Cluster-Based Back-Pressure Routing Algorithm

Lei Ying and R. Srikant
Coordinated Science Lab
and
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
{lying, rsrikant}@uiuc.edu

Don Towsley
Department of Computer Science
University of Massachusetts Amherst
towsley@cs.umass.edu

***Abstract*—We study scalable, distributed, and adaptive routing algorithms for communication networks. The back-pressure algorithm introduced in [21] is a well-known distributed and adaptive routing/scheduling algorithm where nodes only need the queue length information of neighboring nodes to make routing decisions, and packets are adaptively routed in the network according to congestion information, which makes the algorithm resilient to traffic and topology changes. However, the back-pressure algorithm requires routers to maintain a separate queue for each destination, which prevents its implementation in large-scale networks like the Internet. In this paper, we propose a cluster-based back-pressure routing algorithm, which retains the distributability and adaptability of back-pressure routing, while significantly reducing the number of queues that have to be maintained at each node. Since the cluster-based algorithm performs adaptive load-balancing in the network, it has the potential to eliminate the need for *off-line* traffic engineering in the Internet.**

I. Introduction

Traffic engineering has received substantial attention in recent years as ISPs deal with exponential growth in data traffic. Briefly, traffic engineering refers to the optimization of network resources through routing given information about the amount of traffic entering and leaving the network. Although traditionally concerned with the management of flows within a network, traffic engineering has expanded to include management of network resources through the setting of routes at the *inter-domain level*. The output of traffic engineering is typically a set of routes that provide the level of performance required by the ISP. Protocols such as OSPF [14], IS-IS [3], or MPLS [17] are then used to instantiate these paths, either exactly or approximately.

Traffic engineering traditionally generates a set of routes between ingress and egress points in the network [8], [22], [19]. However, these are point to point routes and, thus are not always robust to failures or unpredicted changes in traffic loads. Some work has focussed on computing routes that are robust to changes in loads, [2], [25]. However, the routes are not robust to failures and there is no guarantee that they can achieve the capacity of the network. More recently there has been interest in *inter-domain* traffic engineering. However, this is in its infancy; see [7] for some initial results.

Research supported by NSF grants CNS 05-19691, CNS 05-19922, CCF 06-34891, CNS 07-21286, and CNS 07-21779

In this paper, we propose the use of the back-pressure algorithm for the purpose of traffic engineering. First proposed in [21], this is a distributed and adaptive routing/scheduling algorithm where nodes use the queue length information of neighboring nodes to make routing decisions. Packets are adaptively routed throughout the network in response to congestion information. Thus the back-pressure algorithm is resilient to link failures and topological changes. Moreover, it has been shown to be throughput optimal, i.e., if *any* routing algorithm can support a set of traffic flows, then the back-pressure algorithm can as well. When deployed in a wireless network, there is a network-wide scheduling problem that requires addressing due to the contention aspect of the media. However, this problem disappears in a wireline network [1]. Finally, although first proposed to handle inelastic traffic, the back-pressure algorithm in conjunction with congestion control, can be used to provide fair resource allocation to elastic traffic [12], [5], [6], [15], [20].

Although extremely simple, through its reliance on simple rules based on local neighbor information, the back-pressure algorithm suffers one serious drawback that renders it unsuitable for adoption as a traffic engineering solution within either a large network or the Internet, namely it requires the maintenance of one queue per destination at every router. Given that the Internet includes hundreds of millions of end hosts, this precludes its deployment in an environment such as the Internet. One technique for reducing this state is to aggregate destinations according to the last hop access router that it connects to. However, at the level of the Internet, this can result in hundreds of thousands of destination queues. At the level of an ISP, this might still result in 100s of destinations.

In this paper, we tackle this problem of state explosion by proposing a *cluster-based* back-pressure scheduling/routing algorithm. This algorithm works as follows. Routers are grouped into clusters. Associated with each cluster is one or more gateway routers that allow transit into the cluster. Each router needs only to be aware of the destinations within its cluster and the gateway routers associated with the other clusters. Routing proceeds as follows. If the packet is destined within the same cluster that it was generated, then routers use standard back-pressure. However, when a packet is generated at a source destined to a different cluster, loose source routing is used.

The source first chooses a gateway at the cluster containing the destination and then the packet is routed to the destination through the gateway using back-pressure. Gateway selection is made by considering the queue lengths associated with the specific destination at all gateways to that cluster coupled with gateway queue lengths at the source.

We formally establish that our cluster-based back-pressure algorithm is throughput optimal, thus exhibiting the same optimality properties as the traditional back-pressure algorithm. However, the initial design given in Section III exhibits another serious deficiency, namely that it requires the acquisition of instantaneous queue length information from gateways by traffic sources. In Section IV we modify the initial cluster-based back-pressure design to allow information exchange between the gateways and sources to occur at a much slower time scale than that associated with routing/scheduling decisions while still preserving throughput optimality. In the remainder of this section we discuss possible applications of this new algorithm in the context of traffic engineering and relate it to earlier work in the area.

Consider a single ISP running OSPF. A large ISP typically divides its network into areas connected through a backbone area [14]. OSPF can easily be replaced by a back-pressure protocol where the areas are mapped into clusters. Ingress routers and egress routers then map into sources and destinations within the cluster-based back-pressure approach. Cluster-based back-pressure routing could also be applied at the Internet level where clusters correspond to autonomous systems (ASes). Gateways for every AS would be advertised throughout the Internet and routers would then maintain queues for all of these gateways along with the destinations within their ASes. The number of queues expected to be maintained at each router could be quite large but not unreasonable for current routers. We believe that even this burden can be reduced by dividing clusters further into sub-clusters and so on. Instead of loose source routing specifying a single intermediate gateway to the destination, we have to specify a few more intermediate nodes. Finally, the cluster-based back-pressure routing is applicable in wireless settings as well. With the current and growing interest in wireless mesh networks where the access nodes are static, the cluster-based back-pressure routing is a promising mechanism for effecting traffic engineering among access points in that setting.

Before we move on to describe the cluster-based back-pressure routing, it is important to note other recent proposals that could potentially provide the same function. There have been several proposals for multipath rate controllers [9], [10]. Both argue that robustness is significantly improved by allowing sources to simultaneously route data over multiple paths to a destination. Furthermore [9] shows that network capacity increases as sources are made explicitly aware of and allowed to use an increasing number or paths. Cluster-based back-pressure routing on the other hand provides access to all possible paths between a source and destination without the source needing to know more than the gateways associated with the cluster within which the destination resides. Last, [11] shows that the benefits of using many paths can be obtained by simply using two paths provided that they are constantly replaced with new and better paths. However, this technique is still unable to try all possible paths between a source and destination.

