Back-pressure routing for intermittently connected networks

Jung Ryu
The University of Texas at Austin

Lei Ying
Iowa State University

Sanjay Shakkottai
The University of Texas at Austin

***Abstract*—We study a mobile wireless network where groups or clusters of nodes are intermittently connected via mobile "carriers" (the carriers provide connectivity over time among different clusters of nodes). Over such networks (an instantiation of a delay tolerant network), it is well-known that traditional routing algorithms perform very poorly. In this paper, we propose a two-level Back-Pressure with Source-Routing algorithm (BP+SR) for such networks. The proposed BP+SR algorithm separates routing and scheduling within clusters (fast time-scale) from the communications that occur across clusters (slow time-scale), without loss in network throughput (i.e., BP+SR is throughput-optimal).**

More importantly, for a source and destination node that lie in different clusters, the traditional back-pressure algorithm results in large queue lengths at each node along its path. This is because the queue dynamics are driven by the slowest time-scale (i.e., that of the carrier nodes) along the path between the source and destination, which results in very large end-to-end delays. On the other-hand, we show that the two-level BP+SR algorithm maintains large queues only at a very few nodes, and thus results in order-wise smaller end-to-end delays. We provide analytical as well as simulation results to confirm our claims.

I. Introduction

Networks with sporadic or intermittent connectivity among clusters of nodes are becoming more common – these range from rural applications where mobile carriers such as buses provide connectivity among villages [1] to military high bandwidth applications where groups of mobile soldiers are connected via mobile carriers (e.g., other land-based or aerial vehicles).

Routing over such networks has been of much interest to the delay tolerant networking (DTN) community, see [2], [3]. Intermittent connectivity results in the non-availability of an end-to-end route at each instant of time, thus leading to poor performance by the conventional Internet or ad hoc network routing protocols. A variety of DTN protocols have been proposed to address these issues. However, to a large extent, algorithms that provide guaranteed performance (e.g. stability, end-to-end buffer-usage or delay) have not been developed.

In this paper, we formally study algorithmic performance associated with such intermittently connected networks. In particular, we address the following questions: *(i)* Can we develop routing algorithms that stabilize the network (i.e., does not let the buffers at nodes build up arbitrarily large), *(ii)* When there are multiple paths between a packet source and destination (e.g., multiple carriers), how do we "optimally" load-balance among these paths, and *(iii)* What is the end-to-end delay/buffer-usage performance of these algorithms?

In the mobile ad hoc network setting, there has been a lot of work on back-pressure (BP) algorithms [4] to address similar questions (see for example [5], [6], [7], [8], [9], [10]). While these queue-based algorithms can stabilize networks under any general topology, their performance in intermittently connected network can be very poor, and results in large queue lengths at *each node along its path*. Further, at each node, the BP algorithm needs to maintain a separate queue for each destination, thus leading to a "state-space explosion" in the number of queues as the network scales in size.

To understand this, we observe that an intermittently connected network inherently has two time-scales of communications: *(a)* a fast time-scale at which nodes within a cluster communicate among themselves, and *(b)* a slow time-scale that is limited by the carriers' mobility speeds among various clusters. With the traditional BP algorithms, the dynamics at *each* node along a packet flow path is essentially driven by the slowest time-scale along the path (this is quantified more precisely in later sections). Since each path between clusters has a slow carrier, BP results in large buffers (and hence large delays) at every node along the flow path.

To address this, we propose a two-level Back-Pressure with Source-Routing algorithm (BP+SR) that separates the two times-scales. Routing (and scheduling) within clusters is loosely coupled to routing across clusters (via carriers) only at cluster *gateway* nodes. Such selective coupling at a small number of nodes permits us to maintain large queues only at a very few nodes, and thus results in order-wise shorter end-to-end buffer-usage as compared to traditional BP. A further consequence of the proposed BP+SR algorithm due to the two-level hierarchy is the much reduced number of queues needed to implement the algorithm.

