Rapid Prototyping of a Frequency Hopping Ad Hoc Network System

Martin Braun, Nico Otterbach, Jens Elsner, and Friedrich K. Jondral
Communications Engineering Lab, Karlsruhe Institute of Technology (KIT), Germany
D-76128 Karlsruhe, Germany
martin.braun@kit.edu, nico.otterbach@student.kit.edu, jens.elsner@kit.edu, friedrich.jondral@kit.edu

***Abstract*—Wireless networks in-the-loop is a method to develop software radio systems which operate in networks. It eliminates the gap between simulations and live testing by allowing the developer to use the same code for both types of testing. This allows the parallel development of signal processing and protocols in a network context without having to use intermediary tools. We demonstrate this concept using the example of frequency hopping ad hoc networks, which are both challenging on the signal processing side as well as on the network side.**

***Index Terms*—Software Radio, GNU Radio, Ad hoc networks**

I. Introduction

The concept of Software Radio has changed the domain of radio technology development: A radio transceiver can now be represented entirely in software, with some generic interfaces for I/Q data and control signals (e.g. for centre frequency, bandwidth or analog gain settings).

Ideally, an SDR developer would never have to touch a piece of hardware when developing a radio system: The specifications could be formatted as a set of (software) unit tests, and once these pass, the SDR code is uploaded to its target device.

For wireless networks, and ad hoc networks in particular, this is not sufficient, as it is not only necessary to test individual nodes, but also how they interact in different network scenarios.

Wireless networks in-the-loop is a method to test networks of SDR nodes without leaving the software domain. Essentially, it models the distortions caused by the radio hardware as well as the propagation channels and makes sure signals are exchanged between nodes correctly. We first introduced the concept in [5].

Alternatively, wireless networks in-the-loop provides a simple way to distribute the SDR code among hardware nodes to perform over-the-air measurements. Channels can be sounded to repeat the measurements in a simulated (software-only) environment. These iterations – switching seamlessly between real and simulated radio environments – are the central elements of the development "loop". Because the same code is used for both modes, no additional development steps are necessary, thereby reducing development time.

Our wireless networks in-the-loop implementation uses GNU Radio for node development and radio propagation modelling. The interfaces provided for the node implementations are compatible to the USRP Hardware Drivers (UHD).

In the following section, we describe frequency hopping ad hoc networks, which are a type of radio system which benefits a lot from this kind of development. Section III describes our implementation in further detail. We then demonstrate the utility of wireless networks in-the-loop in Section IV, by applying the concept to frequency hopping ad hoc networks. Finally, Section V concludes.

II. Frequency Hopping Ad Hoc Networks

In scenarios where randomly distributed radio nodes need to form robust communication links, frequency hopping ad hoc networks are a viable option. By using frequency hopping (FH), nodes can both reduce collisions and interference due to spatial re-use of frequencies (*internal interference*) as well as the impact of third-party systems (*external interference*) [1].

However, the development of such systems is very complex: From PHY layer issues, such as synchronization, to MAC protocol design (e.g., how are hop sets exchanged?) a multitude of problems has to be dealt with. A challenging task is to organize parallel data transmission on several channels in an ad hoc network. Several classes of multi-channel medium access protocols exist [2] that all have their strengths and weaknesses.

Theoretical approaches only give very limited insight into the expected performance. To get a basic understanding of the underlying trade-offs in wireless networks, simplified models are employed. If an abstraction level is chosen that still allows the application of semi-analytical tools, the results have very limited applicability. For the analysis of interference in wireless networks, such a tool is stochastic geometry that can be used for interference modeling and yields spatially averaged interference estimates [3]. On the MAC layer, simple Markov-based models find widespread application [4].

Theoretical evaluation of all aspects of complex wireless systems that are necessary to realistically evaluate the network is plainly impossible. The resulting complexities of frequency hopping and the complex network structure make frequency hopping ad hoc networks an ideal candidate for testing our wireless networks in-the-loop implementation.

