International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 4 | ISSN : 2456-3307

A Comparative Analysis of Delay Constrains In Mobile Ad Hoc Networks Using Max Weight Scheduling and Back Pressure Algorithms

[1]B. Sindhupiriyaa, [2]Dr. D. Maruthanayagam
[1]Research Scholar, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India
[2] Head/Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India

ABSTRACT

In Wireless networks where the nodes routing and scheduling packets depend on queue overload differences, one can stabilize the queues for any feasible traffic. The delay analysis of throughput scheduling policies in such systems is extremely difficult due to complex correlations arising between the arrival, service and the queue length process. In this paper we give a regular evaluation of Back Pressure Routing algorithm and Max Weight Scheduling algorithm variants on an experimental testbed. This provides the first direct comparison of delay performance. Our result expose that even in simple network topologies these algorithms induce wide routing loops with connected high packet delay.

Keywords : MANET, Back pressure, MaxWeight Scheduling (MWS), Packet Delay and Throughput

I. INTRODUCTION

MANET is dynamically establishing mobile nodes networks with no fixed infrastructure [1]. Each mobile node is equipped with wireless transmitter and a receiver with a suitable antenna. Nodes in mobile ad hoc networks move freely in the network and they can organize themselves in a random way. The important sector of ad-hoc network is routing protocols because network topologies keep on changing due to the movement of the nodes. All the network related activities like discovering of topology and delivery of packets is performed by the nodes itself. The nodes communicate over wireless links; they have to compete with the effects of radio communication, such as noise and interference. In MANET the links typically have less bandwidth than a wired network [3]. Each node in a wireless ad hoc network functions as a host as well as a router [2]. The control of the network is distributed among all the nodes of the network. In a MANET system, users compete for accessing a shared transmission medium. Since link transmissions cause mutual interference, the medium access layer (MAC) is needed to schedule the links carefully so that packets can be transmitted with minimal collisions. Many scheduling policies have been studied at the MAC layer with the objective of maximizing throughput. These schemes are often called throughput-optimal scheduling schemes. However, the delay analysis of these systems has largely been limited.

The obvious candidate for scheduling and routing in MANET scenario is the Backpressure algorithm, which routes and schedules packets based on differential backlogs (i.e., queue length differences from a one-hop downstream node). This algorithm is known to be stabilizing; however, it is known that it can have poor delay performance [4]. An alternative, which simply looks at backlogs and not differential backlogs, is the Max-Weight algorithm [5]. The Max-Weight Algorithm assigns a weight of (queue-length X channel-rate), and schedules a collection of links that maximizes the total weight (Max-Weight independent set). This algorithm is however, not stabilizing in general. Our focus in this paper is to analyze the expected delay for this system. To that end, we will derive comparison from two algorithms for delay performance, and also provide an accurate estimate of the expected delay for a well-known and extensively

studied of Maximum Weighted Matching (MWM) and Back Pressure Algorithms.

II. MAX WEIGHT SCHEDULING

In single channel wireless networks, links need to be scheduled for data transmission due to the presence of interference among them. Link scheduling has been known to be a challenging problem when the design objective is to optimize achievable throughput (capacity). Let us assume that associated to each link is a queue and packets are queued before they are transmitted over the link. The optimization problem is to choose a set of non-interfering links with the maximum sum of weights, where the link weights are the queue sizes on the links [6][7]. This solution however, needs that a central and high computational complexity algorithm to be executed at each time slot in the network. Many research efforts then conducted in this area to develop scheduling algorithms with lower complexities which are more applicable in wireless networks without central coordinator node, like Ad Hoc networks. Link scheduling algorithm under 1-hop interference model is equivalent to maximum weight matching in graph theory which suffers high computational complexity. Then, a trend of research study in graph theory is to devise approximation algorithms with low complexity that achieves a fraction of optimal solution. A simple approximation algorithm for maximum weight matching problem is greedy algorithm that guarantees to achieve a fraction 1/2 of optimal solution (maximum weight matching). The greedy algorithm begins with an empty set and extends it in each round by adding the heaviest edge currently available.

