A Blueprint for a Manageable and Affordable Wireless Testbed: Design, Pitfalls and Lessons Learned

Ioannis Broustis, Jakob Eriksson, Srikanth V. Krishnamurthy and Michalis Faloutsos
Department of Computer Science and Engineering, University of California, Riverside, CA 92521
{*broustis, jeriksson, krish, michalis*}*@cs.ucr.edu*

***Abstract*—In this paper[1], we describe the deployment efforts of our 802.11 indoor wireless testbed. We elucidate the challenges that we faced and the design decisions that we had to make, sometimes based on technical reasons, and other times due to practicalities. These design challenges involve: (a) accessibility to the software, in order to be able to modify and implement various functionalities, (b) physical extendability, in order to add hardware in the future and, (c) manageability, in order to configure and update the software easily and quickly, for all the nodes in the network. We justify the hardware and software design choices that we make in order to facilitate these requirements. For ease of maintenance and convenience, each node is diskless, and we utilize power-over-ethernet through an Ethernet connection with a central server. We ensure that the software can be easily modified; this provides for easier module implementation and parameter tuning. We explain the different ways of node deployment, decisions that we make on power settings and discuss how and why the receiver sensitivity affects deployment decisions. Finally, we present our observations based on a set of measurements to quantify the stability of the links in our testbed.**

I. Introduction

Our goal in this paper is to delineate the challenges that we faced during the design and deployment of our indoor wireless testbed. Although, given the numerous previous wireless testbed efforts [1], [2], [3], [4], it may seem that such a deployment process is well understood and is easy to implement, in practice a plethora of issues arise. Thus, it is important to identify the requirements from the network and the constraints which typically dictate the deployment strategy. We discuss the design decisions that we had to make and some of the pitfalls that are to be avoided for successful network deployment. We believe that our experiences can serve as guidelines to upcoming deployment efforts on wireless networks.

Many wireless network testbeds have emerged in the last few years. Our deployment effort differs from the prior efforts primarily due to two design choices. First, in our network, the nodes are diskless and load an operating system into their memory directly from a central network server. Second, we use Power over Ethernet (PoE) to empower the nodes. These features were adopted as a result of our assessment of the network requirements and the constraints that limited us, in terms of the possible deployment strategies. We list these below. We believe that, for any network deployment effort, a pre-deployment assessment like this can greatly help in making the right strategies.

1. Functional requirements: It is important to be able to easily tune basic network parameters, such as the frequency, the transmission power and the rate. One should also be able to modify or implement various MAC and routing functionalities.

[1]This work is supported in part by the NSF CAREER Grant No. 0237920 and the NSF NRT grant No. 0335302.

2. Hardware requirements: If a heterogeneity of hardware is to be used, the different components need to be compatible with each other. In addition, one should be able to easily extend the network to include additional nodes and new technological possibilities.

3. Software requirements: It is critical to ensure the ease of performing software configurations and updates uniformly for all nodes. In addition, it is preferable to use open-source software that allows for network-customized modifications.

4. Efficiency and social implications: Deployment should be as quick and reliable as possible. It should be *non-intrusive* and should not interfere with day-to-day operations of other co-located wireless networks.

5. Cost constraints: Given budgetary constraints, the cost should be kept low but without compromising on the desired capabilities.

6. Manageability: Network configurations should be performed from a central location, remotely. One should be able to automatically configure and distribute updates to all nodes from this location, as well as gather logs from each node.

Our architectural design was primarily motivated by the above factors. We describe various possibilities and discuss why our choice of using diskless nodes that are powered over the Ethernet was the appropriate choice. We also describe some of the pitfalls that we ran into and this can forewarn researchers looking to deploy such a network in the future. Finally, we also discuss some of the interesting network characteristics that we have observed via preliminary measurements. We wish to point out here that we have a website that provides more details on our wireless network [5] and this can be a useful source of information during new deployment efforts.

