Discounted Integral Priority Routing For Data Networks

Michael Zargham†, Alejandro Ribeiro†, Ali Jadbabaie†

Abstract—A Discounted Integral Priority (DIP) packet routing algorithm is presented. The method is derived for the network flow model of packet routing used for the derivation of back-pressure type methods. Unlike back-pressure type methods, DIP routing is designed to reduce the queue lengths rather than simply stabilize them. Our work leverages time discounted integral control to generate an adaptive packet routing algorithm which significantly outperforms its optimization motivated counterparts. Our numerical experiments demonstrate fast convergence and significantly smaller steady state queue backlogs as compared with Soft Backpressure and Accelerated Backpressure.

I. INTRODUCTION

This paper considers the problem of joint routing and scheduling in packet networks. Packets are accepted from upper layers as they are generated and marked for delivery to intended destinations. To accomplish delivery of information nodes need to determine routes and schedules capable of accommodating the generated traffic. From a node-centric point of view, individual nodes handle packets that are generated locally as well as packets received from neighboring nodes. The goal of each node is to determine suitable next hops for each flow conducive to successful packet delivery.

The study of the joint scheduling and routing problem, has been built up from the Backpressure (BP) algorithm, [1]. In BP, nodes keep track of the number of packets in their local queues for each flow and share this information with neighboring agents. Nodes compute the differences between the number of packets in their queues and the number of packets in neighboring queues for all flows and assign the transmission capacity of the link to the flow with the largest queue differential. An alternative interpretation of BP is as a dual stochastic subgradient descent algorithm [2], [3]. This leads to a model of the joint scheduling and routing problem as an feasibility-type optimization of per-flow routing variables that satisfy link capacity and flow conservation constraints. Considering the Lagrange dual problem generates distributed algorithms to find stable operating points of wired [4]–[6] and wireless communication networks [7]–[9] and improve convergence. In, [12], Accelerated Backpressure (ABP) extends this idea of SBP further by choosing a strongly convex objective and implementing Accelerated Dual Descent (ADD), a distributed approximation to Newton’s Method in place of gradient descent, [13]. SBP and ABP solve these stabilization problems much faster than BP but these algorithms but are unable to effectively clear backlogged queues even if the capacity to do so is available in the network.

In this paper, we shift into more control oriented thinking and observe that in order to eliminate large queues in steady state an integral control term is necessary. In developing Discounted Integral Priority (DIP) routing, we work within the Soft Backpressure framework but rather than using the queues themselves as routing priorities, we use a time discounted sum of the queue history. These priorities can be computed locally using a simple linear update combining the existing priorities and the newly observed queue lengths. Our method can be connected to a class of heavy ball methods like those introduced in [14]. One of the main challenges in reducing the queues rather than just stabilizing them is the stochastic nature of the packet arrivals. Some related work includes noise cancelation in [15] and adaptive gradient methods in [16].

II. PRELIMINARIES

Consider a given network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links between nodes. Denote as $C_{ij}$ the capacity of link $(i,j) \in \mathcal{E}$ and define the neighborhood of $i$ as the set $n_i = \{ j \in \mathcal{V} | (i,j) \in \mathcal{E} \}$ of nodes $j$ that can communicate directly with $i$. There is also a set of information flows $\mathcal{K}$ with the destination of flow $k \in \mathcal{K}$ being the node $o_k \in \mathcal{V}$. Let $n = |\mathcal{V}|$ be the number of nodes and $K = |\mathcal{K}|$ be the number of information flows in the network and $E = |\mathcal{E}|$ be the number of edges in the network. Define the $n \times E$ matrix $A$ to be the incidence matrix of the graph $\mathcal{G}$ and the reduced incidence matrix $A_k$ as the $(n-1) \times E$ matrix with the row associated with destination node $o_k$ removed. The block diagonal matrix $\bar{A} = \text{diag}[A_k]$ is an $(n-1) \times E \cdot K$ incidence matrix encoding the interrelation of all information flows and nodes in the network.

At time index $t$ terminal $i \neq o_k$ generates a random number

$$a_i^k(t) = a_i^k + \nu_i^k(t)$$

of units of information to be delivered to $o_k$. The random variables $a_i^k(t) \geq 0$ are generated by $\nu_i^k(t)$ which are independent and identically distributed across time with $\mathbb{E}[\nu_i^k(t)] = 0$ and finite support. Thus the expected value $\mathbb{E}[a_i^k(t)] = a_i^k$. At time index $t$ the system state $r_{ij}^k(t) \geq 0$ units of information through neighboring node $j \in n_i$ and receives $r_{ji}^k(t) \geq 0$ packets from neighbor $j$. This research is supported by Army Research Lab MAST Collaborative Technology Alliance, AFOSR complex networks program, ARO-P-57920-NS, NSF CAREER CCF-0952867, and NSF CCF-1017454, ONR MURI N000140810747 and NSF-ECS-0347285.

†Michael Zargham, Alejandro Ribeiro and Ali Jadbabaie are with the Department of Electrical and Systems Engineering, University of Pennsylvania.
The difference between the total number of received packets \( a^k(t) + \sum_{j \in n_i} r^k_{ji}(t) \) and the sum of transmitted packets \( \sum_{j \in n_i} r^k_{ij}(t) \) is added to the local queue – or subtracted if this quantity is negative. Therefore, the number \( q^k_i(t) \) of \( k \)-flow packets queued at node \( i \) evolves according to

\[
q^k_i(t + 1) = \left[ a^k_i(t) + \sum_{j \in n_i} r^k_{ji}(t) - r^k_{ij}(t) \right]^+ ,
\]

where the projection \([·]^+\) into the nonnegative reals is necessary because the number of packets in queue cannot become negative. We remark that (2) is stated for all nodes \( i \neq o_k \) because packets routed to their destinations are removed from the system.

