Layered Backpressure Scheduling for Delay Reduction in Ad Hoc Networks

Leandros A. Maglaras   Dimitrios Katsaros
Department of Computer & Communication Engineering, University of Thessaly, Volos, Greece

Abstract

Packet scheduling in wireless ad hoc networks is a fundamental problem for ad hoc networking. Backpressure scheduling is a solid and throughput optimal policy for such networks, but suffers from increased delays. In this article, we present an holistic approach to improve upon the delay problems of backpressure-type algorithms. We develop two scheduling policies, namely Voting-based and Layered backpressure routing which both are throughput optimal. We experimentally compare the proposed policies against all the delay-aware backpressure policies and conclude that Layered backpressure is a high performance policy compromising a little delay for robustness, low computational complexity and simplicity.

***Keywords*-Delay, throughput, backpressure, packet scheduling, ad hoc networks.**

I. Introduction

Wireless ad hoc (multihop) 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 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, 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 (link/node failures, node mobility) 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 [16]; it is based on the Lyapunov drift theory, and it is known as the *backpressure* ($\mathcal{BP}$) packet scheduling algorithm. Subsequently, the original concept spawn several lines of research in 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 net effect of this mechanism is to increase delay and also to increase the energy consumption of the nodes that play the role of relays. Delay and energy 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 underwater acoustic sensornets which are the target of our work.

To circumvent the delay problems of backpressure, the *shadow queue* algorithm [5] forces the links to stay inactive in order to lead the network to work in a burst mode, since for periods where 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 [19] and thus the computational overhead, and as a side-effect, reduce the delay. To alleviate the delay problems of backpressure, scheduling based on the *combination of backpressure and shortest-paths* have been proposed [18], 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

Research supported by the project "Control for Coordination of Distributed Systems", funded by the EU.ICT program, Challenge ICT-2007.3.7.

efficient. Finally, some recent ideas [8], [12] could be incorporated in an *orthogonal way* to improve analogously the delay performance of all policies at the expense of throughput optimality, but this is beyond the scope of this paper. Here, we take an holistic approach in designing an efficient delay-aware backpressure policy, that is also practical – it has low computational overhead and it is robust to topology changes.

A. Motivation

In the shadow queues ($\mathcal{SQ}-\mathcal{BP}$) methods [3], [5], backpressure is used with a parameter M that forces the links to stay inactive as long as their differential backlog doesn't exceed the value of M, i.e., links with backpressure less than M can not be scheduled. This means that packets stay in queues longer, which can lead to higher delays. Indeed, we tested this intuitive result by evaluating the delay performance of these methods (see a sample performance in Figure 1) and confirmed this behavior. Therefore, since the $\mathcal{SQ}-\mathcal{BP}$ methods are not delay competitive, we do not consider them as viable competitors any more.

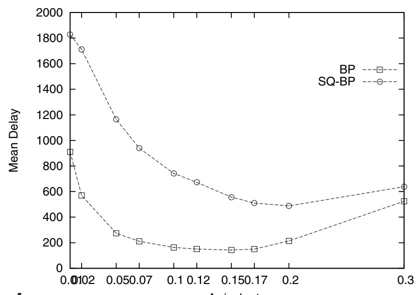


Figure 1. Performance of the $\mathcal{SQ}-\mathcal{BP}$ policy in a $4x4$ grid network ($M = 2$).

The authors of [19] identified the scalability problems of the backpressure policy 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-effect" has the additional benefit of delay reduction. Their policy, namely *cluster-based backpressure* ($\mathcal{CB}-\mathcal{BP}$), requires maintaining one queue per *gateway* at each relay node which leads to an excessive number of gateways, which is 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 [2], [4] (such as the Distributed Clustering Algorithm, 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 delay of $\mathcal{CB}-\mathcal{BP}$ approaches asymptotically the delays of the original $\mathcal{BP}$ (cf. Figure 5c). 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 $\mathcal{CB}-\mathcal{BP}$ method, it is evident that even when a packet has already reached the destination cluster (i.e., the cluster where the destination nodes resides), the method 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 was detected and improved in our experiments (cf. subsection VI-A).

