WIKIPEDIA

Backpressure routing

In queueing theory, a discipline within the mathematical theory of probability, the **backpressure routing algorithm** is a method for directing traffic around a queueing network that achieves maximum network throughput,[1] which is established using concepts of Lyapunov drift. Backpressure routing considers the situation where each job can visit multiple service nodes in the network. It is an extension of max-weight scheduling where each job visits only a single service node.

Contents

Introduction

Backpressure routing is an algorithm for dynamically routing traffic over a multi-hop network by using congestion gradients. The algorithm can be applied to wireless communication networks, including sensor networks, mobile ad hoc networks (MANETS), and heterogeneous networks with wireless and wireline components.[2][3]

Backpressure principles can also be applied to other areas, such as to the study of product assembly systems and processing networks.[4] This article focuses on communication networks, where packets from multiple data streams arrive and must be delivered to appropriate destinations. The backpressure algorithm operates in slotted time. Every time slot it seeks to route data in directions that maximize the *differential backlog* between neighboring nodes. This is similar to how water flows through a network of pipes via pressure gradients. However, the backpressure algorithm can be applied to multi-commodity networks (where different packets may have different destinations), and to networks where transmission rates can be selected from a set of (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. However, the algorithm may introduce large delays, and may be difficult to implement exactly in networks with interference. Modifications of backpressure that reduce delay and simplify implementation are described below under Improving delay and Distributed backpressure.

Backpressure routing has mainly been studied in a theoretical context. In practice, ad hoc wireless networks have typically implemented alternative routing methods based on shortest path computations or network flooding, such as Ad Hoc on-Demand Distance Vector Routing (AODV), geographic routing, and extremely opportunistic routing (ExOR). However, the mathematical optimality properties of backpressure have motivated recent experimental demonstrations of its use on wireless testbeds at the University of Southern California and at North Carolina State University.[5][6][7]

Origins

The original backpressure algorithm was developed by Tassiulas and Ephremides.[2] They considered a multi-hop packet radio network with random packet arrivals and a fixed set of link selection options. Their algorithm consisted of a *max-weight link selection* stage and a *differential backlog routing* stage. An algorithm related to backpressure, designed for computing multi-commodity network flows, was developed in Awerbuch and Leighton.[8] The backpressure algorithm was later extended by Neely, Modiano, and Rohrs to treat scheduling for mobile networks.[9] Backpressure is mathematically analyzed via the theory of Lyapunov drift, and can be used jointly with flow control mechanisms to provide network utility maximization.[10][11][3][12][13] (see also Backpressure with utility optimization and penalty minimization).

How it works

Backpressure routing is designed to make decisions that (roughly) minimize the sum of squares of queue backlogs in the network from one timeslot to the next. The precise mathematical development of this technique is described in later sections. This section describes the general network model and the operation of backpressure routing with respect to this model.

The multi-hop queueing network model

Consider a multi-hop network with *N* nodes (see Fig. 1 for an example with *N* = 6). The network operates in slotted time $t \in \{0, 1, 2, \ldots\}$. On each slot, new data can arrive to the network, and routing and transmission scheduling decisions are made in an effort to deliver all data to its proper destination. Let data that is destined for node $c \in \{1, \ldots, N\}$ be labeled as *commodity c data*. Data in each node is stored according to its commodity. For $n \in \{1, \ldots, N\}$ and $c \in \{1, \ldots, N\}$, let $Q_n^{(c)}(t)$ be the current amount of commodity *c* data in node *n*, also called the *queue backlog*. A closeup of the queue backlogs inside a node is shown in Fig. 2. The units of $Q_n^{(c)}(t)$ depend on the context of the problem. For example, backlog can take integer units of *packets*, which is useful in cases when data is segmented into fixed length packets. Alternatively, it can take real valued units of *bits*. It is assumed that $Q_c^{(c)}(t) = 0$ for all $c \in \{1, \ldots, N\}$ and all timeslots *t*, because no node stores data destined for itself. Every timeslot, nodes can transmit data to others. Data that is transmitted from one node to another node is removed from the queue of the first node and added to the queue of the second. Data that is transmitted to its destination is removed from the network. Data can also arrive exogenously to the network, and $A_n^{(c)}(t)$ is defined as the amount of new data that arrives to node *n* on slot *t* that must eventually be delivered to node *c*.