II. Model and Intuition

Consider a network represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{L}$ is the set of directed links. Denote by (m, n) a link from node m to node n, and let $\eta_{(m,n)}$ denote the maximum rate supportable at link (m, n). We identify a flow by its source (s) and destination (d) nodes and denote it by $[s, d]$. We use $\mathcal{F}$ to denote the set of all flows in the network. Assume that time is discretized, and let $x_s^d[t]$ $(s, d \in \mathcal{N})$ denote the rate at which a flow $[s, d]$ generates packets in time slot t. For simplicity, we assume that the flow arrival rates are constant, i.e., $x_s^d[t] = x_s^d$; and define $\mathbf{x} = \{x_s^d\}$. It is straightforward to extend our results to allow a large class of time-varying stochastic arrival processes. Without loss of generality, we assume that $x_n^n = 0$. Furthermore, let $\boldsymbol{\mu}[t] = \{\mu_{(m,n)}[t]\}$ where $\mu_{(m,n)}[t]$ denotes the transmission rate over link (m, n) at time instant t, and Γ denote the convex hull of the set of all feasible $\boldsymbol{\mu}[t]$. For a wireline network, Γ is simply the set of all $\{\mu_{(m,n)}\}$ satisfying $\mu_{(m,n)} \leq \eta_{(m,n)}$. For wireless networks, Γ is determined by the choice of interference model. Finally, we let $\mu_{(m,n)}^{[s,d]}[t]$ denote the rate at which packets of flow $[s, d]$ are served over link (m, n) at time t. Thus, $\sum_{[s,d]\in\mathcal{F}} \mu_{(m,n)}^{[s,d]} = \mu_{(m,n)}$.

We say $\mathbf{x}$ is supportable in network $\mathcal{G}$ if there exists $\{\mu_{(m,n)}^{[s,d]}\}$ such that the following two conditions hold:

(i) For any flow $[s, d]$ and node $n \neq d$,

$$x_n^d 1_{n=s} + \sum_{m:(m,n)\in\mathcal{L}} \mu_{(m,n)}^{[s,d]} = \sum_{i:(n,i)\in\mathcal{L}} \mu_{(n,i)}^{[s,d]}; \qquad (1)$$

(ii)

$$\left\{ \sum_{[s,d]\in\mathcal{F}} \mu_{(m,n)}^{[s,d]}[t] \right\} \in \Gamma. \qquad (2)$$

Condition (1) is the flow-conservation constraint for flow $[s, d]$ at each node n. Condition (2) is the capacity constraint. Denote by $\mathbf{X}_\mathcal{G}$ the set of all supportable $\mathbf{x}$; $\mathbf{X}_\mathcal{G}$ is called the the capacity region of network $\mathcal{G}$.

Now given network $\mathcal{G}$ and flows $\mathbf{x} \in \mathbf{X}_\mathcal{G}$, the goal is to develop a routing and scheduling algorithm to support $\mathbf{x}$. The back-pressure algorithm is one such algorithm, which is known to be throughput-optimal [21], i.e., it can support any $\mathbf{x}$ such that $(1 + \epsilon)\mathbf{x} \in \mathbf{X}_\mathcal{G}$ for some $\epsilon > 0$. Under the back-pressure algorithm, each node maintains a separate queue for all flows with the same destination. Denote by $q_n^d[t]$ the length of the queue for destination d at node n. At each time slot, node m calculates the queue length difference $q_m^d[t] - q_n^d[t]$, and the link (m, n) is used to serve the queue with the largest difference. An advantage of the back-pressure algorithm is that it is adaptive, i.e., it finds optimal routes in the network without knowing the arrival rates of the flows. Thus, no *a priori* traffic

engineering is required. Further, it can be implemented in a decentralized manner in wireline networks; we will comment on implementability in wireless networks later. However, since the back-pressure algorithm requires per-destination queues at each node, its memory requirements and signalling overhead could be of the same order as the number of nodes in the network, thus preventing its wide-spread implementation.

To obtain a scalable algorithm, a simple idea is to group nodes into clusters, and maintain per-cluster queues for inter-cluster routing and only maintain per-destination queues for intra-cluster routing. Such hybrid routing is similar to the hierarchical routing used in the Internet and the Zone Routing Protocol (ZRP) [16] suggested for wireless networks. A simple cluster-based routing algorithm can work as follows. First the gateways of a cluster have to be identified and announced to the rest of the network. A node is said to be a gateway of a cluster if there exists a link connecting the node with a node in another cluster. Now, consider the following routing protocol for a flow $[s, d]$: (i) first route packets from source s to any gateway of the cluster containing destination d, and (ii) then route the packets to destination d from the gateways. Under this naive scheme, packets destined to the same cluster are indistinguishable to a node outside the destination cluster, and thus, each node only has to maintain a separate queue for every other cluster and a separate queue for each node within the same cluster, which reduces the number of queues at each node. However, this simple idea could significantly degrade the set of supportable $\mathbf{x}$. Consider a network as in Figure 1 where each link has a unit capacity. We group nodes 1 and 2 into cluster A, and nodes $3, 4, 5$ and 6 into cluster B. There are two flows ($[1, 6]$ and $[1, 5]$) in the network, and we assume $x_1^5 = x_1^6 = 1/3$. Using the simple cluster-based

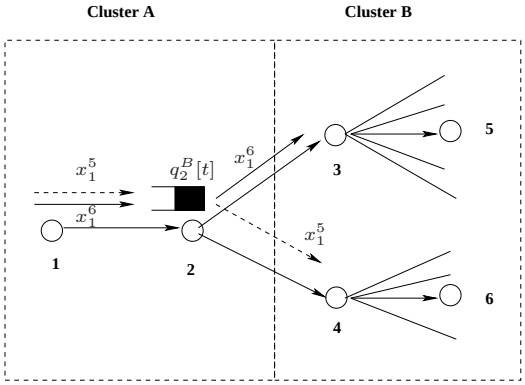

Fig. 1. Simple cluster-based routing

routing scheme, node 2 maintains a single queue for both flows. Since the goal of node 2 is to route packets to any gateway of cluster B, in the worst case, node 2 may transmit all packets of flow $[1, 5]$ to node 4, and all packets of flow $[1, 6]$ to node 3 as shown in Figure 1. Then, no packet can be delivered to its destination even though it is trivial to note that there exist routes between the source and destinations to support the given arrival rates. Thus, maintaining one queue per cluster does not work since packets could be routed to the gateways that are far from or even not connected to the destinations.

On the other hand, it is true that a flow across different clusters does need to be routed to some gateway(s) before reaching its destination. Thus if we can *properly* split the flows and assign them to the *right* gateways, we should be able to support any set of arrival rates within the capacity region. For example, let $x_s^{r,d}$ denote the amount of traffic of flow $[s, d]$ assigned to gateway r, which means that the traffic needs to be routed to node r before reaching the destination. If we split x_1^6 into $x_1^{4,6} = x_1^6 = 1/3$ and $x_1^{3,6} = 0$, and x_1^5 into $x_1^{3,5} = x_1^5 = 1/3$ and $x_1^{4,5} = 0$ as in Figure 2, then the flows can be well-supported in the network. In this case, flows

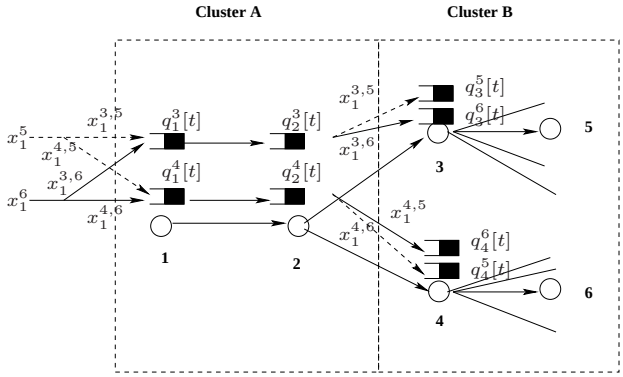

Fig. 2. Properly split the flows

destined to cluster B are first routed to the assigned gateways, so the nodes in cluster A only need to have a separate queue for each gateway (node 3 and node 4) of cluster B instead of a separate queue for *each destination* in cluster B. This number is significantly reduced compared with the original back-pressure algorithm.

