

# Net-X: A Framework for Supporting Multiple Channels and Multiple Interfaces in a Wireless Network

Rishi Bhardwaj\*, Chandrakanth Chereddi<sup>†‡</sup>, Pradeep Kyasanur<sup>†‡</sup>, Paul Roycroft<sup>‡‡</sup>,  
Nistha Tripathi<sup>§‡</sup>, Vijay Raman\* and Nitin H. Vaidya\*

\* University of Illinois at Urbana-Champaign

<sup>†</sup> Google, <sup>‡</sup> University of Michigan at Ann Arbor, <sup>§</sup> Citigroup

## I. INTRODUCTION

Current generation hardware makes it possible to increase the *resources* available at each host by providing for multiple interfaces, multiple antennas, etc at reduced costs. Furthermore, newer hardware is capable of providing support for setting several radio parameters, such as the channel of operation, the data rate, and the transmission power, on a frequent basis. Collectively, we may view the different hardware resources and radio parameters as *interface capabilities* that are available in the network.

Conventional operating systems are built based on the principle of layering and information hiding. Owing to this, low-level interface capabilities are hidden from the higher layers of the protocol stack (e.g., the network layer is usually unaware of the notion of channels and transmission rates). While this simplifies the design of higher layer protocols, it may also severely limit the ability to exploit the available capabilities. Past work by our research group [1] on utilizing multiple channels has demonstrated the need for channel-aware link and routing protocols for effective utilization of channels. For instance, though channel-selection on a link may itself be a fairly localized decision, better global routes may be obtainable by using channel-aware routing metrics. Even for other interface capabilities, one may generally observe that, in the wireless context, there is often a strong coupling between various protocol decisions at different layers, and thus cross-layer design may yield substantial benefit justifying some increased complexity. At the same time, the interaction between layers must

be carefully designed, and ease-of-implementation must also be facilitated via clean interfaces to allow these interactions. This motivates the need to develop extensions to the existing operating systems to enable cross-layer protocol interactions.

A major goal of the Net-X project [2], [3] is to develop generic support for utilizing interface capabilities, such that the support is cleanly integrated into the network stack. As a first step in this direction, we have focused on providing support for utilizing multiple channels and multiple interfaces. We have developed generic architectural support in Linux that provides higher layers fine-grained control over the channels and interfaces used for sending out data [4], [2].

For ease of experimental evaluation, our architecture limits the kernel modifications by keeping most of the policy-related functionality in user space. Furthermore, the current implementation includes new architectural extensions for supporting the use of multiple channels and multiple interfaces, and a set of multichannel protocols [5] to demonstrate the use of the extensions. The implementation is extensible to support other interface capabilities. We are currently in the process of adding support for rate-control. We plan to gradually expand the scope to other aspects. Additionally, we are in the process of deploying an indoor testbed to investigate protocols that can exploit capabilities beyond just architectural support.

## II. SYSTEM ARCHITECTURE

In this section, we briefly describe the major aspects of the system architecture. Details are available in [4], [2].

The current system architecture has three major components:

This work is supported in part by NSF grant CNS 06-27074.

<sup>‡</sup> These authors worked on this project while they were at the University of Illinois at Urbana-Champaign.

Corresponding author's email: vraman3@uiuc.edu

- 1) *Channel abstraction layer*: This kernel component manages multiple channels and interfaces, and provides support for fast interface switching. This component is generic enough to support other multichannel protocols, and other interface capabilities, such as data rates and transmission powers. The channel abstraction layer abstracts the details of multiple channels and interfaces from the higher layers, and is controlled by "IOCTL" commands from the userspace daemon.
- 2) *Kernel multichannel routing support*: This component is used to provide kernel support for on-demand routing. The component informs the userspace daemon when a route discovery has to be initiated, and buffers data packets while the route discovery is pending.
- 3) *Userspace daemon*: The userspace daemon implements the less time-critical components of higher layer protocols (currently this is a multi-channel routing/channel assignment protocol). Most of the higher layer protocol functionality is implemented in this component.

