EAI Endorsed Transactions
on Mobile Communications and Applications

Research Article ICST.ORG

Delay Efficient Backpressure Routing in Wireless Ad Hoc Networks[★]

Leandros A. Maglaras[1], Dimitrios Katsaros[1,*]

[1]University of Thessaly, Department of Electrical & Computer Enginnering, 382 21 Volos, Greece

Abstract

Packet scheduling/routing in wireless ad hoc networks is a fundamental problem for ad hoc networking. Backpressure routing is a solid and throughput optimal policy for such networks, but suffers from increased delays. In this article, we present two holistic approaches to improve upon the delay problems of backpressure-type algorithms. We develop two scheduling algorithms, namely Voting backpressure and Layered backpressure routing, which are throughput optimal. We experimentally compare the proposed algorithms against state-of-the-art delay-aware backpressure algorithms, which provide optimal throughput, for different payloads and network topologies, both for static and mobile networks. The experimental evaluation of the proposed delay reduction algorithms attest their superiority in terms of QoS, robustness, low computational complexity and simplicity.

Received on 02 April 2014; accepted on 29 August 2014; published on 24 September 2014

Keywords: Packet scheduling/routing, Delay Reduction, Ad Hoc Networks, Mobile networks

doi:10.4108/mca.1.4.e5

1. Introduction

Wireless ad hoc (multi-hop) networks lack fixed infrastructure (e.g., base stations, mobile switching centers). The communication between any two nodes that are out of one another's transmission range is achieved through intermediate nodes. These intermediate nodes relay messages to set up a communication channel. Contemporary application areas of the ad hoc networks include modern battlefields, disaster relief, precision agriculture, e-health, ocean monitoring with underwater wireless sensor networks. Packet transmission scheduling in this type of networks is a fundamental issue since it is directly related to the achievement of a prescribed Quality of Service (QoS) and a minimum use of system resources. QoS is usually measured in terms of the average packet delay, transmission rate and maximum delay, and the main system resource to be saved is the nodes' energy so as to prolong network lifetime. In addition to delay and energy optimization, any packet scheduling/routing algorithm for ad hoc networks must be resilient to topology changes and strive for throughput optimality.

The development of a throughput-optimal routing algorithm for packet radio networks which is also robust to topology changes was first presented in [2]. It is based on the Lyapunov drift theory, and it is known as the *backpressure* ($\mathcal{BP}$) packet scheduling algorithm. Subsequently, the original concept spawns several lines of research on the topic. The performance of backpressure deteriorates in conditions of low, and even of moderate, load in the network, since the packets "circulate" in the network, i.e., the backpressure algorithm stabilizes the system using all possible paths throughout the network. The negative effect of this algorithm is to increase delay and also to increase the energy consumption of the nodes that play the role of relays. End to end delay and energy consumption are interconnected. The minimization of the average time that the packets stay in the system, implies a reduction in the average number of hops that the packets travel until they reach their destination, which in turn implies a reduction in the total energy consumption. Delay and energy consumption problems are particularly significant for ad hoc sensor networks. Routing-loop formation is another drawback of backpressure routing. In many real time applications like voice and video, high end-to-end packet delay is unacceptable. Often in such applications, a packet received with a high delay is not better than packet loss. High end-to-end delay can be prevented by not forwarding the packets on longer paths. On the other hand, in order to provide adequate load balancing in case of high traffic load, sufficient routes need to be maintained for any source-destination pair. Generally, these two objectives conflict with each other because few short routes exist in the network.

To circumvent the delay problem of backpressure, the *mean resource routing* algorithm [3] forces the links to

[★]This is an extended version of the paper [1]
[*]Corresponding author. Email: dkatsar@inf.uth.gr

stay inactive in order to lead the network to work in a burst mode, since for periods when the load of the network is low or moderate, link activation is prevented by a parameter M, leading to a delay increase. On the other hand, the relatively high computational cost incurred at each node by backpressure (maintenance of a queue for each possible destination, and update of these queues at each new arrival) inspired approaches based on *node grouping* in order to reduce the number of these queues [4] and thus the computational overhead, and as a side-product, reduce the delay. To alleviate the delay problems of backpressure, scheduling based on the *combination of backpressure and shortest-paths* have been proposed [5], which though demands an excessive number of queues and repeated calculation of all-node-pairs distances in case of topology changes. In summary, all these approaches either impose unpractical and non scalable assumptions, or are not very efficient. Finally, some recent ideas [6, 7] could be incorporated in an *orthogonal way* to improve analogously the delay performance of all policies, but this is beyond the scope of this paper. Here, we take holistic approaches in designing efficient delay-aware backpressure algorithms, which are both practical – have low computational overhead and are robust to topology changes.

1.1. Motivation

In the mean resource routing ($\mathcal{MR}-\mathcal{BP}$) algorithms [3, 8], backpressure is used with a parameter M that forces the links to stay inactive as long as their differential backlog does not exceed the value of M, i.e., links with backpressure less than M cannot be scheduled. This means that packets stay in queues longer, which can lead to higher delays. Indeed, we tested this intuitive result in [1] by evaluating the delay performance of these algorithms and confirmed this behavior. Therefore, since the $\mathcal{MR}-\mathcal{BP}$ algorithms are not delay competitive, we do not consider them as viable competitors any more.

The authors of [4] identified the scalability problems of the backpressure algorithm in *wireline networks* with millions of routers (e.g., Internet) due to the maintenance of several queues per node (one queue per destination, as it is mandated by the original $\mathcal{BP}$), and proposed the creation of clusters[1] of nodes so as to reduce the number of queues per node, which as a "side-product" has the additional benefit of delay reduction. Their algorithm, namely *cluster-based backpressure* ($\mathcal{CB}-\mathcal{BP}$), requires maintaining one queue per *gateway* at each relay node, leading to an excessive number of gateways, which in turn alleviates any performance gains (i.e., increases delays) when the number of clusters becomes, say, more than ten. Even worse, all contemporary clustering algorithms [9, 10] (such as the Distributed Clustering algorithm and the Max-min d-hop clustering algorithm) for wireless ad hoc networks produce quite a large number of clusters and thus several dozens of gateways even for relatively small networks. In these cases, the delays of $\mathcal{CB}-\mathcal{BP}$ algorithms asymptotically reach the delays of the original $\mathcal{BP}$ (cf. Figure 6c). Moreover, the strong dependence of the policy on the identity of the gateways makes it problematic in cases of gateways breakdowns. Finally, considering the technicalities of the $\mathcal{CB}-\mathcal{BP}$ algorithm, it is evident that even when a packet has already reached the destination cluster (i.e., the cluster where the destination nodes resides), the algorithm is agnostic on this information and it may happen to send it again out of the cluster seeking alternative paths to the destination. This behavior is detected and improved in our experiments (cf. subsection 6.1).

The Shortest-paths backpressure ($\mathcal{SP}-\mathcal{BP}$) algorithm [5] assumes the pre - computation of all pairwise-node distances and then application of the backpressure notion on the shortest path(s) between source and destination. Apparently, this algorithm achieves the lower bound of the delay. It does so at the expense of a very complex initialization phase (all-node-pairs distances must be computed). Moreover, it must maintain a quadratic number of queues at each node $[n \times (n-1)]$, whereas the original backpressure maintains only a linear number of queues $(n-1)$ at each node. Also, during the running phase of the algorithm, the processing of such a huge number of queues is time-consuming, which in turn increases delays. Apart from these computational-type problems, frequent topology changes leads the algorithm to break down, since many shortest paths do not exist anymore. Finally, in the topologies where there is only one shortest-path per node pair (which is the most common case), $\mathcal{SP}-\mathcal{BP}$ rapidly consumes the energy of that path, shortening the longevity of the network. This problem becomes worse in the topologies where a lot of shortest paths traverse the same set of nodes (i.e., the nodes with high betweenness centrality [11]). In such topologies, these nodes deplete their energy so fast that the network gets partitioned very rapidly.

1.2. Contributions

The present article develops two scheduling algorithms, namely the *Voting-based Backpressure ($\mathcal{V}o\mathcal{BP}$)* and the *Layered Backpressure ($\mathcal{L}ay\mathcal{BP}$)*.

For the former algorithm, taking a purely localized approach, we require the packets to carry their immediate travel history, so that the relays are prevented from sending the packets back from where they came. This is the opposite behavior of the *ACO* method [12], but they are similar in the sense that they both use hints – pheromone for *ACO* and history for $\mathcal{V}o\mathcal{BP}$. For the $\mathcal{L}ay\mathcal{BP}$ algorithm, taking a less localized approach, we

[1]The authors did not propose any specific clustering algorithm.

create "layers" of nodes and use the identities of these layers to forward the packets toward the destination's layer, and "discouraging" the packets from leaving the destination layer; these layers act as attractors for the packets.

The two proposed algorithms aim to decrease the mean end to end delay of the packets while mainting throughput performance optimal. $\mathcal{L}ay\mathcal{BP}$, which is a global-network-topology-aware algorithmg, splits the network in small subnetworks and guides the packets towards their destination subnetwork, without forcing them to follow specific routes. This procedure can significantly improve the performance of a routing algorithm in terms of mean delay, and it can be used in a any topology regardless of shape and size. On the other hand information about the traffic in a local aspect is exploited by $\mathcal{V}o\mathcal{BP}$. The proposed algorithm dynamically adapts the routing of packets locally, according to the current situation of the network. $\mathcal{L}ay\mathcal{BP}$ is a static method that is based on the split of the network in smaller parts while $\mathcal{V}o\mathcal{BP}$ is a dynamic algorithm that routes packets according to their travel history. The two proposed algorithms exploit different characteristics of the network (topology vs traffic) in order to decrease the end to end delay of packets. Moreover $\mathcal{L}ay\mathcal{BP}$ is global-network-topology-aware algorithm while $\mathcal{V}o\mathcal{BP}$ is a local-network-topology-aware one. The two proposed algorithms may be combined in order to further improve the performance of the system.

