Towards an Architecture for OpenFlow and Wireless Mesh Networks

Peter Dely
Computer Science Department
Karlstad University, Sweden
peter.dely@kau.se

1. INTRODUCTION

A Wireless Mesh Network (WMN) is a wireless multi-hop network, in which stationary mesh routers wirelessly relay traffic on behalf of other mesh routers or client stations and thereby form a wireless backbone. Different routing protocols, such as AODV [8], B.A.T.M.A.N. [6] or OLSR [2] have been proposed for the use in WMNs. Compared to fixed networks, WMNs exhibit a much higher variability in terms of link capacities and connectivity and are subject to external and network internal interference, which needs to be considered in the routing protocol design. For example, the link cost metric is an important factor for routing performance [3].

Albeit their optimized design, currently deployed WMN routing protocols lack desirable features such as support for fast roaming of users, interoperability with fixed network routing protocols and network virtualization. Furthermore there is no unified approach for extending such protocols. OpenFlow [7] is an API for programming networks and is a promising idea for addressing those problems in fixed networks. However due to the different nature of wireless networks, it is unclear how good OpenFlow performs in WMNs.

In this paper, we identify four challenges specific to the use of OpenFlow in WMNs, which are motivated by measurements from an OpenFlow deployment in the KAUMesh mesh testbed [4]. We discuss potential solutions and propose a system architecture that incorporates those solutions.

2. CHALLENGES

2.1 Hard/Software Limitations

OpenFlow has been designed for switches with a hardware data path, which is capable of processing OpenFlow rules at line-rate. Mesh routers are often based on low power embedded system hardware, which might be a bottleneck due to low memory and CPU speed. We have measured the throughput of the user space reference implementation data path [1] on the KAUMesh mesh routers, which are based on Linux, the Intel IXP-400 chip-set and an Intel 667 MHz X-Scale CPU.

Even with a low PHY rate of 18 Mbit/s, the data path is a bottleneck when too many rules are active. With 100 wild-carded rules, the throughput is about 15% lower when compared to standard Linux kernel IP-routing. The CPU utilization with OpenFlow is higher than with kernel-level IP-routing, but still below 20%. This suggests that the bottleneck is the memory bandwidth, which is too small to handle the frequent copying of data between the user to the kernel space required by the used data path implementation.

To alleviate this problem, an optimized, kernel-level version of the data path (such as provided by OpenVSwitch[9]) should be considered, which avoids copying data between user to kernel space. However, no matter if processing is done in user-space or the kernel, looking up rules in software is not as efficient as in a hardware accelerated TCAM provided by vendor switches. Hence, OpenFlow applications in WMNs should avoid large or too complex rule tables.

2.2 Signaling Channel

Exchanging protocol messages between OpenFlow routers and the controller requires IP-layer connectivity. When OpenFlow is used to set up the IP-routing in the mesh network, it is hence necessary to resort to different means for establishing a signaling channel between mesh routers and the controller. For example, two virtual networks, separated by SSIDs, can be used, to split control from data traffic. Routes in the control traffic network can be established using a standard routing protocol, such as OLSR. The data network is then set up using OpenFlow.

This might lead to excessive routing overhead. For small WMNs the overhead should however stay low: Measurements in a 5-node network, in which OLSR was used to set up the control network, showed OLSR produces a total of about 3 kbit/s load on the wireless network. OpenFlow in addition creates ca. 1 kbit/s load for periodic heart-beat messages and about 1-2 kbit load for each new flow.

2.3 Variability of the Wireless Channel

OpenFlow provides network management on a per flow basis. Typically flows are longer lived than the coherence time of the wireless channel. We measured Received Signal Strength Indicator (RSSI) on an indoor IEEE 802.11a link over one hour. Figure 1 shows the autocorrelation of RSSI samples. For a given channel measurement, the RSSI for the next second is fairly predictable. However, more long term predictions seem to be not feasible for indoor environments. Outdoor environments might be more stable though.

In addition to the short-term variations of the wireless channel, which are caused due to small changes in the RF-propagation environment a resulting different fading realizations, there are more long-term changes of the channel caused by external interference and large changes in the RF environment. We measured the UDP throughput for 60 seconds, every two minutes, for one hour. Figure 2 displays the autocorrelation function. For a given 60 second interval, the next interval two minutes later has a high correlation.

Addressing the short-term variability of the wireless channel requires very fine grained control, on a per-packet or even symbol level. This cannot be achieved with OpenFlow in an efficient way and therefore should be left to other mechanisms, such as PHY rate adaptation. In the context of OpenFlow, just considering long-term averages is more suitable. The variability of those long-term average can then be accounted for in OpenFlow routing decisions. The required channel quality monitoring data only needs to be transported to the OpenFlow controller fairly infrequently, route updates are infrequent and hence the overhead should be negligible.