To ensure packet delivery it is sufficient to guarantee that all queues \( q^k_i(t) \) remain stable. In turn, this can be guaranteed if the average rate at which packets exit queues does not exceed the rate at which packets are loaded into them. To state this formally observe that the time average limit of arrivals satisfies \( \lim_{t \to \infty} a^k_i(t) = \mathbb{E} [a^k_i(t)] := a^k_i \) and define the ergodic limit \( r^k_{ij} := \lim_{t \to \infty} r^k_{ij}(t) \). If the processes \( r^k_{ij}(t) \) controlling the movement of information through the network are asymptotically stationary, queue stability follows if

\[
\sum_{j \in n_i} r^k_{ij} - r^k_{ji} \geq a^k_i \quad \forall \; k, i \neq o_k .
\]

(3)

For future reference define the vector \( r := \{ r^k_{ij} \}_{k,(i,j)} \) grouping variables \( r^k_{ij} \) for all information flows and links. Since at most \( C_{ij} \) packets can be transmitted in link \((i, j)\) the routing variables \( r^k_{ij}(t) \) always satisfy the capacity constraints on the network,

\[
\sum_{k} r^k_{ij}(t) \leq C_{ij}
\]

which defines the set of possible routings

\[
C = \{ r \in \mathbb{R}^{K \times E} : r_{ij}(t) \leq C_{ij} \forall (i, j) \in E \}.
\]

(5)

The joint routing and scheduling problem can be now formally stated as the determination of nonnegative variables \( r(t) \in C \) that satisfy (4) for all times \( t \) and whose time average limits \( r^k_{ij} \) satisfy (3). The BP algorithm solves this problem by assigning all the capacity of the link \((i, j)\) to the flow with the largest queue differential \( q^k_i(t) - q^k_j(t) \). Specifically, for each link we determine the flow pressure

\[
K^* := \arg \max_k \left[ q^k_i(t) - q^k_j(t) \right]^+ .
\]

(6)

If the maximum pressure \( \max_k \left[ q^k_i(t) - q^k_j(t) \right]^+ > 0 \) is strictly positive we set \( r^k_{ij}(t) = C_{ij} \) for \( k = K^* \). Otherwise the link remains idle during the time frame. The backpressure algorithm works by observing the queue differentials on each link and then assigning the capacity for each link to the data type with the largest positive queue differential, thus driving the time average of the queue differentials to zero-stabilizing the queues. To generalize, we reinterpret BP as a dual stochastic subgradient descent.

A. Dual stochastic subgradient descent

Since the parameters that are important for queue stability are the time averages of the routing variables \( r^k_{ij}(t) \) an alternative view of the joint routing and scheduling problem is the determination of variables \( r^k_{ij}(t) \) satisfying (3) and \( \sum_k r^k_{ij} \leq C_{ij} \). This can be formulated as the solution of an optimization problem. Let \( f^k_{ij}(r^k_{ij}) \) be any concave function on \( \mathbb{R}_+ \) and consider the optimization problem

\[
r^*_r := \arg \max_{r \in C} \sum_{k, i, j} f^k_{ij}(r^k_{ij})
\]

(7)

s.t. \( \sum_{j \in n_i} r^k_{ij} - r^k_{ji} \geq a^k_i, \; \forall \; k, i \neq o_k \),

where the domain is a the convex polyhedron defined in (5). Since only feasibility is important for queue stability, solutions to (7) ensure stable queues irrespectively of the objective functions \( f^k_{ij}(r^k_{ij}) \). For notational compactness, define \( f(r) = \sum_{k, (i,j)} f^k_{ij}(r^k_{ij}) \).

Since the problem in (7) is concave it can be solved by descending on the dual domain. Start by associating multipliers \( \lambda^k \) with the constraint \( \sum_{j \in n_i} r^k_{ij} - r^k_{ji} \geq a^k_i \) and keep the constraint \( r \in C \) implicit. The corresponding Lagrangian associated with the optimization problem in (7) is

\[
\mathcal{L}(r, \lambda) := \sum_{k, i \neq o_k, j} f^k_{ij}(r^k_{ij}) + \sum_{k, i \neq o_k} \lambda^k \left( \sum_{j \in n_i} r^k_{ij} - r^k_{ji} - a^k_i \right)
\]

(8)

where we introduced the vector \( \lambda := \{ \lambda^k \}_{k \neq o_k} \) grouping variables \( \lambda^k \) for all flows and nodes. The corresponding dual function is defined as

\[
h(\lambda) := \max_{r \in C} \mathcal{L}(r, \lambda).
\]

(9)

To compute a descent direction for \( h(\lambda) \), compute the primal Lagrangian maximizers for given \( \lambda \) according to the vector function \( R : \mathbb{R}^{(n-1)K} \to C \) defined

\[
R(\lambda) := \arg \max_{\lambda \in C} \mathcal{L}(r, \lambda).
\]

(10)

The individual elements \( R^k_{ij}(\lambda) \) are ordered by stacking the \( E \) dimensional subvectors for each information flow \( k \). A descent direction for the dual function is available in the form of the dual subgradient\(^2\) obtained by evaluating the constraint slack associated with the Lagrangian maximizers

\[
\nabla h^k(\lambda) := \sum_{j \in n_i} \left( R^k_{ij}(\lambda) - R^k_{ji}(\lambda) - a^k_i \right).
\]

(11)

Since the Lagrangian \( \mathcal{L}(r, \lambda) \) in (8) is linear in the dual variables \( \lambda^k \) the determination of the maximizers \( R^k_{ij}(\lambda) := \arg \max_{\lambda \in C} \mathcal{L}(r, \lambda) \) can be decomposed into the maximization of separate summands. Considering the coupling constraints \( \sum_k r^k_{ij} \leq C_{ij} \) imposed by the domain \( C \) it suffices to consider variables \( \{ r^k_{ij} \}_k \) for all flows across a given link.

\(^1\)Stability is guaranteed only if the inequalities hold in a strict sense, i.e., \( \sum_{j \in n_i} r^k_{ij} - r^k_{ji} > a^k_i \). Equality is allowed here to facilitate connections with optimization problems to be considered later.

