Minimization of Congestion using Backpressure Routing Algorithm in Wireless Networks

Author[1] : A.PRIYADHARSINI
M.E.,Computer Science and Engineering
Arunai College of Engineering
Tiruvannamalai-606603
Email id: sundharampriya@gmail.com

Author[2]:Mr.S.MOHANARANGAN
M.Tech.,(Ph.D).,Computer Science and Engineering
Arunai College of Engineering Tiruvannamalai-606603
Email id:sramrangan@gmail.com

Abstract—Back-pressure-type algorithms have recently received much attention for jointly routing and scheduling over multi-hop wireless networks. However, this approach has a significant weakness in routing because the traditional back pressure algorithm explores and exploits all feasible paths between each source and destination. While this extensive exploration is essential in order to maintain stability when the network is heavily loaded, under light or moderate loads, packets may be sent over unnecessarily long routes, and the algorithm could be very inefficient in terms of end-to-end delay and routing convergence times. This paper proposes a new routing/scheduling back-pressure algorithm that not only guarantees network stability (throughput optimality), but also adaptively selects a set of optimal routes based on *shortest-path information* in order to minimize average path lengths between each source and destination pair. Our results indicate that under the traditional back-pressure algorithm, the end-to-end packet delay first decreases and then increases as a function of the network load (arrival rate). This surprising low-load behavior is explained due to the fact that the traditional back-pressure algorithm exploits all paths (including very long ones) even when the traffic load is light. On the other-hand, the proposed algorithm adaptively selects a set of routes according to the traffic load so that long paths are used only when necessary, thus resulting in much smaller end-to-end packet delays as compared to the traditional back-pressure algorithm.

Index Terms—Back-pressure routing,shortest path routing, throughput-optimal.

I. INTRODUCTION

Due to the scarcity of wireless bandwidth resources, it is important to efficiently utilize resources to support high throughput, high-quality communications over multi-hop wireless networks. In this context, good routing and scheduling algorithms are needed to dynamically allocate wireless resources to maximize the network throughput

region. To address this, throughput-optimal1 routing and scheduling, first developed in the seminal work of [2], has for a comprehensive survey. While these algorithms be extensively studied [3]–[14].We refer to [15] and [16] maximize the network throughput region, additional issues need to be considered for practical deployment.

With the significant increase of real-time traffic, end-to-end delay becomes very important in network algorithm design. The traditional back-pressure algorithm stabilizes the network by exploiting all possible paths between source–destination pairs (thus load balancing over the entire network). While this might be needed in a heavily loaded network, this seems unnecessary in a light or moderate load regime. Exploring all paths is in fact detrimental—it leads to packets traversing excessively long paths between sources and destinations, leading to large end-to-end packet delays.

This paper proposes a new routing/scheduling back-pressure algorithm that minimizes the path lengths between sources and destinations while simultaneously being overall throughput-optimal. The proposed algorithm results in much smaller end-to-end packet delay as compared to the traditional back-pressure algorithm.

We define a flow using its source and destination. Let f denote a flow in network, denote the set of all flows in the network, F and $A_f[t]$ denote the number of packets generated by flow at time . We first consider the case where each flow associates with a hop constraint H_f . The routing and scheduling algorithm needs to guarantee that the packets from flow f are delivered in no more than H_f hops. Note that this hop constraint is closely related to the end-to-end propagation delay. For this problem, we propose a shortest-path-aided back-pressure algorithm that exploits the shortest-path information to guarantee the hop constraint and is throughput-optimal; i.e., if there exists a routing/scheduling algorithm that can support the traffic with the given hop constraints, then the shortest-path-aided back-pressure can support the traffic as well.

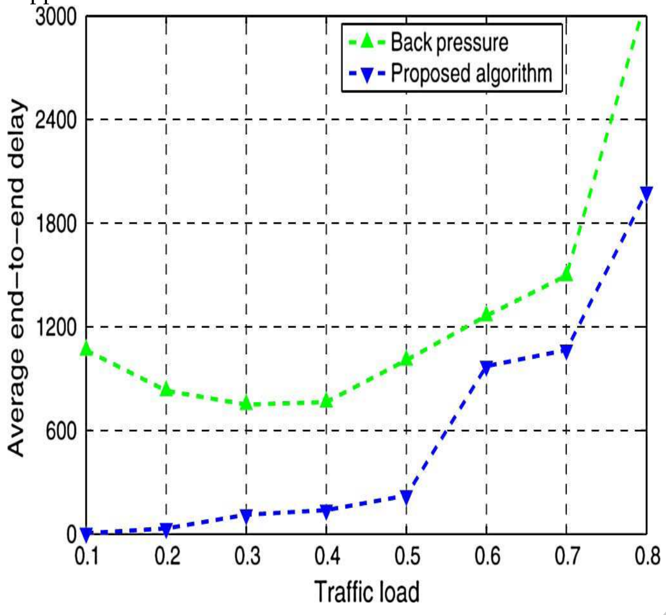

Fig.1. Back pressure via our joint traffic splitting and shortest-path-aided back pressure

