Back-Pressure Routing and Rate Control for ICNs

Jung Ryu, Vidur Bhargava, Nick Paine, and Sanjay Shakkottai
Wireless Networking and Communications Group (WNCG)
Department of Electrical and Computer Engineering
The University of Texas at Austin
{jung.ryu, shakkott}@mail.utexas.edu and {vidurbhargava, nick.a.paine}@gmail.com

ABSTRACT

We study a network composed of multiple clusters of wireless nodes. Within each cluster, nodes can communicate directly using the wireless links; however, these clusters are far away such that direct communication between the clusters is impossible except through "mobile" contact nodes. These mobile contact nodes are data carriers that shuffle between clusters and transport data from source to destination clusters. There are several applications of our network model (e.g., clusters of mobile soldiers connected via unmanned aerial vehicles).

At the same time, much interest has been garnered by cross-layer design for wireless networks in order to improve efficiency and better allocate resources. In this paper, we focus on queue based cross-layer technique known as back-pressure algorithm. The algorithm is known to be throughput optimal, as well as resilient to disruptions in the network, making it an ideal place to start when designing communication protocols for our intermittently connected network.

In this paper, we design a back-pressure routing/rate control algorithm for intermittently connected networks (ICNs). We implement a modified back-pressure routing algorithm on a 16-node experimental test bed, discuss some of the issues regarding design and implementation, and present our experimental results.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: Distributed networks; Wireless communication; Packet-switching networks

General Terms: Algorithms, Experimentation

Keywords: Intermittently connected networks, back-pressure routing

1. INTRODUCTION

There recently has been much interest in intermittently connected networks. Practical use of such networks include connecting rural villages in under-developed countries via satellites or network of buses to ferry data; or in military settings, where clusters of autonomous combat robots rely on reconnaissance aerial vehicles flying over-head in order to exchange battle field information and to coordinate attacks. In either case, building a communication network might take too much time as these combat units must be deployed at moments notice, and wireless connections might be susceptible to enemy jamming or the ranges might not be large enough.

In networks with extremely high delay and intermittent connectivity, currently existing communication protocols suffer serious performance degradation. Routing information, for example, might not be stable since these networks use mobile nodes to provide some level of connectivity – as mobiles move in and out of networks, connections come and go. Further, protocols that depend on congestion signals from the network might not work properly if the signals arrive with high delays. Current research is expanding to incorporate mobile nodes and connections that have high delays into traditional networks in order to increase capacity, reliability and functionality.

In this paper, we consider a network of clusters of nodes connected via "mobile" nodes (see Figure 1 for an example). We assume that these mobiles move in periodic patterns. Internally, each cluster has many nodes connected via a (multi-hop) wireless network. Each cluster has at least one *gateway node.* (We will call the other nodes in a cluster internal nodes.) These gateways are the designated representatives of the clusters, and they are the only ones able to communicate with the mobiles – traffic from one cluster to another cluster (inter-cluster traffic) must be funneled through the gateways, both in the source cluster and in the destination cluster. The mobiles and gateways exchange packets (pick-ups and drop-offs) on contact. Each contact is made over a high capacity link and is long enough for a large quantity of data to be exchanged. The mobiles then move between clusters, and on contact with a gateway in the destination cluster, packet drop-offs are made.

A concrete example of the intermittently connected networks (ICNs) we consider in our paper is as follows: there are clusters of soldiers that are geographically separated in the battle field. The soldiers in the same cluster are connected wirelessly. However, the soldiers in different clusters must rely on unmanned aerial vehicles (UAVs) to provide connectivity because the range of the wireless devices is not large enough. These UAVs fly between clusters in some (approximately) pre-specified pattern (more generally, any ergodic mobility pattern) for the purpose of connecting different clusters. The UAVs can take seconds or minutes to fly from one cluster to another, and when a UAV visits a cluster, it can pick up or drop off data to one of the designated soldiers with a special device that can communicate with the UAVs - the special device must have the capability to take advantage of the brief, but high capacity contact that UAVs provide. Within a cluster, the data is routed to and from the designated soldiers via the internal wireless network.

A key challenge in the network above is the fact that *intermittently connected networks have several time-scales of link variability.* For instance, wireless communication between soldiers within the same cluster is likely to occur at a time-scale several orders of magnitude faster than communication across clusters (which needs to use the mobile carriers, namely, UAVs). In this context, there are essentially two time-scales: *(a)* within a cluster, where wireless links are formed in an order of tens of milliseconds, and *(b)* across clusters where the time-scale could be tens of seconds, to minutes. To communicate from one node to another node in the same cluster poses no significant problem – one can use existing protocols such as TCP. However, for two nodes in two different clusters to communicate, they must use the mobile nodes, as these mobiles move between clusters to physically transport data. Hence, the mobile communication time scale is many, many times greater than the electronic communication time scale. Any communication protocol that relies on fast feed-back (in the order of milliseconds to tens/hundreds of milliseconds) incurs severe performance degradation. The mobiles may be able to transport a large quantity of data (of the order of mega or giga bytes) in one "move," but to move from one cluster in one part of a network to another part still takes time (of the order of seconds to minutes).

The design and development of communication protocols for intermittently connected networks, therefore, must start with an algorithm with as few assumptions about the underlying network structure as possible. The back-pressure routing algorithm [24] was introduced nearly two decades ago by Tassiulas and Ephremides with only modest assumptions about the stability of links, their "anytime" availability, or feasibility of fast feed-back mechanism; yet remarkably it is throughput optimal (throughput performance achieved using any other routing algorithm can be obtained using the back-pressure algorithm [24]) as well as resilient to changes in the network. Over the years, there has been much continued effort to further develop back-pressure type algorithms to include congestion control and to deal with state-space explosion and delay characteristics [14, 22, 5, 15, 1, 30, 29, 13, 27]. However, the traditional back-pressure algorithm will lead to extremely poor performance in intermittently connected networks. This is because the performance of the back-pressure algorithm in the heterogeneous connectivity setting of the ICNs is governed by the link with the "poorest delay performance", and this "poorly" performing link can "poison" the performance of the other links in the network. However, we do believe that the back-pressure algorithm is a reasonable starting point for developing rate control/routing protocols for intermittently connected networks. In this paper, we design, implement and evaluate the performance of a rate control mechanism along with a back-pressure like routing/scheduling algorithm specially tailored for ICNs. As we shall see later, by modifying the back-pressure algorithm to take into account the different time-scales, we can achieve very similar performance to that predicted by theory.

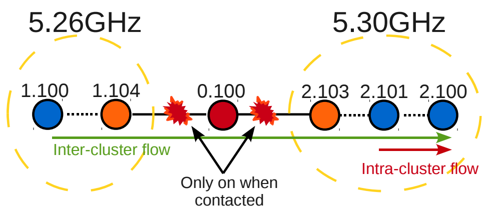


Figure 1: A simple, intermittently connected line network: There are two wireless clusters, each cluster operating at a different frequency (thus clusters do not interfere/directly communicate with each other), and with one mobile node connecting these clusters.

The back-pressure routing is a per-packet dynamic routing algorithm, where per-destination queues are maintained at each node. At each time, a packet currently at a node n (and destined to node d) is forwarded to a neighbor m if the backlog at node m (for destination d) is smaller than that at node n. Thus, the packets create backlog gradients over the network, and each packet sent out afterwards reaches d by following the positive gradient difference over each hop. An analogy that helps in understanding is that of water flow that reaches the sink by seeking the largest drop in elevation as it flows – paths that do not reach the sink will end up accumulating pool of water until no more water can flow along those paths. The back-pressure algorithm is throughput optimal, and resilient to the changes in the network and forms the basis of our implementation, see Section 3 for details.