\(^2\)For an appropriately chosen cost function \( f(r) \), the subgradient is unique which motives the use of gradient notation \( \nabla h(\lambda) \), (see Proposition 1).
After reordering terms it follows that we can compute routes link wise:
\[
R_{ij}^k(\lambda) = \arg\max_{r_{ij}^k \geq 0} \sum_k f_{ij}^k(r_{ij}^k) + r_{ij}^k(\lambda_i^k - \lambda_j^k) \tag{12}
\]
\[
s.t. \quad \sum_k r_{ij}^k \leq C_{ij}
\]
for each link \((i, j) \in E\). Introducing a time index \(t\), subgradients \(\nabla h^k(\lambda_i)\) could be computed using (11) with Lagrangian maximizers \(R_{ij}^k(\lambda_i)\) given by (12). The dual subgradient may also be specified in vector form:
\[
\nabla h(\lambda) = \bar{A}R(\lambda) - a \tag{13}
\]
which is more convenient notion for analysis. A subgradient descent iteration could then be defined to find the variables \(r^*\) that solve (7) via a dual method which generates a sequence \(\lambda_t\); see e.g., [17].

The problem in computing \(\nabla h^k(\lambda_i)\) is that we don’t know the average arrival rates \(a_k^i\). We do observe, however, the instantaneous rates \(a_k^i(t)\) that are known to satisfy \(E[a_k^i(t)] = a_k^i\). Therefore,
\[
[g_t(\lambda)]^k := \sum_{j \in n_i} R_{ij}^k(\lambda) - R_{ji}^k(\lambda) - a_k^i(t), \tag{14}
\]
is a stochastic subgradient of the dual function in the sense that its expected value \(E[g_t(\lambda)] = \nabla h^k(\lambda)\) is the subgradient defined in (11). Stated in vector form \(g_t(\lambda) = \bar{A}R(\lambda) - a - \nu_t\), where \(a_t = a + \nu_t\) and \(E[\nu_t] = 0\) for all \(t\). We can then minimize the dual function using a stochastic subgradient descent algorithm. At time \(t\) we have multipliers \(\lambda_t\) and determine Lagrangian maximizers \(r_{ij}^k(t) := R_{ij}^k(\lambda_t)\) as per (12). We then proceed to update multipliers along the stochastic subgradient direction according to
\[
\lambda^k_i(t+1) = \left[\lambda^k_i(t) - \epsilon \left(\sum_{j \in n_i} r_{ij}^k(t) - r_{ji}^k(t) - a_k^i(t)\right)\right]^+, \tag{15}
\]
where \(\epsilon\) is a constant stepsize chosen small enough so as to ensure convergence; see e.g., [11].

Properties of the descent algorithm in (15) vary with the selection of the functions \(f_{ij}^k(r_{ij}^k)\). Two cases of interest are when \(f_{ij}^k(r_{ij}^k) = 0\) and when \(f_{ij}^k(r_{ij}^k)\) are continuously differentiable, strongly convex, and monotone decreasing on \(\mathbb{R}^+\) but otherwise arbitrary. The former allows recovery of the Backpressure Algorithm while the latter leads to the Soft Backpressure algorithm.

### B. Soft backpressure

Assume now that the functions \(f_{ij}^k(r_{ij}^k)\) are continuously differentiable, strongly convex, and monotone decreasing on \(\mathbb{R}^+\) but otherwise arbitrary. In this case the derivatives \(\partial f_{ij}^k(x)/\partial x\) of the functions \(f_{ij}^k(x)\) are monotonically increasing and thus have inverse functions that we denote as
\[
F_{ij}^k(x) := \left[\partial f_{ij}^k(x)/\partial x\right]^{-1}(x). \tag{16}
\]
The Lagrangian maximizers in (12) can be explicitly written in terms of the derivative inverses \(F_{ij}^k(x)\). Furthermore, the maximizers are unique for all \(\lambda\) implying that the dual function is differentiable. The details are outlined in Proposition 1 originally published in [12].

**Proposition 1.** If the functions \(f_{ij}^k(r_{ij}^k)\) in (7) are continuously differentiable, strongly concave, and monotone decreasing on \(\mathbb{R}^+\), the dual function \(h(\lambda) := \max_{x \in C} \mathcal{L}(r, \lambda)\) is differentiable for all \(\lambda\). Furthermore, the gradient component along the \(\lambda^k_i\) direction is \(g_{ij}^k(\lambda)\) as defined in (14) with
\[
R_{ij}^k(\lambda) = F_{ij}^k\left(\left[-[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)]^+\right]\right), \tag{17}
\]
where \(\mu_{ij}(\lambda)\) is either 0 if \(\sum_k F_{ij}^k\left([-[\lambda_i^k - \lambda_j^k]^+]\right) \leq C_{ij}\) or chosen as the solution to the equation
\[
\sum_k F_{ij}^k\left([-[\lambda_i^k - \lambda_j^k - \mu_{ij}(\lambda)]^+]\right) = C_{ij}. \tag{18}
\]

While (18) does not have a closed for solution it can be computed quickly numerically using a binary search because it is a simple single variable root finding problem. Computation time cost remains small compared to communication time cost.

The Soft backpressure can be implemented using node level protocols. At each time instance nodes send their multipliers \(\lambda^k_i(t)\) to their neighbors. After receiving multiplier information from its neighbors, each node can compute the multiplier differentials \(\lambda^k_i(t) - \lambda^k_j(t)\) for each edge. The nodes then solve for \(\mu_{ij}\) on each of its outgoing edges by using a rootfinder to solve the local constraint in (18). The capacity of each edge is then allocated to the unique information flows via reverse waterfiling as defined in (17). Once the transmission rates are set each node can observe its net packet gain which is equivalent to the stochastic gradient as defined in (14). Finally, each node updates its multipliers by subtracting \(\epsilon\) times the stochastic subgradient from its current multipliers. As with BP, choosing the stepsize \(\epsilon = 1\) causes the multipliers to coincide with the queue lengths for all time.