The Shortest-paths backpressure ($\mathcal{SP}-\mathcal{BP}$) method [18] assumes the precomputation of all pairwise-node distances and then application of the backpressure notion on the shortest path(s) between source and destination. Apparently, this method 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 would increase delays. Apart from these computational-type problems, frequent topology changes would lead the method to break down, since many shortest paths would not exist anymore. Finally, in topologies where there is only one shortest-path per node pair (which is the most common case), $\mathcal{SP}-\mathcal{BP}$ would rapidly consume the energy of that path, shortening the longevity of the network. This problem becomes worse in topologies where a lot of shortest paths traverse the same set of nodes (i.e., nodes with high betweenness centrality [10]); in such topologies, these nodes deplete their energy so fast, that the network gets partitioned very rapidly.

B. Contributions

The present article develops two scheduling policies, namely the *Voting-based Backpressure ($\mathcal{V}o\mathcal{BP}$)* and the *Layered Backpressure ($\mathcal{L}ay\mathcal{BP}$)*. For the former method, 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 [6], 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 $\mathcal{L}ay\mathcal{BP}$ method, taking a less localized approach, we create "layers" of nodes and use the identities of these

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

layers to forward the packets toward the destination's layer, and "discouraging" the packet from leaving the destination layer; these layers act as attractors for the packets.

The article makes the following contributions:

- It experimentally evaluates – for the first time in the literature – all the existing delay-aware backpressure policies, namely $\mathcal{BP}$, $\mathcal{SQ}-\mathcal{BP}$, $\mathcal{CB}-\mathcal{BP}$, $\mathcal{SP}-\mathcal{BP}$ for several network topologies.
- It develops the *Voting-based Backpressure* policy, which "wipes-outs" the ping-pong effect in backpressure-based packet scheduling.
- It develops the *Layered BackPressure ($\mathcal{LayBP}$)*, which divides the network into "layers" according to the connectivity of nodes. This method 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 addition, it can be seen as a "relaxed" version of the $\mathcal{SP}-\mathcal{BP}$ between layers where packets are not forced to travel the shortest-path among nodes, but the packets are "suggested" to follow the shortest path from the source to the destination layer. Therefore, the $\mathcal{LayBP}$ is a hybrid among $\mathcal{CB}-\mathcal{BP}$ and $\mathcal{SP}-\mathcal{BP}$, compromising a little delay for robustness, low computational complexity and simplicity.
- It develops a new random-walk based node clustering algorithm, that exploits the connectivity among nodes.
- Evaluates experimentally the performance of the proposed packet scheduling policies, with all the previous delay-aware backpressure methods.

The rest of this article is organized as follows: Section II describes the network model. In Section III the original backpressure scheme is introduced; Section IV presents the $\mathcal{VoBP}$ scheme; Section V describes the $\mathcal{LayBP}$ algorithm; Section VI presents the simulation environment and results. In Section VII we survey the most important works relevant to this article, and finally Section VIII concludes the article.

II. 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, 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 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 inter-arrival times follow the exponential distributions 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.

III. The original Backpressure algorithm

Backpressure [16] 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 at 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) at time slot t.
- *Scheduling*
  The network controller chooses the control action that solves the following optimization problem:
  $$\mu^*(t) = argmax \sum_{n,m} \mu_{nm} w_{nm}(t),$$
  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*
  At 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.

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).

IV. Voting-based backpressure

The driving idea behind the $\mathcal{V}o\mathcal{BP}$ scheduling policy is the reduction of the path length travelled by a packet by reducing any cycles observed in this path. Apparently, this can be done by having the packet "carrying" its trajectory. Such an approach though, would compromise 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 it is not a viable option, unless we confine the amount of information. This is exactly the direction followed by the $\mathcal{V}o\mathcal{BP}$ algorithm.