Another solution that can be used in intermittently connected networks is to use replication based strategy. However, in the scenario that we consider in this paper, we can exploit the periodic mobile movement pattern to obtain throughput optimal performance of back-pressure routing (note however that the patterns need not be known in advance for our algorithm to work). Replication-based algorithms such as epidemic routing for DTNs are more suitable for networks with random and unpredictable mobile movements; such algorithms result in lower throughput since multiple copies of a piece of data need to be forwarded and stored (and therefore not throughput optimal). However, in certain scenarios, one is not necessarily interested in throughput optimality and just having connectivity is satisfactory. For example, if one is interested in a single file transfer in a general delay-tolerant setting and whether file is completely received is the only concern, the replication-based strategies are potentially a better choice than our BP-based algorithm.

2. MOTIVATION: DIFFICULTIES WITH CLASSICAL BACK-PRESSURE

Consider a simple, intermittently connected line network as shown in Figure 1. We have two clusters, and each cluster is on a different 802.11a channel (5.26 GHz and 5.30 GHz, respectively). We have two gateways (1.104 and 2.103) representing the 5.26GHz and 5.30GHz clusters, respectively. Between these two clusters, we have a "mobile" contact node 0.100 that moves from one cluster to the other every ten seconds. On contact, the mobile and the gateways (the designated nodes in the cluster that can communicate with the mobiles) can exchange 12MB of data (6MB in each direction). Finally, there are two flows; an inter-cluster flow originating from 1.100 and an intra-cluster flow originating from 2.101, and both flows are destined for 2.100.

In this network, routing is straightforward. But the question here is:
What is the rate at which these flows can transmit data? An even more basic question is: *Can these flows attain high and sustainable throughput*[1]*, provided that the link capacity between the mobiles and gateways is high enough (albeit with extreme delays)? How close can we get to the maximum throughput allowed by the network? Can we obtain utility-maximizing rate allocation over an ICN?*

The answer to this is clearly a negative, if TCP is used for rate control, and we shall see that even with traditional back-pressure [24, 5] algorithms that have a theoretical guarantee that the above is possible, in a practical setting, the answer still seems to be negative!

To put the above statement in context, we know back-pressure (BP) routing/rate control algorithm is throughput optimal [24], meaning that if any routing/rate control algorithm can give us certain throughput performance, so can back-pressure algorithm. *Contra-positively, if back-pressure algorithm can not, no other algorithm can do so.* Further, a rate controller based on utility maximization can be added to this framework [5] that is theoretically utility maximizing, and it chooses rates that (averaged over a long time-scale) lead to high and sustainable throughput corresponding to the rates determined via an optimization problem [5].

To see the performance of a traditional BP based routing/rate control algorithm, let us look at Figure 2(a), where we plot the rate trace of the two (inter- and intra-cluster) flows. (Each source uses the BP congestion algorithm [14, 22, 5, 15] which we will describe later). In these plots, we can see that the inter-cluster traffic performs very poorly, even though the mobile-gateway contact has enough capacity. The reason is simple – the BP congestion control uses the local queue length as a congestion signal, and between two successive contacts that can be seconds or minutes apart, the inter-cluster packets have no where to go, and the queues build up to the point that the inter-cluster source mistakenly believes that the network has low-capacity (and low-delay) links, whereas we have high-capacity (but high-delay) links. Because of this, the inter-cluster source is not able to fill the buffer with enough packets at the gateway to fully utilize the contacts (see Figure 2(a)).

Importantly, this rate achieved by the inter-cluster traffic is much lower than that predicted by the theory (the theory predicts that intra-cluster rate is $\approx$ 200KBps, and the inter-cluster rate $\approx$ 100KBps). This is because the theoretical results hold only when the utilities of users are scaled down by a large constant – this is to (intuitively) enable **all** queues in the network to build up to a large enough value in order to "dilute" the effects of the "burstiness" of the intermittently connected link. In the measurements here, the utilities have not been globally scaled to such a sufficiently large value, thus the inter-cluster rate is exceedingly low.

Furthermore, even if the inter-cluster source is aware of the presence of these intermittent mobile-gateway links and therefore can transmit at the correct rate (i.e., a genie computes the rate and tells this to the source), the problem can manifest itself in another way.

[1]By "high" we mean close to the maximum throughput possible, and by "sustainable" we mean stochastically stable.

Consider the same network as in Figure 1, but we just have one inter-cluster flow from 2.100 to 1.100 (now in the opposite direction to what is shown in the Figure). We run the traditional BP algorithm (with no rate controller, as the genie has solved this problem) and fix the source rate at 200KBps. In this case, there is large backlog that builds up not only at node 2.103 (the intermittently connected node), but also at node 2.101 which is an "internal" node (see Figure 2(c)). Note that this happens, even though node 2.101 is *nicely connected* and has no intermittent connectivity issues. This is what we alluded to in the introduction where the intermittently connected links "poisoned" the performance at all other links. In this case, since the backlog at the intermittently connected node 2.103 is necessarily large, node 2.101 can "push" packets into 2.103 only if its own backlog is larger; this effect thus makes all queues in the 5.30 GHz cluster to be large. The problem of large queues everywhere in this second genie-assisted setup is exacerbated as the network size increases. In particular, in our second, larger scale experiments with multiples clusters (see Figure 12), we were even unable to complete a single run of the experiment with traditional BP due to exceedingly long queues within all clusters.

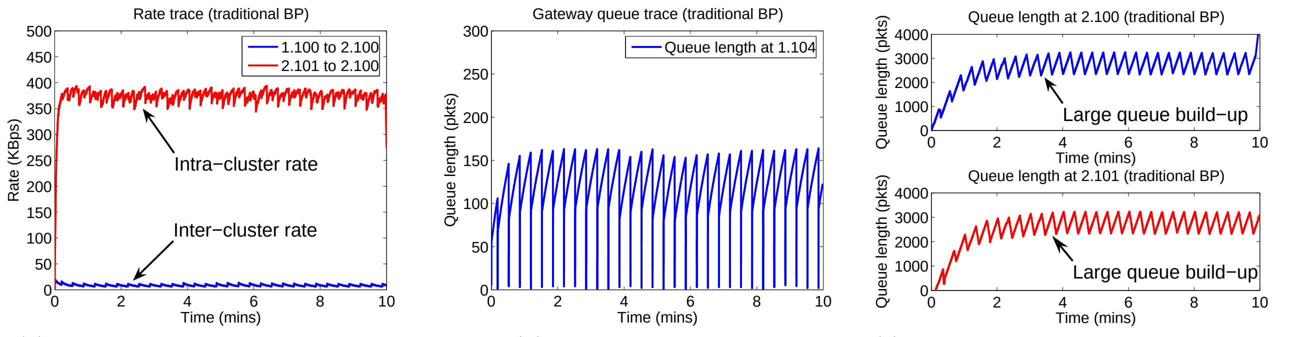
2.1 Main Contributions

In this paper, we designed, implemented and empirically studied the performance of a modified back-pressure algorithm that has been coupled with a utility based rate controller for an intermittently connected network.

1. We built a testbed for evaluating rate control, routing and scheduling over intermittently connected networks. This testbed consists of Linux-based wireless (802.11a) nodes, with a modified MadWifi device driver and a programmable router (using Click [9]). These nodes are organized into several clusters, with intermittent connectivity across these clusters provided by "carrier" nodes. These carrier nodes connect intermittently to various clusters via an Ethernet switch with time-varying connectivity (this switch emulates mobility pattern of the carrier, via a predetermined switching schedule).

2. Using this testbed, we first showed that the traditional back-pressure algorithms coupled with a utility function based rate controller is not suitable for intermittently connected networks, and leads to a large divergence between theoretically predicted rates and actual measurements (shown for a line network). This is because of the time-scale mismatch between the intra-cluster and inter-cluster communications.

3. We have then studied a modified back-pressure (BP) algorithm, that along with a utility-based rate controller loosely decouples the time-scales at the inter-cluster and intra-cluster levels. With this time-scale decoupling, (and unlike the traditional BP algorithm) we now observe a much better match between theory and measurements for a line network. We also present measurement results for a larger sized network (three clusters with two carrier nodes providing intermittent connectivity). A key advantage of this modified BP algorithm is that it maintains large queues only at nodes which are intermittently connected; at all other nodes,



(a) Rate traces under traditional BP

(b) Queue traces at 1.104

(c) Queue traces at 2.100 and 2.101 with fixed 200KBps source rate

Figure 2: Problems with traditional BP in intermittently connected networks: either low inter-cluster rate or large queues at every node

the queue sizes remain small. In addition, the nodes that are not intermittently connected are not aware of the presence of the intermittent links.