Architectural design choices: The key properties of our network architecture are:

- Nodes are diskless. This makes them faster, less noisy and less vulnerable to malfunction.
- A central server distributes the operating system to all nodes, and gathers measurement logs. Hence, all nodes may load the same version of the operating system and applications into their memory. This ensures consistency among the nodes.
- PoE is used to empower all nodes. This allows us to power off/on individual nodes, by *remotely* (de)activating the PoE support.
- We avoid using a personal computer (PC) as a node; this helps us reduce operation noise, conform to space constraints and to avoid some interference effects, as we explain later.

Potential deployment pitfalls: During the course of deployment, we ran into numerous pitfalls. We deliberate on these in the paper and we hope that this can provide guidelines on

what to avoid for future network deployers. We briefly list the major ones here and discuss all of them in greater detail, later in the paper.

• Choice of power: We observe that using the maximum transmission power does not always give the best link level throughput. This is true even when links work in isolation. Some cards exhibit problems when they are operating at their nominal maximum transmission power. We observed a significant throughput increment, when we reduced the power by two dBm below the maximum value.

• Node deployment: High node densities can impede network operations to a great degree; the adverse effect is increased interference between links. We find that when it comes to deploying nodes in practice, one should take into account the maximum transmission power and the receiver sensitivity.

• Hardware functionality verification: All the hardware equipment should be verified extensively for its correct functionality. If not, *hours or even days* of frustrating debugging are likely to be expended.

Preliminary measurements and observations: We have conducted a set of experiments to understand the behavior of our network; we observe that:

• The choice of frequency channel can have a significant effect on the link and network throughput. The main reason is that different frequencies have different penetration properties.

• Obstacles and the type of the material that they are made of, have a great effect on link throughput. In many cases we observed that surfaces, such as glass and metal, create unexpected network topologies.

The rest of the paper is organized as follows. In Section II we describe in detail the hardware and software components of our indoor testbed. We also deliberate on how the network is constructed. We delineate the various possibilities that were considered and explain the reasons behind our design choices. En route, we also discuss the pitfalls that we ran into, how we discovered them and how to avoid them. In Section III, we present our preliminary measurement results. We conclude in Section IV.

II. Testbed Characteristics

Our indoor testbed is comprised of 31 nodes, deployed in the 3rd floor of Engineering Building Unit II at the University of California, Riverside. The network is depicted in figure 1. In this section we describe the hardware and software components of our indoor network testbed, and we justify our decisions toward selecting the specific configuration.

A. Hardware components

Choosing the appropriate hardware for the network is probably the most difficult decision, especially if the available budget is not astronomic. A set of parameters have to be taken into consideration, before making the final choices.

Remote access: Nodes must be fully accessible from a remote location. This also includes being able to power on/off the nodes from a central location. Furthermore, any software changes and implementations must be updated and loaded in the same manner at all nodes, almost simultaneously.

Financial cost: This mainly involves the hardware purchases, i.e., the nodes, perhaps a *server* and a set of switches, and cabling. The nodes must not be very expensive, so that the replacement of a node in case of failure/theft, or the purchase of some additional nodes is affordable at any time. One should also account for some initial hardware purchases that will be used for testing.

Silent and small-size nodes: For the case of indoor deployments, nodes are usually installed on bookshelves, desks, on the floor or next to windows. Hence, nodes have to be silent and with a decent look, so as not to disturb people either aesthetically or acoustically.

Extendability: Since the networking technology advances rapidly, one should provide for the evolution of the network. Nodes should have the resources to support technologies. In particular, each node should have available inputs to support hardware (such as card slots, USB devices, extra antennas) for future use.

Flexible wireless components: The wireless part of the hardware (i.e., wireless card and antenna) should be fully adaptable, to support software modifications. Parameter tuning and implementations should be feasible tasks. Moreover, nodes must be capable of covering a large region with full transmission power. This will allow the researcher to create both sparse and dense network topologies.