$\mathcal{V}o\mathcal{BP}$ uses minimum information stored on the packets in order to select the scheduling policy at each time slot. Every packet holds the last node it has previously visited. This information is used in the scheduling phase of the protocol in order to prevent packets from revisiting the same nodes and travelling 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}. In order to find the worst neighbor all the packets that belong to each queue participate in a voting procedure updating the appropriate counters. The node that collects the more votes according to the information of the packets, is given a weight in the scheduling algorithm preventing the controller from choosing such a routing policy that moves the packets backwards in the network. The Voting-based backpressure algorithm executes at 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:
$$A_{nmd} = \begin{cases} 1/2, & \text{if } m = B_{nmd} \\ 1 & \text{otherwise.} \end{cases}$$
- Each link is assigned a final weight according to w_{nm} and the parameter A_{mnd}, i.e.,
$$w_{nm}(t) = \max_{d \epsilon D}(wt_{nmd}(t) * A_{nmd}).$$
- The network controller chooses the control action that solves the following optimization:
$$\mu^*(t) = argmax \sum_{n,m} \mu_{nm} w_{nm}(t)$$
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 method. Except from the worst node, also the good node G_{nmd} or nodes to send packets can be retrieved from the above procedure giving an extra bonus to this node for the corresponding queue. Parameter A_{nmd} would be defined in the following way:

$$A_{nmd} = \begin{cases} 2, & \text{if } m = G_{nmd} \\ 1/2, & \text{if } m = B_{nmd} \\ 1 & \text{otherwise.} \end{cases}$$

As shown in the experiments (cf. subsection VI-B), this policy 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 method 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 destination. 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 method is presented in a later section, to support the above claims we tested the method in a ring network consisting of twelve nodes. The performance of $\mathcal{V}o\mathcal{BP}$ is very close (for low loads) to that of the shortest path backpressure (see Figure 2). This performance is achieved without the need of any extra queues or other info, like distances among nodes, except from the last visited node that every packet holds.

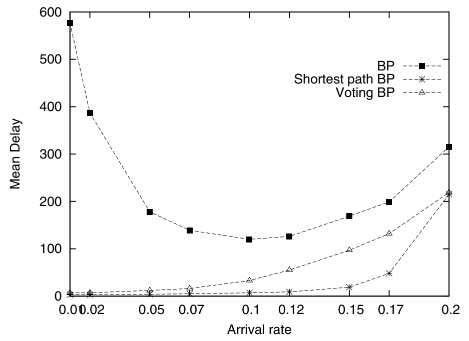


Figure 2. Performance of $\mathcal{V}o\mathcal{BP}$ in a 12-node ring net.

V. Layered backpressure

This section describes a delay-efficient backpressure algorithm based on the creation of *layers*[2] in the network.

[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.

The main idea is to split the network into layers according to the connectivity among them, which also (usually) implies geographic proximity, as well. These layers divide the initial graph into 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, that attract the packets destined for them and then "disallow" the packets to leave the layer. Apparently, if we have only one layer then the policy reduces to the original backpressure algorithm.

For the implementation of the $\mathcal{L}ay\mathcal{BP}$ 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 [19], because $\mathcal{L}ay\mathcal{BP}$ 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 in order to improve the mean end-to-end delay. Apparently, the concept of "correctness" is algorithmic, and we only need to set a partitioning criterion; 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 V-C presents the Layered Backpressure packet scheduling algorithm.

A. Random-walk-based layering

In this section we present a random walk based clustering (RWC) technique for creating the necessary layers for the routing protocol. RWC runs in the initiation phase of the network. The method is centralized and the number of layers Nc to be created are predefined. Actually, we only need an estimation of the number a layers present in the network as long as this number is close to an "optimal" number that would be discovered by an exhaustive cost-optimal algorithm. Techniques for estimating this number can be found at [10]. The layers created by this method 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. In 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 1.

Definition 1 (Neighbourhood of node i): A node j belongs to the neighbourhood 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 method operates as follows:

1) Find the layer $Poslevel$ having the maximum number of members of the current neighbourhood. 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 neighbourhood 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.

B. 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 protocol, that is based on the RWC algorithm, needs to forward packets according to the layer's labels. In order for $\mathcal{L}ay\mathcal{BP}$ to work efficiently with layers, we should assign 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; though 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, but this investigation is beyond the scope of the present work.

