SEMI-ASYNCHRONOUS ROUTING FOR LARGE SCALE HIERARCHICAL NETWORKS

Wei-Cheng Liao[*], *Mingyi Hong*[♮], *Hamid Farmanbar*[†], *and Zhi-Quan Luo*[*]

[*] Dept. of Elect. & Compt. Eng. Univ. of Minnesota Minneapolis, 55454, USA

[♮] Dept. of IMSE, Iowa State Univ., Ames, IA, 50011

[†]Huawei Canada Research Centre Ottawa, ON K2K 3J1, Canada.

ABSTRACT

We consider the distributed network routing problem in a large-scale hierarchical network whereby the nodes are partitioned into subnetworks, each managed by a network controller (NC), and there is a central NC to coordinate the operation of the distributed NCs. We propose a semi-asynchronous routing algorithm for such a network, whereby the computation is distributed across the NCs and is parallel within each NC. A key feature of the algorithm is its ability to handle a certain degree of asynchronism: the distributed NCs can perform their local computation asynchronously at different processing speed. The efficiency of the proposed algorithm is validated through numerical experiments.

***Index Terms*—** Traffic engineering, asynchronous network routing, alternating direction method of multiplier (ADMM)

1. INTRODUCTION

The explosive growth of data traffic has presented many challenges to the provision of communication networks. For example, consider a content provider managing a number of geographically distributed data centers, each of which further consists of thousands of networked computing/storage nodes. For such hierarchical networks, efficient distributed management of network resource for real time content delivery is of critical importance [1]. In this paper, we consider the distributed network routing problem in a large hierarchical network.

Traditionally, the network management problem has been formulated as the traffic engineering (TE) problem and has been extensively studied [2]. One popular approach that allows distributed implementation is the so-called dual decomposition algorithm [1, 3]. However, this approach is not suitable for large-scale networks due to its slow convergence, especially when the design objective function is not strongly convex [4]. Recently, a different design framework based on the well-known Alternating Direction Method of Multipliers (ADMM) [2, 5] has been proposed to solve the network TE problem [6, 7, 8]. Numerically, it has been shown that the ADMM based algorithms can achieve better performance than the dual decomposition approach.

Most of the existing distributed network management algorithms consider each network node as a basic computing agent capable of coordinating with the other nodes. However, for large networks, such decomposition to the node level can lead to substantial communication overhead. To limit the amount of control traffic, contemporary networks such as the software defined network (SDN) advocate a hierarchical architecture [9, 10] whereby a number of network controllers (NCs) are deployed in different geographical locations, each controlling a set of network nodes such as hard drives within cloud center. Further, there is a central NC globally coordinating the behavior of the distributed NCs, see Fig. 1 for illustration. Such "hierarchical" network structure distributes the control and communication to a number of NCs, resulting in faster network provision with reduced communication overhead [1].

This work is supported by a research gift from Huawei Technologies, Inc.

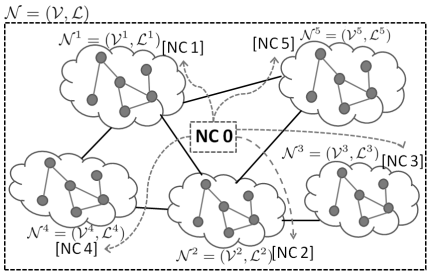
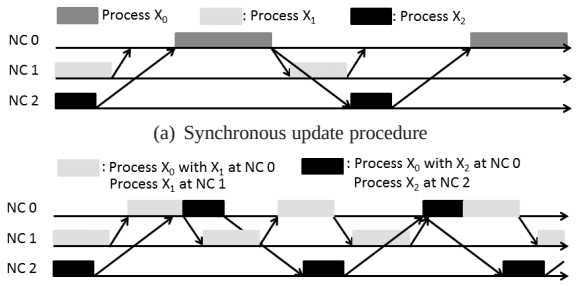


Fig. 1. A wireline network consists of 5 subnetworks. Each of them is controlled by a NC, and these NCs are coordinated globally by a central NC 0.

A major drawback of the existing distributed network provisioning algorithms is the synchronization required at the network operator level. For example, the central NC, say NC 0, cannot perform any update until it receives the latest information from *all* other NCs, see Fig. 2 (a). The efficiency of the entire TE thus depends on that of the slowest NC. The extension for dual decomposition approach with the (partially) asynchronous rule described in [11] requires the design objective to satisfy certain conditions, e.g., strongly convex [12, 13]. Also, the asynchronous ADMM algorithm for network optimization has recently been proposed [14], but no communication delay is considered. A key contribution of this paper is to propose a semi-asynchronous distributed routing strategy that overcomes the above difficulty. In the proposed scheme, the central NC performs the update periodically, each time based on the latest information gathered from a NC; see Fig. 2 (b). The reason that the proposed scheme is called "semi-asynchronous" is that we still need the central NC 0 to use the most recent information from each local NC. No outdated information from the distributed NCs is used at NC 0. As such, our scheme should be distinguished from the so-called *total* asynchronous rule described in [11] which allows use of outdated information.

