Implementation and End-to-end Throughput Evaluation of an IEEE 802.11 Compliant Version of the Enhanced-Backpressure Algorithm*

Kostas Choumas, Thanasis Korakis,
Iordanis Koutsopoulos, and Leandros Tassiulas

Department of Computer and Communication Engineering,
University of Thessaly, Greece
Centre for Research and Technology Hellas, CERTH, Greece
{kohoumas,korakis,jordan,leandros}@uth.gr

Abstract. Extensive work has been done in wireless multihop routing with several ideas based on shortest path or load balancing routing algorithms, that aim at minimizing end-to-end delay or maximizing throughput respectively. Backpressure is a throughput-optimal scheme for multihop routing and scheduling, while Enhanced-Backpressure is an incremental work that reduces end-to-end delay without sacrificing throughput optimality. However, the implementation of both theoretical schemes is not straightforward in the presence of 802.11 MAC, mainly because of their requirement for centralized scheduling decisions that is not aligned with the aspects of CSMA/CA.

This paper proposes a novel scheme, named Enhanced-Backpressure over WiFi (EBoW), which is compatible with the decentralized operation of WiFi networks and efficiently utilizes the benefits of Enhanced-Backpressure design, combining throughput optimality with low end-to-end delay. EBoW router is implemented relying on Click framework for routing configuration. The performance of EBoW is evaluated both on a medium-scale outdoors wireless testbed as well as through experimentations in NS-3 simulator tool. The protocol has been compared against other state of the art routing protocols and we argue that EBoW is much more throughput efficient than the others, while succeeding similar end-to-end delay.

Keywords: Backpressure, wireless mesh, multi-path routing, testbed.

1 Introduction

As the demand for seamless connection increases, the use of wireless multihop networks as a communication infrastructure becomes more and more popular. The efficiency of a multihop mesh network is directly related to the routing protocol. On the one hand, shortest path routing algorithms achieve minimum end-to-end delay. On the other hand, when efficiency is measured in terms of

* This work was supported by European Commission under Marie Curie IRSES grant PIRSES-GA-2010-269132 AGILENet.

throughput, the routing protocol should utilize multiple paths connecting a source-destination pair, avoiding the dominant shortest path approach.

A scheme that achieves throughput optimality is the well-known Backpressure (BP) algorithm, which operates on a time-slotted and centralized schedule and introduces scheduling and routing policies. Enhanced-Backpressure (EBP) is an improved mechanism that can be configured with an appropriate bias that inclines packets to move in the direction of their shortest paths. EBP provides lower delay than BP, without sacrificing throughput efficiency. The seminal work of Tassiulas and Ephremides [1] constitutes the core of BP and EBP, first introduced in the work of Neely *et al.* [2] as Dynamic Routing and Power Control (DPRC) and Enhanced-DPRC respectively. The work of Georgiadis *et al.* [3] offers a comprehensive survey.

Although both schemes are throughput optimal, they have not been implemented, mainly because of the centralized scheduling policy and the time-slotted assumption. These features do not fit with the dominant wireless communication protocols. In this paper we propose an 802.11 compliant version of EBP, named Enhanced-Backpressure over WiFi (EBoW), which implements the EBP aspects in a manner that is compatible with WiFi networks. The experimentation results of the proposed scheme show that EBP principles can be efficiently applied to contemporary WiFi mesh networks.

The novelty of the proposed algorithm is twofold: i) it provides a distributed load balancing scheme, which connects nodes through multiple paths and optimizes throughput efficiency keeping low end-to-end delay, while ii) it is applicable to WiFi ad-hoc meshes that support parallel flows, where a flow defines a stream of packets with specific source and destination. We prove that EBP principles could be well adapted so as to be efficient even if a central scheduling mechanism cannot exist. Particularly, we introduce a scheme in which every node attempts to forward packets to less loaded and closer to the destination neighbors, in a similar way that EBP scheme schedules. If there is no less loaded neighbor that is closer (or at least at the same distance) to the destination, then the node stops forwarding. This feature enables a simultaneously activated scheduling policy that offers more transmission opportunities to other neighboring nodes that experience collisions.

In order to evaluate the proposed routing and scheduling scheme in realistic conditions, we implement it using the Click modular router [4]. By conducting experiments in a realistic wireless testbed, named NITOS [5], we compare our algorithm to other well-known schemes, and we show that it sometimes gains significant throughput increase due to load-balancing inherent characteristics. We also explore and verify our experimentation results in identical setups using NS-3 [6] simulator integrated with Click development.

The rest of the paper is organized as follows. In Section 2 we introduce related work, while in Section 3 we describe BP and EBP schemes in detail. Section 4 describes our proposed scheme and Section 5 presents the implementation features. The numerical results are provided in Section 6, while Section 7 concludes the paper.

2 Related Work

The main part of BP and EBP [3] is the scheduling policy that requires a central node responsible for collecting information about the whole network. According to these schemes, each packet is related to a particular *commodity*, which may be defined by its destination or its source-destination pair or something more specific. For the rest of the paper, we assume that each commodity represents a particular destination, and there is a one-to-one correspondence between sets of commodities and destinations. Furthermore, each node maintains a set of internal network layer (layer 3 of OSI model) queues, while each queue corresponds to a commodity and stores all associated packets with this commodity. The length of a commodity queue is named as commodity *backlog*.