C. The Layered BackPressure protocol

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 queue is denoted as $Q_n^d[t]$. For every queue $Q_n^d[t]$ the node computes the parameter $Dlevel_n^d$ which represents the absolute difference between current and destination node's layer number: $Dlevel_n^d = |Layer(n) - Layer(d)|$. At each time slot t, the network conrtoler observes the queue backlog matrix $Q(t) = (Q_n^d(t))$ and performs the following actions for routing:

Layered BackPressure at 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:

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

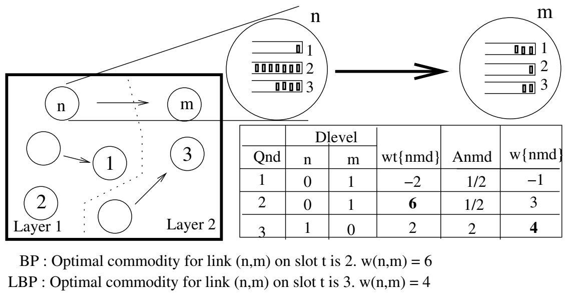
- Each link is assigned a final weight according to w_{nm} and parameter A_{mnd} $w_{nm}(t) = \max_{d \epsilon D}(wt_{nmd}(t) * A_{nmd})$.
- The network conrtoler chooses the control action that solves the following optimization: $\mu^*(t) =$ argmax $\sum_{n,m} \mu_{nm} w_{nm}(t)$ subject to the interference model mentioned above where adjacent links are not allowed to be active simultaneously.

A simple example is illustrated in Figure 3. The network consists of 7 nodes where all nodes are sendets and only nodes with id's $1, 2, 3$ are destination nodes. In Figure 3 we observe the status of the queue backlogs at a specific time slot t. The network is divided in two layers. Layer one includes nodes in the left side of the dotted line. Running the initial backpressure algorithm the commodity that achieves maximum differential backlog for link (n, m) is 2 while for the proposed layer backpressure is 3. If we assume that link (n, m) is scheduled at 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 longer path in order to reach its destination (node 2). With the layer backpressure policy at the same time moment the 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 m, which according to the proposed layer backpressure policy will prevent the packet from returning to node n.

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

[Sketch of proof]. The proof[3] is similar to that presented in [14]. Since, parameter A_{nmd} is bounded (for $\mathcal{LayBP}$ it is a constant, for $\mathcal{VoBP}$ is time-varying but bounded), both policies are throughput optimal.

[3]A detailed proof will be developed in a subsequent work.



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

VI. Evaluation

For the evaluation of the proposed scheduling policies we developed a custom simulator, modeling topology features, packet scheduling, link activation, etc. We considered as competitors all the state-of-the-art delay-aware backpressure policies that have been published so far in the literature, with the exception of the $\mathcal{SQ}-\mathcal{BP}$ policies which was shown in subsection I-A that it provides no delay performance benefits. We simulated 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.

A. Experimental setting

Methods compared. As competing methods we examined all the currently proposed delay-aware backpressure policies along with the original backpressure method as the baseline algorithm. We emphasize here that the shortest path backpressure [18] is the theoretically optimal w.r.t. delay method. This method performs the best in all cases and acts as the lower bound, but it is vulnerable to topology changes (cf. Figure 9). For the methods 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 method when source and destination are in the same cluster, the traffic controler can directly route them without relaying to any gateway, even if source is a gateway of $cluster(d)$ itself.

Performance measures. As performance measures we used the average end-to-end delay and the throughput.

Network topologies. As network topologies, we considered those illustrated in Figure 4. The first network is a

$4 \times 4 = 16$ nodes grid network; it is very common in the literature [5], [18] and it comprises a very "comfortable" case for $\mathcal{SP}-\mathcal{BP}$ since there are more than one shortest paths between each pair of nodes. Also, the other two topologies (Figure 4b and 4c) were borrowed from [2] with 22 and 30 nodes, respectively. For the methods 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".

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 situations where there is a "preference" toward some specific destination node. Since in the conducted experiments with Zipfian-selected destination, the obtained results showed no alteration in the relative performance of the algorithms, all the results we present concern the uniformly-at-random-selected destination. Different values of λ are chosen according to the number of nodes of the network. This means that the same λ might represent low load for a small network, and high traffic for a much larger network.