Fig.1 illustrates the average end-to-end delays under the back-pressure algorithm and the proposed algorithm under different traffic loads. The network used in the simulation is a grid-like network with 64 nodes and 8 data flows .We have two observations.

1) Under the back-pressure algorithm, surprisingly, the delay first decreases and then increases as the traffic load increases. The second part is easy to understand: The queues build up when the traffic load increases, which increases the queuing delays. The first part is because the back-pressure algorithm uses all paths even when the traffic load is light. For example, in a light traffic regime, using shortest paths is sufficient to support the traffic flows. However, under the back-pressure algorithm, long paths and paths with loops are also used. Furthermore, the lighter the traffic load, the more loops are involved in the route. Hence, the end-to-end delay is large.

2) In the proposed algorithm, the set of routes used is intelligently selected according to the traffic load so that long paths are used only when necessary .We can see that under the proposed algorithm, not only is the delay significantly reduced, but also the delay monotonically increases with the traffic load.

We would like to emphasize that under the proposed algorithm, the delay improvement is achieved without losing the throughput-optimality. The proposed algorithm is still throughput-optimal, but yields much smaller end-to-end delays as compared to the traditional back-pressure algorithm.

II. MATHEMATICAL MODELS

We describe in this section mathematical models, which were built to represent a task allocation framework, parallel applications with security constraints.

A. Network Model

Consider a network represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where N is the set of nodes and L is the set of directed links. We assume that $|\mathcal{N}| = N$ and $|\mathcal{L}| = L$. Denote by (m ,n) the link from node m to node n . Furthermore, let $\mu = \{\mu_{(m,n)}\}$ denote a link-rate vector such that $\mu_{(m,n)}$ is the transmission rate over link (m ,n). A link-rate vector µ is said to be admissible if the link-rates specified by µ can be achieved simultaneously. Define Γ to be the set of all admissible link-rate vectors. It is easy to see that Γ depends on the choice of interference model and might not be a convex set. Furthermore, Γ is time-varying if link-rates are time-varying. To simplify our notations ,we assume time-invariant link-rates in this paper. However ,our results can be extended to time-varying link-rates in a straightforward manner. Furthermore, we assume that there exist $\mu_{\min}$ and $\mu_{\max}$ such that $\mu_{\min} \leq \mu_{(m,n)} \leq \mu_{\max}$ for all $(m,n) \in \mathcal{L}$ and all admissible µ .

Next, we define a link vector µ to be obtainable if $\mu \in \mathcal{CH}(\Gamma)$, where $\mathcal{CH}(\Gamma)$ denotes the convex hull of Γ.Note that an admissible rate-vector is a set of rates at which the links can transmit simultaneously, while an obtainable rate-vector is a set of rates that can be achieved including using time sharing. As a simple example, consider a network with two nodes {1, 2} and two links {(1, 2),(2, 1)}. Assume the link capacity is one packet per time slot for both links, and half-duplex constraint so that only one link can transmit at one time. Then, $\mu = \{0.5, 0.5\}$ is not an admissible rate-vector since two links cannot transmit at the same time. However, it is obtainable by time sharing.

B. Traffic Model

For network traffic, we let f denote a flow, s(f) denote the source of the flow, and d(f) the destination of the flow. We use to F denote the set of all flows in the network. Assume that time is discredited, and let $A_f[t](f \in \mathcal{F})$ denote the number of packets injected by flow at time . In this paper, we assume $A_f[t]$ is random and independent and identically distributed across time slots, $A_f[t] = 0$.

