P2P WIRELESS NETWORK BASED ON OPEN SOURCE LINUX ROUTERS

Hristofor Ivanov[1] Miroslav Galabov[2]

Abstract – **In this paper we present our work towards deploying a community wireless network with ad hoc communication and routing between its elements. We describe our network model and implementation of wireless routers, while motivating decisions and pointing out open issues. The main advantage of our approach is the low deployment cost and inherent flexibility in terms of adapting the network configuration with little or no human intervention, which in turn can be exploited to support the dynamic addition, removal and mobility of network elements.**

Keywords – **Wireless Network, Open Source Firmware, Linux Routers**

I. INTRODUCTION

Algorithm design and evaluation in the field of wireless networks is performed using network simulators, such as ns-2 and NCTUns [1] [2], in order to systematically investigate system behavior under different assumptions, operating conditions and environmental settings. But it is also important to deploy and experiment with real-world networks. One key reason is that mathematical tools, even when used in conjunction with elaborate failure models, have limitations and cannot capture the full behavior of physical systems, such as the transmission anomalies in an inhabited area or the actual performance of commercial hardware. Real implementation and testing is thus needed to validate theoretically studied systems. Another, perhaps more important, motivation is that a testbed can actually be *used* to run not only test programs but also *real applications*. It is through such application-driven usage that unexpected system behavior is often discovered or new ideas emerge in terms of system and application functionality.

The deployment of infrastructure-based wireless networks has been straightforward and cost-efficient since wireless access points based on the 802.11a/b/g/n standards have entered mass production. However, this is not the case for ad hoc wireless networks given that wireless routers with ad hoc capabilities are hard to find in the market and are also quite costly. Another problem is that most such platforms are proprietary and closed so that is impossible to change internal settings let alone reprogram the network elements, for example to install a new routing or power management algorithm. This has led research groups to the development of wireless routers and networks based on personal computers and laptops [3]. While this achieves the desired flexibility in terms of software development and testing, it restricts the scope of deployment inside a single building or within an area of few neighboring buildings. Hence a PC-based network is not suited for urban environments where an outdoor, rooftop installation is usually needed to achieve good connectivity. In turn this poses additional requirements such as size constraints, weather protection, power supply and consumption and heat management. Last but not least, using PCs merely for the purpose of implementing wireless routers is too costly and would bring any large-scale deployment effort to a standstill. In this paper we present our work towards developing a community wireless network which employs low-cost, off-the-shelf 802.11b/g/n wireless routers running a custom Linux distribution. The network operates in ad hoc mode, thereby resulting in great flexibility and reduced administration. In the next sections, we discuss our motivation, give an overview of key hardware and software features of the router platform, present the current configuration and functionality, and discuss our experiences so far.

II. MOTIVATION AND REQUIREMENTS

In terms of functionality provided to the end user, we wish to deploy a wireless network of sufficient scale that will be used to run applications in a real-world environment outside of the laboratory. In addition, we wish to augment popular outdoor areas of the city with wireless connectivity to be exploited by mobile computers such as laptops and handheld devices. Besides supporting the typical suite of Internet applications such as e-mail, telnet, ftp and web browsing, we are interested in exploring peer-to-peer, groupware and ubiquitous computing applications. Notably, we do not want to limit participation only to students and faculty, and are strongly interested in attracting other communities like schools, local authorities, or even businesses, perhaps at a later deployment stage. On another dimension, we aim to create a testbed for implementing and evaluating algorithms and software, in the area of networking and distributed systems. Furthermore, besides using the network as a “dump” data carrier, our desire is to be able program the network elements in order to control their operation in a flexible way. We also wish to be able to install application-specific components, possibly on the fly, making the network itself an “active” part of the middleware or application's architecture. We believe that the dual nature of a “living laboratory” approach where a testbed is also used by students in their everyday lives will inspire them to become more actively involved in this area of technology. From a more practical but nevertheless crucial perspective, we aim for a simple, open and autonomous participation model that will encourage our students, but also users from other communities, to adopt the system. We are looking for an approach that is easy to implement, setup, manage and extend,

with as little human intervention as possible; maintenance is neither something we like to nor can afford to spend many resources on. At the same time, one has to strike a balance between performance and flexibility, while keeping the cost of network deployment low. The latter is particularly important because we want to adopt a community-driven approach where each participant covers (at least in part) the costs of installation. We feel that this is necessary to guarantee survivability without relying on a constant inflow of funds, which is hard to achieve in practice, especially in a non-profit academic environment.

