Experiences with BOWL: Managing an Outdoor WiFi Network (or How to Keep Both Internet Users and Researchers Happy?)

T. Fischer, T. Hühn, R. Kuck, R. Merz, J. Schulz-Zander, C. Sengul
TU Berlin/Deutsche Telekom Laboratories
{thorsten,thomas,rkuck,julius,cigdem}@net.t-labs.tu-berlin.de,ruben.merz@telekom.de

Abstract

The Berlin Open Wireless Lab (BOWL) project at Technische Universität Berlin (TUB) maintains an outdoor WiFi network which is used both for Internet access and as a testbed for wireless research. From the very beginning of the BOWL project, we experienced several development and operations challenges to keep Internet users and researchers happy. Development challenges included allowing multiple researchers with very different requirements to run experiments in the network while maintaining reliable Internet access. On the operations side, one of the recent issues we faced was authentication of users from different domains, which required us to integrate with various external authentication services. In this paper, we present our experience in handling these challenges on both development and operations sides and the lessons we learned.

Keywords: WiFi, configuration management, authentication, research, DevOps, infrastructure, testbed

1 Introduction

Wireless testbeds are invaluable for researchers to test their solutions under real system and network conditions. However, these testbeds typically remain experimental and are not designed for providing Internet access to users. In the BOWL project [2, 7, 9], we stepped away from the typical and designed, deployed and currently maintain a live outdoor wireless network that serves both purposes. The benefits are twofold [9]:

- University staff and students have outdoor wireless network access. Our network covers almost the entire Technische Universität Berlin (TUB) campus in central Berlin (see Fig. 1).
- Researchers have a fully reconfigurable research platform for wireless networking experimentation that includes real network traffic (compared to synthetic traffic).

During its lifetime, the BOWL network has significantly evolved from a prototype architecture and design in 2009 [7, 9] towards a production network, which brings out several administrational and development challenges. The network and its components, including traffic generators, routers and switches interconnect with a variety of other networks and infrastructures which are not controlled by the BOWL project, adding to the inherent complexity of running a production network. In this paper, we focus on two of our many challenges that we have experienced in the last year while moving from a prototype to a more stable infrastructure. We present our challenges from the perspective of development and network operations and its reliance on external services, respectively.

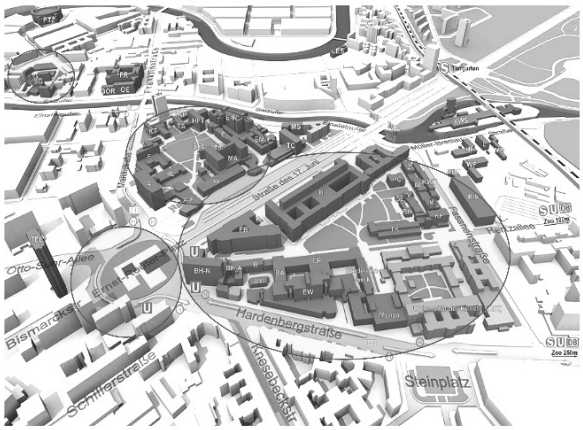

Figure 1: Coverage of the BOWL network on the TU-Berlin campus.

Development challenges were - and still are - numerous [9]. The most prominent is the variety of people that work on different subsets of network components, and change network configuration and operating system images. The requirements for associated services and infrastructure, as well as the research goals, continuously change as we and other users change the way the BOWL network is used on a daily basis. In fact, our experience showed that it was necessary to rewrite the BOWL software significantly during the development as well as the operational lifetime of the BOWL project. Many of the changes were also triggered with the feedback received from external users.

