# Notes on Backpressure Routing

## I. Backpressure Routing for Networks

Backpressure routing refers to an algorithm for dynamically routing traffic over a multi-hop network by using congestion gradients. It usually refers to a data network, but can apply to other types of networks as well. Below we focus on the data network application, where multiple data streams arrive to a network and must be delivered to appropriate destinations. The backpressure algorithm operates in slotted time, and every slot it seeks to route data in directions that maximize the *differential backlog* between neighboring nodes. This is similar to how water would flow through a network of pipes via pressure gradients. However, the backpressure algorithm can be applied to multi-commodity networks and to networks where transmission rates can be selected from different (possibly time-varying) options. Attractive features of the backpressure algorithm are: (i) it leads to maximum network throughput, (ii) it is provably robust to time-varying network conditions, (iii) it can be implemented without knowing traffic arrival rates or channel state probabilities.

The original backpressure algorithm was developed by Tassiulas and Ephremides in [1]. There, they considered a multi-hop packet radio network with random packet arrivals and different link selection options. Their algorithm consisted of a *max-weight link selection* stage and a *differential backlog routing stage*. They did not call their algorithm "backpressure," and at the time of [1] the term "backpressure" had a different meaning in the area of data networks (it refered to a class of window-based flow control techniques, see [2]). The Tassiulas-Ephremides algorithm was first called "backpressure" in [3] and [4], where the algorithm was extended to treat networks with mobility, including ad-hoc mobile networks. An algorithm related to backpressure, designed for computing multi-commodity network flows, was developed in [5].

Backpressure is mathematically analyzed via the theory of Lyapunov drift, and has been unified with utility optimization techniques in [3][6][7][8]. Below we present the backpressure algorithm and its analysis, based on material from [7] and [8] (see those texts for details).

## II. Multi-Hop Queueing Networks

Consider a multi-hop network with $N$ nodes. The network operates in slotted time $t \in \{0, 1, 2, \ldots\}$. On each slot, routing and transmission scheduling decisions are made in an effort to deliver all data to its proper destination. New data also arrives every slot. Let data that is destined for node $c \in \{1, \ldots, N\}$ be labeled as *commodity $c$ data*. Every node keeps internal queues that store data according to its destination. Let $Q_n^{(c)}(t)$ be the current amount of commodity $c$ data in node $n$, also called the *queue backlog*. The units of $Q_n^{(c)}(t)$ depend on the context of the problem. For example, they can be integer units of *packets* or real valued units of *bits*. The queue backlogs from one slot to the next satisfy the following for all $n \in \{1, \ldots, N\}$, $c \in \{1, \ldots, N\}$ such that $n \neq c$:

$$Q_n^{(c)}(t+1) \leq \max\left[Q_n^{(c)}(t) - \sum_{b=1}^{N} \mu_{nb}^{(c)}(t), 0\right] + \sum_{a=1}^{N} \mu_{an}^{(c)}(t) + A_n^{(c)}(t) \quad (1)$$

where $A_n^{(c)}(t)$ is the random amount of new commodity $c$ data that exogenously arrives to node $n$ on slot $t$, and $\mu_{ab}^{(c)}(t)$ is the transmission rate allocated to commodity $c$ traffic on link $(a, b)$ on slot $t$. Note that $\mu_{ab}^{(c)}(t)$ may be more than the amount of commodity $c$ data that is actually transmitted on link $(a, b)$ on slot $t$ (otherwise, there would be no need for the $\max[\cdot, 0]$ operation in (1)). This is because there may not be enough backlog in node $a$. For this same reason, (1) is an inequality, rather than an equality, because $\sum_{a=1}^{N} \mu_{an}^{(c)}(t)$ may be more than the actual endogenous arrivals of commodity $c$ to node $n$ on slot $t$. An important feature of (1) is that it holds even if the $\mu_{ab}^{(c)}(t)$ decision variables are chosen independently of queue backlogs.