The article makes the following contributions:

- It experimentally evaluates well established delay-aware backpressure policies, namely $\mathcal{BP}$, $\mathcal{MR}-\mathcal{BP}$, $\mathcal{CB}-\mathcal{BP}$, $\mathcal{SP}-\mathcal{BP}$ for several network topologies.

- It develops the *Voting-based Backpressure* policy, which "wipes-outs" the ping-pong effect in backpressure-based packet scheduling. The algorithm exploits current traffic in the network in order to lead packets away from already visited nodes, vanishing rooting-loop.

- It develops the *Layered BackPressure ($\mathcal{L}ay\mathcal{BP}$)*, which divides the network into "layers" according to the connectivity of nodes. This algorithm maintains the same number of queues with the original $\mathcal{BP}$, and one order of magnitude less number of queues compared to $\mathcal{SP}-\mathcal{BP}$. It does not require the existence of *gateways* and aggregated queues as does $\mathcal{CB}-\mathcal{BP}$. In complex networks gateways may be so many that $\mathcal{CB}-\mathcal{BP}$ performs like original $\mathcal{BP}$. In addition, it is not a deterministic algorithm like $\mathcal{SP}-\mathcal{BP}$ where packets are forced to travel the shortest-path among nodes. Instead the packets are "suggested" to follow the shortest path from the source to the destination layer.

- It develops the Enhanced Layered Backpressure policy $E\mathcal{L}-\mathcal{BP}$ that improves the behavior of $\mathcal{L}ay\mathcal{BP}$ in case of moving nodes. The new algorithm have the same characteristics with the $\mathcal{L}ay\mathcal{BP}$ and is robust to topology changes in terms of nodes that move from one layer to another.

- It develops a new random-walk based node clustering algorithm, that exploits the connectivity among nodes.

- It evaluates experimentally the performance of the proposed packet scheduling policies, with all the previous delay-aware throughput optimal backpressure algorithms.

The rest of this article is organized as follows: Section 2 describes the network model; Section 3 introduces the original backpressure scheme; Section 4 presents the $\mathcal{V}o\mathcal{BP}$ scheme; Section 5 describes the $\mathcal{L}ay\mathcal{BP}$ algorithm; Section 5.4 describes the $E\mathcal{L}-\mathcal{BP}$ algorithm; Section 6 presents the simulation environment and results; Section 7 reviews the most important works relevant to this article, and finally Section 8 concludes the article.

Table 1 summarizes the basic notations used in the article.

Symbol	*Definition*
G	*Network*
V	*Set of nodes*
L	*Set of edges*
E_i	*Arrival process for node i*
Q_n^d	*Queue of packets destined to node d*
$Layer(n)$	*Layer number of node n*
G_i	*Neighborhood of node i*
C_{nmd}	*Counter for node m*

Table 1. Notation table

2. Network model

We consider a network $G = (V, L)$, where V is the set of nodes (vertices) and L is the set of links (edges). We consider the following generic properties:

- Nodes are static or mobile, the communication links are bidirectional, and nodes communicate in a multi-hop fashion.

- Topology changes may take place, due to nodes getting down or link failure.

- Network nodes are homogeneous and do not have GPS-like hardware. Links have equal to one capacity.
- Concurrent transmissions cause mutual interference since the transmission medium is shared. Matchings are the set of links that can be scheduled simultaneously. A max-weight scheduling policy is used.
- A node cannot transmit and receive at the same time.
- Time is slotted with a time slot t.
- For each node i, there is an arrival process E_i such that $E_i(t)$ is the number of exogenous arrivals up to time t. We assume that packets arrive exponentially with mean arrival rate λ. The source and destination of each packet is randomly selected among all the nodes. Special cases are also investigated by using the Zipf's distribution, where packets 'prefer' some nodes as destinations representing a more realistic scenario.

3. The original Backpressure algorithm

Backpressure [2] is a joint scheduling and routing policy which favors traffic with high backlog differentials. The backpressure algorithm performs the following actions for routing and scheduling decisions in every time slot t.

- *Resource allocation*
  For each link (n, m) assign a temporary weight according to the differential backlog of every commodity (destination) in the network:
  $$wt_{nmd}(t) = max(Q_n^d - Q_m^d, 0).$$
  Then, define the maximum difference of queue backlogs according to:
  $$w_{nm}(t) = \max_{d \epsilon D} wt_{nm}(t).$$
  Let $d_{mn}^*[t]$ be the commodity with maximum backpressure for link (n,m) in time slot t.
- *Scheduling*
  The network controller chooses the control action that solves the following optimization problem:
  $$\mu^*(t) = \underset{\mu \epsilon \Gamma}{argmax} \sum_{(n,m) \epsilon L} \mu_{nm} w_{nm}(t),$$
  , where Γ denotes the set of all schedules subject to the one hop interference model.

  In our model, where the capacity of every link μ_{nm} equals to one, the chosen schedule maximizes the sum of weights. Ties are broken arbitrarily.
- *Routing*
  In time slot t, each link (n, m) that belongs to the selected scheduling policy forwards one packet of the commodity $d_{mn}^*[t]$ from node n to node m. The routes are determined on the basis of differential queue backlog providing adaptivity of the method to congestion.

The backpressure algorithm is throughput-optimal and discourages transmitting to congested nodes, utilizing all possible paths between source and destination. This property leads to unnecessary end-to-end delay when the traffic load is light. Moreover, using longer paths in cases of light or moderate traffic wastes network resources (node energy).

4. Voting-based backpressure

The driving idea behind the $\mathcal{VoBP}$ scheduling algorithm is the reduction of the path length traveled by a packet by reducing any cycles observed on this path. Apparently, this can be done by having the packet "carrying" its trajectory. Such an approach though, would comprise the properties of the backpressure algorithm (i.e., utilize all paths), and also it would be impractical, since these trajectories grow remarkably large for long paths, having as a consequence a considerable increase in the packet size and increased processing time by the routers. Therefore the storage of such "rich" information in the packets is not a viable option, unless we confine the amount of information. This is exactly the direction followed by the $\mathcal{VoBP}$ algorithm.

$\mathcal{VoBP}$ uses minimum information stored in the packets in order to select the scheduling policy in each time slot. Every packet holds the last node it has previously visited. This information is used in the scheduling phase of the algorithm in order to prevent packets from revisiting the same nodes and traveling in circles in the network. Each node n maintains a separate queue of packets for each destination $Q_n^d[t]$ (as the original backpressure algorithm). Each queue d has a counter for each neighbor node C_{nmd}. This counter is updated every time a packet arrives to or leaves from node n having as destination node d. Since the packet holds the information about the last visited node a certain procedure is followed at each transition.

When packet i having as final destination node d arrives at node n from node m the following actions take place.

- At node n C_{nmd} is incremented by one.
- At node m C_{nmd} is decremented by one.
- Packet i updates last visited node from m to n.

In order to find the worst neighbor all the packets that belong to each queue participate in a voting procedure updating the appropriate counters. A vote is given according to Definition 1.

Definition 1 (Vote). A packet (destined to node d) that arrives to node n assigns a vote to node m it has previously visited. This vote is represented by incrementing counter C_{nmd} by one.

The votes are positive counters but have the meaning of negativity. The neighboring node with the most votes is the worst candidate for routing packets towards it. Based on the information of the packets, the scheduling algorithm assigns an appropriate weight to the corresponding links, preventing this way the controller from choosing such a routing policy that moves the packets backwards in the network. Parameter A is used in order to assign weights to links and help the routing mechanism forward packets to delay efficient paths. The tuning of this parameter is evaluated in section 6.1. The Voting-based backpressure algorithm executes in time slot t as follows:

- Each queue votes for the Bad neighbor B_{nmd} of node n according to: $B_{nmd} = \max C_{nmd}$.
- Each link (m, n) is assigned temporary weights according to the differential backlog $wt_{nmd}(t) = max(Q_n^d - Q_m^d, 0)$ and a parameter A_{nmd} according to :

$$A_{nmd} = \begin{cases} 1/A, & \text{if } m = B_{nmd} \\ 1 & \text{otherwise.} \end{cases}$$

- Each link is assigned a final weight according to w_{nm} and parameter A_{nmd}:

$$w_{nm}(t) = \max_{d \epsilon D}(wt_{nmd}(t) * A_{nmd}).$$

- The network controller chooses the control action that solves the following optimization:

$$\mu^*(t) = \underset{\mu \epsilon \Gamma}{argmax} \sum_{(n,m)\epsilon L} \mu_{nm} w_{nm}(t)$$

  subject to the interference model mentioned above where adjacent links are not allowed to be active simultaneously.

In each time slot only nodes that transmit or receive packets update the information in the appropriate queues decreasing the complexity of the algorithm. Except from the worst node, also the good node G_{nmd} ($G_{nmd} = \min C_{nmd}$) or the nodes to send packets can be retrieved from the above procedure giving an extra bonus to this node for the corresponding queue. Good nodes should be the ones with the least or even zero votes, meaning that no packet has visited them in the previous hop. Parameter A_{nmd} would be in that way:

$$A_{nmd} = \begin{cases} A, & \text{if } m = G_{nmd} \\ 1/A, & \text{if } m = B_{nmd} \\ 1 & \text{otherwise.} \end{cases}$$

As shown in the experiments (cf. subsection 6.1), this algorithm can significantly reduce delays in low-loaded wireless networks, and moreover, it can be used in combination with other backpressure-type algorithms to enhance their performance. The algorithm shows extremely good performance in networks with very low connectivity between nodes where only a few paths exist for the packets to follow in order to reach their destinations. The packets by holding the information about the previously visited nodes help the fast propagation of the information.