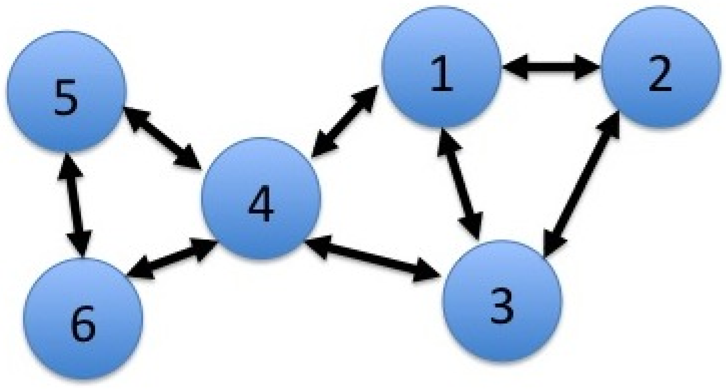

Fig. 1: A 6-node multihop network. Arrows between nodes illustrate current neighbors.

Let $\mu_{ab}(t)$ be the *transmission rate* used by the network over link (*a*,*b*) on slot *t*, representing the amount of data it can transfer from node *a* to node *b* on the current slot. Let $(\mu_{ab}(t))$ be the transmission rate matrix. These transmission rates must be selected within a set of possibly time-varying options. Specifically, the network may have time-varying channels and node mobility, and this can affect its transmission capabilities every slot. To model this, let *S*(*t*) represent the *topology state* of the network, which captures properties of the network on slot *t* that affect transmission. Let $\Gamma_{S(t)}$ represent the set of transmission rate matrix options available under topology state *S*(*t*). Every slot *t*, the network controller observes *S*(*t*) and chooses transmission rates $(\mu_{ab}(t))$ within the set $\Gamma_{S(t)}$. The choice of which $(\mu_{ab}(t))$ matrix to select on each slot *t* is described in the next subsection.

This time-varying network model was first developed for the case when transmission rates every slot t were determined by general functions of a channel state matrix and a power allocation matrix.[9] The model can also be used when rates are determined by other control decisions, such as server allocation, sub-band selection, coding type, and so on. It assumes the supportable transmission rates are known and there are no transmission errors. Extended formulations of backpressure routing can be used for networks with probabilistic channel errors, including networks that exploit the wireless broadcast advantage via *multi-receiver diversity*.[1]

The backpressure control decisions

Every slot *t* the backpressure controller observes *S*(*t*) and performs the following 3 steps:

- First, for each link (*a*,*b*), it selects an *optimal commodity* $c_{ab}^{opt}(t)$ to use.
- Next, it determines what $(\mu_{ab}(t))$ matrix in $\Gamma_{S(t)}$ to use.
- Finally, it determines the amount of commodity $c_{ab}^{opt}(t)$ it will transmit over link (*a*,*b*) (being at most $\mu_{ab}(t)$, but possibly being less in some cases).

Choosing the optimal commodity

Each node a observes its own queue backlogs and the backlogs in its current neighbors. A *current neighbor* of node a is a node b such that it is possible to choose a non-zero transmission rate $\mu_{ab}(t)$ on the current slot. Thus, neighbors are determined by the set $\Gamma_{S(t)}$. In the extreme case, a node can have all $N - 1$ other nodes as neighbors. However, it is common to use sets $\Gamma_{S(t)}$ that preclude transmissions between nodes that are separated by more than a certain geographic distance, or that would have a propagated signal strength below a certain threshold. Thus, it is typical for the number of neighbors to be much less than $N - 1$. The example in Fig. 1 illustrates neighbors by link connections, so that node 5 has neighbors 4 and 6. The example suggests a symmetric relationship between neighbors (so that if 5 is a neighbor of 4, then 4 is a neighbor of 5), but this need not be the case in general.