We consider a wireless network, G with N links denoted by set L. The capacity (maximum number of packets that can be transmitted in one slot) of link I is given by C_I. Let $si\ (t) \in \{ON, OFF\}$ represent the channel state of link l during time slot t. Assume that these channel states are random variable over timeslots and independent across channels, and let ρi l represent the probability of link l to be ON. Each link has its own exogenous arrival stream $\{A1(t)\}_{t=1.}^{\infty}$. Each arrival stream is random variations. in time. Time is slotted. The distribution of the number of packets, $A1(t)$, arriving to a link I in any given time slot t may be arbitrary but time invariant. Each packet has deterministic service time equal to one unit. Assume that the second moments, $E\left[A\frac{2}{I}\right]$, of the arrival processes are finite. Different input streams may be correlated with each other. Let $A(t) = (A_1(t), ..., A_N(t))$ represent the vector of exogenous arrivals, where $A_1(t)$ is the number of packets that arrive to link I during time slot t $(for I \varepsilon 1, ..., N)$, Let $\lambda = (\lambda_1, ..., \lambda \mathrm{N})$ represent the corresponding arrival rate vector[9][10].

The packets arriving at each link are queued. Let $Q_1(t)$ denote the queue length at link I. The queue length vector is denoted by $Q(t) = (Q_1(t) := 1, 2..., \mathrm{N})$. A link can be activated in a time slot t only if the queue is non empty. We use the term activation (scheduling) of a link or a queue interchangeably in this paper. After service, each packet leaves the system. There is a slotted service structure. For each link I, the indicator function Il (t) indicates whether or not link I received service at time slot t. Note that

The evolution of the queue is as follows,

$$Q_1(t+1) = \left(Q_1(t) - I_1(t) C_1(t) 1_{\{a1(t)=ON\}}\right) + + A_{I(t),} I = 1, ..., \mathrm{N}$$

Otherwise, Where,

$$(x)^+ = \begin{cases} x & if\ x > 0 \\ 0 & otherwise \end{cases}$$

Define residual capacity r_1 as follows.

$$r_1(t) = \begin{cases} C_1(t) - Q_1(t)\ if\ Q_1(t) < C_1(t) \\ 0 \qquad\quad otherwise \end{cases}$$

Then, the queue evolution can be written as

$$Q_1(t+1) = Q_1(t) - I_1(t) C_1(t) - r_1(t) 1_{\{a1(t)=ON\}} + A_I(t), \mathrm{I} = 1, ... \mathrm{N}$$

The vector of the scheduled queues is denoted by $I(t) = (I_n(t)) : 1, ... \mathrm{N}.$ Because of interference, there are constraints on the combination of links that can be activated simultaneously. We allow these constraints to be arbitrary. $I(t)$ is a valid activation vector if it satisfies these constraints. Let S be the collection of all activation vectors, I^j. At each timeslot an activation vector $I(t)$ is scheduled. A scheduling policy decides which activation vector is used in every time slot.

Let $\|Y\|$ denote the Euclidean norm of vector Y. The system is considered to be stable if $\lim_{t\to+\infty} E[\sup \|Q(t)\|]$ is bounded. If the system is stable then the throughput is the same as the arrival rates. A throughput vector λ is admissible if there is some scheduling policy under which the system is stable when the arrival rate vector is λ. Let us denote by $\wedge$ the closure of the convex hull of the set of activation vectors, I^j and by C the interior of the convex hull. Note that $\wedge$ is a closed convex set. It has been shown in [6] that if each arrival process is random variation in time, and the first two moments of all the arrival streams $\{A_i(t)\}_{t=1}^{\infty}$ are finite, then $\lambda \in$ C is a necessary condition for a stabilizing scheduling policy to exist. It is also shown that the MWM policy, that chooses the maximum weighted activation vector (matching), stabilizes the system for any arrival rate satisfying the preceding condition.

MWM Scheduling Policy,

$$I(t) = \arg\max_{I^j \varepsilon S} \sum_{i=1}^{N} Q_i C_i 1\{s_i = ON\} I_i^j$$

Where I^j is the i^{th} component of the j^{th} activation vector, I^j, in set S.