A. Main contributions

In this paper, we propose a two-level back-pressure routing algorithm (BP+SR) for intermittently connected networks. Such a network consists of two types of traffic: intra-cluster (i.e., traffic which can be routed only within a cluster) and inter-cluster (traffic with a source and destination that are in different clusters). The intra-cluster traffic is routed via traditional back-pressure within the cluster. For the inter-cluster traffic, the algorithm uses source routing to determine a pair of source gateway and destination gateway in the corresponding source and destination clusters (see Figure 1). Packets are routed from the source to the source-cluster-gateway (and

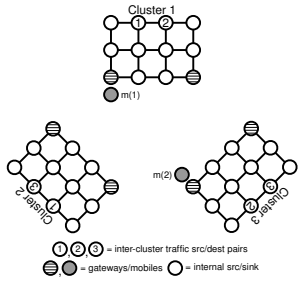


Fig. 1. An example of an intermittently connected network with three 3×4 clusters. The pair of nodes labeled 1 communicate with each other, likewise for pairs 2 and 3.

from the destination-cluster-gateway to the destination node) via a "local" (intra-cluster) back-pressure algorithm. Another back-pressure algorithm is used to determine the "optimal load-balancing" over an *overlay network* consisting of the gateway nodes of various clusters (thus, the various clusters form "stepping stones" to help deliver the packet from the source to destination).

Our contributions are as follows:

1) The proposed two-level back-pressure routing is throughput-optimal, i.e., it stabilizes all queues in the network if at all possible to do so under any algorithm.
2) The algorithm results in much smaller end-to-end buffer usage as compared to traditional BP. In particular, consider a network with a finite number of clusters where each cluster is of size N_c and a carrier moves between the clusters every T time slots and spends π fraction of the time at each cluster. The traditional BP algorithm will have a per-path end-to-end buffer usage[1] that scales as $\Theta(\pi^{-1}N_C T)$ whereas the corresponding quantity with the BP+SR algorithm is $\Theta(\pi^{-1}T)$, which does not depend on the cluster size N_c for reasonably sized clusters where $T >> N_c$.

B. Related work

Due to its throughput optimality and simplicity, back-pressure routing algorithm has attracted much attention since [4]. In [5], [6], [7], the back-pressure algorithm has been combined with congestion control to provide utility-maximizing resource allocation to elastic flows. The problem of delay reduction for back-pressure algorithm has been studied in [10], [9], [11]. However, their algorithm solves the problem of large delay when the offered load on the network is light (when the load is light, back-pressure explores "too many" paths between the source and destination). Our delay problem, however, stems from the intermittent connectivity nature, and thus requires different algorithmic techniques.

A cluster-based back-pressure algorithm has been first studied in [8] to reduce the number of queues in the context of traditional networks. However, the algorithm as proposed in [8] in general does not separate between the fast intra-cluster time scale and the slow inter-cluster timescales due to intermittently connected mobile carriers, thus leading to potentially large queue lengths at all nodes along a path. Our proposed algorithm (BP+SR) in this paper explicitly separates the two time scales by choosing *both* a source and destination gateway using source routing, and by explicitly forming an *overlay network* with potentially multiple "stepping-stones" at various clusters' gateway nodes. This explicit separation is the key property that leads to much smaller buffer usage, and hence smaller end-to-end delays.

A number of DTN routing protocols (using carrier nodes) have been proposed. The authors in [12] categorize them as protocols that utilize controlled and intelligent flooding [12], [13], [14], [15], [16], and protocols that operate by learning the intermittently-connected topology behavior and routing using this knowledge [17], [18], [19]. A comprehensive survey of these algorithms is available in [12]. Lastly, there are studies on DTN routing that are similar to back-pressure routing (gradient-based in [20], or the utility-based in [21]). In general however, to the best of our knowledge, the studies in DTNs do not address throughput-optimality (stabilized routing) or the problem of long delays.

II. System Model

A. Network model

In this section, we introduce the basic system model. We consider an intermittently connected network consisting of multiple clusters. Each cluster $\mathcal{C}_i$ is represented by a graph $\mathcal{G}_{\mathcal{C}_i} = (\mathcal{N}_{\mathcal{C}_i}, \mathcal{L}_{\mathcal{C}_i})$, where $\mathcal{N}_{\mathcal{C}_i}$ is the set of nodes in $\mathcal{C}_i$ and $\mathcal{L}_{\mathcal{C}_i}$ is the set of links. Let $\mathcal{C}(n)$ denote the cluster to which node n belongs. The clusters are geographically separated and two nodes in distinct clusters cannot communicate with each other directly.

