Chapter 30
Layered Backpressure Scheduling for Delay-Aware Routing in Ad Hoc Networks

Dimitrios Katsaros

30.1 Introduction

Packet transmission scheduling in wireless ad hoc (multi-hop) networks is a fundamental issue since it is directly related to the achievement of a prescribed quality of service (QoS). QoS is usually measured in terms of the average packet delay. Additionally, 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 [8], and it is known as the *backpressure* (***BP***) packet scheduling algorithm. Nevertheless, this algorithm and many of its variations suffer from delay problems. Certain features of the backpressure algorithm suggest that long delays will be common. Firstly, backpressure routing uses queue buildup at nodes to create a "gradient" within the network that guides routing. However, this may come at the cost of increased queuing delay. Secondly, backpressure routing tends to explore all paths in a network, including paths with loops and "dead-end" paths that cannot lead to the desired destination. Hence, packets generally may not take the shortest path to their destination, thereby leading to additional delay.

30.2 Motivation

The performance of backpressure deteriorates in conditions of low, and even of moderate, load in the network, since the packets "circulate" in the network increasing the packet delay. To circumvent the delay problems of backpressure, the *shadow queue*

D. Katsaros (✉)
University of Thessaly, Volos, Greece
e-mail: dkatsar@inf.uth.gr

J.H. van Schuppen and T. Villa (eds.), *Coordination Control of Distributed Systems*,
Lecture Notes in Control and Information Sciences 456,
DOI 10.1007/978-3-319-10407-2_30

algorithm [3] (***SQ-BP***) 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. 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 the *cluster-based backpressure* , which is based on node grouping in order to reduce the number of these queues [10] (***CB-BP***), and thus, the computational overhead, and as a side effect, reduces the delay. To alleviate the delay problems of backpressure, scheduling based on the combination of backpressure and shortest paths has been proposed [9], i.e., the *shortest paths backpressure* (***SP-BP***) policy. Finally, some ideas [6] 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 a radically different problem.

30.3 Examples

We shall first present the basic backpressure mechanism with a small example assuming slotted time. Let us suppose that we have a 4-node ad hoc network with two flows from node A to D depicted in Fig. 30.1. Each node maintains a separate queue for each flow. For each queue, the number of backlogged packets is shown. Assume that we have two link sets, $\{(A, B), (C, D)\}$ and $\{(A, C), (B, D)\}$. The links in each set do not interfere and can transmit in the same time slot. The scheduler executes the following three steps at each slot. First, for each link, it finds the flow with the maximum differential queue backlog. For example, for link (A, B), the blue flow has a difference of 2 packets and the black flow has a difference of 7 packets. The maximum value is then assigned as the weight of the link (see Fig. 30.1). Second, the controller selects the set of noninterfering links with the maximum sum of weights for transmission. This requires to compute the sum of link weights for each possible set. In the example, set $\{(A, B), (C, D)\}$ sums to $7 + 4 = 11$ and set $\{(A, C), (B, D)\}$ sums to $6 + 6 = 12$. The scheduler then selects the set with the maximum sum of weights, i.e., $\{(A, C), (B, D)\}$, to transmit at this slot. Finally, packets from the selected flows are transmitted on the selected links, i.e., blue flow on link (A, C) and black flow on link (B, D).

A simulation-based evaluation of the aforementioned backpressure variations revealed their shortcomings. The shadow queues method forces the packets stay in queues longer, which leads to higher delays (see Fig. 30.2). The ***CB-BP*** policy requires maintaining one queue per *gateway* at each relay node which leads 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 (see Fig. 30.3). The ***SP-BP*** method assumes the precomputation of all pairwise-node distances. Apart from this computational-type problem, frequent topology changes would lead the method to break down, since many shortest paths would not exist anymore (see Fig. 30.4).

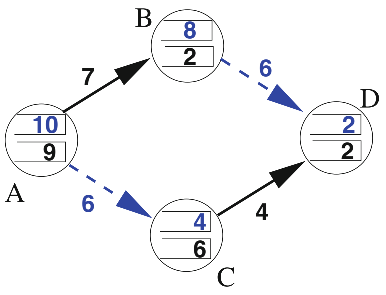

Fig. 30.1 Backpressure scheduling in a network with two flows, black and blue from A to D. Links in sets (A, B), (C, D) (*continuous*) and (A, C), (B, D) (*dashed*) can be scheduled in the same slot

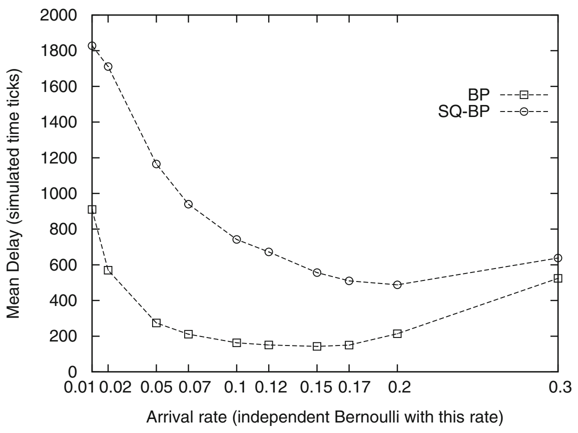

Fig. 30.2 Performance of the ***SQ-BP*** policy in a 4×4 grid network ($M = 2$)

