A Benes Packet Network

Longbo Huang* and Jean Walrand†
* Institute for Interdisciplinary Information Sciences, Tsinghua University, longbohuang@tsinghua.edu.cn
† EECS, University of California, Berkeley, CA, wrl@eecs.berkeley.edu

***Abstract*—Benes networks are constructed with simple switch modules and have many advantages, including small latency and requiring only an almost linear number of switch modules. As circuit-switches, Benes networks are rearrangeably non-blocking, which implies that they are full-throughput as packet switches, with suitable routing.**

Routing in Benes networks can be done by time-sharing permutations. However, this approach requires centralized control of the switch modules and statistical knowledge of the traffic arrivals. We propose a backpressure-based routing scheme for Benes networks, combined with end-to-end congestion control. This approach achieves the maximal utility of the network and requires only four queues per module, independently of the size of the network.

***Index Terms*—Benes Network, Dynamic Control, Stochastic Network Optimization, Queueing**

I. Introduction

Data centers have gradually become one of our most important computing resources. For instance, search engines, web emails such as Gmail and Hotmail, social network websites such as Facebook, and data processing applications such as Hadoop are provided by data centers. Consequently, the networking of servers and resource allocation in data centers have become important problems.

We develop a networking solution, a *Benes packet network*, which consists of a Benes architecture, a flow utility maximization mechanism, and a backpressure-based scheduling algorithm. Specifically, we propose interconnecting the data center servers using a Benes network built with simple commodity switch modules. We formulate the resource allocation objective as a network flow utility maximization problem to guarantee a fair share of the network resources. Lastly, we develop a low-complexity backpressure-based scheduling algorithm, called Grouped-Backpressure (G-BP), to achieve the optimal system performance. The G-BP algorithm is provably optimal and automatically handles changing traffic. Our approach only requires each switch module to maintain *four* queues, independently of the network size, and hence can easily be implemented in practice.

Many papers explore networking solutions for data centers. [1] designs scheduling algorithms for three-stage non-blocking switching fabrics. [2] proposes using a random graph based approach to enable incremental network growth for data centers. [3] proposes a network architecture based on Clos network and random traffic splitting. [4] develops a hierarchical network structure for data centers. [5] uses the preferential attachment approach to design network topologies for data centers. [6] proposes a fat-tree based network architecture. [7] develops a MapReduce-like system based on a cube-like architecture to exploit the in-network aggregation possibilities. [8] designs optical networks for data centers. However, we note that the aforementioned works mostly focus on designing the network architecture and achieving uniform load balancing. Hence, the proposed solutions do not immediately apply to problems where different flows have different service requirements. Moreover, the solutions developed in the above works lack system performance guarantees.

In this work, we aim at obtaining a network solution that combines practicality, generality, provable optimality, and low complexity. Specifically, we propose interconnecting the data center servers by a Benes network. As circuit-switches, Benes networks are known to be rearrangeably non-blocking and can easily be built with only an almost linear number of simple switch modules in the network size [9], [10]. Thus, adopting the Benes network architecture not only guarantees high system throughput and low end-to-end packet delay (if routing and scheduling are done properly), but also eliminates the need for employing expensive switch devices whose cost does not scale easily as the data center size increases. Under the Benes network architecture, we establish a mathematical formulation for determining the allocation of network resources to cope with the heterogeneity of the data traffic service requirements. Our formulation leverages the network utility maximization framework [11], [12], which has been proven to be a general mechanism for handling network resource allocation problems.

Finally, to reap the full benefits of the Benes network architecture and the resource allocation framework in a practical manner, we develop a routing and scheduling algorithm that has provable system performance guarantees and a very low implementation complexity. Our algorithm is constructed based on the recently developed backpressure network optimization technique [13], combined with an end-to-end congestion control mechanism. However, different from previous backpressure algorithms, e.g., [14], [15], [16], which either require that the number of queues each switch module has to maintain is proportional to the network size, or only apply to problems with single-path routing, our algorithm uses a novel *traffic grouping* idea and allows us to use only *four* queues per switch module regardless of the network size. Moreover, our algorithm automatically explores all the possible routes to fully utilize network capacity. These distinct features make our algorithm very suitable for practical implementation.

This paper is organized as follows. In Section II, we present the system model and state our objective. In Section III, we set up the notations. Then, we explain the intuition of our design approach and describe all the needed components of the Group-Backpressure (G-BP) algorithm in Section IV. We present the G-BP algorithm and analyze its performance in Section V. Simulation results are presented in Section VI. We conclude the paper in Section VII.

II. System model

We consider the system shown in Fig. 1, where a Benes network connects a set of communicating servers. In this system, each rectangle is a *switch module* having two input and two output links. Each link has a capacity of 1 packet/slot. The smaller nodes are the *servers*. Traffic flows are generated from the servers on the left, called *input servers*, and are going to the servers on the right, called *output servers*. [1] We assume that the system operates in slotted time, i.e., $t \in \{0, 1, 2, ...\}$.

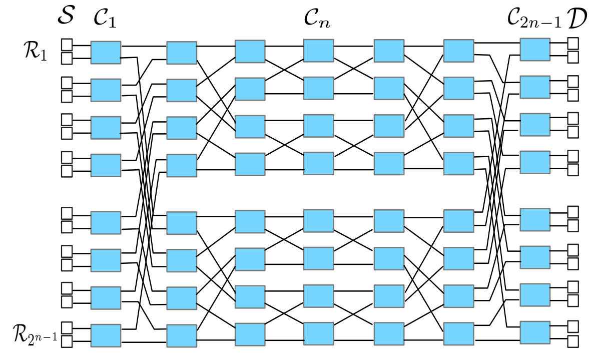


Fig. 1. A 16 × 16 Benes network connecting 16 input servers $\mathcal{S}$ to 16 output servers $\mathcal{D}$. The rectangles are the switch modules that form the Benes network. $\mathcal{R}_i$ refers to row i of the Benes network and $\mathcal{C}_j$ refers to column j.

A. Admission control and flow utility

We label the flows according to their source and destination servers. Specifically, we call the traffic entering from input server s and going to output server d the (s, d) flow. We use $A_{sd}(t)$ to denote the number of (s, d) packets generated at input server s at time t. For every (s, d) flow, the random variables $\{A_{sd}(t), t = 0, 1, \ldots\}$ are i.i.d. and have mean $\lambda_{sd} = \mathbb{E}\big[A_{sd}(t)\big]$. Our results can be extended to incorporate much more general arrival processes, e.g., Makov-modulated arrivals. We also assume that there exists some finite constant $A_{\max}$ such that $0 \le A_{sd}(t) \le A_{\max}$ for all (s, d) and all t.

In every time slot t, each input server performs admission control to determine how many packets to inject into the network. We denote $0 \le R_{sd}(t) \le A_{sd}(t)$ the number of (s, d) flow packets *actually* admitted by input server s for transmission at time t. We then denote the average rate of the (s, d) flow packets by r_{sd}, defined as: [2]

$$r_{sd} \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[R_{sd}(t)\big]. \tag{1}$$

Each (s, d) flow is associated with a utility function $U_{sd}(r_{sd})$, which is concave increasing in its average rate r_{sd}. We assume that the utility functions have finite first derivatives and denote β their maximum value, i.e.,

$$\beta \triangleq \max_{sd} U'_{sd}(0). \tag{2}$$

B. Stability and objective

In this paper, we say that a queue with queue size process $\{Q(t) \ge 0, t = 0, 1, 2, ...\}$ is stable if:

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\big[Q(t)\big] < \infty. \tag{3}$$

Then, we say that a network is stable if all the queues in the network are stable, and call a routing and scheduling policy that ensures network stability a stabilizing policy. We use Λ_n to denote the capacity region of a $2^n \times 2^n$ Benes network, being the set of arrival vectors under which there exist stabilizing routing and scheduling policies.