### C. A General Priority-Based Routing Strategy

The Backpressure and and Soft Backpressure algorithms, as detailed in the preceding sections, are fundamentally priority based routing strategies; see e.g., [11]. When viewed in the optimization framework these priorities are dual variables but the priority-based routing strategy \(R(\lambda)\) defined in (10) can be implemented for any priority vector \(\lambda\). We define a general priority-based routing strategy
\[
r_t = R(\lambda_t) \quad \text{for any sequence } \lambda_t, \forall t. \tag{19}
\]
A priority based routing strategy based on (19) generates a sequence of queue lengths
\[
q_{t+1} = q_t - g_t(\lambda_t) \tag{20}
\]
where the stochastic gradient function, \(g_t(\lambda)\) is defined in (14).

The routing strategy \(R(\lambda_t)\) for any sequence \(\{\lambda_t\}_t \geq 0\) generalizes the notion of Soft Backpressure, [11] to a case where the priorities can be generated by any desirable scheme. Another example of a priority based routing algorithm is Accelerated Backpressure, [12]. Priority sequences \(\{\lambda_t\}_t \geq 0\) are not all equally effective. Determination of sufficient conditions for a priority sequence \(\{\lambda_t\}_t \geq 0\) to yield stable queue lengths is an open problem.
Algorithm 1: Discounted Integral Priority Based Routing

1. Initialize $\lambda_k^i(0) = q_k^i(0)$ for each $k$
2. for $t = 0, 1, 2, \cdots$ do
   3. for all neighbors $j \in n_i$ do
      4. Send priorities $\{\lambda_k^j(t)\}_k$ – Receive priorities $\{\lambda_k^j(t)\}_k$
      5. Compute $\mu_{ij}$ such that
         $$\sum_k F\left(-[\lambda_k^j(t) - \lambda_k^j(t) - \mu_{ij}]^+\right) = C_{ij}$$
         Transmit packets at rate $r_{ij}^j(t) = F\left(-[\lambda_k^j(t) - \lambda_k^j(t) - \mu_{ij}]^+\right)$
   6. end
7. Observe $\{q_k^i(t+1)\}_k$,
8. Compute $\lambda_k^i(t+1) = \alpha \lambda_k^i(t) + q_k^i(t+1)$
9. end

III. DISCOUNTED INTEGRAL PRIORITY BASED ROUTING

Observing that existing protocols for packet forwarding which use optimization motivated priorities tend to stabilize but not reduce the queues, we propose a set of priorities motivated by integral control. In particular, since the queues are state variables which we would like to make small at steady state, we set the priorities to a discounted integral of the queues. Consider the set of routing priorities

$$\lambda_t = \sum_{\tau=0}^t \alpha^{t-\tau} q_\tau$$

where $\alpha \in [0, 1)$ is a discounting factor. Observe that for if $\alpha = 0$, the soft backpressure algorithm $\lambda_t = q_t$ is trivially recovered. Backpressure type algorithms are implemented by observing the stochastic gradients $g_t = g_t(\lambda_t)$ from the change in queue lengths $g_t = q_t - q_{t+1}$. We implement DIR routing using a recursive linear update; $q_t$ is observed and the priorities are updated

$$\lambda_t = \alpha \lambda_{t-1} + q_t,$$

which can be computed without information from neighboring nodes: $\lambda_k^i(t) = \alpha \lambda_k^i(t-1) + q_k^i(t)$ for all nodes $i$, information flows $k$ and times $t$. Under this scheme we do not force any information exchange, rather we allow information to spread through the effect of the changes in the realized routing.

See Algorithm 1 for the distributed implementation of the priority based routing $r_t = R(\lambda_t)$ using the discounted integral priorities $\{\lambda_t\}_{t=0}^\infty$ chosen according to equation (21).

A. Connection to Heavy Ball Methods

The discounted integral priority update can be expressed as a variant on the heavy ball method, [14] with an additional error term caused by the initial queue lengths.

Lemma 1. The discounted integral Priority update can be equivalently expressed by the relation

$$\lambda_{t+1} = \lambda_t - g_t(\lambda_t) - \sum_{\tau=0}^{t-1} \alpha^{t-\tau} g_\tau(\lambda_\tau) + \alpha^{t+1} q_0$$

where $g_\tau(\lambda_\tau)$ is the stochastic gradient at time $\tau$.

Proof: Consider (21) with the sum order reversed,

$$\lambda_t = \sum_{\tau=0}^t \alpha^t q_{t-\tau},$$

making the change in priorities

$$\lambda_{t+1} - \lambda_t = \sum_{\tau=0}^{t+1} \alpha^\tau q_{t+1-\tau} - \sum_{\tau=0}^t \alpha^\tau q_{t-\tau}.$$  

Reorganizing the terms we have

$$\lambda_{t+1} - \lambda_t = \sum_{\tau=0}^t \alpha^{t-\tau} (q_{t+1-\tau} - q_\tau) + \alpha^{t+1} q_0.$$  

From (20), $q_{t+1-\tau} - q_\tau = -g_\tau(\lambda_\tau)$ is the stochastic gradient $g_\tau(\lambda_\tau)$. Pull out the $t$ term to complete the proof.

If one initializes the priority based routing algorithm with empty queues, $q_0 = 0$ or runs the algorithm long enough the error term $\alpha^t q_0$ disappears and we are left with a infinite memory heavy ball type method with dependence on the whole history of stochastic gradients. Further reading on the standard heavy ball and related iterative gradient methods can be found in [18].

IV. STABILITY ANALYSIS

Having established uniqueness of the gradient $\nabla h(\lambda)$ in Proposition 1, we know that $h(\lambda)$ is differentiable which implies it is also a Lipschitz continuous function whose Lipschitz coefficient we denote $\bar{l}$. Going forward we add the assumption that the gradient $\nabla h(\lambda)$ is also Lipschitz continuous.

Assumption 1. The gradient function $\nabla h(\lambda)$ satisfies

$$||\nabla h(\lambda) - \nabla h(\lambda)|| \leq l||\lambda - \bar{\lambda}||.$$  

Lipschitz continuity of the gradient is a standard assumption in convex optimization, [19]. We further characterize the stochastic gradient vector $g_\tau(\lambda)$ defined element-wise in (14).

Definition 1. Define an upper bound $\gamma$ on the norm of the stochastic gradient

$$||g_\tau(\lambda)|| \leq \gamma := \max_{\nu \in \text{supp}(\nu)} ||\bar{A} r + a + \nu||$$

for all $t$ and all $\lambda$.

