A Flow Aggregation Scheme for Seamless QoS Mobility Support in Wireless Mesh Networks

Dario Gallucci, Steven Mudda, Salvatore Vanini
Information Systems and Networking Institute
SUPSI
Manno, Switzerland
Email: [dario.gallucci,steven.mudda,salvatore.vanini]@supsi.ch

Radoslaw Szalski
Institute of Control and Information Engineering
Poznan University of Technology
Poznan, Poland
Email: radoslaw.szalski@put.poznan.pl

***Abstract*—Current solutions for network mobility support in wireless mesh networks lack Quality of Service (QoS) capabilities. Thus, they are not well suited for supporting services with QoS requirements (e.g., Voice over IP or Video on Demand). WiOptiMo is a solution, originally designed for seamless handoff management in the Internet, that was adapted for seamless internetworking in wireless mesh networks. In this paper, we show how its basic infrastructure was modified in order to meet the QoS expectations of mobile users running heterogeneous applications on a wireless mesh network. Specifically, QoS support is provided by aggregating application traffic flows with the same characteristics to limit overhead and by relaying compressed aggregated flows to the appropriate mobility provider. We experimentally evaluate the performance of our aggregation scheme and demonstrate that link utilization is optimized and QoS is improved.**

***Keywords*-Wireless Mesh Networks; Seamless Handover; QoS Mobility Support; Flow Aggregation; Flow Classification.**

I. Introduction

Recent years have witnessed a significant reduction in the costs of mobile computing platforms (e.g., laptops and smartphones), especially the hardware used in WiFi devices and has led to a widespread use of Wireless Mesh Networks (WMNs). WMNs provide multiple services to people using their mobile devices via a combination of fixed and mobile nodes, interconnected via wireless links to form a multi-hop ad-hoc network. WMNs are a cost-effective solution to extend the range of wired infrastructure networks with the help of easy to deploy wireless nodes. For example, the backbone of a telecom service provider can be easily expanded utilizing mechanisms to manage resources of wireless nodes [1] [2]. Existing mechanisms work only in scenarios where wireless connection stability can be ensured. For example, CARMNET [3] [4] utilizes the WMN paradigm to enable nearby wireless devices communicate with each other and proposes a distributed resource management method that can be easily integrated with a telecom IMS software infrastructure. This method (implicitly) assumes that the underlying network connectivity is not affected by topological changes (e.g., gateway changes) caused by the mobility of network's nodes. During those changes, packets for a given application flow might be rejected because of the change of the IP address, or they might be lost due to out-dated routing information. As a consequence, the quality and performance of correspondent applications can significantly decrease. Traditional mobility management schemes designed for IP-based networks are not suitable for WMN architectures. For example, Mobile IP [5] focuses on keeping the IP identity of a mobile node only. However, it introduces network overhead due to the protocol signaling and, consequently, causes a degradation of TCP throughput. On the other hand, since mobility support in pure ad-hoc networks focuses on rerouting (i.e., finding an alternative path in a timely manner, so that a flow can be handed off to the new path upon link disruption), these schemes perform poorly in WMNs. To overcome these limitations, several works have proposed different approaches to provide QoS and seamless mobility support in WMNs. However, many of them are not designed to manage multimedia services with QoS requirements—e.g., Voice over IP (VoIP) or Video on Demand (VoD). In this paper, we present an extension of our WiOptiMo [6] framework (described in section III) to provide generalized QoS mobility support in WMNs. In sections IV and V, we describe our enhanced framework and flow aggregation scheme to provide the required QoS to different types of applications in a WMN scenario. Finally, in section VI, we evaluate the performance improvement with respect to its standard configuration for WMNs.

II. Related Work

The existing work on mobility management in WMNs focuses on providing network-layer mobility support. RFC 4886 [7] specifically addresses the issue of network mobility. The different solutions presented in literature focus on managing the address of a mobile node due to the handoff process. In general, we can distinguish between intra-domain and inter-domain mobility. The first refers to handoffs inside the same network domain, the second to handoffs between different network domains. MobileNAT [8] addresses both intra- and inter-domain mobility. MobileNAT requires a modification at the network layer stack of a mobile node and changes to the standard DHCP protocol, which introduces network latency. SyncScan [9] is a Layer-2 procedure for intra-domain handoff in 802.11 infrastructure mode networks. It achieves good performance at the expense of a required global synchronization of beacon timings between clients and access points (AP). iMesh [10] provides low handoff latency for Layer-3 intra-domain handoffs between APs of a WMN. However, the hand-