We assume that $Q_c^{(c)}(t) = 0$ for all slots $t$ and all $c \in \{1, \ldots, N\}$, as no queue stores data destined for itself.

### A. Dynamic Arrivals

Let $(A_n^{(c)}(t))$ be the matrix of exogenous arrivals on slot $t$. For simplicity, assume this matrix is independent and identically distributed (i.i.d.) over slots with finite second moments and with means:

$$\lambda_n^{(c)} \triangleq \mathbb{E}\left\{A_n^{(c)}(t)\right\}$$

We assume that $\lambda_c^{(c)} = 0$ for all $c \in \{1, \ldots, N\}$, as no data arrives that is destined for itself. Thus, the matrix of arrival rates $(\lambda_n^{(c)})$ is a $N \times N$ matrix of non-negative real numbers, with zeros on the diagonal.

### B. Dynamic Network Topology and Transmission Decision Options

The network might have time-varying channels and/or mobile nodes. Let $S(t)$ represent the *topology state* of the network on slot $t$, which contains information relevant to the node-to-node transmission rates that can be supported on the current slot. Assume the network controller observes $S(t)$ at the beginning of each slot $t$, and chooses a matrix $(\mu_{ab}(t))$ of transmission rates within some abstract set that depends on $S(t)$. That is, the controller chooses:

$$(\mu_{ab}(t)) \in \Gamma_{S(t)} \tag{2}$$

where $\Gamma_{S(t)}$ is the set of rates that can be supported under state $S(t)$. This model fits a variety of networks, including ad-hoc mobile networks, networks with orthogonal channels, as well as networks with inter-channel interference. It assumes the supportable transmission rates are known and there are no transmission errors. Backpressure methods with probabilistic channel errors are considered in [9].

Every slot $t$, the network controller chooses transmission variables $(\mu_{ab}(t)) \in \Gamma_{S(t)}$, and then chooses routing variables $(\mu_{ab}^{(c)}(t))$ subject to the constraints:

$$0 \leq \mu_{ab}^{(c)}(t) \ \ \forall a, b, c, \forall t \tag{3}$$

$$\sum_{c=1}^{N} \mu_{ab}^{(c)}(t) \leq \mu_{ab}(t) \ \ \forall (a, b), \forall t \tag{4}$$

Once these decisions are made, the queues are updated via (1). In case of "queue underflow" at a particular node, where there is not enough backlog of a given commodity in that node to fill all of its offered transmission rates, there may be an additional choice of which outgoing links to use. In this case, any choice that satisfies (1) is valid.

### C. Network Capacity Region

For simplicity, assume the topology state $S(t)$ is i.i.d. over slots with probabilities $\pi_S = Pr[S(t) = S]$ (for simplicity, assume there are a finite or countably infinite number of possible states). The *network capacity region* $\Lambda$ is the closure of the set of all arrival rate matrices $(\lambda_n^{(c)})$ for which there exists an algorithm that observes $S(t)$ every slot $t$ and chooses transmission rates $(\mu_{ab}(t))$ and routing variables $(\mu_{ab}^{(c)}(t))$ according to the constraints (2)-(4) that yields stability of all queues. Stability of all queues implies that the total input rate of traffic into the network is the same as the total rate of data delivered to its destination (see detailed discussion of stability in [8]). It can be shown that for any arrival rate matrix $(\lambda_n^{(c)})$ in the capacity region $\Lambda$, there is a *stationary and randomized algorithm* that chooses decision variables $(\mu_{ab}^*(t))$ and $(\mu_{ab}^{*(c)}(t))$ every slot $t$ based only on $S(t)$ (and hence independently of queue backlogs) that yields the following for all $n \neq c$:

$$\mathbb{E}\left\{\lambda_n^{(c)} + \sum_{a=1}^{N} \mu_{an}^{*(c)}(t) - \sum_{b=1}^{N} \mu_{nb}^{*(c)}(t)\right\} \leq 0 \tag{5}$$