A. Analysis of a specific node parametrization

In the following, we will analyse a network of nodes with the following specifications:

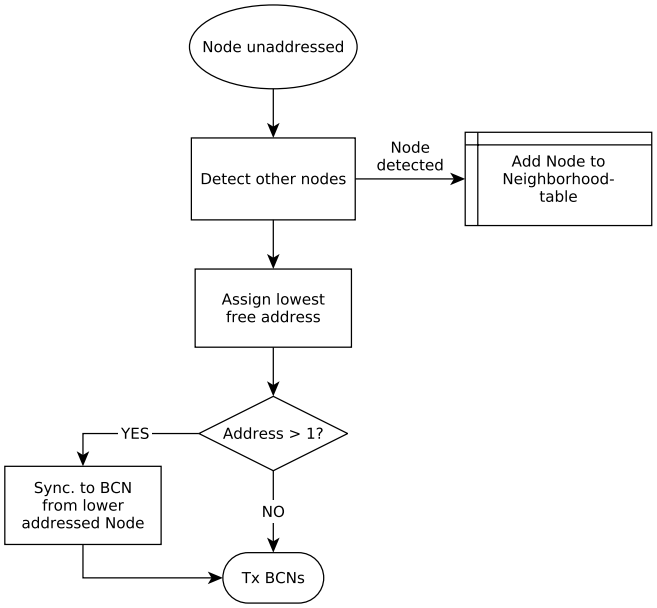


Fig. 1: Flow diagram of the node initialization sequence

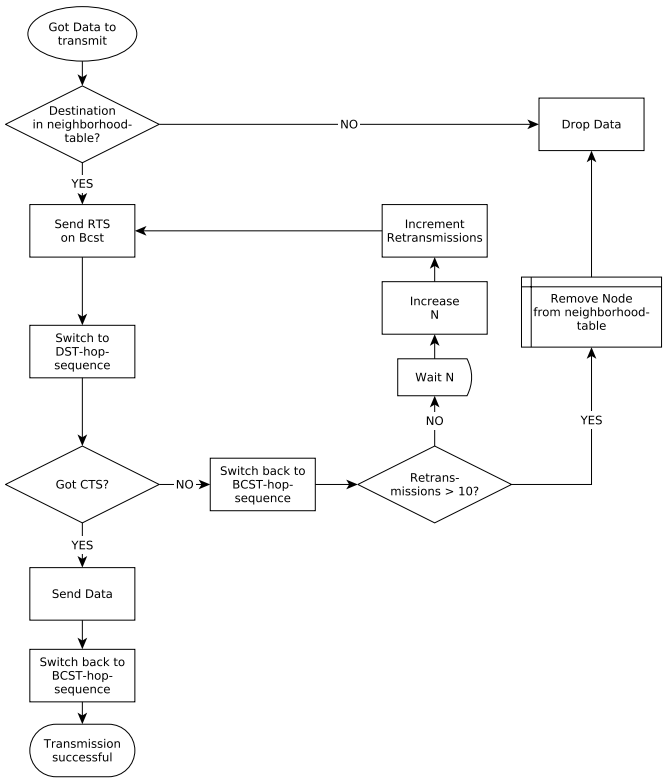


Fig. 2: Flow diagram of a point-to-point data transmission

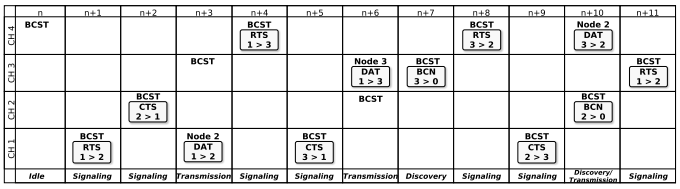


Fig. 3: Example of a channel access by three nodes.

a) Modulation: We use GMSK modulation with a bit rate of $200\,\mathrm{kbps}$. Available frequencies are spaced $1\,\mathrm{MHz}$ apart, every hop has a duration of $50\,\mathrm{ms}$, including a guard time of $8\,\mathrm{ms}$. Data transmission occurs in bursts of variable lengths (but never exceeding one hop).

b) Node addressing: Every node is dynamically assigned a network address, which is used for point-to-point transmission. Upon activation, a node observes the spectrum to detect other nodes (*neighbourhood discovery*). After a certain period of time, the node assigns itself the lowest free network address available. If no nodes are detected, it assigns itself address 1 (i.e., the lowest address possible).

Once a node has an address, it transmits a beacon packet in random intervals, but with a fixed average repetition rate (e.g. one beacon packet per 50 hops) on a hop set reserved for broadcast signals (the *broadcast channel*). These beacons are used by new nodes to identify the available network addresses.

After identifying other nodes and synchronizing with the hopping sequence, a node waits a random time to avoid simultaneous synchronisation with another new node.

Fig. 1 illustrates the addressing and synchronisation sequence.