The set of neighbors of a given node determines the set of outgoing links it can use for transmission on the current slot. For each outgoing link (a,b), the *optimal commodity* $c_{ab}^{opt}(t)$ is defined as the commodity $c \in \{1, \ldots, N\}$ that maximizes the following *differential backlog* quantity:

$$Q_a^{(c)}(t) - Q_b^{(c)}(t)$$

Any ties in choosing the optimal commodity are broken arbitrarily.

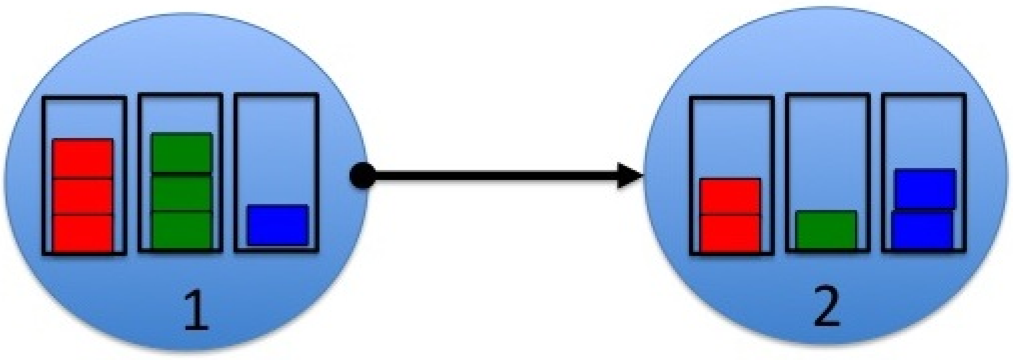

Fig. 2: A closeup of nodes 1 and 2. The optimal commodity to send over link (1,2) is the green commodity. The optimal commodity to send in the other direction (over link (2,1)) is the blue commodity.

An example is shown in Fig. 2. The example assumes each queue currently has only 3 commodities: *red*, *green*, and *blue*, and these are measured in integer units of packets. Focusing on the directed link (1,2), the differential backlogs are:

$$Q_1^{(\text{red})}(t) - Q_2^{(\text{red})}(t) = 1$$

$$Q_1^{(\text{green})}(t) - Q_2^{(\text{green})}(t) = 2$$

$$Q_1^{(\text{blue})}(t) - Q_2^{(\text{blue})}(t) = -1$$

Hence, the optimal commodity to send over link (1,2) on slot t is the green commodity. On the other hand, the optimal commodity to send over the reverse link (2,1) on slot t is the blue commodity.

Choosing the $\mu_{ab}(t)$ matrix

Once the optimal commodities have been determined for each link (a,b), the network controller computes the following weights $W_{ab}(t)$:

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

The weight $W_{ab}(t)$ is the value of the differential backlog associated with the optimal commodity for link (a,b), maxed with 0. The controller then chooses transmission rates as the solution to the following *max-weight* problem (breaking ties arbitrarily):

$$\text{(Eq. 1)} \qquad \text{Maximize: } \sum_{a=1}^{N}\sum_{b=1}^{N}\mu_{ab}(t)W_{ab}(t)$$

$$\text{(Eq. 2)} \qquad \text{Subject to: } (\mu_{ab}(t)) \in \Gamma_{S(t)}$$

As an example of the max-weight decision, suppose that on the current slot t, the differential backlogs on each link of the 6 node network lead to link weights $W_{ab}(t)$ given by:

$$(W_{ab}(t)) = \begin{bmatrix} 0 & 2 & 1 & 1 & 6 & 0 \\ 1 & 0 & 1 & 2 & 5 & 6 \\ 0 & 7 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 7 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \end{bmatrix}$$

While the set $\Gamma_{S(t)}$ might contain an uncountably infinite number of possible transmission rate matrices, assume for simplicity that the current topology state admits only 4 possible choices:

$$\Gamma_{S(t)} = \{\boldsymbol{\mu}_a, \boldsymbol{\mu}_b, \boldsymbol{\mu}_c, \boldsymbol{\mu}_d\}$$

illustration of the 4 possible transmission rate selections under the current topology state $S(t)$. Option (a) activates the single link (1,5) with a transmission rate of $\mu_{15} = 2$. All other options use two links, with transmission rates of 1 on each of the activated links.