Even though a detailed evaluation of the algorithm is presented in a later section, to support the above claims we tested the algorithm in a ring network consisting of twelve nodes (see Figure 1). The performance of $\mathcal{VoBP}$ is similar to that of the shortest path backpressure. This performance is achieved without the need of any extra queues or other information, like all the distances among the nodes, except from the last visited node that every packet holds.

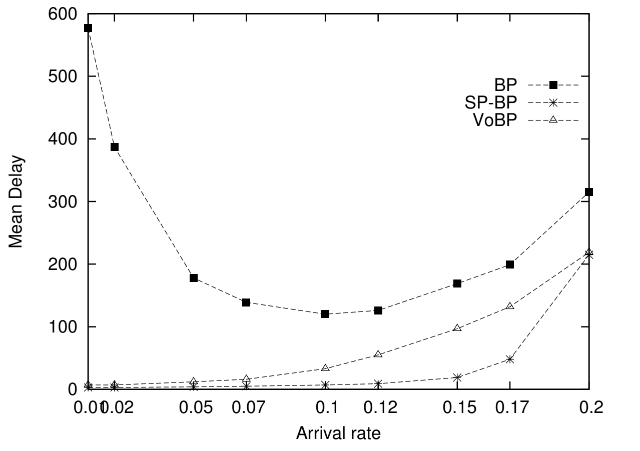


Figure 1. Performance of VoBP in a 12-node ring network

5. Layered backpressure

This section describes a delay-efficient backpressure algorithm based on the creation of *layers*[2] in the network. The main idea is to split the network into layers according to the connectivity among them, which also

[2]We use the term *layer* to describe node partitioning, since the target of this research is underwater sensor networks for surveillance applications, where sensors are placed in layers beneath the sea surface.

(usually) implies geographic proximity, as well. These layers divide the initial graph to k smaller networks. Then the algorithm forwards packets according to the destination layer ID, thus effectively reducing the long paths. In some sense, these layers play the role of attractors, which attract the packets destined for them and then "disallow" the packets to leave the layer. Apparently, if we have only one layer, the algorithm reduces to the original backpressure algorithm.

For the implementation of the $\mathcal{LayBP}$ scheduling policy, every node n keeps a separate queue for each destination/flow going through it and also holds the layer $Layer(n)$ that the destination belongs to. The backpressure scheduling is executed using only the IDs of these layers. This is a significant difference from the work of [4], because $\mathcal{LayBP}$ does not require the knowledge of gateways' queue lengths. The "correct" partitioning of nodes into layers and a proximity-based naming of layers is of paramount importance for the performance of every routing algorithm which based on clustering. Since in the proposed $\mathcal{LayBP}$ algorithm no gateways are required the algorithm is more robust to bad partitioning. Also the algorithm, as it is shown in section 6.1, performs better as the number of clusters grow, in contrast to $\mathcal{CB}-\mathcal{BP}$. Apparently, the concept of "correctness" is algorithmic, and we only need to set a partitioning criterion. Therefore we propose to determine the number of layers based on the network topology, i.e., connectivity. The next two subsections describe the layer-creation and layer-naming algorithm, whereas subsection 5.3 presents the Layered Backpressure packet scheduling algorithm.

5.1. Random-walk-based layering

In this section, we present a random walk based clustering (RWC) method for creating the necessary layers. RWC runs in the initiation phase of the network. The method is centralized and the number of layers Nc to be created are discovered by an exhaustive cost-optimal algorithm. Techniques for estimating this number can be found at [11]. The layers created by this algorithm have approximately the same cardinality, which is denoted by the parameter Npc. Every node has a parameter indicating if the layer it belongs to is "final" or not. At the beginning of the algorithm all nodes are assigned a random layer ID, from *1* to Nc. The final layer of each node is determined by the RWC algorithm after a number of iterations. For all the nodes that do not have their layers fixed by the algorithm yet, we say that they belong to a "temporary" layer.

The algorithm runs in blocks where in each block a node i is selected as a source. The algorithm takes successive random steps from i for a predefined number of steps h for K iterations. Every node holds a counter indicating how many times it is visited from the random walk algorithm in the current block. The neighborhood of the node i is created according to Definition 2.

Definition 2 (Neighborhood of node i). A node j belongs to the neighborhood Gi of the node i, if the number of times the node is visited is at least equal to the times the RWC traversed node i.

After the creation of the neighborhood Gi the proposed algorithm operates as follows:

1. Find the layer $Poslevel$ having the maximum number of members of the current neighborhood. This is the possible layer of all the nodes that belong to the Gi.
2. If this layer is different from the layer of source node i then find a node outside the neighborhood that has this temporary layer and exchange values (layers) with node i.
3. If the number of members of the $Poslevel$ is less than Npc and greater than one then try to find a node outside the layer with this temporary layer and exchange layers with a node belonging to the Gni but having temporary layer different from the $posLevel$.
4. If the number of nodes that belong to the possible level after the previous steps is Npc with deviation of one then the layer of these nodes is fixed to the $Poslevel$.

The procedure moves in small steps inserting at each iteration at most two nodes in the possible layer leading to a relatively fair clustering of the nodes after its termination. This procedure is repeated for each node belonging to a temporary layer until all the layers are fixed. The nodes that after this procedure still belong to temporary layers fix their values according to the layers of their neighbors.

5.2. Naming of Layers

The layers created by the RWC algorithm have labels that do not represent the actual connectivity (proximity) among them. The Layered Backpressure algorithm, which is based on the RWC algorithm, needs to forward packets according to the layer's labels. In order to work efficiently with layers, $\mathcal{LayBP}$ demands the assignment of labels to layers according to their geographic proximity. After the termination of the RWC, another algorithm is used to perform this task. In the present paper, we use a breadth-first-visiting (BFS) algorithm to carry out this task. Although more efficient algorithms can be employed, like fractal curves, e.g., Hilbert curve, which provide a linear ordering of two-dimensional data according to their proximity, this investigation is beyond the scope of the present work.

5.3. The Layered BackPressure algorithm

After the completion of the grouping and the assignment of IDs to the layers, the actual packet scheduling is performed as follows. Each node n maintains a separate queue of packets for each destination. The length of such a queue is denoted by $Q_n^d(t)$. For every queue $Q_n^d(t)$ the node computes the parameter $Dlevel_n^d$ which represents the absolute difference between current and destination nodes' layer number: $Dlevel_n^d = |Layer(n) - Layer(d)|$. In each time slot t, the network controller observes the queue backlog matrix $Q(t) = (Q_n^d(t))$ and performs the following actions for routing:

Layered BackPressure in time slot t:

- Each link (n,m) is assigned a temporary weight according to the differential backlog $wt_{nmd}(t) = (Q_n^d - Q_m^d)$ and parameter A_{nmd} according to:

$$A_{nmd} = \begin{cases} 2, & \text{if } Dlevel_n^d > Dlevel_m^d \\ 1/2, & \text{if } Dlevel_n^d < Dlevel_m^d \\ 1 & \text{otherwise.} \end{cases}$$

- Each link is assigned a final weight according to w_{nm} and parameter A_{mnd}:

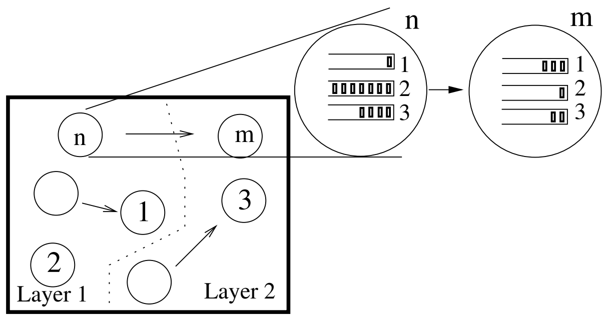
$$w_{nm}(t) = \max_{d \epsilon D}(wt_{nmd}(t) * A_{nmd}).$$

- The network controller chooses the control action that solves the following optimization:

$$\mu^*(t) = \underset{\mu \epsilon \Gamma}{argmax} \sum_{(n,m) \epsilon L} \mu_{nm} w_{nm}(t)$$

  subject to the interference model mentioned above where adjacent links are not allowed to be active simultaneously.

A simple example is illustrated in Figure 2. The network consists of *7* nodes where all nodes are senders and only nodes with id's *1,2,3* are destination nodes. We observe the status of the queue backlogs in a specific time slot t. The network is divided in two layers. Layer one includes nodes on the left side of the dot line. Running the initial backpressure algorithm the commodity that achieves maximum differential backlog for link (n, m) is *2* while for the proposed layered backpressure is *3*. If we assume that link (n, m) is scheduled in that time slot then for the initial backpressure algorithm a packet destined for node *2* will be forwarded to node m pushing it further away and forcing it to follow a longer path in order to reach its destination (node *2*). With the layered backpressure policy at the same time link (n, m) if scheduled forwards a packet of commodity *3* from node n to node m which is closer to the destination of the packet. Furthermore, node m belongs to the same cluster as node n, which according to the proposed layer backpressure policy will prevent the packet from returning to node n.



	Dlevel				
Qnd	n	m	wt{nmd}	Anmd	w{nmd}
1	0	1	-2	1/2	-1
2	0	1	**6**	1/2	3
3	1	0	2	2	**4**

BP : Optimal commodity for link (n,m) on slot t is 2. w(n,m) = 6
LBP : Optimal commodity for link (n,m) on slot t is 3. w(n,m) = 4

Figure 2. An example of the $\mathcal{LayBP}$ execution.

