A Distributed Newton's Method for Joint Multi-Hop Routing and Flow Control: Theory and Algorithm

Jia Liu* Hanif D. Sherali†
* Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210
† Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA 24061

***Abstract*—The fast growing scale and heterogeneity of current communication networks necessitate the design of distributed cross-layer optimization algorithms. So far, the standard approach of distributed cross-layer design is based on dual decomposition and the subgradient algorithm, which is a first-order method that has a slow convergence rate. In this paper, we focus on solving a joint multi-path routing and flow control (MRFC) problem by designing a new distributed Newton's method, which is a second-order method and enjoys a quadratic rate of convergence. The major challenges in developing a distributed Newton's method lie in decentralizing the computation of the Hessian matrix and its inverse for both the primal Newton direction and dual variable updates. By appropriately reformulating, rearranging, and exploiting the special problem structures, we show that it is possible to decompose such computations into source nodes and links in the network, thus eliminating the need for global information. Furthermore, we derive closed-form expressions for both the primal Newton direction and dual variable updates, thus significantly reducing the computational complexity. The most attractive feature of our proposed distributed Newton's method is that it requires almost the same scale of information exchange as in first-order methods, while achieving a quadratic rate of convergence as in centralized Newton methods. We provide extensive numerical results to demonstrate the efficacy of our proposed algorithm. Our work contributes to the advanced paradigm shift in cross-layer network design that is evolving from first-order to second-order methods.**

I. Introduction

The scale of communication networks has been growing rapidly in recent years as the demand for data access continues to rise exponentially. As a result, maintaining a centralized network control unit has become increasingly difficult or even undesirable in many situations, such as joint multi-path routing and congestion control in the Internet, sensor networks, or wireless ad hoc networks. In such cases, distributed algorithms are not only desirable, but also necessary.

In the literature, the standard distributed approach for jointly optimizing multi-path routing and flow control (MRFC) is based on the Lagrangian dual decomposition framework and the subgradient method for dual updates, where subgradients (first-order supports of the dual function) are used as search directions (see, e.g., [1]–[3] and references therein). The dual decomposition framework and the subgradient approach are also related to the celebrated throughput-optimal "back-pressure" algorithm, which is the foundation of a large number of interesting routing and scheduling schemes (see the seminal work of Tassiulas and Ephremides [4] and many other later works in this direction).[1] However, despite its simplicity and theoretical appeal, the first-order subgradient method does not work well in practice because it suffers from a slow rate of convergence (and typically exhibits a zigzagging phenomenon if the objective function is ill-conditioned) and is very sensitive to step-size selections. These limitations motivate us to design a distributed Newton's method for the MRFC problem. The fundamental rationale of this approach is that a distributed Newton's method would exploit both first and second-order information (more precisely, the gradient and Hessian of the underlying problem) in determining search directions. As a result, a properly designed distributed Newton's method also enjoys the same *quadratic convergence rate* as in the classical Newton-type methods [6], [7].

However, due to a number of major challenges, research on second-order based distributed algorithms is still in its infancy and results are rather limited. To our knowledge, only a handful of works exist in this area (see Section II for more detailed discussions). The first major challenge is that the computation of the primal Newton direction typically requires taking the inverse of the Hessian matrix of the underlying problem (or solving a linear equation system), which is not always easy for large-scale problems, let alone being done in a distributed fashion. Therefore, when designing distributed second-order algorithms, one needs to figure out how to decompose the inverse of Hessian matrix and distribute each piece to each network entity (i.e., a node or a link) in such a way that each piece can be computed using only local information or via a limited scale of information exchange between network entities. This task is not trivial except in some special problems, and certainly not in our case. The second major challenge is that the computation of dual variables (which represent certain pricing information) in a distributed second-order method also requires taking the inverse of some complex transformation of the Hessian matrix and needs global information. In fact, how to compute the dual variables in a distributed way has remained largely unaddressed until very recently, when some interesting ideas based on Gaussian belief propagation [8] (to avoid direct matrix inversion) or matrix splitting [9] (to iteratively compute the matrix inverse) were proposed for some relatively simpler network optimization problems [10]–[13]. However, it remains unclear whether these ideas can be

[1] It can be shown by some appropriate scaling that the queue lengths can be interpreted as the dual variables in the dual decomposition framework – see [5, Section 4] for a more detailed discussion.

extended to more complex cross-layer optimization problems. Therefore, our goal in this work is centered around tackling these difficulties. The main results and contributions in this work are as follows:

- We show that, by appropriately reformulating and rearranging, it is possible to expose a *block diagonal* structure in the Hessian matrix of the MRFC problem. As a result, the Hessian matrix can be decomposed with respect to source nodes and links in the network. Furthermore, we show that the inverse of each submatrix block can be computed in closed-form, thus significantly reducing the computational complexity. This complexity reduction is made possible by a keen observation of the special structure of the submatrix for each network entity.
- Based on the above block-diagonal structure and the second-order properties of the coefficient matrix of Problem MRFC, we derive closed-form expressions for both the primal Newton direction and dual variables computations by generalizing the matrix-splitting idea of [13]. Moreover, we provide important "back-pressure" interpretations and insights for these closed-form expressions, as well as the connections to, and differences from, first-order methods, thus further advancing our understanding of second-order approaches in network optimization theory.
- We provide extensive numerical studies to demonstrate the efficacy of our proposed distributed Newton's method. Our numerical results show that the proposed algorithm converges two orders of magnitude faster than the first-order subgradient approach.

To our knowledge, this paper is the first work that develops a *fully distributed* Newton's algorithm for the MRFC problem. Our work contributes to a new and exciting paradigm shift in cross-layer network design that is evolving from first-order to second-order methods.