4. Finally, we present a practical implementation of shadow queues developed by the authors in [3]. Using our implementation, we observe that we have a nice trade-off between controlling the end-to-end inter-cluster delay and the network capacity utilization.

3. BACK-PRESSURE ALGORITHM

We begin with a mathematical overview of the back-pressure (BP) algorithm [24]. The time is slotted, with t denoting t^{th} time slot. We represent the network by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ corresponds to the set of nodes in the network and $\mathcal{L}$ being the collection of links. We denote $\mu_{(n_1,n_2)}[t]$ to be the transmission rate (measured in packets/time-slot) of link $(n_1, n_2) \in \mathcal{L}$ between $n_1, n_2 \in \mathcal{N}$ at time t, and $\vec{\mu}[t] = \{\mu_{(n_1,n_2)}[t],\ (n_1, n_2) \in \mathcal{L}\}$. Finally Γ is the convex hull of the collection of all feasible transmission rates in the network. We observe that $\vec{\mu}[t]$ and Γ could depend on the interference model used for the network. (We also refer to [18] and Section 5 for additional details).

Let $f_{[s,d]}$ denote the flow from s to d. $\mathcal{F}$ denotes the set of all flows. Let $x_{[s,d]}$ be the rate at which s generates data for d, and let $\vec{x} = \{x_{[s,d]}, [s, d] \in \mathcal{F}\}$. Let $\mathcal{C}$ denote the capacity region of the network under Γ.

Each node in the network maintains a queue for every other node in the network. Let $q_i^j[t]$ denote the length of queue for node j maintained at node i; the queue for node i maintained at i is assumed to be zero for all time slots, i.e. $q_i^i[t] = 0\ \forall t$. In each time slot t, each node n obtains queue information from its neighbor m ($(n, m) \in \mathcal{L}$). Define $P^j_{(n,m)}[t] = q^j_n[t] - q^j_m[t]$. Let

$$j_{(n,m)}[t] = \arg\max_j P^j_{(n,m)}[t]. \qquad (1)$$

In each time slot t, the network computes (1) and $\vec{\mu}[t]$ such that

$$\vec{\mu}[t] = \arg\max_{\vec{\mu}\in\Gamma} \left\{ \sum_{(m,n)\in\mathcal{L}} \mu_{(m,n)} P^{j_{(m,n)}[t]}_{(m,n)}[t] \right\}. \qquad (2)$$

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. Note that the maximization problem (sometimes called the MaxWeight problem) eq. (2) can be solved in a distributed way for a wired network, but is a global problem for wireless networks due to the coupled interference constraint and therefore is an NP-hard problem.

The above routing and scheduling algorithm is proven to be throughput optimal [24]; i.e. BP stabilizes all queues in the network if at all possible to do so under any algorithm, and the capacity region under BP is the largest possible. There are also many extensions available that deal with theoretically addressing both the distributed aspects as well as lowering the complexity [1, 3, 30].

3.1 Rate Control and Utility Maximization

More recently, utility maximization has been addressed in a back-pressure framework [14, 22, 5, 15, 1] to address rate control issues.

We use the formulation in [1] to describe the idea here. Each flow $f_{[s,d]}$ originating from s and destined for d has a utility function $U_{f_{[s,d]}}(x_{f_{[s,d]}})$ which is a function of the rate $x_{f_{[s,d]}}$ it is served at. Let s_f and d_f denote the source and the destination of the flow f, respectively. We assume that all utility functions are strictly concave, with continuous derivatives. The utility maximization problem is the following:

$$\max_{\vec{x}\in\mathcal{C}} \sum_{f\in\mathcal{F}} U_{x_f}(x_f). \qquad (3)$$

Let $x_f[t]$ denote the rate at which the flow f is served in time slot t. The rate control algorithm that maximizes (3) is the following. In each time slot t, the source s_f injects $\kappa > 0$ packets into the queue $q^{d_f}_{s_f}$ if and only if

$$U'_f(x_f[t]) - \beta q^{d_f}_{s_f}[t] > 0, \qquad (4)$$

where $\beta > 0$ is a control parameter and U'_f is the first derivative of flow f's utility function. The parameter β controls how close to the optimal rate allocation the system performs, but this comes at the price of longer queues. We also refer to [1] for a discussion on this implementation.

4. IMPLEMENTATION

Our implementation consists of two parts. The first part is the modification of the MadWifi wireless device driver to support differentiated levels of channel access on a frame by frame basis through varying MAC contention parameters such as AIFS and the contention window sizes. The second part is our implementation of the modified back-pressure routing algorithm on the Click Modular Router [9] which utilizes the modified MadWifi to approximately solve the MaxWeight optimization problem (2) without a global knowledge. We describe each part below.

Each node has a Via C7 1GHz processor with up to 1GB of memory and runs Linux 2.6.31. It also had an Atheros 5212 802.11a/b/g wireless card and an Ethernet port. We put all our nodes in the monitor mode and used the modified MadWifi driver.

MAC and PHY: We modified the MadWifi driver so that it supports four hardware queues, with each queue having different AIFS, $CW_{\max}$ and $CW_{\min}$ values shown in Table 1. (When two wireless transmissions contend for access to the same channel, the wireless transmission with smaller MAC parameter values will statistically have more access.) Each hardware queue is given a priority number ranging from 0 to 3. The modified device driver inspects the TOS field of the IP header of a packet, and injects it into the hardware queue with the same priority number as the TOS field. Our modification is very similar to the one in [27], with only minor differences. The differences essentially stem from the fact that we use only four priority levels, whereas [27] uses eight levels. The more substantial difference between our work and that in [27] is in the algorithms for rate control, as well as the implementation at the routing layer. The work in [27] focuses on modifying TCP to work with back-pressure over a traditional wireless network. However, over an intermittently connected network such as ours, TCP breaks, thus requiring a new approach for the rate control. Furthermore, from the implementation perspective, we use the Click router for supporting a variety of routing and rate control algorithms. By separating the MAC contention resolution implementation from the BP routing/rate control algorithm in Click, we are able to study the effects of the two separately.

Routing and Rate Control: We have implemented the traditional as well as the modified back-pressure algorithm (presented in Section 5) in Click Modular Router. Each packet (with 1KB payload) that is sent out is assigned a value between 0 to 3 that is written to the TOS field. The assigned value depends on the queue length difference that the wireless transmission source has with the next-hop destination. Each node broadcasts a beacon on its wireless card every 500msecs. The beacon contains the information about the queues the node maintains. The nodes also use the beacons to discover their neighbors. All data packets received by a node is acknowledged and the ACKs sent to the transmitting node also contains the queue information. Thus, ACKs and beacons are used to calculate eq. (1). The transmitting node retransmits a data packet if an ACK for that packet is not received within 250msecs. The hop-by-hop ACK guarantees that all packets are received correctly by the final destination, and the ACKs are also used to throttle the transmission rate. We did not use any RTS-CTS in our implementation.

A source node inspects the queue for its destination every 5msecs. It runs the back-pressure rate control algorithm in eq. (4) with $\kappa = 3$; i.e., it generates three packets if (4) is positive, and generates zero packets other-wise (let this variable be $NumGen$). Then it uses the following update algorithm to estimate the rate x (in Bps):

$$x = 0.999x + 0.001\frac{\text{packet\_size*NumGen}}{\text{5msecs}}$$

Note that this is simply an exponential filtering of the rate estimate, based on $NumGen$ (we refer to [1] for a discussion of this implementation).