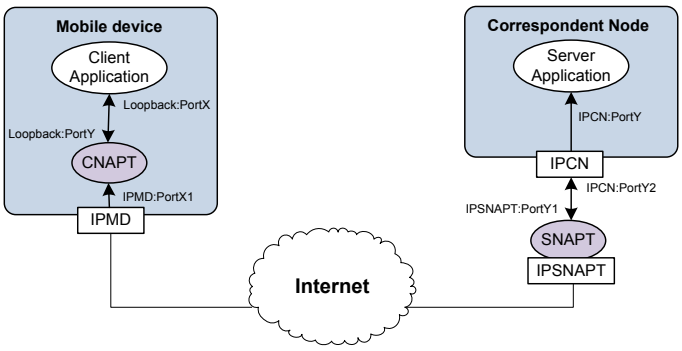

Figure 1: WiOptiMo's CNAPT and SNAPT IP decoupling.

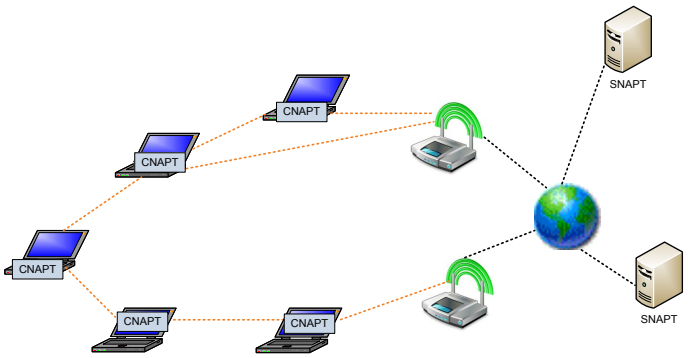

Figure 2: WiOptiMo configuration for a WMN.

off latency depends on the number of nodes between the new and old AP. BASH [11] focuses on the design of a intra-domain Layer-2 seamless handoff scheme for 802.11 WMNs, but the handoff protocol requires modifications at every mobile client. Authors of [12] use tunneling, as well as the standard Mobile IPv6 solution [13] and most of the existing network-layer mobility management schemes based on Mobile IP, such as Mobile Party [14] and AODV-PRD [15]. Tunneling introduces extra delay for the encapsulation/decapsulation of packets and has intrinsically low flexibility. Finally, SMesh [16] provides a 802.11 mesh network architecture for both intra-domain and inter-domain handoffs. For intra-domain handoffs, SMesh generates high network overhead, which grows linearly with the number of mobile clients. In case of inter-domain handoffs, network overhead generated by SMesh is proportional to the number of connections of a mobile client. The WiOptiMo framework provides mobility support by separately managing each application's flow, to meet the QoS expectations of all applications. In [6], we describe the architecture of WiOptiMo and present how it is adapted to handle a WMN context in [17]. In the next sections, we show how its architecture has been modified to handle efficiently multiple application's flow with different QoS requirements.

III. The WiOptiMo framework

WiOptiMo enables handoffs initiated by a mobile device. It manages the mobility of every device with the help of two software modules: Client Network Address & Port Translator (CNAPT) and Server Network Address & Port Translator (SNAPT). Together, these two components provide decoupling between the IP address assigned to a mobile device and the IP address used to access a service on the Internet. CNAPT and SNAPT hide any change of the IP address when a mobile host moves between different access networks, inside the same domain or between different domains. In Figure 1, we present a scenario where a mobile device with IP address IPMD has an active TCP session to a corresponding node with IP address IPCN. The TCP data packets are first relayed to the local CNAPT, which in turn relays them to the SNAPT. Upon receiving packets, the SNAPT (processes and) forwards them to the IPCN address. When the mobile device moves to a new network and gets a new IP address, the change in IP does not affect the application layer because the application packets are sent to the the local CNAPT, which relays them to the SNAPT with fixed IP address (IPSNAPT). This mechanism also allows a mobile node of a WMN to change gateway transparently (e.g., when node moves out of the reach of the initial gateway due to the mobility of the associated user), without suffering service disruption. To correctly manage the handoff process, CNAPT and SNAPT exchange handshaking packets with each other using a control socket.
In a generalized setting, mobile devices have CNAPT installed on them, while an Internet server or any node in a network (as in the scenario previously described) have SNAPT installed on them.