Now the question is how to "properly" split flows. Finding the optimal split requires complete information about the network traffic and topology, which is not only difficult to obtain, but also not desirable to use since the split has to be recomputed every time the traffic or topology changes. Therefore, we propose a traffic controller at each source to dynamically allocate the packets to gateways. Note that the queue length at node n indicates the congestion level at node n, so $q_s^r[t] + q_r^d[t]$ reflects the congestion level of reaching destination d via gateway r. Thus at each time slot, our traffic controller assigns the arriving packets to the least congested gateway choice, i.e., deposits the packets into queue r with the smallest $q_s^r[t] + q_r^d[t]$. For example, $q_1^4[t] + q_4^5[t]$ measures the congestion level of routing flow $[1, 5]$ via node 4. Since packets of flow $[1, 5]$ cannot be delivered once routed to node 4, $q_4^5[t]$ will build up. Then the traffic controller will eventually set $x_1^{3,5}[t] = x_1^5$ and flow $[1, 5]$ will be routed to node 3, which is optimal as discussed in the previous paragraph.

In the following sections, we will use the intuition gained from the simple example to present a scalable cluster-based back-pressure algorithm for general topology networks.

III. Cluster-Based Back-Pressure Algorithm

Assume that the network is divided into non-overlapping clusters. Let $\mathcal{C}(n)$ denote the cluster containing node n. Node n is defined to be a gateway of cluster $\mathcal{C}(n)$ if there exists a node m such that $m \notin \mathcal{C}(n)$ and $(m,n) \in \mathcal{L}$, and node n is defined to be an interior-node of cluster $\mathcal{C}(n)$ if node n is not a gateway. As in Figure 3, node 1 is a gateway and node 2 is an interior-node. Denote by $\mathcal{B}_{\mathcal{C}}$ the set of gateways of cluster $\mathcal{C}$, and $\mathcal{I}_{\mathcal{C}}$ the set of interior-nodes of cluster $\mathcal{C}$. As in Figure 3, $\mathcal{B}_A = \{1,6\}$ and $\mathcal{I}_A = \{2,3,4,5\}$. Denote by $\mathcal{Q}_n$ the set

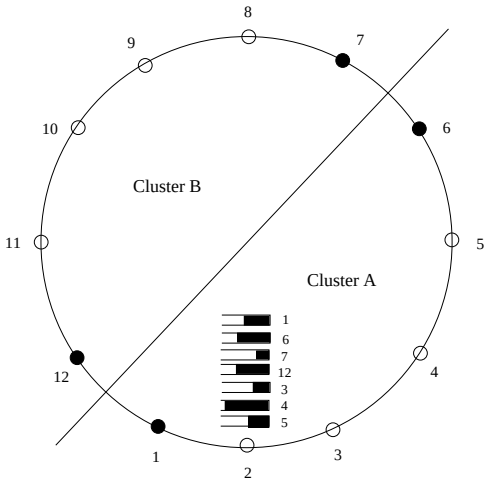

Fig. 3. Example of clustering

of all gateways in the network and the interior-nodes of the cluster containing node n except node n itself, i.e.,

$$\mathcal{Q}_n = \cup_{\mathcal{C}} \mathcal{B}_{\mathcal{C}} \cup \mathcal{I}_{\mathcal{C}(n)} \setminus \{n\}.$$

As in Figure 3, $\mathcal{Q}_2 = \{1,6,7,12\} \cup \{3,4,5\}$. In the cluster-based back-pressure algorithm, node n only needs to maintain a separate queue for each node in $\mathcal{Q}_n$. This could be, in general, far less than the total number of nodes in the network.

As discussed in the previous section, the traffic controller at source s dynamically splits flow $[s,d]$ at each time slot. If s and d are in different clusters or source s is a gateway of cluster $\mathcal{C}(d)$, then every packet of flow $[s,d]$ needs to be relayed to some gateway before being delivered to the destination. In this case, the traffic controller needs to decide which gateway to relay to. We define

$$Q_s^{r,d}[t] = q_s^r[t] + q_r^d[t]$$

for each gateway r of $\mathcal{C}(d)$, which reflects the congestion level of reaching destination d via gateway r. If source s is an interior-node of cluster $\mathcal{C}(d)$, then there is an additional option: directly route packets to destination d without relaying to any gateway. We then define

$$Q_s^{0,d}[t] = q_s^d[t]$$

for $s \in \mathcal{I}_{\mathcal{C}(d)}$, which measures the congestion level of this option. We also define $Q_s^{0,d}[t] = \infty$ if $s \notin \mathcal{I}_{\mathcal{C}(d)}$ to indicate that the direct delivery option is feasible only when s is an interior-node of $\mathcal{C}(d)$. The traffic controller makes the traffic control decision based on $\{Q_s^{r,d}[t]\}_{r \in \{0\} \cup \mathcal{B}_{\mathcal{C}(d)}}$.

The algorithm consists of the three components described next.

Cluster-based back-pressure routing algorithm:

Traffic Controller: At time slot t, node s splits the external arrival x_s^d into $\{x_s^{r,d}[t]\}_{r \in \{0\} \cup \mathcal{B}_{\mathcal{C}(d)}}$ such that

$$x_s^{r,d}[t] = \begin{cases} x_s^d, & r = r^* \\ 0, & r \neq r^* \end{cases}, \tag{3}$$

where

$$r^* = \arg \min_{r \in \{0\} \cup \mathcal{B}_{\mathcal{C}(d)}} Q_s^{r,d}[t].$$

Note that r^* indicates the least congested gateway choice when $r^* > 0$, and $r^* = 0$ indicates that the best option is to directly route to destination d without relaying to any gateway. After obtaining $\{x_s^{r,d}[t]\}_{r \in \{0\} \cup \mathcal{B}_{\mathcal{C}(d)}}$, node n deposits $x_s^{r,d}[t]$ packets into queue r at node s at time slot t.

Regulator: Gateway r maintains a real-queue and a regulated-queue for each destination d in the interior of cluster $\mathcal{C}(r)$, as in Figure 4. At each time slot, the rate at which packets can

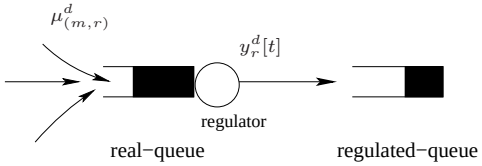

Fig. 4. Regulator at node r for destination d

be transferred from the real-queue to the regulated-queue is

$$y_r^d[t] = (1+\delta) \sum_{s:[s,d] \in \mathcal{F}} x_s^{r,d}[t],$$

where $\delta > 0$ can be chosen to be arbitrarily small. We denote by $q_r^d[t]$ the length of the regulated-queue, and $\hat{q}_r^d[t]$ the length of the real-queue.