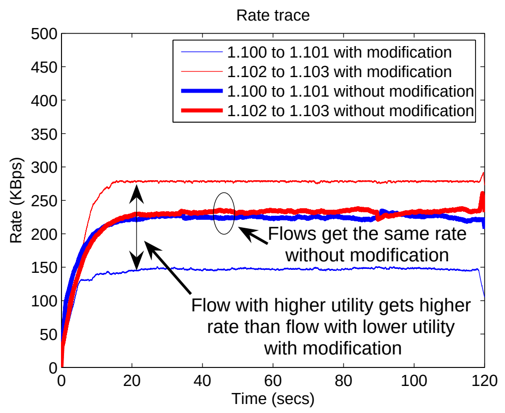


Figure 3: Rate allocations under modified and unmodified MadWifi. Modified MadWifi can give more channel access (thus higher rate) to flows with higher utility.

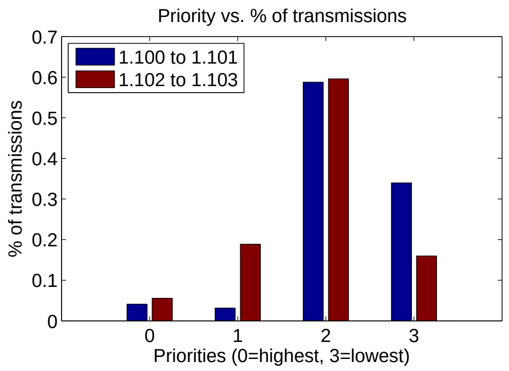


Figure 4: Priority vs. % of TX: Flows with "higher" utility can get more rate by trying to access the channel more aggressively. This aggressive access is facilitated by smaller AIFS and contention windows.

Effect of the Modifications: To see how important the modifications are to the utility maximizing BP algorithm (i.e., rate control + BP), we ran a simple experiment with four wireless nodes (labeled 1.100, 1.101, 1.102 and 1.103) on the 5.26GHz channel. There are two single-hop flows in this experiment (however, all four nodes are in the same collision domain). The nodes 1.100 and 1.102 transmit to 1.101 and 1.103, respectively, with the utility functions $U(x) = 400\log(x)$ and $U(x) = 800\log(x)$, respectively, where x is measured in Bps by the sources. If the queue difference between the source and the destination is greater than or equal to 20, between 19 and 17, between 16 and 15, and lower than or equal to 14, we assign TOS levels 0, 1, 2 and 3, respectively (we represent this mapping by the threshold array $L = \{20, 16, 14\}$).

Priority #	0	1	2	3
AIFS	1	3	5	7
CWmin	1	7	31	255
CWmax	7	63	255	1023

Table 1: MAC scheduling parameters of the four MadWifi hardware queues

The optimal rate allocation for our simple experiment is for the flow from 1.102 to 1.103 to have twice as much rate as the flow from 1.100 to 1.100. With the modified wireless device driver, we indeed obtain a rate allocation that is very close to the optimal (see Figure 3). However, with no modifications, the rates allocated to both flows are the same. The source 1.102 has a higher marginal utility function and therefore can maintain a larger queue, which in turn translates into assigning more urgent priority to more of its outgoing packets (see Figure 4). (The figure does not capture the ACKs that were always transmitted for each data packet with priority 0.)

Finding the optimal threshold array L can be difficult, and suboptimal levels can have severe impact on the performance. To illustrate this, we conduct another experiment with three wireless nodes (from 1.101 – 1.102 – 1.103, on 5.26GHz) in a line.

The long flow from 1.101 to 1.103 has the utility function $K_1 \log(x_1)$ and the short flow from 1.102 to 1.103 has the utility function $K_2 \log(x_2)$. Let f_1 and f_2 be the fraction of time that the links 1.101-1.102 and 1.102-1.103 are active, respectively. Due to the coupled wireless interference constraint, $f_1 + f_2 \leq 1$. The optimal rate allocation can then be obtained by solving

$$\begin{aligned} \text{maximize} \quad & K_1 \log(x_1) + K_2 \log(x_2) \qquad (5) \\ \text{subject to} \quad & f_1 + f_2 \leq 1 \\ & x_1 \leq f_1 C \\ & x_1 + x_2 \leq f_2 C \end{aligned}$$

assuming that both links 1.101-1.102 and 1.102-1.103 have the same link capacity C. (C is the one hop transmission rate *between two nodes with no other transmissions in the range* and was measured to be around 465KBps.)

If $K_1 = K_2 = 200$, the optimal rate allocation is such that the short flow 1.102-1.103 is allocated twice as much rate as the long flow 1.101-1.103, since 1.101-1.103 requires two transmissions for every packet to reach the destination. Using queue difference levels $L = \{25, 13, 5\}$ (i.e. when the queue difference is ≥ 25, priority 0 is assigned to the packets, etc.), we were able to obtain a near optimal allocation (thin trace lines in Figure 5(a)). However, when using the queue difference levels $L' = \{25, 18, 10\}$, we obtained a suboptimal allocation (thick trace lines). In particular, the short flow from 1.102 to 1.103 suffers 25% reduction in throughput. The reason is that the queue difference levels were set too high. In order to maximize eq. (2), links with more weight $P^{j(m,n)}_{(m,n)}$ should receive more channel access (at the cost of causing interference to links with lower weight). However, because the difference levels were set too high, links received similar rates, even though they had different weights. It is difficult to know what the optimal difference levels are before hand, as the optimal levels are driven in part by the β parameter in eq. (4) and the choice of the utility functions.

When we use queue difference levels $L' = \{25, 5, 2\}$, we were able to produce a suboptimal allocation with $K_1 = 800$ and $K_2 = 200$. In this setting, the long flow from 1.101 to 1.103 should be allocated twice as much rate as the short flow from 1.102 to 1.103. Indeed, with the same queue difference levels L, we got close to the optimal allocation (see Figure 5(b)).

We will use the optimal rate allocations in this line network as a benchmark against which we compare the rate allocations of the modified BP algorithm for intermittently connected networks in the next section. So for the sake of completeness, we include the case when $K_1 = 400$ and $K_2 = 200$ in Figure 5(c). In this case, the rate allocation should be equal for both the short and the long flows.

On our MadWifi+Click platform, we will implement a modified BP algorithm for intermittently connected networks. We emphasize that the focus of this paper is *not* an implementation of traditional back-pressure routing with the modified driver; rather, our aim is to decouple the two time scales (mobile-gateway and internal-internal) in such networks. In order to do this, we need the MAC algorithm to support differentiated levels of contention, and we need the results from this single time-scale 1.101-1.102-1.103 line network against which we can compare our results to see how effectively we achieve our goal.

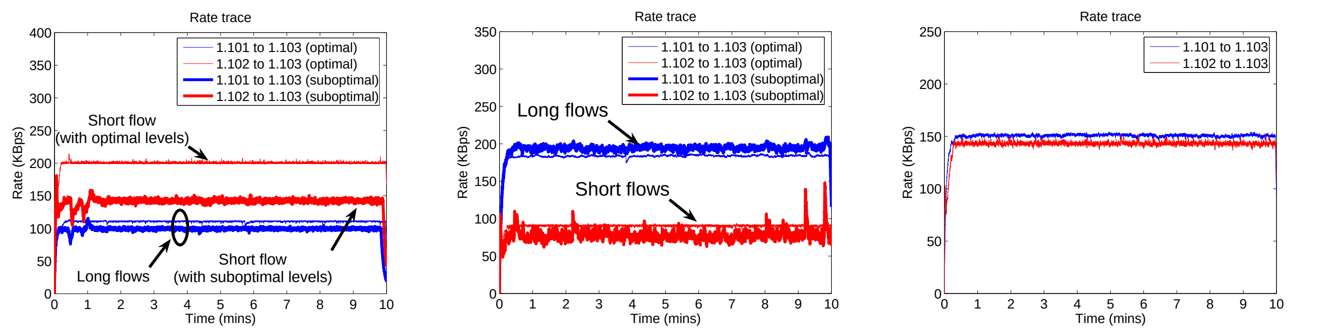
5. BACK-PRESSURE IN ICN

We begin with a description of the modified back-pressure algorithm for intermittently connected networks. The issue that the traditional back-pressure algorithm would suffer in a multiple time-scale environment (such as an intermittently connected network) was identified in [18]. The authors in [18] have proposed a modified BP algorithm to address this issue, where the time-scales at the inter-cluster and intra-cluster levels have been (loosely) decoupled by a hierarchical BP approach. Essentially, there are two BP algorithms operating, each at different queue-scales and time-scales, and the coupling between them occurs at the gateway nodes. In this paper, we build on their work in order to *(i)* study rate control, *(ii)* to address real implementation concerns, and *(iii)* to study the performance improvements that we can measure in a real network.