The upper bound $\gamma$ is necessarily finite because $\mathcal{C}$ is a compact set and $\nu_\pi$ has finite support. A precise characterization of $\gamma$ will depend on the network structure and the variance of $\nu_\pi$, however for our current results existence of $\gamma$ is sufficient.

Lemma 2. The sequence of positive random variables

$$\delta_t = ||(1 - \alpha) \lambda_t - q_t||$$

generated by Algorithm 1, satisfies

$$\delta_{t+1} \leq \delta_t \quad \text{for} \quad \delta_t \geq \frac{\gamma \alpha}{1 - \alpha}$$

for a discount factor $\alpha \in (0, 1)$.
**Proof:** Substitute (21) into (29) at time $t + 1$ to get

$$\delta_{t+1} = \left\| (1 - \alpha) \sum_{\tau=0}^{t+1} \alpha^{t+1-\tau} q_\tau - q_{t+1} \right\|. \tag{31}$$

Pulling out the first term of the sum

$$\delta_{t+1} = \left\| (1 - \alpha)q_{t+1} + (1 - \alpha) \sum_{\tau=0}^{t} \alpha^{t+1-\tau} q_\tau - q_{t+1} \right\|. \tag{32}$$

Simplifying terms and factoring out $\alpha$ yields

$$\delta_{t+1} = \alpha \left\| (1 - \alpha) \sum_{\tau=0}^{t} \alpha^{t-\tau} q_\tau - q_{t+1} \right\|. \tag{33}$$

Applying the definition in (21), we have

$$\delta_{t+1} = \alpha \left\| (1 - \alpha) \lambda_t - q_t + g_t(\lambda_t) \right\|. \tag{34}$$

Consider $\delta_t^2$ as an inner product

$$\delta_{t+1}^2 = \alpha^2 \left( \delta_t^2 - 2((1 - \alpha) \lambda_t - q_t) g_t(\lambda_t) + \left\| g_t(\lambda_t) \right\|^2 \right) \leq \alpha^2 (\delta_t^2 + 2\delta_t \gamma + \gamma^2) \tag{35}$$

from our upper bound in (28) and the observation that

$$-((1 - \alpha) \lambda_t - q_t) g_t(\lambda_t) \leq \left\| (1 - \alpha) \lambda_t - q_t \right\| \left\| g_t(\lambda_t) \right\| \tag{36}$$

by Cauchy-Schwartz. Defining the concave quadratic function

$$p(x) = (\alpha^2 - 1)x^2 + 2\gamma\alpha^2x + \gamma^2\alpha^2, \tag{37}$$

we have $\delta_{t+1} - \delta_t \leq p(\delta_t)$. The roots of $p(x)$ can be computed

$$x_{r_1} = \frac{\gamma\alpha}{1 + \alpha}; \quad x_{r_2} = \frac{\gamma\alpha}{1 - \alpha}. \tag{38}$$

From concavity of $p(x)$, we have

$$p(x_{r_2} + \Delta x) \leq p(x_{r_2}) + p'(x_{r_2})\Delta x \leq -2\alpha\gamma\Delta x. \tag{39}$$

By letting $\Delta x = \delta_t - x_{r_2}$, we can conclude

$$\delta_{t+1}^2 - \delta_t^2 = p(\delta_t) \leq 0 \tag{41}$$

for all $\delta_t \geq x_{r_2}$, completing the proof. ■

Proposition 2 is interesting because although $\delta_t$ is a random variable, we observe that due to the dynamics embedded in the problem if it gets to large it must decrease.

**Proposition 2.** The discounted integral priority algorithm yields sequences of queues $q_t$ and priorities $\lambda_t$ for which there exists $T$ such that

$$\delta_t = \left\| (1 - \alpha) \lambda_t - q_t \right\| < \alpha \gamma \frac{2 - \alpha}{1 - \alpha} \tag{42}$$

for all $t \geq T$ where $\alpha \in (0, 1)$ is the discounting factor, and $\gamma$ is defined in (28).

**Proof:** Consider the characterization of $\delta_{t+1}$ in equation (34). Applying the triangle inequality,

$$\delta_{t+1} \leq \alpha \delta_t + \alpha \left\| g_t(\lambda_t) \right\|. \tag{43}$$

The single iteration change in $\delta_t$ is bounded

$$\delta_{t+1} - \delta_t \leq (1 - \alpha) \delta_t + \alpha \left\| g_t(\lambda_t) \right\| \leq \alpha \gamma \tag{44}$$

because $(1 - \alpha) < 0$ and $\left\| g_t(\lambda_t) \right\| \leq \gamma$ from (28). From Lemma 2, we know that $\delta_{t+1} \leq \delta_t$ when $\delta_t \geq \frac{\gamma}{1 - \alpha}$. Applying the single step increase in equation (44) for $\delta_t < \frac{\gamma}{1 - \alpha}$, we conclude that

$$\delta_{t+1} \leq \gamma \frac{\alpha}{1 - \alpha} + \alpha \gamma = \gamma \frac{2 - \alpha}{1 - \alpha} \tag{45}$$

for all $\delta_t$, completing the proof. ■

Proposition 2 indicates that the queue lengths can never become too different from a scaled version of the priority vector. The statement of the proof hold for $t > T$ which is necessitated by the possibility of initializing with a large $\delta_0$. If $\lambda_0$ and $q_0$ are initialized such that $\delta_0 < \alpha \gamma \frac{2 - \alpha}{1 - \alpha}$ then $\delta_t$ will remain in this ball for all $t$.

**Lemma 3.** The single step dual update is bounded

$$\left\| \lambda_{t+1} - \lambda_t \right\| \leq \frac{\gamma}{1 - \alpha} : = \Delta \tag{46}$$

for all $t \geq T$ where $\alpha \in (0, 1)$ is the discounting factor, $\gamma$ is the gradient norm bound initial queues satisfy $q_0 = 0$.