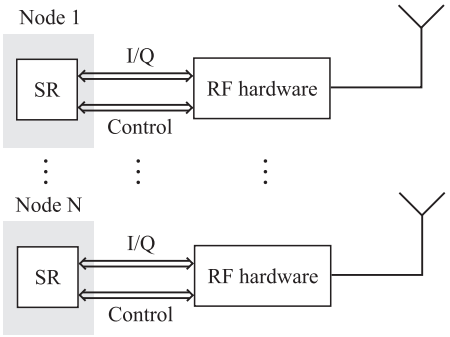
c) Point-to-point data transmission: Every node has its own hop sequence for data transmission, which is defined by its network address. When one nodes wants to transmit to another, it sends a request-to-send (RTS) packet to the destination node. If this node is not occupied, it answers with a clear-to-send (CTS) packet. The transmitting node can then transmit the data packets.

If no CTS packet is received, it retries the process after a random duration which increases with every try. After a certain number of tries, the receiving node is assumed to be no longer available, and it is removed from the list of neighbours, as shown in Fig. 2.

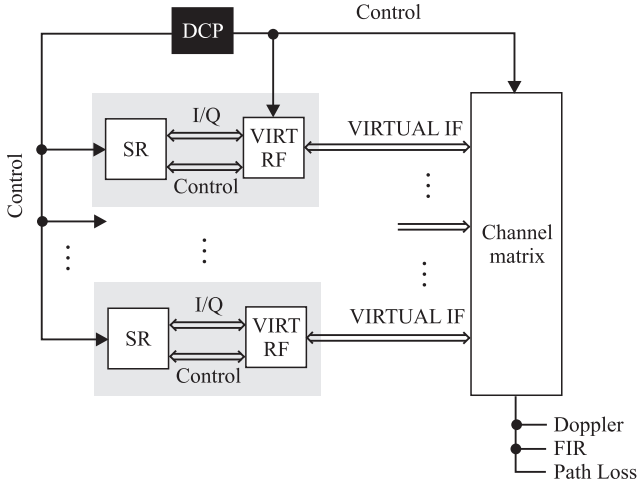
Fig. 3 shows an example of three nodes accessing four channels in total. First, node 1 attempts to transmit to node 2, receives a CTS signal and transmits. The same happens between nodes 1 and 3. Nodes 3 and 2 also send out a beacon signal (BCN) on the broadcast channel (BCST).

III. Wireless Networks in-the-loop

A method to speed up development time and remove the gap between simulation and implementation is *Wireless Networks in-the-loop*. The principle is simple: Instead of alternating between simulations (e.g. using Matlab) and testing live implementations, a radio system is directly developed using a software radio framework. In order to transmit and receive, such radio systems are connected to some kind of RF hardware, which performs the necessary steps to convert between passband and complex baseband signals (frequency



(a) Software Radio nodes operating with RF hardware



(b) Software Radio (SR) nodes operating in a virtual environment. A dispatch and control process (DCP) controls the individual nodes.

Fig. 4: The two modes of the wireless networks in-the-loop: Real and simulated environments

mixing, filtering, rate conversion etc.). The communication between the SR and the hardware usually consists of I/Q samples as well as control signals, such as the choice of the centre frequency, bandwidth or gain settings.

A wireless networks in-the-loop system can understand these data flows between SR and hardware and redirect them to an internally processed wave propagation model. Here, signals are passed between nodes digitally, including signal impairments caused by the hardware, such as I/Q imbalance or thermal noise, and of course the influence of the wireless channels, such as path loss and fading.

Fig. 4 illustrates the principle. Either the nodes operate with attached hardware, or within the simulated environment. For the individual nodes, it is irrelevant in which mode they are operating: From the software's perspective, they receive and transmit I/Q samples, regardless if they were physically transmitted or not. The typical workflow with such a system would be to quickly iterate between simulated and real measurements, thereby improving the quality of the SR node in every step. Because there is only one code base to work on, there is no friction when switching between simulation and live testing.

The three major challenges for developing a wireless networks in-the-loop environment are:

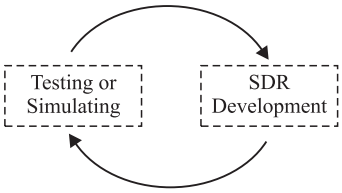


Fig. 5: The typical development cycle using wireless networks in-the-loop

1) Abstracting the hardware layer in software, which includes emulating the device drivers and distorting the signal in the same way the hardware does.
2) Modelling the wireless propagation channels between nodes, including fading and path loss.
3) Controlling the SR processes and connecting them either to virtual or real RF hardware, depending on the mode of operation.