A. *WiOptiMo Architecture for a WMN*

In [18], we present a general configuration of our WiOptiMo for a WMN. We exploit the flexibility of location where a SNAPT can be installed to address scalability issues that might arise in a WMN. In this scenario, multiple SNAPTs can be deployed on mesh routers or on Internet nodes to avoid network congestion in a single spot. Every mobile wireless device has CNAPT installed on it to provide independent mobility support. We use a combination of network status monitoring and user configurable policy to enable every CNAPT to choose a suitable SNAPT that will relay its application flows. At start-up, each CNAPT connects to a fixed SNAPT specified in a configuration file. Then, it receives a list of other available SNAPTs from the currently connected SNAPT, and measures the delay towards them by means of passive and active monitoring of the control connection towards the SNAPTs, used for handshaking. CNAPTs also take into account the bandwidth used by applications in order to make a more wise SNAPT choice. The CNAPTs select a SNAPT to relay their data depending on the measured delay and estimated remaining throughput (based on the application's bandwidth requirements). This selection policy also helps in reducing the overload on any single SNAPT. Figure 2 shows WiOptiMo's architecture for a WMN. The SNAPTs can be managed by private administrators (otherwise called mobility service providers), who may require a fee for the use of their mobility service. This circumstance might foster the competition between mobility service providers, forcing them to increase the quality of provided service and benefit the entire WMN.

B. *Implementation changes*

We adapted WiOptiMo's implementation (both CNAPT and SNAPT) for low profile devices and to provide a fast handoff procedure. Figure 3 shows the changes to the basic

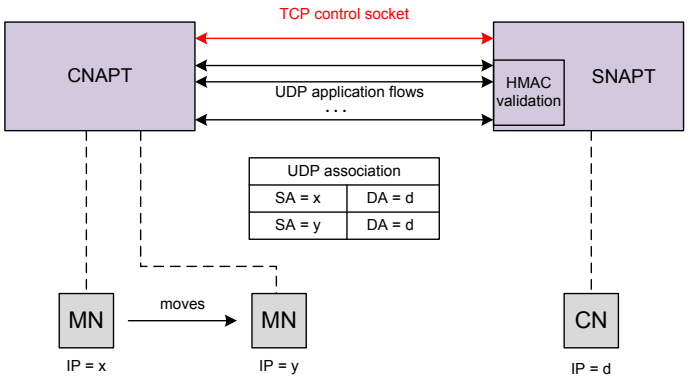

Figure 3: WiOptiMo adaptation for a WMN.

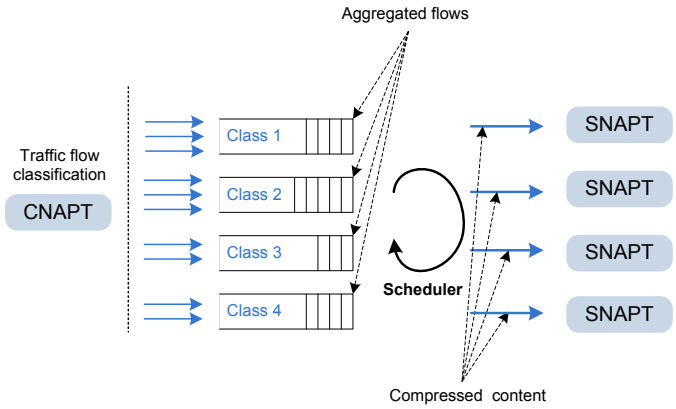

Figure 4: Software architecture of the aggregation scheme.

implementation of WiOptiMo. A TCP control socket still manages the communication between a CNAPT and a SNAPT. It provides network configuration parameters (e.g., the MTU of the underlying network) and also transmits data packets in a fall-back mode when middle-boxes, such as firewalls and/or NATs, block UDP packets. Further, the control socket is used to authenticate the CNAPT and to exchange a session key for providing data authenticity and integrity during a handoff. The CNAPT relays data packets to SNAPTs (and vice versa) using UDP sockets—this solution increases performance during handoffs, because UDP does not need to retransmit lost packets nor does it perform any connection setup. When a SNAPT receives a UDP data packet, it validates it using HMAC [19] and tests it against replay attacks using a sequence number. During handoffs (i.e., when the source IP address of data packets changes), the SNAPT updates the return IP address for the flow and transmits a keep-alive request to the CNAPT, which will reset the control connection or hasten the detection of a timeout. This event will then trigger the re-establishment of the control socket connection to the SNAPT.