B. Experimental results

The goal of the experimental evaluation is, apart from testing the delay and throughput performance of all the competing methods, to validate all the claims reported in subsection I-A out 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 methods so as to tune them for the rest of the experiments.

Tuning of $\mathcal{V}o\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$. 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 evaluated the performance of the methods for various A (Figure 5a). The obtained results showed 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 I-A, we stated that the performance of $\mathcal{CB}-\mathcal{BP}$ deteriorates with increasing number of clusters; indeed this is the pattern of its behavior, as shown in Figure 5c (tested on the grid topology), where beyond 5 clusters, the performance of $\mathcal{CB}-\mathcal{BP}$ is alsmost 25 times worse than that of $\mathcal{L}ay\mathcal{BP}$. In other topologies, this number might be even smaller. That behavior is predicted by the description in [19], where the increased number of clusters implies increased number of gateways and thus more delays. Combining this observation with the fact that all ad hoc clustering protocols produce a number of clusters which is linear or logarithmic in the number of network nodes, we deduce that $\mathcal{CB}-\mathcal{BP}$ can hardly be used with automated algorithmic network clustering algorithms. Actually, $\mathcal{CB}-\mathcal{BP}$ requires a very small constant number of clusters.

Delay and throughput performance. The plots in Figure 6 depict the delay performance of the competing methods. 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).

It came to our surprise that the $\mathcal{V}o\mathcal{BP}$ method performs better than all the methods (expect 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 has to do with the average node degree of the network. For instance, the delay of $\mathcal{V}o\mathcal{BP}$ is significantly higher (see Figure 6c) for the 30-node network (presented in Figure 4c) which is quite dense.

The plots in Figure 7 depict the throughput performance of the competing methods. The obvious observation concerns the optimality of all methods, including optimality of $\mathcal{V}o\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$ according to Theorem 1.

Impact of topology change. To validate the claims about the impact of topology on the shortest path backpressure method, we performed an experiment on a small sample network of 10 nodes (see Figure 8). 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 9 show that $\mathcal{SP}-\mathcal{BP}$ practically breaks down – as expected – whereas $\mathcal{L}ay\mathcal{BP}$ maintains its strength. We do expect that dramatic changes will impact also $\mathcal{L}ay\mathcal{BP}$, but $\mathcal{L}ay\mathcal{BP}$ is more robust to topology changes since it does not require exact knowledge of shortest paths among nodes, but rather "shortest paths" among clusters. $\mathcal{L}ay\mathcal{BP}$ actually represents "shortest paths" at a coarse granularity.

VII. Relevant work

A lot of packet scheduling protocols have been proposed in the relevant literature. Algorithms for packet scheduling/routing can be classified as data-centric, hierarchical, location-based and those based on network flow or quality

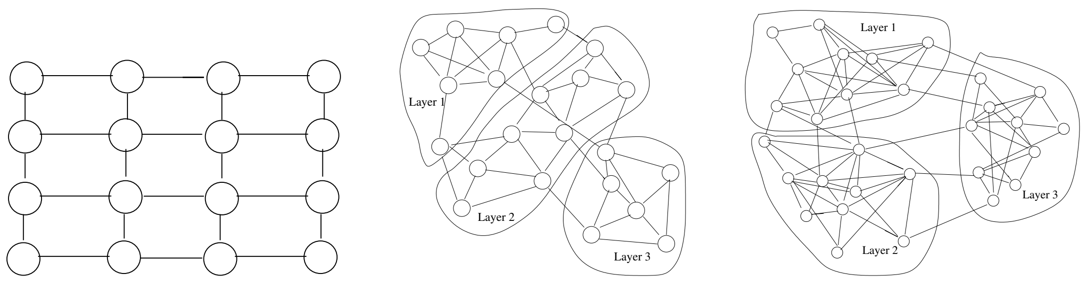


Figure 4. Network topologies for the experiments presented (16, 22 and 30 nodes).