Taking all of the above factors into account, we decided that the best solution is to use diskless nodes, with PoE (Power-over-Ethernet) capability. PoE makes it easy to power on/off the nodes remotely. This is possible if one has remote access to the machine that provides power - in our case a network switch. Accessing the PoE switch remotely makes is easy to enable or disable the power supply on the switch's Ethernet ports. Consequently, this makes it possible to power-on/off a node, by simply activating/deactivating the PoE support on the corresponding switch port.

We performed a two-month extensive research for possible hardware products that could serve as our network nodes. After experimenting with various hardware possibilities, we finally decided that the Soekris net4826 [6] box fulfills our requirements the most. This compact board is based on a 266 MHz i586 processor. It has one 10/100 Mbit Ethernet port, 64 MB SDRAM main memory and uses a CompactFlash circuit soldered onboard for program and data storage. It can be expanded using up to two miniPCI boards. Also, with a small effort, the board can support USB if needed in the future.

We matched the net4826 board with a flexible miniPCI WiFi card. Since we seek to perform research on wireless networking and communications, we want to be able to tune most of the 802.11 MAC protocol parameters [7], as well as implement additional functionalities. Moreover, we wish to be able to work with different versions of 802.11, and have a large coverage range with high transmission power. We tested various wireless cards for several weeks, and we finally decided on the EMP 8602-6G 802.11a/b/g miniPCI cards. These cards embed the Atheros AR5006 chipset [8], which is controlled by the MadWifi driver [9]. This driver implements most of the 802.11 MAC functionality [7]; hence, it is easy to modify the driver code in order to change parameters, or implement new features. In addition, being able to work with different 802.11 versions is helpful, especially if other wireless networks co-exist in the same region. In particular, we have deployed our indoor testbed in our campus building, wherein part of the university's 802.11g wireless network is also deployed. By switching to 802.11a, our network is able to avoid interference

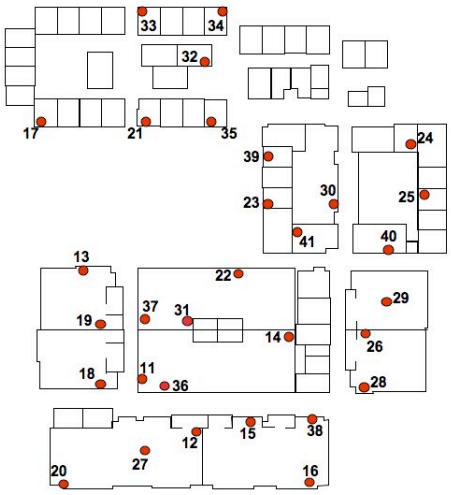

Fig. 1. Our indoor–testbed deployment. Nodes are represented by dots along with their IDs.

from this co-located 802.11g network. Finally, after a thorough search for external antennas, we decided to utilize RD2458-5-RSMA 5-dBi gain Triband Rubber Duck antennas.

Why not use a PC as node: At this point, one may simply ask: *"Why not utilize a trivial desktop or laptop PC as a node?"* We admit that this was our first thought, and we experimented with this option for quite some time. Due to the increased cost, limited extendability and the theft probability of a laptop, we focused on a desktop PC testing. However, we soon concluded that a desktop PC is not a good choice, for three main reasons.

• First, it is difficult to remotely power on/off the desktop PC. Even if the motherboard and the operating system support such a functionality, there will be a problem whenever the operating system hangs. In such cases, one would have to manually reboot the nodes-PCs; this becomes difficult and time consuming, especially if nodes are deployed in rooms where access is not always granted.

• Second, a cheap desktop PC usually occupies quite a lot of space. Moreover, some PC boxes start making noise after a short period of time.