The clusters are connected by a set of mobile carrier nodes $\mathcal{M}$, which move around to carry packets from one cluster to another. For each cluster, the set of nodes that can communicate with the mobiles is fixed. These nodes are named as gateways as shown in Figure 1. Those nodes that cannot communicate with the mobiles directly are named as internal nodes. Let $\mathcal{I}_{\mathcal{C}_i}$ and $\mathcal{H}_{\mathcal{C}_i}$ denote the set of internal nodes and gateways in $\mathcal{C}_i$, respectively.

We use $g(i,j)$ to denote gateway j in cluster $\mathcal{C}_i$. To simplify the notations, we assume that gateways $g(\cdot,j)$ have the access to mobile $m(j)$ only. The mobiles change gateways every T time slots, which is called a super time slot. We let τ denote the τ^{th} super time slot.

We assume that the mobilities follow Markov processes. Given that mobile $m(j)$ is at gateway $g(i_1,j)$ at the beginning of super time slot τ, the probability that it moves to gateway $g(i_2,j)$ at the beginning of the super time slot $\tau+1$ is $\mathbb{P}\left(m(j)-g(i_2,j) \text{ in } \tau+1 | m(j)-g(i_1,j) \text{ in } \tau\right) = \mathbf{P}_{m(j)}(i_1,i_2)$ where "$-$" means that the mobile and the gateway are in contact. Let $\mathbf{P}_{m(j)}$ denote the transition probability matrix of mobile $m(j)$ and let $\pi_{m(j)}$ be the corresponding

[1]We expect the scaling behavior of end-to-end delay to be similar to that of end-to-end buffer usage. This is because typically the scheduling delay (due to local contention for the wireless air-interface) at each node will not scale with N_c, the cluster size. Thus for the networks we consider, the scaling behavior of the end-to-end delay (consisting of queueing, scheduling and carrier transport delays along the path) will have the dominant term that corresponds to the queueing delay. This intuition is borne out through simulations in section V.

stationary distribution. The Markov chains are assumed to be aperiodic and irreducible. The assumption that $g(\cdot, j)$ only have access to $m(j)$ is not necessary; we make this assumption to simplify our notations.

B. Traffic model

A traffic flow is defined by its source and destination. We assume that the sources and destinations are all internal nodes. If the source and the destination lie in the same cluster, then the traffic flow is called an *intra-cluster* traffic, and the intra-cluster traffic can be routed only within the cluster. If the source and the destination lie in different clusters, the traffic flow is called an *inter-cluster* traffic. We let $[s, d]$ denote the flow from s to d, $\mathcal{F}$ denote the set of all flows, and $\mathcal{F}_{\text{inter}}$ and $\mathcal{F}_{\text{intra}}$ be the sets of all inter- and intra-cluster flows, respectively. We assume deterministic arrivals in this paper. Finally we let x_s^d be the number of packets source s generates per time slot, $\mathbf{x} = \{x_s^d : [s, d] \in \mathcal{F}\}$, and $\mathbf{x}_{\text{intra}(\mathcal{C})}$ be the set of intra-cluster traffic rates in cluster $\mathcal{C}$.

Note that all inter-cluster traffic flows must be forwarded to the gateways in source clusters, then carried over to the gateways in destination clusters via the mobile nodes before reaching their destinations.

C. Communication model

Let $\mu_{(n_1,n_2)}[t]$ denote the transmission rate (packets/time slot) of link (n_1, n_2) at time t, and $\vec{\mu}_{\mathcal{C}_i}[t] = \{\mu_{(n_1,n_2)}[t],\ (n_1, n_2) \in \mathcal{L}_{\mathcal{C}_i}\}$. Let $\Gamma_{\mathcal{C}_i}$ be the convex hull of the set of all feasible transmission rates in cluster $\mathcal{C}_i$. We note that in general, $\vec{\mu}_{\mathcal{C}_i}[t]$ and $\Gamma_{\mathcal{C}_i}$ depend on the interference model used for cluster $\mathcal{C}_i$.

We assume that a mobile and a gateway can send R packets to each other per contact. We assume that the transmissions between mobiles and gateways do not cause interference to other transmissions.

III. Two-level Back-Pressure Algorithm

In this section, we introduce our two-level back-pressure routing algorithm.

A. Queueing architecture

In our algorithm, the network maintains two types of queues. The first type, referred to as type-I, will be denoted by q, and the second type, type-II, will be denoted by u.