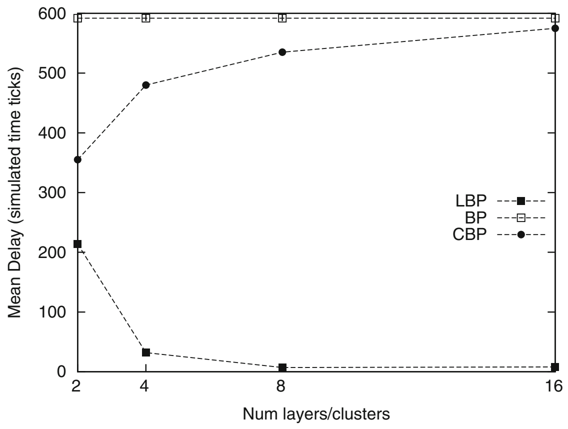

Fig. 30.3 The impact of the number of clusters

Despite the aforementioned efforts, the delay problem of backpressure has not been adequately solved. The challenge is to take a *holistic* approach in designing an efficient delay-aware backpressure policy, that is also practical—one that will have low computational overhead and that will be robust to topology changes.

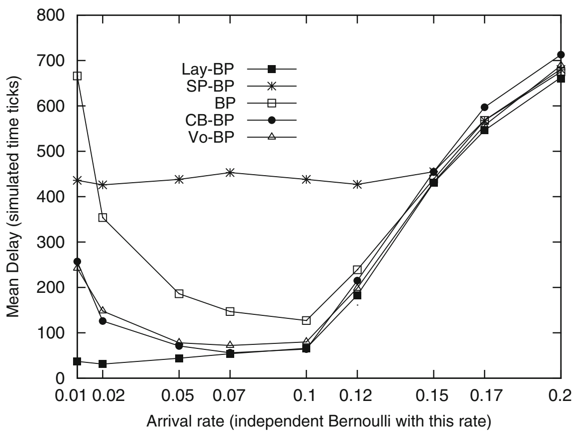

Fig. 30.4 Impact of topology changes on the delay performance of ***SP-BP***. The network consists of two clusters connected to each other with two links only. We inactivate one of the two links without recalculating the shortest paths among all node pairs

30.4 Theory and Concepts

Formally, 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 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

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

  Let $d^*_{mn}[t]$ be the destination 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:

$$\mu^*(t) = \text{argmax} \sum_{n,m} \mu_{nm} w_{nm}(t),$$

  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 destination $d^*_{mn}[t]$ from node n to node m. The routes

are determined on the basis of differential queue backlog providing adaptivity of the method to congestion.

30.5 Overview of Research Contributions

To provide a holistic solution, we designed the *Layered Backpressure* (***LayBP***), which divides the network into "layers" according to the connectivity of nodes. This method maintains the same number of queues with the original ***BP*** and one order of magnitude less number of queues compared to ***SP-BP***. It does not require the existence of *gateways* and aggregated queues as does ***CB-BP***. In addition, it can be seen as a "relaxed" version of the ***SP-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 ***LayBP*** is a hybrid among ***CB-BP*** and ***SP-BP***, compromising a little delay for robustness, low computational complexity, and simplicity.

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 $D\text{level}_n^d$ which represents the absolute difference between current and destination node's layer numbers: $D\text{level}_n^d = |\text{Layer}(n) - \text{Layer}(d)|$. At 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 at time slot t:

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

$$A_{nmd} = \begin{cases} 2, & \text{if } D\text{level}_n^d > D\text{level}_m^d \\ 1/2, & \text{if } D\text{level}_n^d < D\text{level}_m^d \\ 1 & \text{otherwise.} \end{cases}$$

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

The ***LayBP*** algorithm is throughput optimal, in the sense that it can stabilize queues for any stabilizable arrival rate.

In order to prove that the ***LayBP*** is throughput optimal, the Lyapunov stability criterion is used. The idea behind the Lyapunov drift technique is to define a nonnegative function, called the Lyapunov function, which represents the aggregate congestion of all queues (Q_n^d) of the network. The drift of the function at 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 strategies, we use

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

as the Lyapunov function.

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

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

This equation can be rewritten in the following form:

$$w_{nm}(t) = \max_{d\varepsilon D}\left(A_{nmd} * Q_n^d - A_{nmd} * Q_m^d\right).$$

which is equivalent to

$$w_{nm}(t) = \max_{d\varepsilon D}\left(\theta_n^d * Q_n^d - \theta_m^d * Q_m^d\right),$$

where weights θ_i^d are used to offer priority service.

Queue dynamics at each time slot satisfy

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

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 $A_n^d(t)$ represents the process of exogenous commodity d data arriving at source node n.