IV. QoS support in WiOptiMo

We need an efficient delivery of heterogeneous traffic to meet the QoS requirements of applications. Since WiOptiMo relays each outgoing data flow from a client to a server application (through the link between CNAPT and SNAPT), every flow from a mobile device to its intended destination can be managed separately, according to its characteristics. In this section, we present the improvements to the WiOptiMo framework that enable it to efficiently deal with QoS, while still providing mobility support.

A. Flow classification

To meet the QoS requirements of applications, data flows are relayed to different SNAPTs based on their delay and throughput needs. In this regard, we identified four different flow classes according to the minimum throughput and maximum delay requirements of applications: *High Throughput and High Delay* (HT & HD), *High Throughput and Low Delay* (HT & LD), *Low Throughput and High Delay* (LT & HD), *Low Throughput and Low Delay* (LT & LD). In terms of throughput, the minimum threshold for classifying HT flow classes is 64kbit/s. We set the maximum delay for LD classes to 1s. As previously stated, during the normal workflow, a CNAPT periodically measures delay (one-trip time) and throughput (amount of received data over a time period) towards the different SNAPTs. Then, for each application flow, it detects the class type on the basis of process name, protocol and port number. Every class has an assigned delay and throughput requirements and data flows get relayed to a SNAPT that meets their delay and throughput requirements.

While our solution for flow classification is conceptually similar to DiffServ [20], it doesn't have its drawbacks. First, flow classification is performed dynamically per SNAPT, so that new flows are allocated depending on the current network performance statistics (e.g., the increase of the delay with the increase of the load). Second, our framework might refuse to serve a flow if its QoS requirements cannot be met, hence avoiding to disrupt the traffic already allocated. Moreover, the routing layer, as explained in [18], knows which traffic is managed by WiOptiMo. In this way, a QoS-aware routing mechanism can be executed whenever needed. In particular, network statistics about each single flow are reported to the routing layer so that there is no loss of granularity in the traffic management.

V. Flow Aggregation Mechanism

WiOptiMo allocates a UDP socket for each application flow (i.e., TCP connection). This behaviour does not favor the efficient handling of application flows with short frequent sessions (e.g., DNS requests), because useless computational overhead can be generated. It is also inefficient in terms of performance because the wireless link can be under-utilized. Furthermore, major unfairness may occur between competing flows—a major drawback when wireless links have high latency [21]. A naive solution would be to aggregate all data into a single flow, however applications with high bandwidth requirements would delay low latency applications. To overcome these issues, we designed a class based aggregation technique. Classified flows that belong to the same class are treated as a single aggregate and transmitted to a SNAPT using the same UDP socket. Our objective is to maximize the utilization of the available link bandwidth and reduce network overhead, thereby increasing the achieved throughput without significantly impacting the latency requirements.

Figure 4 presents the details of our aggregation mechanism.

TABLE I: Different Parameters of the Experiment.

Application Class	Packet Size (Bytes)	Range of bit-rate (bit/s)	Range of Flows
HT & HD	1360	1M - 20M	1 - 5
HT & LD	576	128k - 2M	1 - 5
LT & HD	1360	15k - 1M	1 - 5
LT & LD	100	15k - 128k	1 - 5

HT - High Throughput LT - Low Throughput
HD - High Delay LD - Low Delay

We implemented four connection queues, one for each of the application classes defined in Section IV-A. The queues feed into a scheduler, which uses a connection strategy based on flows' priority: the scheduler sends classes with more stringent requirements in terms first of delay and then of bandwidth. To reduce the amount of exchanged data, we enabled compression of the aggregated flows—packets are appended to the aggregated compound until their cumulative compressed size does not exceed the 70% of the underlying network's MTU. We chose this threshold to maximize the effectiveness of aggregation without having to resort to a slower algorithm.

VI. Experimental Results

In this section, we present the experiments conducted to assess the performance and QoS support of WiOptiMo with flow aggregation.