Lower Bound Analysis: In this section, we present our methodology to derive lower bounds on the average packet delay for a given multi-hop wireless network. The first step is to identify the bottlenecks in the system. We then explain how to lower bound the average delay of the packets in a given bottleneck. Finally, we present a greedy algorithm which takes as input, a system with possibly multiple bottlenecks, and returns a lower bound on the system wide average packet delay. Link interference causes certain bottlenecks to be formed in the system. Define a bottleneck to be a set of links $X \subset L$ such that no more than one of its links can be scheduled simultaneously. Our idea of bottleneck is equivalent to identifying cliques in the conflict graph which was used by [10][11] to estimate the capacity region of a given wireless network. We call these sets of links, *exclusive sets*. We demonstrate our methodology to derive lower bounds on the average size of the queues corresponding to the links that belong to an exclusive set. Then by the definition [5] to estimate the capacity region of a given wireless network. We call these sets of links, *exclusive sets*.

The Algorithm 1 maintains a table $T(i)$ which indicates the number of times link i has been used in the bottleneck. The value of $T(i)$ is initialized to C_i. The algorithm proceeds by greedily searching for a bottleneck that yields the maximum lower bound. For each link in the chosen bottleneck, the value of $T(i)$ is decremented by 1 and the process is repeated until the table T has a non zero entry. Thus it decomposes the wireless network into several single queue systems. The average delay of the system can then be easily computed. Note that the decomposition obtained by the greedy algorithm is not the optimal decomposition. The optimal decomposition can alternately be obtained by using a dynamic programming approach with the cost of increased computation complexity.

Algorithm 1: Computing the Lower Bound

1: **for** $i = 1 to N$ *do*
2: $T(i) \leftarrow C_i$
3: **end for**
4: $BOUND \leftarrow$
5: **repeat**
6: Find the bottleneck which maximizes $E[Q_x]$
7: $BOUND \leftarrow BOUND + E[Q_x]$
8: **for all** $i \in X$ **do**
9: $T(i) \leftarrow T(i) - 1$
10: **end for**
11: **until** $\forall i, \quad T(i) = 0$
12: **return** *BOUND*

The lower bound may be loose on account of the following. We assume that the queuing in the bottlenecks is independent of each other, which may not be possible because of interference. Moreover, in the derivation of the lower bound by the reduction technique, we have neglected the non-empty queue constraints by grouping the arrivals into a single queue, and hence we underestimate the delay. Since the exclusive sets do not completely characterize the capacity region of the network, it may also be expected that if the input load is close to a boundary of the capacity region C, which is different from the boundaries generated by the exclusive sets, the lower bound may perform poorly. Thus, in certain cases, the delay of the system under MWM policy may be close to infinity while the lower bound is much smaller. This

motivates the development of an upper bound for the system, which is tight in the sense that whenever the upper bound goes to infinity, the delay of the system under a throughput optimal policy also becomes infinite[12].

Upper Bound Analysis: Generalized Maximum Weighted Matching (GMWM (**w**)) policies, parameterized by weights w_i which is described in below. The MWM policy is a special case, where all the weights w_i are unity. We establish the following bounds on the sum of the expected queue lengths and the expected delay in the system [13].
GMWM Scheduling Policy,

$$I(t)\arg\max_{Ij\in S}\sum_{i=1}^{N}\left(\omega_i Q_i 1_{\{s_{1=ON}\}} I_i^j\right)$$

where I^j is the i^{th} component of the j^{th} activation vector, I^j, in set S and $\omega_i > 0$ are fixed constants.

III. BACKPRESSURE ROUTING ALGORITHMS

Backpressure routing refers to an algorithm for dynamically routing traffic over a multi-hop network by using congestion gradients. It usually refers to a data network, but can apply to other types of networks as well. Below we focus on the data network application, where multiple data streams arrive to a network and must be delivered to appropriate destinations. The backpressure algorithm operates in slotted time, and every slot it seeks to route data in directions that maximize the differential backlog between neighboring nodes. This is similar to how water would flow through a network of pipes via pressure gradients. However, the backpressure algorithm can be applied to multi-commodity networks and to networks where transmission rates can be selected from different (possibly time varying) options.