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

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

$$\begin{aligned}\sum_{nd} \theta_n^d (Q_n^d(t+1))^2 \le{}& \sum\nolimits_{nd} \theta_n^d (Q_n^d(t))^2 + \sum\nolimits_{nd} \theta_n^d (\sum\nolimits_b \mu_{nb}^d(t))^2 \\ &+ \sum\nolimits_{nd} \theta_n^d (A_n^d(t) + \sum\nolimits_a \mu_{an}^d(t))^2 - 2\sum\nolimits_{nd} \theta_n^d Q_n^d(t) \\ &\times \left(\sum\nolimits_b \mu_{nb}^d(t) - A_n^d(t) - \mu_{an}^d(t)\right).\end{aligned}$$

It is not difficult to show that

$\Delta L(Q) \leq 2BN - 2\sum_{nd} \theta_n^d \varepsilon Q_n^d(t)$,where $B \triangleq \frac{1}{2N} \sum_{n\varepsilon N} \theta_{\max}[(\mu_{\max,n}^{\text{out}})^2 + (A_n^{\max} + \mu_{\max,n}^{in})^2]$.

Using the above, we can rewrite drift inequality as follows:

$\Delta L(Q) \leq 2B'N\theta_{\max} - 2\sum_{nd} \theta_n^d \varepsilon Q_n^d(t)$, where $B' \triangleq \frac{1}{2N^2} \sum_{n\varepsilon N}[(\mu_{\max,n}^{\text{out}})^2 + (A_n^{\max} + \mu_{\max,n}^{in})^2]$.

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 as follows:

$\lim\sup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t} E\{\sum_{n,d} \theta_n^d Q_n^d(\tau)\} \leq \frac{NB'\theta_{\max}}{\varepsilon_{\max}}$, which proves the optimality.

30.5.1 The Enhanced Layered Backpressure Policy

Routing protocols must be dynamic in order to cope with mobility of nodes in modern wireless networks. Widely varying mobility characteristics are expected to have a significant impact on the performance of routing protocols that are based on node grouping (like ***CB-BP***, ***LayBP***) in order to route packets even if links among nodes are updated. In case of grouping-based routing protocols, high mobility of nodes which lead them to change groups degrades the performance of the methods since this "wrong" information is used in the routing procedure. Although ***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 and destination's node layers. It is clear that ***LayBP*** behavior can be affected by "misplaced" nodes. In this case, packets may be forwarded to layers different than the desired, making the method inappropriate.

In order to cope with node mobility, we incorporate in ***LayBP*** algorithm another step in which moving nodes and all the one-hop neighbors recalculate their cluster according to their neighborhood. In the initiation phase, every node has a counter $C0_{nl}$ for every layer ID, indicating how many neighbors belong to it, and a variable Layer_n indicating the layer that node n belongs to. For every time slot t, the following actions are performed:

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

This algorithm can be executed every k time slots according to how fast we want the system to adapt to node mobility. Only moving nodes and their one-hop neighbors update the information on their counters in order to find the appropriate layer. Also, one-hop neighbors of the moving 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 algorithm does not perform reclustering, but only "helps" layers incorporate moving nodes.

30.6 Further Reading

Recent interesting work on the backpressure family of algorithms comprises the study of the trade-off between throughput optimality and delay [1, 4, 5, 7] and attempts to decouple the routing and scheduling components of the algorithm [2].

References

1. Alresaini M, Sathiamoorthy M, Krishnamachari B, Neely MJ (2012) Backpressure with adaptive redundancy (BWAR). In: Proceedings of IEEE INFOCOM, pp 2300–2308
2. Athanasopoulou E, Bui L, Ji T, Srikant R, Stoylar A (2013) Back-pressure-based packet-by-packet adaptive routing in communication networks. IEEE/ACM Trans Networking 21(1):244–257
3. Bui L, Srikant R, Stolyar A (2009) Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing. In: Proceedings of IEEE INFOCOM mini conference.
4. Cui Y, Lau VKN, Wang R, Huang H, Zhang S (2012) A survey on delay-aware resource control for wireless systems—Large deviation theory, stochastic Lyapunov drift, and distributed stochastic learning. IEEE Trans Inf Theory 58(3):1677–1701
5. Huang L, Moeller S, Neely MJ, Krishnamachari B (2013) LIFO-Backpressure achieves near optimal utility-delay tradeoff. IEEE/ACM Trans Networking 21(3):831–844
6. Moeller S, Sridharan A, Krishnamachari B, Gnawali O (2010) Routing without routes: the backpressure collection protocol. In: Proceedings of IPSN
7. Si W, Starobinski D (2013) On the channel-sensitive delay behavior of lifo-backpressure. In: Proceedings of Allerton, pp 715–722
8. Tassiulas L, Ephremides A (1992) Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Trans Autom Control 37(12):1936–1948
9. Ying L, Shakkotai S, Reddy A (2009)On combining shortest path and back-pressure routing over multihop wireless networks.In: Proceedings of IEEE INFOCOM
10. Ying L, Srikant R, Towsley DF (2008) Cluster-based backpressure routing algorithm. In: Proceedings of IEEE INFOCOM, pp 484–492