III. THROUGHPUT-OPTIMAL ROUTING/SCHEDULING WITH HOP CONSTRAINTS

In this section, we consider the case where each flow is associated with a hop constraint H_f . Packets of flow f need to be delivered within H_f hops. We propose a shortest-path-aided back-pressure algorithm, which is throughput-optimal under hop-constraints. The algorithm is also a building block for the algorithm to be proposed in Section V, which seamlessly integrates the back-pressure and the shortest-path routing. Next, we characterize the network throughput region under hop constraints.

A. Network Throughput Region Under Hop Constraints

We denote by 1_Φ the indicator function with condition Φ,i.e., $1_\Phi = 1$ if condition holds, and $1_\Phi = 0$ otherwise. Given traffic $\mathbf{A} = \{A_f\}_{f\in\mathcal{F}}$ and hop constraint $\mathbf{H} = \{H_f\}_{f\in\mathcal{F}}$,we define $\Lambda_\mathcal{G}$ by saying that $(\mathbf{A},\mathbf{H}) \in \Lambda_\mathcal{G}$ if there exists such that the following conditions hold.

(i) For any three-tuple $\{n, d, h\}$ such that $n \neq d$ and $N-1 \geq h > 0$, we have

$$A_f 1_{\substack{s(f)=n,d(f)=d\\ H_f=h}} + \sum_{m:(m,n)\in\mathcal{L}} \hat{\mu}^{\{n,d,h\}}_{\{m,d,h+1\}} = \sum_{i:(n,i)\in\mathcal{L}} \hat{\mu}^{\{i,d,h-1\}}_{\{n,d,h\}}. \tag{1}$$

(ii) If $h < H^{\min}_{n\to d}$, then

$$\hat{\mu}^{\{n,d,h\}}_{\{m,d,h+1\}} = 0 \tag{2}$$

where $H^{\min}_{n\to d}$ is the minimum number of hops from node n to node d.

(iii)

$$\{\hat{\mu}_{(m,n)}\}_{(m,n)\in\mathcal{L}} \in \mathcal{CH}(\Gamma) \tag{3}$$

where

$$\hat{\mu}_{(m,n)} = \sum_{\substack{d:d\in\mathcal{D}\\ h:N-1\geq h>0}} \hat{\mu}^{\{n,d,h\}}_{\{m,d,h+1\}}$$

and D is the set of all destinations.

a) Condition (i) is the flow-conservation constraint, which states that the number of incoming packets to node n with hop constraint h is equal to the number of outgoing packets from node n with hop constraint h-1. Note that the hop constraint reduces by one after a packet is sent out by node n because it takes one hop to transmit the packet from node n to

one of its neighbour .We only consider hop constraints up to N-1 because the longest loop-free path has no more than N-1 hops, and considering only loop-free routes does not change the network throughput region.
b) Condition (ii) states that a packet should not be transmitted from node m to node n if node n cannot deliver the packet within the required number of hops.
c) Condition (iii) is the capacity constraint, which states that the rate-vector $\hat{\boldsymbol{\mu}}$ should be obtainable.

B. Queue Management

We introduce our queue management scheme .Recall $H^{\min}_{m\to d}$ is the minimum number of hops from node m to node d (or the length of the shortest path from node to node).Note that $H^{\min}_{m\to d}$ can be computed in a distributed fashion using algorithms such as the Bellman–Ford algorithm. Thus, we assume that node m knows $H^{\min}_{m\to d}$ for all destinations $d \in \mathcal{D}$, and $H^{\min}_{n\to d}$ for n such that $(m, n) \in \mathcal{L}$.
.

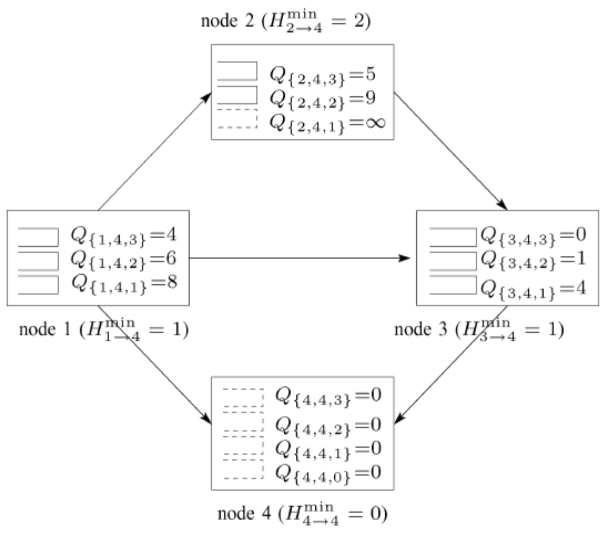

Fig.2.Illustration of queue management and computation of back pressure.