Any internal node n_1 maintains a separate type-II queue $u_{n_1}^g$ for each gateway g in the same cluster, and maintains a separate type-I queue $q_{n_1}^{n_2}$ for each node n_2 in the same cluster. A gateway g_1 maintains a separate type-II queue $u_{g_1}^{g_2}$ for each of other gateways g_2 in the network. For each node n in the same cluster, gateway g maintains both a type-I queue and a type-II queue for node n. (The difference between the two types of queues will become clear in the algorithm description.) A mobile m maintains a separate type-II queue u_m^g for each gateway g in the network. We use $q_a^b[t]$ ($u_a^b[\tau]$) to denote the length of the type-I (type-II) queue maintained by node a for node b during time slot t (super time slot τ).

Note that $u_n^n = 0$ and $q_n^n = 0$ at all times for any node n.

As a consequence of our routing algorithm (which we will describe shortly), type-II queues are operated at $\Theta(T)$ time scale, whereas the type-I queues are operated at $\Theta(1)$ time scale. The difference in time scale results in the type-II queue length being $\Theta(T)$ times larger than type-I queue length. Under the original back-pressure routing algorithm, all nodes would have queues of length $\Theta(T)$. However, under our routing algorithm, only a few nodes have type-II queues; this will result in shorter end-to-end delays.

B. Two-level back-pressure with source-routing algorithm

Before we describe our routing algorithm in a detail, we use an example to provide an overview. Consider an inter-cluster flow from the source s to the destination d. For new incoming packets, the source s first chooses the source and destination gateways g_s and g_d these packets should be routed through. g_s lies in the same cluster as the source and g_d lies in the same cluster as the destination; g_s and g_d are the points of exit and entry, respectively.

The packets that are assigned with the source gateway and destination gateway pair (g_s, g_d) are first deposited into type-II queue $u_s^{g_s}$ and routed to the gateway g_s by intra-cluster back-pressure routing. The back-pressure routing inside the clusters is performed with type-I queue lengths. See Part II of our algorithm description.

Once the packets reach g_s, they are transferred to type-II queue $u_{g_s}^{g_d}$ and relayed to mobiles, which carry the packets to the destination cluster or some intermediate clusters. If the packets are carried to an intermediate cluster, the packets will be transferred to type-II queues for g_d, and may be routed to a gateway that is in the same cluster and has smaller backlogs. After that, the packets are again transferred to type-II queues, and relayed to mobiles, which may carry the packets to another cluster.

After the packets reach their destination gateways, they are transferred to type-I queue $q_{g_d}^d$ and forwarded to the destination using the intra-cluster back-pressure routing. See Part IV.

We now introduce our routing algorithm.

Part I: Selecting source and destination gateways

At the beginning of super time slot τ, the inter-cluster traffic source s picks the source and destination gateways $g_s^*[\tau]$ and $g_d^*[\tau]$ such that

$$(g_s^*[\tau], g_d^*[\tau]) \in \arg \min_{\substack{g_s \in \mathcal{H}_{\mathcal{C}(s)} \\ g_d \in \mathcal{H}_{\mathcal{C}(d)}}} \left(u_s^{g_s}[\tau] + u_{g_s}^{g_d}[\tau] + u_{g_d}^d[\tau]\right).$$

This route selection is done at the beginning of each super time slot (i.e., every T time slots).

Part II: Traffic control at the source nodes

- For an **inter-cluster** flow $[s, d]$, the source node s deposits the new arrived packets into queue $u_s^{g_s^*[\tau]}$ during time slot $t \in [\tau T, (\tau + 1)T)$. The identities of the source gateway $g_s^*[\tau]$ and the destination gateway $g_d^*[\tau]$ are recorded in the headers of the packets.
- For an **intra-cluster** flow $[s, d]$, the source node s deposits the new arrived packets into queue q_s^d.

- Define $\theta_s^{g_s}[\tau] = u_s^{g_s}[\tau]/K_s$, where $K_s = T/|\mathcal{C}(s)|$. ($|\mathcal{C}|$ is the number of nodes in cluster $\mathcal{C}$.) Consider the queues associated with gateway $g_s \in \mathcal{H}_{\mathcal{C}(s)}$. If $\theta_s^{g_s}[\tau] > q_s^{g_s}[t]$ at time $t \in [\tau T, (\tau+1)T)$, η packets are transferred from queue $u_s^{g_s}$ to queue $q_s^{g_s}$ at the beginning of time slot t. (η is some positive value greater than the largest transmission rate out of any node inside a cluster.)