These four possibilities are represented in matrix form by:

$$\boldsymbol{\mu}_a = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{\mu}_b = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{\mu}_c = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{\mu}_d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Observe that node 6 can neither send nor receive under any of these possibilities. This might arise because node 6 is currently out of communication range. The weighted sum of rates for each of the 4 possibilities are:

- Choice (a): $\sum_{ab} W_{ab}(t)\mu_{ab}(t) = 12$.
- Choice (b): $\sum_{ab} W_{ab}(t)\mu_{ab}(t) = 1$.
- Choice (c): $\sum_{ab} W_{ab}(t)\mu_{ab}(t) = 1$.
- Choice (d): $\sum_{ab} W_{ab}(t)\mu_{ab}(t) = 12$.

Because there is a tie for the maximum weight of 12, the network controller can break the tie arbitrarily by choosing either option $\boldsymbol{\mu}_a$ or option $\boldsymbol{\mu}_d$.

Finalizing the routing variables

Suppose now that the optimal commodities $c_{ab}^{opt}(t)$ have been determined for each link, and the transmission rates $(\mu_{ab}(t))$ have also been determined. If the differential backlog for the optimal commodity on a given link (*a*,*b*) is negative, then no data is transferred over this link on the current slot. Else, the network offers to send $\mu_{ab}(t)$ units of commodity $c_{ab}^{\text{opt}}(t)$ data over this link. This is done by defining *routing variables* $\mu_{ab}^{(c)}(t)$ for each link (*a*,*b*) and each commodity *c*, where:

$$\mu_{ab}^{(c)}(t) = \begin{cases} \mu_{ab}(t) & \text{if } c = c_{ab}^{opt}(t) \text{ and } Q_a^{(c_{ab}^{opt}(t))}(t) - Q_b^{(c_{ab}^{opt}(t))}(t) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The value of $\mu_{ab}^{(c)}(t)$ represents the transmission rate offered to commodity *c* data over link (*a*,*b*) on slot *t*. However, nodes might not have enough of a certain commodity to support transmission at the offered rates on all of their outgoing links. This arises on slot *t* for node *n* and commodity *c* if:

$$Q_n^{(c)}(t) < \sum_{b=1}^{N} \mu_{nb}^{(c)}(t)$$

In this case, all of the $Q_n^{(c)}(t)$ data is sent, and null data is used to fill the unused portions of the offered rates, allocating the actual data and null data arbitrarily over the corresponding outgoing links (according to the offered rates). This is called a *queue underflow* situation. Such underflows do not affect the throughput or stability properties of the network. Intuitively, this is because underflows only arise when the transmitting node has a low amount of backlog, which means the node is not in danger of instability.

Improving delay

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 throughput optimality or stability properties of backpressure because the network has at most one packet at any time and hence is trivially stable (achieving a delivery rate of 0, equal to the arrival rate).

It is also possible to implement backpressure on a set of pre-specified paths. 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 an *enhanced* version that biases link weights towards desirable directions.[9] Simulations of such biasing have shown significant delay improvements.[1][3] Note that backpressure does not require First-in-First-Out (FIFO) service at the queues. It has been observed that Last-in-First-Out (LIFO) service can dramatically improve delay for the vast majority of packets, without affecting throughput.[7][14]

Distributed backpressure

Note that once the transmission rates $(\mu_{ab}(t))$ have been selected, the routing decision variables $\mu_{ab}^{(c)}(t)$ can be computed in a simple distributed manner, where each node only requires knowledge of queue backlog differentials between itself and its neighbors. However, selection of the transmission rates requires a solution to the max-weight problem in Eqs. (1)-(2). In the special case when channels are orthogonal, the algorithm has a natural distributed implementation and reduces to separate decisions at each node. However, the max-weight problem is a centralized control problem for networks with inter-channel interference. It can also be very difficult to solve even in a centralized way.