2. SYSTEM MODEL

Consider a large-scale connected wireline network $\mathcal{N} = (\mathcal{V}, \mathcal{L})$ which is controlled by $K + 1$ NCs as illustrated in Fig. 1. Let $\mathcal{V}$ denote the set of network nodes. It is partitioned into K subsets, i.e., $\mathcal{V} = \cup_{i=1}^{K} \mathcal{V}^i$, $\mathcal{V}^i \cap \mathcal{V}^j = \emptyset$, $\forall\, i \neq j$. The set of directed links that connect nodes of $\mathcal{V}$ is denoted as $\mathcal{L} \triangleq \{l = (s_l, d_l) \mid \forall\, s_l, d_l \in \mathcal{V}\}$. Here $l = (s_l, d_l)$ denotes the directed link from node s_l to node d_l. The NC i, $i = 1 \sim K$, controls $\mathcal{V}^i$ and the links connecting these nodes, i.e., $\mathcal{L}^i \triangleq \{l = (s_l, d_l) \in \mathcal{L} \mid \forall\, s_l, d_l \in \mathcal{V}^i\}$. This



(a) Synchronous update procedure

(b) Semi-asynchronous update procedure

Fig. 2. The illustration of different updating schemes for distributed network systems. Here we denote the local variables for NC i, $i = 0 \sim 2$, as $\mathbf{x}_i$.

network $\mathcal{N}^i = (\mathcal{V}^i, \mathcal{L}^i)$ is called the subnetwork i. Also define $\mathcal{L}^0_{ij} \triangleq \{l = (s_l, d_l) \in \mathcal{L} \mid \forall\ s_l \in \mathcal{V}^i, d_l \in \mathcal{V}^j\}$ as the set of links connecting two neighboring subnetworks i and j.

We assume that there are a total of M data flows to be transported over the network. Each data flow m is demanded by the destination node $d_m \in \mathcal{V}$ from the source node $s_m \in \mathcal{V}$. We use $r_m \geq 0$ to denote flow m's rate, and use $f_{l,m} \geq 0$ to denote its rate on link $l \in \mathcal{L}$. We also assume a master node exists which controls the data flow rates $\{r_m\}_{m=1}^M$. The central NC 0 controls the subnetwork $\mathcal{N}^0$, consisting of the master node and the links connecting different subnetworks, i.e., $\mathcal{L}^0 = \cup_{i \neq j} \mathcal{L}^0_{ij}$.

Given these notations, there are two types of network constraints. The first set of constraints is related to the link capacity. Assume each link $l \in \mathcal{L}$ has a fixed capacity, C_l. The total flow rate on link l is constrained by

$$\mathbf{1}^T \mathbf{f}_l \leq C_l,\ \forall\ l \in \mathcal{L}, \tag{1}$$

where $\mathbf{1}$ is the all-one vector and $\mathbf{f}_l \triangleq [f_{l,1}, \ldots, f_{l,M}]^T$. The second set of constrains ensures the per-node flow conservation condition holds. That is, for any node $v \in \mathcal{V}$ and data flow m, the total incoming flow should be equal to the total outgoing flow:

$$\sum_{l \in \mathrm{In}(v)} f_{l,m} + 1_{v=s(m)} r_m = \sum_{l \in \mathrm{Out}(v)} f_{l,m} + 1_{v=d(m)} r_m, \quad m = 1 \sim M,\ \forall\ v \in \mathcal{V}, \tag{2}$$

where $\mathrm{In}(v) \triangleq \{l \in \mathcal{L} \mid\ d_l = v\}$ and $\mathrm{Out}(v) \triangleq \{l \in \mathcal{L} \mid\ s_l = v\}$ denote the set of links going into and coming out of a node v respectively; $1_{v=x} = 1$ if $v = x$, otherwise $1_{v=x} = 0$.