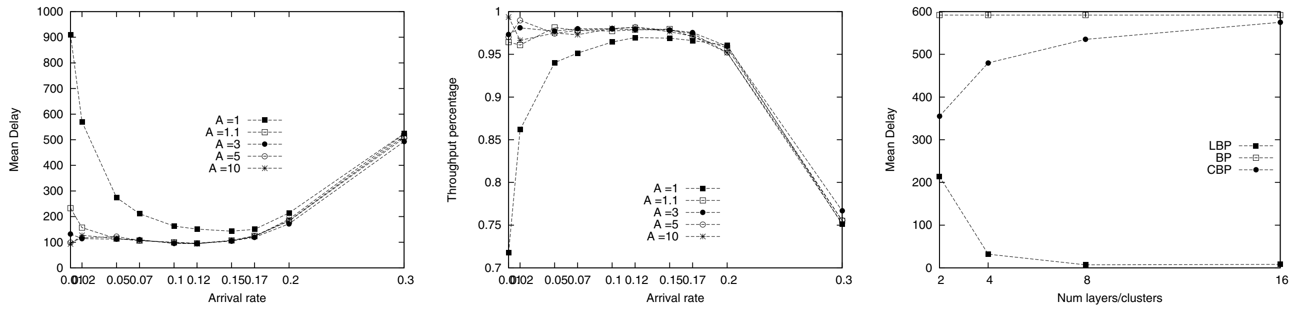


Figure 5. Tuning $\mathcal{V}o\mathcal{BP}$ and $\mathcal{L}ay\mathcal{BP}$ (first two plots) and impact of the number of clusters (rightmost plot).

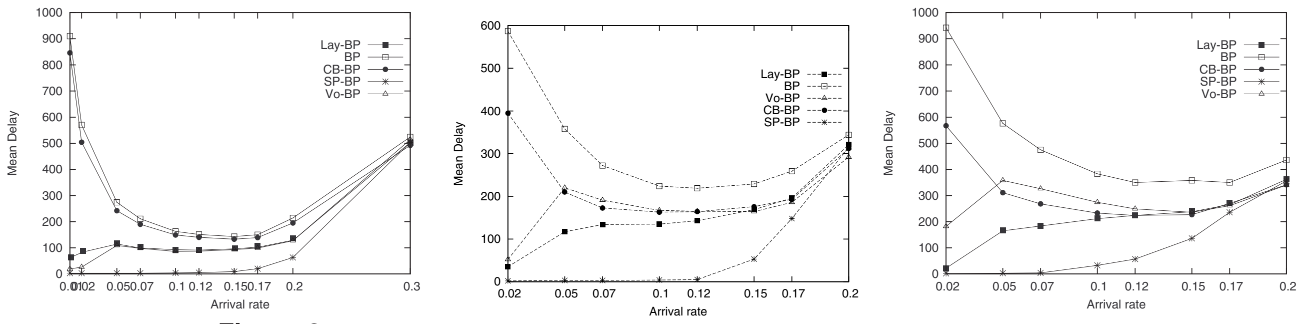


Figure 6. Delay performance for the three network topologies (16, 22 and 30 nodes).

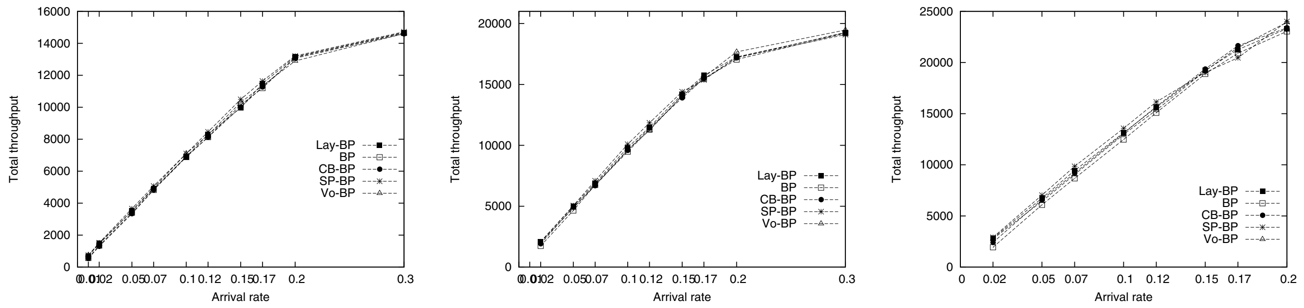


Figure 7. Throughput performance for the three network topologies (16, 22 and 30 nodes).

of service awareness. The interested reader should refer to [1] for detailed categorization. In data-centric protocols [9], aggregation of data during the relaying of data is utilized. Hierarchical routing protocols [11] assign different roles to nodes. The nodes are divided into clusters and cluster-heads are assigned with the role of data aggregation

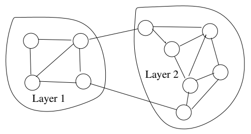


Figure 8. Topology of a small sample 10-node network.