Part III: Traffic control at the gateway nodes

For each gateway g_2 belonging to the same cluster, gateway g_1 computes the following quantity at the beginning of each super time slot: $l_{g_1,g_2}[\tau] \in \arg\max_{l \in (\bigcup_{\mathcal{C}} \mathcal{H}_{\mathcal{C}})} (u_{g_1}^l[\tau] - u_{g_2}^l[\tau])$. Define $\theta_{g_1}^{g_2}[\tau] = \frac{(u_{g_1}^{l_{g_1,g_2}[\tau]}[\tau] - u_{g_2}^{l_{g_1,g_2}[\tau]}[\tau])}{K_{g_1}}$ where $K_{g_1} = T/|\mathcal{C}(g_1)|$. At each time slot $t \in [\tau T, (\tau+1)T)$, g_1 transfers η packets from $u_{g_1}^{l_{g_1,g_2}[\tau]}$ to $q_{g_1}^{g_2}$ if $\theta_{g_1}^{g_2}[\tau] > q_{g_1}^{g_2}[t]$. The next destination of the transferred packets is temporarily set to g_2; when g_2 receives those packets, they are inserted into $u_{g_2}^{g_d}$, where g_d can be found in the packet headers.

Part IV: Traffic control at the destination gateways

When the packets arrive at their destination gateways, they are deposited into queue $u_{g_d}^d$. Let $\theta_{g_d}^d[\tau] = u_{g_d}^d[\tau]/K_{g_d}$, where $K_{g_d} = T/|\mathcal{C}(g_d)|$. In each time slot $t \in [\tau T, (\tau+1)T)$, η packets are transferred from $u_{g_d}^d$ to $q_{g_d}^d$ if $\theta_{g_d}^d[\tau] > q_{g_d}^d[t]$.

Part V: Routing and scheduling within a cluster

In each time slot t, each cluster $\mathcal{C}$ computes $\vec{\mu}_{\mathcal{C}}[t]$ such that

$$\vec{\mu}_{\mathcal{C}}[t] = \arg\max_{\vec{\mu} \in \Gamma_{\mathcal{C}}} \left\{ \sum_{(m,n) \in \mathcal{L}_{\mathcal{C}}} \mu_{(m,n)} P_{(m,n)}[t] \right\}$$

where $P_{(m,n)}[t] = q_m^{j_{(m,n)}[t]}[t] - q_n^{j_{(m,n)}[t]}[t]$ and $j_{(m,n)}[t] = \arg\max_j \left\{ q_m^j[t] - q_n^j[t] \right\}$. After the computation, node m transmits $\mu_{(m,n)}[t]$ packets out of queue $j_{(m,n)}[t]$ to node n in time slot t.

Part VI: Routing between gateways and mobiles

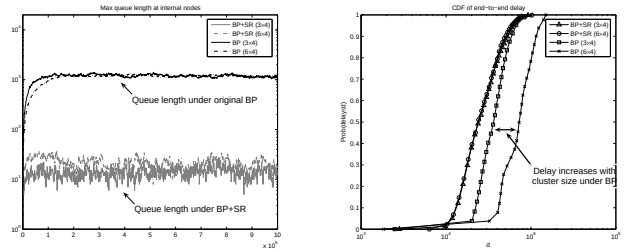
At the beginning of each super time slot τ, the mobile m and the gateway g that are in contact compute the following: $j_{(m,g)}[\tau] = \arg\max_{j \text{ gateway}} \left\{ u_m^j[\tau] - u_g^j[\tau] \right\}$ and $j_{(g,m)}[\tau] = \arg\max_{j \text{ gateway}} \left\{ u_g^j[\tau] - u_m^j[\tau] \right\}$. Afterwards, m transmits R packets from the queue $u_m^{j_{(m,g)}[\tau]}$ to g, and the gateway g transmits R packets from the queue $u_g^{j_{(g,m)}[\tau]}$ to m, maximizing $R\left(P_{(m,g)}[\tau] + P_{(g,m)}[\tau]\right)$.

C. Overlay network

We note that the gateways and mobiles form a virtual overlay network. An inter-cluster packet first enters the virtual network at the selected source gateway, and then leaves the overlay network at the selected destination gateway. Two nodes of the overlay network are connected by a cluster network instead of a single physical link. The transmissions between two neighboring nodes in the overlay network are accomplished by intra-cluster backpressure routing/scheduling.