The remainder of this paper is organized as follows. In Section II, we review some related work in the literature, putting our work in a historical and comparative perspective. Section III introduces the network model and problem formulation. Section IV provides preliminary discussion on the centralized Newton's method and points out the challenges in distributed implementations. Section V is the key part of this paper, which develops the principal components of our proposed distributed Newton's method. Section VI provides numerical results and Section VII concludes this paper.

II. Related Work

Early attempts at second-order methods for network optimization (centralized or distributed) date back to the 1980s. Bertsekas and Gafni [14] proposed a centralized projected Newton's method for multi-commodity flow problems, and Klincewicz [15] proposed a distributed conjugate gradient direction method to solve a pure minimum cost flow routing problem. We remark that these conjugate gradient based algorithms belong to the class of (fixed-metric) quasi-Newton methods [6] and have a slower convergence rate compared to pure Newton's methods. A more recent work on a distributed (variable-metric) quasi-Newton method was reported in [16], where Bolognani and Zampieri showed that the celebrated BFGS algorithm [6] can be decentralized to solve the optimal reactive power flow problem in smart-grids. The aforementioned quasi-Newton based methods also rely on gradient projections to find feasible search directions. In contrast, most of the recent works in this area [10]–[13], [17], including ours, are based on the interior-point approach [18], which has been shown to be more efficient. To our knowledge, the first known interior-point based algorithm for a pure flow control problem (i.e., fixed routes) was reported in [17], where Zymnis *et al.* proposed a centralized truncated-Newton primal-dual interior-point method. For the same problem, Bickson *et al.* [10], [11] later developed a distributed algorithm based on the Gaussian belief propagation technique, but without providing a provable guarantee for its convergence. On the other hand, Jadbabaie *et al.* [12] designed a distributed Newton's method for solving a pure minimum cost routing problem (i.e., fixed source rates), where a consensus-based local averaging scheme was used to compute the Newton direction. Although convergence of this scheme can be established by using spectral graph theory [19], its convergence rate could potentially be unsatisfactory. Our work is most related to the work of Wei *et al.* [13]. However, we consider a cross-layer MRFC problem, while their work studied the same pure flow control problem as in [10], [11], [17]. Although there exists some similarity in our matrix splitting approach with that in [13], we point out that due to a completely different problem setting, the proof to show the applicability of the matrix splitting technique in MRFC and the resulting distributed algorithm are completely new. Moreover, several important networking insights can be drawn from these new analyses and proofs.

III. Network Model and Problem Formulation

We first introduce the notation used in this paper. We use boldface to denote matrices and vectors. We let $\mathbf{A}^T$ denote the transpose of the matrix $\mathbf{A}$. $\mathrm{Diag}\{\mathbf{A}_1, \ldots, \mathbf{A}_N\}$ represents the block diagonal matrix with matrices $\mathbf{A}_1, \ldots, \mathbf{A}_N$ on its main diagonal. $\mathrm{diag}\{\mathbf{A}\}$ represents the vector containing the main diagonal entries of $\mathbf{A}$. We let $(\mathbf{A})_{ij}$ represent the entry in the i-th row and j-th column of $\mathbf{A}$. We let $\mathbf{I}$ denote the identity matrix with dimension determined from the context. $\mathbf{A} \succ 0$ represents that $\mathbf{A}$ is symmetric and positive definite (PD). $\mathbf{1}$ and $\mathbf{0}$ denote vectors whose elements are all ones and zeros, respectively, where their dimensions are determined from the context. $(\mathbf{v})_m$ represents the m-th entry of vector $\mathbf{v}$. For a vector $\mathbf{v}$ and a matrix $\mathbf{A}$, $\mathbf{v} \geq \mathbf{0}$ and $\mathbf{A} \geq \mathbf{0}$ represent that $\mathbf{v}$ and $\mathbf{A}$ are element-wise nonnegative.

In this paper, a multi-hop network is represented by a directed graph $\mathcal{G} = \{\mathcal{N}, \mathcal{L}\}$, where $\mathcal{N}$ and $\mathcal{L}$ are the set of nodes and links, respectively. We assume that $\mathcal{G}$ is connected. The cardinalities of the sets $\mathcal{N}$ and $\mathcal{L}$ are $|\mathcal{N}| = N$ and $|\mathcal{L}| = L$, respectively. We use the so-called *node-arc incidence matrix* (NAIM) $\mathbf{A} \in \mathbb{R}^{N \times L}$ to represent the network topology of $\mathcal{G}$ [20]. Let $\mathrm{Tx}(l)$ and $\mathrm{Rx}(l)$ denote the transmitting and receiving nodes of link l, respectively. Then, the entries in $\mathbf{A}$

are defined as follows:

$$(\mathbf{A})_{nl} = \begin{cases} 1, & \text{if } n = \mathrm{Tx}(l), \\ -1, & \text{if } n = \mathrm{Rx}(l), \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

In the network, different source nodes send different data to their intended destination nodes through *multi-path* and *multi-hop* routing. Suppose that there is a total of F sessions in the network, representing F different commodities. We denote the source and destination nodes of session f as $\mathrm{Src}(f)$ and $\mathrm{Dst}(f)$, respectively. The source flow rate of session f is denoted by a scalar $s_f \in \mathbb{R}_+$. For session f, we use a *source–destination vector* vector $\mathbf{b}_f \in \mathbb{R}^N$ to represent the supply–demand relationship of session f. More specifically, the entries in $\mathbf{b}_f$ are defined as follows:

$$(\mathbf{b}_f)_n = \begin{cases} 1, & \text{if } n = \mathrm{Src}(f), \\ -1, & \text{if } n = \mathrm{Dst}(f), \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