**Proof:** Consider equation (23) from Lemma 1, putting the $g_t(\lambda_t) = \alpha \mu g_t(\lambda_t)$ term back in the sum we have

$$\left\| \lambda_{t+1} - \lambda_t \right\| = \left\| \sum_{\tau=0}^{t} \alpha^{t-\tau} g_t(\lambda_\tau) \right\|. \tag{47}$$

when the $q_0 = 0$. Taking the norm before adding yields

$$\left\| \lambda_{t+1} - \lambda_t \right\| \leq \left\| \sum_{\tau=0}^{t} \alpha^{t-\tau} \left\| g_t(\lambda_\tau) \right\| \right\|. \tag{48}$$

Applying the stochastic gradient norm bound from (28),

$$\left\| \lambda_{t+1} - \lambda_t \right\| \leq \left\| \sum_{\tau=0}^{t} \alpha^{t-\tau} \gamma \right\|$$

and replacing the finite sum with the infinite sum we have

$$\left\| \lambda_{t+1} - \lambda_t \right\| \leq \frac{\gamma}{(1 - \alpha)} \tag{50}$$

completing the proof. ■

Lemma 3 defines an upper bound on the distance the priority vector can move in a single iteration. This distance is important later when we consider the effect of the histories, $\{\lambda_\tau, \tau < t\}$.

**Lemma 4.** Define the Lyapunov function

$$L_t = h(\lambda_t) - h(\lambda^*) \tag{51}$$

the update is given as

$$\mathbb{E}[L_{t+1} - L_t | \mathcal{F}_t] \leq -\sum_{\tau=0}^{t} \alpha^\tau \nabla h(\lambda_\tau) \nabla h(\lambda_{t-\tau}) \tag{52}$$

when $q_0 = 0$ where $\mathcal{F}_t$ includes all information up to time $t$. 
The gradients at those points satisfy

\[ L_{t+1} - L_t = h(\lambda_{t+1}) - h(\lambda_t) \]

as the \( h(\lambda^*) \) terms cancel. By convexity of \( h(\lambda) \),

\[ L_{t+1} - L_t \leq \nabla h(\lambda_t)'(\lambda_{t+1} - \lambda_t). \]  

(54)

Applying equation (26) with zero initial queues to substitute for \( (\lambda_{t+1} - \lambda_t) \), we have

\[ L_{t+1} - L_t \leq \nabla h(\lambda_t)' \sum_{\tau=0}^{t} \alpha^{t-\tau}(q_{\tau+1} - q_{\tau}). \]  

(55)

Observing that

\[ \mathbb{E}[q_{\tau+1} - q_{\tau}|F_{t-\tau}] = \mathbb{E}[-q_{\tau}(\lambda_{t-\tau})|F_{t-\tau}] = -\nabla h(\lambda_{t-\tau}) \]

(56)

we complete the proof by substituting (56) into (55) and applying linearity of expectation.

Lemma 4 defines a Lyapunov function which is the optimality gap for the dual problem and its result tells us that if the dual gradient is lined up with the time discounted average of previous dual gradients the energy in the system will decrease. This follows intuitively from our connection infinite memory heavy ball methods.

**Lemma 5.** Consider dual feasible points \( \lambda \) and \( \tilde{\lambda} \) satisfying

\[ (\lambda - \lambda^*)' (\tilde{\lambda} - \lambda^*) = \phi > 0. \]  

(57)

The gradients at those points satisfy

\[ \nabla h(\lambda)'\nabla h(\tilde{\lambda}) \geq \frac{l^2}{4}||\lambda - \lambda^*|| \cdot ||\tilde{\lambda} - \lambda^*|| \]

(58)

where \( l \) is the Lipschitz constant for the \( \nabla h(\cdot) \).

**Proof:** For convex functions with Lipschitz gradients we have

\[ h(\lambda^*) \geq h(\lambda) + \nabla h(\lambda)'(\lambda^* - \lambda) + \frac{l}{2}||\lambda^* - \lambda||^2. \]  

(59)

Reorganizing terms and observing that \( h(\lambda) - h(\lambda^*) \geq 0 \) for all \( \lambda \) have

\[ \nabla h(\lambda)'(\lambda - \lambda^*) \geq \frac{l}{2}||\lambda - \lambda^*||^2. \]

(60)

Now we apply equation (60) at both \( \lambda \) and \( \tilde{\lambda} \) and multiply

\[ \frac{l^2||\lambda - \lambda^*||^2}{4} \]

\[ \leq \nabla h(\lambda)'(\lambda - \lambda^*) \nabla h(\tilde{\lambda})'(\tilde{\lambda} - \lambda^*). \]

(61)

We divide both sides by \( (\lambda - \lambda^*)'(\tilde{\lambda} - \lambda^*) \) and reorganize this expression as

\[ \frac{l^2||\lambda - \lambda^*||^2}{4(\lambda - \lambda^*)'(\lambda - \lambda^*)} \]

\[ \leq \nabla h(\lambda)'(\lambda - \lambda^*)'(\tilde{\lambda} - \lambda^*) \nabla h(\tilde{\lambda})'(\tilde{\lambda} - \lambda^*). \]

(62)

Observing that

\[ \frac{(\lambda - \lambda^*)'(\tilde{\lambda} - \lambda^*)}{(\lambda - \lambda^*)'(\lambda - \lambda^*)} \]

is a rank 1 matrix with its sole eigenvalue at 1, and applying the definition of \( \phi \) we simplify (62) to

\[ \frac{l^2||\lambda - \lambda^*||^2}{4\phi} \leq \nabla h(\lambda)'\nabla h(\tilde{\lambda}). \]  

(63)

Finally by Cauchy-Schwartz inequality we have

\[ \frac{||\lambda - \lambda^*|| \cdot ||\tilde{\lambda} - \lambda^*||}{\phi} \geq \frac{||\lambda - \lambda^*|| \cdot ||\tilde{\lambda} - \lambda^*||}{(\lambda - \lambda^*)'(\lambda - \lambda^*)} \geq 1 \]

yielding the inequality

\[ \frac{l^2||\lambda - \lambda^*|| \cdot ||\tilde{\lambda} - \lambda^*||}{4} \leq \nabla h(\lambda)'\nabla h(\tilde{\lambda}) \]

(64)

which completes the proof.