A. Performance of WiOptiMo with flow aggregation

We conducted experiments in three different scenarios:

1) Baseline: without WiOptiMo.
2) WiOptiMo basic.
3) WiOptiMo with flow aggregation mechanism.

Measurements showed that the performance of the baseline and WiOptiMo basic configurations are comparable (the degradation on throughput and the additional end-to-end delay introduced by the WiOptiMo solution are negligible, as presented also in [6]). For this reason, we report only the results for the baseline and WiOptiMo with flow aggregation scenarios.
In the next paragraphs, we show that our flow aggregation scheme achieves a better link utilization and reduces the amount of bytes exchanged in the network.

Experiment setup: We installed the WiOptiMo SNAPT on a Dell Optiplex 760 (server) and WiOptiMo CNAPT on a Dell Precision M4300 (client) with LinkSys Dual-Band Wireless A+G PCI Card. To avoid interference with nearby 802.11 access points operating on the 2.4 GHz band, we connected the client and server through a Netgear WNDR3800 wireless router (with OpenWRT 12.09 and only 802.11a networking enabled). Both client and server operated on a Linux distribution (Ubuntu 12.04 with Linux kernel 3.11).

We used the *Iperf* [22] network testing tool to send a stream of UDP packets (at a specific bit-rate) to server and measured the number of bytes sent between client and server using the *dumpcap* utility [23]. Instead of using the default UDP packets generated by Iperf—all packets contain same data—we configured the Iperf utility to generate UDP packets containing random text stored in a file. We performed experiments under the four different classes described in Section IV-A. For each flow class, we fixed the size of data in every UDP packet transmitted by the Iperf utility. We repeated experiments 10 times, to get more reliable results. Table I shows the characteristics of every flow generated by Iperf to measure the performance of WiOptiMo (for each application class).

We measured the performance of WiOptiMo by varying the number of flows and bit-rate of each flow, and observing their impact on the percentage of bytes saved on the link, due to flow aggregation and compression. It is calculated by subtracting pre-aggregation (and compression) bytes and post-aggregation (and compression) bytes, and dividing this difference by the pre-aggregation (and compression) bytes. This metric measures the bytes saved in the packet transfer between the client and server with the flow aggregation configuration, compared to the baseline configuration. It captures the energy spent to transfer data to the server. Since WiOptiMo performs flow aggregation and compression, this metric will enable us to measure the amount of energy that could be saved without impacting the QoS of applications.

Results: Figure 5 shows the percentage of bytes saved for applications with high throughput and high delay network requirements. We observe that for bit rates lower than 10Mbit/s, the percentage of bytes saved increases as the number of flows increases. Even for a single application flow, WiOptiMo with flow classification and aggregation helps in reducing, on average, the 60% of data sent between client and server. For bit-rates higher than 10Mbit/s, the percentage of bytes saved is still high but its relationship with the number of flows is no longer linear. This behaviour is due to the saturation of the system's modules capacity (wireless card, aggregation and compression mechanisms).

In Figure 6, we observe that when applications have high throughput and low delay requirements, savings by WiOptiMo increase from 38% for single flow to a maximum of 82.5% for applications with 5 flows. For all flows, the percentage of bytes saved increases until the bit-rate reaches about 400kbit/s. For much higher rates we observe that the percentage of bytes saved remains constant.

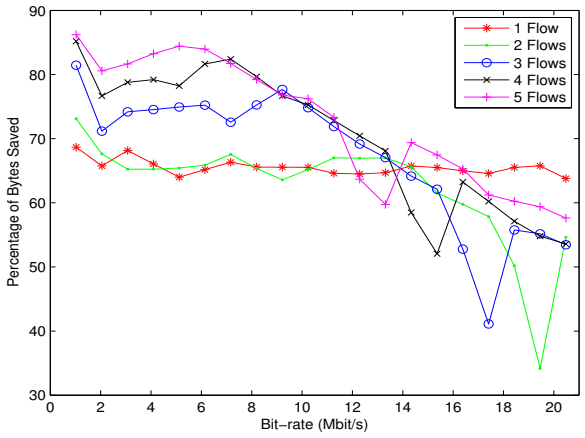

Figure 5: Percentage of bytes saved due to flow aggregation in HT & HD applications.

