Software Radio on General-Purpose Processors

Philip Mackenzie, Linda Doyle, Donal O'Mahony, Keith Nolan

Networks and Telecommunications Research Group. Trinity College, Dublin 2.

Abstract - *This paper presents the development of software radio systems in a flexible and adaptive way. This is achieved using a re-configurable communications stack and the use of general-purpose processors. Implementations and novel applications are presented.*

I Introduction

Bose [1] investigated the use of general-purpose processors (GPPs) for pure software radio. In his thesis he describes a framework for the development of software radio solutions using a minimal hardware front-end and suite of optimised signal processing algorithms. Up until then, the bulk of work in this area was concentrated on software defined radio (SDR). SDR uses dedicated re-programmable hardware to perform RF signal processing. By presenting working implementations, he has dispelled the myth that general-purpose processors are unsuitable for real-time RF signal processing. This has created a boundary between SDR and 'pure software radio', thus the term 'pure software radio' refers to real-time radio systems running on general-purpose processors using a minimal radio front-end.

The work described here has investigated the use of pure software radio from a networking perspective. The proliferation of mobile phones and increasing numbers of PDAs has shown a growing trend towards mobile wireless nodes. Mobility presents many challenges to the system designer as they strive to provide the quality of service and features offered by fixed networks. We believe that a flexible re-configurable network node gives the designer freedom to develop superior mobile nodes. This flexibility is achieved by using a re-configurable communications stack featuring software radio as the physical layer.

This paper presents work carried out in designing a software radio system that can meet the requirements of a flexible network node and that runs on general purpose processors. In Section II the use of general purpose processors for software radio is described. Section III describes a flexible platform for experimentation with networking technologies and demonstrates how this is used with software radio. Section IV discusses an implementation of this architecture. Section V presents some novel applications.

II Software Radio on General Purpose Processors

In the past few years there has been a move from traditional analogue RF hardware to digital systems that can perform base band signal processing. These systems digitise the base band RF signal and perform signal conditioning and demodulation using dedicated DSPs. As the power of DSPs has increased, there has been a move towards digitising the RF signal closer to the antenna. This has been coupled with an increase in the re-programmability of radio interfaces.

Our approach has been to use the GPP to replace the dedicated DSP. Using a GPP, the designer of the software radio system is presented with the ultimate in flexibility and ease of design. Software radio components can be implemented in familiar languages such as C and C++. Designers can use the familiar approaches of object-orientated programming and debugging to develop real-time software radio systems. This increases productivity significantly and thus reduces system development time.

III Software Architecture

In order to optimise the GPP as a platform for software radio a suitable software architecture was designed. In designing the software architecture the functionality of the wireless node from the application running on the node to the physical layer of the node was considered. Tomorrow's video, audio and data applications will put more demand on the communication stack, e.g. robust video streaming. Increasing wireless demands for quality of service (QoS) and bandwidth will require applications to know more about the underlying network, in particular the physical wireless layer. Today's network stacks are typically monolith models and offer limited reconfiguration. Therefore to achieve the flexibility required it was decided to use a component-based approach in designing the entire communication system so that key aspects of the system could be replaced and re-programmed at runtime. The main concern in designing the software architecture was to preserve performance while maintaining flexibility. This has been achieved using a component based dynamic stack.

A. Introducing a Component Based Dynamic Stack

All layers in our communication stack are independent components, each performing a specific task. The communication stack is formed dynamically by assembling a hierarchy of components to form the network application. Each layer in the hierarchy is an independent component. Different stack configurations can be assembled by selecting suitable networking components from a suite of available layers. Networking layers can be built using encapsulation. This makes it possible for a layer to be made up of a hierarchy of other layers. Data is represented in the form of message blocks. Each message block is passed through the stack using a simple set of primitives, send_upwards() and send_downwards(). Name value attributes can be associated with a message block that allows layers to provide information about data on a per-block basis. Other layers can then use this information to process the message block. The interconnection between layers is adjustable at runtime and layers can be loaded dynamically. This gives the stack the ability to reconfigure itself at runtime, allowing the creation of a flexible node. More information about the stack and its applications can be found in [2].