Theorem 1 (Throughput optimality). The $\mathcal{LayBP}$ and $\mathcal{VoBP}$ algorithms are throughput optimal.

Lemma 1 is useful for the proof of the theorem.

Lemma 1. If V, U, μ, A are nonnegative real numbers and $V \leq max[U-, 0] + A$ then
$V^2 \leq U^2 + \mu^2 + A^2 - 2U(\mu - A)$

Proof. The original backpressure algorithm is throughput optimal. In order to prove that the $\mathcal{VoBP}$ and $\mathcal{LayBP}$ are also throughput optimal, the Lyapunov stability criterion is used. The idea behind the Lyapunov drift technique is to define a non-negative function, called the Lyapunov function, which represents the aggregate congestion of all queues (Q_n^d) in the network. The drift of the function in two successive time slots is then taken, and in order for the policy to be throughput optimal, this drift must be negative, when the sum of queue backlogs is sufficiently large. For both policies, we use:

$$L(Q) = \sum_{nd} \theta_n^d (Q_n^d)^2 \tag{1.1}$$

as the Lyapunov function where weights θ_n^d are used to offer priority service.

Recall that each link is assigned a final weight according to w_{nm} and parameter A_{mnd}:

$$w_{nm}(t) = \max_{d \epsilon D}(wt_{nmd}(t) * A_{nmd}). \tag{1.2}$$

The Equation 1.2 can be rewritten in the following form:

$$w_{nm}(t) = \max_{d\epsilon D}(A_{nmd} * Q_n^d - A_{nmd} * Q_m^d). \quad (1.3)$$

which is equivalent to :

$$w_{nm}(t) = \max_{d\epsilon D}(\theta_n^d * Q_n^d - \theta_m^d * Q_m^d), \quad (1.4)$$

where weights θ_i^d are used to offer priority service similar to [13].

Queue dynamics in each time slot satisfy :

$$Q_n^d(t+1) \leq max[Q_n^d(t) - \sum_b \mu_{nb}^d(t), 0] + A_n^d(t) - \sum_a \mu_{an}^d(t) \quad (1.5)$$

,where $\mu_{nm}^d(t)$ are routing control variables, representing the amount of commodity d data delivered over link (n,m) during slot t and $A_n^d(t)$ represent the process of exogenous commodity d data arriving to source node n.

$\sum_b \mu_{nb}^d(t)$ represents the total amount of commodity d delivered over all outgoing links of node n. We assume that only the data $Q_n^d(t)$ currently in node n at the beginning of slot t can be transmitted during that slot. The above expression is an inequality rather than an equality because the actual amount of commodity d data arriving to node n during slot t may be less than $\sum_a \mu_{an}^d(t)$ if the neighboring nodes have little or no commodity d data to transmit.

We apply Lemma 1 to the queuing equation 1.5 and obtain:

$$(Q_n^d(t+1))^2 \leq ([Q_n^d(t))^2 + (\sum_b \mu_{nb}^d(t))^2 + (A_n^d(t) + \sum_a \mu_{an}^d(t))^2 - 2[Q_n^d(t)(\sum_b \mu_{nb}^d(t) - A_n^d(t) - \mu_{an}^d(t)) \quad (1.6)$$

Multiplying both sides with θ_n^d, summing over all valid entries (n,d) and using the fact that the sum of squares of non-negative variables is less than or equal to the square of the sum we take :

$$\sum_{nd} \theta_n^d (Q_n^d(t+1))^2 \leq \sum_{nd} \theta_n^d (Q_n^d(t))^2 + \sum_{nd} \theta_n^d (\sum_b \mu_{nb}^d(t))^2 + \sum_{nd} \theta_n^d (A_n^d(t) + \sum_a \mu_{an}^d(t))^2 - 2\sum_{nd} \theta_n^d Q_n^d(t)(\sum_b \mu_{nb}^d(t) - A_n^d(t) - \mu_{an}^d(t)) \quad (1.7)$$

The equation can be rewritten:

$$\Delta L(Q) \leq 2BN - 2\sum_{nd} \theta_n^d \epsilon Q_n^d(t) \quad (1.8)$$

,where

$$B \triangleq \frac{1}{2N} \sum_{n\varepsilon N} \theta_{max}[(\mu_{max,n}^{out})^2 + (A_n^{max} + \mu_{max,n}^{in})^2] \quad (1.9)$$

Using the above we can rewrite drift inequality :

$$\Delta L(Q) \leq 2B'N\theta_{max} - 2\sum_{nd} \theta_n^d \epsilon Q_n^d(t) \quad (1.10)$$

,where

$$B' \triangleq \frac{1}{2N^2} \sum_{n\varepsilon N} [(\mu_{max,n}^{out})^2 + (A_n^{max} + \mu_{max,n}^{in})^2] \quad (1.11)$$

This drift inequality is in the exact form for application of the Lyapunov drift lemma, proving the stability of the algorithm.

The weighted sum of all queues is :

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t} E\{\sum_{n,d} \theta_n^d Q_n^d(\tau)\} \leq \frac{NB'\theta_{max}}{\epsilon_{max}} \quad (1.12)$$

□

In $\mathcal{VoBP}$ algorithm where the weights are dynamically varying over time, such that $\theta_n^d \in [\theta_{min}, \theta_{max}]$, network stability is still ensured [13].

5.4. The Enhanced Layered Backpressure policy

Routing protocols must be dynamic in order to cope with the mobility of nodes in modern wireless networks. Widely varying mobility characteristics are expected to have a significant impact on the performance of the routing protocols that are based on node grouping (like $\mathcal{CB}-\mathcal{BP}$, $\mathcal{LayBP}$) in order to route packets even if links among nodes are updated. In case of grouping-based routing algorithms, the high mobility of nodes which leads them to change groups, degrades the performance of the algorithms since this 'wrong' information is used in the routing procedure. Although $\mathcal{LayBP}$ does not use gateways, it still suffers from this behavior if the layer that the moving nodes belong to, is not updated. The differential backlog of each link is computed according to the difference between current node's layer and destination's node layer. It is clear that the $\mathcal{LayBP}$ behavior can be affected by 'misplaced' nodes. In this case a packet/packets may be forwarded to layers different from the desired making the algorithm inappropriate.

Looking at Figure 3 we see that node *7* (moving node) initially belongs to cluster *2* at moment *T0*. At moment *T1* node *7* moved to cluster *1* but is agnostic of it. The packets destined to node *7* are still forwarded by the $\mathcal{LayBP}$ to cluster *2* making it difficult to reach their final destination thus making the intermediate node's queues to grow up.

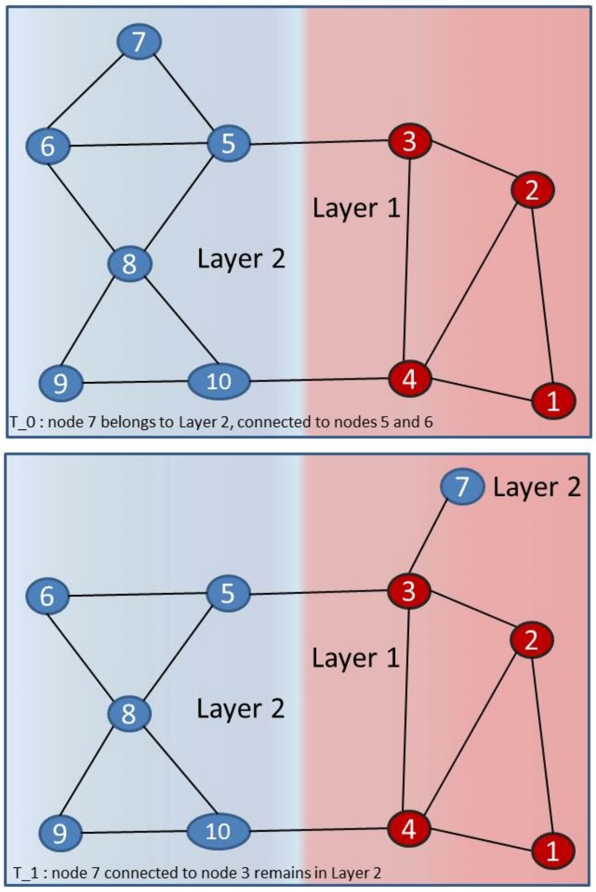

7
6
5
3
2
1
4
8
9
10
Layer 1
Layer 2
T_0 : node 7 belongs to Layer 2, connected to nodes 5 and 6
7
Layer 2
6
5
3
2
8
Layer 1
Layer 2
9
10
4
1
T_1 : node 7 connected to node 3 remains in Layer 2

Figure 3. Example network with a moving node.

In order to cope with node mobility we incorporate in the $\mathcal{LayBP}$ algorithm another step. Both moving and static nodes use this additional step in order to recalculate their cluster according to their neighborhood. In the initiation phase every node has a counter $C0_{n,l}$ for every layer ID, indicating how many neighbors belong to it and a variable $Layer_n$ indicating the layer that node n belongs to. In every time slot t the following actions are performed:

- Calculate $C_{n,l}$, the total number of neighbors that belong to every layer detected.
- if $C_{n,l} >= C0_{n,l}$ for $l = Layer_n$ then the moving node remains in the same layer.
- else calculate the layer with the most neighbors $M_{n,l} = maxC_{n,l}$. if $M_{n,l} > C_{n,l}$ then the moving node belongs to layer $M_{n,l}$.
- assign for every layer l, $C0_{n,l} = C_{n,l}$ as new initial values for the next time slot.

This procedure can be executed every k time slots according to how fast we want the system to adapt to node mobility. The moving nodes need to execute the procedure every second in order to find the appropriate layer they belong to. The static nodes need to update their information more rarely, than the moving nodes do, since a certain number of neighbors must be replaced in order to affect them. The procedure does not perform reclustering, but only 'helps' layers incorporate moving nodes.