Such a stationary and randomized algorithm that bases decisions only on $S(t)$ is called an *$S$-only algorithm*. It is often useful to assume that $(\lambda_n^{(c)})$ is *interior* to $\Lambda$, so that there is an $\epsilon > 0$ such that $(\lambda_n^{(c)} + \epsilon 1_n^{(c)}) \in \Lambda$, where $1_n^{(c)}$ is 1 if $n \neq c$, and zero else. In that case, there is an $S$-only algorithm that yields the following for all $n \neq c$:

$$\mathbb{E}\left\{\lambda_n^{(c)} + \sum_{a=1}^{N} \mu_{an}^{*(c)}(t) - \sum_{b=1}^{N} \mu_{nb}^{*(c)}(t)\right\} \leq -\epsilon \tag{6}$$

As a technical requirement, we assume that the second moments of transmission rates $\mu_{ab}(t)$ are finite under any algorithm for choosing these rates. This trivially holds if there is a finite maximum rate $\mu_{max}$.

## III. LYAPUNOV DRIFT AND BACKPRESSURE

The backpressure algorithm is designed by choosing transmission and routing variables to greedily minimize a bound on the drift of a Lyapunov function. Specifically, define $\boldsymbol{Q}(t) = (Q_n^{(c)}(t))$ as the matrix of current queue backlogs. Define the Lyapunov function:

$$L(t) \triangleq \frac{1}{2}\sum_{n=1}^{N}\sum_{c=1}^{N} Q_n^{(c)}(t)^2$$

This is just a sum of the squares of queue backlogs (multiplied by $1/2$ only for convenience in later analysis). The above sum is the same as summing over all $n, c$ such that $n \neq c$, because $Q_c^{(c)}(t) = 0$ for all $c \in \{1, \ldots, N\}$ and all slots $t$.

The *Lyapunov drift* $\Delta(t)$ is defined:

$$\Delta(t) \triangleq \mathbb{E}\left\{L(t+1) - L(t) | \boldsymbol{Q}(t)\right\}$$

By squaring the queue update equation (1) and using the fact that $(\max[q-b, 0]+a)^2 \leq q^2+b^2+a^2+2q(a-b)$ for $q \geq 0$, $a \geq 0$, $b \geq 0$, it is not difficult to show that for all slots $t$ and under any algorithm for choosing transmission and routing variables $(\mu_{ab}(t))$ and $(\mu_{ab}^{(c)}(t))$:

$$\Delta(t) \leq B + \sum_{n=1}^{N}\sum_{c=1}^{N} Q_n^{(c)}(t)\mathbb{E}\left\{\lambda_n^{(c)}(t) + \sum_{a=1}^{N}\mu_{an}^{(c)}(t) - \sum_{b=1}^{N}\mu_{nb}^{(c)}(t) | \boldsymbol{Q}(t)\right\} \tag{7}$$

where $B$ is a finite constant that depends on the second moments of arrivals and the maximum possible second moments of transmission rates.

### A. Minimizing the Drift Bound by Switching the Sums

The backpressure algorithm is designed to observe $\boldsymbol{Q}(t)$ and $S(t)$ every slot $t$ and choose $(\mu_{ab}(t))$ and $(\mu_{ab}^{(c)}(t))$ to minimize the right-hand-side of the drift bound (7). Because $B$ is a constant and $\lambda_n^{(c)}$ are constants, this amounts to maximizing:

$$\mathbb{E}\left\{\sum_{n=1}^{N}\sum_{c=1}^{N} Q_n^{(c)}(t)\left[\sum_{a=1}^{N}\mu_{an}^{(c)}(t) - \sum_{b=1}^{N}\mu_{nb}^{(c)}(t)\right] | \boldsymbol{Q}(t)\right\}$$

where we have pushed the finite sums through the expectations to illuminate the maximizing decision. By the principle of *opportunistically maximizing an expectation*, we can maximize the above expectation by maximizing the function inside of it (given the observed $\boldsymbol{Q}(t)$, $S(t)$). That is, we choose $(\mu_{ab}(t))$ and $(\mu_{ab}^{(c)}(t))$ subject to the constraints (2)-(4) to maximize:

$$\sum_{n=1}^{N}\sum_{c=1}^{N} Q_n^{(c)}(t)\left[\sum_{a=1}^{N}\mu_{an}^{(c)}(t) - \sum_{b=1}^{N}\mu_{nb}^{(c)}(t)\right]$$