Our BP+SR includes routing and scheduling within each cluster and in the overlay network, and so is a two-level algorithm.

Further comments on the overlay network can be found in section VIII-A of our technical report [22].



(a) Max queue length at internal node (type I queues for BP+SR)

(b) CDF of end-to-end delay for inter-cluster traffic

Fig. 2. Simulation results. Observe that the end-to-end delay for inter-cluster traffic does not increase with the size of the clusters.

IV. Performance of BP+SR algorithm

A. Throughput optimality

Theorem 1: Fix any $\delta > 0$. Given external arrival $\mathbf{x}$ such that $(1+\delta+\epsilon)\mathbf{x}$ is supportable for some $\epsilon > 0$ (i.e., there exists an algorithm that can stabilize the network with traffic load $(1+\delta+\epsilon)\mathbf{x}$), all queues are bounded under the BP+SR algorithm.

Proof: The proof is provided in the appendix of [22]. ■

B. Buffer usage and delay

We now compare the delay under the original back-pressure routing and our proposed algorithm, and have the following observation (a detailed analysis of delay and the number of queues required for our algorithm can be found in [22]):

Observation: Consider a network consisting of a finite number of $\sqrt{N_c} \times \sqrt{N_c}$ grid clusters, each containing N_c nodes. Assume that the traffic load is strictly within the network throughput region and $N_c^2 = O(T)$; assume that all gateways across the network have the same stationary distribution $\left(\pi_{m(j)}\right)_{g(i,j)} = \pi, \forall\, i,j$. Then the average end-to-end delay of inter-cluster traffic (and the average end-to-end buffer usage) is $\Theta(\pi^{-1}T)$ for BP+SR and $\Theta(\pi^{-1}N_cT)$ for BP.

C. Queue maintenance

Consider a network consisting of N_s clusters, each having N_c nodes with N_g gateways (in general, we assume $N_c >> N_s, N_g$). The number of queues maintained by a node under BP is proportional to the total number of traffic destinations in the network, which can be as large as the total number of nodes in the network ($\approx N_s \times N_c$).

Under BP+SR, each internal node maintains a queue for each node in its own cluster only, for the maximum total of $N_c + N_g$ (type I + type II) queues. Each gateway maintains a type II queue for other gateways in the network and a type I queue for each internal node in the same cluster, for the maximum total of $N_sN_g + N_c$ queues. The number of queues maintained by a mobile is N_sN_g.

Therefore, the maximum number of queues at a node under BP+SR is $N_sN_g + N_c$, which is substantially fewer than under BP (N_sN_c).

V. Simulations

We consider two networks: the first network consists of clusters of 12 nodes (3×4) and the second network consists of clusters of 24 nodes (6×4). Each network has three clusters. Figure 1 is the first network we simulate. There are three pairs of inter-cluster traffic sources/sinks, labeled with 1, 2 and 3, i.e. the two nodes labeled with 1 communicate with each other and likewise for 2's and 3's. The inter-cluster traffic sources generate data at a rate of 0.4 pkts/time slot. Mobile $m(1)$ comes into contact with gateways $g(i,1)$, $i = 1,2,3$ only; similarly for $m(2)$. T is set to 1000. In the simulation, we have randomly generated intra-cluster traffic, such that the combined intra- and inter-cluster traffics utilize all internal links at 90% of their capacity. Each internal link has a capacity of 1 pkt/time slot. η is set to 10.

The probability that mobile $m(1)$ goes to gateway $g((i+1 \bmod 3),1)$ when it is at gateway $g(i,1)$ is 0.8 (gateway $g(i,1)$ belongs to cluster i); the probabilities that it stays at $g(i,1)$ or goes to $g((i-1 \bmod 3),1)$ both equal to 0.1. The probability that mobile $m(2)$ goes to $g((i-1 \bmod 3),2)$ is 0.8; the probabilities that it stay or goes to $g((i+1 \bmod 3),2)$ both equal to 0.1.

The number of packet transferred per contact between a mobile and a gateway is 1500 pkts/contact ($R = 1500$ pkts/contact).

We compare BP+SR to BP, as BP (and its variants) is the only other throughput optimal routing algorithm. In Figure 2, we illustrate the maximum values of various types of queues. Figure 2(a) shows the evolution of the longest queue under BP and the evolution of the longest type-I queue under BP+SR. We can see that the longest type-I queue is substantially smaller than the longest queue under BP.