B. A Software Radio Physical Layer

Using the dynamic stack structure it is possible to create a flexible physical layer featuring software radio. The software radio physical layer encapsulates another set of dynamic layers that form the software radio system. Software radio can take advantage of dynamic layer loading and reconfiguration as these features are built into the structure of the stack. This means that the software radio implementation becomes truly flexible. For example, new modulation schemes and coding algorithms can be swapped in and out as required. Fig 1 shows a dynamic stack using software radio as the physical layer. All layers can be replaced or reconfigured at runtime.

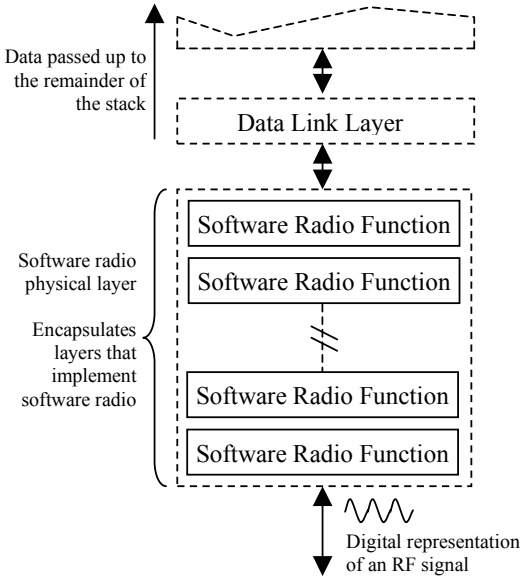

Fig. 1. A dynamic stack using software radio as the physical layer. Each layer can be replaced or reconfigured at runtime.

This system allows existing dedicated hardware to be replaced with a dynamic physical layer that uses a minimal hardware front-end.

IV Implementation

The dynamic communications stack including software radio has been implemented on Windows 2000. The software radio hardware is minimal and all signal processing functions are performed in software. Figure 2 shows a screenshot of one software radio application. This implementation demodulates an FM audio signal. A 10.7MHz IF signal is filtered and band-pass sampling is used to sample the signal at 2MHz. Demodulation occurs in real-time and the audio is played back through the PCs sound card.

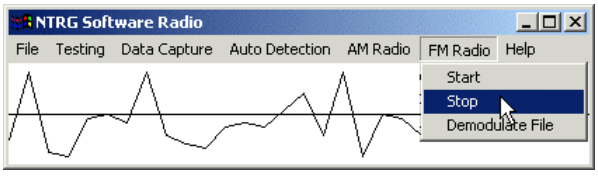

Fig 2 - A Software Radio implementation demodulating an FM audio signal

This implementation uses the dynamic stack structure to form the RF functionality and features automatic modulation detection (see section V).

V Novel Applications

Our dynamic software radio stack is capable of adjusting its layers to suit the particular wireless scenario in use. The following sections give an overview of two novel applications we have developed to demonstrate the advantages of integrating software radio into the dynamic stack.

A. Automatic Modulation Scheme Loading

Bose [3] presents a framework for changing modulation schemes on a per-packet basis. In this case the modulation scheme is determined by a packet header that identifies the modulation scheme. We have shown that it is possible to determine the modulation scheme of a signal without prior knowledge of the modulation scheme used at the transmitter [4] and we have implemented this scheme using the dynamic stack structure. In this scenario, the stack detects the modulation scheme of the incoming signal and reconfigures the stack accordingly. This powerful feature could be used to provide flexibility for roaming mobile devices. Devices could dynamically load new modulation schemes and become part of new networks without changing hardware. Fig 3 shows the receive case of a software radio stack configuration involving automatic modulation detection. Any number of modulation schemes can be added to the stack at runtime.