• Third, for most desktop PCs it is not efficient to have more than one wireless card plugged-in. More specifically, in most motherboards the PCI slots are very closely placed to each other. For PCI wireless cards that are connected to external antennas, there is significant signal leakage at the antenna's mount point on the card [10]. In addition, if two wireless cards are closely placed into two parallel card slots, and if both are active at the same time, electromagnetic waves enter from one card to the other. This largely degrades the performance of each card, even if the two cards operate on two different channels. To verify this, we installed two wireless PCMCIA cards into a desktop PC, using PCI→PCMCIA converters. We used two laptops as receivers. We also installed the same type of cards into the laptops. We set the cards to various different channels and used the *ttcp* networking tool to perform a set of simple experiments. We first ran some tests with only one card (installed in the PC) active at a time, and we measured the achieved TCP throughput towards one of the laptops. Furthermore, we ran tests with both cards (installed in the PC) active at the same time. We observed that, even when the two cards were set to 802.11a and 802.11g respectively, the achieved throughputs were almost half of what each card could achieve in isolation!

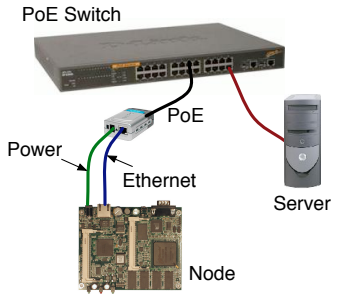

Fig. 2. Connection between a node and a switch in our testbed.

Thus, we decided that a dektop PC cannot serve as a node. However, for very small-scale testbeds, deployed inside a room wherein size and noise are not issues, this appears to be a feasible solution. It appears to be restricting, however, if one decides to use more than one WiFi card on the PC. The Soekris box that we are using was designed with this aspect in mind.

Moreover, in order to be able to manage nodes from a central location, we use a desktop PC (Pentium IV at 1.8 GHz) as a *server*. Each node is connected to the server through its Ethernet interface. A set of D-Link-DES-1526 PoE switches connect the nodes with the server and also provide power to the nodes[2]. Through the server we have total control of both the individual nodes and the network as a whole. The server can connect to each node, via the switches, through a secure shell *(ssh)* and configure the node as preferred. Note here that, since the net4826 does not support the same IEEE 802.3af PoE standard as our PoE switches, we use a D-Link-P50 PoE→12V adapter for each node (figure 2). This adapter isolates the power provided by the PoE switch and forwards it to an output towards the net4826. It also forwards the network data from/to the Ethernet interface of net4826. Finally, before deployment one should make sure that there is a network connection between the switches. We placed our switches in rooms (network closets), which housed the building's switches. These closets are interconnected with optical fibers only, and not with regular network cabling (i.e., cat5). Hence, we had to find a way to provide connectivity among the switches. For this, we used a CVT-100BTFC 10/100 Base-TX to 100 Base-FX Media Converter for each switch. The network connection is pictorially represented in figure 3.

B. Software configuration

Software installations and configurations are also a very important and time-consuming part of the testbed development. The healthy operation of the network testbed depends on the integrity of both the operating system and the applications that the nodes are running. Hence, mature software design decisions have to be made; these decisions should take into account the following aspects:

[2]Note that in order to provide power to the nodes, we only have to connect them to the PoE switches.

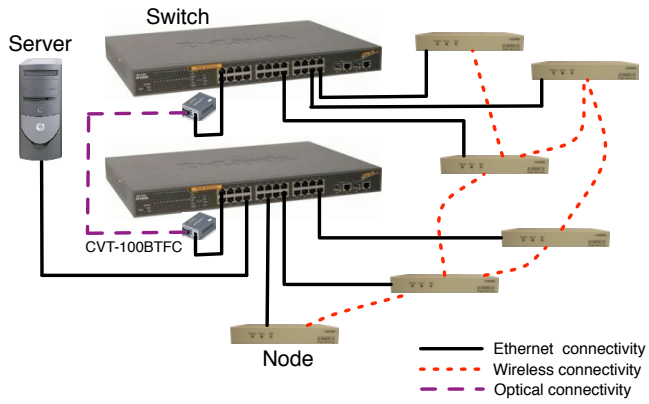

Fig. 3. Network inter-connectivity with our testbed: a pictorial representation. The switches are connected through optical fiber. For simplicity we do not show the D-Link-P50 adapters; this detail is depicted in figure 2.