A. Implementation details

For our implementation, we chose to limit the software available for the nodes to GNU Radio and the hardware to UHD-compatible[1] devices. GNU Radio is a free software library for developing software radio applications [6], and its open source nature gives us great flexibility. Also, it is easily portable to a large variety of platforms.

Our implementation includes:

- Connectivity to UHD. In particular, virtual interfaces to the hardware are provided, which allow switching between virtual and real hardware from outside the software radio node process.
- A subsystem to connect multiple SR nodes virtually in a wave propagation model.
- A dispatch and control process (DCP), which controls the individual nodes and connects them in the virtual wave propagation model when running in simulated mode.
- A simple channel sounding tool to measure channels in real mode for further use in simulated mode.

B. Developing SR-based networks

So how can this method be used to develop radio systems? The first step is to develop a prototype for a single node, which is not difficult if an established SR framework such as GNU Radio is utilized. During this step, all the standard tools of software development, such as unit testing, can and should be applied.

As soon as a first prototype is ready for a network operation test, it can be tested using the wave propagation model. First, a scenario is defined, which specifies the number of participating nodes, their position (these might also be generated randomly), the type of fading channels between the nodes and sources of external interference.

[1]USRP Hardware Drivers, an open source driver suite for devices by Ettus Research LLC.

Fig. 6: The test setup for the first live testing, using four USRP N210

To test the prototype, a supervising process dispatches an instance of the SR node for every time it appears in the scenario, as well as another process which connects the individual nodes by simulated fading channels. This simulation can be repeated with several different scenarios.

By logging relevant metrics such as packet loss, bit error rate etc., the performance of the individual node can be evaluated in a simulated environment. These results can then be used to further develop the SR implementation.

Once the network simulation works satisfactorily, we can deploy real hardware and live-test the network. It is likely that this test will expose new problems. Before returning to the simulated environment, it is possible to perform channel sounding measurements, such that the simulated measurements are similar to those in the real scenario.

The key point is that the code base for the SR nodes stays the same during all of these iterations. This eliminates any friction typically involved when switching between simulations and live measurements. Also, simulations and live tests both have their specific advantages and disadvantages, e.g., simulations can easily produce many different, but repeatable scenarios, where as live tests are per definition realistic. Only when the underlying code of the SR node is the same in both tests can results from one be transferred to another.

IV. Case Study: Developing and Testing in the Loop

The frequency hopping ad hoc networks from Section II were used to test the current state of our wireless networks in-the-loop implementation. In the first step, we created a node with the specifications from Section II-A using GNU Radio. This prototype is kept very simple, but it includes functionalities to test packet error rates, which we will use as a first performance metric.

Our hardware test bed consists of four USRP N210, as can be seen in Fig. 6, so first tests are run using four nodes to be able to compare the results between simulation and live testing.

Due to the simplicity of the individual node, it is possible to run four nodes in real time on one computer, which simplifies the testing. All USRPs are connected by Ethernet over a switch and can thus be controlled from a single machine. Also, the switching between live tests and simulations is simple and seamless.

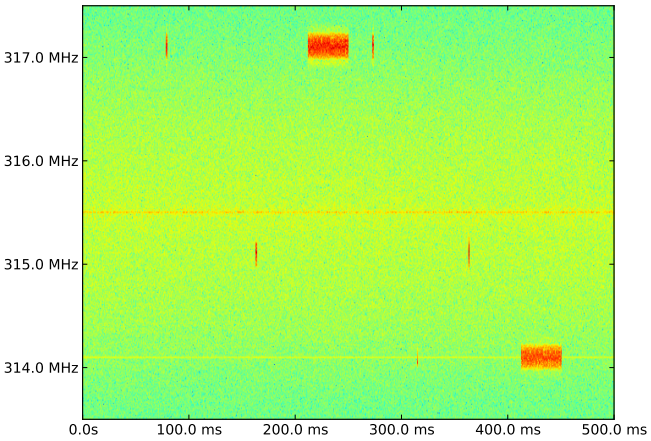


Fig. 8: Visualization of data exchange between two nodes.

To prove the concept of our implementation, a specific scenario is repeated both in the simulation and over the air. Fig. 7 shows how the nodes are communicating with each other (see Section II for a closer description).

When using the USRPs, a fifth device is used to observe the spectrum (see the waterfall diagram in Fig. 8). During simulations, the spectrum can be observed directly from the wave propagation model.

The network behaves similarly in both cases. As an example, the neighbourhood discovery takes the same amount of time regardless of the operation mode, which shows us that the testing of the network aspect is functional.