Depending on the routing and scheduling algorithm, the network queueing structure can be quite different. Our objective is to find a low-implementation-complexity stabilizing routing and scheduling policy that maximizes the aggregate flow utility of the network, i.e.,

$$\begin{aligned} \max: \quad & U(\boldsymbol{r}) \triangleq \sum_{s,d} U_{sd}(r_{sd}) \\ \text{s.t.} \quad & \boldsymbol{r} \in \Lambda_n, \end{aligned} \tag{4}$$

where $\boldsymbol{r} = (r_{sd}, \forall (s, d))$ with r_{sd} being the average rate of the (s, d) flow defined in (1). We denote by $\boldsymbol{r}^{\mathsf{opt}}$ the rate vector that achieves the optimal utility over all stabilizing policies.

Note that our formulation (4) is indeed very general. The heterogeneity of traffic flow service requirements can easily be taken into account by designing appropriate utility functions. Also note that, although our system model is similar to those in [14], in our paper, the queueing structure is also part of the algorithm design problem.

C. Discussion

The problem of optimal routing and scheduling in a Benes network can be solved by using the well-known backpressure routing algorithm [13]. However, this approach requires each node to maintain a separate queue for each output server. Thus, each node has to maintain 2^n queues, which is not practical when the size of the Benes network (number of servers) increases. Recent works [15] and [16] propose backpressure-based algorithms that use much fewer queues. However, the algorithm in [15] requires the network nodes to maintain a separate queue for each cluster of the network nodes and needs a pre-defined clustering algorithm, whereas the method in [16] is designed for single-path routing. Below, we develop a novel low-complexity approach called Grouped-Backpressure (G-BP). Our approach allows us to use only *four* queues per node regardless of the network size.

III. Benes network structure and labeling

In this section, we explain the structure of Benes networks and set up our notations.

[1]It is straightforward to include bi-directional traffic flows.

[2]Throughout this paper, we assume that all the limits exist.

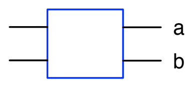
A. Benes network construction

Here we explain how a $2^n \times 2^n$ Benes network is constructed [9] [10]. First, start with a basic 2×2 Benes network as in Fig. 2(a). Then, construct a $2^n \times 2^n$ Benes network as follows:

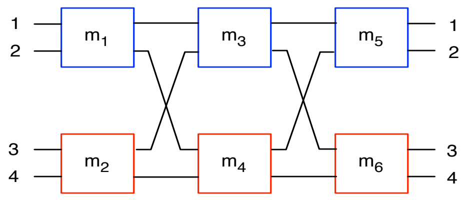
(Step I)-Concatenation: Vertically concatenate two $2^{n-1} \times 2^{n-1}$ Benes networks. Call them the *upper subnetwork* and the *lower subnetwork*, e.g., m_3 and m_4 in Fig. 2(b). Then, horizontally place two columns of 2^{n-1} basic 2×2 modules, one on each side of the concatenated subnetworks. Call the modules on the left of the concatenated subnetworks the *input switch modules*, e.g., m_1 and m_2, and the modules on the right the *output switch modules*, e.g., m_5 and m_6.

(Step II)-Connect input modules: Connect the upper output link of the input module in row k to the k^{th} input link of the upper subnetwork, and connect its lower output link to the k^{th} input link of the lower subnetwork.

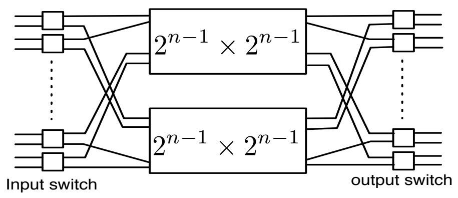
(Step III)-Connect output modules: Connect the k^{th} output link of the upper subnetwork to the upper input link of the k^{th} output module, i.e., the output module in row k, and connect the k^{th} output link of the lower subnetwork to the lower input link of the k^{th} output module.



(a) A basic 2×2 Benes network



(b) A 4×4 Benes network



(c) A general $2^n \times 2^n$ Benes network

Fig. 2. The structure of Benes networks.

B. Labeling a Benes network with servers

We now specify how we label a $2^n \times 2^n$ Benes network. We denote $\mathbb{B}_n$ the $2^n \times 2^n$ Benes network (excluding the input and output servers). Then, we divide the Benes network into rows and columns. In a $2^n \times 2^n$ Benes network, there are 2^{n-1} rows, denoted by $\{\mathcal{R}_i, i = 1, ..., 2^{n-1}\}$. We then denote the $2n-1$ columns by $\{\mathcal{C}_j, j = 1, ..., 2n-1\}$. For any node m in the Benes network, we use i_m and j_m to denote its row number and column number. For the input and output servers connecting to the Benes network, we label them using their row numbers. The set of input servers are denoted by $\mathcal{S} = \{1, 2, ..., 2^n\}$ and the set of output servers are denoted by $\mathcal{D} = \{1, 2, ..., 2^n\}$. Note that both $\mathcal{S}$ and $\mathcal{D}$ have 2^n rows (the small squares in Fig. 1). As in Section III-A, we call the nodes in $\mathcal{C}_1$ the *input switch* modules and the nodes in $\mathcal{C}_{2n-1}$ the *output switch* modules.

From the construction rules of Benes networks and the way the servers are connected to the Benes network, we see that for every node $m \in \mathbb{B}_n$, there are two nodes in column $\mathcal{C}_{j_m+1}$ to which it connects (for a node $m \in \mathcal{C}_{2n-1}$, it connects to two nodes in $\mathcal{D}$). We denote the node with a smaller row number by m_u and the other one by m_l. There are also two nodes in column $\mathcal{C}_{j_m-1}$ that connect to m (if $m \in \mathcal{C}_1$, there are two nodes in $\mathcal{S}$ connecting to it). We denote these two nodes by $\mathcal{M}_m$. Among these nodes, those that have m as their next hop with a smaller row number are denoted by $\mathcal{M}_m^u$, and the other nodes having m as their next hop node with a larger row number are denoted by $\mathcal{M}_m^l$, i.e.,

$$\mathcal{M}_m^u = \{m' \in \mathcal{C}_{j_m-1} \mid m'_u = m\},$$
$$\mathcal{M}_m^l = \{m' \in \mathcal{C}_{j_m-1} \mid m'_l = m\}.$$

Note that $\mathcal{M}_m^u$ and $\mathcal{M}_m^l$ may contain more than one node, e.g., $\mathcal{M}_{m_3}^u$ in Fig. 2. For $m \in \mathcal{C}_1$, we simply use $\mathcal{M}_m$ to denote the input servers that connect to it. For each input server $s \in \mathcal{S}$, we use $m(s)$ to denote the node in $\mathcal{C}_1$ it connects to. We call the servers in rows 1 to 2^{n-1} the *upper division* servers, and call all the other servers the *lower division* servers. We then call a flow whose destination is an upper division server an *upper division flow*. Otherwise it is a *lower division flow*.

For a $2^n \times 2^n$ Benes network $\mathbb{B}_n$, we define the nodes in $\mathcal{C}_n$ as the *partition* nodes. Below, we denote the upper outgoing link of a switch module by link a and the lower outgoing link by link b (see Fig. 2(a)). We use $\mathcal{O}_m^a$ to denote the set of output servers that can be reached by traversing the upper outgoing link a of node m and use $\mathcal{O}_m^b$ to denote the set of output servers that can be reached by traversing link b.

IV. Intuition and Key Components of Grouped-Backpressure

In this section, we present the idea and all the needed components for our Grouped-Backpressure algorithm (G-BP), which will be used to achieve the optimal flow utility under the Benes network architecture.