It is not immediately obvious what decisions maximize the above. This can be illuminated by switching the sums. Indeed, the above expression is the same as below:

$$\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N}\mu_{ab}^{(c)}(t)[Q_a^{(c)}(t) - Q_b^{(c)}(t)]$$

The weight $Q_a^{(c)}(t) - Q_b^{(c)}(t)$ is called the current *differential backlog* of commodity $c$ between nodes $a$ and $b$. Thus, we want to choose decision variables $(\mu_{ab}^{(c)}(t))$ so as to maximize the above weighted sum, where weights are differential backlogs. Intuitively, this means that we want to allocate larger rates to commodities and links for which there is larger differential backlog.

Clearly we should not send data in a direction of negative differential backlog, and so we should choose $\mu_{ab}^{(c)}(t) = 0$ whenever $Q_a^{(c)}(t) - Q_b^{(c)}(t) < 0$. Further, given $\mu_{ab}(t)$ for a particular link $(a, b)$, it is not difficult to show that the optimal $\mu_{ab}^{(c)}(t)$ selections, subject to (3)-(4), are determined as follows: First find the commodity $c_{ab}^{opt}(t) \in$

$\{1, \ldots, N\}$ that *maximizes the differential backlog* for link $(a, b)$. We call $c_{ab}^{opt}(t)$ the *optimal commodity for link* $(a, b)$ *on slot* $t$. If this maximizing differential backlog is negative, assign $\mu_{ab}^{(c)}(t) = 0$ for all commodities $c \in \{1, \ldots, N\}$ on link $(a, b)$. Else, allocate the full link rate $\mu_{ab}(t)$ to the commodity $c_{ab}^{opt}(t)$, and zero rate to all other commodities on this link. With this choice, we have:

$$\sum_{c=1}^{N} \mu_{ab}^{(c)}(t)[Q_a^{(c)}(t) - Q_b^{(c)}(t)] = \mu_{ab}(t)W_{ab}(t)$$

where $W_{ab}(t)$ is the differential backlog of the optimal commodity for link $(a, b)$ on slot $t$ (maxed with 0):

$$W_{ab}(t) \triangleq \max[Q_a^{(c_{ab}^{opt}(t))}(t) - Q_b^{(c_{ab}^{opt}(t))}(t), 0]$$

It remains only to choose $(\mu_{ab}(t)) \in \Gamma_{S(t)}$. We do this by solving the following:

$$\text{Maximize:} \quad \sum_{a=1}^{N} \sum_{b=1}^{N} \mu_{ab}(t)W_{ab}(t) \tag{8}$$

$$\text{Subject to:} \quad (\mu_{ab}(t)) \in \Gamma_{S(t)} \tag{9}$$

Choosing transmission rates $(\mu_{ab}(t))$ according to (8)-(9) is called the *max-weight algorithm*. It maximizes a linear sum of transmission rates, where the weights $W_{ab}(t)$ correspond to differential backlogs. The *backpressure algorithm* uses the max-weight decisions for $(\mu_{ab}(t))$, and then chooses routing variables $(\mu_{ab}^{(c)}(t))$ via the maximum differential backlog as described above.