III. Network Model

Our approach is illustrated in Figure 1, showing an indicative configuration that comprises several stationary and mobile network elements with typical deployment options to support fixed terminal devices. It is motivated and described in more detail below. Due to the fact that a large part of the city is densely built with many tall buildings standing next to each other, network elements (labeled as ad hoc routers in Figure 1) must come primarily in the form of stationary devices installed on roofs and balconies in order to achieve better connectivity. Their primary role is to provide an IP-plug, which can be used to connect one or more local client devices to the rest of the wireless network. At the same time, they may serve as hot-spots providing IP connectivity to mobile devices in range. We also wish for mobile devices themselves to serve as active network elements.

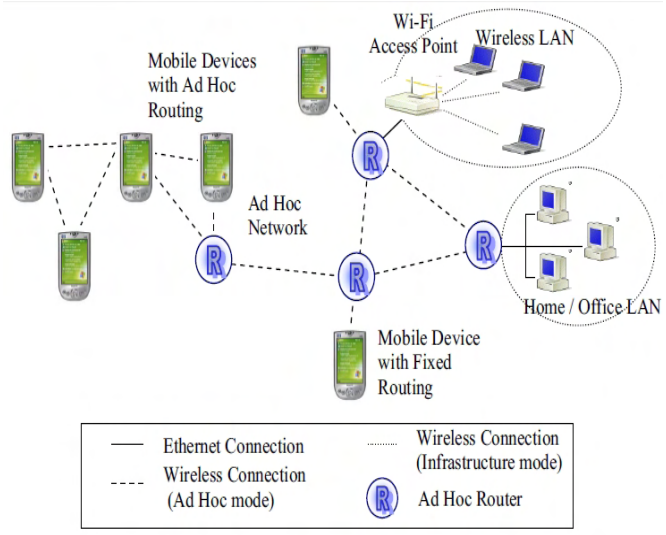

Figure 1. Network Architecture Model

This option can be used to eliminate the need for installing a stationary network element at home, and so that users may share part of the network's load even when on the move, if so desired, in accordance to a true community spirit. It should also be noted that mobile network elements can be particularly convenient for conducting on-site experiments of the type "what will happen if we place a new network element at location X?" for the purpose of teaching, demonstration, testing and configuration planning. A pure ad hoc and peer-to-peer networking approach was chosen to support this functionality. This is because we wish to be able to add new network elements without having to reconfigure the ones that have already been deployed. Even though such changes can be performed using a combination of remote configuration tools and scripting, they still require human intervention and assume that all network elements are up and running during the update process. Automatic adaptation when network elements are removed or do not respond is even more important. Failures are likely to occur occasionally, be it due to software glitches, hardware problems, power outages, or people resetting the equipment installed in their homes by accident. Moreover, in the case of mobile network elements topology changes are the rule rather than the exception thereby making manual reconfiguration practically impossible. In terms of end-user devices (terminals), we support two options: fixed and mobile terminals. In the first case, a device such as a personal computer or IP-enabled appliance connects to the ad hoc wireless network via a local area network. Several terminals can be connected to the network in this fashion, via the same network element. In the second case, a mobile device such as a laptop or handheld computer connects to the network via an ad hoc wireless connection to any nearby network element of the system. Mobile terminals may optionally assume the role of network elements with routing functionality, thereby enhancing robustness and increasing the bandwidth of the system. The distinction between fixed and mobile terminals is at the level of the IP protocol layer software and transparent for the applications residing on terminals.