A. The idea

The idea of Grouped-Backpressure is to "group" all the flows into two mixed flows, the upper division flow and the lower division flow. Then, we construct a scheme for routing the *mixed* traffic in the first half of the network based on a fictitious reference system. This approach allows us to use very few queues per node. However, due to traffic mixing, we lose the ability to control each individual flow inside the network. Hence, the flows can be routed arbitrarily inside the network, in which case certain nodes may receive more traffic than they can handle and become unstable. In order to resolve this problem, we impose a special queueing structure at each node to ensure that routing and scheduling is done in a fully symmetric manner. With this approach, we guarantee that every flow is split into sub-flows with equal rates and routed through the partition nodes. In the second half of the network, by Lemma 3, each packet will traverse a unique path to its destination. Hence, we will do a "free-flow" routing. Using the symmetric structure of the Benes network, we then show that the G-BP algorithm can stabilize the network and achieve maximum utility. Our approach is demonstrated in Fig. 3.

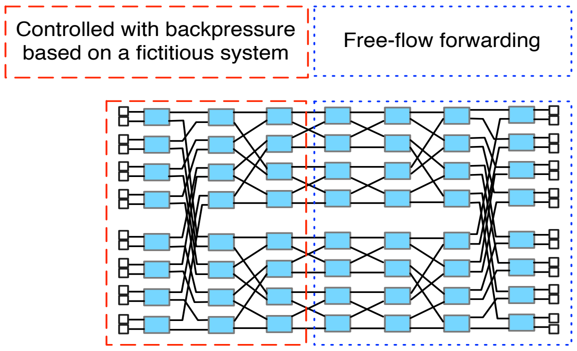


Fig. 3. Pictorial illustration of G-BP. The 1st half of the network is controlled by a backpressure-like algorithm based on a fictitious reference system. The 2nd half of the network uses a "free-flow" scheme for packet delivery.

B. *A fictitious reference system*

In order to guide the routing and scheduling of the grouped traffic, we create a fictitious reference system as follows.

1) Remove all the nodes in columns $n+1$ to $2n-1$.
2) Create two fictitious destination nodes D_1 and D_2 .
3) Connect each partition node, i.e., a node in $\mathcal{C}_n$, to D_1 with a link of capacity 1 packet/slot and to D_2 with a link of capacity 1 packet/slot.

An example of the fictitious system is shown in Fig. 4 for a 16×16 Benes network. The fictitious system will be used as a reference system to guide us on serving the grouped traffic. Specifically, we will design a backpressure-based algorithm for the fictitious system, and use the *exact* same actions to control the nodes in columns 1 to $n-1$ in the physical system. This approach has the useful property that it allows us to use only 4 queues per node.

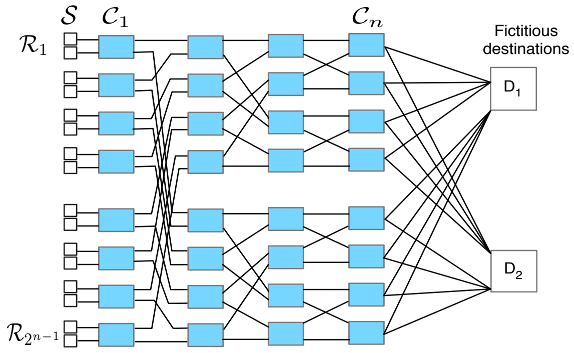


Fig. 4. The fictitious reference system for the 16×16 Benes network.

C. *Queue structure and load balancing*

Since in the reference system we only have 2 destinations and do not distinguish flows inside the network, if routing is not done carefully, it can happen that most of the traffic going to an output port is routed to a single partition node and causes instability of the node. In order to resolve this issue, we impose a special queueing structure on the switch nodes to balance all the traffic, so that each flow is equally split among all possible paths and routed to the partition nodes. Doing so, we guarantee that as long as the traffic rate is supportable (will be explained later), no node will be overwhelmed.

We now specify our queueing structure for both the fictitious system and the physical system:

1) Input servers in both systems: For each input server $s \in \mathcal{S}$, we maintain 2 queues per node as follows:

- $Q_s^{\mathsf{U}}(t)$: number of *upper division* flow packets stored at input server s;
- $Q_s^{\mathsf{L}}(t)$: number of *lower division* flow packets stored at input server s.

These two queues evolve according to the following dynamics:

$$Q_s^{\mathcal{T}}(t+1) = \left[Q_s^{\mathcal{T}}(t) - \mu_{s,m(s)}^{\mathcal{T}}(t)\right]^+ + R_s^{\mathcal{T}}(t). \qquad (5)$$

Here the notation $\mathcal{T} \in \Omega_s \triangleq \{\mathsf{U},\mathsf{L}\}$ denotes the "type" of the traffic at the input servers, $\mu_{s,m(s)}^{\mathcal{T}}(t)$ is the rate allocated to serve type $\mathcal{T}$ traffic at server s, and $R_s^{\mathcal{T}}(t)$ denotes the aggregate arrival to $Q_s^{\mathcal{T}}(t)$, i.e.,

$$R_s^{\mathsf{U}}(t) = \sum_{d \le 2^{n-1}} R_{sd}(t),\ R_s^{\mathsf{L}}(t) = \sum_{d > 2^{n-1}} R_{sd}(t). \qquad (6)$$

2) Switch modules in columns 1 *to* $n-1$ *in both systems:* We maintain 4 queues per node as follows:

- $Q_m^{\mathsf{UU}}(t)$: number of *upper division* flow packets that will be routed through m_u;
- $Q_m^{\mathsf{UL}}(t)$: number of *upper division* flow packets that will be routed through m_l;
- $Q_m^{\mathsf{LU}}(t)$: number of *lower division* flow packets that will be routed through m_u;
- $Q_m^{\mathsf{LL}}(t)$: number of *lower division* flow packets that will be routed through m_l.

Now define $\Omega_B \triangleq \{\mathsf{UU},\mathsf{UL},\mathsf{LU},\mathsf{LL}\}$ and use $\mathcal{T} \in \Omega_B$ to denote the type of these queues at the switch nodes. We see that the queues evolve according to the following dynamics:

$$\begin{aligned} &Q_m^{\mathcal{T}}(t+1) && (7)\\ &\quad \le \left[Q_m^{\mathcal{T}}(t) - \mu_{m,m(\mathcal{T})}^{\mathcal{T}}(t)\right]^+ + R_m^{\mathcal{T}}(t),\ \forall \mathcal{T} \in \Omega_B. \end{aligned}$$

Here $m(\mathcal{T})$ is the next hop node corresponding to the type $\mathcal{T}$ traffic, i.e., $m(\mathcal{T}) = m_u$ for $\mathcal{T} \in \{\mathsf{UU},\mathsf{LU}\}$ and $m(\mathcal{T}) = m_l$ otherwise. $\mu_{m,m(\mathcal{T})}^{\mathcal{T}}(t)$ is the rate allocated to serve type $\mathcal{T}$ packets at switch module m. $R_m^{\mathcal{T}}(t)$ is the input to $Q_m^{\mathcal{T}}(t)$, given by:

$$R_m^{\mathsf{UU}}(t) = X_m(t)R_m^{\mathsf{U}}(t),\ R_m^{\mathsf{UL}}(t) = (1 - X_m(t))R_m^{\mathsf{U}}(t), \quad (8)$$

$$R_m^{\mathsf{LU}}(t) = Y_m(t)R_m^{\mathsf{L}}(t),\ R_m^{\mathsf{LL}}(t) = (1 - Y_m(t))R_m^{\mathsf{L}}(t), \quad (9)$$

where $R_m^{\mathsf{U}}(t)$ and $R_m^{\mathsf{L}}(t)$ are the *aggregate* upper and lower division arrivals to node m, i.e.,

$$R_m^{\mathsf{U}}(t) = \sum_{m' \in \mathcal{M}_m^u} \mu_{m',m}^{\mathsf{UU}}(t) + \sum_{m' \in \mathcal{M}_m^l} \mu_{m',m}^{\mathsf{UL}}(t), \qquad (10)$$

$$R_m^{\mathsf{L}}(t) = \sum_{m' \in \mathcal{M}_m^u} \mu_{m',m}^{\mathsf{LU}}(t) + \sum_{m' \in \mathcal{M}_m^l} \mu_{m',m}^{\mathsf{LL}}(t). \qquad (11)$$

The variables $X_m(t)$ and $Y_m(t)$ are i.i.d. Bernoulli variables taking values 0 or 1 with equal probabilities, introduced for ensuring an equal division of the flow rates. Note that we have used inequality in (7). This is because the actual packet arrivals to $Q_m^{\mathcal{T}}(t)$ may be less than $R_m^{\mathcal{T}}(t)$ as the upstream nodes may not have enough packets to fulfill the allocated transmission rates. Our queueing structure and traffic splitting scheme are demonstrated in Fig. 5.