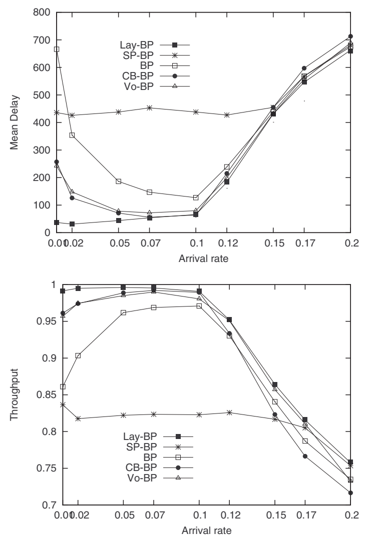


Figure 9. Impact of topology changes on the delay and throughput performance of $\mathcal{SP}-\mathcal{BP}$ and $\mathcal{LayBP}$.

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 [16]. System stability and throughput maximization have been exclusively studied [13], [15]. Related work on delay-efficient and throughput-optimal backpressure algorithms is developed in [5], [7], [18].

VIII. Conclusions and future work

This paper considered the problem of packet scheduling in wireless networks using backpressure-type algorithms. The purpose of the work is to address the delay-efficiency problems of the backpressure-type algorithms. In this context, the paper described two algorithms; the first method, namely $\mathcal{VoBP}$, alleviates the ping-pong effect in packet scheduling, and the second algorithm investigated the issue of creating layers of nodes and performing backpressure scheduling using the IDs of the layers so as to "push" the packets toward the destination layer. The experimental evaluation of the proposed methods against the state-of-the-art methods attested the superiority of the latter method. Our future work involves the combination of *voting* and *layering*, i.e., application of voting inside layers, as well as the application of ideas in [8] to develop delay-efficient, but throughput suboptimal policies.

Acknowledgements

The authors wish to thank Dr. Kostas Tsoukatos for advising on the throughput optimality of the proposed methods, and for pointing out the relevance of the proof with that in [14].

References

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

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

[3] E. Athanasopoulou, L. Bui, T. Ji, R. Srikant, and A. Stoylar. Backpressure-based packet-by-packet adaptive routing in communication networks, 2010. Available at: http://arxiv.org/abs/1005.4984.

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

[5] L. Bui, R. Srikant, and A. Stolyar. Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. In *Proc. INFOCOM Mini*, 2009.

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

[7] G. R. Gupta and N. Shroff. Delay analysis for multi-hop wireless networks. In *Proc. INFOCOM*, 2009.

[8] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari. LIFO-Backpressure achieves near optimal utility-delay tradeoff, 2010. Available at: http://arxiv.org/abs/1005.4984.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proc. MOBICOM*, 2000.

[10] D. Katsaros, N. Dimokas, and L. Tassiulas. Social network analysis concepts in the design of wireless ad hoc network protocols. *IEEE Network magazine*, 24(6):23–29, 2010.

[11] A. Manjeshwar and D. Agrawal. TEEN: A protocol for enhanced efficiency in wireless sensor networks. In *Proc. IPDPS*, 2001.

[12] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. In *Proc. IPSN*, 2010.

[13] M. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *Proc. INFOCOM*, pages 1723–1734, 2005.

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

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

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

[17] Y. Xu, J. Heidemann, and D. Estrin. Geography-informed energy conservation for ad hoc routing. In *Proc. MOBICOM*, 2001.

[18] L. Ying, S. Shakkotai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. In *Proc. INFOCOM*, 2009.

[19] L. Ying, R. Srikant, and D. F. Towsley. Cluster-based backpressure routing algorithm. In *Proc. INFOCOM*, pages 484–492, 2008.