A distributed approach for interference networks with link rates that are determined by the signal-to-noise-plus-interefernce ratio (SINR) can be carried out using randomization.[9] 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 transmitter 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 and send that information back to the transmitters. The SINR 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. 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 past decisions). 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 to the max-weight problem, and yield constant-factor throughput results. The second class of algorithms consider additive approximations to the max-weight problem, based on updating solutions to the max-weight problem over time. Algorithms in this second class seem to require static channel conditions and longer (often non-polynomial) convergence times, although they can provably achieve maximum throughput under appropriate assumptions.[15][4][13] Additive approximations are often useful for proving optimality of backpressure when implemented with out-of-date queue backlog information (see Exercise 4.10 of the Neely text).[13]

Mathematical construction via Lyapunov drift

This section shows how the backpressure algorithm arises as a natural consequence of greedily minimizing a bound on the change in the sum of squares of queue backlogs from one slot to the next.[9][3]

Control decision constraints and the queue update equation

Consider a multi-hop network with N nodes, as described in the above section. Every slot t, the network controller observes the topology state $S(t)$ and chooses transmission rates $(\mu_{ab}(t))$ and routing variables $(\mu_{ab}^{(c)}(t))$ subject to the following constraints:

$$\text{(Eq. 3)} \qquad (\mu_{ab}(t)) \in \Gamma_{S(t)}$$

$$\text{(Eq. 4)} \qquad 0 \leq \mu_{ab}^{(c)}(t) \qquad \forall a, b, c, \forall t$$

$$\text{(Eq. 5)} \qquad \sum_{c=1}^{N} \mu_{ab}^{(c)}(t) \leq \mu_{ab}(t) \qquad \forall (a,b), \forall t$$

Once these routing variables are determined, transmissions are made (using idle fill if necessary), and the resulting queue backlogs satisfy the following:

$$\text{(Eq. 6)} \qquad Q_n^{(c)}(t+1) \leq \max\left[Q_n^{(c)}(t) - \sum_{b=1}^{N} \mu_{nb}^{(c)}(t), 0\right] + \sum_{a=1}^{N} \mu_{an}^{(c)}(t) + A_n^{(c)}(t)$$

where $A_n^{(c)}(t)$ is the random amount of new commodity c data that exogenously arrives to node n on slot t, and $\mu_{nb}^{(c)}(t)$ is the transmission rate allocated to commodity c traffic on link (n,b) on slot t. Note that $\mu_{nb}^{(c)}(t)$ may be more than the amount of commodity c data that is actually transmitted on link (a,b) on slot t. This is because there may not be enough backlog in node n. For this same reason, Eq. (6) 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 Eq. (6) is that it holds even if the $\mu_{ab}^{(c)}(t)$ decision variables are chosen independently of queue backlogs.

It is assumed 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.

Lyapunov drift

Define $\boldsymbol{Q}(t) = (Q_n^{(c)}(t))$ as the matrix of current queue backlogs. Define the following non-negative function, called a Lyapunov function:

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

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

$$\Delta(t) = E\left[L(t+1) - L(t) \mid \boldsymbol{Q}(t)\right]$$

Note that the following inequality holds for all $q \geq 0, a \geq 0, b \geq 0$:

$$(\max[q-b, 0] + a)^2 \leq q^2 + b^2 + a^2 + 2q(a-b)$$

By squaring the queue update equation (Eq. (6)) and using the above inequality, 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 $(\mu_{ab}^{(c)}(t))$:[3]

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

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

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 Eq. (7). Because B is a constant and $\lambda_n^{(c)}$ are constants, this amounts to maximizing:

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

where the finite sums have been pushed through the expectations to illuminate the maximizing decision. By the principle of *opportunistically maximizing an expectation*, the above expectation is maximized by maximizing the function inside of it (given the observed $\boldsymbol{Q}(t)$, $S(t)$). Thus, one chooses $(\mu_{ab}(t))$ and $(\mu_{ab}^{(c)}(t))$ subject to the constraints Eqs. (3)-(5) to maximize:

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

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. The idea is to choose decision variables $(\mu_{ab}^{(c)}(t))$ so as to maximize the above weighted sum, where weights are differential backlogs. Intuitively, this means allocating larger rates in directions of larger differential backlog.