A layer in the communication stack acts as a modulation detector and loads the appropriate layers to demodulate the incoming signal. The modulation detection layer performs an analysis of the incoming signal to determine the modulation scheme used. This information is added to the attributes of the message block being passed up the stack. The modulation scheme loader makes sure that the appropriate layer is available to demodulate the incoming signal. If a suitable layer is not present, the loader will load the appropriate modulation scheme and reconfigure the stack so that demodulation can continue.

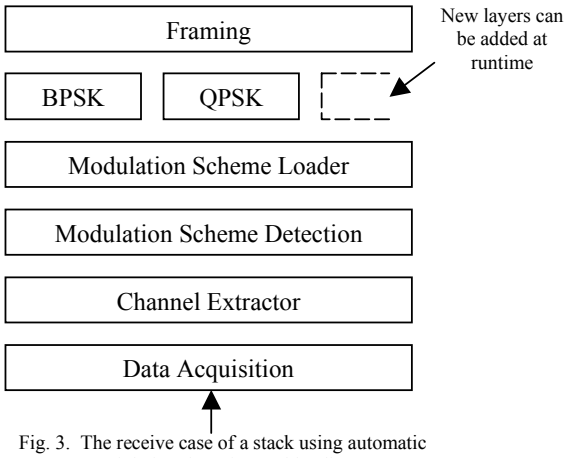

Fig. 3. The receive case of a stack using automatic modulation detection and loading.

B. Peer-Peer Component Sharing

Tuttlebee [5] describes Over-The-Air Reconfiguration (OTAR) for software radio. Using this concept, a remote device can be reprogrammed by the transfer of new software into the device. Up to now hardware restrictions has meant that OTAR has only been used for once-off projects such as satellites. We believe that advances in hardware will allow OTAR to be used in ad hoc networks.

An ad hoc network is a loosely connected set of mobile wireless nodes without any centralised or hierarchical structure. In this scenario, it is feasible that such large numbers of nodes will be using many different protocols, modulation schemes and location-specific parameters for RF communication. The ability of nodes to share information about network conditions is of key importance to ensuring realiable communication. We have developed this concept into 'peer-peer component sharing', which goes beyond information sharing. In this scheme nodes can offer each other actual layer components as downloads. These components are the layers making up the structure of the dynamic stack. This apprach allows a node to reconfigure its communication stack without contacting a central source. For example, a node entering a new location could contact the nearest node using a common modulation scheme to download the most popular communication component used in that area. These settings would include information such as available bandwidth, frequency allocations and noise conditions. In addition, nodes can share location specific data that can be used to optimise communications. Flexibility of this nature is not available in any current technology.

VI Conclusion

This paper has demonstrated a flexible approach to implementing software radio systems using a dynamic communications stack. This platform has proved to be a powerful solution for rapidly building and testing new communications systems and forms a basis for developing highly flexible mobile nodes.

References

[1] V.G. Bose, "Design and Implementation of Software Radios Using a General Purpose Processor", PhD Thesis, Massachusetts Institute of Technology, June 1999.

[2] O'Mahony D, Doyle L.E., "An Adaptable Node Architecture for Future Wireless Networks" in Mobile Computing: Implementing Pervasive Information and Communication Technologies, Kluwer series in Interfaces in OR/CS, Kluwer Publishing, in press for August 2001

[3] Bose, V. G., Morris, R. Hu, R., "Dynamic Physical Layers for Wireless Networks Using Software Radio", International Conference on Acoustics, Speech, and Signal Processing, Salt Lake City, UT May 2001.

[4] Nolan. K., "Software Radio Signal Space Based Adaptive Modulation For Wireless Networks", unpublished

[5] Tuttlebee, Walter., "The Impact of Software Radio" CEC Software Radio Workshop, Brussels, May 1997