Backpressure routing is an algorithm for dynamically routing traffic over a multi-hop network by using congestion gradients. The algorithm can be applied to wireless communication networks, including sensor networks, mobile ad hoc networks (MANETS), and heterogeneous networks with wireless and wire line components. Attractive features of the backpressure algorithm are:

- It leads to maximum network throughput.
- It is robust in comparison with time-varying network conditions.
- It can be implemented without knowing traffic arrival rates or channel state probabilities.
- A numeric value that maximize or minimize the objective function, and plays a major role in routing.

The objective function specifies how much each variable contributes to the optimization problem. In this case the optimal routing in multi-hop wireless networks is the problem and variables include queue length, the quality of the links and etc. The scheduling and routing of Backpressure guarantee the optimal throughput performance and is a promising technique for improving throughput in a wireless multi-hop mesh network. In the classic Back-pressure, each node makes queue for each flow. Each node makes decision for routing and scheduling based on the existing congested packets in the queue and the status of the network [2].

Basic Back-pressure Algorithm

The basic back-pressure algorithm [2], [3] is detailed in Algorithm 1. Each forwarding node in the network uses per flow FIFO queuing and we let q_n^f (t) denote the number of packets queued for flow f at node n at time t. Roughly speaking, whenever it has a transmission opportunity each node n forwards a packet from the flow $f*$! to the next hop $m*(f*)$ that jointly maximizes the utility

$$u_{n,m}^f(t) = \left(q_n^f(t) - q_m^f(t)\right)R_{n,m}$$

Where $R_{n,m}$ is the mean transmit rate of the link from node n to node m.

Algorithm:

- For each flow f and neighbor node m, node n computes utility $u_{n,m(t)=\left(q_n^f(t)-q_m^f(t)\right)R_{n,m}}^f m^*(f) = \arg\max_m u_{n,m,}^f f^* = \arg\max\left(u_{n,m*(f)}^f\right)$ (ties broken arbitrarily).
- If $u_{n,m,}^{f*} f^* > 0,$ then node n schedules flow $f*$ and forwards $\min\left(q_n^{f*}(t); R_{n,m*(f*)}\right)$ packets to neighbor $m*(f*)$.
- Otherwise node n takes no action at time t.

Observe that this back-pressure algorithm tends to transmit packets to the neighbor(s) with the smallest queue and highest link rate and intuitively the queue backlogs provide a "gradient" down which packets are routed. However, it commonly occurs that $q_n^f(t)$ and $q_m^f(t)$ differ by only a small amount. The routing "gradient" is then both small and rapidly fluctuating, in which case routing loops can readily be induced. Furthermore, observe that even when a neighbor has no connectivity to the destination of a flow, packets will still be forwarded to this neighbor until such time as a sufficiently large queue backlog has developed to prevent further packets stop being forwarded in that direction. However, all the packets already sent in that direction will never reach the flow destination.

Stages of Backpressure algorithm:

Backpressure [2][14] is a joint scheduling and routing policy which favors traffic with high backlog differential. 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

$$\omega t_{nmd}(t) = \max\left(Q_n^d - Q_{m,}^d 0\right).$$

Then, define the maximum difference of queue backlogs according to:

$$\omega t_{nm}(t) = \max_{d \in D} \omega \mathrm{t}_{nm}(t).$$

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:

$$\mu^*(t) = \arg\max_{\mu \varepsilon \dagger} \sum_{(n,m)\varepsilon L} \mu_{nm}(t)$$

where T 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).

IV. EXPERIMENT AND RESULTS

Experiment SetUp: We conducted our experiments on a testbed constructed from 10 switched leoxsys network boxes. These boxes run Windows with BSNL (WiFi) and have a 433MHz CPU, 2GB RAM, and four 100Mbps ethernet ports. As the link rates in the scenarios (see Fig 1) that we consider are less than the 100Mbps physical rate, we included a delay component within the Routing element which introduces a specified minimum delay between packets delivered to the Device element, thereby allowing lower link rates to be emulated in a controlled manner. The MTU is taken as 1400 Bytes and the interval between neighbor-update packet transmissions is set to 1 millisecond.