Lightweight: The nodes are not very powerful in terms of CPU and memory; due to budgetary constraints, heavy duty machines were not purchased. As a consequence, the operating system should not occupy all the available system resources. On the other hand, a large variety of free useful software tools should be supported.
Easy configuration: We want to have the ability of modifying many aspects of the software, both in terms of implementation and parameter tuning. Thus, the software should allow for such modifications.
Security: The system should be as secure as possible. Even though this is not a primary issue with our testbed (since it lies under the umbrella of our department's firewall) it should be a concern in general.
Uniformity: All nodes should run the exact same version of the operating system, drivers and applications. Moreover, frequent updates and software modifications should be mapped to all nodes.

1) The operating system: We address the above factors in a combined solution. We decide that nodes should run a Linux distribution, which is well-documented and widely accepted. In this way, most configuration issues can be easily resolved, mostly through a quick Internet search. Since nodes must run the exact same Linux configuration, the easiest way is to have them load the same Linux kernel, at every boot. This provides us with a set of advantages. First, we may perform any software updates or configurations centrally, i.e., at the server side. Whenever a node boots the Linux image, it will also load our latest modifications. Thus, every time a researcher implements a scheme, this will be stored in the server and loaded by all nodes at boot. Second, since the Linux distribution is located at the server, all nodes load a copy of a Linux image at network boot. Third, each node needs to load the Linux kernel, as well as a small set of modules in its main memory. In other words, the number of files that each node loads at start-up is relatively small, and occupies only a small portion of the node's main memory. Fourth, we are able to load different wireless driver versions to some nodes. As an example, if we wish to create a WLAN topology, we may centrally decide which nodes will be the access points, so that we have these nodes boot the AP driver, and the rest of the nodes boot the client driver. Last (but not the least), each node is able to load all the required files into its main memory; *hence, nodes need not be equipped with hard disks*. For this, we have modified the Debian v3.1 Linux distribution with kernel version 2.6.16.19, to fit the needs of our testbed. Nodes load the Linux kernel over NFS *(Network File System)* , which securely allows machines to mount a disk partition on a remote machine, as if it were on a local hard drive.

Having diskless nodes provides a twofold advantage: (a) The cost per node becomes much lower. (b) The hard disk is probably the most vulnerable hardware component, since it is comprised of mechanical parts. Simply put, having a diskless system, without any mechanical parts, implies lower risk of node damage in the long run. Hence in our testbed, the diskless nodes retrieve their Linux kernels and mount their root directories directly from the central server. They can also write into the server's disk, through NFS. This means that a researcher can maintain his/her own, independent experimental setup, including the kernel and every component of the distribution. We use a separate desktop PC to compile the kernel, modules, applications and drivers that will be used on the testbed. This desktop PC runs a Linux distribution that is configured in the same way as the one loaded by the nodes.

The testbed server runs a separate Debian v3.1 Linux distribution (even though any unix distribution can be used for the server). The server needs to run the NFS [11], BOOTP [12] and TFTP daemons. Our server assigns an IP address from a pool of addresses to each identified node-client. Each node obtains an IP address prior to loading its Linux distribution. For security reasons, the wired Ethernet segment of our testbed is separated from the regular building network. All the Ethernet interfaces on our wired testbed network have IP addresses in the private 10/24 range, i.e., 10.0.0.*. The server's IP address is 10.0.0.1[3], while the IP addresses 10.0.0.2 – 10.0.0.10 are reserved for potential future servers. Furthermore, the IP addresses 10.0.0.11 – 10.0.0.41 are currently assigned to nodes. Finally, we are using two PoE switches, with IP addresses 10.0.0.253 and 10.0.0.254. For convenience, we have also assigned IDs to nodes, which map to the last 8 bits of their IP addresses. For example, the IP address 10.0.0.31 is assigned to the Ethernet interface of node 31. These IP addresses are static, i.e., the assignment is fixed. In addition to the Ethernet address assignment, we have also assigned IP addresses to the wireless interfaces of the nodes. Addresses for these intrefaces are assigned in the private 192.168.1.* subnet, using the same last 8 bits as in the wired address. Hence, the IP address 192.168.1.31 is assigned to the wireless interface of node 31.