5.1 A Modified BP Algorithm

We present a simplified version of the algorithm described by the authors in [18]. Consider slotted time t. We assume that the time duration between two contacts as measured at the gateways is T time slots, where T is a very large number. A time slot is the time scale of one wireless packet transmission, and T is the time scale of the mobility (thus, a time slot is roughly a few milliseconds long, and T is roughly $> 10^3$ to reflect the mobility time scale which is seconds or minutes long). Within each cluster, the back-pressure routing algorithm runs as it was originally designed.

In brief, the gateway nodes advertise their queue-lengths *scaled down by a factor T* within their respective clusters. All other nodes in the cluster advertise their queue lengths without any scaling, and the back-pressure algorithm operates within the cluster based on the *advertised* queue lengths (and not the actual queue lengths). However, when a gateway comes in contact with a mobile, it uses its *actual* queue length to compute the back-pressure (i.e., eq. (1)) between itself and the mobile.



(a) $K_1 = K_2 = 200$. The short flow rate should be two times the long flow rate.

(b) $K_1 = 800,\ K_2 = 200$. The long flow rate should be two times the short flow rate.

(c) $K_1 = 400,\ K_2 = 200$. The long and short flows should get the same rate.

Figure 5: Rate allocations in 1.101-1.102-1.103 line network. We can control which flow gets more rate by controlling the utility function parameters K_1 and K_2. As K_1 increases relative to K_2, the long flow gets more and more rate. These figures also highlight how improper selection of queue difference levels can impact rate allocation performance (thick trace lines).

A packet from a source eventually gets to the gateway of the cluster in which the destination lies. This destination gateway g maintains *two queues* for each node n in the same cluster as itself. One queue is to receive inter-cluster packets destined for node n from the mobiles (call this queue $\hat{q}_g^n$). The length of $\hat{q}_g^n$ is used for back-pressure routing with the mobiles. Once an inter-cluster traffic packet destined for n arrives at the gateway g, it is put into $\hat{q}_g^n$. The other queue (call it q_g^n) is the intra-cluster queue within the destination cluster: This is the queue length advertised by the gateway for back-pressure computation within this cluster (q_g^n can also be used to relay intra-cluster packets for n.) In each time slot t, g transfers η ($\eta << R$, where R is the number of packets transferred between mobiles and gateways on contact) packets from $\hat{q}_g^n$ to q_g^n if and only if

$$\frac{\hat{q}_g^n[t]}{T} \geq q_g^n[t]. \qquad (6)$$

Once put into q_g^n, the inter-cluster packets are routed to the destination using back-pressure routing in the destination cluster.

Also, to reduce the number of queues to be maintained by each node in our implementation, a gateway that receives packets destined for a different cluster (i.e., is a way-point gateway), does not send out the inter-cluster packets to the internal nodes within its cluster. This way, an internal node only has to maintain a queue for every other node in the same cluster as itself and only for the other nodes in different clusters that are destinations of inter-cluster traffics originating from the same cluster.

The algorithm in [18] requires the knowledge of the time scale difference between the wireless packet transmission and the mobility. But in fact, even a rough estimate (anything $\Theta(T)$) of the difference is good enough and their throughput optimality result would still hold. The best scaling factor T would be the ratio of the time duration it takes the mobiles to make two contacts to the intra-cluster time slot; however, this is difficult to measure precisely. If too large estimate is used for T, it would result in longer queues at the gateways and longer time before the inter-cluster rates converge. If too small estimate is used, this would result in fluctuations in the instantaneous inter- and intra-cluster rates. This rate fluctuations can be seen in Figures 6(a), 6(b), and 6(c).

In order to speed up our experiments so that they may be conducted in a reasonable amount of time, we choose the time scale of the mobility to be of the order of tens of seconds, which is long enough for $\Theta(1000)$ wireless 1KB packet transmissions on 802.11a channels. To make sure that the bottleneck is not the mobile-gateway part of the network (more interesting case happens when congestion occurs in a different cluster), we choose R to be the same order as T.

5.2 Verification on a Line Network

5.2.1 Utility Maximizing Performance

We experimentally verify the modified back-pressure algorithm on a simple intermittently connected line network shown in Figure 1. We used a 100Mbps Cisco switch to emulate the mobile-gateway contacts. Each gateway node is equipped with one wireless card and an Ethernet port. The gateways use the wireless card to communicate with the internal nodes and the Ethernet port to communicate with the "mobile."

On each contact, up to 6000 packets can be transferred. Each packet has a payload of 1KB, in addition to IP and Ethernet and the modified BP headers (roughly 6MB per contact). The mobile contact node switched clusters every 10 seconds, so two consecutive contacts at a gateway are 20 seconds apart. Thus, the average rate (averaged over many contacts) from the source cluster (the left cluster) to the destination cluster (the right cluster) is 300KBps. We also chose $T = 6000$ and $L = \{25, 13, 5\}$.

The purpose of the modified BP algorithm is to have the inter-cluster traffic source be totally unaware of the mobile-gateway contacts, and to disturb any intra-cluster traffic as little as possible. We also want to obtain utility maximizing rate allocation, *even though the clusters are physically separated.*

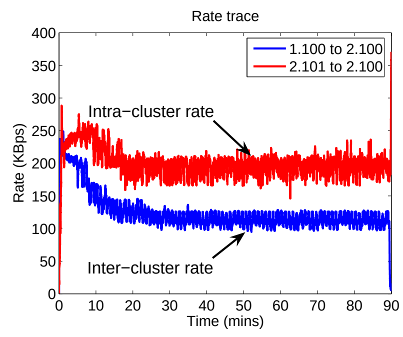
Let x_1 denote the inter-cluster rate, and x_2 denote the intra-cluster rate. The utility functions for inter- and intra-cluster traffics are

$$U_1(x_1) = K_1 \log(x_1)$$

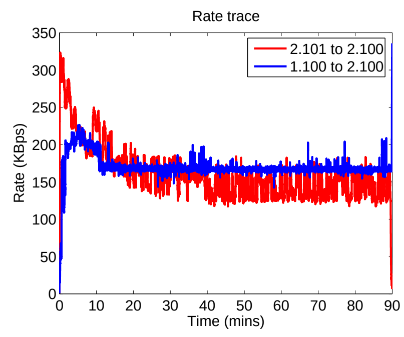
and

$$U_2(x_2) = K_2 \log(x_2),$$

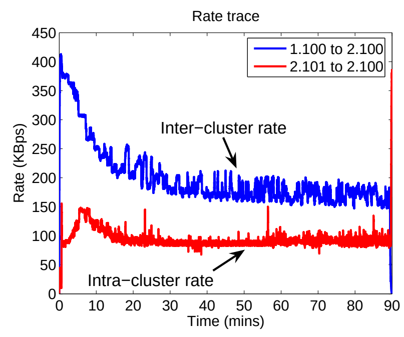
respectively. We made sure the only bottleneck is the destination cluster. (If the bottlenecks are either the source



(a) $K_1 = K_2 = 200$. Intra-rate should be $2\times$ inter-rate.

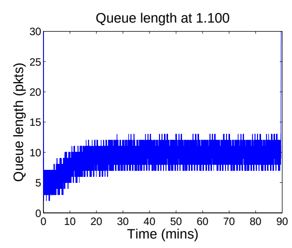


(b) $K_1 = 400$, $K_2 = 200$. Intra- and inter-rates should be equal.