The scheduling policy of both schemes is actually a *maximum weight matching*[1] algorithm that chooses to transmit packets corresponding to particular commodities through specific links, so as to maximize the aggregate *link-commodity weight*. In contrast to traditional schemes where each link is assigned a single weight, BP and EBP assign multiple weights to a link, corresponding to different commodities. BP utilizes a link-commodity weight that is linear to the difference among the commodity backlogs of the adjacent nodes of the corresponding link, which is denoted as *differential backlog*. In case of EBP, the relative weight is also linear to the difference of the distances between the adjacent nodes of the link and the destination of the commodity, which similarly is denoted as *differential distance*.

Finally, the routing decision of each node is implied by the centralized scheduling policy, as each node transmits a packet associated with the commodity and through the link of the corresponding scheduling choice. Unlike traditional routing mechanisms for wired and wireless networks, BP and EBP routing does not perform any explicit path computation from source to destination.

Several algorithms based on BP aspects have been proposed, which mainly require heuristic modifications of BP principles and sometimes introduction of new MAC protocols. XPRESS [7] is a well designed cross-layer architecture, which implements accurately most of the BP designs. Actually, it forces the wireless network to act like a hypothetical wireless switch, operating on a TDMA MAC, as it is originally proposed in theory. On the other hand, ad-hoc wireless environments without centralized control are commonly used, and a TDMA MAC is not applicable in such environments.

Backpressure-based Rate Control Protocol (BRCP) [8] is an approach of BP scheduling over a predefined routing tree of sensors with a single commodity (or destination) related to the root of the tree. It is actually a distributed version, where each sensor decides the prioritization of its current transmission based on the current differential backlog. Obviously, this scheme does not apply on a wireless mesh with multiple commodities.

[1] Matching is a set of pairwise non-adjacent links and the maximum weight matching has the maximum aggregate weight.

DiffQ [9] is a different scheduling approach for wireless mesh networks that support multiple commodities. It enables packet transmissions of commodities through links with positive differential backlog and gives higher priority to those with larger differentials. This design requires changes of the 802.11 MAC layer. Furthermore, the routing decision depends on a shortest path routing protocol that does not feature load balancing characteristics and therefore does not take advantage of these BP inherent features.

The Backpressure Collection Protocol (BCP) [10] is a remarkable implementation and experimentation of BP design in wireless sensor networks over 802.15.4. In sensor networks there is only one commodity, since all packets have the same destination. So sensors include only one network layer queue, a feature that differentiates BCP from the BP scheme. Furthermore, BCP uses a BP related weight that is enriched with a penalty mechanism, while the use of an EBP oriented weight seems to be more efficient.

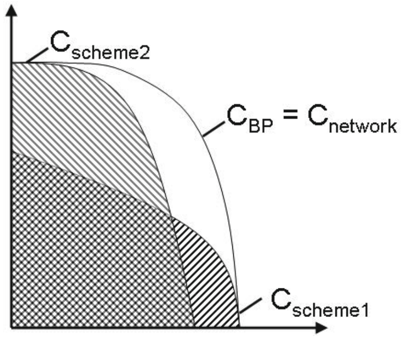
Horizon [11] is another system design for distance-vector routing in wireless multihop networks, which is inspired by the BP principles and it is compatible with 802.11 MAC and TCP. It is the first system architecture that implemented these principles in a wireless system design. However, Horizon uses an inexplicably simplified forwarding algorithm that actually depends on backlogs and not on differential backlogs, as BP does. Congestion Diversity Protocol (CDP) [12] is another distance-vector routing protocol that is queue length aware and not BP inspired. In like manner, it does not depend on differential backlogs and moreover does not keep different queues for different commodities. Both protocols, Horizon and CDP, do not include scheduling policies. The proposed EBoW scheme is compared with these implementations.

In contrast, SouRCe Routing (SRCR) is a shortest path routing protocol implemented in Roofnet [13]. It uses Expected Transmission Time (ETT) [14], which is the state of the art for defining wireless link weight for forwarding packets, and applies Dijkstra algorithm to explore the shortest route from a source to a destination. Since several evaluation works show that SRCR efficiently routes packets in mesh networks, the proposed algorithm will be compared to SRCR to measure the efficiency of the load balancing feature that comes as inherent characteristic in BP.

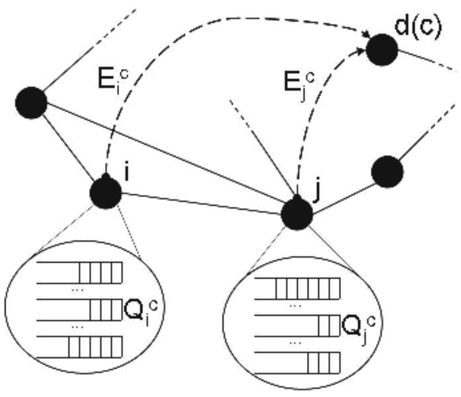
Summarizing, the most remarkable implemented routing protocols for WiFi ad-hoc meshes with multiple flows are these of Horizon, CDP and SRCR. Horizon and CDP are two notable distance-vector routing schemes that attempt to use multiple paths for packet forwarding using BP principles. Their common feature is their effort to avoid overloaded paths. On the other hand, SRCR uses always the shortest path ignoring alternative less loaded routes.