C. Network deployment

In an indoor setting, obstacles such as walls, furniture, doors and people affect the signal strength to a large extent. In addition, the environment is highly variable: people move often, doors open and close, metallic window shades absorb/reflect the signal, and nearby devices (microwaves, portable telephones etc.) interfere randomly.

All the above factors need to be taken into consideration, every time a new node is initiated into the testbed. The level of environmental variability determines the network connec-

[3]Note that the server is equipped with two Ethernet interfaces; one for the testbed (with IP address 10.0.0.1) and one for connectivity to the outer world.

tivity, and the created topology. There are three strategies that could be applied during deployment.

First, one could place nodes randomly, regardless of the environmental variability. Given that there exists an available Ethernet port at the selected location[4], one must further only rely on the maximum transmission power required to connect to other nodes. This approach however is quite risky. Even if the WiFi card is very powerful, we may not always want nodes to transmit with very high power (to be discussed). As a result, with lower power, a randomly placed node may lose connectivity from the rest of the network.

Second, one could take into account the variability and the obstacles around the potential location. For example, one may avoid placing a node near a door that people open and close frequently. In addition, microwave ovens and portable phones (2.4 GHz band) may also cause significant interference to nearby testbed receivers. Even though the environmental variability is unpredictable to a large extent, with this approach one should try to reduce its effects as much as possible, by placing nodes away from such devices.

Finally, one could actually take advantage of the environment, whenever this is possible. For example, one may deliberately place a node behind an obstacle, in order to create a desired topology.

With all of the above possibilities, one should consider the maximum transmission power that can be supported by the wireless cards. The best strategy for deployment from the above is the one that can provide both dense and sparse topologies, whenever this is desired. Depending on the available budget and building resources, one may place a very large number of nodes, close to each other. This can probably guarantee good connectivity, and it is a safe approach, since not all nodes are required to be powered-on at the same time. Hence, a different set of active nodes creates a different topology each time. If however, the deployment of many nodes is not feasible, a careful placement of nodes is required, and this task is not trivial at all. During the deployment of our testbed, we came across situations where the connectivity was not a given, even though we expected that it would be. As an example, we experienced cases, in which the distance between two nodes was 5-10m, and two glass windows only existed in between; however the link between the nodes was either too poor, or non-existent (e.g. links 11→18 and 39→35 in figure 1). In other cases, a distance of 20-30m with walls, desks and people in between could not affect the good connectivity between nodes (e.g. link 20→16). In addition, multiple simple tests were performed to observe the connectivity at different network instances. We describe some of them in the next sub-section.

D. Some pitfalls and mistakes to avoid

So far we have described the hardware and software components, as well as the deployment efforts with our testbed. We have also commented on some design and development problems that we have faced. Here we deliberate on the difficulties one may expect and the mistakes to avoid, during the design and set-up of a wireless testbed, based on our experiences.

[4]Remember that an available Ethernet port is required to connect the node with the PoE switch.