From a purely operational point of view, authentication of users to the BOWL network has proven surprisingly complex. A project-specific remote authentication dial-in user service (RADIUS) installation is used as the

pivot point to integrate a number of other distributed and disparate authentication solutions. Users include (1) centrally managed university IT accounts, (2) users from our own department, (3) users of the affiliated external institution Deutsche Telekom Laboratories (hereafter T-Labs), (4) project-only user accounts, and (5) eduroam users. TUB user authentication is a critical part of the contractual relationship with the university central IT department. The major challenge we faced and still face is the recovering from errors that might lie in external authentication services that we rely on to support these accounts. In this paper, we present a major outage we went through due to such problems and the lessons we learned.

2 BOWL (Berlin Open Wireless Lab) and DevOps Challenges

The main task of the BOWL project is to satisfy two somehow conflicting requirements from two user groups – Internet users and researchers (which are often developers). We see the following requirements as DevOps challenges:

- **Researchers demand a configurable network (development):** The testbed is intended for a wide selection of research topics ranging from enhancing measurement-based physical layer models for wireless simulation [8] to routing protocols [11, 12]. Hence, one of the goals of the BOWL project is to allow multiple researchers to access the network, deploy experimental services, change configurations and run new experiments or repeat old experiments while still ensuring Internet access. Therefore, the BOWL project required the development of several tools to automate software and configuration deployment in the testbed. We discuss our experience with these tools, and how they evolved in Section 4.
- **Internet users demand a reliable network (operation):** Changing the network configuration, deploying and running experiments should not affect the availability of Internet access. This implies that basic connectivity should not be affected, or only for a negligible time duration. It also means that services such as authentication, DHCP and DNS need to remain available in any experiment setup. How BOWL network architecture addressed this problem is summarized in Section 3. A major operational challenge is the authentication of different type of users (e.g., Internet users from TUB and T-Labs, and researchers) to the BOWL network, and we discuss this in detail in Section 5.

3 BOWL Network Architecture

In addition to its outdoor network, the BOWL project is in charge of two additional networks: (1) a smoketest network, for early development and testing and (2) an indoor network, for small-scale deployment and testing. These networks are used for development and staging before a full-scale deployment and measurements in the outdoor network. Therefore, the research usage pattern of the outdoor network is more bursty, with periods of heavy activity followed by lighter usage, whereas the smoketest and the indoor networks have been in heavy use since their deployment in early 2008. In this paper, we mainly focus on our experience with the outdoor network.

The BOWL network architecture was first presented in [9]. In this section, we summarize this architecture to give the necessary information to understand the BOWL environment and its challenges. The outdoor network comprises more than 60 nodes deployed on the rooftops of TUB buildings. It spans three different hardware architectures (ARM, MIPS and x86). Each node is powered by Power over Ethernet (PoE), which simplifies cabling requirements. All nodes are equipped with a hardware watchdog, multiple IEEE 802.11a/b/g/n radio interfaces and a wired Ethernet interface. One radio interface is always dedicated to Internet access, the additional radio interfaces are free to be used in research experiments, and the wired interface is used for network management and Internet connectivity. All nodes are connected via at least 100 Mbit/s Ethernet to a router that is managed by the project. A VLAN network ensures a flat layer 2 connectivity from our router to each node. Our router ensures connectivity to the BOWL internal network, the TUB network and the Internet. In its default configuration (which is called the *rescue* configuration), the network is set up as a bridged layer 2 infrastructure network. Association to the access interface and encryption of the traffic is protected by WPA2 (from the standard IEEE 802.11i [1]). Authentication is performed with IEEE 802.1x and RADIUS.

Each node runs OpenWrt [5] as the operating system. The OpenWrt build system typically produces a minimally configured image. To tailor this image to each node, the image is configured at boot time by an auto-configuration system that applies a so-called *configuration* to the image. A configuration includes all the configuration files that go under the `/etc/config` directory (the layout is specific to OpenWrt), and additional files, scripts and packages that may be needed by the experimenter. The details of the auto-configuration system are explained in Section 4.