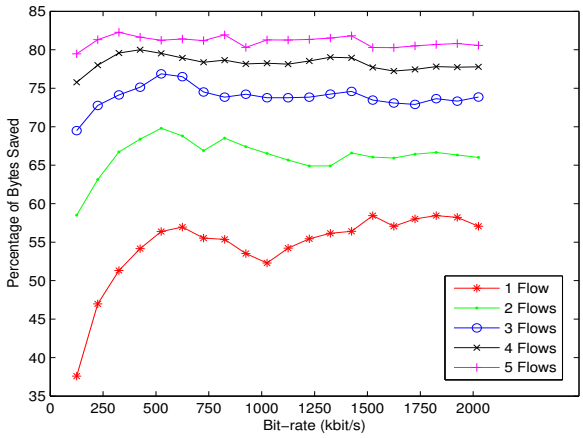

Figure 6: Percentage of bytes saved due to flow aggregation in HT & LD applications.

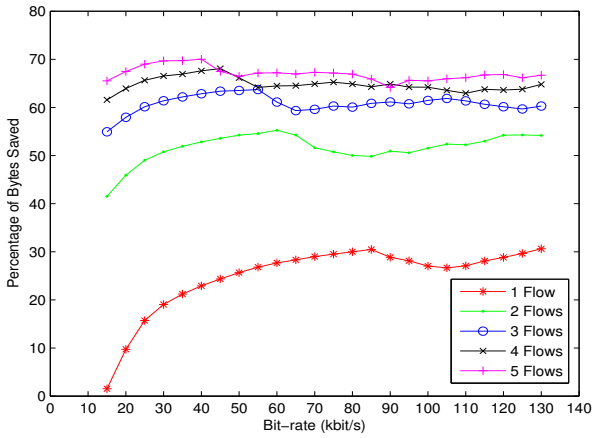

Figure 8: Percentage of bytes saved due to flow aggregation in LT & LD applications.

For low throughput and high delay tolerant applications (see Figure 7), we observe that for low bit-rates (∼125kbit/s), the percentage of bytes saved is not significant because no additional savings could be achieved by compressing and aggregating data packets arriving at long intervals of time. For higher bit rates (that is after the size of the aggregated packets allows better compression), savings increase and then stay constants (we can achieve a maximum savings of around 90%). In Figure 7, we also observe that savings achieved by WiOptiMo increase as the number of application flows increases.

Finally, for applications with low throughput and low delay requirements, we could achieve a maximum saving of 70% (see Figure8). Even at very low bit-rate (∼20kbit/s), WiOptiMo is able to save 10% of the data transferred between client and server.

B. *QoS support by WiOptiMo*

In the second set of experiments, we tested the capability of the WiOptiMo with an aggregation schema to provide QoS support. We used Iperf and measured the throughput between client and server using two different flow classes (HT & LD and HT & HD), in two distinct configurations: with a single SNAPT and with two SNAPTs. We show that a software configuration with multiple SNAPTs increases the network throughput and then helps preserving the QoS of applications.

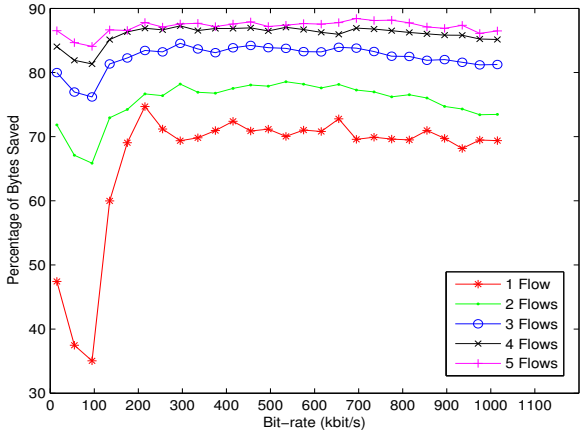

Figure 7: Percentage of bytes saved due to flow aggregation in LT & HD applications.

Experiment setup: We setup a wireless mesh network testbed to measure the QoS offered by WiOptiMo. The testbed consists of three static Internet-sharing nodes and two wireless mobile nodes. Each static node consists of an ALIX.2D2 system board, which supports two mini-PCI radios. We used one Wistron DNMA92 miniPCI card for each board, which is in turn connected to two 802.11n antennas. Each board mounts a 500 MHz AMD Geode LX800 processor and 256 MB DDR DRAM, runs Debian Wheezy 7.0 with Linux Kernel 3.12.6, and uses an ath9k driver for WiFi.
We used two ASUS EeePC 900 (with a Atheros 5008 Wireless Card, a 900MHz Celeron Processor and 1GB DDR RAM) as mobile nodes in our experiments. They operated on Debian Wheezy 7.0 with an ath5k WiFi driver.