Back-Pressure Scheduler: At each time slot, the network first computes $\boldsymbol{\mu}[t]$ which is the solution to the following optimization problem:

$$\boldsymbol{\mu}[t] = \arg \max_{\boldsymbol{\mu} \in \Gamma} \sum_{(m,n) \in \mathcal{L}} \mu_{(m,n)}[t] P_{m,n}[t], \tag{4}$$

where

$$P_{(m,n)}[t] = q_m^{j^*_{(m,n)}[t]}[t] - q_n^{j^*_{(m,n)}[t]}[t]$$

and

$$j^*_{(m,n)}[t] = \begin{cases} \arg \max\limits_{r \in \cup_{\mathcal{C}} \mathcal{B}_{\mathcal{C}}} (q_m^r[t] - q_n^r[t]) & \text{if } n \in \cup_{\mathcal{C}} \mathcal{B}_{\mathcal{C}} \\ \arg \max\limits_{j \in \mathcal{Q}_m} (q_m^j[t] - q_n^j[t]) & \text{otherwise} \end{cases}.$$

After obtaining $\mu_{(m,n)}[t]$ and $j^*_{(m,n)}[t]$, node m transmits the packets in queue $j^*_{(m,n)}[t]$ to node n over link (m,n) at rate $\mu_{(m,n)}[t]$. If $j^*_{(m,n)}[t] \neq n$, the packets are deposited into queue $j^*_{(m,n)}[t]$ at node n. If $j^*_{(m,n)}[t] = n$ and n is a gateway, then the packets in queue $j^*_{(m,n)}[t]$ at node m are

either destined for node n or routed to their destinations via gateway n. In the latter case, the packets with destination d are deposited in the real-queue maintained for node d at node n. □

Remarks: A few clarifying remarks are in order regarding the algorithm:

(i) **Complexity of scheduling:** In wireline networks where simultaneous transmissions on different links do not interfere with each other, the solution of optimization problem (4) is

$$\mu_{(m,n)}[t] = \eta_{(m,n)}.$$

Recall that $\eta_{(m,n)}$ is the capacity of link (m,n). So in wireline networks, node m simply uses the full capacity of link (m,n) to serve queue $j^*_{(m,n)}[t]$ at time slot t, and the back-pressure scheduler is totally distributed.

In wireless networks, solving (4) requires centralized information and could also be computationally very complex. However, there has been a lot of recent activity on distributed implementation of the scheduler [13], [23], [4], [18]. We can use these distributed scheduling algorithms to solve (4). However, the throughput might degrade if the distributed scheduler can only support a fraction of the capacity region as is the case in some of the recent papers.

(ii) **Signalling overhead:** The traffic controllers require queue length information $q_r^d[t]$ at time slot t, and the regulators need traffic control information $x_s^{r,d}[t]$ at time slot t. Both of these pieces of information can be easily collected using ideas similar to the way that price is collected in many optimization-based flow control algorithms. Let us first consider the regulator information $x_s^{r,d}[t]$. Each flow could append its current rate $x_s^{r,d}[t]$ to its packet header. The gateway node could simply add these rates to compute the output rate of the regulator. One difficulty with this implementation is in making sure that the rate of each flow is counted only once within each time slot. To rectify this problem, each source could announce its rate only once every T slots, thus avoiding double counting. This introduces delays in the algorithm which we will deal with in the next section. We note that the use of the regulator is only to prove the stability of the cluster-based algorithm. We believe that the regulator may not be necessary in practice. This has to be verified through extensive simulations which will be reported in a longer version of the paper. If the regulator is not required in practice, then the signalling overhead due to the regulator can be eliminated. On the other hand, the queue length information $q_r^d[t]$ needed at the source is crucial for the correctness of the algorithm. This information is easily collected by appending it to the packet header and then echoing it back to the source from the destination by piggy-backing it on to acknowledgments. Again this introduces a delay in the algorithm, but does not affect network stability as will be shown in the next section.

(iii) **Hierarchical routing:** After a packet from flow $[s,d]$ is deposited in the queue maintained for gateway r by the traffic controller, it will be placed in queue r at other nodes as determined by the back-pressure scheduler. So before reaching node r, the packet is routed as though it is destined to gateway r. Then after the packet reaches node r, it will be deposited in the queue maintained for destination d. Since only the nodes in cluster $\mathcal{C}(d)$ maintain queue d, the packet is routed within the cluster to the destination.

(iv) **Clustering:** The inter-cluster and intra-cluster topology can be arbitrary. The only requirement is that the identity of the gateway nodes of a cluster be known to other clusters. Within each cluster, the identity of every node has to be known to every other node in the cluster to allow per-destination queueing within a cluster.

Next we analyze the performance of the cluster-based back-pressure algorithm. We first present a lemma showing that for any supportable $\mathbf{x}$, there exists a traffic split $\{x_s^{r,d}[t]\}_{r\in\{0\}\cup\mathcal{B}_{\mathcal{C}(d)}}$ that is also supportable by the network, where $x_s^{r,d}$ for $r \in \mathcal{B}_{\mathcal{C}(d)}$ is the amount of traffic from flow $[s,d]$ that has to be routed to gateway r before reaching the destination, and $x_s^{0,d}$ is the amount of traffic that needs to be directly routed to destination d without relaying to any gateway (that is a feasible option only when s is an interior node of $\mathcal{C}(d)$ as discussed). The intuition behind this lemma is clear: any packet of flow $[s,d]$ needs to be either relayed to some gateway, or directly routed to the destination. However, the proof of the existence of such a splitting is non-trivial, and the detailed proof is presented in [24]. In the following lemma, $\mu^r_{(m,n)}[t]$ denotes the rate used to transmit packets in queue r at node m over link (m,n).

Lemma 1: Given $\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$, there exist $\{x_s^{r,d}\}_{r\in\{0\}\cup\mathcal{B}_{\mathcal{C}(d)}}$ and $\boldsymbol{\mu} \in \Gamma$ such that

(i) For any flow $[s,d] \in \mathcal{F}$,

$$x_s^{0,d}1_{s\in\mathcal{I}_{\mathcal{C}(d)}} + \sum_{r\in\mathcal{B}_{\mathcal{C}(d)}} x_s^{r,d} = x_s^d;$$

(ii) For any node n and destination d such that $n \in \mathcal{B}_{\mathcal{C}}(d)$ and $d \in \mathcal{I}_{\mathcal{C}}(d)$,

$$\sum_{s:[s,d]\in\mathcal{F}} x_s^{n,d} = \sum_{i:(n,i)\in\mathcal{L}, i\in\mathcal{I}_{\mathcal{C}}(d)} \mu^d_{(n,i)};$$

(iii) For any node n and gateway r such that $n \neq r$,

$$\sum_{d\in\mathcal{C}(r)} x_n^{r,d} + \sum_{m:(m,n)\in\mathcal{L}} \mu^r_{(m,n)} = \sum_{i:(n,i)\in\mathcal{L}} \mu^r_{(n,i)};$$

(iv) For any node n and destination d such that $n \neq d$ and $n \in \mathcal{I}_{\mathcal{C}}(d)$,

$$x_n^{0,d} + \sum_{m:(m,n)\in\mathcal{L}} \mu^d_{(m,n)} = \sum_{i:(n,i)\in\mathcal{L}, i\in\mathcal{I}_{\mathcal{C}}(d)} \mu^d_{(n,i)}.$$

Proof: We refer the interested reader to [24]. ■

As demonstrated in Figure 5, condition (i) implies that $x_s^{r,d}$ is split from flow $[s,d]$; condition (iii) is the flow-conservation constraint of the queue for gateway r at node n; and condition (iv) is the flow-conservation constraint of the queue for destination d at node n. Condition (ii) is the flow-conservation constraint at gateways, which implies that the amount of traffic out from node n and destined to destination d is equal to the amount of flows destined to node d and routed via gateway n.