3 Enhanced-Backpressure Explained

Before starting the description we will introduce the main terminology of BP and EBP. A throughput vector includes the throughputs of all network flows, where a flow throughput is the average packet delivery rate of this flow. Furthermore,



(a) Capacity region of a network with two flows



(b) Commodity backlogs and shortest distances of two adjacent nodes

Fig. 1. Capacity region and Network snapshot

the capacity region of a certain scheme consists of the throughput vectors that this scheme is able to manage, while the capacity region of a network is the union of the capacity regions of all possible schemes. Figure 1(a) illustrates the above mentioned regions for a network with two flows. Tassiulas and Ephremides [1] proved that under a slotted environment, the capacity region of a network is the same as the capacity region of the BP scheme, which is superset of the capacity region of any other scheme. Neely *et al.* [2] also showed that EBP has the same capacity region with BP. The goal of the proposed scheme is to succeed an extended capacity region similar to that of the BP and EBP schemes.

Before proceeding, we introduce also some notations. We consider a multihop wireless network with node, link and commodity sets denoted as $\mathcal{N}, \mathcal{L}$ and $\mathcal{C}$ respectively. If $i, j \in \mathcal{N}$ are two adjacent nodes in the network, then $l = (i, j) \in \mathcal{L}$ is a directional link. Furthermore, if $c \in \mathcal{C}$ denotes a commodity, then $d(c) \in \mathcal{N}$ is the destination node that corresponds to commodity c. Finally, as it is shown in Figure 1(b), Q_i^c symbolizes backlog of commodity c at node i and E_i^c stands for the length of the shortest path (or the distance) from node i to destination $d(c)$. Consequently, differential backlog and differential distance of commodity c through link (i, j) are denoted as $\Delta Q_{i,j}^c = Q_i^c - Q_j^c$ and $\Delta E_{i,j}^c = E_i^c - E_j^c$ respectively. In addition, $R_{i,j}$ is the actual data rate of link (i, j).

The core of BP and EBP schemes is the maximum link-commodity weight matching algorithm. The BP link-commodity weight for a commodity $c \in \mathcal{C}$ and a link $(i, j) \in \mathcal{L}$ is either zero or the positive product of the actual data rate $R_{i,j}$ of the specific link and the differential backlog $\Delta Q_{i,j}^c$, as it is given in equation 1. Furthermore, EBP can be configured so as to incline packets to move in the direction of their shortest paths, using another appropriately defined link-commodity weight. More specifically, in EBP differential distance $\Delta E_{i,j}^c$ is added to the above mentioned differential backlog, as it is defined in equation 2. The distance should be estimated using hop count or another more sophisticated approach for measuring the shortest path length. Finally, the EBP scheme succeeds the same throughput optimality as the BP scheme, while at the same time it reduces packet delivery time.

$$w_{i,j}^c = \max\{\Delta Q_{i,j}^c \cdot R_{i,j}, 0\} \tag{1}$$
$$\bar{w}_{i,j}^c = \max\{(\Delta Q_{i,j}^c + \Delta E_{i,j}^c) \cdot R_{i,j}, 0\} \tag{2}$$

4 EBoW Design

4.1 Algorithm Description

The goal of the Enhanced-Backpressure over WiFi (EBoW) scheme is to improve throughput efficiency of wireless mesh networks, adopting the EBP principles, while simultaneously succeeding low delay, similar to the shortest path routing schemes. Due to its characteristics, this scheme can dynamically avoid overloaded paths, create parallel routes and therefore balance the traffic load in the network, increasing in this way its throughput efficiency.

Below we describe the main principle behind EBoW explaining the practices adapted, in order to achieve the benefits mentioned. Consider a network with multiple nodes that operate in a multihop environment. In such a setup, EBoW runs in every node that can act as source, destination or relay. Every node maintains a set of internal network layer queues, where each queue corresponds to a commodity, as specified in the BP and EBP schemes (further details in Section 2 and Section 3).

When a node receives or generates a packet that needs to be forwarded, it identifies the related commodity, recognizing the destination of the packet, and pushes the packet to the corresponding network layer queue. Moreover, each node that has packets in its queues, initiates a procedure to schedule the transmission of a packet. The most important part of this procedure is the calculation of the link-commodity weights of equation (3).

$$\hat{w}_{i,j}^c = (\Delta Q_{i,j}^c + \Delta E_{i,j}^c) \cdot R_{i,j} \tag{3}$$

Next, the node finds the link-commodity pair with the maximum weight, with positive differential backlog $\Delta Q_{i,j}^c > 0$ and non-negative differential distance $\Delta E_{i,j}^c \geq 0$. If there is at least one link-commodity pair that satisfies these conditions, the node selects the one with the maximum weight and transmits a packet from the corresponding queue. Then, the packet is passed down to the data-link layer (layer 2 of OSI model), while it is tagged to be transmitted through the link that is related with this pair. This algorithm is the routing policy of the EBoW scheme. On the other hand, in case there is no pair that meets these requirements, the node remains inactive and does not schedule transmissions. These requirements constitute the scheduling policy of the EBoW scheme.