Note that once the transmission rates $(\mu_{ab}(t))$ have been selected, the routing decisions are extremely easy, and are distributed over each node (requiring only knowledge of queue backlogs in your own node and your neighbors). However, the max-weight problem (8)-(9) can be very difficult in networks with inter-channel interference. Often, an *approximation* of the max-weight transmission decision is used. It is known that using a $\gamma$-multiplicative approximation (where $0 < \gamma \leq 1$) stabilizes the network whenever arrival rates $(\lambda_n^{(c)})$ are within $\gamma\Lambda$, a $\gamma$-scaled version of the capacity region. However, using a $C$-additive approximation stabilizes the network for all rates in the capacity region. Thus, additive approximations do not reduce throughput (although they can increase delay), whereas multiplicative approximations can reduce throughput. The fact that we can be off by an additive constant is useful if we implement backpressure with out-of-date queue backlog information (see exercise 4.10 in [8]).

A remarkable property of the backpressure algorithm is that it acts greedily every slot $t$ based only on the observed topology state $S(t)$ and queue backlogs $\boldsymbol{Q}(t)$ for that slot. Thus, it does not require knowledge of the arrival rates $(\lambda_n^{(c)})$ or the topology state probabilities $\pi_S = Pr[S(t) = S]$.

### *B. Improving Delay*

It is important to note that the backpressure algorithm does not use any pre-specified paths. Paths are learned dynamically, and may be different for different packets. Delay can be very large, particularly when the system is lightly loaded so that there is not enough pressure to push data towards the destination. As an example, suppose one packet enters the network, and nothing else ever enters. This packet may take a loopy walk through the network and never arrive at its destination because no pressure gradients build up. This does not contradict the above analysis, because the network has at most one packet at any time and hence is trivially stable.

It is also possible to implement backpressure subject to some pre-specified paths that can be used. This can restrict the capacity region, but might improve in-order delivery and delay. Another way to improve delay, without affecting the capacity region, is to use the *enhanced* version of backpressure developed in [4], which adds a "differential progress-to-destination" term to the backpressure index $W_{ab}(t)$. Simulations of this are given in [7][9]. Note that the queue dynamics (1) do not require First-in-First-Out (FIFO) operation. It has been observed in implementations and analysis in [10][11] that using Last-in-First-Out (LIFO) service can dramatically improve delay for the vast majority of packets, without affecting throughput.

### *C. Performance Analysis*

Here we show that the backpressure algorithm stabilizes the multi-hop network. For simplicity, we use the basic algorithm defined above, assuming the full max-weight solution is obtained on every slot $t$ (analysis for $\gamma$-multiplicative algorithms is basically the same, where a factor $\gamma$ is multiplied in appropriate places).

Because the backpressure algorithm observes $\boldsymbol{Q}(t)$ and $S(t)$ every slot $t$ and chooses decisions $(\mu_{ab}(t))$ and $(\mu_{ab}^{(c)}(t))$ to minimize the right-hand-side of the drift bound (7), we have:

$$\Delta(t) \leq B + \sum_{n=1}^{N}\sum_{c=1}^{N} Q_n^{(c)}(t)\mathbb{E}\left\{\lambda_n^{(c)}(t) + \sum_{a=1}^{N}\mu_{an}^{*(c)}(t) - \sum_{b=1}^{N}\mu_{nb}^{*(c)}(t)|\boldsymbol{Q}(t)\right\} \quad (10)$$

where $(\mu_{ab}^*(t))$ and $(\mu_{ab}^{*(c)}(t))$ are any alternative decisions that satisfy (2)-(4), including randomized decisions.

Now assume $(\lambda_n^{(c)}) \in \Lambda$. Then there exists an $S$-only algorithm that satisfies (5). Plugging this into the right-hand-side of (10) and noting that the conditional expectation given $\boldsymbol{Q}(t)$ under this $S$-only algorithm is the same as the unconditional expectation (because $S(t)$ is i.i.d. over slots, and the $S$-only algorithm is independent of current queue backlogs) yields:

$$\Delta(t) \leq B$$

Thus, the drift of a quadratic Lyapunov function is less than or equal to a constant $B$ for all slots $t$. From this, we immediately conclude that all network queues are mean rate stable [8] (this also implies all network queues are rate stable [12]).