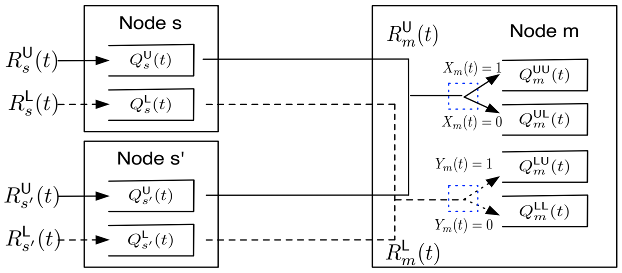


Fig. 5. The queueing structure and traffic splitting method.

3) Partition nodes in the fictitious system: Each node $m \in \mathcal{C}_n$ maintains only two queues $Q_m^{D_1}(t)$ and $Q_m^{D_2}(t)$ with the following dynamics:

$$Q_m^{D_i}(t+1) \leq \left[Q_m^{D_i}(t) - \mu_{m,D_i}(t)\right]^+ + R_m^{D_i}(t). \quad (12)$$

Here $R_m^{D_1}(t) = R_m^{\mathsf{U}}(t)$ and $R_m^{D_2}(t) = R_m^{\mathsf{L}}(t)$ are the aggregate arrivals defined in (10) and (11).

4) Nodes in columns n to $2n-1$ in the physical system: Each node $m \in \cup_{j=n}^{2n-1}\mathcal{C}_j$ maintains two First-In-First-Out (FIFO) queues $Q_m^a(t)$ and $Q_m^b(t)$, one for the upper output link a and the other for the lower output link b (see Fig. 2(a)). The arrivals are placed into the queues according to their destinations, i.e.,

$$Q_m^a(t+1) = \left[Q_m^a(t) - \mu_{m,m_u}(t)\right]^+ + \sum_{m' \in \mathcal{M}_m} \mu_{m',m}^a(t), \quad (13)$$

$$Q_m^b(t+1) = \left[Q_m^b(t) - \mu_{m,m_l}(t)\right]^+ + \sum_{m' \in \mathcal{M}_m} \mu_{m',m}^b(t). \quad (14)$$

Here $\mu_{m',m}^a(t) = \sum_s \sum_{d \in \mathcal{O}_m^a} \tilde{\mu}_{m',m}^{sd}(t)$, where $\tilde{\mu}_{m',m}^{sd}(t)$ denotes the *actual* number of flow (s,d) packets sent from m' to m at time t, and $\mu_{m',m}^b(t) = \sum_s \sum_{d \in \mathcal{O}_m^b} \tilde{\mu}_{m',m}^{sd}(t)$ denotes the number of packets that need to traverse the lower outgoing link b to their destinations.

Notice that in both systems, each partition node only maintains two queues and does not further split the traffic. This is because in the fictitious system, the next hop nodes of a partition node are D_1 and D_2, whereas in the physical system, the flow (s,d) packets at the partition nodes will be delivered to output server d following a unique path according to Lemma 3.

D. The arrival admission queue

Since the arrivals to the network are dynamic, in order to perform packet admission in a fair manner, we introduce an auxiliary variables $\gamma_{sd}(t)$ and create the following virtual *admission queue* for every flow (s,d):

$$H_{sd}(t+1) = \left[H_{sd}(t) - R_{sd}(t)\right]^+ + \gamma_{sd}(t). \quad (15)$$

Intuitively, $\gamma_{sd}(t)$ indicates how many flow (s,d) packets should have been admitted into the network. However, due to the randomness of the arrivals, this may not be feasible at every time t. Hence, the admission queue $H_{sd}(t)$ is created to ensure that in the long run, the admitted packets have a rate that is no smaller than the rate they should have got.

E. The output regulation queue

Here we specify the last component needed for our algorithm. Note that the above subsections have been dealing with reducing the number of queues per node and balancing the traffic inside the Benes network. In order to guarantee stability of the network, one also needs to ensure that the total traffic going to any output port of the Benes network does not exceed its capacity. To do so, we create the following *regulation queue* for each output port $d \in \{1, ..., 2^n\}$ (or equivalently, output server d):

$$q_d(t+1) = \left[q_d(t) - (1-\eta)\right]^+ + \sum_s R_{sd}(t). \quad (16)$$

That is, the input to this queue are all the admitted packets destined for output port d, and the service rate of the queue is $1-\eta$ for some small $\eta > 0$ for all time. The intuition here is that if these virtual queues are stable, then the average traffic rate for any output port is no more than $1-\eta$. The reason we have the small η "slack" is to ensure queue stability for the nodes in columns n to $2n-1$ in the physical network.

V. The Grouped-Backpressure algorithm (G-BP)

In this section, we present the construction of the G-BP algorithm and its performance.

A. Constructing G-BP

For notation purposes, we first define the aggregate network queue vector of the fictitious network as follows:

$$\boldsymbol{Z}(t) = \big(Q_s^{\mathcal{T}}(t),\, \forall\, s,\, \mathcal{T} \in \Omega_s,\, Q_m^{\mathcal{T}}(t), \forall\, m \in \cup_{j=1}^{n-1}\mathcal{C}_j,\, \mathcal{T} \in \Omega_B,$$
$$Q_m^{D_1}(t), Q_m^{D_2}(t),\, \forall\, m \in \mathcal{C}_n,\, H_{sd}(t),\, \forall\, (s,d),\, q_d(t), \forall\, d\big).$$

Then, we define the following Lyapunov function:

$$L(t) \triangleq \frac{1}{2}\sum_{s,\mathcal{T}\in\Omega_s}[Q_s^{\mathcal{T}}(t)]^2 + \frac{1}{2}\sum_{m\in\cup_{j=1}^{n-1}\mathcal{C}_j}\sum_{\mathcal{T}\in\Omega_B}[Q_m^{\mathcal{T}}(t)]^2 \quad (17)$$
$$+\frac{1}{2}\sum_{m\in\mathcal{C}_n}\sum_{i=1,2}[Q_m^{D_i}(t)]^2 + \frac{1}{2}\sum_{s,d}[H_{sd}(t)]^2 + \frac{1}{2}\sum_d [q_d(t)]^2.$$

Now define a Lyapunov drift as follows:

$$\Delta(t) \triangleq \mathbb{E}\Big[L(t+1) - L(t) \mid \boldsymbol{Z}(t)\Big]. \quad (18)$$

Using the facts that $0 \leq A_{sd}(t) \leq A_{\max}$ and that all the link capacities in the network are bounded, we obtain the following lemma for the drift. In the lemma, the parameter $V \geq 1$ is a control parameter offered by the algorithm to control the flow utility performance.