The first scheduling condition requires a positive differential backlog $\Delta Q_{i,j}^c > 0$ and exists because of the remark of Li *et al.* [15], who explained why the capacity of a chain of nodes is reduced due to the inherent characteristics of 802.11. Nodes centrally located in the chain experience more collisions than border nodes. This is due to the fact that border nodes inject more packets into the chain than the subsequent nodes can forward, so these packets are eventually dropped. The time that border nodes spend to send those extra packets decreases the overall throughput, since it prevents transmissions of subsequent nodes. Our proposed algorithm takes this observation into account and therefore introduces the positive differential backlog condition to prevent it. So, in EBoW if a node observes that an adjacent node features a higher backlog than itself, it refrains from forwarding packets to it, thereby providing more transmission opportunities to this node. In this way, the capacity per hop is allocated more efficiently.

The second scheduling condition namely non-negative differential distance $\Delta E^c_{i,j} \geq 0$ exists in order to avoid excessively long routes of packets towards their destinations. Eventhough the EBP scheme takes into account distance to destination by incorporating it into the link weights, it often allows a packet to move further away from its destination. However, this is a typically undesired feature for traditional wireless applications, where users are interested in experiencing low end-to-end delay and jitter. This observation explains why the non-negative differential distance is a necessary condition for packet forwarding in our proposed scheme, which takes practical WiFi applications into consideration.

In essence , the main differentiation of EBoW with respect to the original EBP scheme is that it is structured around a distributed architecture, which makes it more suitable for real WiFi applications. EBP is a centralized algorithm which assigns non-negative weights to all link-commodity pairs and schedules packet transmissions based on the maximum weight matching principle. In contrast, in EBoW each node makes its forwarding decisions independently, according to the algorithm described before. More specifically, each node $i \in \mathcal{N}$ finds neighbor $j \in \mathcal{N} : (i, j) \in \mathcal{L}$ and commodity $c \in \mathcal{C}$ that maximize $\hat{w}^c_{i,j}$ (ties broken arbitrarily) subject to $\Delta Q^c_{i,j} > 0$ and $\Delta E^c_{i,j} \geq 0$, and pushes down from the

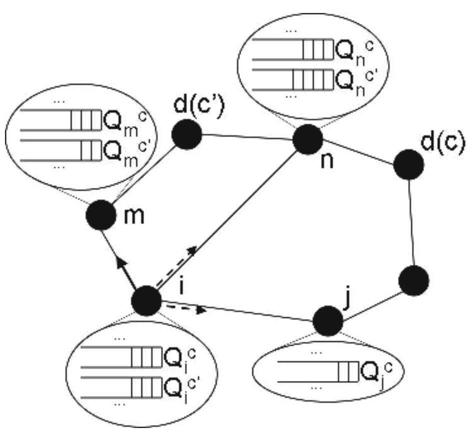


Fig. 2. Network example illustrating EBoW design

network to the data-link layer a packet that corresponds to commodity c for transmission through link (i,j). If there are no neighbor j and commodity c that satisfy these conditions, the node remains inactive.

For example, in Figure 2, considering that the path length is measured in hops and the data rate of each link is equal to unit, then $\hat{w}^c_{i,j} = ((3-2)+(2-2))\cdot 1 = 1$ and $\hat{w}^{c'}_{i,m} = ((3-2)+(2-1))\cdot 1 = 2$, while $\Delta Q^c_{i,n} = 3-3 = 0$, $\Delta E^c_{i,m} = 2-3 < 0$, $\Delta E^{c'}_{i,j} = 2-3 < 0$ and $\Delta Q^{c'}_{i,n} = 3-4 < 0$. As a result, node i forwards to neighbor m a packet that corresponds to commodity c'.

4.2 Distance Calculation and Broadcast Packets

The way that the algorithm estimates the distance between two nodes is another important point to mention. In the previous example, we used hop count for simplicity. In fact, the proposed algorithm uses a more sophisticated technique of distance measurement. The distance E^c_i between node i and destination $d(c)$ is expressed as a scale factor θ of the aggregate expected transmission time of a packet (queuing and processing delay is ignored), which is routed through the shortest path from source i to destination $d(c)$. The scale factor is adjusted so that $\Delta E^c_{i,j}$ and $\Delta Q^c_{i,j}$ to have the same range of values. The expected transmission time through a link is given by the well known formula of the Expected Transmission Time (ETT) metric [14]. According to this formula:

$$ETT = \frac{1}{d_f \cdot d_r} \cdot \frac{S}{B} \tag{4}$$

where d_f and d_r are the expected forward and reverse link delivery probabilities (the product of these two is the probability of a successful acknowledged transmission), S is the average packet size and B is the average packet rate that the rate controller assigns. So, if $e_{i,j}$ stands for the ETT weight of link (i,j), then

$$E^c_i = \theta \cdot \sum_{\forall (k,l)\in\mathcal{L} \text{ in the shortest path } i \rightsquigarrow d(c)} e_{k,l} \tag{5}$$

Every node needs to know all commodity backlogs of its neighbors, as well as the aggregate ETT of the shortest paths from itself and its neighbors to every destination. The ETT calculation for every link, especially of the forward and reverse probabilities d_f and d_r, is based on a probing mechanism that periodically forces nodes to send broadcast packets and inform neighbors about the number of the broadcast packets they have received. Due to this mechanism, each node estimates the ETT weights of its outgoing links, while simultaneously learning about the commodity backlogs of its neighbors, which are included in the broadcast packets.

However, nodes need to know ETT weights of links that are located multi-hops away from them. So, before a source starts forwarding a packet, it initiates a broadcast query flooding the network in order to reach the destination of the packet. While the query packet is passing through network links, it is tagged with the path that it followed until that point. The destination receives these queries and replies through all paths that the query flood explored. Similarly, each reply packet is tagged with the ETT weights of all links that this packet passed through. In this way, once the source (or a relay) receives the reply packets, it learns about all necessary ETT weights in order to estimate the distances from itself and its neighbors to the destination.