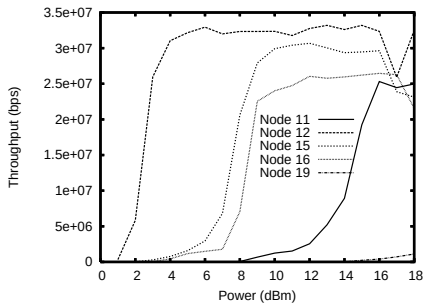

Fig. 4. Achieved throughputs on the links between node 20 and its neighbors. We observe that the maximum transmission power *(18 dBm for this channel (56))* affects the reliability of the results.

a) Hardware: The correct functionality of the hardware should be verified early. After assembly, each node must be tested thoroughly before deployment. This will make debugging much quicker when a problem appears (e.g. when a set of nodes suddenly loses wireless connectivity from the rest of the network).

Furthermore, the capabilities of the wireless hardware, in terms of transmission power, must also be verified. Some wireless cards cannot transmit with the same maximum power on all of the frequency bands. For example we observed that when set to channel 56 (5.28 GHz), the maximum transmission power of our WiFi card can be up to 18 dBm. However, if we set the card to channel 11 (2.412 GHz), the maximum supported power can be 19 dBm. More significantly, we observed the following two phenomena, with regards to power settings:

1) When two nodes are placed very close to each other (i.e., less than 8 meters apart) without obstacles in between, then they should not communicate with high power. We placed two nodes 3m apart, selected channel 56, set the transmission power to 15 dBm, and ran a set of high-volume TCP traffic experiments from one node to the other. We observed that the achieved throughput was extremely low. By setting the transmission power to 1 dBm, we further observed that the link throughput reached its maximum possible value. *We experimented with different cards and different frequencies; we observed the same phenomenon*. We believe that this is happening because the receiver is saturated, i.e., the signal is too strong for the A/D converter, and the gain control circuitry cannot compensate this effect.

2) For some wireless cards, it is better not to use a transmission power close to the maximum value due to potential manufacturing defects. We observed this, while running a set of fully-saturated UDP experiments between a large number of links. As an example, we conducted a large number of experiments on the links of node 20, on channel 56, with different transmission powers. Only one link was active at a time, i.e., links worked in isolation. We observed that, even though the maximum achievable power is 18 dBm, reliable results can be produced with powers of up to 16 dBm. This is depicted in figure 4. This figure presents the achieved throughput for each link from node 20 to each of its neighbors. For powers larger than 16 dBm we observe that the results are not reliable, for all of the links under investigation. Hence, the transmission power of the wireless card, at values very close to the maximum, should be tested before being used.

b) Density of deployment: The strength of the transmitted signal by a node and the reception quality should not be the only concerns. Increasing the network density in order to have a good connectivity is a pitfall: the signal strength received by a neighbor will be higher, however the interference and contention in the network will also increase. If the receiver sensitivity (carrier sensing threshold) can be tuned, however, then the level of interference and contention will change accordingly. Note, however, that many WiFi cards do not allow the tuning of the receiver sensitivity.

Moreover, as one may expect, the network connectivity is different, for different modes of operation (i.e., 802.11b or 802.11a). Even if we keep the transmission power fixed, the modulation technique used with 802.11a (Orthogonal Frequency Division Multiplexing - OFDM [13]) is different from that with 802.11b (Complementary Code Keying - CCK [7]); this may affect the maximum distance that the signal can travel. Due to the higher carrier frequency of the 802.11a the signal cannot penetrate as far as with 802.11b [14], since it is more heavily absorbed by obstacles. Indeed, we observed that the connectivity problems mentioned above, for links 11→18 and 39→35, do not exist for the 802.11b mode of operation. In addition, however, we observed that the node degree is now much larger, even more than double in some cases. As a result, one should have a knowledge of the achievable connectivities on the different channels. Finally, note that the materials that make up the obstacles play a significant role in dictating the achieved topology. For example, it is more difficult for the signal to penetrate a brick or metallic wall, than a wooden one. Perhaps the best deployment strategy is the most costly one: deploy a large number of nodes to guarantee connectivity, and use different subsets of nodes to achieve different network connectivities, with different levels of interference.