The kernel components interact with the Linux TCP/IP implementation and the interface device drivers, while the userspace daemon is built using standard userspace networking libraries.

#### A. Channel abstraction layer

The channel abstraction layer (CAL) is implemented as a part of the bonding driver present in the Linux kernel. Its key components include:

- 1) *Unicast component*: Enables specifying the channel to use to reach a neighbor.
- 2) *Broadcast component*: Provides support for sending broadcast packets over multiple channels.
- 3) *Scheduling and queuing component*: Supports interface switching by buffering packets when necessary, and scheduling switching across channels.

In addition, the madwifi driver for Atheros-based NICs (which are used by our wireless nodes) has been modified to better support channel switching.

### III. ONGOING WORK

We briefly summarize some ongoing work:

#### A. Support for Rate-control

We are currently in the process of extending the Net-X architecture to add support for rate-control decisions that can be dictated by the userspace daemon. These extensions involve providing the ability to specify

the minimum and maximum rates to be used when communicating with a particular neighbor. The channel abstraction layer (CAL) has been extended to maintain information about rate-policies. The device-driver (in our case the madwifi driver for Atheros chipset based NICs), is modified to expose an interface whereby it can be informed of the minimum and maximum rates to use for a particular MAC destination. The driver can use its own internal rate-control algorithm within this valid rate-range. The CAL uses this interface to pass-down rate policy information to the driver. We also have basic mechanisms in place that can potentially allow fine-grained power control; however it is quite expensive to change power-settings, and so we do not have immediate plans of incorporating this.

#### B. Improved routing/channel assignment

We are working on link/channel quality aware routing metrics, and improved channel assignment algorithms. Furthermore, we are also working on a cross-channel interference aware routing metric.

#### C. Testbed deployment and measurements

We are in the process of evaluating our multichannel implementation on an indoor testbed of multi-radio nodes. Our testbed comprises of a set of Soekris net4521 single board computers equipped with two wireless interfaces (one pcmcia and one mini-pci interface). The number of interfaces can be expanded up to three, though our current implementation utilizes only two of those interfaces. The wireless cards are based on Atheros chipset driven by madwifi, and we operate them in 802.11a mode.

We have undertaken a set of measurements on our testbed deployment for evaluating the routing delay and throughput. Our initial measurements show that the channel switching delay plays a vital role in increasing the routing delay. Furthermore, our measurements show that broadcast packets such as RREQ (route request) and RREP (route response), used by our routing protocol, incur a high delay to propagate to all the nodes in our testbed setup. The throughput measurements show that the availability of multiple interfaces and multiple channels provide a significant improvement in throughput in case of an UDP flow. However, the throughput performance is not good enough for a TCP flow due to the need for supporting two-way traffic (data and ACK) in TCP.