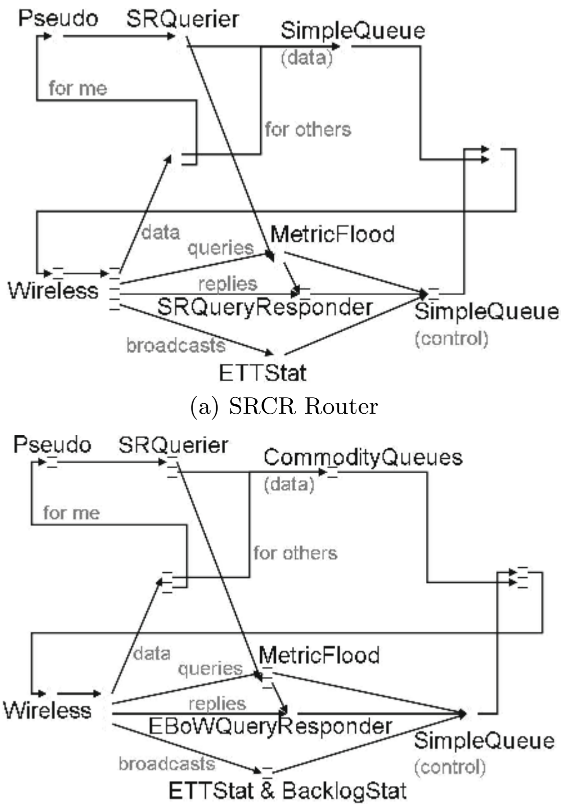
5 Implementation Details

The software development of EBoW scheme is based on the Click modular router framework [4], a novel software architecture for building flexible and configurable routers, that lie on network nodes and forward packets. A Click modular router consists of packet processing modules called elements. Individual elements implement simple router functions like packet classification, queuing and interfacing with network devices. Complete router configurations are built by connecting elements into a graph, where packets flow along the graph's edges. Most of these elements are given by the existing framework, while it is possible to construct additional and more specific elements.

The Click modular router is able to run as a linux-kernel module or a user-level executable on top of linux operated boxes. The Click framework includes a package of elements and a configuration of a user-level SRCR router (designed by the Roofnet [13] team). The configuration of the EBoW router shares a lot of common features with the corresponding one of the SRCR router, as it is illustrated in Figure 3. So, a brief description of the SRCR router follows, and then it is presented how the EBoW router is differentiated from this.

The User-Level SRCR Click Router (see Figure 3(a)) uses a *pseudo*-interface that connects the application layer with the underlying network layer and a *wireless* interface that receives and transmits packets over the air. Once a packet is received from the application layer (through the *pseudo*-interface), it is forwarded to the *SRQuerier* element. The main goal of this element is to estimate the shortest path from this node to the final destination of the packet. If *SRQuerier* does not have already the information for this packet (through past investigation for the shortest path of another packet with the same destination), it forwards the packet to the *MetricFlood* element. This element initiates a broadcast query flooding the network in order to reach the destination of the particular packet.

The query packets of the flood visit the *MetricFlood* elements of the relays until the destination. Once the destination is reached, the query packet is forwarded from *MetricFlood* to the *SRQueryResponder* element. *SRQueryResponder* initiates a reply packet that follows the shortest path to the initial node, visiting the corresponding element of each relay until the source. The shortest path is recognized by *SRQueryResponder* in the following way: the destination receives multiple query packets that were transmitted through the flooding method, through multiple paths. Each query packet is tagged with the path that it followed, as well as with the ETT weights of the intermediate links. So the destination explores the intermediate link weights and finds the shortest route applying Dijkstra algorithm.



(a) SRCR Router

(b) EBoW Router

Fig. 3. Click Configurations

Once the source receives the reply packet, it learns also about the shortest path to the destination, because the reply packet is similarly tagged with the path that it followed. So *SRQuerier* has the relative information for the following packets with the same destination. Therefore, the following packets are forwarded to the *SimpleQueue* element, which is able to store up to 50 packets, and implements the network layer queue. There is also another *SimpleQueue* element that stores the control packets, like query, reply and broadcast packets. The broadcast packets are generated and processed by the *ETTStat* element, for estimation of the ETT weights of the outgoing links, as it was described earlier in Subsection 4.2. Both queues store the packets, and forward them to the *wireless* interface on demand.

The User-Level EBoW Click Router (see Figure 3(b)) uses the same underlying probing mechanism (*ETTStat*) that estimates the ETT weights of the network links, while it features an additional probing mechanism (*BacklogStat*) for broadcasting commodity backlogs of each node to its neighbors. Furthermore, as it was mentioned in Subsection 4.2, either the source or the relay of a packet needs to know the distances from itself and its neighbors to the destination, so as to be able to estimate the differential distances. So, *SRQueryResponder* is replaced by the new *EBoWQueryResponder* element, which responds with a broadcast reply flood, instead of sending a single reply packet through the shortest path.