Clearly one 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 Eqs. (3)-(5), 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*). If the maximizing differential backlog is negative for link (*a*,*b*), 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, it follows that:

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

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

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

It remains only to choose $(\mu_{ab}(t)) \in \Gamma_{S(t)}$. This is done by solving the following:

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

$$\text{Subjectto} : (\mu_{ab}(t)) \in \Gamma_{S(t)}$$

The above problem is identical to the max-weight problem in Eqs. (1)-(2). 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.

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 $\pi_S = Pr[S(t) = S]$.

Performance analysis

This section proves throughput optimality of the backpressure algorithm.[3][13] For simplicity, the scenario where events are independent and identically distributed (i.i.d.) over slots is considered, although the same algorithm can be shown to work in non-i.i.d. scenarios (see below under Non-i.i.d. operation and universal scheduling).

Dynamic arrivals

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

$$\lambda_n^{(c)} = E\left[A_n^{(c)}(t)\right]$$

It is assumed 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.

Network capacity region

Assume the topology state $S(t)$ is i.i.d. over slots with probabilities $\pi_S = Pr[S(t) = S]$ (if $S(t)$ takes values in an uncountably infinite set of vectors with real-valued entries, then π_S is a probability distribution, not a probability mass function). A general algorithm for the network observes $S(t)$ every slot t and chooses transmission rates $(\mu_{ab}(t))$ and routing variables $(\mu_{ab}^{(c)}(t))$ according to the constraints in Eqs. (3)-(5). The *network capacity region* Λ is the closure of the set of all arrival rate matrices $(\lambda_n^{(c)})$ for which there exists an algorithm that stabilizes the network. 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. It can be shown that for any arrival rate matrix $(\lambda_n^{(c)})$ in the capacity region Λ, 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 $n \neq c$:[9][13]

$$\text{(Eq. 8)} \qquad E\left[\lambda_n^{(c)} + \sum_{a=1}^{N} \mu_{an}^{*(c)}(t) - \sum_{b=1}^{N} \mu_{nb}^{*(c)}(t)\right] \leq 0$$

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 Λ, 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 $n \neq c$:

$$\text{(Eq. 9)} \qquad E\left[\lambda_n^{(c)} + \sum_{a=1}^{N} \mu_{an}^{*(c)}(t) - \sum_{b=1}^{N} \mu_{nb}^{*(c)}(t)\right] \leq -\epsilon$$

As a technical requirement, it is assumed 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 μ_{max}.

Comparing to S-only algorithms

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 Eq. (7), we have:

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

where $(\mu_{ab}^*(t))$ and $(\mu_{ab}^{*(c)}(t))$ are any alternative decisions that satisfy Eqs. (3)-(5), including randomized decisions.

Now assume $(\lambda_n^{(c)}) \in \Lambda$. Then there exists an S-only algorithm that satisfies Eq. (8). Plugging this into the right-hand-side of Eq. (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. This fact, together with the assumption that queue arrivals have bounded second moments, imply the following for all network queues:[16]

$$\lim_{t\to\infty} \frac{Q_n^{(c)}(t)}{t} = 0 \text{ with probability } 1$$

For a stronger understanding of average queue size, one can assume the arrival rates $(\lambda_n^{(c)})$ are interior to Λ, so there is an $\epsilon > 0$ such that Eq. (9) holds for some alternative S-only algorithm. Plugging Eq. (9) into the right-hand-side of Eq. (10) yields:

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

from which one immediately obtains (see[3][13]):

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

This average queue size bound increases as the distance ϵ to the boundary of the capacity region Λ goes to zero. This is the same qualitative performance as a single M/M/1 queue with arrival rate λ and service rate μ, where average queue size is proportional to $1/\epsilon$, where $\epsilon = \mu - \lambda$.