We assume node m maintains a separate queue, named queue {m ,d ,h}, for those packets required to be delivered to node d within h hops. For destination d , node m maintains queues for $h = H_{m\to d}^{\min}, \ldots, N-1$, where N-1 is a universal upper bound on the number of hops along loop-free paths. As an example, consider the directed network shown in Fig. 2, and assume that D={4} (i.e., there is only one destination).Each non destination node maintains up to three queues (because for this topology, there are no loop-free paths longer than three hops). Node 1 has queues corresponding to h=1,2,3, respectively. Node 2 does not have a direct path to node 4, hence it maintains only two queues corresponding to h=2,3. Node 3 maintains three separate queues corresponding to h=1, 2, 3, in spite of the observation that there is only one feasible route from node 3 to node 4.We maintain these additional queues because the global network topology is not known by individual nodes. Finally, all queues at the destination for packets meant to itself are set to zero .In Fig. 2, queues into which packets potentially arrive are marked in solid lines, and the "virtual"queues .

C. Queue Dynamics

Let $Q_{\{n,d,h\}}[t]$ denote the queue length at time slot t , and $\mu_{\{m,d,k\}}^{\{n,d,h\}}[t]$ denote the service rate allocated to transmit packets from queue {m ,d, k} to queue{n ,d ,h} over link (m, n) at time t . Since the packets in queue {m ,d ,k} need to be delivered within k hops, they can be only deposited to queues {n ,d, h}. For example, packets from queue {2, 4, 3} can be transferred to queue {3, 4, 2} or queue {3, 4, 1}. Thus, we impose the following constraint on routing: The packets in queue {m ,d, k} can be only transferred to queues {n, d ,h} for $h < k - 1$, i.e., $\mu_{\{m,d,k\}}^{\{n,d,h\}}[t] = 0 \; for \; all \; h \geq k.$

The dynamics of queue{ n ,d. h} (h≠ d) is as follows:

$$Q_{\{n,d,h\}}[t+1] = Q_{\{n,d,h\}}[t] + A_f[t]1_{s(f)=n,d(f)=d,H_f=h} + \sum_{\substack{k:k-1\geq h \\ m:(m,n)\in\mathcal{L}}} \nu_{\{m,d,k\}}^{\{n,d,h\}}[t] - \sum_{\substack{l:h-1\geq l \\ i:(n,i)\in\mathcal{L}}} \nu_{\{n,d,h\}}^{\{i,d,l\}}[t]$$

Where $\nu_{\{n,d,h\}}^{\{i,d,l\}}[t]$ is the actual number of packets transferred from queue {n ,d ,h} to queue {i ,d, l} and is smaller than $\mu_{\{n,d,h\}}^{\{i,d,l\}}[t]$ when there are not enough packets in queue {n ,d ,h}. Define $u_{\{n,d,h\}}^{\{i,d,l\}}[t]$ to be the unused service. We have

$$\nu_{\{n,d,h\}}^{\{i,d,l\}}[t] = \mu_{\{n,d,h\}}^{\{i,d,l\}}[t] - u_{\{n,d,h\}}^{\{i,d,l\}}[t].$$

We also define $Q_{\{n,n,h\}} = 0$ for all h, i.e., packets delivered are removed from the network immediately.

D. Shortest-Path-Aided Back-Pressure Algorithm

Recall that we have per-hop queues for each destination, which is different from the back-pressure algorithm in [2]. Thus, we first define the back pressure of link (m ,n) under our queue management scheme. We define $P_{\{m,d,k\}}^{\{n,d,h\}}[t]$, the back pressure between queue {m, d, k}and queue {n, d, h} over link (m ,n) , as follows:

- $P_{\{m,d,k\}}^{\{n,a,n\}}[t] = Q_{\{m,d,k\}}[t] - Q_{\{n,d,h\}}[t]$ if $h \leq k-1$ and $h \geq H_{n\to d}^{\min}$;
- $P_{\{m,d,k\}}^{\{n,d,h\}}[t] = -\infty$ otherwise (note that queue $\{n,d,h\}$ does not exist if $H_{n\to d}^{\min} > h$).