Finally, the most important difference between EBoW and SRCR configurations, is the structure that stores the data packets. The EBoW router uses the *CommodityQueues* element instead of the *SimpleQueue* one. This structure includes a variety of internal network layer queues, one for each known commodity. Once a packet that corresponds to a new commodity arrives, the structure produces a new internal network layer queue. When a packet is to be forwarded from *CommodityQueues* to the *wireless* interface, the *CommodityQueues* element estimates the EBoW weights, pulls a packet from the appropriate internal queue and tags it for forwarding through the appropriate link. In case that there is no link-commodity pair that meets the previous mentioned requirements (see Section 4), then the element does not forward any packet to the interface.

In summary, the distinctive features of EBoW router are three:

– The extension of probing mechanism that is used also to propagate commodity backlog information.
– The modification of *SRQueryResponder* element in order to respond more than once through every path that queries followed until the destination.
– The replacement of *SimpleQueue* element with an appropriate *CommodityQueues* one, that includes dynamically added and removed network layer queues and applies the EBoW scheme.

6 Experimentation and Results

In this section, we present evaluation experiments of the implemented scheme under various scenarios. For the purposes of the experimental evaluation, we used the realistic medium-scale NITOS testbed.. Experimentation with implemented mechanisms in realistic infrastructure, may lead to results that depend on varying traffic, interference conditions and topology settings that cannot be fully controlled. In order to arrive at solid results regarding the evaluation of our protocol, we decided to run multiple executions of each experimental scenario that is presented in this section. More specifically, each experiment is run 5 times and lasts 10 minutes. The reported results present average values. Moreover, we decided to run the same experiments under fully controlled settings in a simulation environment and for this purpose we used the NS-3 platform. Comparison between the results obtained through experimentation on the different platforms, enhances the validity of the followed evaluation approach.

Through our experiments, we compare EBoW with SRCR, Horizon and CDP, which are considered as state of the art routing protocols for WiFi networks. For comparison reasons, we also implemented the Horizon and CDP mechanisms based on the Click framework and the SRCR Click configuration. Performance comparison between EBoW, SRCR, Horizon and CDP, is presented in terms of throughput and end-to-end delay. Our experiments are organized in two parts, where in the first part the execution of an experimental scenario in a simple proof-of-principle network is presented, while in the second part we conduct a complex experiment in a topology of 20 nodes that introduces randomness and approximates realistic conditions. Details about the experimentation platform used in each case follow.

6.1 Experimentation Platforms

NITOS is a wireless outdoor testbed deployed across several floors and thus it provides for easy setup of multi-hop routes (Figure 4(a)). It is a non-RF-isolated wireless testbed, so we used 802.11a to eliminate interference, since commercial 802.11 products in Greece use only 802.11bg. The nodes used for the experiments feature a 1GHz VIA C3 processor and a Wistron CM9 wireless card. These wireless cards come along with a special version of the open source MAD-WiFi driver that enables the Click router to transmit and receive packets. The main features of the nodes and their software specifications are depicted in Table 1.

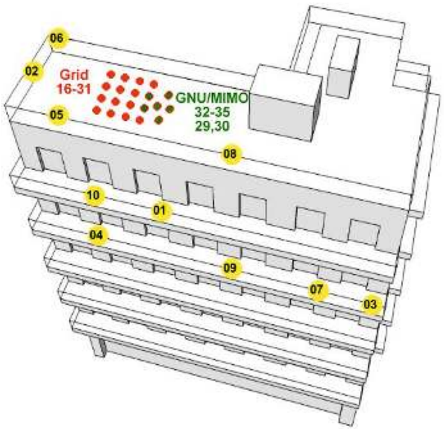
Table 1. Basic Configuration of NITLAB nodes

Model	Orbit radio nodes
CPU / Memory	1 GHz VIA C3 processor / 512 MB RAM
Operating system	Debian GNU/Linux 4.0r8 ”etch” / kernel ver 2.6.16
Wireless card	Wistron CM9 mini-pci / chipset Atheros AR5213A
Wireless driver	madwifi-old r846 2005-02-16 (modified)

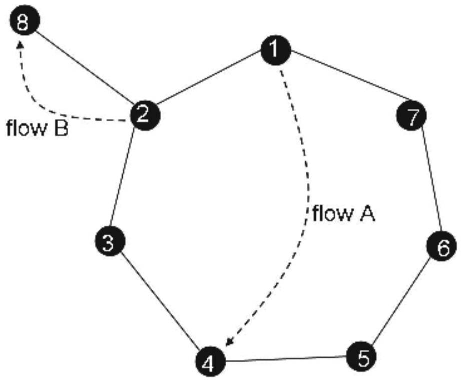
NS-3 is a valid simulation framework that is able to simulate network topologies under accurately controlled conditions. The experimenter has the flexibility to decide about the preferred OSI layer protocols. We exploited this feature to run our experiments using specific routing protocols. We decided to use the NS-3 simulator, because of its ability to be integrated with the Click framework, in order to be able to test the developed Click configurations in a simulation environment. As a result, we were able to use the same Click configuration in both real and simulated experiments.