To complete the hardware set-up, we installed WiOptiMo SNAPT on two Dell Optiplex 760 (servers) and a Lenovo ThinkPad T410a had WiOptiMo CNAPT installed on it. Both the machines operated on a Linux distribution (Ubuntu 12.04 with Linux kernel 3.11). Two static nodes (gateways) and

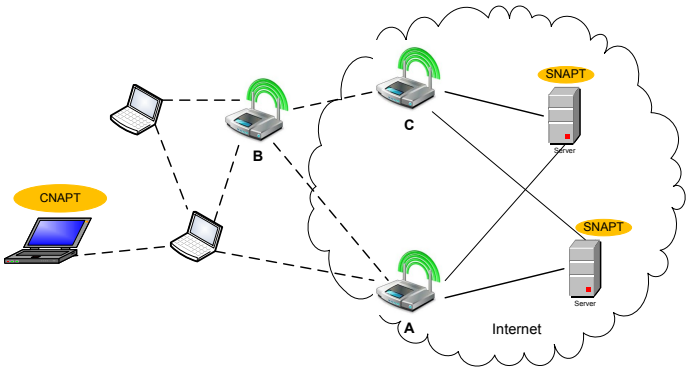

Figure 9: Testbed mesh network architecture.

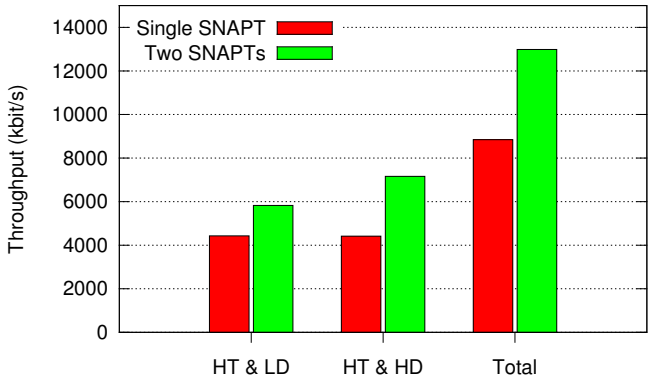

Figure 10: Throughput with multiple SNAPTs.

two servers were connected to the Internet with an Ethernet connection, while the rest of the nodes participate in the mesh network. We set the bandwidth of Ethernet connection to 10Mbit/s. The gateways performed NAT between the mesh network and the Internet. We ran the Optimised Link State Routing Protocol daemon (OLSRd, version 0.6.2) [24] on each node for network path resolution and configured the network to ensure that the two SNAPTs could be reached by separate gateways. The final testbed architecture is shown in Figure 9.

Results: Figure 10 shows the throughput comparison for two scenarios: with single SNAPT and with two SNAPTs (with different network delays) that could be reached from separate gateways. The results clearly show that in the first scenario the available bandwidth gets divided equally between the two application classes. In the second scenario, the HT & HD class achieves on average higher throughput compared to HT & LD class because the data of HT & LD class always gets routed to the SNAPT with lowest delay. Specifically, in the two SNAPT scenario, we observe a higher throughput compared to the bandwidth available towards each single gateway. Finally, we did not observe any significant additional delay in the network due to the introduction of WiOptiMo.

VII. Conclusion

In this paper, we have proposed a flow classification and aggregation scheme for enabling the WiOptiMo framework to manage multiple applications with different QoS requirements in a wireless mesh networking environment. We evaluated the proposed scheme on a Linux-based wireless mesh network testbed. Experimental results show that the aggregation mechanism we designed improves network performance in terms of link utilization and QoS, while still providing mobility support, without requiring any changes to be made to the network protocol stacks of either the mobile or fixed end systems. In the future, we would like to define and integrate into WiOptiMo, a requirements based policy that optimizes the use of mobility services and rewards users who do not waste network resources.

Acknowledgment

This work is supported by a grant from Switzerland through the Swiss Contribution to the enlarged European Union (PSPB-146/2010, CARMNET).