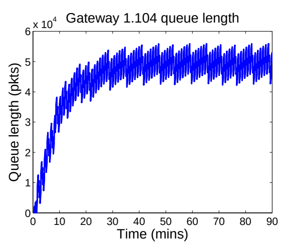


(c) $K_1 = 800$, $K_2 = 200$. Inter-rate should be $2\times$ intra-rate.

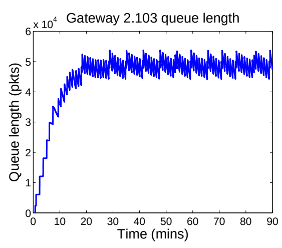
Figure 6: Rate allocation in the network shown in Figure 1 (compare against optimal rate allocations in Figures 5(a), 5(b), and 5(c)). The presence of the intermittent link is hidden both to the inter- and intra-cluster sources since they achieve the same rates in the network shown in Figure 1 as in the two-hop 1.101-1.102-1.103 line network.



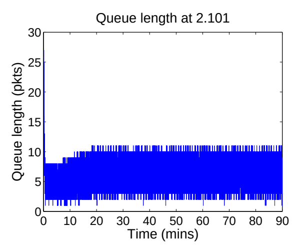
(a) Inter-cluster source 1.100



(b) Source gateway 1.104

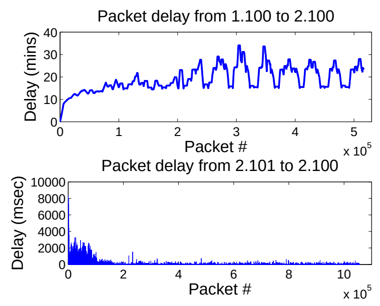


(c) Dest. gateway 2.103

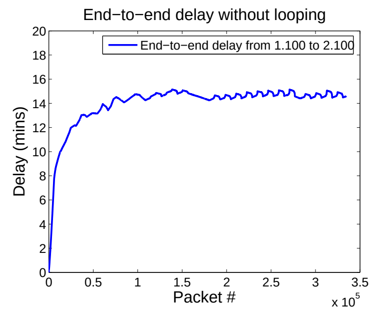


(d) Intermediate node 2.101

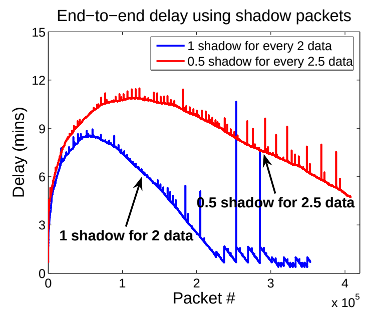
Figure 7: Queue traces under the modified BP. Large queues are confined only to gateways and mobiles. Internal nodes have small queues. ($K_1 = K_2 = 200$)



(a) Inter-cluster delay vs. intra-cluster delay



(b) End-to-end delay from 1.100 to 2.100 without looping



(c) End-to-end delay from 1.100 to 2.100 using shadow packets

Figure 8: Inter-cluster delay: we can improve delay performance by preventing packet "looping" and by using shadow packets. Note the significant reduction in delay as we back-off from the capacity region boundary by using shadow packets. ($K_1 = K_2 = 200$)

cluster or the mobile-gateway contacts, this could easily be learned by the source.) Thus, the optimal rate allocation is the solution to the maximization problem (5) (with f_1 and f_2 denoting the fraction of time the wireless channel 2.103-2.101 and 2.101-2.100 are active, respectively). In summary:

- Recall that the traditional BP rate controller fails to give an optimal rate allocation (unless a large scaling is done, resulting in large queue sizes), and resulted in low inter-cluster rates as seen in Figure 2(a). However, using the modified BP, we get a high, sustained throughput for both the inter- and intra-cluster flows, and their rates are (shown Figures 6(a), 6(b) and 6(c)) close to the theoretically computed values. The rates are also close to the ones (with optimal queue difference levels) shown in Figures 5(a), 5(b) and 5(c). Thus, the modified BP successfully hides the presence of the intermittent links.

- We also verify that large queues occur only at gateways; we plot the queue trace at the source and the destination gateways in Figures 7(b) and 7(c). The queues at internal nodes 1.100 and 2.102 are small as seen in Figures 7(a) and 7(d). We note that under the traditional back-pressure algorithm, large queues are maintained at every node in the source cluster to achieve sustained rates – see Figure 2(c). This is due to the "burstiness" of the intermittent links.

5.2.2 End-to-End Delay: Shadow Packets

We compare the end-to-end delay that inter- and intra-cluster packets see. Since inter-cluster packets must pass through the gateways and mobiles with large queues, they incur large delays. However, intra-cluster packets need not, and hence incur relatively smaller delays. This can be seen in Figure 8(a). One factor causing the large inter-cluster packet delay is that some packets are "looping" between the gateways and the mobile. ("Looping" through large FIFO queues a few times can increase delays significantly.) When we prevent this looping, we get much smoother delays (but still large), as seen in Figure 8(b).

Another factor that contributes to the large inter-cluster delays is that our utility-maximizing rate controller operates very close to the boundary of the capacity region. This is known to require large queues and can thus cause long delays. The authors in [3] deal with this problem by introducing the notion of shadow packets and queues. Their essential idea here is to trade-off throughput for low delays. Our implementation of shadow packets is as follows:
Shadow packet implementation: For every $\kappa = 3$ packets that the inter-cluster source injects into its queue according to eq. (4), it marks one red (or shadow). The other two packets are marked blue. These shadow packets are dummy packets and do not contain any useful data (but still have 1KB payload). The blue packets contain real data. Thus, the real, useful rate is $0.66x$. The gateways and mobiles have two FIFO sub-queues for each inter-cluster destination. The red and blue packets are separated into these two sub-queues. The blue packets get transmission priority over the red packets, i.e. shadow packets are transmitted if and only if there are no blue packets that can be served. The total size (blue queue size + red queue size) is used for back-pressure routing.

The end-to-end delay with shadow packets is shown in Figure 8(c). (We also did another experiment where we send 1 shadow packet for every 5 data packets on average.) The delay curve first increases as we first need to build large queues at the gateways and mobiles. But as the real packets have priority, only the dummy, shadow packets are left behind to hold the steady-state queue sizes required for back-pressure to work. The inter-cluster delay decreased from roughly 15mins (Figures 8(a) and 8(b)) to roughly 1-2mins using shadow packets. Note that as we send fewer shadow packets per data packet, the delay decreases slower. This is because it takes longer for enough shadow packets to accumulate up to the steady-state queue size.

6. EXPERIMENTAL RESULTS ON A LARGER NETWORK

We implemented the modified back-pressure algorithm on our 16-node test bed. The network we conducted our experiment is shown in Figure 12. The nodes in each cluster were placed only a few feet apart. (The clusters were on different channels.) Each node in a cluster uses packet filtering based on the source IP address; so for example, 1.101 can accept packets only from 1.102 and is only aware of 1.102's presence. Thus, 1.101 will only transmit to 1.102. We are aware that any transmission from, say, 1.101 causes interference on all other transmissions because the wireless range is large enough to cover the entire cluster. However, a node can receive only one transmission at a time, and a failed transmission from, say, 1.103 (which will also transmit to 1.102 only) to 1.102 due to the interference caused by a simultaneous transmission from 1.101 would not have been received by 1.102 anyways even if the nodes were placed farther apart. (We are aware that the way we have closely laid out the nodes to conduct our experiment does not completely model the network depicted in Figure 12. For example, as depicted in the figure the nodes 1.103 and 1.101 are supposedly placed far apart, but they are within each other's transmission range. Thus, there can be a collision between transmissions from 1.101 and 1.103 in the depicted network, but not in our physical network (because of CSMA). However, in practice the carrier sensing range of 802.11 is larger than the transmission range. Hence, the network we are actually modeling is where the nodes 1.101 and 1.103 are out of each other's transmission range, but still within the carrier sense range.) Thus, placing the nodes close does not make our experimental results less valid.