6.2 Measurement Methodology

The traffic on the real testbed is generated using Iperf [16], a powerful tool for traffic generation and measurement. The experimental setup consists of several pairs of nodes that initiate UDP traffic flows by running Iperf clients at the sources and Iperf servers at the destinations. We also use the Iperf tool to collect throughput performance in each experiment. In order to monitor end-to-end delay performance per packet, we developed a custom timestamp mechanism



(a) NITOS Testbed



(b) Topology of Experiment I

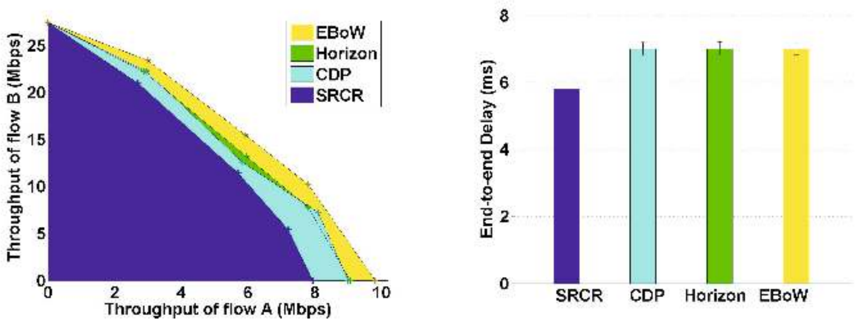
Fig. 4. Experimental Setup

using the Click framework. For experiments conducted in the NS-3 platform, we used the same mechanism to gather delay measurements, while traffic generation and throughput monitoring were performed using the *OnOffApplication* and *PacketSink* NS-3 application modules. Description of the conducted experiments and discussion about the obtained results follow.

6.3 Experiment I

The topology and connectivity map of our first experiment are illustrated in Figure 4(b). A ring topology of 7 nodes is designed, enabling a 3-hop and 4-hop path with the same source (node 1) and destination (node 4) nodes. Each network node is able to communicate through single-hop transmissions with each one of the two nodes that exist before and after it in the ring topology. In the presented topology, an extra node (node 8) also exists, which is able to communicate directly only with node 2.

The experiment consists of two parallel active flows, namely flow A and flow B, where through flow A node 1 transmits to node 4, and through flow B node 2 transmits to node 8. We conduct experiments of varying traffic rate for both of these flows, in order to estimate the capacity region supported by the network across the different protocols. Results obtained through experimentation in NITOS testbed and NS-3 simulator are very close, and for this reason we present in the following figures only the realistic testbed results. In Figure 5(a), we illustrate the capacity regions of the examined schemes. For example, the throughput vector $[6Mbps, 15Mbps]$ that indicates throughputs $6Mbps$ and $15Mbps$ for the flows A and B respectively, exists inside the capacity region of EBoW, while it is outside the capacity region of the SRCR scheme.



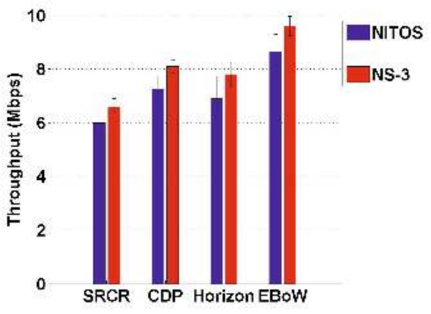
(a) Capacity Region (b) End-to-end Delay for flow A

Fig. 5. Results of Experiment I

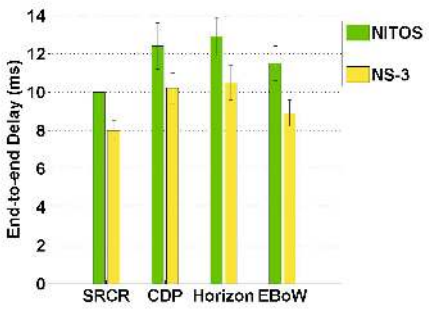
SRCR as a shortest path routing algorithm indicates the shortest route for each flow. Due to the symmetric nature of the designed topology, nearly equal ETT weights are reported for each link. As a result, SRCR obviously selects the shortest 3-hop path 1-2-3-4 that features the lowest aggregate ETT weight, in comparison with the 4-hop path 1-7-6-5-4. As SRCR does not feature any load balancing mechanism, it is not able to detect simultaneous ongoing transmissions, as these of flow B in this scenario. Under the SRCR approach, node 2 acts as a bottleneck that significantly reduces the throughput for both flows.

On the other hand, the rest of the schemes under consideration, better exploit the existence of two paths between the source and destination nodes of flow A. Although the 4-hop path is longer in terms of ETT weight, it features less traffic load in comparison with the 3-hop path due to the simultaneous transmissions of node 2. As a result, the 4-hop path is able to provide higher throughput performance. In addition, we notice that EBoW approach features an extended capacity region compared to the regions of the other two distance vector routing protocols. The main reason for this phenomenon comes from the inherent scheduling policy of the EBoW scheme, through which nodes that feature non positive differential backlogs are scheduled not to forward packets, while relay nodes that follow are provided with more transmission opportunities (see more details in Section 4).

Another important factor that has to be considered is end-to-end delay performance. We notice that end-to-end delay reported for flow B is equal for all schemes, as they all use the same 1-hop route for packet forwarding. The end-to-end delay performance for flow A yielded in each approach, is depicted in Figure 5(b). As clearly shown in this figure, end-to-end delay performance for flow A is reported quite similar for all load balancing schemes. Another observation is that the SRCR scheme outperforms all the other schemes in terms of end-to-end delay. The lower end-to-end delay measurements reported for SRCR are expected, as this approach is based on the shortest path principle. However, end-to-end delay performance of SRCR is not significantly lower, which is due to the collisions that occur frequently in the heavy loaded 3-hop path and negatively affect delay performance.