Lemma 1: Under any control policy, the following property holds for the drift at any time t:

$$\Delta(t) - V\mathbb{E}\Big[\sum_{s,d} U_{sd}(\gamma_{sd}(t)) \mid \boldsymbol{Z}(t)\Big] \quad (19)$$
$$\leq B - \sum_d q_d(t)(1-\eta) - \sum_{m\in\mathcal{C}_n,i} Q_m^{D_i}(t)\mathbb{E}\Big[\mu_{m,D_i}(t) \mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{s,d}\mathbb{E}\Big[VU_{sd}(\gamma_{sd}(t)) - H_{sd}(t)\gamma_{sd}(t) \mid \boldsymbol{Z}(t)\Big]$$

$$-\sum_{s}\sum_{d\leq 2^{n-1}}\mathbb{E}\Big[R_{sd}(t)\big[H_{sd}(t)-q_d(t)-Q^{\mathsf{U}}_s(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{s}\sum_{d>2^{n-1}}\mathbb{E}\Big[R_{sd}(t)\big[H_{sd}(t)-q_d(t)-Q^{\mathsf{L}}_s(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{s}\mathbb{E}\Big[\mu^{\mathsf{U}}_{s,m(s)}(t)\big[Q^{\mathsf{U}}_s(t)-\frac{1}{2}Q^{\mathsf{UU}}_{m(s)}(t)-\frac{1}{2}Q^{\mathsf{UL}}_{m(s)}(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{s}\mathbb{E}\Big[\mu^{\mathsf{L}}_{s,m(s)}(t)\big[Q^{\mathsf{L}}_s(t)-\frac{1}{2}Q^{\mathsf{LU}}_{m(s)}(t)-\frac{1}{2}Q^{\mathsf{LL}}_{m(s)}(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{m\in\mathcal{C}_{n-1}}\sum_{\mathcal{T}\in\{\mathsf{UU},\mathsf{UL}\}}\mathbb{E}\Big[\mu^{\mathcal{T}}_{m,m(\mathcal{T})}(t)\big[Q^{\mathcal{T}}_m(t)-Q^{D_1}_{m(\mathcal{T})}(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{m\in\mathcal{C}_{n-1}}\sum_{\mathcal{T}\in\{\mathsf{LU},\mathsf{LL}\}}\mathbb{E}\Big[\mu^{\mathcal{T}}_{m,m(\mathcal{T})}(t)\big[Q^{\mathcal{T}}_m(t)-Q^{D_2}_{m(\mathcal{T})}(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{m\in\cup_{j=1}^{n-2}\mathcal{C}_j}\sum_{\mathcal{T}\in\{\mathsf{UU},\mathsf{UL}\}}\mathbb{E}\Big[\mu^{\mathcal{T}}_{m,m(\mathcal{T})}(t)\big[Q^{\mathcal{T}}_m(t)-\frac{1}{2}Q^{\mathsf{UU}}_{m(\mathcal{T})}(t)-\frac{1}{2}Q^{\mathsf{UL}}_{m(\mathcal{T})}(t)\big]\mid \boldsymbol{Z}(t)\Big]$$
$$-\sum_{m\in\cup_{j=1}^{n-2}\mathcal{C}_j}\sum_{\mathcal{T}\in\{\mathsf{LU},\mathsf{LL}\}}\mathbb{E}\Big[\mu^{\mathcal{T}}_{m,m(\mathcal{T})}(t)\big[Q^{\mathcal{T}}_m(t)-\frac{1}{2}Q^{\mathsf{LU}}_{m(\mathcal{T})}(t)-\frac{1}{2}Q^{\mathsf{LL}}_{m(\mathcal{T})}(t)\big]\mid \boldsymbol{Z}(t)\Big].$$

Here $B=\Theta(1)$ is a constant given by:

$$B=\frac{1}{2}[2^n(10n-2)+A^2_{\max}(2^{3n-1}+2^{2n+1}+2^{3n})],\quad (20)$$

and the expectation is taken over the random arrivals as well as the potential randomness in the actions. $\Diamond$

Proof: Due to space consideration, we omitted the proof. Please see [17]. ■

Note that since the Benes network size is $\Theta(2^n)$, the n value is only logarithmic in the network size. Hence, B is indeed only polynomial in the network size. Also note that the above lemma is for the fictitious system. Based on the above lemma, we now describe our algorithm for the *physical* system. In the algorithm, we will operate the nodes in $\mathcal{S}$ and $\cup_{j=1}^{n-1}\mathcal{C}_j$ in the physical system exactly as we operate them in the fictitious system. For these nodes, the actions will be chosen in every time slot to minimize the right-hand-side (RHS) of the drift expression (19). For all the modules in columns n to $2n-1$, we simply do a free-flow routing.

Grouped-Backpressure (G-BP) At every time slot t, observe $\boldsymbol{A}(t)$ and $\boldsymbol{Z}(t)$, and perform the following:

- Auxiliary Variable Selection: For every (s,d) flow, choose $\gamma_{sd}(t)$ to solve:
$$\max:\quad VU_{sd}(\gamma_{sd}(t))-H_{sd}(t)\gamma_{sd}(t)\quad (21)$$
$$\text{s.t.}\quad 0\leq\gamma_{sd}(t)\leq A_{\max}.$$
- Admission Control: For every input server s: If $d\leq 2^{n-1}$, i.e., an upper division flow, choose $R_{sd}(t)=A_{sd}(t)$ if $H_{sd}(t)-q_d(t)-Q^{\mathsf{U}}_s(t)>0$; else choose $R_{sd}(t)=0$. If $d>2^{n-1}$, i.e., a lower division flow, choose $R_{sd}(t)=A_{sd}(t)$ if $H_{sd}(t)-q_d(t)-Q^{\mathsf{L}}_s(t)>0$; else choose $R_{sd}(t)=0$.
- Routing and Scheduling:
  - For any node $m\in\cup_{j=1}^{n-1}\mathcal{C}_j\cup\mathcal{S}$: define the following weights for the outgoing link $[m,m_u]$:
$$W^{\mathsf{U}}_{m,m_u}(t)\triangleq\max\Big[Q^{\mathsf{UU}}_m(t)-\tilde{Q}^{\mathsf{U}}_{m_u}(t),0\Big],\quad (22)$$
$$W^{\mathsf{L}}_{m,m_u}(t)\triangleq\max\Big[Q^{\mathsf{LU}}_m(t)-\tilde{Q}^{\mathsf{L}}_{m_u}(t),0\Big],\quad (23)$$
where $\tilde{Q}^{\mathsf{U}}_{m_u}(t)$ and $\tilde{Q}^{\mathsf{L}}_{m_u}(t)$ are defined as:
$$\tilde{Q}^{\mathsf{U}}_{m_u}(t)=\begin{cases}\frac{1}{2}Q^{\mathsf{UU}}_{m_u}(t)+\frac{1}{2}Q^{\mathsf{UL}}_{m_u}(t) & j_m\leq n-2,\\ Q^{D_1}_{m_u}(t) & j_m=n-1,\end{cases}\quad (24)$$
$$\tilde{Q}^{\mathsf{L}}_{m_u}(t)=\begin{cases}\frac{1}{2}Q^{\mathsf{LU}}_{m_u}(t)+\frac{1}{2}Q^{\mathsf{LL}}_{m_u}(t) & j_m\leq n-2,\\ Q^{D_2}_{m_u}(t) & j_m=n-1.\end{cases}\quad (25)$$
Then, we choose the service rates $\mu^{\mathsf{UU}}_{m,m_u}(t)$ and $\mu^{\mathsf{LU}}_{m,m_u}(t)$ for link $[m,m_u]$ to solve:
$$\max:\quad \mu^{\mathsf{UU}}_{m,m_u}(t)W^{\mathsf{U}}_{m,m_u}+\mu^{\mathsf{LU}}_{m,m_u}(t)W^{\mathsf{L}}_{m,m_u}\quad (26)$$
$$\text{s.t.}\quad \mu^{\mathsf{UU}}_{m,m_u}+\mu^{\mathsf{LU}}_{m,m_u}\leq 1,\ \mu^{\mathsf{UU}}_{m,m_u},\mu^{\mathsf{LU}}_{m,m_u}\in\{0,1\}.$$
To solve for $\mu^{\mathsf{UL}}_{m,m_l}(t)$ and $\mu^{\mathsf{LL}}_{m,m_l}(t)$, we replace $Q^{\mathsf{UU}}_m(t)$ and $Q^{\mathsf{LU}}_m(t)$ with $Q^{\mathsf{UL}}_m(t)$ and $Q^{\mathsf{LL}}_m(t)$ in (22) and (23). Also, we replace m_u and D_1 with m_l and D_2 in (24) and (25). If $m=s\in\mathcal{S}$, we simply replace $Q^{\mathsf{UU}}_m(t)$ and $Q^{\mathsf{UL}}_m(t)$ with $Q^{\mathsf{U}}_s(t)$ and $Q^{\mathsf{L}}_s(t)$ in (22) and (23), and replace m_u by $m(s)$ in (24) and (25).
  - For every node $m\in\cup_{j=n}^{2n-1}\mathcal{C}_j$: Each module serves each FIFO queue for each outgoing link according to (13) and (14) with $\mu_{m,m_u}(t)=\mu_{m,m_l}(t)=1$ for all time.
- Queue Updates: In the fictitious system, choose the service rates $\mu_{m,D_1}(t)$ and $\mu_{m,D_2}(t)$ to solve:
$$\max:\quad Q^{D_1}_m(t)\mu_{m,D_1}(t)+Q^{D_2}_m(t)\mu_{m,D_2}(t)\quad (27)$$
$$\text{s.t.}\quad \mu_{m,D_1}(t),\mu_{m,D_2}(t)\in\{0,1\}.$$
Then, update all the queues in both the fictitious system and the physical system according to their dynamics. $\Diamond$