By default, every node runs a default *rescue* image and uses the aforementioned *rescue* configuration. Researchers install *guest* images in extra partitions and

use *guest* configurations. Because of the unique needs of experiment monitoring and reconfiguration at runtime, a network management and experiment monitoring system was developed, which also went through significant changes from its version presented in [9]. In essence, it comprises two main components: a node-controller, which runs on each node and a central node-manager. Each node-controller connects to one node-manager. However, with the recent changes, several node-managers can be now run in parallel i.e. one for each experiment if several parallel experiments are needed to be run or for development. Our typical operation requires one node manager per network (e.g., smoketest, indoor and outdoor). Thanks to the underlying VLAN infrastructure and virtualization of the central router, the traffic generated by each experiment can be isolated, if multiple experiments are running in the network. More details on this topic can be found in [9].

Unwanted side effects due to using experiment software (e.g., crashes, slowing down of network services) are expected to occur in practice but their effect needs to be minimized as much as possible. This is achieved thanks to the locally installed images. Indeed, a node that is experiencing problems can be *rescued* by an immediate reboot into the rescue image. This mode of operation is implemented making use of hardware and software watchdogs that periodically check that certain services are operational. One example is that, node-controllers at each node periodically check connectivity to the central node manager and when a disconnection is detected, the node is rebooted to the rescue image within 60s. Note that since each node independently triggers a switch to the rescue mode based on its own hardware and software watchdogs, nodes do not go down all at the same time limiting network disruptions. More details on how experiment problems are detected can be found in [9].

In the remainder of the paper, we focus on how we addressed two main challenges: the development challenge of supporting multiple network configurations for different researchers and the operational challenge of authentication in the BOWL network.

4 A Development Challenge: Supporting Multiple Network Configurations for Wireless Experimentation

One of the main goals of the BOWL project is to allow multiple researchers to create experiments, and be able to run and repeat their experiments in a consistent fashion. In the remainder of this section, we first summarize the system that we started off with around mid 2008, and describe how it evolved during the lifetime of the project. Essentially, the reliability of the

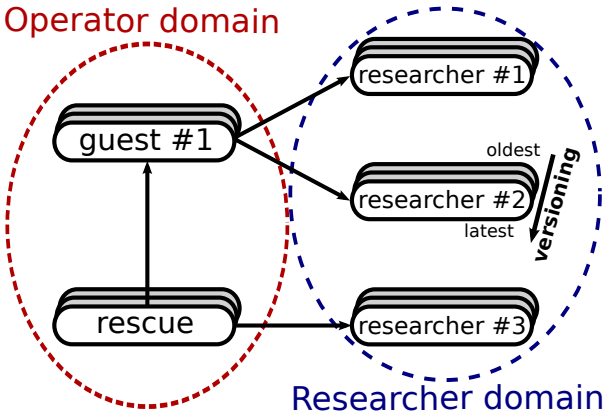

Figure 2: An example of how three researchers maintain their own configuration in the BOWL network.

BOWL network was jeopardized due to several configuration glitches and therefore, our complete software rewrite decisions were significantly affected by the need to maintain network reliability at all times.

The node configuration of a given experiment consists of two parts: (1) an operating system image and (2) an experiment configuration. OpenWrt manages the whole configuration of the operating system using the universal configuration interface (UCI)[6]. We also take advantage of the UCI. As the network is used for very different purposes, it becomes necessary to maintain consistent network configurations across the users. Therefore, initially, we had a configuration database and stand-alone scripts to apply these configurations from a central server manually. As more nodes were deployed in the BOWL network, it became a necessity to have a more scalable and manageable solution. To this end, the existing node-manager and node-controller framework was extended to support node configurations. The important components to a BOWL user are: (i) the web-based front-end to a configuration database, and (ii) a client-server auto-configuration process that runs in node-controllers and the node-manager, respectively. The auto-configuration scheme was added after mid 2010 due to the several failures that occurred with the earlier version. Figure 2 illustrates how, for instance, three researchers maintain their configurations in the BOWL system.