In this work, we are interested in maximizing the minimum rate of all data flows. The problem can be formulated as the following linear program

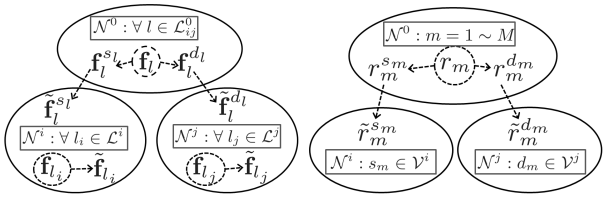
$$\max_{\mathbf{f},\,\mathbf{r}} \quad r_{\min} \quad \text{s.t.} \quad \mathbf{f} \geq 0,\ r_m \geq r_{\min},\ m = 1 \sim M \tag{3a}$$

$$\text{(1) and (2)}, \tag{3b}$$

where $\mathbf{f} \triangleq \{\mathbf{f}_l \mid l \in \mathcal{L}\}$ and $\mathbf{r} \triangleq \{r_{\min}, r_m \mid m = 1 \sim M\}$. It is worth noting that the minimum rate is picked here because it assures a fair rate allocation between data flows, and such utility has been adopted by many recent works; e.g., [8, 15]. Other objective functions can be used as well, for example the sum rate of all users, or the proportional fairness criteria.

3. A SEMI-ASYNCHRONOUS ROUTING ALGORITHM

In this section, for problem (3), we propose a semi-asynchronous algorithm that distributes the computation across NCs and allows parallel updates within each NC. The key step is to introduce a few auxiliary variables so that the coupling flow conservation and capacity constraints become separable among the NCs.



(a) Splitting the link flow rate. (b) Splitting the data flow rate.

Fig. 3. The structure of the introduced local auxiliary variables. The variable inside the dash circle indicates the original variable, and the variables pointed by dash line are the corresponding auxiliary variables.

Specifically, observe that each flow rate defined on the bordering links, $f_{l,m}, l \in \mathcal{L}^0_{ij}$ and $\forall\ m$, is shared among two flow conservation constraints, one for node $s_l \in \mathcal{V}^i$ and the other for node $d_l \in \mathcal{V}^j$. Each element of $\mathbf{f}$ also appears in one capacity constraint. To make the subproblems separable among NCs, we introduce the following variable splitting (see Fig. 3 for illustration)

- The flow rate $\mathbf{f}_l$ on each bordering link is split into *four* copies, namely $\tilde{\mathbf{f}}_l^{s_l}$, $\tilde{\mathbf{f}}_l^{d_l}$, $\mathbf{f}_l^{s_l}$, and $\mathbf{f}_l^{d_l}$. Among them, $\tilde{\mathbf{f}}_l^{s_l}$, $\tilde{\mathbf{f}}_l^{d_l}$ will be controlled by NC 0, $\mathbf{f}_l^{s_l}$ and $\mathbf{f}_l^{d_l}$ are individually managed by the two neighboring NCs.
- Each data flow rate r_m is split into the following copies: $\tilde{r}_m^{s_m}$, $\tilde{r}_m^{d_m}$, $r_m^{s_m}$, and $r_m^{d_m}$. The tuple $(\tilde{r}_m^{s_m}, \tilde{r}_m^{d_m})$ is managed by NC 0, $r_m^{s_m}$ and $r_m^{d_m}$ are managed by the source and the destination NCs of flow m.
- Within each subnetwork, introduce a new copy $\tilde{\mathbf{f}}_l$ for the link flow rate $\mathbf{f}_l$, $\forall\ l \in \mathcal{L} \setminus \mathcal{L}^0$.

For notational simplicity, we have created a few groups of the variables and denote them as $\mathbf{x}_0$, $\mathbf{x}^i_{01}$, $\mathbf{x}^i_{02}$, $\mathbf{x}_{i1}$,$\mathbf{x}_i$, $\mathbf{x}_{i2}$ $\mathbf{x}_{i3}$; see Table 1 for detailed definitions.

Obviously, the original variables and their splits should be identical, therefore we have the following sets of equality constraints

$$\underbrace{\mathbf{x}^i_{01} = \mathbf{x}^i_{02}}_{\text{in } \mathcal{N}^0},\quad \underbrace{\mathbf{x}_{i1} = \mathbf{x}^i_{01}}_{\text{in } \mathcal{N}^i \text{ and } \mathcal{N}^0}\ ,\ \underbrace{\mathbf{x}_{i2} = \mathbf{x}_{i3}}_{\text{in } \mathcal{N}^i},\ i = 1 \sim K. \tag{4}$$

