestrian mobility simulator, and the mobility scenario is that of an urban area modelled as a grid of streets (we refer to [11] for a full details of the mobility scenario).

Figure 2 (left) shows the average normalized energy consumption per node for three radio configurations:

HP ONLY: No LP radio. HP radio is used for both neighbor discovery and data download.

SLEEP/ON: Dual radio. HP radio sleeps when not active.

OFF/ON: Dual radio. HP radio is turned off when not active.

We see clearly that a significant energy can be saved by having a dual radio architecture where the HP radio is only used on demand and otherwise suspended. In Figure 2 (right) we see that a decrease in download performance (application goodput) can be expected since some contact opportunities are lost due to the delay in turning on (or waking up) the HP radio.

Our framework is ideal for exploring further the tradeoff between energy consumption and application performance. In our simple application, contact opportunities can be lost because we only use the LP channel for blindly sending beacons, i.e. nodes do not synchronize their wake-ups over the LP channel. It is likely that a more advanced LP control channel could give better application performance when compared with a HP only configuration. Our framework is also ideal for exploring effects due to different radio ranges, different energy profiles etc.

5. CONCLUSIONS

In this paper we have described our design and implementation of a framework for OMNeT++ that allows wireless mobile MiXiM nodes to be equipped with multiple controllable radios. Each radio can be dynamically suspended and woken up which enables the simulation of, among others, applications that exploit radio hierarchies for power efficient neighbor and service discovery and connection setup. We have demonstrated the usefulness of our framework with a demonstration simulation scenario, based on a dual radio node architecture and an opportunistic content distribution application. Our code is available as a MiXiM branch at `http://github.com/olafur/mixim`.

6. REFERENCES

[1] Cognitive radio cognitive network simulator. `http://stuweb.ee.mtu.edu/~ljialian/`.

[2] The network simulator ns-2. `http://www.isi.edu/nsnam/ns/`.

[3] R. A. Calvo and J. P. Campo. Adding multiple interface support in ns-2. Technical report, University of Cantabria, 2008.

[4] Laura Marie Feeney and Martin Nilsson. Investigating the energy consumption of a wireless network interface in an ad hoc networking environment. In *Proc. IEEE Infocom*, April 2001.

[5] Laura Marie Feeney and Daniel Willkomm. Energy framework: an extensible framework for simulating battery consumption in wireless networks. In *Proc. SIMUTools*, 2010.

[6] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating wireless and mobile networks in omnet++ the mixim vision. In *Proc. Simutools*, 2008.

[7] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta, and Roy Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proc. ACM MobiSys*, 2006.

[8] Trevor Pering, Vijay Raghunathan, and Roy Want. Exploiting radio hierarchies for power-efficient wireless device discovery and connection setup. In *Proc. IEEE VLSID*, 2005.

[9] Eugene Shih, Paramvir Bahl, and Michael J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proc ACM MobiCom*, 2002.

[10] András Varga and Rudolf Hornig. An overview of the omnet++ simulation environment. In *Proc. Simutools*, 2008.

[11] Ólafur Helgason, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson. Does mobility matter? In *Proc. IEEE/IFIP WONS*, 2010.

[12] Ólafur Helgason, Emre Yavuz, Sylvia Kouyoumdjieva, Ljubica Pajevic, and Gunnar Karlsson. A mobile peer-to-peer system for opportunistic content-centric networking. In *Proc. ACM SIGCOMM MobiHeld workshop*, 2010.

[13] Ólafur Ragnar Helgason and Kristján Valur Jónsson. Opportunistic networking in OMNeT++. In *Proc. Simutools, OMNeT++ workshop*, 2008.