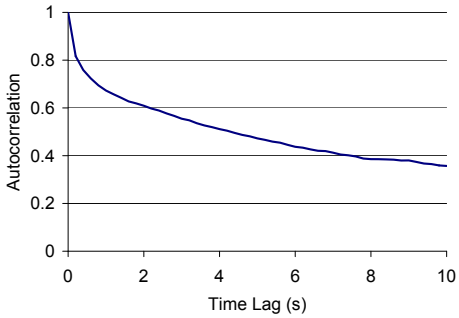

Figure 1: Autocorrelation of RSSI of a WLAN link

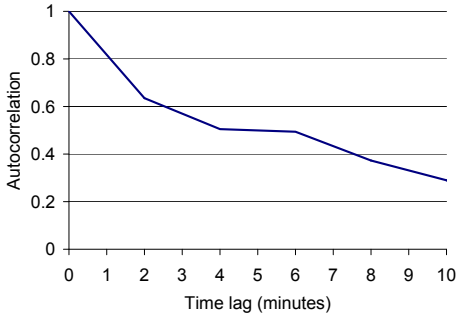

Figure 2: Autocorrelation of throughput

2.4 Capacity Management

One of the main challenges when operating a WMN is to provide good throughput to the users. Routing is a key component to achieve that goal. To enable an OpenFlow application that optimizes routing, the capacity of end-to-end connections needs to be estimated. This is relatively simple in fixed networks, where link capacities are known and no interference between links should occur.

In WMN link capacities change fast and intra-network and external interference is common. Furthermore, even given the knowledge of link capacities and interference, the DCF MAC protocol of the predominately used IEEE 802.11 standard makes such estimation difficult. There is a considerable amount of literature on capacity models for WMNs (e.g. [5]). To this point it is however unclear how such models should be used in the OpenFlow context. Extending the OpenFlow protocol to convey for example interference information is certainly possible, but not necessarily wise as it would replicate functionality of other protocols, such as IEEE 802.11k. Rather a generic interface between OpenFlow and such protocols should be created.

3. MANAGEMENT ARCHITECTURE

As future work we intend to design a management architecture for WMNs based on OpenFlow, that keeps the technology independence of OpenFlow intact, but addresses the unique properties of WMNs by adding an technology abstraction layer (see Figure 3).

The *Monitoring/Resource Estimation Interface* translates technology specific parameters such as the used modulation scheme into abstract information about available resources and provides this to OpenFlow applications. For example, an application can query the available bandwidth on a path and use this information to optimize routing.

Similar to FlowVisor [10], the *Technology Abstraction Interface* intercepts and modifies OpenFlow messages to meet the requirements of the underlying PHY technology. For example, IEEE 802.11 uses ACK frames, which necessitates using the correct sender MAC address for data frames. The

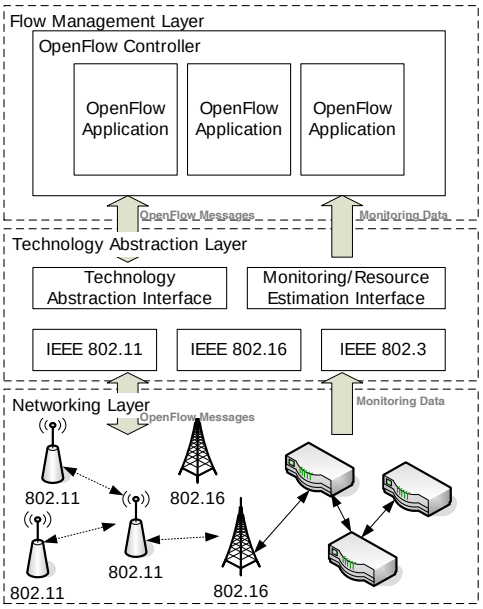

Figure 3: Proposed Management Architecture

Technology Abstraction Interface modifies the OpenFlow rule actions field accordingly.

4. CONCLUSIONS

The unique properties of WMNs require special attention when using OpenFlow in this context. Based on four important considerations such as capacity and network management, we propose an architecture the takes into account those properties and allows developers to write OpenFlow applications in an technology agnostic way.

5. REFERENCES

[1] Openflow git server. URL: http://gitosis.stanford.edu/openflow/.

[2] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[3] Douglas S. J.. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-Throughput path metric for multi-Hop wireless routing. *Proceedings of the 9th annual international conference on Mobile computing and networking - MobiCom '03*, page 134, 2003.

[4] Peter Dely. KAUMesh Website, 2011. URL: http://www.kau.se/kaumesh.

[5] K Duffy, DJ Leith, T Li, and D Malone. Modeling 802.11 mesh networks. *IEEE Communications Letters*, 10(8):635–637, 2006.

[6] D Johnson, N Ntlatlapa, and C Aichele. Simple pragmatic approach to mesh routing using batman. In *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries, CSIR, Pretoria, South Africa, 6-7 October 2008*, 2008.

[7] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38:69–74, March 2008.

[8] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[9] Ben Pfaff, Justin Pettit, Teemu Koponen, Keith Amidon, Martin Casado, and Scott Shenker. Extending networking into the virtualization layer. In *In 8th ACM Workshop on Hot Topics inNetworks (HotNets-VIII) (October 2009)*.

[10] Rob Sherwood, Glen Gibb, Kok kiong Yap, Martin Casado, Nick Mckeown, and Guru Parulkar. Can the production network be the testbed. In *In USENIX Symposium on Operating Systems Design and Implementation*, 2010.