By properly allocating the variables $\mathbf{x} = \{\mathbf{x}_i\}_{i=0 \sim K}$ to the constraints of problem (3), these constraints become separable over subnetworks. Moreover, (1) and (2) become independent to each other. Specifically, the reformulated constraints can be expressed as

$$\mathcal{X}_0 = \{r_{\min}, \mathbf{x}_{02} \mid \mathbf{1}^T \mathbf{f}_{l_0} \leq C_{l_0},\ \mathbf{f}_{l_0} \geq 0,\ r_m \geq r_{\min},\ \forall\ l_0 \in \mathcal{L}_0,\ \forall\ m\},$$

$$\mathcal{X}_{i1} = \{\mathbf{x}_{i1}, \mathbf{x}_{i2} \mid \sum_{l \in \mathrm{In}(v) \cap \mathcal{L}^i} \tilde{f}_{l,m} + \sum_{l \in \mathrm{In}(v) \cap \mathcal{L}^0} \tilde{f}^v_{l,m} + 1_{v=s_m} \tilde{r}^v_m = \sum_{l \in \mathrm{Out}(v) \cap \mathcal{L}^i} \tilde{f}_{l,m} + \sum_{l \in \mathrm{Out}(v) \cap \mathcal{L}^0} \tilde{f}^v_{l,m} + 1_{v=d_m} \tilde{r}^v_m,\ \forall\ v \in \mathcal{V}^i, \forall\ m\},$$

$$\mathcal{X}_{i2} = \{\mathbf{x}_{i3} \mid \mathbf{1}^T \mathbf{f}_{l_i} \leq C_{l_i},\ \mathbf{f}_{l_i} \geq \mathbf{0},\ \forall\ l_i \in \mathcal{L}^i\},\ i = 1 \sim K.$$

In summary, problem (3) is equivalently reformulated as

$$\max_{\mathbf{x}} \quad r_{\min} \ \text{ s.t. (4)},\ \{r_{\min}, \mathbf{x}_{02}\} \in \mathcal{X}_0, \tag{5a}$$

$$\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\} \in \mathcal{X}_{i1},\ \mathbf{x}_{i3} \in \mathcal{X}_{i2},\ i = 1 \sim K. \tag{5b}$$

After this reformulation, except the equality constraint (4), the objective function and the constraints are separable over subnetworks. This reformulation is a generalization of our previous synchronous routing algorithm [8]. The main difference is that in [8], a similar splitting is done for *each* node and link in the network,

Notations	Definitions	Physical meaning
$\mathbf{x}_0$	$\cup_{i=1}^K(\mathbf{x}_{01}^i \cup \mathbf{x}_{02}^i) \cup \{r_{\min}\}$	The variables stored in $\mathcal{N}^0$
$\mathbf{x}_{01}^i$	$\{\{r_m^v, \mathbf{f}_l^v\}_{l\in\mathcal{L}^0, v\in\mathcal{V}^i, \forall\ m}\}$	The auxiliary variables copied from $\mathbf{x}_{02}^i$
$\mathbf{x}_{02}^i$	$\{r_m, \mathbf{f}_{l_0} \mid l_0 \in \cup_{j\neq i}(\mathcal{L}_{ij}^0 \cup \mathcal{L}_{ji}^0),$ $\forall\ m$ s.t. s_m or $d_m \in \mathcal{V}^i\}$	The bordering flow rate variables and the data flow rates of $\mathcal{N}^i$
$\mathbf{x}_{02}$	$\cup_{i=1}^K \mathbf{x}_{02}^i$	All data flow rates and the flow rate variables between subnetworks
$\mathbf{x}_i$	$\mathbf{x}_{i1}^i \cup \mathbf{x}_{i2} \cup \mathbf{x}_{i3}$	The variables stored in $\mathcal{N}^i$
$\mathbf{x}_{i1}$	$\{\{\tilde{r}_m^v, \tilde{\mathbf{f}}_l^v\}_{l\in\mathcal{L}^0, v\in\mathcal{V}^i, \forall\ m}\}$	The auxiliary variables copied from $\mathbf{x}_{01}^i$ in $\mathcal{N}^0$
$\mathbf{x}_{i2}$, $\mathbf{x}_{i3}$	$\{\{\tilde{\mathbf{f}}_{l_i}\}_{l_i\in\mathcal{L}^i}\}$, $\{\{\mathbf{f}_{l_i}\}_{l_i\in\mathcal{L}^i}\}$	The flow rate variables within $\mathcal{N}^i$, and their corresponding local copies

Table 1. Summary of physical meaning and the relationship for variables stored in $\mathcal{N}^0$ and $\mathcal{N}^i$, $i = 1 \sim K$

which can result in too many auxiliary variables for large networks. Moreover, this new reformulation enables semi-asynchronous update rules, as will be explained shortly.