For every link l, we let $x_l^{(f)} \geq 0$ represent the flow amount of session f on link l. We assume that the network is a flow-balanced system, i.e., the following flow balance constraints hold at each node:

$$\sum_{l \in \mathcal{O}(n)} x_l^{(f)} - \sum_{l \in \mathcal{I}(n)} x_l^{(f)} = s_f, \quad \text{if } n = \mathrm{Src}(f),$$
$$\sum_{l \in \mathcal{O}(n)} x_l^{(f)} - \sum_{l \in \mathcal{I}(n)} x_l^{(f)} = 0, \quad \text{if } n \neq \mathrm{Src}(f), \mathrm{Dst}(f),$$
$$\sum_{l \in \mathcal{I}(n)} x_l^{(f)} - \sum_{l \in \mathcal{O}(n)} x_l^{(f)} = s_f, \quad \text{if } n = \mathrm{Dst}(f),$$

where $\mathcal{O}(n)$ and $\mathcal{I}(n)$ represent the sets of outgoing and incoming links at node n, respectively. We define $\mathbf{x}^{(f)} \triangleq [x_1^{(f)}, \ldots, x_L^{(f)}]^T \in \mathbb{R}^L$ as the *routing vector* for session f across all links. Using the notation $\mathbf{A}$, $\mathbf{b}_f$, and $\mathbf{x}^{(f)}$, the flow balance constraints above can be compactly written as

$$\mathbf{A}\mathbf{x}^{(f)} - s_f \mathbf{b}_f = \mathbf{0}, \quad \forall f = 1, 2, \ldots, F. \quad (3)$$

Moreover, upon taking a closer look at (3), it is easy to see that the coefficient matrix $\mathbf{A}$ is not of full row rank (because all rows sum up to zero). To eliminate the redundant rows, we let $\mathbf{A}^{(f)} \in \mathbb{R}^{(N-1)\times L}$ be obtained by deleting the row corresponding to node $\mathrm{Dst}(f)$ from $\mathbf{A}$. It is easy to verify that $\mathbf{A}^{(f)}$ is of full row rank [20]. Likewise, we let $\widetilde{\mathbf{b}}^{(f)} \in \mathbb{R}^{N-1}$ be obtained by deleting the entry corresponding to node $\mathrm{Dst}(f)$ from $\mathbf{b}^{(f)}$. Accordingly, we rewrite (3) as:

$$\mathbf{A}^{(f)}\mathbf{x}^{(f)} - s_f \widetilde{\mathbf{b}}^{(f)} = \mathbf{0}, \quad \forall f = 1, 2, \ldots, F. \quad (4)$$

We assume that each link in the network is capacitated. The capacity of link l, denoted by C_l, is assumed to be fixed, which models conventional wireline networks or wireless networks with static, orthogonal channels and fixed transmission power. Since the total network flow traversing a link cannot exceed the link's capacity limit, we have $\sum_{f=1}^{F} x_l^{(f)} \leq C_l$, $l = 1, \ldots, L$. We associate a utility function $U_f(s_f) : \mathbb{R}_+ \to \mathbb{R}$ with each session f. We assume that the utility functions are additive so that the overall network utility is given by $\sum_{f=1}^{F} U_f(s_f)$. We also assume that the utility functions U_f are strictly concave, monotonically increasing, twice continuously differentiable, and reversely self-concordant (see [7], [18] for the definition of self-concordance; as an example, $\log(s_f)$, which represents "proportional fairness" [2], is reversely self-concordant). Our objective is to maximize the sum of utilities. Putting together the routing and flow control constraints described earlier, we can formulate the MRFC problem as follows:

MRFC:

$$\begin{aligned} \text{Maximize} \quad & \sum_{f=1}^{F} U_f(s_f) \\ \text{subject to} \quad & \mathbf{A}^{(f)}\mathbf{x}^{(f)} - s_f \widetilde{\mathbf{b}}^{(f)} = \mathbf{0}, && \forall f = 1, \ldots, F \\ & \sum_{f=1}^{F} x_l^{(f)} \leq C_l, && \forall l = 1, \ldots, L \\ & x_l^{(f)} \geq 0, \ \forall f, l; \quad s_f \geq 0, \ \forall f. \end{aligned}$$

Due to its convexity and separable structure in the dual domain, Problem MRFC can be solved distributedly based on a dual decomposition and subgradient optimization framework (see, e.g., [1] or [5, Section 4] for a quick overview). However, as mentioned earlier, its convergence performance is unsatisfactory in practice. In what follows, we will investigate a new distributed second-order method to solve Problem MRFC.

IV. Centralized Newton's Method: A Primer

In this section, we provide some preliminary discussion on using the centralized Newton's method to solve Problem MRFC, along with an analysis of the challenges in developing a distributed Newton's algorithm.