The back pressure of link (m ,n) is defined to be

$$P_{(m,n)}[t] = \max\left\{\max_{d\in\mathcal{D},k,h} P_{\{m,d,k\}}^{\{n,d,h\}}[t], 0\right\}$$

Considering the example shown in Fig. 1, it can be verified that $P_{(1,2)} = 0$, $P_{(1,3)} = Q_{\{1,4,3\}} - Q_{\{3,4,2\}} = 3$, $P_{(1,4)} = Q_{\{1,4,1\}} - Q_{\{4,4,0\}} = 8$, $P_{(2,3)} = Q_{\{2,4,2\}} - Q_{\{3,4,1\}} = 5$, and $P_{(3,4)} = Q_{\{3,4,1\}} - Q_{\{4,4,0\}} = 4$.

Shortest-Path-Aided Back-Pressure Algorithm2

Consider time slot .

Step 0: The packets injected by flow f are deposited into queue $\{s(f), d(f), H_f\}$ maintained at node s(f) .

Step 1: The network first computes $\mu^*[t]$ that solves the following optimization problem:

$$\mu^*[t] = \arg\max_{\mu \in \Gamma} \sum_{(m,n)\in \mathcal{L}} \mu_{(m,n)} P_{(m,n)}[t] \qquad (4)$$

In this algorithm, we allow the packets in queue {m, d ,k} to be transferred to queues {n, d, h} for any h such that h≤ k-1, which is more general than the algorithm proposed in [1], where the packets in queue {m, d, k}can be transmitted only to queue {n , d, k-1] . Where μ is an admissible link-rate vector and $\mu_{(m,n)}$ is the rate over link (m ,n) .

Step 2: Consider link(m, n) . If $\mu^*_{(m,n)}[t] > 0$ and $P_{(m,n)} > 0$, node m selects a pair of queues, say {m, d, k} and {n, d, h}, such that

$$Q_{\{m,d,k\}}[t] - Q_{\{n,d,h\}}[t] = P_{(m,n)}[t]$$

and transfers packets from queue {m, d, k} to queue {n ,d ,h} at rate $\mu^*_{(m,n)}[t]$.

The next theorem shows that the shortest-path-aided backpressure algorithm is throughput-optimal under per-flow hop constraints, and the proof is presented in Appendix A.

Theorem 1: Given traffic A and hop constraint H such that $((1+\epsilon)\mathbf{A}, \mathbf{H}) \in \Lambda_{\mathcal{G}}$ for some $\epsilon > 0$,the network can be stabilized under the shortest-path-aided back-pressure algorithm, and packets delivered are routed over paths that satisfy corresponding hop constraints .

IV. THROUGHPUT-OPTIMAL AND HOP-OPTIMAL ROUTING/SCHEDULING

In Section III, we proposed the shortest-path-aided back-pressure algorithm that is throughput-optimal and supports per-flow hop constraint .In this section, we consider the scenario where no hop constraint is imposed. Recall that N-1 is an upper bound on the number of hops of loop-free paths. Define Ħ such that Ħ[f]=N-1for all f € F . Then, we can assume that a flow is always associated with hop constraint Ħ, i.e., all loop-free paths are allowed. Note that considering only loop-free paths does not change the network throughput region. Thus, we say A is within the network throughput region if $(\mathbf{A}, \bar{\mathbf{H}}) \in \Lambda_{\mathcal{G}}$, which is also written as $\mathbf{A} \in \Lambda_{\mathcal{G}}$.

In this section, we propose an algorithm that is both throughput-optimal and hop-count optimal, i.e., minimizing the average path lengths. Recall that the motivation to develop a hop-optimal algorithm is that such an algorithm will not only minimize the number of transmissions required to support the traffic, but also reduce the average end-to-end transmission delay. (As we will later see from simulations, minimizing hop count does seem to result in smaller end-to-end delays.)

A. Hop Minimization

Given traffic $\mathbf{A} \in \Lambda_{\mathcal{G}}$, we let $S_{\mathbf{A}}$ denote the set of routing/scheduling policies that stabilize the network .We further define $A_{f,h,\mathcal{P}}[\infty]$ to be the rate at which flow f delivers packets over paths with exactly h hops under policy P , which is well defined when the network can be stabilized. Our objective is to find a policy $\mathcal{P}^*$ such that

$$\mathcal{P}^* = \arg\min_{\mathcal{P}\in \mathcal{S}_{\mathbf{A}}} \sum_{f\in\mathcal{F}} \sum_{N-1\geq h>0} hA_{f,h,\mathcal{P}}[\infty]. \qquad (5)$$