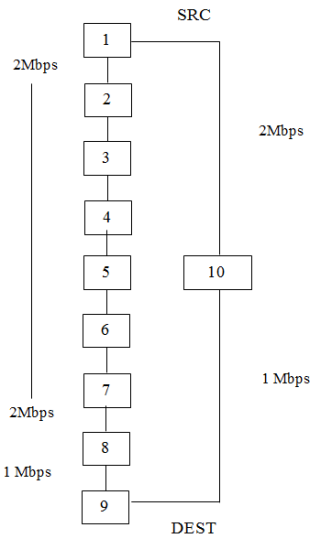

Figure 1: Network topologies. (SRC and DEST denote packet flow's between source and destination nodes, resp.)

Performance Metrics: It is important to emphasize that our goal is not to achieve exhaustive testing, which

is in any case impossible due to the large number of network topologies, device configurations and traffic mixes that exist in modern networks. Instead we seek to define a small number of benchmark tests that are amenable to systematic, reproducible testing and which exercise the core functionality of the routing algorithms. As we will see, relatively simple network configurations are already sufficient to uncover a number of basic issues with existing backpressure algorithms.

We consider the following performance metrics.

a) Packet delay This is computed at the sender as the difference between the time when a packet P is transmitted and the time Trans(P) when the packet is delivered to the application layer at the receiver. As the source and destination clocks will not be perfectly synchronized, we compute Trans(P) as follows. The source node sends TS(p) in the header of each transmitted packet. The receiver echos this TS(p) value to the sender in a small 8 byte packet sent via a dedicated ethernet port operating at 100Mbps and connected by a cross-over cable. Since the transmit time of an 8 byte packet at 100Mbps is approximately 0.6μs, the time Trans(P) can be accurately estimated as the time when this echo packet arrives at the source and the packet delay is then calculated as Trans(P) − TS(p). The *packet delay* can break down into a component due to network delay (the time between when a packet leaves the source and when it arrives at the destination) and a component due to reassembly delay (the time between when a packet enters the reassembly queue at the receiver and when it is delivered in order to the application layer).

b) Throughput This is computed at the receiver as X/T where X is the amount of data received over a time period T, where T = 500s.

c) Buffer Requirements We measure the average occupancy of every per-flow queue at every node, with packet-level timing granularity. Also the average re-assembly queue size at destination nodes. Each experiment is conducted for 5 different runs, with each run being of 500 seconds duration. Results are plotted with 95% confidence intervals marked.

Results: we present performance measurements over the topologies shown in Fig. 1 for the two algorithms like ***Back-Pressure algorithm (BP) and Max Weight Scheduling (MWS).*** The topologies are intentionally kept simple, so that we can have a rough understanding on the minimum delay, however they require routing along multiple paths to achieve the network capacity and also they can induce implicit routing loops. The results shown here are for TCP Poisson traffic.

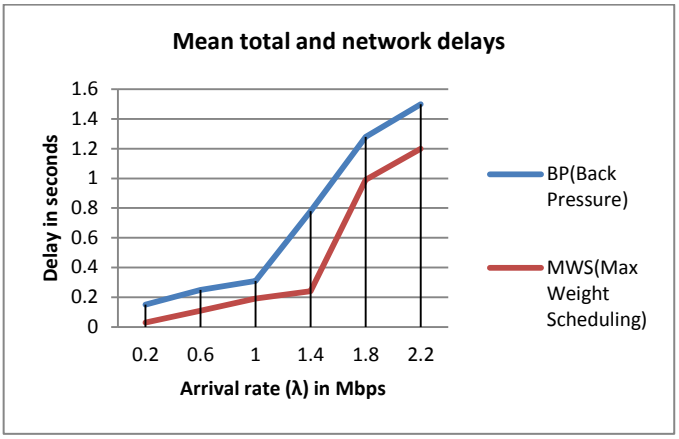

Figure 2: Mean total and network delays

We begin by considering the network topology shown in Fig. 1. Fig. 2 shows the mean measured packet delay (and also network delay) vs offered load. We can make number of observations from this result.