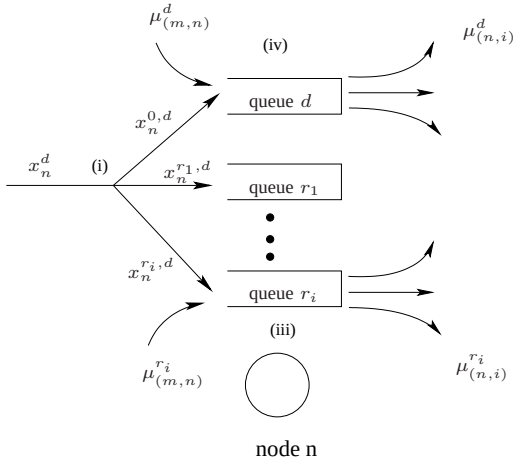

Fig. 5. Flow conservation constraints at node n

To simplify the notations, we define $x^j_{\text{in}(n)}$, $\mu^j_{\text{in}(n)}$ and $\mu^j_{\text{out}(n)}$ such that

$$x^j_{\text{in}(n)} = \begin{cases} \sum_{s:[s,j]\in\mathcal{F}} x^{n,j}_s & \text{if } n \in \cup_{\mathcal{C}}\mathcal{B}_{\mathcal{C}} \text{ and } j \in \mathcal{I}_{\mathcal{C}(n)} \\ \sum_{d\in\mathcal{C}(j)} x^{j,d}_n & \text{if } j \in \cup_{\mathcal{C}}\mathcal{B}_{\mathcal{C}} \\ x^{0,j}_n & \text{if } n \in \cup_{\mathcal{C}}\mathcal{I}_{\mathcal{C}} \text{ and } j \in \mathcal{I}_{\mathcal{C}(n)} \end{cases} \quad (5)$$

$$\mu^j_{\text{in}(n)} = \begin{cases} 0 & \text{if } n \in \cup_{\mathcal{C}}\mathcal{B}_{\mathcal{C}}, j \in \mathcal{I}_{\mathcal{C}(n)} \\ \sum_{m:(m,n)\in\mathcal{L}} \mu^j_{(m,n)} & \text{otherwise} \end{cases} \quad (6)$$

and

$$\mu^j_{\text{out}(n)} = \begin{cases} \sum_{i:(n,i)\in\mathcal{L}, i\in\mathcal{I}_{\mathcal{C}}(d)} \mu^j_{(n,i)} & \text{if } j \notin \cup_{\mathcal{C}}\mathcal{B}_{\mathcal{C}} \\ \sum_{i:(n,i)\in\mathcal{L}} \mu^j_{(n,i)} & \text{otherwise} \end{cases}. \quad (7)$$

Then flow-conservation constraints (ii)-(iv) can be written as

$$x^j_{\text{in}(n)} + \mu^j_{\text{in}(n)} = \mu^j_{\text{out}(n)}. \quad (8)$$

Based on Lemma 1, we will show that the cluster-based back-pressure algorithm can support any $\mathbf{x}$ such that $(1+\delta+\epsilon)\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$ for some $\epsilon > 0$. Recall that δ is a parameter defined in the regulator. We first introduce more notations. For time slot t, we define $x^j_{\text{in}(n)}[t]$, $\mu^j_{\text{in}(n)}[t]$, and $\mu^j_{\text{out}(n)}[t]$ for $j \in \mathcal{Q}_n$ as the arrival and service rates at time t. Note that the actual number of packets with destination j and transmitted over link (m,n) at time t may be less than $\mu^j_{(m,n)}[t]$ when there are not enough packets in queue j at node m, so we define $u^j_n[t]$ to be the difference between the unused service rate at node n and the fictitious arrivals to node n (due to the unused service at the incoming links). Then if queue j is a regulated-queue, we have

$$q^j_n[t+1] - q^j_n[t] = y^j_n[t] + \mu^j_{\text{in}(n)}[t] - \mu^j_{\text{out}(n)}[t] + u^j_n[t];$$

and otherwise,

$$q^j_n[t+1] - q^j_n[t] = x^j_{\text{in}(n)}[t] + \mu^j_{\text{in}(n)}[t] - \mu^j_{\text{out}(n)}[t] + u^j_n[t].$$

Since

$$y^j_n[t] = (1+\delta)\sum_{s:[s,d]\in\mathcal{F}} x^{n,j}_s[t] = (1+\delta)x^j_{\text{in}(n)}[t],$$

we can obtain that

$$\begin{aligned} &\Delta q^j_n[t] := q^j_n[t+1] - q^j_n[t] \\ \leq\ & (1+\delta)x^j_{\text{in}(n)}[t] + \mu^j_{\text{in}(n)}[t] - \mu^j_{\text{out}(n)}[t] + u^j_n[t]. \end{aligned} \quad (9)$$

Next define

$$\eta_{\max} = \max_n \left(\sum_{m:(m,n)\in\mathcal{L}} \eta_{(m,n)} + \sum_{i:(n,i)\in\mathcal{L}} \eta_{(n,i)} \right),$$

which is the maximum amount of queue length change allowed at each time slot. It can be easily verified that

$$|u^j_n[t]| \leq \eta_{\max}$$

for any j and t, and given $\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$,

$$\left|q^j_n[t+1] - q^j_n[t]\right| \leq \eta_{\max} \quad (10)$$

for any queue j.

Theorem 2: Given external arrivals $\mathbf{x}$ such that $(1+\delta+\epsilon)\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$ for some $\epsilon > 0$, all queues are bounded under the cluster-based back-pressure algorithm.