B. Dual Decomposition

To solve optimization problem, we define $\beta_{n,d,h}$ to be the Lagrange multiplier associated with . Then, we can obtain a partial Lagrange dual function as follows:

$$L(\boldsymbol{\beta}) = \min_{\{A_{f,h}\},\hat{\boldsymbol{\mu}}\in\mathcal{CH}(\Gamma)} \Bigg(\sum_{f\in\mathcal{F},N\geq h>0} hA_{f,h} + \sum_{\{n,d,h\}} \beta_{\{n,d,h\}} \times \left(A_{\text{in}(\{n,d,h\})} + \hat{\mu}_{\text{in}(\{n,d,h\})} - \hat{\mu}_{\text{out}(\{n,d,h\})}\right) \Bigg)$$

where

$$\hat{\mu}_{\text{out}(\{n,d,h\})} = \sum_{i:(n,i)\in\mathcal{L}} \hat{\mu}^{\{i,d,h-1\}}_{\{n,d,h\}}$$

$$\hat{\mu}_{\text{in}(\{n,d,h\})} = \sum_{m:(m,n)\in\mathcal{L}} \hat{\mu}^{\{n,d,h\}}_{\{m,d,h+1\}}$$

and

$$A_{\text{in}(\{n,d,h\})} = \sum_{f\in\mathcal{F}} A_{f,h} 1_{s(f)=n,d(f)=d}.$$

C. Joint Traffic-Splitting and Shortest-Path-Aided Back-Pressure Algorithm

We propose a joint traffic splitting and shortest-path-aided back-pressure algorithm. First, note that

$$\sum_{n,d,h} \beta^*_{n,d,h} \left(\hat{\mu}_{\text{in}(\{n,d,h\})} - \hat{\mu}_{\text{out}(\{n,d,h\})}\right)$$

is linear in terms of $\hat{\boldsymbol{\mu}}$. Thus, we have

$$\max_{\hat{\boldsymbol{\mu}}\in\mathcal{CH}(\Gamma)} \sum_{n,d,h} \beta^*_{n,d,h} \left(\hat{\mu}_{\text{in}(\{n,d,h\})} - \hat{\mu}_{\text{out}(\{n,d,h\})}\right)$$
$$= \max_{\boldsymbol{\mu}\in\Gamma} \sum_{n,d,h} \beta^*_{n,d,h} \left(\mu_{\text{in}(\{n,d,h\})} - \mu_{\text{out}(\{n,d,h\})}\right)$$

Note that the Lagrange multiplier $\beta(n,d,h)$ is related to queue length $Q\{n,d,h\}$, and (7)–(10) are the same as conditions (i)-(iii) defined in Section IV-A, so equality (14) motivates us to use the shortest-path-aided back pressure defined by (4).

V. SIMULATIONS

In this section, we use simulations to study the performance of the proposed joint traffic-splitting and shortest-path-aided back-pressure algorithm .We use the term *the joint algorithm* to refer to the joint traffic-splitting and shortest-path-aided backpressure algorithm. The simulations were implemented using OMNeT++.

A. Simulation Setup

We consider a network with 64 nodes as shown in Fig. 3.The network consists of four clusters, and each cluster is a 4*4 regular grid with two randomly added links .Two neighboring clusters are connected by two links. Here, only two links are used to connect two clusters instead of four or

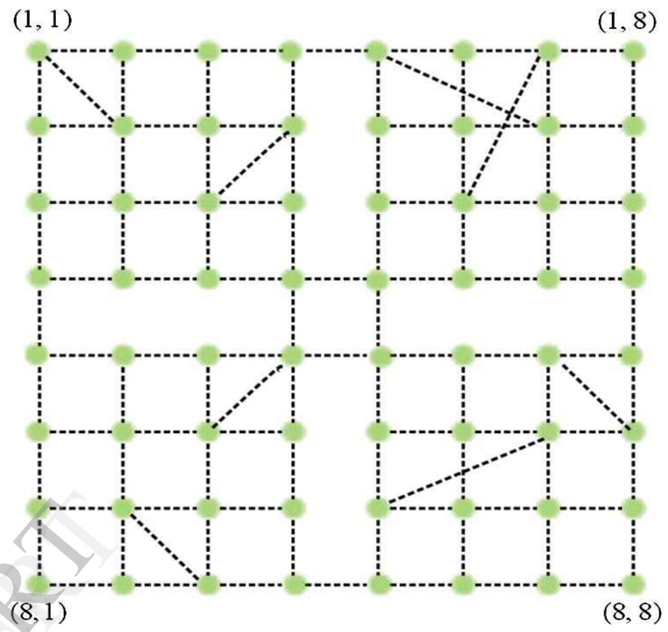

Fig.3. Topology of the network used in the simulations.