6. Evaluation

For the evaluation of the proposed scheduling policies we developed a custom simulator, modeling topology features, packet scheduling, link activation, etc. We consider as competitors the state-of-the-art delay-aware backpressure policies that have been published so far in the literature, with the exception of the $\mathcal{MR}-\mathcal{BP}$ policies which were shown in subsection 1.1 to provide no delay performance benefits. We simulate the execution of the algorithms for various network topologies for large simulation times and recorded their average performance. In the interest of space, we show only the most representative results.

6.1. Static networks

Experimental setting. As of competing algorithms we examine currently proposed delay-aware backpressure algorithms along with the original backpressure algorithm as the baseline algorithm. We emphasize here that the shortest path backpressure [5] is theoretically optimal w.r.t. delay. This algorithm performs the best in all cases and acts as the lower bound, but it is vulnerable to topology changes (cf. Figure 10). For the algorithms needing some form of clustering, i.e., $\mathcal{CB}-\mathcal{BP}$ and $\mathcal{LayBP}$, the clustering is performed with the proposed random-walk clustering algorithm. For the $\mathcal{CB}-\mathcal{BP}$ in order to improve the misbehavior of the algorithm when the source and destination are in the same cluster, the traffic controller can directly route them without relaying to any gateway, even if source is a gateway of $cluster(d)$ itself.

Performance metrics. As performance measures we use the average end-to-end delay and the number of messages successfully delivered per unit time (network throughput).

Network topologies. As network topologies, we consider a 4×4 grid network with 16 nodes, and those illustrated in Figure 4. The grid network is very common in the literature [3, 5] and it comprises a very "comfortable" case for $\mathcal{SP}-\mathcal{BP}$ since there are more than one shortest path between each pair of nodes. Also, the other two topologies (those in Figure 4) are borrowed from [9] with *22* and *30* nodes, respectively. For the algorithms that use clusters, the clustering is depicted in the figure; the clusters of the grid net are comprised by the four quadrants. In summary, we double the size of the network (from *16* to *30* nodes) and also variate the type of connectivity among clusters, i.e., clusters in a "ring" (*16*-node net), clusters "almost in a line" (*22*-node net), and clusters in a "all-to-all communication". The performance of the algorithms are similar regardless of the size of the network.

Packet generation. The exogenous arrival processes at each node are independent Bernoulli processes with rate λ. The destination of every generated packet is randomly chosen. Experiments are also conducted where the destination is chosen according to Zipf's law in order to simulate the situations where there is a "preference" toward some specific destination node. Since in the conducted experiments with Zipfian-selected destinations, the obtained results show no alteration in the relative performance of the algorithms, all the results we present concern the uniformly-at-random-selected destinations. Different values of λ are chosen according to the number of nodes in the network in order to simulate moderate network traffic. This means that the same λ might represent low load for a small network, but high traffic for a much larger network.

Experimental results. The goal of the experimental evaluation is, apart from testing the delay and throughput performance of all the competing algorithms, to validate all the claims reported in subsection 1.1 about the misbehavior of $\mathcal{CB}-\mathcal{BP}$ and $\mathcal{SP}-\mathcal{BP}$. First of all, we examine the impact of parameter A on the performance of the proposed algorithms so as to tune them for the rest of the experiments.

Parameter A - Tuning of the proposed algorithms. Both $\mathcal{V}o\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$ use the parameter A to forward packets; $\mathcal{L}ay\mathcal{BP}$ uses it to forward packets to the destination cluster according to layers' IDs, and in $\mathcal{V}o\mathcal{BP}$, the parameter A plays the role of discouraging packets to move backwards in the network. The case of $A = 1$ corresponds to the traditional backpressure algorithm. We evaluate the performance of the algorithms for various A and we present in (Figure 5) the ones concerning $\mathcal{L}ay\mathcal{BP}$ ($\mathcal{V}o\mathcal{BP}$ algorithm showed similar performance). The obtained results show no significant gains (either in terms of delay reduction or throughput increase) for values other than A=*2*; thus, for the rest of the experiments, we use A=*2*.

Impact of layer/cluster number. In section 1.1, we stated that the performance of $\mathcal{CB}-\mathcal{BP}$ deteriorates with the increasing number of clusters; indeed this is the pattern of its behavior, as shown in Figure 6 (tested on a grid topology growing with the number of clusters), where beyond *5* clusters, the performance of $\mathcal{CB}-\mathcal{BP}$ is almost *25* times worse than that of $\mathcal{L}ay\mathcal{BP}$. The same behavior is observed in all the network topologies we simulate. That behavior is predicted by the description in [4]. From this description it is evident that the increased number of clusters implies an increased number of gateways and thus more delays. $\mathcal{CB}-\mathcal{BP}$ requires a very small constant number of clusters, though most ad hoc clustering algorithms produce a number of clusters which is linear or logarithmic in the number of network nodes. Moreover, the proposed $\mathcal{L}ay\mathcal{BP}$ algorithm performs better as the network is divided in more clusters (see Figure 6). This positive characteristic makes the proposed routing algorithm more efficient when combined with most of the automated algorithmic network clustering algorithms.

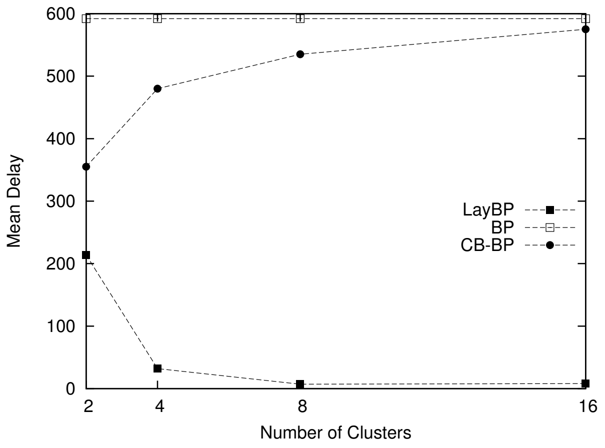
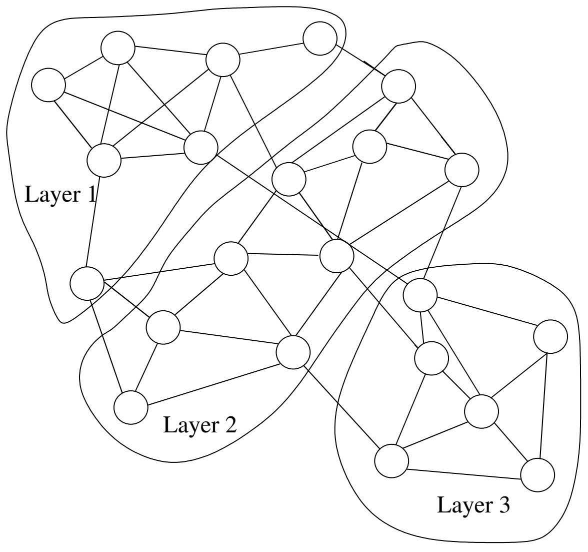


Figure 6. Impact of the number of clusters on the grid network topology.

Delay and throughput performance. The plots in Figure 7 depict the delay performance of the competing algorithms. The first generic observation is that the performance of $\mathcal{L}ay\mathcal{BP}$ follows – as a trend – the performance of $\mathcal{SP}-\mathcal{BP}$, whereas the performance of $\mathcal{CB}-\mathcal{BP}$ follows – as a trend – that of the original $\mathcal{BP}$. Additionally, in all cases $\mathcal{L}ay\mathcal{BP}$ is the best policy outperformed only by $\mathcal{SP}-\mathcal{BP}$ (the theoretically optimal policy).

The $\mathcal{V}o\mathcal{BP}$ algorithm performs better than all the algorithms (except of course from $\mathcal{SP}-\mathcal{BP}$), when the network traffic is extremely low and in the topologies where there do not exist many alternative paths for the traveling packets. This behavior of the $\mathcal{V}o\mathcal{BP}$ algorithm is directly associated with the average node degree of the network. For instance, the delay of $\mathcal{V}o\mathcal{BP}$ is significantly higher (see Figure 7c) for the *30*-node network (presented in Figure 4b) which is quite dense.



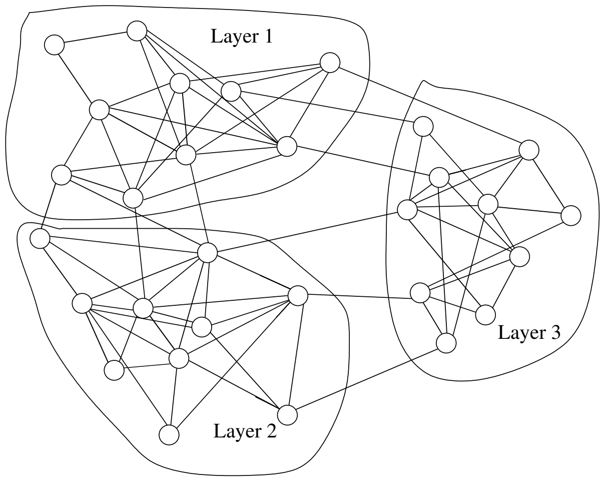
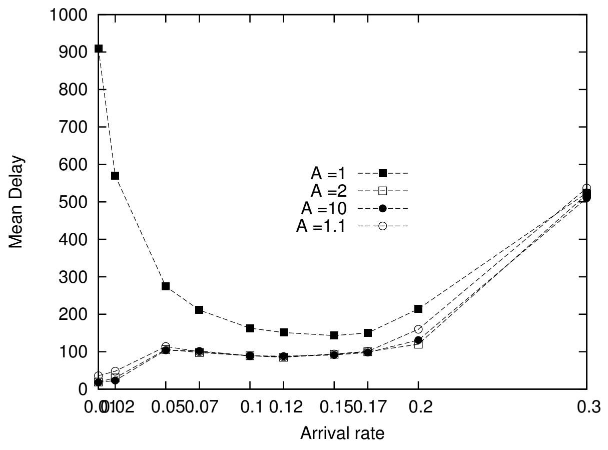


Figure 4. Network topologies with 22 and 30 nodes.