A. Problem Reformulation

We start by reformulating MRFC into a form that only has equality constraints so that the centralized Newton's method can be applied. Following the standard approach used for interior-point methods [18], we employ a logarithmic barrier function to represent each inequality constraint including non-negativity restrictions, and we accommodate this within the objective function. The augmented objective function is rewritten as follows:

$$f(\mathbf{y}) \triangleq -t \sum_{f=1}^{F} U_f(s_f) - \sum_{l=1}^{L} \log\left(C_l - \sum_{f=1}^{F} x_l^{(f)}\right) - \sum_{f=1}^{F} \log(s_f) - \sum_{l=1}^{L}\sum_{f=1}^{F} \log(x_l^{(f)}), \quad (5)$$

where $\mathbf{y} \triangleq \left[s_1, \ldots s_F, (\mathbf{x}^{(1)})^T, \ldots, (\mathbf{x}^{(F)})^T\right]^T \in \mathbb{R}^{(L+1)F}$ and where $t > 0$ is a parameter to track the central path in the interior-point method as $t \to \infty$ [7]. Furthermore, we denote

$$\mathbf{M} \triangleq \left[\begin{array}{ccc|ccc} \widetilde{\mathbf{b}}^{(1)} & & & -\mathbf{A}^{(1)} & & \\ & \ddots & & & \ddots & \\ & & \widetilde{\mathbf{b}}^{(F)} & & & -\mathbf{A}^{(F)} \end{array}\right] \in \mathbb{R}^{(N-1)F \times (L+1)F}.$$

With this notation, we can write the revised MRFC problem as follows:

R-MRFC:

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{y}) \\ \text{subject to} & \mathbf{M}\mathbf{y} = \mathbf{0}. \end{array} \tag{6}$$

Observe that the approximation accuracy of R-MRFC can be controlled by t: as t increases, the first term in $f(\mathbf{y})$ dominates the barrier functions and R-MRFC becomes a better approximation of MRFC. It can be shown that the approximation error is bounded by K/t [7], where K is some constant depending on problem specifics. This implies that, as $t \to \infty$, the solution of R-MRFC converges to that for the original problem. Moreover, as t increases, solving R-MRFC does not suffer from numerical instability as long as $f(\mathbf{y})$ is self-concordant, which is guaranteed here because $-U_f(\cdot)$ and all barrier terms are self-concordant (we refer readers to [7], [18] for more detailed discussions on self-concordance).

B. Centralized Newton's Method for the R-MRFC Problem

Starting from an initial feasible solution $\mathbf{y}^0$, the centralized Newton's method searches for an optimal solution using the following iterative rule:

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \pi^k \Delta\mathbf{y}^k, \quad k \geq 0, \tag{7}$$

where $\pi^k > 0$ is a positive step-size. In (7), $\Delta\mathbf{y}^k$ denotes the Newton direction, which is the solution to the following linear equation system (obtained by deriving the Karush-Kuhn-Tucker (KKT) system of the second-order approximation of $f(\mathbf{y})$ [6], [7]):

$$\begin{bmatrix} \mathbf{H}_k & \mathbf{M}^T \\ \mathbf{M} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y}^k \\ \mathbf{w}^k \end{bmatrix} = -\begin{bmatrix} \nabla f(\mathbf{y}^k) \\ \mathbf{0} \end{bmatrix}, \tag{8}$$

where $\mathbf{H}_k \triangleq \nabla^2 f(\mathbf{y}^k) \in \mathbb{R}^{(L+1)F \times (L+1)F}$ is the Hessian of $f(\mathbf{y})$ at $\mathbf{y}^k$, and the vector $\mathbf{w}^k \in \mathbb{R}^{(N-1)F}$ contains the dual variables associated with the flow balance constraints $\mathbf{M}\mathbf{y} = \mathbf{0}$ at the k-th iteration.

It can be easily verified that the coefficient matrix of the linear equation system in (8) is nonsingular. Therefore, the primal Newton direction $\Delta\mathbf{y}^k$ and the dual variables $\mathbf{w}^k$ can be uniquely determined by solving (8). However, solving for $\Delta\mathbf{y}^k$ and $\mathbf{w}^k$ simultaneously via (8) clearly requires global information. The key towards solving (8) in a decentralized manner is to rewrite it as the following iterative steps:

$$\Delta\mathbf{y}^k = -\mathbf{H}_k^{-1}(\nabla f(\mathbf{y}^k) + \mathbf{M}^T\mathbf{w}^k), \tag{9}$$

$$(\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T)\mathbf{w}^k = -\mathbf{M}\mathbf{H}_k^{-1}\nabla f(\mathbf{y}^k). \tag{10}$$

Thus, given $\mathbf{y}^k$, we can solve for $\mathbf{w}^k$ from (10), and hence, solve for $\Delta\mathbf{y}^k$ from (9), which can thus be used in (7) to compute $\mathbf{y}^{k+1}$ (with an appropriate step-size π^k). Unfortunately, the iterative steps in (9) and (10) remain difficult to decentralize because the Hessian matrix $\mathbf{H}_k$ has a coupled structure with respect to different sessions (see [5, Section 6.1] for more details). Moreover, as we shall see in Section V, computing the inverse of $(\mathbf{M}\mathbf{H}_k^{-1}\mathbf{M}^T)$ still requires global information. Therefore, in the next section, our goal is to further reformulate R-MRFC to enable an iterative scheme to compute $\Delta\mathbf{y}^k$ and $\mathbf{w}^k$ in a distributed fashion.

V. A Distributed Newton's Method

In this section, we will present the key components of our proposed distributed Newton's method. We will first further reformulate Problem R-MRFC in Section V-A, following which we will introduce some basic second-order structural properties of the reformulated problem in Section V-B. The distributed computations of the primal Newton direction and the dual variables will be presented in Sections V-C and V-D, respectively. Finally, Section V-E addresses some other implementation issues.