3.1. A Semi-Asynchronous Algorithm Based on BSUM-M

In this subsection, we develop a semi-asynchronous algorithm for (5) by applying the BSUM-M algorithm [16]. Intuitively, this approach is very similar to the multi-block ADMM approach, but with modifications in primal update rules and in the dual variable update. Therefore, the variables of each NC can be viewed as a variable block, and these variable blocks can be updated in different speed. The augmented Lagrangian function of problem (5) is

$$\begin{aligned} L_\rho(\mathbf{x},\boldsymbol{y}) = & \underbrace{r_{\min} + \sum_{i=1}^K \{\boldsymbol{y}_0^{iT}[\mathbf{x}_{01}^i - \mathbf{x}_{02}^i] - \frac{\rho}{2}\|\mathbf{x}_{01}^i - \mathbf{x}_{02}^i\|^2}_{\triangleq r_{\min}+\sum_{i=1}^K L_{i0}(\mathbf{x}_{01}^i, \mathbf{x}_{02}^i, \boldsymbol{y}_0^i)} \\ & + \underbrace{(\boldsymbol{y}_1^{iT}[\mathbf{x}_{i1} - \mathbf{x}_{01}^i] - \frac{\rho}{2}\|\mathbf{x}_{i1} - \mathbf{x}_{01}^i\|^2)\}}_{\triangleq L_{i1}(\mathbf{x}_{i1}, \mathbf{x}_{01}^i, \boldsymbol{y}_1^i)} \\ & + \underbrace{(\boldsymbol{y}_2^{iT}[\mathbf{x}_{i2} - \mathbf{x}_{i3}] - \frac{\rho}{2}\|\mathbf{x}_{i2} - \mathbf{x}_{i3}\|^2)}_{\triangleq L_{i2}(\mathbf{x}_{i2}, \mathbf{x}_{i3}, \boldsymbol{y}_2^i)}, \end{aligned} \tag{6}$$

where $\boldsymbol{y} \triangleq \{\boldsymbol{y}_0^i, \boldsymbol{y}_1^i, \boldsymbol{y}_2^i \mid i = 1 \sim K\}$ is the dual variable for (4) and ρ is the augmented Lagrangian parameter. Notice that the variable $\mathbf{x}_i$ in $\mathcal{N}^i$, $i = 1 \sim K$ appears separately in (6), and each of them is coupled with $\mathbf{x}_{01}^i$ in $\mathcal{N}^0$ only through $L_{i1}(\mathbf{x}_{i1}, \mathbf{x}_{01}^i, \boldsymbol{y}_1^i)$. By exploiting this separability property, we propose an Async-Routing algorithm to solve problem (5) semi-asynchronously among NCs with NC 0 being the central controller. The detailed algorithm is summarized in Table 2 and 3.

Specifically, for each NC i, $i = 1 \sim K$, we propose to update $\mathbf{x}_i$ with the latest $\mathbf{x}_{01}^i$ from NC 0. The procedure includes two steps for updating, respectively, $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}$ and $\mathbf{x}_{i3}$ as expressed in step 3 and 4 of Table 2. On the other hand, whenever NC 0 is idle and it has an updated $\mathbf{x}_{i1}$ from NC i, it computes the updated $\{r_{\min}, \mathbf{r}, \mathbf{x}_{02}^i\}$ and $\mathbf{x}_{01}^i$ by step 4-5 of Table 3. After that, the updated $\mathbf{x}_{01}^i$ is transmitted back to NC i to trigger the next round of update there. We should notice that, when NC 0 updates its local variables for NC i, it does not change the coupling variable $\mathbf{x}_{01}^j$ with respect to $\mathbf{x}_j$ in $\mathcal{N}^j$, $j \neq i$. The local update at NC j would not be affected since it still locally has the latest $\mathbf{x}_{01}^j$. Thus, it satisfies the semi-asynchronous scheme we described in Sec. 1. After these primal variables are updated within NC i and NC 0, the corresponding dual variables $\{\boldsymbol{y}_0^j\}_{j=1\sim K}$, $\boldsymbol{y}_1^i$, and $\boldsymbol{y}_2^i$ are locally updated in step 7-8 of Table 2 and step 7-9 of Table 3. Note that $\alpha^{(r)}$ is the stepsize for updating the dual variables at iteration index r. Moreover, since the proposed scheme is semi-asynchronous, each NC has its own iteration index k_i, $i = 0 \sim K$. On the other hand, NC 0 is synchronous