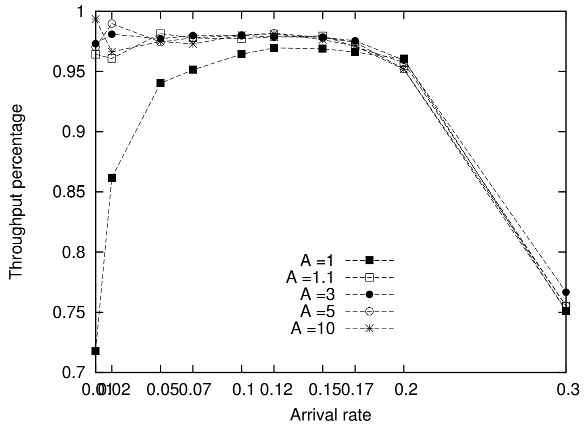


Figure 5. Tuning of parameter A - delay prformance of $\mathcal{LayBP}$

The plots in Figure 8 depict the throughput performance of the competing algorithms. The obvious observation concerns the optimality of all algorithms, including optimality of $\mathcal{VoBP}$ and $\mathcal{LayBP}$ according to Theorem 1.

Maximum end to end delay. The quality of service (QoS) is defined by the following technical parameters: transmission rate, loss rate,average throughput, maximum end-to-end delay, and maximum delay jitter. The maximum end to end delay is an important metric of QoS when critical message transmission is concerned. In Figure 9 the maximum end to end delay is shown for the *30* node topology. We observe that both $\mathcal{BP}$ and $\mathcal{CB}-\mathcal{BP}$ undergo a high maximum end to end delay for low and moderate traffic. Both proposed algorithms manage to keep the maximum end to end delay low while they even outperform $\mathcal{SP}-\mathcal{BP}$ for moderate network traffic. This is due to the fact that $\mathcal{SP}-\mathcal{BP}$ guide all packets through short routes leading to congestion over the links that compose these paths and making packets suffer from very long delays.

Impact of topology change (link breakdown). To validate the claims about the impact of topology on the shortest path backpressure algorithm, we perform an experiment on a small sample network of *10* nodes. The network consists of two clusters; the first includes six nodes and the second cluster includes four nodes. The clusters are connected to each other with only two links. We inactivate one of the two links that connect the two clusters without recalculating the shortest paths among all node pairs. The results presented in Figure 10 show that $\mathcal{SP}-\mathcal{BP}$ practically breaks down – as expected

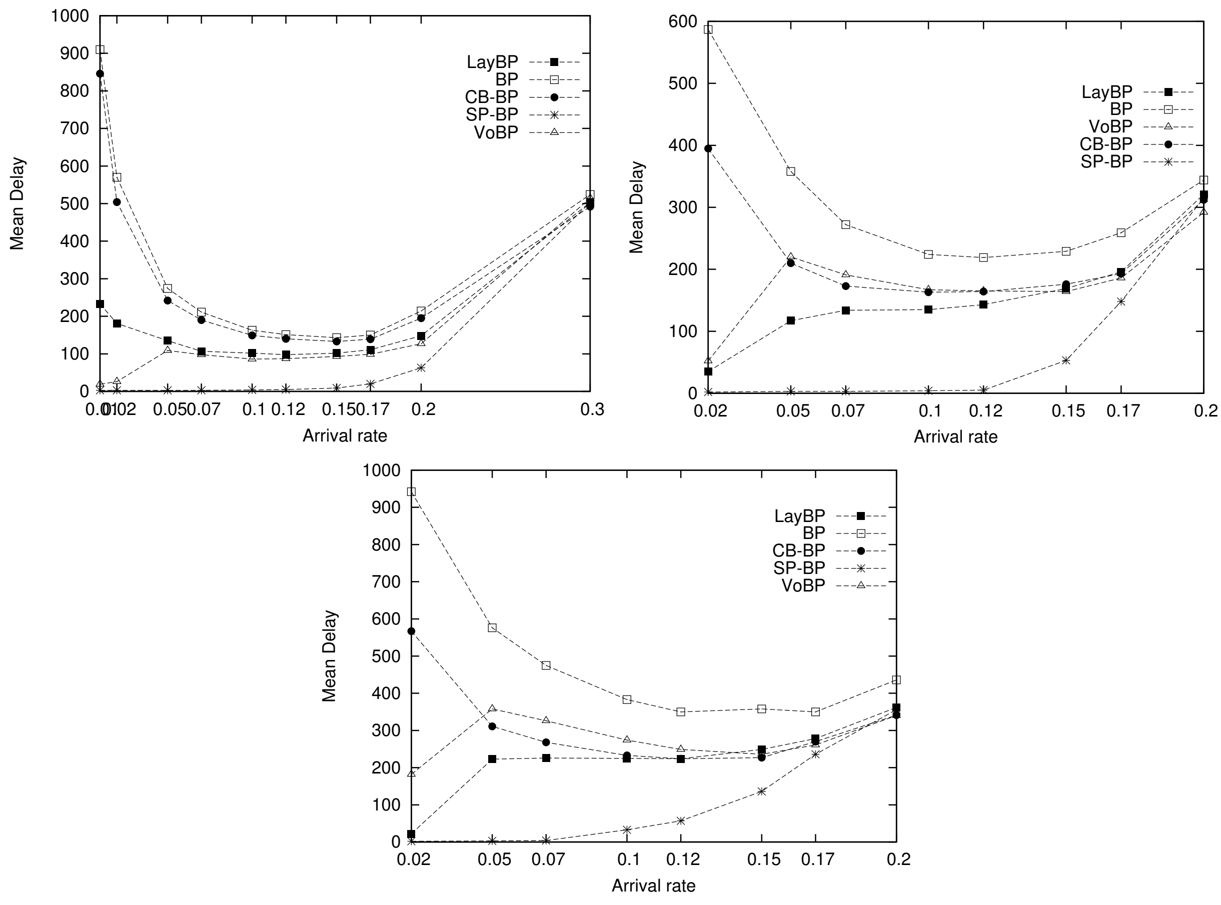


Figure 7. Delay performance for the three network topologies (16, 22 and 30 nodes).

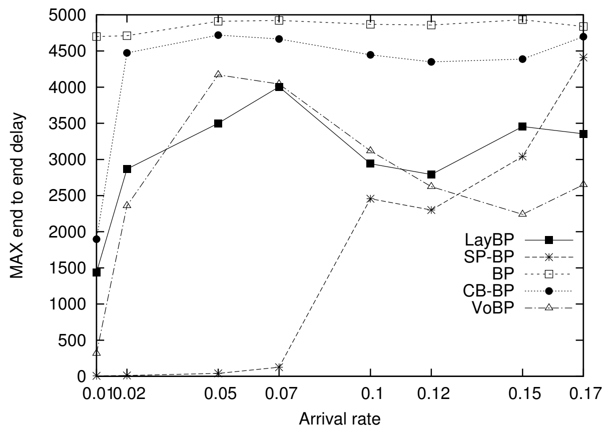


Figure 9. Maximum end to end delay for the network topology with 30 nodes

– whereas $\mathcal{L}ay\mathcal{BP}$ maintains its strength. We expected such dramatic changes to have an immediate impact on $\mathcal{L}ay\mathcal{BP}$ also, but $\mathcal{L}ay\mathcal{BP}$ is slighty affected as shown in Figure 10. $\mathcal{L}ay\mathcal{BP}$ is more robust to topology changes since it does not require exact knowledge of the shortest paths among nodes, but rather shortest paths among clusters.

Since both proposed $\mathcal{V}o\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$ algorithms favor packets to follow delay efficient paths, but in neither case in a deterministic way, the routing algorithm may get "stuck" only temporarily due to link breakdown. Both algorithms manage to overcome effectively link breakdown, a common situation in ad hoc networks, for both low and moderate network traffic.