more. This is to "force" inter cluster flows to be routed over long paths when the traffic load is high so that the traffic-splitting behavior of the joint algorithm can be easily observed. All links are bidirectional links with capacity one packet/time slot for both directions. All links are assumed to be orthogonalized so they can transmit simultaneously. The propagation delay of a link is assumed to be zero.

TABLE I
FLOWS IN THE NETWORK

Flow ID	(Source, Destination)
1	((1,3), (2,5))
2	((2,3), (2,7))
3	((2,2), (1,6))
4	((3,4), (2,7))
5	((1,1), (1,7))
6	((4,3), (5,4))
7	((4,6), (6,6))
8	((5,3), (5,6))

Eight traffic flows were created in the network, as listed in Table I. Flows 1–5 are inter-cluster flows, and the rest are intra-cluster flows. The packet arrivals of all flows follow Poisson processes. We fixed the arrival rates of intra-cluster flows to be0.2 packets/time slot. All inter-cluster flows have the same arrival rate, denoted by λ(packets/time slot).

B. End-to-End Packet Delays

We also computed the average end-to-end packet delay, averaging over all successfully delivered packets. Similar to the hop count, in Fig.4, we observe that the back pressure performs very poorly when λ is small. This can be attributed to the excessive looping in the route of each packet and can roughly be interpreted as a random walk on the two-dimensional network.

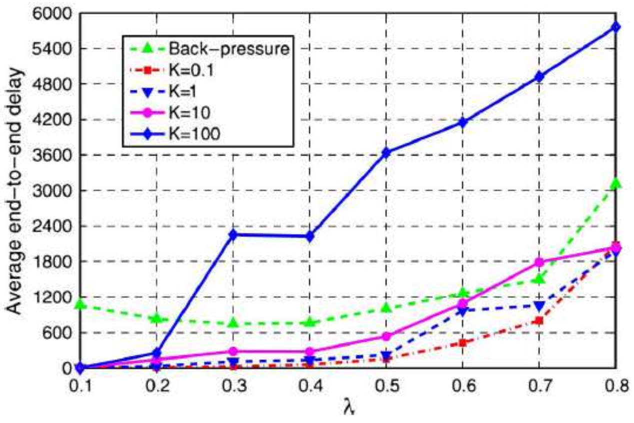

Fig.4. Average end-to-end packet delays under the back-pressure algorithm and the joint algorithm with different K's.

When λ is large, we also observe some improvement of the joint algorithm, with K=0,1,1and10, over the back pressure algorithm. The improvement decreases because the joint algorithm has to exploit long paths in a heavy traffic regime. We further note that the joint algorithm with K=100 performs very poorly in terms of end-to-end packet delay while it has the smallest average hop count. As we have seen in the analysis of Theorem 3, $K \to \infty$ minimizes the average hop count, but results in large queues, hence large end-to-end packet delays.

C .Queue Lengths

Here, we study the total queue length at each node. The average queue length was obtained by averaging over the 100 000 iterations and over all nodes in the network. Fig.5 illustrates the average queue lengths under the joint algorithm with different K's .We observe that the average queue length increases as K increases.

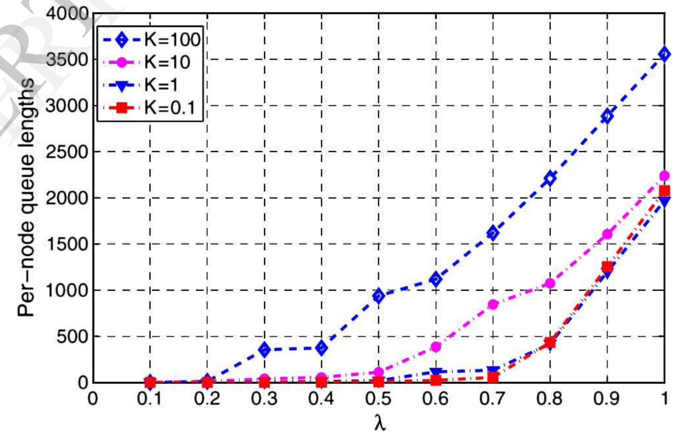

Fig.5. Performance of the joint algorithm with different values of K.