Async-Routing Algo.: Processed by NC i for $\mathcal{N}^i$, $i = 1 \sim K$

1: **Initialization** $\mathbf{x}_i^{(0)} = 0$, $\boldsymbol{y}_1^{i(0)} = \boldsymbol{y}_2^{i(0)} = \mathbf{0}$, $\mathbf{x}_{01}^i = \mathbf{0}$, and $k_i = 0$
2: **Repeat**
3: Update $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}$ by solving
$\{\mathbf{x}_{i1}^{(k_i+1)}, \mathbf{x}_{i2}^{(k_i+1)}\} \leftarrow \arg \max_{\{\mathbf{x}_{i1},\mathbf{x}_{i2}\}\in\mathcal{X}_{i1}} L_{i1}(\mathbf{x}_{i1}, \mathbf{x}_{01}^i, \boldsymbol{y}_1^{i(k_i)}) + L_{i2}(\mathbf{x}_{i2}, \mathbf{x}_{i3}^{(k_i)}, \boldsymbol{y}_2^{i(k_i)})$.
It can be solved in parallel over each data flow m.
4: $\mathbf{x}_{i3}^{(k_i+1)} \leftarrow \arg\max_{\mathbf{x}_{i3}\in\mathcal{X}_{i2}} L_{i2}(\mathbf{x}_{i2}^{(k_i+1)}, \mathbf{x}_{i3}, \boldsymbol{y}_2^{i(k_i)})$. It can be solved in parallel over each link of $\mathcal{L}_i$.
5: Send $\mathbf{x}_{i1}^{(k_i+1)}$ to NC 0
6: Receive the updated $\mathbf{x}_{01}^i$ from NC 0.
7: $\boldsymbol{y}_1^{i(k_i+1)} \leftarrow \boldsymbol{y}_1^{i(k_i)} - \alpha^{(k_i)}(\mathbf{x}_{i1}^{(k_i+1)} - \mathbf{x}_{01}^i)$
8: $\boldsymbol{y}_2^{i(k_i+1)} \leftarrow \boldsymbol{y}_2^{i(k_i)} - \alpha^{(k_i)}(\mathbf{x}_{i2}^{(k_i+1)} - \mathbf{x}_{i3}^{(k_i+1)})$
9: $k_i \leftarrow k_i + 1$
10: **Until** a desired stopping criterion is met

Table 2. The updating procedure for each NC i, $i = 1 \sim K$

with each of others, it keeps all the iteration indexes. Observe that for every time interval T, which is longer than the maximum time required for each NC to update its local auxiliary variables and exchanging information with NC 0, every variable block will be updated at least once. Hence, the variable update sequence can be viewed as essentially cyclic rule [4].

From the computational perspective, it is worth mentioning that the update of $\{r_{\min}, \mathbf{r}, \mathbf{x}_{02}^i\}$ at NC 0 only relates to the data flow rates and the flow rate leaving from or going to $\mathcal{N}^i$. Moreover, all the subproblems at NCs are decoupled over links or data flows, and have either parallel closed-form solution [8], or can be solved by some existing efficient network optimization procedures, e.g., [11]. Thus, the update can be very efficient. The convergence of the proposed Async-Routing algorithm is addressed by the following theorem.

Theorem 1 *If the step size $\alpha^{(r)}$ in the proposed Async-Routing algorithm satisfies*

$$\sum_{r=1}^{\infty} \alpha^{(r)} = \infty, \ \lim_{r\to\infty} \alpha^{(r)} = 0.$$

Then the $\mathbf{x}^{(r)}$ generated by Async-Routing algorithm satisfies (4) *in the limit as $r \to \infty$, and every limit point of $\mathbf{x}^{(r)}$ is a primal optimal solution of* (5).

The detailed proof follows a similar line for BSUM-M [16] with proper modifications for essentially cyclic rule, so it is omitted due to space limitation. Moreover, the proof relies on the constraints being polyhedral, and the objective function is linear or strongly convex for $\mathbf{x}$, which is different from being strongly convex for (partially) asynchronous dual decomposition [12, 13]. Hence, the approach is suitable for proportional fairness or sum-rate design as well.

Async-Routing Algo.: Processed by NC 0 for $\mathcal{N}^0$

1: **Initialization** $\mathbf{x}_0^{(0)} = \mathbf{0}$, $\boldsymbol{y}^{(0)} = \mathbf{0}$, and $k_i = 0, i = 0 \sim K$
2: **Repeat**
3: Pick one unprocessed $\mathbf{x}_{i1}$ that has been received or wait until receiving one $\mathbf{x}_{i1}$, $i = 1 \sim K$
4: Update $\{\mathbf{x}_{02}, r_{\min}\}$ by solving $\{\mathbf{x}_{02}^{(k_0+1)}, r_{\min}^{(k_0+1)}\}$

$$\leftarrow \arg \max_{\{\mathbf{x}_{02}, r_{\min}\}} r_{\min} + \sum_{i=1}^{K} L_{i0}(\mathbf{x}_{01}^{i(k_0)}, \mathbf{x}_{02}^{i}, \boldsymbol{y}_0^{i(k_0)})$$

$$\text{s.t. } \{r_{\min}, \mathbf{x}_{02}\} \in \mathcal{X}_0,\ \mathbf{f}_l = \mathbf{f}_l^{(k_0)},\ \forall\, l \notin \cup_{j \neq i} \mathcal{L}_{ij}^0 \cup \mathcal{L}_{ji}^0$$