We note that G-BP only controls the first half of the physical system with the backpressure actions. All the nodes in columns n to $2n-1$ simply serve the flows with a "free-flow" manner, i.e., always serve the flows at the maximum rate. This is different from the usual backpressure algorithms that control all the queues in the network to ensure stability.

B. Performance analysis

In this section, we prove that G-BP achieves a near-optimal performance. To carry out our analysis, we first have the following theorem, which characterizes the capacity region of a Benes network. In the theorem, we use $\boldsymbol{r}=(r_{sd},\forall(s,d))$ to denote the vector of arrival rates, where r_{sd} represents the average rate of the (s,d) flow.

Theorem 1: [9] [10] The capacity region of the Benes network $\mathbb{B}_n$ is given by:

$$\Lambda_n=\{\boldsymbol{r}\mid\sum_{s=1}^{2^n}r_{sd}\leq 1,\ \sum_{d=1}^{2^n}r_{sd}\leq 1,\ r_{sd}\geq 0,\forall s,d\}.\quad\square$$

We now show that under the special structure of the Benes network, our queueing structure and traffic splitting scheme generate a balanced routing across the network. This is summarized in the following lemma, where we use $\overline{\mu}^{\mathcal{T}}_{m,m(\mathcal{T})}$ to denote the time average transmission rates of the type $\mathcal{T}$ traffic from m to $m(\mathcal{T})$. Specifically,

$$\overline{\mu}^{\mathcal{T}}_{m,m(\mathcal{T})} \triangleq \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\tilde{\mu}^{\mathcal{T}}_{m,m(\mathcal{T})}(t)\big].$$

Here $\tilde{\mu}^{\mathcal{T}}_{m,m(\mathcal{T})}(t)$ denotes the *actual* number of type $\mathcal{T}$ packets sent over the link $[m, m(\mathcal{T})]$ at time t. Similarly, we use $\overline{\mu}^{sd}_{m}$ to denote the average rate of the flow (s, d) packets going through a node m, i.e.,

$$\overline{\mu}^{sd}_{m} \triangleq \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\big[\tilde{\mu}^{sd}_{m,m_u}(t)+\tilde{\mu}^{sd}_{m,m_l}(t)\big],$$

where $\tilde{\mu}^{sd}_{m,m_u}(t)$ and $\tilde{\mu}^{sd}_{m,m_l}(t)$ denote the actual numbers of flow (s, d) packets sent from node m to node m_u and node m_l at time t, respectively.

***Lemma** 2:* If the fictitious network is stable, then,

(a) For every node $m \in \cup_{j=1}^{n-1}\mathcal{C}_j$,
$$\overline{\mu}^{\mathsf{UU}}_{m,m_u} = \overline{\mu}^{\mathsf{UL}}_{m,m_l}, \overline{\mu}^{\mathsf{LU}}_{m,m_u} = \overline{\mu}^{\mathsf{LL}}_{m,m_l}. \tag{28}$$

(b) The average rate of any (s, d) flow packets going through any partition node $m \in \mathcal{C}_n$ satisfies $\overline{\mu}^{sd}_{m} = r_{sd}/2^{n-1}$. $\Diamond$

Proof: See Appendix A. ■

We now present the performance results of G-BP. Recall that β is defined in (2) to be the maximum first derivative among all utility functions, and that $\boldsymbol{r}^{\mathsf{opt}} \in \Lambda_n$ denotes the optimal solution to our utility maximization problem.

***Theorem** 2:* (i) Both the fictitious network and the physical network are stable under G-BP. (ii) Denote $\boldsymbol{r}^{\mathsf{G\text{-}BP}}$ the time average rate vector achieved by G-BP. We have:

$$U(\boldsymbol{r}^{\mathsf{G\text{-}BP}}) \geq U(\boldsymbol{r}^{\mathsf{opt}}) - \frac{B}{V} - 2^n\beta\eta. \quad \square \tag{29}$$

Proof: A sketch of our proof is given in Appendix B. For full details, please see [17]. ■

From (29), we see that the utility performance of G-BP can arbitrarily approach the optimal as we increase V and decrease η. However, doing so will increase the average network delay. Hence, there is a natural tradeoff between the utility performance and the network delay.

Note that though the performance results in Theorem 2 look similar to previous results in [14], the proof is indeed quite different. This is because in our case, we impose a special queueing structure on the network, and the second half of the network uses a free-flow routing. These two features make the analysis very different from the usual backpressure algorithms.

C. Discussion on implementation

We note that the G-BP algorithm can easily be implemented in a fully distributed manner. Specifically, one can maintain the virtual admission queues at the input servers and maintain the virtual output regulation queues at the output servers using counters, as shown in Fig. 6. With this arrangement, the auxiliary variable selection step can easily be done locally at the input servers, and the routing and scheduling step can easily be done by each node exchanging queue information only with its four neighbors. The admission control step requires the input servers to know the regulation queue sizes. This can be achieved by message passing the regulating queue sizes along the network using prioritized packets. Similarly, the update of the regulation queues requires the knowledge of the arrivals for the output ports. This can be approximated by using the local arrivals to the output servers as the input to the regulation queues. Though message passing and queue approximation may incur performance loss in practice, we will see in the simulation section that, the G-BP algorithm is indeed very robust and can still achieve near-optimal performance even under different message passing delays and the regulation queue approximation.

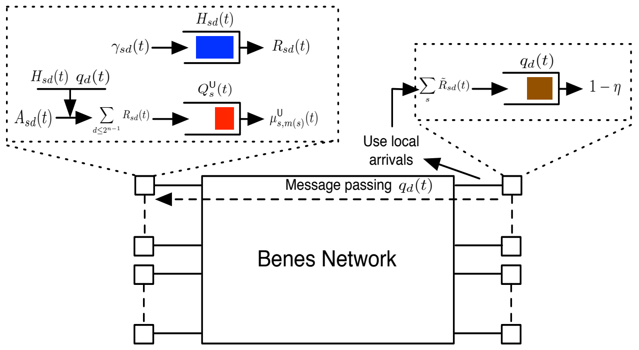


Fig. 6. Implementation of G-BP. The virtual admission queues are maintained at the input servers, while the virtual regulation queues are maintained at the output servers. Message passing is used to send regulation queue information through the network for admission control. The regulation queues can use the local arrivals to the output servers as the input.

VI. Simulation

In this section, we present the simulation results of G-BP on a $2^4 \times 2^4$ size Benes network. For simplicity, we assume that $A_{sd}(t) = A_{\max} = 2$ for all time.