Lemma 5 tells us that if we look at the angle between the vectors connecting to dual feasible points to the optimal. As long as that angle remains positive, the angle between the gradients at those points is also positive.

**Lemma 6.** At time \( t \), given a value \( L_t \) satisfying

\[ L_t \geq \hat{t}\Delta \tau \]

(65)

where \( \hat{t} \) is the Lipschitz constant for \( h(\lambda) \), \( \Delta \) is maximum single step distance defined in Lemma 3 and a time \( \tau < t \), the gradient inner product satisfies

\[ \nabla h(\lambda_t)'\nabla h(\lambda_{t-\tau}) \geq 0. \]

(66)

**Proof:** Consider the point \( \lambda_{t-\tau} \), given the single step maximum distance \( \Delta \) we know that

\[ ||\lambda_t - \lambda_{t-\tau}|| \leq \Delta \tau. \]

(67)

Provided that \( ||\lambda_{t-\tau} - \lambda^*|| \leq ||\lambda_t - \lambda^*|| \), the largest angle between \( \lambda_{t-\tau} \) and \( \lambda_t \) occurs when \( ||\lambda_t - \lambda^*|| = ||\lambda_{t-\tau} - \lambda^*|| \), from basic trigonometry. Formally,

\[ \phi_{\tau} = (\lambda_t - \lambda^*)'(\lambda_{t-\tau} - \lambda^*) \geq 1 - \frac{(\Delta \tau)^2}{2||\lambda_t - \lambda^*||^2} \]

from the Law of Cosines. From (67) and (68) it is guaranteed that \( \phi_{\tau} > 0 \) when \( ||\lambda_t - \lambda^*|| \geq \Delta \tau \). Applying the definition of \( L_t \) in (65) we have,

\[ h(\hat{t}) - h(\lambda_t) \geq \hat{t}\Delta \tau. \]

(69)

Applying the Lipschitz continuity of \( h(\lambda) \), we have

\[ ||\lambda_t - \lambda^*|| \geq \Delta \tau \]

(70)

guaranteeing \( \phi_{\tau} > 0 \) allowing us to apply us to apply Lemma 5 which completes the proof.

**Proposition 3.** The Lyapunov Drift is characterized as

\[ \mathbb{E}[L_{t+1} - L_t|F_t] \leq \kappa_1L_t - \kappa_2L_t \]

(71)

where \( \hat{\lambda} = \alpha^{\frac{1}{\alpha}} \in (0,1) \), \( \kappa_1 = \frac{\tau^2}{1-\alpha} \) and \( \kappa_2 = 2L \).
Proof: Let \( N = \min_x \) such that \( \tau \geq \frac{L^*}{\hat{L}} \). Then applying Lemma 4, pulling the first term from the sum we have
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\nabla h(\lambda_t) \nabla h(\lambda_t) - \sum_{\tau=1}^{N} \alpha^\tau \nabla h(\lambda_t) \nabla h(\lambda_{t-\tau}).
\]
Applying Lemma 6, we discard additional terms up to \( N \),
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\nabla h(\lambda_t) \nabla h(\lambda_t) - \sum_{\tau=1}^{N} \alpha^\tau \nabla h(\lambda_t) \nabla h(\lambda_{t-\tau}).
\]
Observing from (28) that \( -\nabla h(\lambda_t) \nabla h(\lambda_{t-\tau}) \leq \gamma^2 \) we have
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\nabla h(\lambda_t) \nabla h(\lambda_t) + \sum_{\tau=1}^{N} \alpha^\tau \gamma^2.
\]
Adding additional terms to make the summation to infinity, we use the geometric series to get
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\nabla h(\lambda_t) \nabla h(\lambda_t) + \frac{\alpha^N}{1-\alpha} \gamma^2.
\]
and from the definition of \( N \) we have
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\nabla h(\lambda_t) \nabla h(\lambda_t) + \frac{\gamma^2}{1-\alpha} \hat{L} t.
\]
Define an alternative update step
\[
\lambda_t^+ = \lambda_t - \frac{1}{t} \nabla h(\lambda_t).
\]
Applying the Lipschitz Gradient Assumption, we have
\[
h(\lambda_t^+) \leq h(\lambda_t) - \frac{1}{2t} ||\nabla h(\lambda_t)||^2 \]
and observing \( h(\lambda^+) \leq h(\lambda^+) \) allows us to substitute in \( L_t \),
\[
-||\nabla h(\lambda_t)||^2 \leq -2tL_t.
\]
Finally plugging (99) into (76) completes the proof.

Lemma 7. We have an attractive region, \( L_t \leq L^* \) where
\[
L^* := \frac{\lambda \left( \kappa_2 - \epsilon \right) \ln \hat{\alpha}}{\ln \hat{\alpha}}.
\]
where \( \lambda \) is the Lambert W-function and \( \epsilon \in (0, \kappa_2) \). When \( L_t \geq L^* \), the Lyapunov drift satisfies
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\epsilon L_t.
\]
Proof: Define the function
\[
v(x) = (\kappa_2 - \epsilon) x - \kappa_1 \hat{\alpha} x.
\]
Consider the root \( x_r \) such that \( v(x_r) = 0 \). Organizing terms appropriately we have
\[
\hat{\alpha}^{x_r} \exp(x_r) = \frac{\kappa_2 - \epsilon}{\kappa_1}.
\]
for which we have a known solution
\[
x_r = \frac{-W\left(-\frac{(\kappa_2-\epsilon)\ln \hat{\alpha}}{\kappa_1} \right)}{\ln \hat{\alpha}}
\]
from the study of the Lambert W-function, [20]. We compute the derivative of \( v(x) \),
\[
v'(x) = \kappa_2 - \epsilon - \kappa_1 \frac{\hat{\alpha} x}{\ln \hat{\alpha}}
\]
and observe that \( v'(x) > 0 \) because from the definitions in Proposition 3 we have \( \kappa_1 > 0, \kappa_2 - \epsilon > 0 \) and \( \ln(\hat{\alpha}) < 0 \) as a result of \( \hat{\alpha} \in (0,1) \). Since \( v(x) \) is an increasing function
\[
v(x_r + \Delta x) \geq v(x_r) = 0 \]
for any positive \( \Delta x \). From Proposition 3 and equation (82),
\[
\mathbb{E}[L_{t+1} - L_t | F_t] \leq -\epsilon L_t.
\]
Form (86) we have \( v(L_t) \geq 0 \) for any \( L_t \geq L^* \) completing the proof.