IV. Implementation of the Ad Hoc Wireless Router

The choice of the wireless technology to be used was "naturally" trivial, with 802.11b/g/n being the only practical option given its wide industrial adoption, high-speed potential, and support for both infrastructure and ad hoc operation modes. But the quest for a suitable router platform proved somewhat more adventurous. As already mentioned, we desire a platform that can be programmed, ideally from scratch, in order to have as much development flexibility as possible. We also do most of our software development work in the Linux system environment, the open-source mentality and strong community support being the main reasons for this preference. Another restriction was that the device should fit in a reasonably sized weather-proof container without running into problems due to low external temperatures or –far more likely– overheating during summer. With PC platforms being out of the question due to their high price, big size and indoors-only deployment scope, our initial approach was to search for an embedded system board with interfaces that can be used to add Wi-Fi cards. A similar approach is followed in other projects, such as the WAND project [7]. We found that the WRAP Wireless Router Application Platform, designed by Pascal Dornier (PC Engines GmbH), met our requirements. Consequently we built a prototype wireless router using a WRAP board and two Prism 2.5-based Mini PCI wireless interface cards, running embedded Linux (Figure 2a).

Figure 2a. WRAP Linux router

Figure 2b. TP-Link TL-WR1043ND *Linux router*

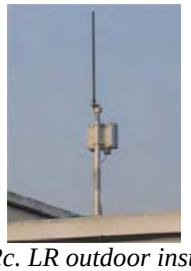

Figure 2c. LR outdoor installation

Unfortunately, although we were satisfied with its performance, the overall cost of the package turned up to be more than what a typical student could afford. Changing our strategy, we turned our focus on the readily available wireless products in the market and searched for platforms that would meet our requirements. We decided for the TP-Link TL-WR941ND and TL-WR1043ND broadband wireless routers (Figure 2b), which come with a Linux-based firmware and source code published under the General Public License. The TL-WR941ND device (v 2.0) features 4MB Flash memory, 32MB RAM, Atheros AR9132@400MHz, 10/100/1000 Ethernet controllers and an TP-Link TL-SG1008 8 port 10/100/1000 switch. The switched ports are separated into two different Virtual LANs (vlans), one for the LAN interface of the router comprising of 4 switched ports, and another intended for external Internet connectivity (WAN port).

The enhanced TL-WR1043ND version 1.8 features 8MB of Flash memory, 32MB RAM, Atheros AR9132@400MHz. Both devices have on-board interfaces for connecting to indoor or outdoor antennas. Experimentation in the lab with TL-WR941ND and OpenWRT proved that this particular hardware/software combination (henceforth referred to as the LR platform) met our requirements, and was therefore chosen as a basis for developing our own ad hoc wireless router and community network. Routing of IP packets in the ad hoc network is dynamic, as a function of topology changes. We decided to use the Optimized Link Source Routing (OLSR) protocol, due to its inherent flexibility, scalability potential and reduced administration overhead characteristics [4][5]. OLSR is a proactive, table-driven routing protocol based on the issue of Multipoint Relays (MPRs). It is considered to be well suited for large and dense networks with low mobility rates [6]. We tested a number of different solutions to choose a stable and highly configurable implementation.

The internal router configuration is as follows. The LAN interface, which is attached to the 5-port switch, is identified as eth0. The switch separates the ports into two Virtual LANs, one comprising of 4 ports (LAN segment), and another assigned to the WAN port. The LAN segment is addressed as vlan0 while the WAN port is identified as vlan1. Finally, the Wireless interface is identified as eth1. In the default configuration, eth1 and vlan0 are bridged together. We decided to separate these interfaces because we assume that devices connected to the router's LAN are terminals that do not (have any reason to) support ad hoc routing. A diagram that illustrates the router setup approach is shown in Figure 3.