Proof: Define a Lyapunov function $V[t]$ as follows:

$$V[t] = \sum_n \sum_{j\in\mathcal{Q}_n} \left(q^j_n[t]\right)^2.$$

The drift of $V[t]$ is defined as

$$\begin{aligned} \Delta V[t] &:= V[t+1] - V[t] \\ &= \sum_n \sum_{j\in\mathcal{Q}_n} \left(q^j_n[t+1] + q^j_n[t]\right) \Delta q^j_n[t] \\ &\leq \sum_n \sum_{j\in\mathcal{Q}_n} \left(q^j_n[t+1] + q^j_n[t]\right) \times \\ &\quad \left((1+\delta)\, x^j_{\text{in}(n)}[t] + \mu^j_{\text{in}(n)}[t] - \mu^j_{\text{out}(n)}[t] + u^j_n[t]\right) \\ &\leq \sum_n \sum_{j\in\mathcal{Q}_n} \left(2q^j_n[t] + \eta_{\max}\right) u^j_n[t] + \\ &\quad \sum_n \sum_{j\in\mathcal{Q}_n} \left(2q^j_n[t] + \eta_{\max}\right) \times \\ &\quad \left((1+\delta)\, x^j_{\text{in}(n)}[t] + \mu^j_{\text{in}(n)}[t] - \mu^j_{\text{out}(n)}[t]\right). \end{aligned}$$

Define

$$A_{\mathbf{x}}[t] = \sum_n \sum_{j\in\mathcal{Q}_n} q^j_n[t] x^j_{\text{in}(n)}[t]$$

and

$$B_{\boldsymbol{\mu}}[t] = \sum_n \sum_{j\in\mathcal{Q}_n} q^j_n[t] \left(\mu^j_{\text{out}(n)}[t] - \mu^j_{\text{in}(n)}[t]\right).$$

Note that $u^j_n[t] = 0$ if $q^j_n[t] \geq \eta_{\max}$, so

$$\left(2q^j_n[t] + \eta_{\max}\right) u^j_n[t] \leq 3\eta^2_{\max},$$

and

$$\Delta V[t] \leq K + 2(1+\delta)A_{\mathbf{x}}[t] - 2B_{\boldsymbol{\mu}}[t]$$

where $K = 4N^2\eta_{\max}^2$ and N is the number of nodes in the network.

From Lemma 1, we have that if $(1+\delta+\epsilon)\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$, there exist $\{x_s^{r,d}\}$ and $\boldsymbol{\mu} \in \Gamma$ such that

$$\begin{aligned} x_s^d &= x_s^{0,d}1_{s\in\mathcal{I}_{\mathcal{C}(d)}} + \sum_{r\in\mathcal{B}_{\mathcal{C}(d)}} x_s^{r,d}; \\ 0 &= \sum_n \sum_{j\in\mathcal{Q}_n} q_n^j[t]\left((1+\delta+\epsilon)x_{\text{in}(n)}^j + \mu_{\text{in}(n)}^j - \mu_{\text{out}(n)}^j\right). \end{aligned}$$

Letting

$$\tilde{A}_{\mathbf{x}}[t] = \sum_n \sum_{j\in\mathcal{Q}_n} q_n^j[t]x_{\text{in}(n)}^j,$$

and

$$\tilde{B}_{\boldsymbol{\mu}}[t] = \sum_n \sum_{j\in\mathcal{Q}_n} q_n^j[t]\left(\mu_{\text{out}(n)}^j - \mu_{\text{in}(n)}^j\right),$$

we have

$$(1+\delta+\epsilon)\tilde{A}_{\mathbf{x}}[t] = \tilde{B}_{\boldsymbol{\mu}}[t],$$

and

$$\begin{aligned} \Delta V[t] &\leq K + 2(1+\delta)A_{\mathbf{x}}[t] - 2B_{\boldsymbol{\mu}}[t] \\ &= K + 2(1+\delta)A_{\mathbf{x}}[t] - 2(1+\delta+\epsilon)\tilde{A}_{\mathbf{x}}[t] \\ &\quad +2\tilde{B}_{\boldsymbol{\mu}}[t] - 2B_{\boldsymbol{\mu}}[t] \\ &\leq_{(a)} K + 2(1+\delta)\left(A_{\mathbf{x}}[t] - \tilde{A}_{\mathbf{x}}[t]\right) - 2\epsilon\tilde{A}_{\mathbf{x}}[t] \\ &\leq_{(b)} K - 2\epsilon\tilde{A}_{\mathbf{x}}[t], \end{aligned} \tag{11}$$

where inequality (a) holds since

$$\tilde{B}_{\boldsymbol{\mu}}[t] \leq B_{\boldsymbol{\mu}}[t], \tag{12}$$

and inequality (b) holds since

$$\tilde{A}_{\boldsymbol{\mu}}[t] \geq A_{\boldsymbol{\mu}}[t]. \tag{13}$$

Inequalities (12) and (13) are established in [24].

Inequality (11) implies that $\Delta V[t] < -K$ if $\tilde{A}_{\mathbf{x}}[t] > K/\epsilon$. If $\tilde{A}_{\mathbf{x}}[t] \leq K/\epsilon$, from inequality (13), we get

$$\begin{aligned} \Delta V[t] &\leq K + 2(1+\delta)A_{\mathbf{x}}[t] - 2B_{\boldsymbol{\mu}}[t] \\ &\leq \tilde{K} - 2B_{\boldsymbol{\mu}}[t], \end{aligned}$$

where $\tilde{K} = K + 2(1+\delta)K/\epsilon$. Define $\eta_{\min}$ to be the smallest link capacity of all links, i.e., $\eta_{\min} = \min_{(m,n)\in\mathcal{L}} \eta_{(m,n)}$. According to the back-pressure scheduler (4), we have that

$$B_{\boldsymbol{\mu}}[t] \geq \tilde{K}$$

if there exist $q_m^j[t]$ and $q_n^j[t]$ such that $(m,n) \in \mathcal{L}$ and

$$q_m^j[t] - q_n^j[t] \geq \frac{\tilde{K}}{\eta_{\min}}. \tag{14}$$

Since $q_n^n[t] = 0$ for any n, it is easy to verify that inequality (14) holds if there exists $q_n^j[t]$ such that

$$q_n^j[t] \geq \frac{N\tilde{K}}{\eta_{\min}}.$$

Thus we obtain that $\Delta V[t] \leq -K$ if there exists $q_n^j[t] \geq \frac{N\tilde{K}}{\eta_{\min}}$, which further implies that there exists t_0 such that for any $t > t_0$,

$$V[t] \leq \bar{K}^2$$

where $\bar{K} = N\left(N\tilde{K}/\eta_{\min} + \eta_{\max}\right)$. From the definition of $V[t]$, we can conclude that for any $t \geq t_0$,

$$q_n^j[t] \leq \sqrt{V[t]} \leq \bar{K}. \tag{15}$$

Next we consider $\{\hat{q}_r^d[t]\}$, which are the lengths of the real-queues at gateways. Let $M_r^d[t]$ denote the total number of packets that are destined to node d via gateways r, and have not reached the regulated queue for node d at gateway r yet. Assume that there exists $t > t_0$ such that

$$M_r^d[t] > N\bar{K} + \eta_{\max},$$

which implies

$$\hat{q}_r^d[t] > \eta_{\max}$$

due to inequality (15). Thus we can get that

$$M_r^d[t+1] = \sum_{s:[s,d]\in\mathcal{F}} x_s^{r,d}[t] + M_r^d[t] - y_r^d[t] \leq M_r^d[t],$$

which implies that that for $t > t_0$,

$$M_r^d[t] \leq \max\{M_r^d[t_0], N\bar{K} + 2\eta_{\max}\}.$$

Thus all the elements of $\{\hat{q}_r^d[t]\}$ are bounded as well. ■

IV. Fast Schedule with Slow Traffic Control and Regulation

As discussed in the remarks after the presentation of the cluster-based back-pressure algorithm, it is difficult to obtain the instantaneous $q_r^d[t]$ at source nodes and $x_s^{r,d}[t]$ at gateways.