III. Experimental Evaluation

In this section, we present experimental results on the stability of our network testbed. For these experiments we have set the wireless interfaces of the nodes to the 802.11a mode of operation, in order to avoid interference from the only co-located 802.11g campus network. For this set of experiments we use channel 56. We do not use the maximum power provided by the WiFi cards, for the reasons that were discussed in the previous section. Furthermore, for these experiments we use identical power settings on both the transmitter and the receiver node of a link. In order to get reliable results, we run a large number of tests between different links, during different periods of the day, as well as on different days. We use 30 seconds of fully-saturated 1500-byte/packet UDP traffic.

We experimented with various transmission power levels; here we present results with the power set to 16 dBm. As one may observe in figure 5, link throughputs remain relatively stable over time. This is especially the case for high quality links, such as the link 28→29. On the other hand, poor links present a slightly higher variability. This is an effect of the variable environment of deployment, as was explained in the previous section. As an example, the link 13→18 has a poor quality; this is why we observe such a low throughput, i.e., at least 5 times lower than for the other links. For certain time periods we observe a throughput spike (for 13→18), such as at times 50 min, 120 min and 470 min of operation.

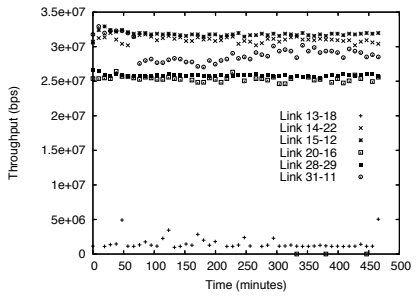

Fig. 5. The link throughput values remain almost the same over time. For relatively poor links we observe an expected, small variability in throughput.

At these instances, the throughput grows even as much as 5 times that of the nominal values. Even though the efficiency of poor links is largely affected by noise, the variability of the environment plays a very important role as well. For high quality links however, we observe that the throughput does not vary significantly.

IV. Conclusions

We described our experiences on the design, set-up and deployment of a wireless network. We deliberated on the hardware components, the software configuration and some basic deployment strategies. We also discussed the difficulties and pitfalls that one may face, when building a similar indoor testbed.

A network testbed, when designed and administered properly, can provide new insights on any kind of networking and communications research. Many basic assumptions made for analytical and/or simulation studies are likely to be invalid in a real network deployment. However, in order to serve its purpose, the testbed must first be examined for its correct functionality. The hardware components must be thoroughly tested and the software must be properly configured. In addition, nodes need to be deployed taking into account the environmental variability, the maximum transmission power supported, as well as the created network topology and connectivity for different power levels.

References

[1] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MOBICOM*, 2005.
[2] S. Sanghani, T. Brown, S. Bhandare, and S. Doshix. Ewant: The emulated wireless ad hoc network testbed. In *IEEE WCNC*, 2003.
[3] N. Vaidya, J. Bernhard, V.V. Veeravalli, P.R. Kumar, and R.K. Iyer. Illinois wireless wind tunnel: a testbed for experimental evaluation of wireless networks. In *ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, 2005.
[4] E. Nordstrom, P. Gunningberg, and H. Lundgren. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *TRIDENTCOM*, 2005.
[5] UCR Wireless Networking Research Testbed. http://networks.cs.ucr.edu/testbed.
[6] Soekris/net4826. http://www.soekris.com/net4826.htm.
[7] ANSI/IEEE802.11-Standard. 1999 edition.
[8] Atheros/AR5006chipset. http://www.atheros.com/pt/ar5006bulletins.htm.
[9] MadWifi-Driver. http://madwifi.org.
[10] J. Kaba and D. Raichle. Testbed on a desktop: Strategies and techniques to support multi-hop manet routing protocol development. In *ACM Mobile ad hoc networking and computing*, 2001.
[11] RFC 3010. NFS version 4 Protocol.
[12] RFC 951. The Bootstrap Protocol.
[13] ANSI/IEEE802.11a-Standard. 1999 edition.
[14] http://en.wikipedia.org/wiki/IEEE_802.11 text.