A. Problem Rearrangement

As mentioned earlier, the first major hurdle in decentralizing Newton's method is the coupled structure of the Hessian matrix for $f(\mathbf{y})$. To address this, we start by evaluating the first and second partial derivatives of $f(\mathbf{y})$. Since $f(\mathbf{y})$ is separable with respect to each flow commodity f and link l, the only non-zero partial derivatives are:

$$\frac{\partial f(\mathbf{y}^k)}{\partial s_f} = -tU_f'(s_f) - \frac{1}{s_f}, \qquad \frac{\partial f(\mathbf{y}^k)}{\partial x_l^{(f)}} = \frac{1}{\delta_l} - \frac{1}{x_l^{(f)}},$$

$$\frac{\partial^2 f(\mathbf{y}^k)}{\partial (s_f)^2} = -tU_f''(s_f) + \frac{1}{(s_f)^2}, \qquad \frac{\partial^2 f(\mathbf{y}^k)}{\partial (x_l^{(f)})^2} = \frac{1}{\delta_l^2} + \frac{1}{(x_l^{(f)})^2},$$

$$\frac{\partial^2 f(\mathbf{y}^k)}{\partial x_l^{(f_1)} \partial x_l^{(f_2)}} = \frac{1}{\delta_l^2},$$

where $\delta_l \triangleq C_l - \sum_{f=1}^F x_l^{(f)}$ represents the *unused link capacity* of link l. From the partial derivative computations, we note that $f(\mathbf{y})$ is separable with respect to each link. This prompts us to rearrange $\mathbf{y}$ and $\mathbf{M}$ based on links as follows:

$$\widetilde{\mathbf{y}} \triangleq \left[s_1 \cdots s_F \middle| x_1^{(1)} \cdots x_1^{(F)} \middle| \cdots\cdots \middle| x_L^{(1)} \cdots x_L^{(F)} \right]^T \in \mathbb{R}^{(L+1)F},$$

$$\text{and } \widetilde{\mathbf{M}} = \begin{bmatrix} \widetilde{\mathbf{B}} & \mathbf{A}_1 & \cdots & \mathbf{A}_L \end{bmatrix},$$

$$\text{where } \widetilde{\mathbf{B}} \triangleq \begin{bmatrix} \widetilde{\mathbf{b}}^{(1)} & & \\ & \ddots & \\ & & \widetilde{\mathbf{b}}^{(F)} \end{bmatrix} \text{ and } \mathbf{A}_l \triangleq \begin{bmatrix} -\mathbf{a}_l^{(1)} & & \\ & \ddots & \\ & & -\mathbf{a}_l^{(F)} \end{bmatrix},$$

and where in the definition of $\mathbf{A}_l$, the vector $\mathbf{a}_l^{(f)}$ is the l-th column in the matrix $\mathbf{A}^{(f)}$ (i.e., $\mathbf{A}^{(f)} = \left[\mathbf{a}_1^{(f)}, \mathbf{a}_2^{(f)}, \ldots, \mathbf{a}_L^{(f)}\right]$).

As a result, R-MRFC can be equivalently rewritten as follows:

R2-MRFC:

$$\begin{array}{ll} \text{Minimize} & f(\widetilde{\mathbf{y}}) \\ \text{subject to} & \widetilde{\mathbf{M}}\widetilde{\mathbf{y}} = \mathbf{0}. \end{array} \tag{11}$$

By the same token as in Section IV-B, the Newton direction of R2-MRFC is the solution to the following linear equation system:

$$\begin{bmatrix} \widetilde{\mathbf{H}}_k & \widetilde{\mathbf{M}}^T \\ \widetilde{\mathbf{M}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \Delta\widetilde{\mathbf{y}}^k \\ \widetilde{\mathbf{w}}^k \end{bmatrix} = -\begin{bmatrix} \nabla f(\widetilde{\mathbf{y}}^k) \\ \mathbf{0} \end{bmatrix}, \tag{12}$$

where $\widetilde{\mathbf{w}}^k$ represents the dual variables for the flow balance constraint $\widetilde{\mathbf{M}}\widetilde{\mathbf{y}} = \mathbf{0}$. As we will show later in Section V-C, the new Hessian $\widetilde{\mathbf{H}}_k$ is a *block-diagonal matrix*, for which the inverse computation is easier to decentralize. Also, the entries in $\widetilde{\mathbf{w}}^k$ are arranged as $[(\widetilde{\mathbf{w}}_k^{(1)})^T, \ldots (\widetilde{\mathbf{w}}_k^{(F)})^T]^T$, where $\widetilde{\mathbf{w}}_k^{(f)}$ is of the following form:

$$\widetilde{\mathbf{w}}_k^{(f)} \triangleq \left[\widetilde{w}_1^{(f)}, \ldots, \widetilde{w}_{\mathrm{Dst}(f)-1}^{(f)}, \widetilde{w}_{\mathrm{Dst}(f)+1}^{(f)}, \ldots, \widetilde{w}_N^{(f)}\right]^T \in \mathbb{R}^{N-1}.$$

Note that in the equation above, we have dropped the iteration index k within [ ]for notational simplicity. For the same reason, in the rest of the paper, the iteration index k will be dropped whenever such an omission does not cause confusion. Also, we let $\widetilde{w}_{\mathrm{Dst}(f)}^{(f)} \equiv 0$, for all f. This will help simplify the closed-form expressions in Theorem 6. More detailed discussions on the physical meaning of the dual variables $\widetilde{\mathbf{w}}^k$ will also be provided in Section V-C.