In this section we propose a two-time-scale back-pressure algorithm where the traffic control and regulator run at a slow time-scale, i.e., they update their control parameters every T time slots, which reduces the amount of information exchanged between sources and gateways. We also use delayed information at traffic controllers and regulators to account for the propagation delays. The scheduler is still implemented at a fast time-scale, i.e., the scheduling decision is made at every time slot. Let $\tau_{m,n}$ denote the propagation delay from node m to node n, and $\tau_{\max} = \max_{m,n}\tau_{m,n}$. We choose T such that $T > 3\tau_{\max}$. The information is exchanged as follows.

(i) At time slot hT, source s sends a request message (RM) to every gateway of cluster $\mathcal{C}(d)$. The message includes the identity of destination d. The RM packet does not have to be a special packet, it can be piggy-backed on a data packet.

(ii) When gateway r receives the RM, it checks the destination identity, and sends an acknowledgment (ACK) back to the source with the queue length information $q_r^d[hT]$ included. These packets are received at the corresponding sources before time slot $hT + 2\tau_{\max}$. These ACK packets need not be special packets either, they can be piggy-backed on TCP acks for example.

(iii) When source s receives $\{q_r^d[hT]\}$ from all gateways, source s computes $\tilde{x}_s^{r,d}[hT]$ based on $q_s^r[hT]$ and $q_r^d[hT]$. After that, source s sends a control message (CM) to gateway r. Again the CM can be piggy-backed on a data packet. The control message contains the identity of destination d and the value of $\tilde{x}_s^{r,d}[hT]$. Gateway r receives the CM before time slot $hT+3\tau_{\max}$.

(iv) After gateway r receives the CM, it checks the destination identity in the CM, and adds up all $\tilde{x}_s^{r,d}[hT]$ with destination d.

(v) From time slot $(h+1)T$ to time slot $(h+2)T-1$, the sources and regulators control and regulate the traffic based on $\{\tilde{x}_s^{r,d}[hT]\}$. Since $T > 3\tau_{\max}$, $\{\tilde{x}_s^{r,d}[hT]\}$ are available at the sources and regulators at time slot $(h+1)T$.

Based on the above information exchange, the two-time-scale cluster-based back-pressure algorithm is as follows.

Two-Time-Scale Cluster-Based Back-Pressure Algorithm:

Traffic Controller: During time slots $\{hT, \ldots, (h+1)T-1\}$, the source of flow $[s,d]$ receives $\{q_r^d[hT]\}$ from the gateways of the cluster containing destination d, and computes $\{\tilde{x}_s^{r,d}[hT]\}_{r\in\{0\}\cup\mathcal{B}_{\mathcal{C}(d)}}$ such that

$$\tilde{x}_s^{r,d}[hT] = \begin{cases} x_s^d, & r = r^* \\ 0, & r \neq r^* \end{cases}, \tag{16}$$

where $r^* = \arg\min_{r\in\{0\}\cup\mathcal{B}_{\mathcal{C}(d)}} Q_s^{r,d}[hT]$.

For time slot $t \in \{hT, \ldots, (h+1)T-1\}$, node s splits flow x_s^d according to $\tilde{x}_s^{r,d}[(h-1)T]$, i.e.,

$$x_s^{r,d}[t] = \tilde{x}_s^{r,d}[(h-1)T].$$

Regulator: Each gateway r maintains a real-queue and a regulated-queue for each destination d in the interior of cluster $\mathcal{C}(r)$. During time slot $\{(h-1)T, \ldots, hT-1\}$, node r receives the information of $\tilde{x}_s^{r,d}[(h-1)T]$. During time slot $t \in \{hT, \ldots, (h+1)T-1\}$, the rate at which packets can be transferred from the real-queue to the regulated-queue is set to be

$$y_r^d[t] = (1+\delta) \sum_{s:[s,d]\in\mathcal{F}} \tilde{x}_s^{r,d}[(h-1)T].$$

Back-Pressure Scheduler: At each time slot, the network computes $\boldsymbol{\mu}[t]$, which solves the following optimization problem:

$$\boldsymbol{\mu}[t] = \arg\max_{\boldsymbol{\mu}\in\Gamma} \sum_{(m,n)\in\mathcal{L}} \mu_{(m,n)}[t] P_{m,n}[t]. \tag{17}$$

Then a service rate $\mu_{(m,n)}[t]$ is allocated to queue $j^*_{(m,n)}[t]$ at node m to transmit packets over link (m,n) to queue $j^*_{(m,n)}[t]$ at node n at time slot t. $P_{(m,n)}[t]$ and $j^*_{(m,n)}[t]$ are defined as in the cluster-based back-pressure algorithm. □

Theorem 3: Given $\mathbf{x}$ such that $(1+\delta+\epsilon)\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$ for some $\epsilon > 0$, all queues are bounded under the two-time-scale cluster-based back-pressure algorithm.

Proof: Define a Lyapunov function $V[h]$ such that

$$V[h] = \sum_n \sum_{j\in\mathcal{Q}_n} \left(q_n^j[hT]\right)^2.$$

Then the drift of $V[h]$ is

$$\begin{aligned} \Delta V[h] &:= V[h+1] - V[h] \\ &= \sum_n \sum_{j\in\mathcal{Q}_n} \left(q_n^j[(h+1)T] + q_n^j[hT]\right) \times \\ &\qquad \left(q_n^j[(h+1)T] - q_n^j[hT]\right), \end{aligned}$$

where

$$\begin{aligned} q_n^j[(h+1)T] - q_n^j[hT] \leq \sum_{s=hT}^{(h+1)T-1} \Big(&(1+\delta)\, x_{\text{in}(n)}^j[s] + \\ &\mu_{\text{in}(n)}^j[s] - \mu_{\text{out}(n)}^j[s] + u_n^j[s] \Big). \end{aligned}$$

From the definitions of $A_{\mathbf{x}}[t]$ and $B_{\boldsymbol{\mu}}[t]$, and the fact that $|q_n^j[s+1] - q_n^j[s]| \leq \eta_{\max}$, it is easy to verify that

$$\Delta V[h] \leq K_1 + 2(1+\delta)T A_{\mathbf{x}}[(h-1)T] - 2 \sum_{s=hT}^{(h+1)T-1} B_{\boldsymbol{\mu}}[s],$$

where $K_1 = 4N^2(T+2)^2\eta_{\max}^2$.