(a) Throughput



(b) End-to-end Delay

Fig. 6. Results of Experiment II

6.4 Experiment II

In the second experiment we extend our validation in a random setup that includes 20 nodes and 3 randomly selected 4-hop flows. Figure 6(a) shows the average throughput achieved for the 3 flows under each approach, on top of the two different platforms. As we can see, the proposed scheme features significant throughput improvement compared with the other schemes, across both platforms. Under EBoW, transmitter nodes are able to utilize multiple paths and moreover refrain from forwarding packets to nodes that feature high backlogs and thus result in network capacity improvement. In addition, EBoW performs better even in cases where the source-destination node pair is connected through a unique path, as it provides more transmission opportunities to nodes that feature higher backlogs.

Figure 6(b) demonstrates end-to-end delay measurements obtained during experimentation in the extended topology, across the different environments. According to these results, we notice that SRCR provides the lowest delay values among the compared approaches. An observation of notable importance is that the EBoW scheme outperforms the rest load balancing schemes. This result is obtained under experimentation in more generic topologies compared with the simple ring topology used in the previous experiment. More complex topologies provide larger paths in terms of aggregate ETT, which are avoided by the proposed scheme based on the non-negative differential distance forwarding requirement, but not by the rest load balancing schemes.

Results obtained between the different platforms are quite close but have some characteristics that provide for further discussion. First of all, we notice that real testbed experiments yield lower throughput and higher delay performance in comparison with simulation results. This comes from the fact simulation environments are not able to accurately estimate performance of realistic networks. However, experimentation in each platform aids in arriving at relative conclusions, regarding the superiority of our protocol. At this point we also remark the higher deviation values observed during testbed experimentation, which result because of the volatile nature of the realistic testbed environment.

7 Conclusions and Future Work

In this paper we propose an implemented EBP inspired scheme that exploits multi-path flow forwarding and outperforms the state of the art routing protocols in terms of throughput, while it keeps low packet delay close to the delay of the shortest path routing protocols. The new scheme features significant throughput benefits comparing to other routing protocols with load balancing efforts. We intent to extend the current work towards two directions: The first one is to compare the proposed protocol with centralized EBP based implementation approaches, and to see the advantages and disadvantages of the distributed VS the centralized version. The second one is to combine the proposed scheme with more sophisticated scheduling EBP inspired policies that will be implemented in a distributed manner and will allow the scheduling of transmissions in the neighborhood based on the load difference of the contenting hops. The scheduling will be based on prioritization schemes of 802.11 such as those proposed in 802.11e (different AIFS or different back-off values based on traffic queues).

References

1. Tassiulas, L., Ephremides, A.: Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Transactions on Automatic Control 37(12), 1936–1948 (1992)
2. Neely, M.J., Modiano, E., Rohrs, C.E.: Dynamic power allocation and routing for time varying wireless networks. In: IEEE INFOCOM, vol. 1, pp. 745–755 (April 2003)
3. Georgiadis, L., Neely, M., Tassiulas, L.: Resource allocation and cross layer control in wireless networks. Foundations and Trends in Networking 1(1), 1–149 (2006)
4. Morris, R., Kohler, E., Jannotti, J., Frans Kaashoek, M.: The click modular router. ACM SOSP 34(5), 217–231 (1999)
5. NITLab: Network Implementation Testbed Laboratory, `http://nitlab.inf.uth.gr/NITlab/index.php/testbed`
6. ns-3: Network Simulator, `http://www.nsnam.org`
7. Laufer, R., Salonidis, T., Lundgren, H., Guyadec, P.L.: Xpress: a cross-layer backpressure architecture for wireless multi-hop networks. In: ACM MobiCom, pp. 49–60 (2011)
8. Sridharan, A., Moeller, S., Krishnamachari, B., Hsieh, M.: Implementing backpressure-based rate control in wireless networks. In: IEEE ITA, pp. 341–345 (February 2009)
9. Warrier, A., Janakiraman, S., Ha, S., Rhee, I.: Diffq: Practical differential backlog congestion control for wireless networks. In: IEEE INFOCOM, pp. 262–270 (April 2009)
10. Moeller, S., Sridharan, A., Krishnamachari, B., Gnawali, O.: Routing without routes: The backpressure collection protocol. In: ACM/IEEE IPSN, pp. 279–290 (April 2010)
11. Radunović, B., Gkantsidis, C., Gunawardena, D., Key, P.: Horizon: Balancing tcp over multiple paths in wireless mesh network. In: ACM MobiCom, pp. 247–258 (2008)

12. Bhorkar, A.A., Javidi, T., Snoereny, A.C.: Achieving congestion diversity in wireless ad-hoc networks. In: IEEE INFOCOM, pp. 521–525 (April 2011)
13. Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and evaluation of an unplanned 802.11b mesh network. In: ACM MobiCom, pp. 31–42 (2005)
14. Draves, R., Padhye, J., Zill, B.: Routing in multi-radio, multi-hop wireless mesh networks. In: ACM MobiCom, pp. 114–128 (2004)
15. Li, J., Blake, C., De Couto, D.S.J., Lee, H.I., Morris, R.: Capacity of ad hoc wireless networks. In: ACM MobiCom, pp. 61–69 (2001)
16. Iperf: The TCP/UDP Bandwidth Measurement Tool, `http://dast.nlanr.net/Projects/Iperf/`