The packet error rates are different when switching between the modes, though: In simulated mode, we have a packet error rate below 1%, which increases to 2-5% in the live mode, depending on how the transmitters are positioned relative to each other. This difference is to be expected: In our current implementation, the hardware influences are not correctly modelled yet, which means the signal reaching the individual node is less distorted than in reality. By reconfiguring the signal processing of the virtual RF hardware, this can be changed to more accurately reflect the true values.

Now we know the simulation and the live testing results are comparable, we can start developing in the simulation mode (since it requires less hardware), but can continue to switch between modes in order to get more information about the nodes true performance. In simulation mode, we can add nodes without having to purchase additional hardware, knowing that the results will still reflect reality.

V. Conclusion

Wireless networks in-the-loop is a powerful tool for developing and testing SR-based networks. Using the example of frequency hopping ad hoc networks, we could demonstrate the capabilities of wireless networks in-the-loop and demonstrate how it can be used to reduce development time.

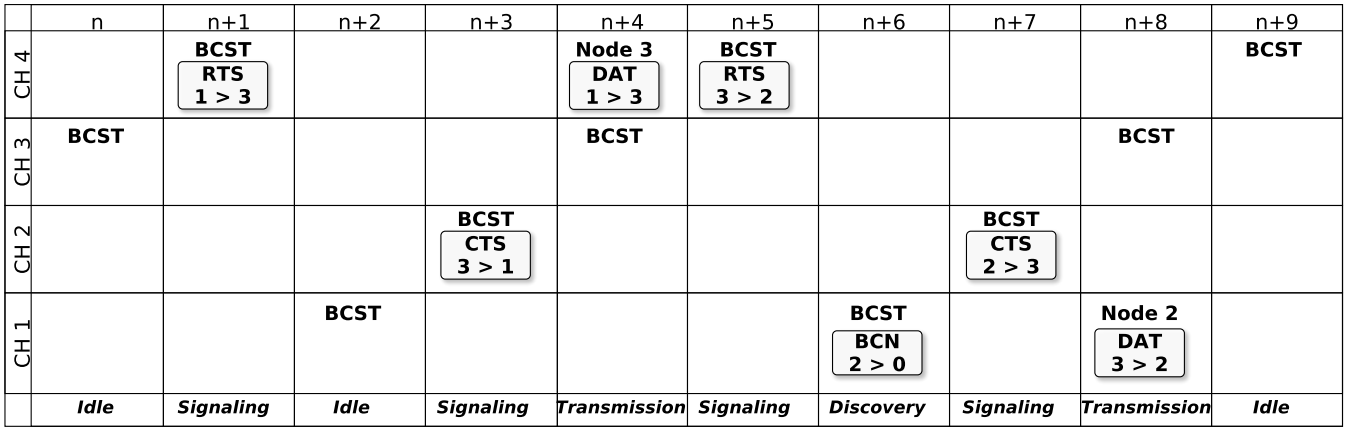


Fig. 7: Example for a data transmission between two nodes.

While our implementation still has some missing features, it is already able to show that network operation testing is possible, regardless of the mode. This allows developing nodes with a complex physical and network layer and produce realistic test results, which could not be generated with other tools which abstract the PHY layer of receivers when testing networks.

REFERENCES

[1] J. Elsner, "Interference Mitigation in Frequency Hopping Ad Hoc Networks," *Forschungsberichte aus dem Institut für Nachrichtentechnik des Karlsruher Instituts für Technologie*, vol. 29, 2012. [Online]. Available: http://digbib.ubka.uni-karlsruhe.de/volltexte/1000031465

[2] J. Mo, H.-S. W. So, and J. Walrand, "Comparison of Multichannel MAC Protocols," *IEEE Transactions on Mobile Computing*, vol. 7, no. 1, pp. 50–65, Jan. 2008.

[3] F. Baccelli and B. Blaszczyszyn, "Stochastic geometry and wireless networks, volume 1+2: Theory and applications," *Foundations and Trends in Networking*, 2009.

[4] J. Mo, "Performance Modeling of Communication Networks with Markov Chains," *Synthesis Lectures on Communication Networks*, 2010.

[5] J. Elsner, M. Braun, S. Nagel, K. Nagaraj, and F. K. Jondral, "Wireless Networks In-the-Loop: Software Radio as the Enabler," *Software Defined Radio Forum Technical Conference, Washington DC*, 2009.

[6] "GNU Radio website." [Online]. Available: www.gnuradio.org