- The total delay is much higher (almost x 3 times) than the network delay at most of the offered loads, which shows the severity (and also the significance) of packet reordering on the delay performance of the algorithms.
- For both BP and MWS algorithms first the delay (both network and total delay) decreases and then increases with offered load. This is expected as at lighter loads packets are routed where ever excess network capacity is available, thus causing high packet reordering, and at higher loads the increased delays are associated with both reordering and queuing delays.

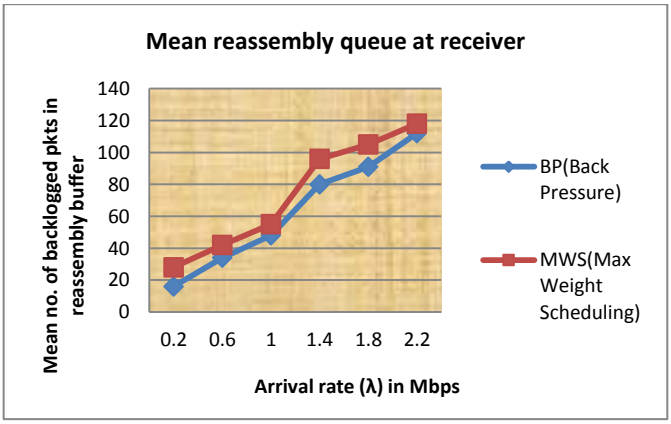

Figure 3: Mean reassembly queue at receiver

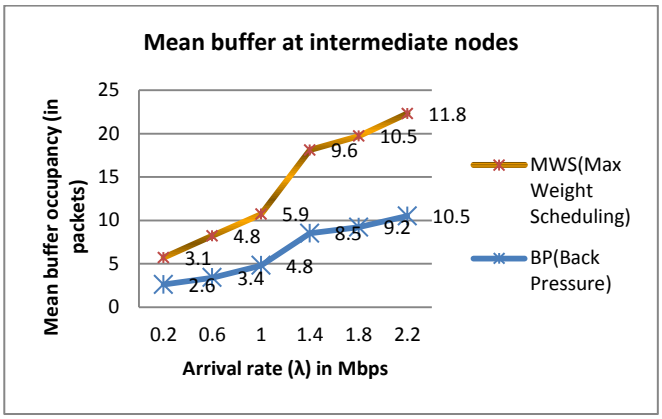

Figure 4: Mean buffer at intermediate nodes

Figs. 3 and 4 give insight into the buffer sizing requirements of the algorithms considered. Fig. 3 plots the mean number of packets in the reassembly buffer at the receiver. The reassembly buffer is used to ensure in-order delivery of packets to the application layer and so its occupancy is dependent on the amount of packet re-ordering. It can be seen that the reassembly buffer requirements for the BP and MWS algorithms are high, with on average more than 100 packets buffered at the destination at an offered load of 1.9Mbps. Fig. 4 shows the mean buffer occupancy at forwarding nodes. Observe that, as expected the queue length with the BP algorithm are essentially the same at low to moderate loads, and at moderate to high offered loads the MSW grow faster than the queues in BP routing.

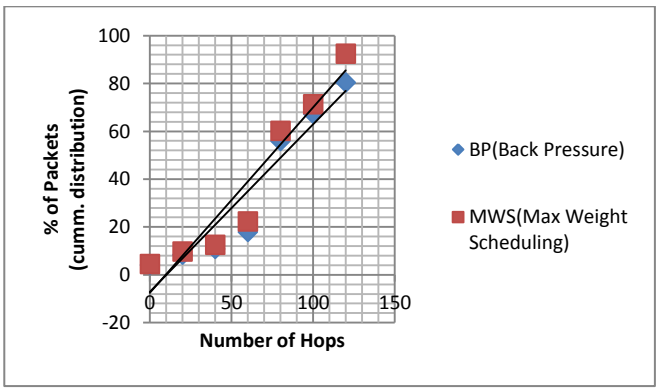

Figure 5: Hop count distribution at 1.4Mbps

We can also understand the severity of packet-reordering via hop count results show in Fig. 5, which plots the cumulative distribution of the number of hops traversed by a packet as it travels from SRC to DEST. Observe that at 1.4Mbps, 15-20% of the packets in all the algorithm traverse more than 20 hops, and the maximum number of hops traversed for all these algorithms is above 100 hops. Also it can be observed that both BP and MSW algorithms show essentially same performance. We can note that even such mild gain in hop count performance leads to significant gain in the delay performance as can be seen in Fig. 2.