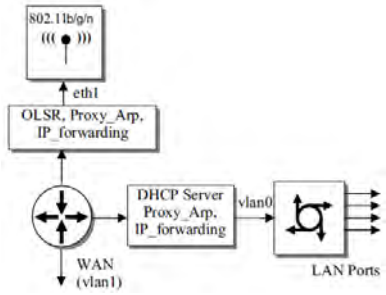

Figure 3. The internal router configuration

V. Address Management

An important requirement of our network is that routers should be able to join, quit and reappear, perhaps at another location, at arbitrary times, without the individual network elements having to be explicitly reconfigured by an administrator. At the level of IP, this means that address assignment and routing should be performed dynamically and in a location-independent manner.

On switched local area networks, it is possible to automatically assign IP addresses to a priori unknown hosts via the Dynamic Host Configuration Protocol (DHCP) [8]. This requires that the DHCP server is reachable in one hop from the requesting host, which is definitively not the case in the envisioned network. One solution is to use DHCP relay [9] to forward requests to a central server. However, this would make our system fragile as it introduces multiple single points of failure, namely the server itself plus every ad hoc router that is used to create the network path between the host and the server. Decentralized IP address assignment algorithms have been proposed to address this problem in a distributed fashion, including [10] [11] [12] [13], yet no implementation seems to be readily available.

Since we wanted to start the deployment we decided to adopt a static IP addressing plan, as a temporary and easy to implement solution. A registry of IP addresses and subnet addresses was created for the ad hoc network, and each router is statically assigned an IP address which uniquely identifies it on the network. Furthermore, each router is given a subnet network address for 32 IP addresses to be used in the context of its local LAN. The first address is assigned to the router's LAN interface, which leaves room for 29 IP-enabled devices that can be connected to the LAN interface of each router. Along this scheme, each router features a DHCP server that

manages a pool of 29 addresses and provides automatic IP address assignment for the LAN interface, where we also activated Proxy ARP and IP Forwarding using the proc file system. Hence devices that connect to a router through the LAN interface are automatically configured. In turn, OLSR running on the WLAN (vlan0) interface is configured to advertise the subnet corresponding to the LAN segment to the rest of the system.

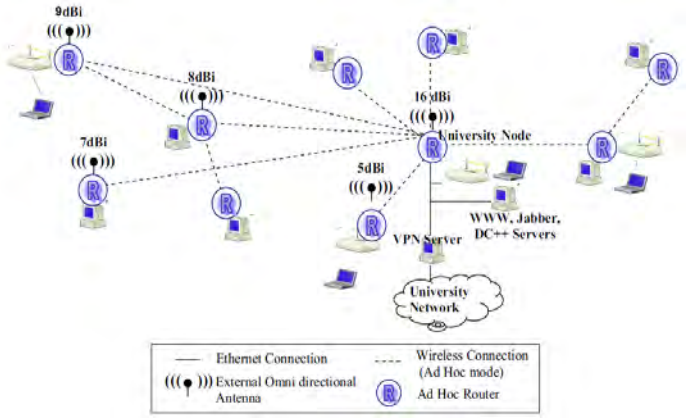

Figure 4. The Ad Hoc wireless network testbed covering.

VI. Conclusion

The network model is kept open so that many different communities are free to join, contribute to and exploit it. However this does not imply that all participants should be allowed to access the same set of services. Public Internet access, via the University's infrastructure, is a typical example. Of course, nor should any participant be able to retrieve account names and passwords transmitted over the ad hoc network. Security and access control is implemented using a Virtual Private Networking (VPN) approach. VPN technologies provide an efficient solution with central control of user access mechanisms. Our implementation employs the Point-to-Point Tunneling Protocol (PPTP) [14] with MPPE128 encryption [15], which is supported by almost every operating system including those running on most handheld devices. The VPN server is installed on a dual-CPU (2400MHz) personal computer with 2048MBs of RAM. Linux was the operating system of choice and PoPToP was used as the PPTP server [16]. Measurements performed on the local network showed a packet encryption/decryption rate of 950Kbytes/sec, which we consider sufficient for the purpose of initial deployment. The VPN server is equipped with two Ethernet interfaces. The first interface connects to the University LAN through which access to the Department's servers and public Internet is provided. The second interface connects to the LAN interface (vlan0) of an ad hoc router that connects the VPN server to the community wireless network. The network segment between this particular router and the VPN server was assigned an individual network address and the OLSR daemon on the router was configured to advertise the corresponding network address. Using the IP Masquerade feature provided by the Linux kernel (IP Tables) we configured the VPN server to perform Network Address Translation (NAT) between the computers attached to the VPN and the University network. NAT was necessary since the IP address pool of the University cannot be used for the purposes of the ad hoc network for reasons of security and scalability. With this setup, students and faculty may connect to and access the University network from any terminal connected to any ad hoc router. We have successfully tested Linux, Windows XP/ Vista and Windows 7 PPTP implementations over the wireless network.