Using the web-based front-end, a researcher can pick a configuration, image and the node partition to deploy its experiment. From this step on, the user flashes his own image to this partition and nodes are configured by the auto-configuration process at boot time (or before the image is booted). However, currently, a researcher still needs to record the information about which image was used with which experiment configuration. In the future, we are planning to automate this lab bookkeeping

process. Finally, a reservation system prevents node and image usage conflicts. Currently, the reservation system used in BOWL is primitive, in the sense that the entire network is reserved to a single researcher for a given period of time. Each researcher is responsible of his image and configuration and deploys this image to a given node partition. Hence, merging of multiple images from different experimenters is not expected.

This framework, complete with a new auto-configuration scheme, is in use since mid 2010 by the BOWL group and visiting researchers, that also remotely access our network. We learned several lessons since then, which resulted in the current state of the framework as we use today. For instance, one issue resulted from the inheritance of configurations in the database. It was not obvious to us at the beginning that researchers would have difficulties discovering the inheritance hierarchy. But some of our early users applied changes to the base configuration expecting them to take effect in the descendant configuration. To avoid such problems, we now expose the inheritance hierarchy to the users of our system and visualize it in the web-based front-end. Finding a right way to do this also was a challenging task. Furthermore, being too accommodating was not a good idea and we ended up limiting the functionality of the web-based front-end. Earlier, researchers could push a configuration to a given node by just pressing a button. However, since installing images and configurations were separated from each other, it sometimes resulted in applying a wrong configuration to the wrong image. Therefore, we removed this functionality from the front-end. Actually, this was the main reason why an auto-configuration scheme was added to the system. A final lesson learned was not to assume any network stability during configurations. With our first auto-configuration implementation, the nodes fetched their configurations from the node-manager right after booting. However, if there were any network instabilities during this time, the watchdog would trigger and interfere with the auto-configuration. We now avoid this problem by having nodes first fetch their configurations before booting the image, configure the image, and boot only if all checks pass. While our development activities have slowed down as users become more used to working with our framework, we are still looking into simplifying things even further to lower the entry barrier of using the BOWL network.

5 An Operational Challenge: Authentication in the BOWL network

In exchange of the rooftop usage and installation support, the BOWL project has contractual obligations with TUB to provide wireless Internet access to staff and students. Hence, we need to provide the usual authentication and accounting services that would be expected from any WiFi access network. To this end, we use the widely deployed FreeRADIUS software [4], which is a server implementation of RADIUS [10]. When a user tries to authenticate to our network, the authenticator (hostapd) at the WiFi access point communicates with the RADIUS server. Using challenge-based protocols, the RADIUS server determines whether credentials provided by a user are valid. Using the results from this decision process, the access point either allows the user to join the network or rejects him.

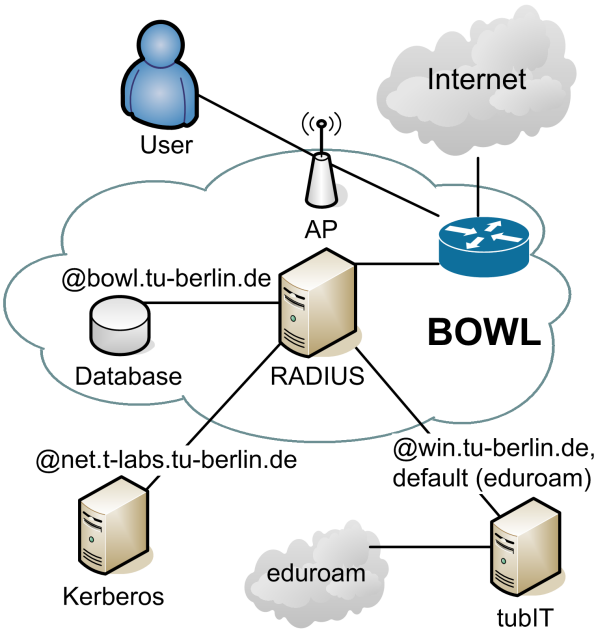

Figure 3: Logical diagram of the BOWL authentication infrastructure.