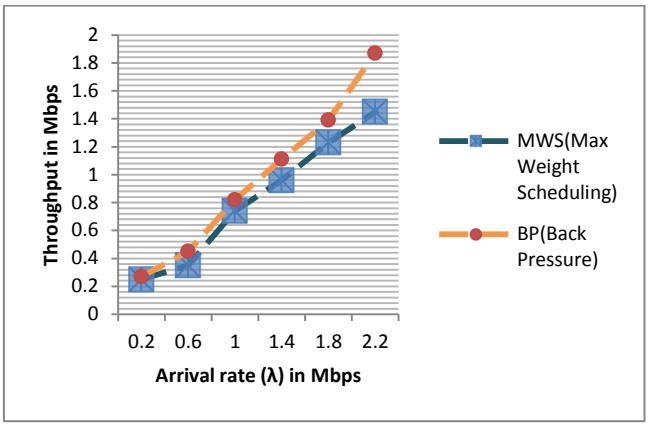

Figure 6: Throughput Performance

Fig. 6 shows the measured throughput for BP and MWS algorithms. As these algorithms are throughput optimal, it can be seen that throughput attained is roughly same as the offered load. This result demonstrates the potential for substantial throughput gains via multi-path back-pressure routing.

V. CONCLUSION

In this paper we present an experimental performance evaluation between back-pressure routing algorithms, and MaxWeight Scheduling providing the comparison of delay performance. We find that, while lowering delay relative to the BP and MWS algorithms, the absolute value of delay nevertheless remains large due to a tendency for the algorithms to induce extensive routing loops. Apart from MWS algorithm, the BP algorithms considered involve design parameters that significantly affect performance yet lack guidelines as to how they should be chosen. In our tests we found that the appropriate value are strongly dependent on the network and flow configuration. Motivated by these observations, our future work includes enhancing MWS with a mechanism to adaptively updating the design parameters for each flow by maintaining a recent history of delay and throughput trends.

VI. VI.REFERENCES

[1]. L. Bui, R. Srikant, and A. Stolyar. A novel architecture for reduction of mdelay and queueing structure complexity in the back-pressure algorithm. IEEE/ACM Trans. Network., 19(6):1597–1609, 2011.

[2]. A. Stolyar. Large number of queues in tandem: Scaling properties under back-pressure algorithm. Queueing Systems, 67(2):111–126, 2011.

[3]. L. Tassiulas and A.Ephermides. Dynamic server allocation to parallel queues with randomly varying connectivity. IEEE Trans. Automat. Contr., 39:466–478, 1993.