Extensions of the above formulation

Non-i.i.d. operation and universal scheduling

The above analysis assumes i.i.d. properties for simplicity. However, the same backpressure 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., backpressure still stabilizes the system whenever $(\lambda_n^{(c)}) \in \Lambda$.[9] More generally, using a *universal scheduling* approach, it has been shown to offer stability and optimality properties for arbitrary (possibly non-ergodic) sample paths.[17]

Backpressure with utility optimization and penalty minimization

Backpressure has been shown to work in conjunction with flow control via a drift-plus-penalty technique.[10][11][3] This technique greedily maximizes a sum of drift and a weighted penalty expression. The penalty is weighted by a parameter V that determines a performance tradeoff. This technique ensures throughput utility is within $O(1/V)$ of optimality while average delay is $O(V)$. Thus, utility can be pushed arbitrarily close to optimality, with a corresponding tradeoff in average delay. Similar properties can be shown for average power minimization[18] and for optimization of more general network attributes.[13]

Alternative algorithms for stabilizing queues while maximizing a network utility have been developed using fluid model analysis,[12] joint fluid analysis and Lagrange multiplier analysis,[19] convex optimization,[20] and stochastic gradients.[21] These approaches do not provide the $O(1/V)$, $O(V)$ utility-delay results.

See also

- AODV
- Diversity backpressure routing (DIVBAR)[1]
- Drift plus penalty
- ExOR
- Geographic routing
- List of ad hoc routing protocols
- Lyapunov optimization

References

1. 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.
2. L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks, *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.
3. 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.
4. L. Jiang and J. Walrand. *Scheduling and Congestion Control for Wireless and Processing Networks*, Morgan & Claypool, 2010.
5. A. Sridharan, S. Moeller, and B. Krishnamachari, "Making Distributed Rate Control using Lyapunov Drifts a Reality in Wireless Sensor Networks," 6th Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt), April 2008.
6. A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: Practical Differential Backlog Congestion Control for Wireless Networks," Proc. IEEE INFOCOM, Rio de Janeiro, Brazil, 2009.
7. 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.
8. 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.
9. 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.
10. M. J. Neely. Dynamic Power Allocation and Routing for Satellite and Wireless Networks with Time Varying Channels. Ph.D. Dissertation, Massachusetts Institute of Technology, LIDS. November 2003.
11. M. J. Neely, E. Modiano, and C. Li, "Fairness and Optimal Stochastic Control for Heterogeneous Networks," Proc. IEEE INFOCOM, March 2005.
12. A. Stolyar, "Maximizing Queueing Network Utility subject to Stability: Greedy Primal-Dual Algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401-457, 2005.
13. M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems,* Morgan & Claypool, 2010.
14. L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "LIFO-Backpressure Achieves Near Optimal Utility-Delay Tradeoff," Proc. WiOpt, May 2011.

15. E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," Proc. ACM SIGMETRICS, 2006.
16. M. J. Neely, "Queue Stability and Probability 1 Convergence via Lyapunov Optimization," Journal of Applied Mathematics, vol. 2012, doi:10.1155/2012/831909.
17. 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.
18. 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
19. A. Eryilmaz and R. Srikant, "Fair Resource Allocation in Wireless Networks using Queue-Length-Based Scheduling and Congestion Control," Proc. IEEE INFOCOM, March 2005.
20. 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.
21. 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.

Primary sources

- L. Tassiulas and A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
- 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.
- M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*, Morgan & Claypool, 2010.

Retrieved from "https://en.wikipedia.org/w/index.php?title=Backpressure_routing&oldid=885384609"

This page was last edited on 27 February 2019, at 18:30 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.