References

[1] S. Jakubczak, D. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Link-alike: using wireless to share network resources in a neighborhood," pp. 1–14, October 2008.

[2] C. Middleton and A. Potter, "Is it good to share? a case study of fon and meraki approaches to broadband provision," in Proceedings of International Telecommunications Society 17th Biennial Conference, 2008.

[3] M. Glabowski and A. Szwabe, "Carrier-grade internet access sharing in wireless mesh networks: the vision of the carmnet project," in Proceedings of The Ninth Advanced International Conference on Telecommunications, June 2013, (in press).

[4] P. Walkowiak, R. Szalski, S. Vanini, and A. Walt, "Integrating carmnet system with public wireless networks," ICN 2014, The Thirteenth International Conference on Networks, feb 2014, pp. 172–177.

[5] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in ipv6," RFC 3775, June 2004.

[6] G. A. D. C. et al., "Wioptimo: A cross-layering and autonomic approach to optimized internetwork roaming," in AHSWN Journal, May 2007, pp. 104–113.

[7] T. Ernst and L. H., "Network mobility support goals and requirements," in RFC 4886, July 2007.

[8] M. Buddhikot, A. Hari, K. Singh, and S. Miller, "Mobilenat: A new technique for mobility across heterogeneous address spaces," in ACM Mobile Networks Apps, vol. 10, no. 3, June 2005, pp. 289–302.

[9] I. Ramani and S. Savage, "Syncscan: Practical fast handoff 802.11 infrastructure networks," in 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2005), vol. 1, 2005, pp. 675–684.

[10] V. Navda, A. Kashyap, and S. R. Das, "Design and evaluation of imesh: an infrastructure-mode wireless mesh network," in IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WOWMOM), Italy, June 2005, pp. 164–170.

[11] Y. He and D. Perkins, "Bash: A backhaul-aided seamless handoff scheme forwireless mesh networks," in International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2008). IEEE, June 2008, pp. 1–8.

[12] R. Huang, C. Zhang, and Y. Fang, "A mobility management scheme for wireless mesh networks," in IEEE GLOBECOM 2007, Washington DC, USA, November 2007, pp. 5092–5096.

[13] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in ipv6," in RFC 3775, June 2004.

[14] M. Sabeur, G. A. Sukhar, B. Jouaber, D. Zeghlache, and H. Afifi, "Mobile party: A mobility management solution for wireless mesh network," in 3rd IEEE Int. Conf. Wireless and Mobile Comp., Networking, and Commun. (WiMob), October 2007.

[15] S. Speicher and C. H. Cap, "Fast layer 3 handoffs in aodv-based ieee 802.11 wireless mesh networks," in 3rd Int. Symp. Wireless Commun. Syst. (ISWCS), 2006, pp. 233–237.

[16] Y. Amir, C. Danilov, R. Musaloiu-Elefteri, and N. Rivera, "The smesh wireless mesh network," ACM Transactions on Computer Systems, vol. 28, no. 3, September 2010, pp. 6:1–6:49.

[17] D. Gallucci, S. Giordano, D. Puccinelli, N. Tejawsi, and S. Vanini, "Fixed mobile convergence: The quest for seamless mobility," in Fixed/Mobile Convergence Handbook. CRC Press, 2010, pp. 185–196.

[18] S. Vanini, D. Gallucci, S. Giordano, and A. Szwabe, "A delay-aware num-driven framework with terminal-based mobility support for heterogeneous wireless multi-hop networks," in ICTF 2013 Information and Communication Technology Forum, 2013, (in press).

[19] H. Krawczyk, M. Bellare, and R. Canetti, "Hmac: Keyed-hashing for message authentication," RFC 2014, February 1997.

[20] "Ietf diffserv working group page," http://datatracker.ietf.org/wg/diffserv/charter, 2014.

[21] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt, "Flow aggregation for enhanced tcp over wide-area wireless," in IEEE Conference on Computers and Communications, 2003, pp. 1754–1764.

[22] [Online]. Available: https://github.com/esnet/iperf (2014)

[23] [Online]. Available: http://www.wireshark.org/docs/man-pages/dumpcap.html [retrieved: May, 2014]

[24] "An ad-hoc wireless mesh routing daemon," http://www.olsr.org, 2014.