[4]. L. Ying, S. Shakkottai, and A. Reddy. On combining shortest-path and back-pressure routing over multihop wireless networks. In Proceedings of IEEE Infocom, 2009.
[5]. Tassiulas, L., & Ephremides, A. (1992). Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. IEEE Transaction on Automatic Control, 37(12), 1936–1948
[6]. A. L. Stolyar, "Maximizing Queueing Network Utility subject to Stability: Greedy-Primal Dual Algorithm," *Queueing Systems*, vol. 50, no.4, pp. 401-457, 2005.
[7]. M. J. Neely et al, "Fairness and Optimal Stochastic Control for Heterogeneous Networks," *in Proc. IEEE Infocom*, 2005.
[8]. Sharma, G., Mazumdar, R. R., Shroff, N. B. (2006). On the complexity of scheduling in wireless networks. In Proceedings ACM MobiCom (pp. 227–238). NY, USA
[9]. Tassiulas L. (1998). Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In Proceedings of IEEE INFOCOM, vol. 2, pp. 533–539
[10]. Modiano, Eytan, Shah, Devavrat, Zussman, Gil (2006). Maximizing throughput in wireless networks via gossiping. In Proceedings of ACM SIGMETRICS, 34, vol. 34, no. 1, pp. 27–38.
[11]. Gupta, A., Lin, X., Srikant, R. (2009). Low-complexity distributed scheduling algorithms for wireless networks. IEEE/ACM Transaction on Networking, 17, 1846–1859.
[12]. David, A. (1983). A survey of heuristics for the weighted matching problem. Networks 13(4), 475–493.
[13]. Hoepman, J. H. (2004). Simple distributed weighted matchings. In In eprint cs.DC/0410047
[14]. Preis, R. (1998). Linear time 1/2-approximation algorithm for maximum weighted matching in general graphs. In In general graphs, symposium on theoretical aspects of computer science, STACS 99 (pp. 259–269). Springer: Berlin
[15]. Chaporkar, P., Kar, K., Luo, Xiang., & Sarkar, S. (2008). Throughput and fairness guarantees through maximal scheduling in wireless networks. IEEE Transactions on Information Theory, 54(2), 572–594.
[16]. Wu, X., Srikant, R., Perkins., & James R. (2007). Scheduling efficiency of distributed greedy scheduling algorithms in wireless networks. IEEE Transactions on Mobile Computing 6, 595–605.
[17]. Abbas, A. M., & Kure, O. (2010) Quality of service in mobile ad hoc networks: A survey. International Journal of Ad Hoc and Ubiquitous Computing, 6, 75–98
[18]. Neely, M. J. (2008a). Order optimal delay for opportunistic scheduling in multi-user wireless uplinks and downlinks. IEEE/ ACM Transaction on Networking 16(5), 1188–1199.
[19]. Neely, M. J. (2008b). Delay analysis for max weight opportunistic scheduling in wireless systems. In Communication, Control, and Computing, 2008 46th Annual Allerton Conference on, pp. 683–691.
[20]. Le, L. B., Jagannathan, K., & Modiano, E. (2009). Delay analysis of maximum weight scheduling in wireless ad hoc networks. In Information sciences and systems, 2009. CISS 2009. 43rd annual conference on, pp. 389–394.
[21]. Neely, M. J. (2009). Delay analysis for maximal scheduling with flow control in wireless networks with bursty traffic. IEEE/ACM Transaction Networking, 17(4), 1146–1159, ISSN 1063–6692.
[22]. Gupta, G. R., & Shroff, N. B. (2010). Delay analysis for wireless networks with single hop traffic and general interference constraints. IEEE/ACM Transaction on Networking, 18, 393–405.
[23]. Gupta, G. R., & Shroff, N. B. (2011). Delay analysis and optimality of scheduling policies for multihop wireless networks. IEEE/ACM Transaction on Networking, 19, 129–141.
[24]. X. Wang and K. Kar. Cross-layer rate control for end-to-end proportional fairness in wireless networks with random access. In *MOBIHOC '05*, pages 157–168, New York, NY, USA, 2005. ACM.
[25]. Venkataramana Badarla and Douglas J. Leith On Delay Performance of Throughput Optimal Back-pressure Routing: Testbed Results, Hamilton Institute, NUI Maynooth, Ireland,IEEE

About the Authors

B.Sindhupiriyaa received her M.Phil Degree from Periyar University, Salem in the year 2015. She has received her M.C.A Degree from Anna University,Coimbatore in the the year 2011. She is pursuing her Ph.D Degree at Sri Vijay Vidyalaya College of Arts & Science (Affiliated Periyar University),Dharmapuri, Tamilnadu, India. Her areas of interest include Mobile Computing and Wireless Networking.

Dr.D.Maruthanayagam received his Ph.D Degree from Manonmaniam Sundaranar University,Tirunelveli in the year 2014. He received his M.Phil Degree from Bharathidasan University, Trichy in the year 2005. He received his M.C.A Degree from Madras University, Chennai in the year 2000. He is working as HOD Cum Professor, PG and Research Department of Computer Science, Sri Vijay Vidyalaya College of Arts & Science, Dharmapuri, Tamilnadu, India. He has 15 years of experience in academic field. He has published 4 books, 24 papers in International Journals and 28 papers in National & International Conferences so far. His areas of interest include Computer Networks, Grid Computing, Cloud Computing and Mobile Computing.