Load Deviation In sensor networks an important performance metric of a routing algorithm is load deviation which is strictly related to energy consumption. Due to the limited energy resources, there is a need for a routing algorithm to balance the load among all nodes while keeping the total load relatively low. In Figure 11 the load deviation of the typical *16* grid network is presented. Is is shown that both $\mathcal{L}ay\mathcal{BP}$ and $\mathcal{V}o\mathcal{BP}$ outperform $\mathcal{CB}-\mathcal{BP}$ and $\mathcal{BP}$. On the other hand, though $\mathcal{SP}-\mathcal{BP}$ algorithm has the lower load deviation, this does not clearly represent its performance in terms of energy consumption. Complex calculations that all nodes need to perform every second impose further energy

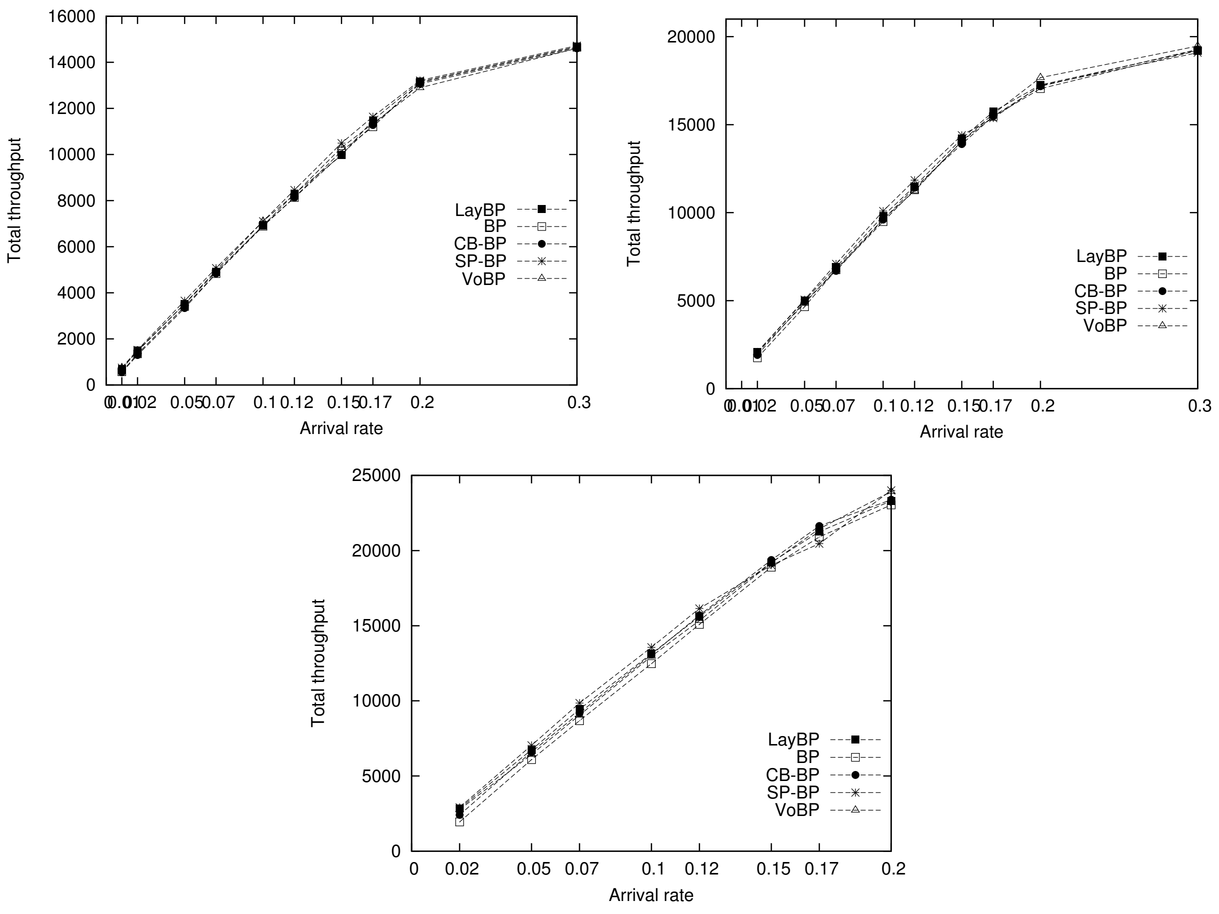


Figure 8. Throughput performance for the three network topologies (16, 22 and 30 nodes).

consumption. This energy consumption is not calculated in the present simulation and thus the performance of $\mathcal{SP}-\mathcal{BP}$ is over estimated.

6.2. Dynamic networks

Experimental setting. We evaluate the performance of the cluster-based algorithms in networks with mobile nodes in order to measure the adaptivity of the $E\mathcal{L}-\mathcal{BP}$ algorithm. It is important here to mention that $E\mathcal{L}-\mathcal{BP}$ copes with topology changes in terms of mobility of nodes, which is different than the case evaluated earlier, where a link breakdown was used in order to show the lack of adaptivity of $\mathcal{SP}-\mathcal{BP}$ algorithm to topology changes. For the evaluations we conduct, we assume that moving nodes update their link information, according to some handshaking mechanism, but are unaware of layer changes.

Network topologies. As network topology, we consider that illustrated in Figure 12. The topology is a network of *13* nodes where node *13* moves from layer *1* (time slot *0*) to layer *2* (time slot *1000*) and finally to layer *3* (time slot *2000*). $\mathcal{BP}$, $\mathcal{CB}-\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$ are unaware of layer change while all the active links of the moving node are updated. In $E\mathcal{L}-\mathcal{BP}$ algorithm, the update procedure runs in every time slot and a moving node joins an appropriate layer according to its neighbor's information.

Packet generation. In order to have a clear view of the performance of the cluster-based algorithms we generate in each experiment only one flow from node *1* to the moving node. The exogenous arrival processes are independent Bernoulli processes with rate λ.

Experimental results. Delay and throughput performance. It is clear that $\mathcal{CB}-\mathcal{BP}$ brakes down in situations where nodes change clusters. Based on the fact that $\mathcal{CB}-\mathcal{BP}$ can direct packets to the correspondent queue, only after they have entered the correct cluster, makes evident that its performance under such situations is severely bad. Indeed, in our simulations, the performance of $\mathcal{CB}-\mathcal{BP}$ for such cases is *20* times worse than the rest algorithms, and thus we do not show it in the plots. The plots in Figure 13 show the delay and the throughput performance of $\mathcal{L}ay\mathcal{BP}$, $E\mathcal{L}-\mathcal{BP}$ and the original backpressure algorithm.

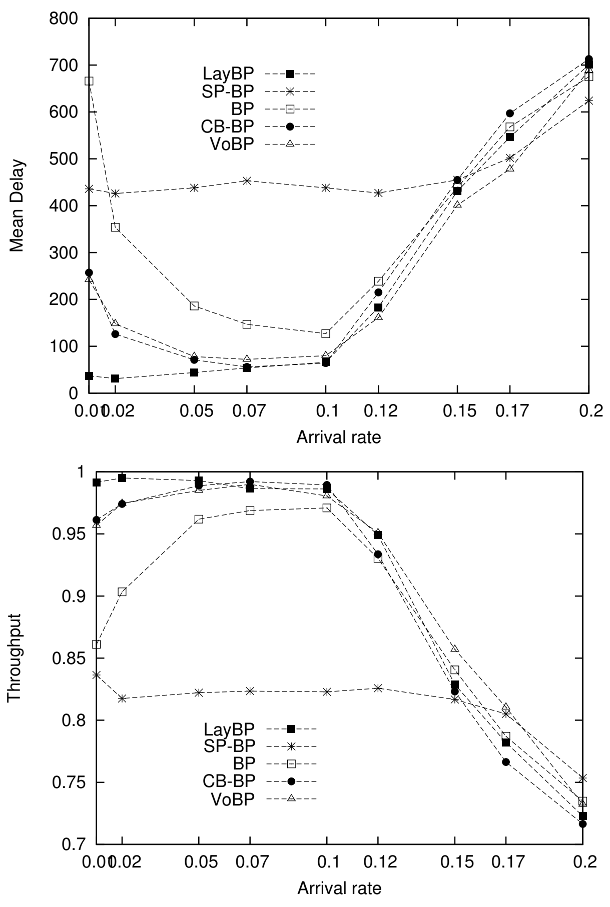


Figure 10. Impact of topology changes on the delay and throughput performance of $\mathcal{SP}-\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$.

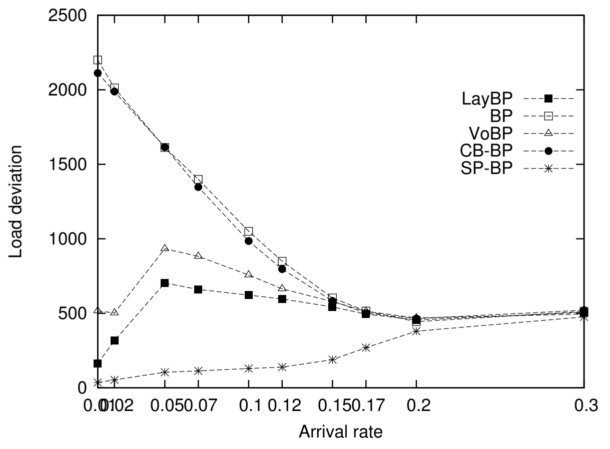


Figure 11. Load deviation for the grid network topology

By studying the delay performance of the compared algorithms,we see that $\mathcal{L}ay\mathcal{BP}$ behaves worse than the original $\mathcal{BP}$ in cases of moving nodes. The $E\mathcal{L}-\mathcal{BP}$ algorithm gives the best outcome since a moving node updates the layer information according to its neighborhood. It is clear that in situations where nodes move along many layers (second dynamic topology) the behavior of $\mathcal{L}ay\mathcal{BP}$ degrades, while $E\mathcal{L}-\mathcal{BP}$ retains its behavior. All the algorithms give similar outcomes as far as the number of packets successfully delivered per unit time is concerned (Network Throughput).

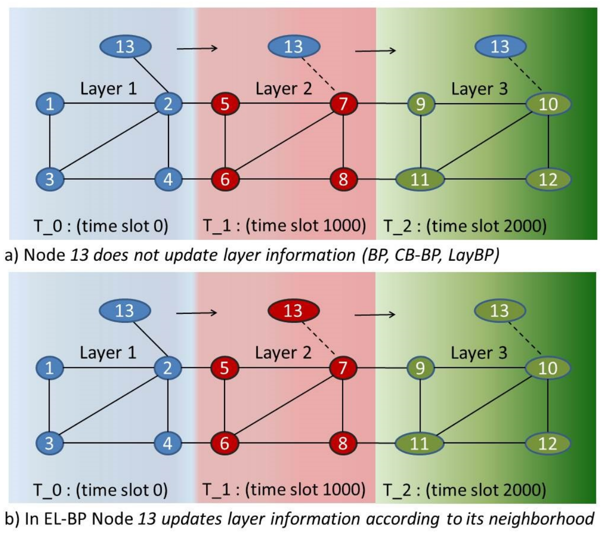

Layer 1
Layer 2
Layer 3
T_0 : (time slot 0)
T_1 : (time slot 1000)
T_2 : (time slot 2000)
a) Node 13 does not update layer information (BP, CB-BP, LayBP)
b) In EL-BP Node 13 updates layer information according to its neighborhood

Figure 12. Dynamic network with 3 layers.