B. Basic Second-Order Properties of $\mathbf{a}_l^{(f)}$ and $\widetilde{\mathbf{b}}^{(f)}$

In R2-MRFC, we have introduced two new vectors, namely, $\mathbf{a}_l^{(f)}$ and $\widetilde{\mathbf{b}}^{(f)}$. Here, we state some of their basic second-order properties, which will be used extensively later in simplifying the distributed computations of the primal Newton direction and dual variables. Most of these properties can be verified using simple matrix computations based on the definitions of $\mathbf{a}_l^{(f)}$ and $\widetilde{\mathbf{b}}^{(f)}$. Thus, we omit the formal proofs of these properties in this paper. We first define an index function $\beta_f(n)$, $n \neq \mathrm{Dst}(f)$, as follows:

$$\beta_f(n) \triangleq \begin{cases} n & \text{if } n < \mathrm{Dst}(f), \\ n-1 & \text{if } n > \mathrm{Dst}(f). \end{cases} \tag{13}$$

Then, we have the following basic second-order property for $\widetilde{\mathbf{b}}(f)$:

Lemma 1. *The rank-one matrix $\widetilde{\mathbf{b}}^{(f)}(\widetilde{\mathbf{b}}^{(f)})^T$ has the following structure:*

$$\left(\widetilde{\mathbf{b}}^{(f)}(\widetilde{\mathbf{b}}^{(f)})^T\right)_{ij} = \begin{cases} 1, & \textit{if } i = j = \beta_f(\mathrm{Src}(f)). \\ 0, & \textit{otherwise}. \end{cases}$$

For $\mathbf{a}_l^{(f)}$, we have the following two second-order properties:

Lemma 2. *The rank-one matrix $\mathbf{a}_l^{(f)}(\mathbf{a}_l^{(f)})^T$ has the following structure:*

- *Case 1: If $\mathrm{Tx}(l), \mathrm{Rx}(l) \neq \mathrm{Dst}(f)$, then $\mathbf{a}_l^{(f)}(\mathbf{a}_l^{(f)})^T$ has four non-zero entries, where*

$$\left(\mathbf{a}_l^{(f)}(\mathbf{a}_l^{(f)})^T\right)_{ij} = \begin{cases} 1, & \textit{if } i = j,\ i = \beta_f(\mathrm{Tx}(l)) \textit{ or } \beta_f(\mathrm{Rx}(l)), \\ -1, & \textit{if } i = \beta_f(\mathrm{Tx}(l)),\ j = \beta_f(\mathrm{Rx}(l)) \\ & \textit{or } i = \beta_f(\mathrm{Rx}(l)),\ j = \beta_f(\mathrm{Tx}(l)), \\ 0, & \textit{otherwise}; \end{cases}$$

- *Case 2: If $\mathrm{Rx}(l) = \mathrm{Dst}(f)$, then $\mathbf{a}_l^{(f)}(\mathbf{a}_l^{(f)})^T$ has one non-zero entry, where*

$$\left(\mathbf{a}_l^{(f)}(\mathbf{a}_l^{(f)})^T\right)_{ij} = \begin{cases} 1, & \textit{if } i = j = \beta_f(\mathrm{Tx}(l)), \\ 0, & \textit{otherwise}. \end{cases}$$

Lemma 3. *The rank-one matrix $\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T$ has the following structure:*

- *Case 1: If $\mathrm{Tx}(l), \mathrm{Rx}(l) \neq \mathrm{Dst}(f_1)$ and $\mathrm{Tx}(l), \mathrm{Rx}(l) \neq \mathrm{Dst}(f_2)$, then $\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T$ has four non-zero entries, where*

$$\left(\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T\right)_{ij} = \begin{cases} 1, & \textit{if } i = \beta_{f_1}(\mathrm{Tx}(l)),\ j = \beta_{f_2}(\mathrm{Tx}(l)) \\ & \textit{or } i = \beta_{f_1}(\mathrm{Rx}(l)),\ j = \beta_{f_2}(\mathrm{Rx}(l)), \\ -1, & \textit{if } i = \beta_{f_1}(\mathrm{Tx}(l)),\ j = \beta_{f_2}(\mathrm{Rx}(l)) \\ & \textit{or } i = \beta_{f_1}(\mathrm{Rx}(l)),\ j = \beta_{f_2}(\mathrm{Tx}(l)), \\ 0, & \textit{otherwise}; \end{cases}$$

- *Case 2: If $\mathrm{Rx}(l) = \mathrm{Dst}(f_1)$ or $\mathrm{Rx}(l) = \mathrm{Dst}(f_2)$, then $\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T$ has two non-zero entries, where*

$$\left(\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T\right)_{ij} = \begin{cases} 1, & \textit{if } i = \beta_{f_1}(\mathrm{Tx}(l)),\ j = \beta_{f_2}(\mathrm{Tx}(l)), \\ -1, & \textit{if } i = \beta_{f_1}(\mathrm{Tx}(l)),\ j = \beta_{f_2}(\mathrm{Rx}(l)) \\ & \textit{or } i = \beta_{f_1}(\mathrm{Rx}(l)),\ j = \beta_{f_2}(\mathrm{Tx}(l)), \\ 0, & \textit{otherwise}. \end{cases}$$

- *Case 3: If $\mathrm{Tx}(l) = \mathrm{Src}(f_1)$, $\mathrm{Rx}(l) = \mathrm{Dst}(f_2)$, or $\mathrm{Tx}(l) = \mathrm{Dst}(f_1)$, $\mathrm{Rx}(l) = \mathrm{Src}(f_2)$, then $\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T$ has one non-zero entry, where*

$$\left(\mathbf{a}_l^{(f_1)}(\mathbf{a}_l^{(f_2)})^T\right)_{ij} = \begin{cases} -1, & \textit{if } i = \beta_{f_1}(\mathrm{Tx}(l)),\ j = \beta_{f_2}(\mathrm{Rx}(l)), \\ & \textit{or } i = \beta_{f_1}(\mathrm{Rx}(l)),\ j = \beta_{f_2}(\mathrm{Tx}(l)), \\ 0, & \textit{otherwise}. \end{cases}$$

