A Design for OpenFlow Implementation of Fixed Backoff-time Switching Method in Wireless Mesh Networks

Sritrusta Sukaridhoto Nobuo Funabiki Toru Nakanishi Kan Watanabe

Graduate School of Natural Science and Technology, Okayama University

Abstract

As a flexible and cost-efficient scalable Internet access network, we have studied architectures, protocols, and design optimizations of the *Wireless Internet-access Mesh NETwork (WIMNET)* that is composed of wirelessly connected *access points (APs)*. Previously, we proposed the *Fixed Backoff-time Switching (FBS) method* in the CSMA/CA protocol for the AP in WIMNET to improve the real-time traffic performance by giving the necessary activation chance to each link, and presented its Linux implementation. In this paper, we present a design for the *OpenFlow* implementation of the FBS method based on a previous work by Dely et al.

1 Fixed Backoff-time Switching Method

1.1 Overview

To achieve scalability with low costs, the *Wireless Internet-access Mesh NETwork (WIMNET)* is composed of wirelessly connected *access points (APs)*, where a host can access to the Internet using multi-hop wireless links between APs through an Internet gateway. Because these links need to share the limited bandwidth for a wireless communication channel, its proper allocation to individual links is essential to afford real-time applications in WIMNET. To solve this problem, we have proposed the *Fixed Backoff-time Switching (FBS) method* to improve the real-time traffic performance by giving the necessary activation chance to each link [1].

The FBS method actually selects either of the shorter *active backoff-time* and the longer *passive backoff-time* for each link transmission by comparing the *target link activation rate* and the *actual link activation rate*, so that the link can handle the necessary traffic. Any backoff-time is assigned a different fixed value, so that no pair of the conflicting links may be activated simultaneously. Besides, the backoff-time for a link with larger traffic is assigned a smaller value than that for a link with smaller one, so that congested links can be activated preferentially. During communications, every time a node holding packets detects that the channel becomes free, it updates the target activation rate and the actual activation rate. If the actual one is smaller than the target one, it selects the active backoff-time to let the link be activated, because the current activation is not sufficient to handle its traffic. If it is larger, it selects the passive one to let other links be activated with higher priorities.

1.2 Target Link Activation Rate

For a wireless link l_{ij} transmitting packets from AP_i to AP_j for $i = 1, \cdots, N$ and $j = 1, \cdots, N$ in WIMNET with N APs, the target link activation rate rt_{ij} can be calculated by:

$$rt_{ij} = \frac{p_{ij}(t)}{t} \tag{1}$$

where $p_{ij}(t)$ represents the total number of bits that the applications in AP_i request transmitting to AP_j until time t (sec).

1.3 Actual Link Activation Rate

The *actual link activation rate* ra_{ij} for link l_{ij} is obtained by dividing the number of successfully transmitted frames with the number of possibly activating chances for the link:

$$ra_{ij} = \frac{sf_{ij}}{ac_{ij}} \tag{2}$$

where sf_{ij} represents the total number of successfully transmitted frames for link l_{ij}, and ac_{ij} does the number of possibly activating chances for link l_{ij}, which is incremented every time AP_i detects that the channel becomes free. In our Linux implementation [2], we used *tx_packets* variable for sf_{ij} and *attempts* variable for ac_{ij} from *minstrel* data rate structures. To use *minstrel* data structures, we recompiled the Linux kernel with *debugfs* and *minstrel_ht* options[3].

1.4 Active/Passive Backoff-time

The *active backoff-time* ta^m_{ij} and the *passive backoff-time* tp^m_{ij} for link l_{ij} are calculated by the following procedure.

1. Calculate the number of bits to be transmitted per second rb_{ij} for link l_{ij} by taking the summation

of the bit rates for all the communication requests using l_{ij}:

$$rb_{ij} = \sum_{k \in H_{ij}} hr_k \tag{3}$$

where H_{ij} represents the set of the host indices using link l_{ij} in the routing path, and hr_k does the requested bit rate (bps) of host k.