If we want a stronger understanding of average queue size, we must assume the arrival rates $(\lambda_n^{(c)})$ are interior to $\Lambda$, so there is an $\epsilon > 0$ such that (6) holds for some alternative $S$-only algorithm. Plugging (6) into the right-hand-side of (10) yields:

$$\Delta(t) \leq B - \epsilon\sum_{n=1}^{N}\sum_{c=1}^{N} Q_n^{(c)}(t)$$

from which we immediately obtain (see [7][8]):

$$\limsup_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{n=1}^{N}\sum_{c=1}^{N}\mathbb{E}\left\{Q_n^{(c)}(\tau)\right\} \leq \frac{B}{\epsilon}$$

It is interesting to note that this average queue size bound increases as the distance $\epsilon$ to the boundary of the capacity region $\Lambda$ goes to zero. This is the same qualitative performance as a single M/M/1 queue with arrival rate $\lambda$ and service rate $\mu$, where average queue size is proportional to $1/\epsilon$, where $\epsilon \triangleq \mu - \lambda$.

### *D. Non-I.I.D. Operation and Universal Scheduling*

The above analysis assumes i.i.d. properties for simplicity. The same algorithm can be shown to operate robustly in non-i.i.d. situations. When arrival processes and topology states are ergodic but not necessarily i.i.d., it still stabilizes the system whenever $(\lambda_n^{(c)}) \in \Lambda$ [4]. More generally, using a *universal scheduling* approach, it has been shown to offer stability and optimality properties for arbitrary (possibly non-ergodic) sample paths [13].

### *E. Distributed Backpressure*

Recall that the selection of routing variables $(\mu_{ab}^{(c)}(t))$ is distributed over each node once the rates $(\mu_{ab}(t))$ have been determined. The max-weight rate selection problem (8)-(9) can be distributed over each node in cases when the network has orthogonal channels, but is a centralized problem for general networks with inter-channel interference. A distributed approach to backpressure for ad-hoc mobile networks is given in [4], using a signal-to-interference ratio model. There, each node randomly decides to transmit every slot $t$ (transmitting a "null" packet if it currently does not have a packet to send). The actual transmission rates, and the corresponding actual packets to send, are determined by a 2-step handshake: On the first step, the randomly selected transmiter nodes send a pilot signal with signal strength proportional to that of an actual transmission. On the second step, all potential receiver nodes measure the resulting interference level and send that information back to the transmitters. The signal-to-interference levels for all outgoing links $(n, b)$ are then known to all nodes $n$, and each node $n$ can decide its $\mu_{nb}(t)$ and $(\mu_{nb}^{(c)}(t))$ variables based on this information.

The resulting throughput is not necessarily optimal because of the random transmission decisions. However, the random transmission process can be viewed as a part of the channel state process (provided that null packets are sent in cases of underflow, so that the channel state process does not depend on queue states). Hence, the resulting

throughput of this distributed implementation is optimal over the class of all routing and scheduling algorithms that use such randomized transmissions.

Alternative distributed implementations can roughly be grouped into two classes: The first class of algorithms consider constant multiplicative factor approximations, and yield constant-factor throughput results. The second class of algorithms consider additive approximations, based on updating solutions to the max-weight problem over time. The second class of algorithms seem to require static channel conditions and longer (often non-polynomial) convergence times, although they can provably achieve maximum throughput under appropriate assumptions. See Chapter 6 of [8] and references therein for more details on this.