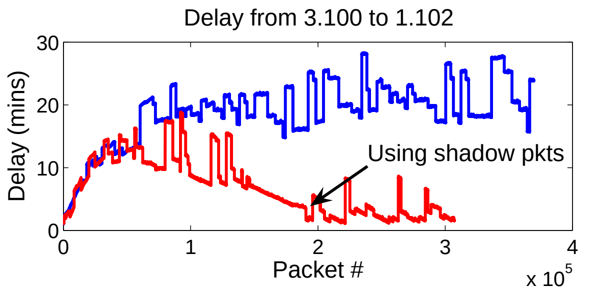
Using a 100Mbps switch to emulate the mobile-gateway contacts, up to 6000 packets can be transmitted between a mobile and a gateway (so up to 12000 packets total) per contact. We picked $T = 6000$, $\kappa = \eta = 3$, and used queue difference levels $L = \{25, 13, 5\}$. After the contact is finished, the mobiles pick one of the other two gateways randomly, and initiate another contact 14 seconds later. We are aware that 14 seconds is not long enough to model most mobility in the real world. However, we picked 14 seconds to speed up our experiments so that we can have many contacts within a reasonable amount of time. (In case the intermittent connectivity time scale is of hours/days, our BP-based algorithm could result in very long queues since our data sources transmit at a high and stable rate for throughput optimality. However, in such cases, one potentially would be more interested in just connectivity in applications like a single file transfer; in such cases, replication-based algorithms seem to be a better choice.) Finally, all flows have the same utility function of $200 \log(\cdot)$ in this study.

In summary, we have the following results from our experiment on the network in Figure 12:

- We observe that even in this larger network, the intra-cluster queues remain very small (between 10-15 pkts). See Figures 11(b) and 11(a). Only the inter-cluster flows suffer large delays due to longer queues (between 10^4 and 3×10^4 pkts).

- Furthermore, using our implementation of the shadow queues and packets (the idea was proposed by the authors in [3]; we have developed an implementation for intermittently connected networks), we can "back-off" from operating on the boundary of the throughput region (i.e., utility optimal), and improve the delay performance for the inter-cluster flows; the inter-cluster delay decreased from ≈ 20 mins (blue, solid trace in Figure 10) to ≈ 2 mins using our shadow packets (red, dashed traces in Figure 10).

- To get a baseline on the approximate values we should expect, we used the fluid deterministic optimization



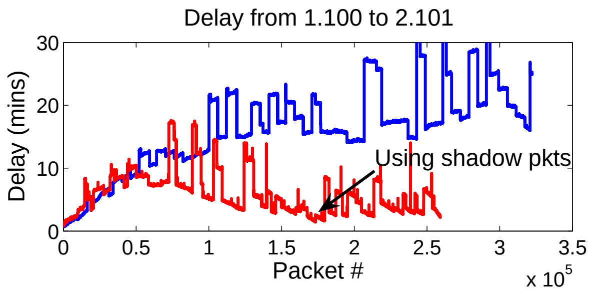


Figure 10: Inter-cluster delays in the network in Figure 12 (red, dashed=using 1 shadow packet for every 2 data packets)

(that ignores ACKs, collisions, retransmissions) to obtain the "optimal" rate allocations to be $x_1 = 106$, $x_2 = 83$, $x_3 = 191$, $x_4 = 146$ and $x_5 = 87$ (all KBps). Our experimental numbers are $x_1 = 84$, $x_2 = 68$, $x_3 = 120$, $x_4 = 107$ and $x_5 = 86$ (all KBps). The rates differ from the fluid approximation anywhere from a few percent to about 33%. One source of the discrepancy is that the theoretical framework as in eq. (5) assumes a fluid model with no contention loss and exponential back-offs. However, there are many packet collisions that trigger the exponential back-offs that decrease the "link capacity" in the (idealized) fluid model. Another source of the discrepancy is that the theory does not model the ACKs. We used at least one ACK for each data packet over each hop (so there are two channel accesses to deliver a packet each hop). All ACKs are transmitted at priority 0 (highest priority) and are about 80B long (IP+BP+queue information).

7. RELATED WORKS

The back-pressure algorithm, in one form or another, has been implemented in wireless mesh networks in [27, 17, 13, 21, 10]. [27] improves TCP performance over a wireless ad-hoc network by utilizing the back-pressure scheduling algorithm with backlog based contention resolution algorithm. [17] improves multi-path TCP performance by taking advantage of the dynamic and resilient route discovery algorithmic nature of BP. The authors in [13] have implemented and studied back-pressure routing over a wireless sensor network. They have used the utility-based framework of the traditional BP algorithm, and have developed implementations with good routing performance for data gathering (rate control is not studied in [13]). Their chief objective is to deal with the poor delay performance of BP. [21] is an implementational study of how the performance of BP is affected by the network conditions, such as the number of active flows, and under what scenarios backlog based contention resolution algorithm is not necessary. [10] studies utility maximization with queue-length based throughput optimal CSMA, for single-hop flows (no routing or intermittent connectivity). Lastly, [1] is not an implementation, but discusses a lot of issues related to BP routing with rate control implementation. Our study differs from all of them in that we focus on the disparate time-scales issue in an intermittently connected network (thus, queues throughout the network get "poisoned" with the traditional BP), and study modifications that loosely decouple the time-scales for efficient rate control.

Initially, the approach taken in intermittently connected networks and DTNs for routing was based on packet replications. The simplest way to make sure packets are delivered is to flood the "mobile" portion of the network so that the likelihood of a packet reaching the destination increases as more and more replicates are made [26]. A more refined approach is to control the number of replicates of a packet so that there is a balance between increasing the likelihood and still leaving some capacity for new packets to be injected into the network [2, 20, 4, 26, 12]. Another refined approach is to learn the intermittently connected topology and use this knowledge to route/replicate through the "best" contacts and encounters and avoid congestion [7, 8, 19, 25, 11].

[16, 28, 6] study networks that are closer to ours. In [28], distant groups of nodes are connected via mobiles, much like our network but with general random mobility. At intra-group level, a MANET routing protocol is used for route discovery, and at inter-group level, the Spray-and-Wait is used among mobiles to decrease forwarding time and increase delivery probability. [16] augments AODV with DTN routing to discover routes and whether those routes support DTN routing and to what extent they support end-to-end IP routing and hop-by-hop DTN routing. [6] studies how two properties of the mobile nodes, namely whether a mobile is dedicated to serve a specific region (ownership) and whether the mobile movement can be scheduled and controlled by regions (scheduling time), affect the performance metrics such as delay and efficiency.

All the aforementioned replication-based algorithms are valuable as they provide insight into engineering an efficient and robust ICN protocol, and as discussed in the introduction, the only viable choice may be to use the replication-based algorithm over networks with unpredictable and random mobility. There is (to the best of our knowledge) not much literature on rate control over ICNs. We demonstrated

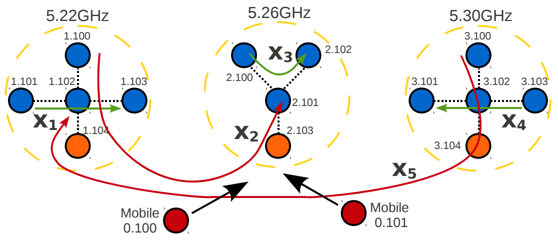
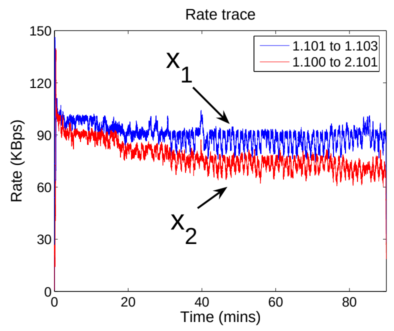
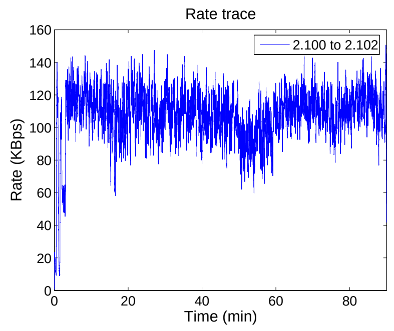