In the simulation, we assume that every flow has a utility function $\log(1 + r_{sd})$. In every time slot, each flow can admit 0, 1 or 2 packets. We simulate the system for $V \in \{5, 10, 20, 50, 100\}$ and $\eta = 0.01$. Each simulation is run for 10^5 slots. To test the robustness of G-BP against the delay and sparsity in message passing and the regulation queue approximation, we simulate four different cases. (i) The original G-BP algorithm, where the message passing delay is zero and the regulation queue is exact. (ii) The case when the input to the regulation queue q_d are the actual packet arrivals to the output server d (the service rate is still $1-\eta$), and admission control at time t uses $q_d(t-(2n-1))$ instead of $q_d(t)$. (iii) Similar to the second case, but admission control at time t uses $q_d(t-5(2n-1))$. (iv) Similar to the second case, but the regulation queue information is only sent every $5(2n-1)$ slots and has a delay of $5(2n-1)$. That is, admission control at time t uses $q_d(t_0)$ where $t_0 = \max[(\lfloor\frac{t}{5(2n-1)}\rfloor - 1)5(2n-1), 1]$.

Fig. 7 shows the performance of the G-BP algorithm. Here the average delay (in number of slots) is computed using the set of packets that are delivered when the simulation ends. For all simulations, this set contains more than 99.9% of the total packets that enter the network. We see that as we increase

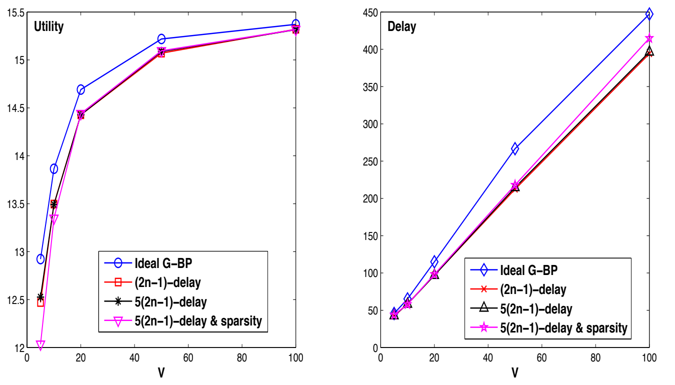


Fig. 7. The aggregate flow utility and average packet delay under G-BP. One can see that G-BP works very well even with message passing delay and regulation queue approximation.

the V value, the aggregate flow utility quickly converges to its optimal value. However, doing so also leads to a linear increase of the average packet delay. We also see from the figure that, G-BP is indeed very robust to the delay and sparsity in message passing, and the regulation queue approximation.

In Fig. 8, we plot a recorded queue process of the network under G-BP for $V = 10$. In this case, we change each flow's utility function to $w_{sd}\log(1+r_{sd})$ in the middle of the simulation, where w_{sd} takes values $1, 2$ or 3 equally likely. We see that after the change, G-BP quickly adapts to the new utility functions and performs admission and routing accordingly.

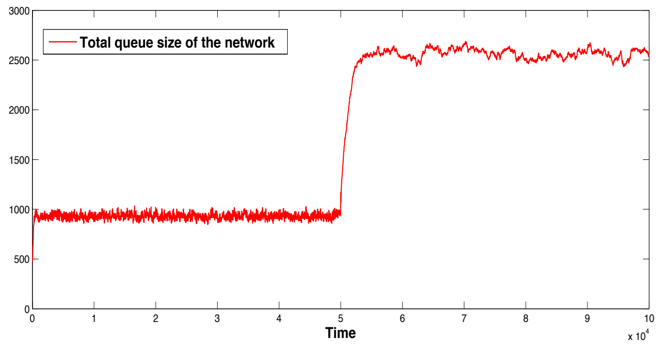


Fig. 8. The total network queue size under G-BP with $V = 10$.

Finally, we also evaluate the average packet delay as a function of the network size, to test the scalability of the algorithm. As comparison, we also simulate an "enhanced" G-BP algorithm, in which we replace all the queue values in the algorithm with the queue value plus the node's hop count to the destination, i.e., its column number. The idea is to create "bias" towards the packet destinations. This enhancement is similar to the EDRPC algorithm developed in [18]. We can see from Fig. 9 that the average packet delay under G-BP scales as $\Theta(n^2)$. Since the Benes network size is $\Theta(2^n)$, this implies that the *average delay grows only logarithmically in the network size*.

Note that in Fig. 9 we have plotted the average delay in number of slots. To get some physical understanding of the results, assume that each packet has 500 bytes and each link has a capacity of 1 Gbit/second, which are both quite common in practice. Then, every slot is 4 microseconds. We see that

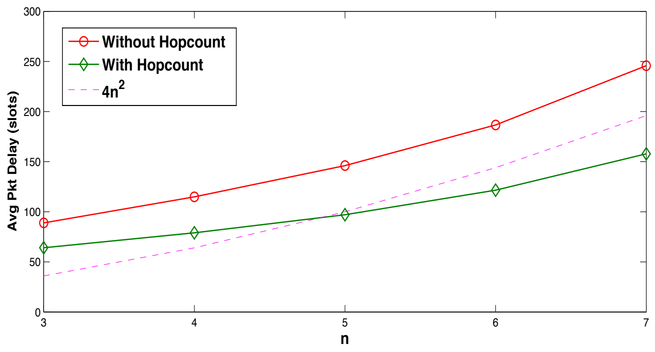


Fig. 9. Average delay as a function of the network size under $V = 10$.

the average packet delay under Benes network with G-BP is roughly 1 millisecond when the network size is 128×128. This demonstrates the good delay performance of our network design approach.

VII. Conclusion

In this work, we develop a novel networking solution called *Benes packet network*, which consists of a Benes network built with simple commodity switches, a flow utility maximization mechanism, and a Grouped-Backpressure (G-BP) routing and scheduling algorithm. We show that this combination can achieve a near-optimal flow utility and ensure small end-to-end delay for the traffic flows. Our approach also only requires each switch module to maintain at most four queues regardless of the network size, and can easily be implemented in practice in a fully distributed manner.

Acknowledgement

This work is supported in part by MURI grant BAA 07-036.18. It was completed when Longbo Huang was a post-doctoral researcher at University of California, Berkeley.

Appendix A – Proof of Lemma 2

For our analysis, we will first present an important fact and a lemma regarding the structure of the Benes network. First, from the construction rules of the Benes networks, we first have the following observation:

Fact 1: For a $2^n \times 2^n$ Benes network, its partition nodes coincide with the partition nodes of its two $2^{n-1} \times 2^{n-1}$ subnetworks. $\Diamond$

We then have the following simple lemma, which can be seen from the construction rules of Benes networks.

Lemma 3: (a) Starting from any partition node $m \in \mathcal{C}_n$, there is a unique path to any output server $d \in \mathcal{D}$. (b) For every node $m \in \mathcal{C}_{n+l}, l \geq 0$, we have:

$$\mathcal{O}^a_m = \{\kappa_m 2^{n-l}+1, ..., (\kappa_m + \frac{1}{2})2^{n-l}\}, \tag{30}$$

$$\mathcal{O}^b_m = \{(\kappa_m + \frac{1}{2})2^{n-l}+1, ..., (\kappa_m + 1)2^{n-l}\}, \tag{31}$$

where $\kappa_m \triangleq (i_m - 1) \mod 2^l$. $\Box$

Now we prove Lemma 2.

Proof: (Lemma 2) We first prove Part (a). From the queueing dynamic equation (7), we see that for any node $m \in \cup_{j=1}^{n-1}\mathcal{C}_j$, the input rates into $Q^{\mathsf{UU}}_m(t)$ and $Q^{\mathsf{UL}}_m(t)$ are equal because of random splitting. Similarly, the input rates into $Q^{\mathsf{LU}}_m(t)$ and $Q^{\mathsf{LL}}_m(t)$ are the same. Hence, if the fictitious network is stable, the output rates from these queues are equal to their input rates [13]. Therefore (28) holds.

Now we prove Part (b) by induction. First it holds for any 4×4 Benes network. This is because if the fictitious network is stable, then the input switch modules split the incoming flows equally into the two partition nodes (see Fig. 2).