5: Update $\mathbf{x}_{01}^i$ by solving

$$\mathbf{x}_{01}^{i(k_0+1)} \leftarrow \arg \max_{\mathbf{x}_{01}^i}\ L_{i0}(\mathbf{x}_{01}^i, \mathbf{x}_{02}^{i(k_0+1)}, \boldsymbol{y}_0^{i(k_0)}) + L_{i1}(\mathbf{x}_{i1}, \mathbf{x}_{01}^i, \boldsymbol{y}_1^{i(k_0)})$$

Step 4 and 5 both can be solved in parallel over the assumed master node and the links connecting $\mathcal{V}^0$ and $\mathcal{V}^i$.
6: Send $\mathbf{x}_{01}^{i(k_0+1)}$ to NC i.
7: $\boldsymbol{y}_0^{j(k_0+1)} \leftarrow \boldsymbol{y}_0^{j(k_0)} - \alpha^{(k_0)}(\mathbf{x}_{01}^{j(k_0+1)} - \mathbf{x}_{02}^{j(k_0+1)}), j = 1 \sim K$
8: $\boldsymbol{y}_1^{i(k_0+1)} \leftarrow \boldsymbol{y}_1^{i(k_0+1)} - \alpha^{(k_i)}(\mathbf{x}_{i1} - \mathbf{x}_{01}^{i(k_0+1)})$
9: $\boldsymbol{y}_1^{j(k_0+1)} \leftarrow \boldsymbol{y}_1^{j(k_0+1)}, j \neq i$, $k_0 \leftarrow k_0 + 1$ and $k_i \leftarrow k_i + 1$
10: **Until** A desired stopping criteria is met

Table 3. The updating procedure for NC 0

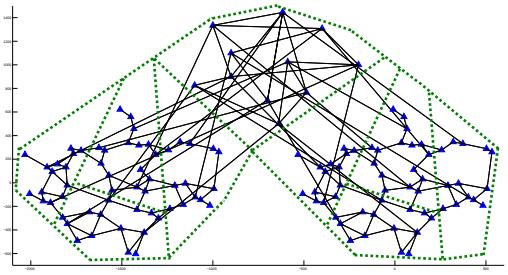


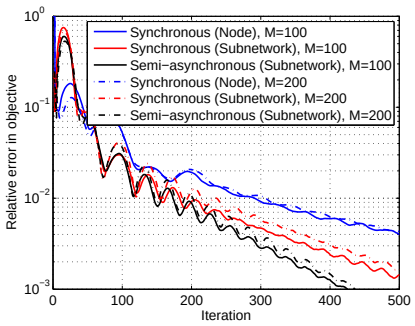
Fig. 4. The considered network topology with 9 groups of nodes.

Before closing this subsection, we would like to comment on the choice of the parameterρ. In order to further reduce the communication overhead between NCs with fewer number of iterations, one can apply different ρ for each individual constraint of (4). Each ρ is adaptively adjusted by the primal and dual residual of that constraint, see [5, Chap.3.3] with minor modification to incorporate the effect of time varying of $\alpha^{(r)}$. For brevity, we will omit the details here.

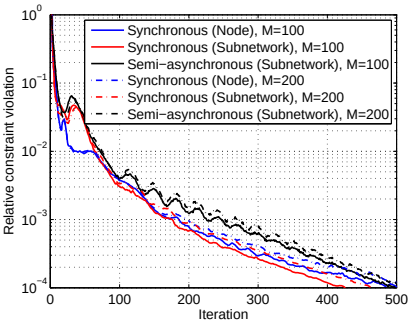
4. SIMULATION RESULTS AND CONCLUSIONS

In this section, we report some numerical results on the performance of the proposed Async-Routing algorithms as applied to a network with 126 network nodes. These network nodes are split into 9 subnetworks with 306 directed links within these subnetworks and 100 directed links connecting the subnetworks. The topology and the connectivity of this network are shown in Fig. 4. These link capacities are generated uniformly randomly in each simulation sample. It follows the rule that links within each subnetworks is [50,100] (MBits/s) and links between each subnetwork is [20,50] (MBits/s). The source and the destination nodes of each data flow is randomly selected from network nodes, and all simulation results are averaged over 200 randomly selected data flow pairs and link capacity.