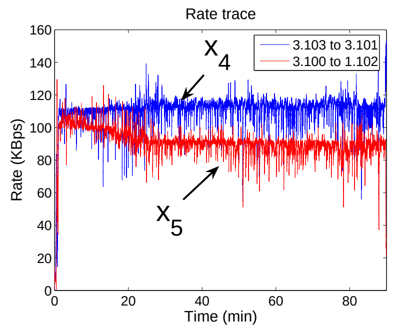
Figure 12: Our experimental network with 16 nodes



(a) Rates x_1 and x_2 in the 5.22GHz cluster

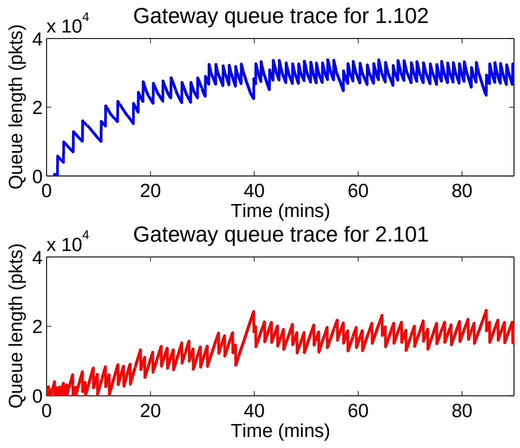


(b) Rate x_3 in the 5.26GHz cluster

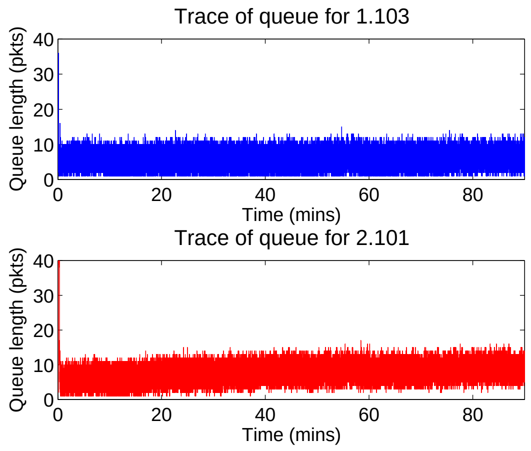


(c) Rates for x_4 and x_5 in the 5.30GHz cluster

Figure 9: Rate allocation in the 16-node network. See Figure 12 for the labels x_1, x_2, x_3, x_4 and x_5. x_2 and x_5 are inter-cluster flows.



(a) Queue trace at gateway 1.104



(b) Queue trace at 1.102

Figure 11: Queue trace in the network shown in Figure 12. Large queues are confined to the "mobile-gateway" portion of the network, and we have large reduction in delay through shadow packets (Figure 10).

that it is possible to obtain utility maximizing rate allocation, even though there is the "mobile-gateway" time scale that operates much slower than the wireless communication time scale, and all inter-cluster packets have to pass through the two different time scales.

8. CONCLUSION

In this paper, we have presented a back-pressure rate-control/routing algorithm adapted for intermittently connected networks, where we can exploit ergodic mobility patterns to obtain throughput optimal performance. Our proposed algorithm solved the time-scale coupling of the traditional back-pressure algorithm; namely, intermittent connectivity feeds-back the wrong congestion signal to the inter-cluster source, making it believe that the connection is a low-rate link, or in order to have high inter-cluster rate, one has to maintain large queues at internal nodes that are no where near the intermittent links. We have verified that our algorithm works on a simple line network, and on a larger 16-node network.

9. ACKNOWLEDGMENTS

The authors would like to express their gratitude to Prof. Christine Julien and Prof. Sriram Vishwanath for permitting the use of the Pharos testbed [23] to generate the experimental results presented in this paper. The authors would also like to thank Prof. R. Srikant (UIUC), Prof. Lei Ying (ISU) and the anonymous reviewers for their valuable feedback. We acknowledge the support of NSF Grants CNS-0347400 and CNS-0721380 and the DARPA ITMANET program.

10. REFERENCES

[1] U. Akyol, M. Andrews, P. Gupta, J. Hobby, I. Saniee, and A. Stolyar. Joint scheduling and congestion control in mobile ad-hoc networks. In *IEEE Infocom*, 2008.

[2] A. Balasubramanian, B. N. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. In *Proc. of ACM SIGCOMM*, 2007.

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

[4] J. Burgess, B. Gallagher, D. Jense, and B. N. Levine. MaxProp: Routing for vehicle-based

disruption-tolerant networks. In *Proc. IEEE INFOCOM*, 2006.

[5] A. Eryilmaz and R. Srikant. Joint congestion control, routing, and MAC for stability and fairness in wireless networks. *IEEE JSAC*, 24:1514–1524, 2006.

[6] K. A. Harras and K. C. Almeroth. Inter-regional messenger scheduling in delay tolerant mobile networks. In *Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*.

[7] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *Proc. of SIGCOMM*, 2004.

[8] E. Jones, L. Li, and P. Ward. Practical routing in delay-tolerant networks. In *Proc. of SIGCOMM*, 2005.

[9] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. In *ACM Transactions on Computer Systems*, 2000.

[10] J. Lee, J. Lee, Y. Yi, S. Chong, A. Proutiere, and M. Chiang. Implementing utility-optimal CSMA. In *Allerton*, 2009.

[11] K. Lee, Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong. Max-contribution: On optimal resource allocation in delay tolerant networks. In *IEEE INFOCOM*, 2010.

[12] A. Lindgren, A. Doria, and O. Scheln. Probabilistic routing in intermittently connected networks. In *ACM MobiHoc*, 2003.

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

[14] M. J. Neely. *Dynamic power allocation and routing for satellite and wireless networks with time varying channels*. PhD thesis, Massachusetts Institute of Technology, 2003.

[15] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. In *Proc. of IEEE INFOCOM*, 2005.

[16] J. Ott, D. Kutscher, and C. Dwertmann. Integrating DTN and MANET routing. In *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*.

[17] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: Balancing TCP over multiple paths in wireless mesh network. In *Proc. of Mobicom*, 2008.

[18] J. Ryu, L. Ying, and S. Shakkottai. Back-pressure routing in intermittently connected networks. In *IEEE Infocom Mini-conference*, 2010.

[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] 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.

[21] A. Sridharan, S. Moeller, and B. Krishnamachari. Implementing backpressure-based rate control in wireless networks. In *ITA workshop*, 2009.

[22] A. L. Stolyar. Maximizing queueing network utility subject to stability: greedy primal-dual algorithm. *Queueing Systems*, 50:401–457, 2005.

[23] D. Stovall, N. Paine, A. Petz, J. Enderle, C. Julien, and S. Vishwanath. Pharos: An application-oriented testbed for heterogeneous wireless networking environments. Technical Report TR-UTEDGE-2009-006, The Center for Excellence in Distributed Global Environments, The University of Texas at Austin, 2009.

[24] 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*, 37(12):1936–1949, December 1992.

[25] N. Thompson, S. C. Nelson, M. Bakht, T. Abdelzaher, and R. Kravets. Retiring replicants: Congestion control for intermittently-connected networks. In *Proc. of IEEE INFOCOM*, 2010.

[26] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical Report CS-2000-06, Department of Computer Science, Duke University, April 2000.

[27] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee. DiffQ: Practical differential backlog congestion control for wireless networks. In *IEEE Infocom*, 2009.

[28] J. Whitbeck and V. Conan. HYMAD: Hybrid DTN-MANET routing for dense and highly dynamic wireless networks. In *In Proceedings: IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC 2009)*.

[29] 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.

[30] L. Ying, R. Srikant, and D. Towsley. Cluster-based back-pressure routing algorithm. In *Proceedings of IEEE INFOCOM*, 2008.