C. Distributed Computation of the Primal Newton Direction

Using the same approach as in Section IV-B, Eq. (12) can be rewritten as the following iterative steps:

$$\Delta\widetilde{\mathbf{y}}^k = -\widetilde{\mathbf{H}}_k^{-1}(\nabla f(\widetilde{\mathbf{y}}^k) + \widetilde{\mathbf{M}}^T\widetilde{\mathbf{w}}^k), \tag{14}$$

$$(\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T)\widetilde{\mathbf{w}}^k = -\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\nabla f(\widetilde{\mathbf{y}}^k). \tag{15}$$

Now, consider the Hessian matrix inverse $\widetilde{\mathbf{H}}_k^{-1}$ of Problem R2-MRFC. As mentioned earlier, since $f(\mathbf{y}^k)$ is separable based on links, $\widetilde{\mathbf{H}}_k$ has the following *block diagonal* structure:

$$\widetilde{\mathbf{H}}_k = \mathrm{Diag}\left\{\mathbf{S}, \mathbf{X}_1, \ldots, \mathbf{X}_L\right\} \in \mathbb{R}^{(L+1)F \times (L+1)F},$$

where $\mathbf{S} \triangleq \mathrm{Diag}\left\{-tU_1''(s_1) + \frac{1}{s_1^2}, \ldots, -tU_F''(s_F) + \frac{1}{s_F^2}\right\} \in \mathbb{R}^{F \times F}$ is diagonal; and where $\mathbf{X}_l \in \mathbb{R}^{F \times F}$ is a symmetric matrix with entries being defined as follows:

$$(\mathbf{X}_l)_{f_1, f_2} = \begin{cases} \frac{1}{\delta_l^2} + \frac{1}{\left(x_l^{(f_1)}\right)^2}, & \text{if } f_1 = f_2, \\ \frac{1}{\delta_l^2}, & \text{if } f_1 \neq f_2. \end{cases} \tag{16}$$

It then follows from the block diagonal structure of $\widetilde{\mathbf{H}}_k$ that $\widetilde{\mathbf{H}}_k^{-1} = \mathrm{Diag}\left\{\mathbf{S}^{-1}, \mathbf{X}_1^{-1}, \ldots, \mathbf{X}_L^{-1}\right\}$. Noting that $\mathbf{S}^{-1}$ is diagonal, we obtain the following result:

Lemma 4 (Distributedness of computing $\mathbf{S}^{-1}$)**.** *The matrix $\mathbf{S}^{-1}$ can be computed in a distributed fashion (source node-wise) as $\mathbf{S}^{-1} = \mathrm{Diag}\{\frac{1}{-tU_1''(s_1)+1/s_1^2}, \ldots, \frac{1}{-tU_F''(s_F)+1/s_F^2}\}$.*

Next, we consider the computation of $\mathbf{X}_l^{-1}$. For convenience, we define a new vector $\widehat{\mathbf{x}}_l \triangleq \left[x_l^{(1)}, \ldots, x_l^{(F)}, \delta_l\right]^T \in$

$\mathbb{R}^{F+1}$. Then, by exploiting the special structure of $\mathbf{X}_l$ in (16), we can establish the following important result:

Theorem 5 (Distributed closed-form expression for $\mathbf{X}_l^{-1}$)**.** *The entries of $\mathbf{X}_l^{-1}$ can be computed distributedly (link-wise) in closed-form as follows:*

$$(\mathbf{X}_l^{-1})_{f_1 f_2} = \begin{cases} \left(x_l^{(f_1)}\right)^2 \left(1 - \frac{\left(x_l^{(f_1)}\right)^2}{\|\widehat{\mathbf{x}}_l\|^2}\right), & \text{if } 1 \le f_1 = f_2 \le F, \\ -\frac{\left(x_l^{(f_1)} x_l^{(f_2)}\right)^2}{\|\widehat{\mathbf{x}}_l\|^2}, & \text{if } 1 \le f_1 \ne f_2 \le F. \end{cases}$$

The proof of Theorem 5 is based on a keen observation of the decomposable structure of $\mathbf{X}_l$ in (16) and the Sherman–Morrison–Woodbury matrix inversion formula [6]. Due to limited space, we refer readers to [5, Appendix A] for the details of the proof.

Combining Lemma 4, Theorem 5 and all related discussions earlier, we can show that the primal Newton direction can be computed distributedly as indicated in the following theorem:

Theorem 6. *Given dual variables $\widetilde{\mathbf{w}}$, the Newton direction Δs_f and $\Delta x_l^{(f)}$ for each source rate s_f and link flow rate $x_l^{(f)}$ can be computed using local information at each source node s and link l, respectively, as follows:*

$$\Delta s_f = \frac{s_f\left(t s_f U_f'(s_f) + 1 - s_f w_{\mathrm{Src}(f)}^{(f)}\right)}{1 - t s_f^2 U_f''(s_f)}, \quad \forall f, \tag{17}$$

$$\Delta x_l^{(f)} = \left(x_l^{(f)}\right)^2 \times \left[\left(1 - \frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}\right)\left(\frac{1}{x_l^{(f)}} - \frac{1}{\delta_l} + \widetilde{w}_{\mathrm{Tx}(l)}^{(f)} - \widetilde{w}_{\mathrm{Rx}(l)}^{(f)}\right) + \sum_{f'=1,\neq f}^{F} \frac{(x_l^{(f')})^2}{\|\widehat{\mathbf{x}}_l\|^2}\left(\frac{1}{x_l^{(f')}} - \frac{1}{\delta_l} + \widetilde{w}_{\mathrm{Tx}(l)}^{(f')} - \widetilde{w}_{\mathrm{Rx}(l)}^{(f')}\right)\right], \ \forall l, f. \tag{18}$$

The key steps of proving Theorem 6 are: (i) applying $\widetilde{\mathbf{H}}_k^{-1} = \mathrm{Diag}\left\{\mathbf{S}^{-1}, \mathbf{X}_1^{-1}, \ldots, \mathbf{X}_L^{-1}\right\}$, Lemma 4 and Theorem 5 to Eq. (14); and (ii) using Lemmas 1 and 2 to simplify the result. Due to limited space, we relegate the details of the proof to [5, Appendix B].