From Lemma 1, given $(1+\delta+\epsilon)\mathbf{x} \in \mathbf{X}_{\mathcal{G}}$, there exist $\mathbf{x}_s^{r,d}$ and $\boldsymbol{\mu} \in \Gamma$ such that

$$(1+\delta+\epsilon)\tilde{A}_{\mathbf{x}}[(h-1)T] = \tilde{B}_{\boldsymbol{\mu}}[(h-1)T].$$

Also it is easy to see that for $s \in \{hT, \ldots, (h+1)T-1\}$,

$$\tilde{B}_{\boldsymbol{\mu}}[(h-1)T] \leq N^2 T \eta_{\max}^2 + \tilde{B}_{\boldsymbol{\mu}}[s].$$

Thus, we can conclude that

$$\begin{aligned} &\Delta V[h] \\ \leq\ & K_1 + 2(1+\delta)T A_{\mathbf{x}}[(h-1)T] - 2 \sum_{s=hT}^{(h+1)T-1} B_{\boldsymbol{\mu}}[s] \\ =\ & K_2 + 2(1+\delta)T(A_{\mathbf{x}}[(h-1)T] - \tilde{A}_{\mathbf{x}}[(h-1)T]) \\ & -2\epsilon T \tilde{A}_{\mathbf{x}}[(h-1)T] - 2 \sum_{s=hT}^{(h+1)T-1} \Big(B_{\boldsymbol{\mu}}[s] - \tilde{B}_{\boldsymbol{\mu}}[s]\Big) \\ \leq\ & K_2 - 2\epsilon T \tilde{A}_{\mathbf{x}}[(h-1)T], \end{aligned} \tag{18}$$

where $K_2 = K_1 + 2N^2T^2\eta_{\max}^2$, and the last inequality follows from inequalities (12) and (13). Based on inequality (18), we can show that all queues are bounded following the argument used in Theorem 2. ■

V. Conclusions and Future Work

In this paper, we proposed a scalable, distributed, and adaptive routing algorithm — the cluster-based back-pressure algorithm. The algorithm retains the throughput-optimality of the original back-pressure algorithm, and also significantly reduces the memory complexity by reducing the number of queues maintained at each node. The algorithm can be extended to the following directions. Future research on this topic includes:

A. Stochastic models

For ease of exposition, we only analyzed networks with deterministic arrivals and time-invariant links. However, our analysis can be extended to networks with stochastic arrivals and time-varying links. The algorithm is expected to be throughput-optimal for stochastic models as-well.

B. Multilevel clustering

In this paper, we only study single-level clustering. It is easy to see that we could further form clusters within each cluster. Multilevel clustering could further reduce the number of queues maintained at each node, but would require the loose source routing to specify multiple intermediate nodes, and the traffic controller would have to collect queue length information from these intermediate nodes.

C. Asynchronous version

The algorithms presented in this paper will be implemented asynchronously in a real network. We expect the algorithm to be throughput-optimal even in this case, but the proofs have to be modified to verify this.

D. Randomized algorithms for reducing signalling overhead

In our algorithms, a source needs to contact all gateways of the cluster containing the destination once every T time slots. We could further reduce this overhead by only contacting two gateways every T time slots: one which the source is using and another one that is randomly picked. Then the source picks the less congested gateway. This approach reduces the amount of information exchanged, but could result in longer queues. The performance of this approach will be studied in future work.

E. Policy-based routing

We studied a network model where link capacities are the only constraints. In reality, there could be many other constraints due to policies imposed by network administrators. For example, ISPs may prohibit intra-cluster flows (flows whose source and destination are in the same cluster) from being routed out of the cluster. A subject of future work is to customize the cluster-based back-pressure routing algorithm for networks with various policy-constraints and study its performance.

References

[1] B. Awerbuch and T. Leighton. “A simple local control approximation algorithm for multicommodity flow,” *Proc. IEEE Conference on Foundations of Computer Science*, Oct. 1993.

[2] D. Applegate and E. Cohen. “Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs,” *Proc. ACM SIGCOMM*, 2003.

[3] R. Callon. “Use of OSI IS-IS for routing in TCP/IP and dual environments,” Request For Comments (Standard) RFC 1195, Internet Engineering Task Force, December 1990.

[4] A. Eryilmaz, A. Ozdaglar, and E. Modiano, “Polynomial complexity algorithms for full utilization of multi-hop wireless networks,” *Proc. IEEE INFOCOM*, 2007.

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

[6] A. Eryilmaz and R. Srikant, “Joint congestion control, routing and mac for stability and fairness in wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1514–1524, August 2006.

[7] N. Feamster, J. Borkenhagen, and J. Rexford. “Guidelines for interdomain traffic engineering,” *Proc. ACM SIGCOMM*, Oct. 2003.

[8] B. Fortz and M. Thorup. “Internet traffic engineering by optimizing OSPF weights,” *Proc. IEEE INFOCOM*, Mar. 2000.

[9] H. Han, Srinivas Shakkottai, C.V. Hollot, R. Srikant, and D. Towsley. “Overlay TCP for multi-path routing and congestion control,” *IEEE/ACM Transactions on Networking*, **14**(6), Dec. 2006.

[10] F. Kelly and T. Voice. “Stability of end-to-end algorithms for joint routing and rate control,” *Computer Communication Review*, **35**(2), pp. 5–12, 2005.

[11] P. Key, L. Massoulie, and D. Towsley. “Path selection and multipath congestion control,” *Proc. INFOCOM*, May 2007.

[12] X. Lin and N. Shroff, “On the stability region of congestion control,” *Proc. Allerton Conference on Communications, Control and Computing*, 2004.

[13] E. Modiano, D. Shah, and G. Zussman, “Maximizing throughput in wireless networks via gossiping,” in *ACM SIGMETRICS / IFIP Performance*, 2006.

[14] J. Moy. “OSPF Version 2,” Request For Comments (Standard) RFC 2328, Internet Engineering Task Force, April 1998.

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

[16] M. R. Pearlman and Z. J. Haas, “Determining the optimal configuration for the zone routing protocol,” *IEEE Journal on Selected Areas in Communications*, August 1999.

[17] E. Rosen, A. Viswanathan, and R. Callon. “Multiprotocol label switching architecture,” Request For Comments (Standards Track) RFC 3031, Internet Engineering Task Force, January 2001.

[18] S. Sanghavi, L. Bui, and R. Srikant, “Distributed link scheduling with constant overhead,” *Proc. ACM SIGMETRICS*, San Diego, CA, June 2007.

[19] A. Sridharan, R. Guerin, and C. Diot. “Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks,” *IEEE/ACM Transactions on Networking*, **13**(2), April 2005.

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

[21] L. Tassiulas and A. Ephremides. “Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks,” *IEEE Transactions on Automatic Control*, Vol. 37, No. 12, pp. 1936-1949, December 1992.

[22] Y. Wang, Z. Wang, L. Zhang. “Internet traffic engineering without full mesh overlaying,” *Proc. INFOCOM*, April 2001.

[23] Y. Yi, G. de Veciana, and S. Shakkottai, “Learning contention patterns and adapting to load/topology changes in a mac scheduling algorithm,” *Proc. WiMesh*, 2006.

[24] L. Ying, R. Srikant, and D. Towsley. “Cluster-based backpressure routing algorithm,” Technical Report. Available at http://www.ifp.uiuc.edu/~srikant/WorkingPapers/yinsritow07.pdf.

[25] C. Zhang, Z. Ge, J. Kurose, Y. Liu, and D. Towsley. “Optimal routing with multiple traffic matrices: Tradeoff between average case and worst case performance,” *Proc. ICNP 2005*, November 2005.