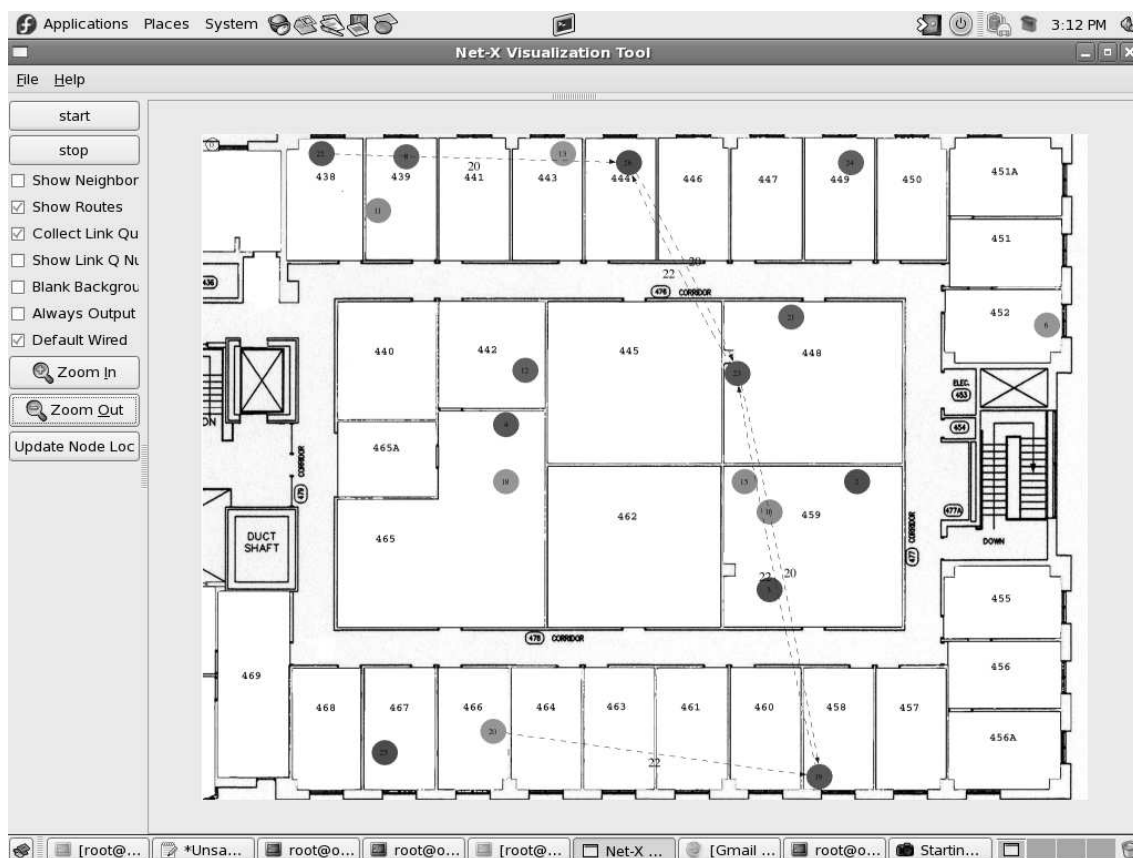


Fig. 1. A map of indoor network in our lab showing the testbed nodes location

#### D. Net-X visualization tool

We are developing a graphical visualization tool to assist in network visualization and management. Figure 1 shows a snapshot of the nodes deployed in our lab as seen using the visualization tool.

#### IV. OTHER DETAILS FOR THE DEMO

We intend to demonstrate our wireless testbed along with the visualization tool. We will be setting up two Soekris net4521 boards in mesh mode, and will be generating IP traffic between them. The nodes will be controlled by a laptop. We will be using an additional laptop for the visualization tool. All the laptops and the net4521 boards require wired internet connectivity and power supply. The setting up time is approximately one hour.

#### REFERENCES

- [1] P. Kyasanur and N.H. Vaidya, "Capacity of multi-channel wireless networks: impact of number of channels and interfaces," in *Proc. of MobiCom '05*, 2005, pp. 43-57.
- [2] P. Kyasanur, C. Chereddi, and N.H. Vaidya, "Net-X: System eXtensions for Supporting Multiple Channels, Multiple Interfaces, and Other Interface Capabilities," *Technical Report, CSL, UIUC*, August 2006.
- [3] "Net-X project," <http://www.crhc.uiuc.edu/wireless/netx.html>.
- [4] C. Chereddi, P. Kyasanur, and N.H. Vaidya, "Design and implementation of a multi-channel multi-interface network," in *Proc. of REALMAN '06*, 2006, pp. 23-30.
- [5] P. Kyasanur and N.H. Vaidya, "Routing and link-layer protocols for multi-channel multi-interface ad hoc wireless networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, 2006, vol 10, no. 1, pp. 31-43.