One of the main reasons that makes authentication in the BOWL network a challenging task is the interconnections with other networks and the need to provide access to different type of accounts. FreeRADIUS does support this by allowing access decisions based on local account databases or using the results of requests proxied to further upstream services, which may in turn again be other RADIUS implementations or entirely different services. Currently, the BOWL network needs to provide access for the following types of accounts (see Figure 3):

- **TUB accounts** as held by students and members of staff in another RADIUS server, administered by TUB. Access is provided using PEAP with MSCHAPv2. The BOWL network does not hold (or ever sees in any other way) passwords associated with these accounts, because it just proxies the encrypted challenge and response messages.
- **eduroam** [3] access is provided by TUB using the

same scheme as described above. Accounting data for this and the previous scheme are forwarded to TUB.

- **Accounts for the local department FG INET**, administered by the department of which BOWL is a part. The upstream authentication service is a Kerberos installation. Access is provided using TTLS with PAP, because this kind of upstream service requires that the FreeRADIUS server handles the passwords of the users.
- **Local accounts** for demonstration and guest access purposes, administered by the BOWL network. Implemented using PEAP with MSCHAPv2. Contrary to the previous schemes, all schemes available as default settings in FreeRADIUS provide working options here. The credentials are held in a local database.
- **Experiment-specific accounts** for researchers, administered by the BOWL network. Implemented in a vein similar to the local accounts. These special accounts are available for us to be able to filter out data about traffic generated for the purpose of experimentation from the accounting database.

From this list it follows immediately that support requirements towards users tend to vary with upstream authentication source. Administration and support complexity inevitably increases rapidly with additional supported schemes. This complexity which results from the highly interconnected nature of BOWL is only bound to increase. For example, there are discussions whether some parts of Deutsche Telekom Laboratories are to be provided access to BOWL using a limited subset of the accounts held in an Active Directory service. Also, there are plans to move local accounts into a LDAP installation for centralized administration.

In the process of creating all these authentication interconnections, we have learned that unlike some other pieces of server software, FreeRADIUS makes it somewhat difficult to set up a fresh installation with self-written configuration files, because of the inherent complexity of the flow of authentication requests within the server. The developers make a point of telling their users to proceed only from the default settings, making small incremental changes. Therefore, keeping the configuration files in a version control system has proven to be even more invaluable than with any other service. In summary, FreeRADIUS setup and handling can be daunting and time-consuming for the administrator who works with it extensively for the first time. However, we still feel that we have made the right choice. The software is freely available under the terms of the GPL, it works without any need for modification on the BOWL network and it provides an extremely rich feature set.

Now, monitoring of availability of external authentication services has become one of our major challenges, which requires working test accounts for those services. Monitoring software like Nagios provides support for self-written plug-ins, but not all upstream service providers are prepared to provide such accounts. Testing installations are needed, but they are hard to realize as they require testing configurations on live nodes. Furthermore, the upstream providers may be required to accept and serve requests from these testing installations. Also, obviously, it must be avoided that the accounting database is not polluted by bogus/testing data. All of this must be done carefully, as FreeRADIUS has proven to be a piece of software to which configuration changes need to be made with special care because of unintentional interactions with other configuration sections.

One important consequence from not being able to fully test and monitor external authentication services is the loss of usage of the network. This is quite annoying when it is due to problems in external services that we do not fully control. And loss of control is not just a hypothetical scenario. During the spring of 2011, no TUB users were able to authenticate to the BOWL network. Local testing revealed that the reason did not lie in the BOWL network installation; requests were passed on to the upstream server correctly. The fact that all authentication protocols in use are encrypted and stateless made further debugging difficult. The hospitalization of our main technical contact person at TUB, who was also the only person knowledgeable about the RADIUS configurations, at exactly this point in time put another obstacle in our way to successfully resolve this issue. Eventually, it was found that a server certificate of one of the upstream servers had expired, leading to rejection of user authentication attempts. Luckily, the BOWL network bounced back from this incident, and we observed a speedy uptake by users again shortly afterwards. The first power users returned the morning after the upstream servers were fixed; the number of distinct users increased continuously and two weeks later, the number of distinct users per day peaked.