In the following, for each constraint of (4), $\alpha^{(r)} = 100\rho/(\sqrt{r} + 100)$ and ρ is initialized as 0.0005. The parameters for ρ adaptation are $\mu = 100$, $\tau^{\text{incr}} = \tau^{\text{decr}} = 1.2$. We compare the proposed Async-Routing algorithm with two benchmarks. 1) [Synchronous (Node)] The synchronous ADMM routing algorithm that decomposes the network to the node level [8]. 2) [Synchronous (Subnetwork)] The synchronous version of Async-Routing algorithm where NC 0 updates its local variables $\mathbf{x}_0$ only when the latest $\mathbf{x}_{1i}, \forall\, i$, are available. Two performance metrics are used. The relative error in objective and constraint violation are, respectively, defined as $|r_{\min}^{(k_0)} - r_{\min}^{\text{optimal}}|/r_{\min}^{\text{optimal}}$ and the maximum $|x - x^{\text{local}}|/\max\{1, x\}$ over all variables where x (resp. x^{local}) is the original variable (resp. local auxiliary one). Moreover, for fair comparison, we count one iteration for Async-Routing algorithm as NC 0 has updated 9 times, so on average, every NCs have communicated with NC 0 one time. We also assume for each NC i, the time delay for local computation and information exchange with NC 0 is uniformly distributed, and it follows $[1, 50]$ (unit time).



(a) The relative objective error versus the iteration



(b) The relative constraint violation versus the iteration

Fig. 5. The performance of the convergence rate for all the comparing algorithms.

Fig. 5 shows the performance of these algorithms when the number of data flows is 100 and 200. One can observe that the scheme that decomposes to subnetworks is much more efficient than the one that decomposes to network nodes, regardless of whether it is synchronous or not. This is mainly due to the fact the number of auxiliary variables is much smaller if we take subnetwork structure into consideration. For the semi-asynchronous scheme, the efficiency is very close to the synchronous counterpart in both performance metrics. We should emphasize that in practice, the semi-asynchronous scheme does not need to wait for other NCs and hence can be much more efficient if it only uses similar number of iterations as that of synchronous scheme. A future work is to implement the Async-Routing algorithm in a parallel system to validate the efficiency. Further, we can observe a similar performance trend for the number of data flows up to 200, showing the scalability of the proposed algorithm.

5. REFERENCES

[1] A. Ghosh, S. Ha, E. Crabbe, and J. Rexford, "Scalable multi-class traffic management in data center backbone networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2673–2684, Dec. 2013.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.

[3] M. Chiang, S. Low, A. R. Calderbank, and J. C. Doyle, "Layering as optimization decompositioon: A mathematical theory of network architectures," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.

[4] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1995.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[6] C. Feng, H. Xu, and B. Li, "An alternating direction method approach to cloud traffic management," *submitted to IEEE/ACM Trans. Networking*, 2014.

[7] M. Leinonen, M. Codreanu, and M. Juntti, "Distributed joint resource and routing optimization in wireless sensor networks via alternating direction method of multipliers," *IEEE Trans. Wireless Communications*, vol. 12, no. 11, pp. 5454–5467, Nov. 2013.

[8] W.-C. Liao, M. Hong, H. Farmanbar, X. Li, Z.-Q. Luo, and H. Zhang, "Min flow rate maximization for software defined radio access networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 6, pp. 1282–1294, Jun. 2014.

[9] Open Networking Foundation, "Software-defined networking: The new norm for networks," 2012, White paper.

[10] Huawei Technologies Inc., "5G: A technology vision," 2013, White paper.

[11] D. P. Bertsekas, P. Hosein, and P. Tseng, "Relaxation methods for network flow problems with convex arc costs," *SIAM Journal on Control and Optimization*, vol. 25, no. 5, pp. 1219–1243, Sep. 1987.

[12] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," *IEEE/ACM Trans. Networking*, vol. 7, no. 6, pp. 861–874, Dec. 1999.

[13] L. Bui, A. Eryilmaz, R. Srikant, and W. Xinzhou, "Asynchronous congestion control in multi-hop wireless networks with maximal matching-based scheduling," *IEEE/ACM Trans. Networking*, vol. 16, no. 4, pp. 826–839, Aug. 2008.

[14] E. Wei and A. Ozdaglar, "On the O(1/k) convergence of asynchronous distributed alternating direction method of multipliers," 2013, Preprint, available at arXiv:1307.8254.

[15] E. Danna, S. Mandal, and A. Singh, "A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering," in *Proc. of IEEE INFOCOM*, 2012, pp. 846–854.

[16] M. Hong, T.-H. Chang, X. Wang, M. Razaviyayn, S. Ma, and Z.-Q. Luo, "A block successive upper bound minimization method of multipliers for linearly constrained convex optimization," *submitted for publication, available at http://arxiv.org/abs/1401.7079*, 2014.