In [22], we have figures that show the maximum values of different type-II queues. It is shown that those type-II queues are of order $\Theta(T)$. In our simulations, we found that under BP *each node in the network* had six queues of order $\Theta(T)$, corresponding to the six different inter-cluster traffic destinations. Under BP+SR, each inter-cluster source had two queues of order $\Theta(T)$, corresponding to the two gateways in each cluster. The other nodes with $\Theta(T)$ queues were gateways and mobiles, as can be seen in the figures above.

In Figure 2(b), we show the CDF of the total end-to-end packet delays for inter-cluster traffic. The average end-to-end delay under BP+SR is roughly 28,000 time slots in both 3×4 and 6×4 cases; the delay under BP is 38,000 time slots in 3×4 case and 78,000 in 6×4 case. As discussed in section IV-B, the delay doubled under BP as the cluster size doubled. Delays for intra-cluster flows stayed the same under BP+SR; BP+SR and BP both showed short delays for intra-cluster flows, as expected.

VI. Conclusion

In this paper, we have developed the two-level BP+SR for intermittently connected networks that provides two main benefits over the original back-pressure routing algorithm: 1) our proposed algorithm reduced the number of queues required at each node; 2) it reduced the size of the queues, thereby reducing the end-to-end delay.

VII. Acknowledgment

This work is partially supported by the DTRA grants HDTRA1-08-1-0016, HDTRA1-09-1-0055, NSF grants CNS-0347400, CNS-0721380, the U.S. Department of Defense, and the DARPA ITMANET program.

References

[1] A. Pentland, R. Fletcher, and A. Hasson, "DakNet: Rethinking connectivity in developing nations," *IEEE Computer*, vol. 37, pp. 78–83, 2004.
[2] Delay tolerant networking research group, http://www.dtnrg.org.
[3] K. Fall, "A delay-tolerant network architecture for challenged Internets," in *Proc. of SIGCOMM*, 2003.
[4] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1949, December 1992.
[5] A. L. Stolyar, "Maximizing queueing network utility subject to stability: greedy primal-dual algorithm," *Queueing Systems*, vol. 50, pp. 401–457, 2005.
[6] M. J. Neely, E. Modiano, and C. Li, "Fairness and optimal stochastic control for heterogeneous networks," in *Proc. of IEEE INFOCOM*, 2005.
[7] A. Eryilmaz and R. Srikant, "Joint congestion control, routing, and MAC for stability and fairness in wireless networks," *IEEE JSAC*, vol. 24, pp. 1514–1524, 2006.
[8] L. Ying, R. Srikant, and D. Towsley, "Cluster-based back-pressure routing algorithm," in *Proceedings of IEEE INFOCOM*, Phoenix, AZ, April 2008.
[9] L. Bui, R. Srikant, and A. Stoylar, "Novel architectures and algorithms for delay reduction in back-pressure scheduling and routing," in *Proc. of IEEE INFOCOM*, 2009.
[10] 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.
[11] M. J. Neely, "Dynamic power allocation and routing for satellite and wireless networks with time varying channels," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.
[12] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. of ACM SIGCOMM*, 2007.
[13] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.
[14] J. Burgess, B. Gallagher, D. Jense, and B. N. Levine, "MaxProp: Routing for vehicle-based disruption-tolerant networks," in *Proc. IEEE INFOCOM*, 2006.
[15] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Department of Computer Science, Duke University, Technical Report CS-2000-06, April 2000.
[16] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks," in *Proc. of Fourth ACM International Symposium on Mobile and Ad Hoc Networking and Computing*, 2003.
[17] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proc. of SIGCOMM*, 2004.
[18] E. Jones, L. Li, and P. Ward, "Practical routing in delay-tolerant networks," in *Proc. of SIGCOMM*, 2005.
[19] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive Wi-Fi mobility data," in *Proc. of IEEE INFOCOM*, 2004.
[20] R. D. Poor, "Gradient routing in ad hoc networks," 2000, MIT Media Laboratory.
[21] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Single-copy routing in intermittently connected mobile networks," in *IEEE SECON*, 2004.
[22] J. Ryu, L. Ying, and S. Shakkottai, "Back-pressure routing for intermittently connected networks," WNCG, The University of Texas at Austin, Technical Report, 2009.