In the simulations, we varied to observe the performance of the back-pressure algorithm and the joint algorithms under different traffic loads. For each, the simulation is executed for 100 000 iterations. When ties occurred in deciding the traffic split or computing the back pressure of a link, we selected the first obtained solution.

D. File Transfer Delay

We also investigated file transfer delays (the duration from the time a file enters the network until it is received at the destination). We compared the back-pressure algorithm with the joint algorithm with K=1. In this simulation, files belonging to the same flow are injected into the source of the flow one by one, and the second file arrives after all packets of the first file are sent out from the source. After a file arrives, the packets of the file are injected into the source node with a constant rate until the complete file is injected .

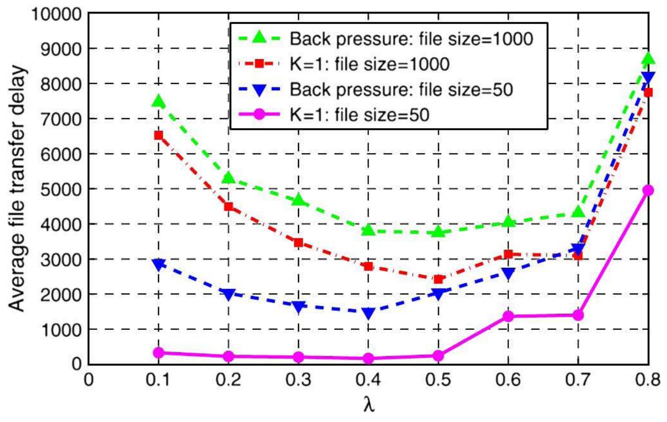

Fig.6. Back-pressure versus the joint algorithm with K=1

.

Under back-pressure algorithm and the joint algorithm, some packets may be queued in the network for a very long time .We therefore assume the packets of a file are coded using rate less codes so that a file can be completely recovered when 90% of the coded packets are received. Fig. 6 illustrates the file transfer delays of the joint algorithm with K=1 and the back-pressure algorithm. As we can see, when the mean file size is 50, the joint algorithm performs significantly better than the back-pressure algorithm in both light or medium traffic regimes ,but performs similarly to the back-pressure algorithm in the heavy traffic regime. This is because in the heavy traffic regime, the end-to-end packet delays of the two algorithms are similar.

VI. CONCLUSION

In this paper, we have proposed new routing/scheduling algorithms that integrate the back-pressure algorithm and shortest path routing. Using simulations, we have demonstrated a significant end-to-end delay performance improvement using the proposed algorithm.

REFERENCES

[1] L. Ying, S. Shakkottai, and A. Reddy, "On combining shortest-path and back-pressure routing over multihop wireless networks," in *Proc.IEEE INFOCOM*, Rio de Janeiro, Brazil, 2009, pp. 1674–1682.
[2] L. Tassiulas and A. Ephremides, "Stability properties of constrained
queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[3] L. Tassiulas and A. Ephremides, "Dynamic server allocation to parallel queues with randomly varying connectivity," *IEEE Trans. Inf. Theory*, vol. 39, no. 2, pp. 466–478, Mar. 1993.
[4] X. Lin and N. Shroff, "Joint rate control and scheduling in multihop
wireless networks," in *Proc. IEEE CDC*, Paradise Island, Bahamas, Dec. 2004, vol. 2, pp. 1484–1489.
[5] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks
using queue-length-based scheduling and congestion control," in *Proc. IEEE INFOCOM*, 2005, vol. 3, pp. 1794–1803.

BIOGRAPHIES

A.Priyadharsini received her B.E degree in Computer Science & Engineering at Sri Lakshmi Ammal Engineering College, Chennai, Tamilnadu, in 2011. Currently, she is studying M.E. Computer Science and Engineering at Arunai College of Engineering, Tiruvannamalai, Tamilnadu. Her project and research area includes networking, wireless communication.

S.Mohanarangan, obtained his M.Tech from Sathyabama university, chennai, Tamilnadu. Currently he is doing his Ph.D in Anna university, chennai, Tamilnadu. He is working as an Associate Professor and Head of the Department of Computer Science and Engineering in Arunai College of Engineering, Tiruvannamalai, Tamilnadu .He is having 14 years of experience in teaching . His research area is networking and communication.