Lemma 8. For any \( T \) there exists a \( t > T \) such that
\[
L_t \leq L^*
\]
where \( L^* \) is the neighborhood defined in Lemma 7.
Proof: Define a \( T \)-parameterized family of sequences of random variables
\[
\xi_T(t) = L_t 1\{L_r > L^*, \forall \tau \in [T,t]\}
\]
where \( 1\{\cdot\} \) is the indicator function. Considering Lemma 7 we have
\[
\mathbb{E}[\xi_T(t+1)] \leq (1-\epsilon)L_t 1\{L_r > L^*, \forall \tau \in [T,t]\} \leq (1-\epsilon)\xi_T(t).
\]
Equation (90) allows application of Supermartingale Convergence Theorem(see Theorem E7.4 in [21]), which guarantees that \( \xi_T(t) \) converges almost surely. Furthermore, equation (90) applied recursively tells us that \( \xi_T(t) \) converges to 0 in expectation. Combining these two convergence results, we have that \( \xi_T(t) \) converges almost surely to 0. Thus for any \( T \) we have a Supermartingale
\[
\xi_T(t) \overset{a.s.}{\to} 0
\]
which ensures that there exists a \( t \geq T \) such that \( L_t \leq L^* \).
Soft Backpressure (SBP) but neither do any work to remove Backpressure (ABP). BP stabilizes the queues more quickly than ABP. The main benefit of introducing discounted integral priorities is the ability to drive large queues out of the system. We use Proposition 2 to say the queue lengths cannot be too different from the duals to complete our proof.

**Proof:** just plug it in to Prop 2!

Proposition 4 is our main result. It proves that the queues are always eventually less than the threshold defined in (93). Our proof builds a supermartingale from $L_t$, and uses Lipschitz continuity of $h(\lambda)$ to show that $\lambda_i$ is always eventually in the error neighborhood $||\lambda_i - \lambda^*|| \leq tL^*$. We use Proposition 2 to say the queue lengths cannot be too different from the duals to complete our proof.

V. NUMERICAL RESULTS

The main benefit of introducing discounted integral priorities is the ability to drive large queues out of the system. We compare the performance of DIP routing with that of stabilization algorithms in a set of numerical experiments. Our experiments take place on the network in Figure 1, with edge capacities chosen uniformly randomly in $(0, 50]$ and there are $K = 5$ data types with unique destination nodes. Each queue starts with an expected initial backlog of 10 packets. The arrival process has an expectation of 5 packets per queue which results in an expected $5(n-1)K = 45$ packets entering the system at each time. The resulting problem (7) is feasible but non-trivial. Our choice of objective function is

$$f^{k}_{ij}(r^{k}_{ij}) = -\frac{1}{2} r^{k}_{ij} + \beta^{k}_{ij} r^{k}_{ij}. \quad (93)$$

The quadratic term captures an increasing cost of routing larger quantities of packets across a link and help to eliminate myopic routing choices that lead to sending packets in cycles. The linear term $\beta$ is introduced to reward sending packets to their destinations. In our simulations, $\beta^{k}_{ij} = 10$ for all edges routing to their respective data type destinations $j = o_k$ and all other $i, j, k, \beta^{k}_{ij} = 0$. Figure 2, is a characteristic example of the relative behaviors of the algorithms being studied. Backpressure (BP) stabilizes the queues but the total number of backlogged packets remains large. Accelerated Backpressure (ABP) stabilizes the queues more quickly. However, Soft Backpressure (SBP) but neither do any work to remove existing queue build up. Discounting Integral Priority (DIP) routing however drives the total packets queues significantly below their starting levels.

In order to more precisely characterize the relative behavior of these algorithms, we repeat the numerical experiment 100 times and record the average-average backlog. Two averages are used, we take a time average over 500 iterations to account for fluctuations and we average over the number of queues. Thus the data being presented is the average number of packets in each queue over the whole trial. Figure 3 is the distribution over all 100 trials. The key observation is that the Discounted Integral Priority (DIP) routing algorithm results in an average queue length of well below the initial average queue length while the other algorithms are at best close to the starting value. In fact the average backlog is close to the average arrival rate. Another important observation is highlighted in Table I; these results are achieved with an extremely tight variance over many trials. Every realization of the problem has different edge capacities but the same arrival statistics. The implication is that the Discounted Integral Priorities successfully learn information about the arrival statistics and the remaining steady state backlogs result from the fact that the arrivals are stochastic.

VI. CONCLUSION

The Discounted Integral Priority routing algorithm (DIP) is a significant improvement over its optimization motivated backpressure-type counterparts. By introducing the notion of integral control, the steady state error is significantly reduced with the error. We have shown that the DIP routing guarantees queue stability and demonstrated numerically that drives out the initial queues.

Future direction for this work includes expanding the analysis to prove not only stability but queue reduction. Also, we plan to explore the DIP routing’s ability to handle periodic

![Fig. 1. Several numerical experiments for the Discount Integral Priority (DIP) routing algorithm presented in this section are performed on this simple 10 node network with 5 data types. The destinations are unique for each data type and are chosen randomly.](image1)

![Fig. 2. ABP and SBP stabilize the queues much more effectively than BP, but DIP routing drives the queues down and keeps them stable there.](image2)
Fig. 3. ABP and SBP stabilize the queues relatively quickly, preventing them from getting too far above the initial queues. ABP is faster than SBP but exhibits more variance. DIP routing however drives the queues well below the initial queue lengths.

large arrival shocks, something stabilizing algorithms struggle with. Based on our current results, we can expect good performance as long as the shocks are not so frequent that they make the problem infeasible.

**References**