### *F. Backpressure with Utility Optimization and Penalty Minimization*

Backpressure has been shown to work in conjunction with flow control in [3][6][7], where it yields performance within $O(1/V)$ of optimal throughput-utility, with $O(V)$ average delay, where $V$ is a control parameter that can be chosen as large as desired. This gives an explicit $[O(1/V), O(V)]$ utility-delay tradeoff. Similar properties are shown for average power minimization in [14] (see also a unified treatment in [8]). This is accomplished via a *drift-plus-penalty* approach. Specifically, while backpressure makes control decisions to greedily minimize a bound on the drift $\Delta(t)$ every sot $t$, the drift-plus-penalty algorithm makes control decisions to greedily minimize a bound on the following drift-plus-penalty expression:

$$\Delta(t) + V\mathbb{E}\left\{penalty(t)|\boldsymbol{Q}(t)\right\}$$

where $penalty(t)$ is related to the penalty whose time average is to be minimized. This reduces to the original backpressure algorithm when $V = 0$, but makes decisions that incorporate the penalty more heavily as $V$ is increased.

In the same way that minimizing a bound on $\Delta(t)$ leads to an expression $\Delta(t) \leq B$ whenever the network can be stabilized (as explained above), minimizing a bound on the drift-plus-penalty leads to an expression of the form:

$$\Delta(t) + V\mathbb{E}\left\{penalty(t)|\boldsymbol{Q}(t)\right\} \leq B + Vp^*$$

where $p^*$ is the optimal time average penalty, from which we immediately conclude that all queues are stable and time average penalty is within $B/V$ of the optimal value $p^*$ (see [8] for these details, and for showing average queue size is $O(V)$).

Alternative algorithms for stabilizing queues with utility optimization are developed in [15][16] using a fluid model analysis, in [17] using convex optimization, and in [18] using stochastic gradients. These approaches do not provide the $O(1/V), O(V)$ utility-delay results given above.

## REFERENCES


[1] L. Tassiulas and A. Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transacations on Automatic Control*, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.

[2] D. P. Bertsekas and R. Gallager. *Data Networks*. New Jersey: Prentice-Hall, Inc., 1992.

[3] M. J. Neely. *Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels*. PhD thesis, Massachusetts Institute of Technology, LIDS, 2003.

[4] M. J. Neely, E. Modiano, and C. E Rohrs. Dynamic power allocation and routing for time varying wireless networks. *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89-103, January 2005.

[5] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multicommodity flow. *Proc. 34th IEEE Conf. on Foundations of Computer Science*, Oct. 1993.

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

[7] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1-149, 2006.

[8] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[9] M. J. Neely and R. Urgaonkar. Optimal backpressure routing in wireless networks with multi-receiver diversity. *Ad Hoc Networks (Elsevier)*, vol. 7, no. 5, pp. 862-881, July 2009.

[10] S. Moeller, A. Sridharan, B. Krishnamachari, and O. Gnawali. Routing without routes: The backpressure collection protocol. *Proc. 9th ACM/IEEE Intl. Conf. on Information Processing in Sensor Networks (IPSN)*, April 2010.

[11] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari. LIFO-backpressure achieves near optimal utility-delay tradeoff. *Proc. WiOpt*, May 2011.

[12] M. J. Neely. Queue stability and probability 1 convergence via lyapunov optimization. *Arxiv Technical Report*, Oct. 2010.

[13] M. J. Neely. Universal scheduling for networks with arbitrary traffic, channels, and mobility. *Proc. IEEE Conf. on Decision and Control (CDC)*, Atlanta, GA, Dec. 2010.

[14] M. J. Neely. Energy optimal control for time varying wireless networks. *IEEE Transactions on Information Theory*, vol. 52, no. 7, pp. 2915-2934, July 2006.

[15] A. Eryilmaz and R. Srikant. Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control. *Proc. IEEE INFOCOM*, March 2005.

[16] A. Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, vol. 50, no. 4, pp. 401-457, 2005.

[17] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. *Proc. of 43rd IEEE Conf. on Decision and Control, Paradise Island, Bahamas*, Dec. 2004.

[18] J. W. Lee, R. R. Mazumdar, and N. B. Shroff. Opportunistic power scheduling for dynamic multiserver wireless systems. *IEEE Transactions on Wireless Communications*, vol. 5, no.6, pp. 1506-1515, June 2006.