7. Relevant work

A lot of packet scheduling algorithms have been proposed in the relevant literature. Algorithms for packet scheduling/routing can be classified into data-centric, hierarchical, location-based and those based on network flow or quality of service awareness. The interested reader should refer to [14] for detailed categorization. In data-centric protocols [15], aggregation of data during the relaying of data is utilized. Hierarchical routing algorithms [16] assign different roles to nodes. The nodes are divided into clusters and cluster-heads are assigned with the role of data aggregation and reduction in order to save energy. Location-based [17] protocols on the other hand use the position information of the nodes in order to forward the information to the desired region rather than the whole network. In some approaches, route setup is modeled and solved as a network flow problem [2]. System stability and throughput maximization based on [2] have been exclusively studied [18–21]. Related work on delay-efficient backpressure algorithms is developed in [3, 5, 22–26].

8. Conclusions and future work

This paper considers the problem of packet scheduling in wireless networks using backpressure-type algorithms. The purpose of the work is to address the delay-efficiency

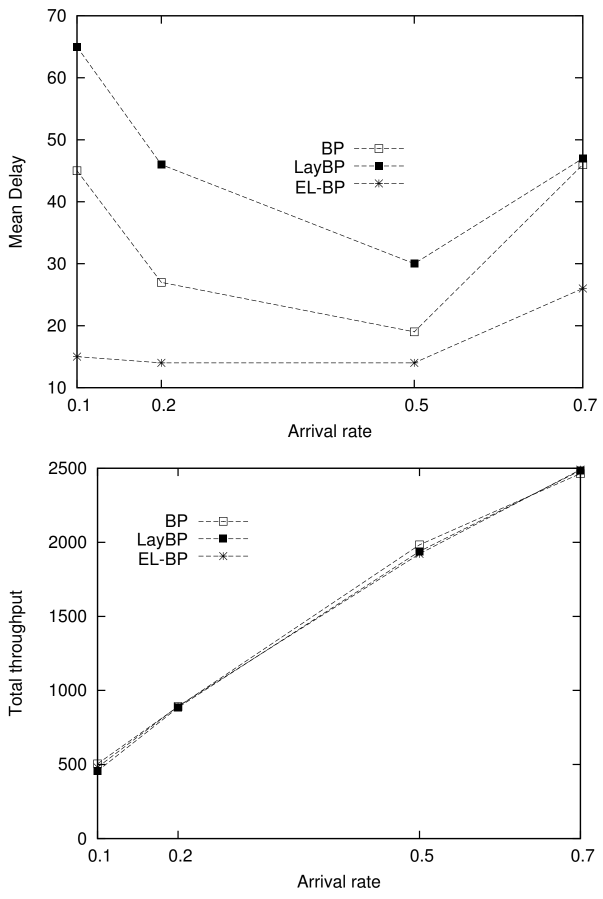


Figure 13. Impact of moving nodes on the delay and on the throughput performance of $\mathcal{BP}$, $\mathcal{LayBP}$ and $\mathcal{EL{-}BP}$ for the dynamic network with the 3 layers.

problems of the backpressure-type algorithms. In this context, the paper describes two algorithms. The first algorithm, namely $\mathcal{VoBP}$, alleviates the ping-pong effect in packet scheduling. The second algorithm, namely $\mathcal{LayBP}$, investigates the issue of creating layers of nodes and performing backpressure scheduling using the IDs of the layers in order to "push" the packets toward the destination layer.

We conclude that the Voting backpressure performs ideally for low connectivity topologies while the Layered backpressure is a high performance policy compromising a little delay for robustness, low computational complexity and simplicity. Robustness of a routing algorithm in mobility and different connectivity of nodes is a very important aspect of wireless communications and a variation of Layered backpressure algorithm, namely the Enhanced Layered Backpressure, can adapt to moving nodes with exchange of local information between nodes. Our future work involves the combination of *voting* and *layering*, i.e., application of voting inside layers and a more thorough approach on mobile - networks with the use of $\mathcal{EL{-}BP}$.

References

[1] Maglaras, L.A. and Katsaros, D. (2011) Layered backpressure scheduling for delay reduction in ad hoc networks. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a* (IEEE): 1–9.

[2] Tassiulas, L. and Ephremides, A. (1992) Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE TAC* **37**(12): 1936–1948.

[3] Bui, L., Srikant, R. and Stolyar, A. (2009) Novel architectures and algorithms for delay reduction in backpressure scheduling and routing. In *INFOCOM 2009, IEEE* (IEEE): 2936–2940.

[4] Ying, L., Srikant, R., Towsley, D. and Liu, S. (2011) Cluster-based back-pressure routing algorithm. *Networking, IEEE/ACM Transactions on* **19**(6): 1773–1786.

[5] Ying, L., Shakkottai, S., Reddy, A. and Liu, S. (2011) On combining shortest-path and back-pressure routing over multihop wireless networks. *IEEE/ACM Transactions on Networking (TON)* **19**(3): 841–854.

[6] Huang, L., Moeller, S., Neely, M.J. and Krishnamachari, B. (2013) Lifo-backpressure achieves near-optimal utility-delay tradeoff. *Networking, IEEE/ACM Transactions on* **21**(3): 831–844.

[7] Moeller, S., Sridharan, A., Krishnamachari, B. and Gnawali, O. (2010) Routing without routes: The backpressure collection protocol. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks* (ACM): 279–290.

[8] Athanasopoulou, E., Bui, L.X., Ji, T., Srikant, R. and Stolyar, A. (2013) Back-pressure-based packet-by-packet adaptive routing in communication networks. *Networking, IEEE/ACM Transactions on* **21**(1): 244–257.

[9] Amis, A.D. and Prakash, R., Clusterhead selection in wireless ad hoc networks. US Patent 6,829,222 B2, 2004.

[10] Basagni, S., Mastrogiovanni, M., Panconesi, A. and Petrioli, C. (2006) Localized protocols for ad hoc clustering and backbone formation: A performance comparison. *IEEE TPDS* **17**(4): 292–306.

[11] Katsaros, D., Dimokas, N. and Tassiulas, L. (2010) Social network analysis concepts in the design of wireless ad hoc network protocols. *IEEE Network magazine* **24**(6): 2–9.

[12] Dorigo, M., Maniezzo, V. and Colorni, A. (1996) Ant system: Optimization by a colony of cooperating agents. *IEEE TSMC: Part B* **26**(1): 29–41.

[13] Neely, M.J., Modiano, E. and Rohrs, C.E. (2005) Dynamic power allocation and routing for time varying wireless networks. *IEEE JSAC* **23**(1): 89–103.

[14] Akkaya, K. and Younis, M. (2005) A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks* **3**(3): 325–349.

[15] Intanagonwiwat, C., Govindan, R. and Estrin, D. (2000) Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking* (ACM): 56–67.

[16] Manjeshwar, A. and Agrawal, D.P. (2001) Teen: a routing protocol for enhanced efficiency in wireless sensor networks. In *Parallel and Distributed Processing Symposium, International* (IEEE Computer Society), **3**: 30189a–30189a.

[17] Xu, Y., Heidemann, J. and Estrin, D. (2001) Geography-informed energy conservation for ad hoc routing. In *Proceedings of the 7th annual international conference on Mobile computing and networking* (ACM): 70–84.

[18] Neely, M.J., Modiano, E. and Li, C.P. (2008) Fairness and optimal stochastic control for heterogeneous networks. *Networking, IEEE/ACM Transactions on* **16**(2): 396–409.

[19] Stolyar, A.L. (2005) Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems* **50**(4): 401–457.

[20] Markakis, M.G., Modiano, E. and Tsitsiklis, J.N. (2014) Max-weight scheduling in queueing networks with heavy-tailed traffic. *IEEE/ACM Transactions on Networking (TON)* **22**(1): 257–270.

[21] Li, R., Eryilmaz, A. and Li, B. (2013) Throughput-optimal wireless scheduling with regulated inter-service times. In *INFOCOM, 2013 Proceedings IEEE*: 2616–2624. doi:10.1109/INFCOM.2013.6567069.

[22] Gupta, G. and Shroff, N. (2009) Delay analysis for multi-hop wireless networks. In *INFOCOM 2009, IEEE*: 2356–2364. doi:10.1109/INFCOM.2009.5062162.

[23] Huang, L. and Neely, M.J. (2011) Delay efficient scheduling via redundant constraints in multihop networks. *Performance Evaluation* **68**(8): 670 – 689. doi:http://dx.doi.org/10.1016/j.peva.2011.02.001. Special Issue: Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks: selected papers from WiOpt 2010.

[24] Huang, L. and Neely, M. (2011) Delay reduction via lagrange multipliers in stochastic network optimization. *Automatic Control, IEEE Transactions on* **56**(4): 842–857. doi:10.1109/TAC.2010.2067371.

[25] Xue, D. and Ekici, E. (2013) Delay-guaranteed cross-layer scheduling in multihop wireless networks. *IEEE/ACM Transactions on Networking (TON)* **21**(6): 1696–1707.

[26] Jones, N.M., Paschos, G.S., Shrader, B. and Modiano, E. (2014) An overlay architecture for throughput optimal multipath routing. In *Proceedings of the 15th ACM International Symposium on Mobile Ad Hoc Networking and Computing* (ACM): 73–82. doi:10.1145/2632951.2632957.