Now suppose the same is true for a $2^{n-1} \times 2^{n-1}$ Benes network, we want to show that it also holds for a $2^n \times 2^n$ Benes network.

To see this, note from Fig. 2 that each $2^n \times 2^n$ Benes network consists of two $2^{n-1} \times 2^{n-1}$ subnetworks, 2^{n-1} input switch modules and 2^{n-1} output switch modules. According to the structure of the Benes network, any input switch module has one link connecting to the upper $2^{n-1} \times 2^{n-1}$ subnetwork and the other one connecting to the lower $2^{n-1} \times 2^{n-1}$ subnetwork. From Part (a), we see that half of a flow's rate will be routed through the upper subnetwork and the other half will be routed through the lower subnetwork. Now consider the upper subnetwork and view the flow traffic into this subnetwork as its own external input. Since this subnetwork is also stable, the flow's traffic will be equally split and routed via its partition nodes by induction. Since all the partition nodes coincide according to Fact 1, we see that the lemma follows. ■

Appendix B – Proof sketch of Theorem 2

Here we provide a proof sketch for Theorem 2. The full details can be found in [17]. Our proof consists of two parts.

(**Part A - Stability**) We start by showing that all the queues at the input servers are stable. This is done by noticing that, since $U'_{sd}(\cdot) < \beta$, whenever $H_{sd}(t) > V\beta$, $\mathsf{G\text{-}BP}$ will set $\gamma_{sd}(t) = 0$. Hence, using the fact that $0 \leq \gamma_{sd}(t) \leq A_{\max}$ for all time, we have:

$$0 \leq H_{sd}(t) \leq V\beta + A_{\max}, \qquad (32)$$

for all (s, d) flows and for all time. Then, we see from the admission control rules that, for $d \leq 2^{n-1}$, only when $q_d(t) + Q^{\mathsf{U}}_s(t) < H_{sd}(t)$ will $q_d(t)$ and $Q^{\mathsf{U}}_s(t)$ get new arrivals (similar for $Q^{\mathsf{L}}_s(t)$). This shows that $q_d(t)$ and $Q^{\mathsf{U}}_s(t)$ will also be deterministically upper bounded. Proceeding in this manner and using the routing and scheduling rules of $\mathsf{G\text{-}BP}$, we can show that all the queues in the fictitious network are bounded, and hence stable.

To prove that the second half of the physical network is stable, we use the fact that all the regulation queues are stable. This implies that the total traffic rate going to any output server is no more than $1 - \eta$. Then, using Lemmas 2 and 3, and the structure of the Benes network, we show that each queue in $\cup_{j=n}^{2n-1}\mathcal{C}_j$ of the physical network has an input rate of no more than $1 - \eta$, while the service rate is always 1. Now using the fact that at any time, the number of packets entering a queue is finite, we show that all the queues in $\cup_{j=n}^{2n-1}\mathcal{C}_j$ are stable.

(**Part B - Utility**) To prove the utility performance, we first construct a sequence of admission control vectors $\{\boldsymbol{R}^{(\boldsymbol{A},k)} = (R^{\boldsymbol{A},k}_{sd}, \forall\, s, d)\}_{k=1}^{\infty}$ with probabilities $\{p^{(\boldsymbol{A},k)}\}_{k=1}^{\infty}$ for every arrival rate vector $\boldsymbol{A}$, and a set of auxiliary variables $\{\gamma_{sd}, \forall\, s, d\}$ that satisfy: (i) $\gamma_{sd} = r_{sd} \triangleq \mathbb{E}\big[\sum_k p^{(\boldsymbol{A})}_k R^{(\boldsymbol{A},k)}_{sd}\big]$, (ii) $\boldsymbol{r} = (r_{sd}, \forall\, s, d) \in \Lambda_n$, and (iii) $\sum_{sd} U_{sd}(\gamma_{sd}) \geq U(\boldsymbol{r}^{\mathsf{opt}}) - \beta\eta 2^n$. Then, we show that since $\boldsymbol{r} \in \Lambda_n$, there exists a stationary and randomized routing and scheduling policy Π that can support the rate $\boldsymbol{r}$ over the network in a *fully symmetric* manner, i.e., each flow is equally split by every node when routed to the next hop nodes. Having established the above two steps, we then plug the above admission control vectors, the auxiliary variables, and the policy Π into the RHS of the drift expression (19) to obtain:

$$\Delta(t) - V\mathbb{E}\bigg[\sum_{s,d} U_{sd}(\gamma^{\mathsf{G\text{-}BP}}_{sd}(t)) \mid \boldsymbol{Z}(t)\bigg] \leq B - VU(\boldsymbol{r}^{\mathsf{opt}}) + V\eta\beta 2^n. \qquad (33)$$

We then use (33) and a telescoping sum argument to show that $\sum_{s,d} U_{sd}(\overline{\gamma}^{\mathsf{G\text{-}BP}}_{sd}) \geq U(\boldsymbol{r}^{\mathsf{opt}}) - B/V - \beta\eta 2^n$, where $\overline{\gamma}^{\mathsf{G\text{-}BP}}_{sd}$ is the average of $\gamma^{\mathsf{G\text{-}BP}}_{sd}(t)$ under $\mathsf{G\text{-}BP}$. Finally, using the fact that all the admission queues are stable, we have $r^{\mathsf{G\text{-}BP}}_{sd} \geq \overline{\gamma}^{\mathsf{G\text{-}BP}}_{sd}$ for all (s, d), which then implies $\sum_{s,d} U_{sd}(r^{\mathsf{G\text{-}BP}}_{sd}) \geq U(\boldsymbol{r}^{\mathsf{opt}}) - B/V - \beta\eta 2^n$ and proves the theorem.

References

[1] M. Katevenis N. Chrysos. Scheduling in non-blocking buffered three-stage switching fabrics. *Proceedings of IEEE Infocom, Barcelona, Spain*, April 2006.

[2] A. Singla, C. Hong, L. Popa, and P. B. Godfrey. Jellyfish: Networking data centers randomly. *Proceedings of NSDI*, 2012.

[3] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz nd P. Patel, and S. Sengupta. Vl2: A scalable and flexible data center network. *Proceedings of SIGCOMM*, 2009.

[4] C. Guo, H. Wu, K.Tan, L. Shi, Y. Zhang, and S. Lu. Dcell: A scalable and fault-tolerant network structure for data centers. *Proceedings of SIGCOMM*, 2008.

[5] L. Gyarmati and T. Anh Trinh. Scafida: A scale-free network inspired data center architecture. *Proceedings of SIGCOMM*, 2010.

[6] M. Al-Fares, A. Loukissas, and A. Vahdat. A scalable commodity data center network architecture. *Proceedings of SIGCOMM*, 2008.

[7] P. Costa, A. Donnelly, A. Rowstron, and G. OShea. Camdoop: Exploiting in-network aggregation for big data applications. *Proceedings of NSDI*, 2012.

[8] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan. c-through: Part-time optics in data centers. *Proceedings of SIGCOMM*, 2010.

[9] V. E. Beneš. Permutation groups, complexes and rearrangeable connecting network. *Bell System Technical Journal, 43, 4:16191640*, 1964.

[10] V. E. Beneš. Mathematical theory of connecting networks and telephone traffic. *Academic Press Inc., New Yrok*, 1965.

[11] F. Kelly. Charging and rate control for elastic traffic. *European Transactions on Telecommunications*, vol. 8, pp. 33-37, 1997.

[12] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, Vol. 95, No. 1, January 2007.

[13] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks*. Foundations and Trends in Networking Vol. 1, no. 1, pp. 1-144, 2006.

[14] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Trans. on Networking, vol. 16, no. 2, pp. 396-409*, April 2008.

[15] L. Ying, R. Srikant, and D. Towsley. Cluster-based back-pressure routing algorithm. *Proc. of IEEE INFOCOM*, April 2008.

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

[17] L. Huang and J. Walrand. A Benes packet network. *ArXiv Technical Report*, 2012.

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