The most that an operational team can do in these cases is to rely on its own monitoring tools in order to be able to find the source of problems as quickly as possible; and to build open and positive relationships with upstream operations teams that make communication and collaboration as smooth as possible. We also noticed that solving the problem was delayed due to the unavailability of the only person with the know-how. Based on this experience, on our side, we try to make sure that the BOWL system knowledge is shared among multiple people, who can handle issues independently.

6 Current State and Lessons Learned

To manage a live and experimental testbed is a significant challenge, as one needs to keep both Internet users and researchers happy. In this paper, we described the auto-configuration and authentication solutions that we run to be able to serve both communities.

We learned several lessons during this phase, which we summarize as follows:

1. It is important to have complete and thorough documentation that details the know-how of the BOWL project group. Using our system for the first time is currently not trivial. Therefore, more time needs to be invested in educating future users and simplifying operation.
2. Early adopters of the BOWL framework proved that people always find a way to use an interface differently than you expect them to. Well-defined user interfaces with less functionality turned out to be much more useful than providing more functionality with specifications unclear to the user. Therefore, it is better to design simple first, and add extra functionality when only it is absolutely required by the users.
3. While building the BOWL framework, we once more realized how important user-friendly interfaces are. People should be exposed all the necessary information to run the system correctly easily.
4. In a live network, network disruptions will happen. Therefore, all functionality should be designed around issues that can rise from network instability.
5. Our authentication problems showed that the most important thing is to maintain a good contact with all the parties that can affect operation. More than expected, the problem lies outside our own network, and we need to rely on problem solving skills of the upstream service providers.
6. FreeRADIUS configuration changes should be maintained in a version control system. This makes it a lot easier to revert to a previously working version.
7. The complexity of any important component of the network, such as authentication services, is only going to increase as the number of interconnections increases. Being aware of this fact aids in the planning of upcoming changes and aids with the integration into previously existing configuration options.
8. Finally, we learned that it is essential not to create information bottlenecks in a project team, and there should always be multiple people who know how to handle problems independently of others.

7 Acknowledgments

We thank Harald Schiöberg for the architecture of the BOWL testbeds and the implementation of the original BOWL software suite. This work was supported by Deutsche Telekom Laboratories, in the framework of the BOWL project.

References

[1] 802.11-2007 IEEE standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.

[2] Berlin Open Wireless Lab. `http://www.bowl.tu-berlin.de/`.

[3] eduroam. `http://www.eduroam.org/`.

[4] FreeRADIUS. `http://www.freeradius.org/`.

[5] OpenWrt. `http://openwrt.org/`.

[6] The UCI System. `http://wiki.openwrt.org/doc/uci`.

[7] M. Al-Bado, A. Feldmann, T. Fischer, T. Hühn, R. Merz, H. Schiöberg, J. Schulz-Zander, C. Sengul, and B. Vahl. Automated online reconfigurations in an outdoor live wireless mesh network. In *Proceedings of the ACM SIGCOMM Conference (demo session)*, August 2009.

[8] M. Al-Bado, R. Merz, C. Sengul, and A. Feldmann. A site-specific indoor link model for realistic wireless network simulations. In *4th International Conference on Simulation Tools and Techniques (SimuTools)*, 2011.

[9] R. Merz, H. Schiöberg, and C. Sengul. Design of a configurable wireless network testbed with live traffic. In *Proceedings of TridentCom 2010*, volume 46 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST)*, pages 189–198. Springer, May 2010.

[10] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS), 2000.

[11] N. Sarrar. Implementation and evaluation of an opportunistic mesh routing protocol. Master's thesis, Technische Universität Berlin, 2009.

[12] F. Sesser. A performance analysis of scalable source routing (ssr) in real-world wireless networks. Master's thesis, Technische Universität München, 2011.