References

[1] Lee Breslau et al., “Advances in Network Simulation”. IEEE Computer, 33 (5), May 2000, pp. 59-67.
[2] S.Y. Wang, C.L. Chou, C.H. Huang, C.C. Hwang, Z.M. Yang, C.C. Chiou, and C.C. Lin, "The Design and Implementation of the NCTUns 1.0 Network Simulator", Computer Networks, Vol. 42, Issue 2, June 2003, pp. 175-197.
[3] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, Robert Morris, “Link-level Measurements from an 802.11b Mesh Network”, SIGCOMM 2004, Aug 2004
[4] T. Clausen et al., “Optimized Link State Routing Protocol”, IEEE INMIC Pakistan, 2001.
[5] T. Clausen, P. Jacquet, “Optimized Link State Routing Protocol (OLSR)”, Request for Comments 3626 (Experimental), Network Working Group, Internet Engineering Task Force (IETF).
[6] P. Trakadas et al., “Efficient Routing in PAN and Sensor Networks”, ACM Mobile Computing and Communications Review (MC2R), Vol. 8, Num.1, January 2004, pp 6-9.
[7] Stefan Weber, Vinny Cahill, Siobhan Clarke and Mads Haahr. Wireless Ad Hoc Network for Dublin: A Large-Scale Ad Hoc Network Test-Bed . E RCIM News, 2003, vol. 54.
[8] R. Droms, “Dynamic Host Configuration Protocol”, Request for Comments 2131 (Standards Track), Network Working Group, Internet Engineering Task Force (IETF).
[9] Matthew J. Miller, William D. List, Nitin H. Vaidya, “A Hybrid Network Implementation to Extend Infrastructure
Reach”, Technical Report, January 2003.
[10] MansoorMihsin and Ravi Prakash, “IP Address Assignment in a Mobile Ad Hoc Network”, MILCOM 2002.
[11] Nitin H. Vaidya, “Weak Duplicate Address Detection in Mobile Ad Hoc networks”, Proceedings of 3rd International Symposium on Mobile Ad Hoc Networking & Computing, 2002.
[12] S. Nesargi and R. Prakash, “MANETconf: Configuration of hosts in a mobile ad hoc network”, INFOCOM 2002.
[13] C.E. Perkins, E.M. Royer, and S.R. Das, “IP address autoconfiguration for ad hoc networks”, Internet Draft, IETF
Working Group MANET, (Work in Progress), July 2000.
[14] K. Jamzeh et al., “Point-to-Point Tunneling Protocol (PPTP)”, Request For Comments 2637 (Informational), Network Working Group, Internet Engineering Task Force (IETF).
[15] G. Pall, G. Zorn, “Microsoft Point-to-Point Encryption (MPPE) Protocol”, Request For Comments 3078 (Informational), Network Working Group, Internet Engineering Task Force (IETF).
[16] PoPToP Project, URL: http://www.poptop.org/

[1] Hristofor Ivanov, St. Cyril and St. Methodius University of Veliko Turnovo, Veliko Turnovo 5000, Bulgaria, E-mail:globul@mail.bg
[2] Miroslav Galabov, St. Cyril and St. Methodius University of Veliko Turnovo, Veliko Turnovo 5000, Bulgaria, E-mail:lexcom@abv.bg