2. Sort every link in descending order of rb_{ij}, where the tiebreak is resolved by the number of hosts using this link for the routing path.
3. Set this sorted order to the link priority p_{ij} for l_{ij}.
4. Calculate the active/passive backoff-times for l_{ij}:

$$\begin{aligned} tamin_{ij}^m &= CW_{\min} \cdot \left(2^{m-1} + 2^{m-2} \cdot \tfrac{p_{ij}-1}{P}\right), \\ tamax_{ij}^m &= CW_{\min} \cdot \left(2^{m-1} + 2^{m-2} \cdot \tfrac{p_{ij}}{P}\right), \\ ta_{ij}^m &= rand\left[tamin_{ij}^m, tamax_{ij}^m\right], \end{aligned} \tag{4}$$

where $tamin_{ij}^m$ and $tamax_{ij}^m$ represent the minimum and maximum values for the active backoff-time for l_{ij} when the retry counter is m, $CW_{\min}$ does the initial CW size, and P does the largest priority among the links.

$$\begin{aligned} tpmin_{ij}^m &= CW_{\min} \cdot \left(2^{m-1} + 2^{m-2} \cdot \tfrac{P+p_{ij}-1}{P}\right), \\ tpmax_{ij}^m &= CW_{\min} \cdot \left(2^{m-1} + 2^{m-2} \cdot \tfrac{P+p_{ij}}{P}\right), \\ tp_{ij}^m &= rand\left[tpmin_{ij}^m, tpmax_{ij}^m\right]. \end{aligned} \tag{5}$$

where $tpmin_{ij}^m$ and $tpmax_{ij}^m$ represent the minimum and maximum values for the passive backoff-time for l_{ij}.

5. Assign the value of active/passive backoff-time through the modified *iw* userspace [4].

2 Design for OpenFlow Implementation

OpenFlow [5] is one implementation for *Software defined networking (SDN)* that is a form for the network virtualization where the control plane is separated from the data plane and is implemented as software applications. This architecture allows network administrators to control traffics without requiring physical access to hardware devices in networks. In this paper, we present our design for implementing the FBS method in OpenFlow, based on the idea by Dely. et al [6]. Figure 1 illustrates the architecture of the OpenFlow node for the FBS method in our design. The data process is described in the following procedure:

1. The node receives the number of bits transmitted from each network interface, and give this information into *FBSDaemon* that is a *daemon* application using perl.

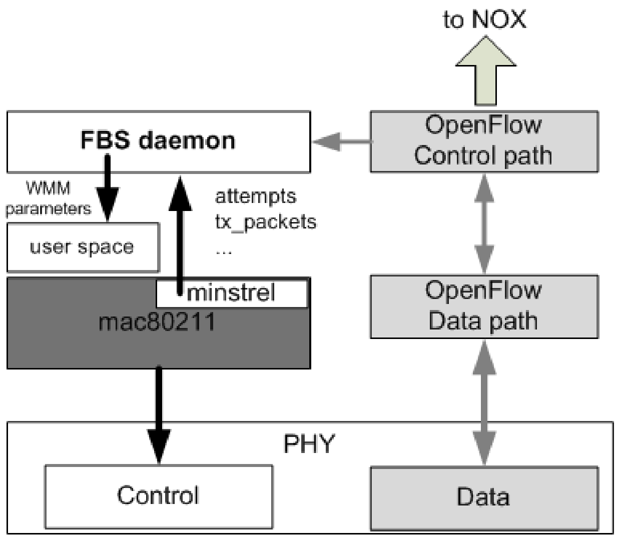

Fig. 1 Architecure of OpenFlow node for FBS method.

2. *FBSDaemon* receives the necessary information from *minstrel*, calculates both link activation rates, and compares them to select the value for CW_{min} from the active/passive backoff-time.
3. Disable the random backoff-time in the Linux kernel.
4. Assign the backoff-time through the *iw* userspace as WMM parameters.

In our future works, we will implement this design for the OpenFlow implementation of the FBS method, and evaluate the performance using the testbed network.

Acknowledgments

This work is partially supported by KAKENHI (22500059).

References

[1] S. Sukaridhoto, N. Funabiki, T. Nakanishi, K. Watanabe, and S. Tajima, "A fixed backoff-time switching method for CSMA/CA protocol in wireless mesh networks," to appear in IEICE Trans. Commun, 2013.

[2] S. Sukaridhoto, N. Funabiki, T. Nakanishi, K. Watanabe and S. Tajima, "A Linux implementation design of fixed backoff-time switching method for wireless mesh networks", IEICE Tech. Report, NS2012-67, pp.83-88, Sep. 2012.

[3] Linux Wireless, `http://linuxwireless.org`.

[4] iw, `http://wireless.kernel.org/en/users/Documentation/iw`.

[5] OpenFlow, `http://www.openflow.org`.

[6] P. Dely, A. Kassler, and N. Bayer, "OpenFlow for wireless mesh networks," Proc. ICCCN, 2011.