Remark 1. *An important remark for Theorem 6 is in order. Theorem 6 not only provides a closed-form expression for a distributed primal Newton direction, but also offers an interesting network interpretation. Here, we can think of the difference of the dual variables $(\widetilde{w}_{\mathrm{Tx}(l)}^{(f)} - \widetilde{w}_{\mathrm{Rx}(l)}^{(f)})$ in (18) as "the queue length difference" in the back-pressure algorithm, although $\widetilde{w}_n^{(f)}$ itself cannot be exactly interpreted as queue length (since it can be positive or negative). Note that in (18), $1 - \frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}$ and $\frac{(x_l^{(f')})^2}{\|\widehat{\mathbf{x}}_l\|^2}$ are all positive. Hence, if the positive $(\widetilde{w}_{\mathrm{Tx}(l)}^{(f)} - \widetilde{w}_{\mathrm{Rx}(l)}^{(f)})$–values outweigh the negative ones, i.e., the "pressure" on the transmitter side of link l is greater than the "pressure" on the receiver side, then $x_l^{(f)}$ will be increased in the next iteration. Note also that, unlike first-order methods, the decision to increase or decrease $x_l^{(f)}$ considers not only the "pressure difference" of flow f, but also the "pressure difference" from other flows at link l (via an appropriate weighting scheme as evident in (18)).*

D. Distributed Computation of the Dual Variables

As mentioned earlier, given $\mathbf{y}^k$, the dual variables $\mathbf{w}^k$ can be computed using (15). However, solving for $\mathbf{w}^k$ using (15) cannot be implemented in a distributed fashion because computing $(\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T)^{-1}$ still requires global information. In what follows, we will study the special structure of $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$. Then, we will show how to apply a matrix splitting scheme to iteratively compute $\mathbf{w}^k$ without requiring global information.

First, noting that $\widetilde{\mathbf{M}}$ can be written as a partitioned matrix as $\widetilde{\mathbf{M}} = \left[\widetilde{\mathbf{B}}, \mathbf{A}_1, \ldots, \mathbf{A}_L\right]$, we decompose $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$ as

$$\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T = \widetilde{\mathbf{B}}\mathbf{S}^{-1}\widetilde{\mathbf{B}}^T + \sum_{l=1}^{L} \mathbf{A}_l \mathbf{X}_l^{-1} \mathbf{A}_l^T. \tag{19}$$

Let $\Phi(n) \triangleq \mathcal{I}(n) \cup \mathcal{O}(n)$ be the set of all links that touch node n. Let $\Gamma(n_1, n_2) \triangleq \{l \in \mathcal{L} : \mathrm{Tx}(l) = n_1 \text{ and } \mathrm{Rx}(l) = n_2, \text{ or } \mathrm{Tx}(l) = n_2 \text{ and } \mathrm{Rx}(l) = n_1\}$ be the set of all links between nodes n_1 and n_2. Then, using the decomposable structure in (19) and after some algebraic manipulations, we have the following important result for the structure of $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$ (see [5, Section 6.4] for detailed derivations):

Theorem 7. *The matrix $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$ can be written in the following partitioned form:*

$$\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T = \begin{bmatrix} \mathbf{D}_1 - \widehat{\mathbf{D}}_1 & -\mathbf{G}_{12} & \cdots & -\mathbf{G}_{1F} \\ -\mathbf{G}_{21} & \mathbf{D}_2 - \widehat{\mathbf{D}}_2 & \cdots & -\mathbf{G}_{2F} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{G}_{F1} & -\mathbf{G}_{F2} & \cdots & \mathbf{D}_F - \widehat{\mathbf{D}}_F \end{bmatrix},$$

where the structures of the matrices $\mathbf{D}_f$, $\widehat{\mathbf{D}}_f$, and $\mathbf{G}_{f_1 f_2}$ are as follows: (i) $(\mathbf{D}_f)_{ii} = \sum_{l\in\Phi(n)}(x_l^{(f)})^2 + \frac{1}{-tU_f''(s_f)+1/(s_f)^2}$ if $i = \beta_f(\mathrm{Src}(f))$ or $\sum_{l\in\Phi(n)}(x_l^{(f)})^2$ otherwise; $(\mathbf{D}_f)_{ij} = -\sum_{l\in\Gamma(n_1,n_2)}(x_l^{(f)})^2$ if $i = \beta_f(n_1)$ and $j = \beta_f(n_2)$; (ii) $(\widehat{\mathbf{D}}_f)_{ii} = \sum_{l\in\Phi(n)} \frac{(x_l^{(f)})^4}{\|\widehat{\mathbf{x}}_l\|^2}$, $(\widehat{\mathbf{D}}_f)_{ij} = -\sum_{l\in\Gamma(n_1,n_2)} \frac{(x_l^{(f)})^4}{\|\widehat{\mathbf{x}}_l\|^2}$ if $i = \beta_f(n_1)$, and $j = \beta_f(n_2)$; (iii) $(\mathbf{G}_{f_1 f_2})_{ij} = \sum_{l\in\Phi(n)} \frac{(x_l^{(f_1)} x_l^{(f_2)})^2}{\|\widehat{\mathbf{x}}_l\|^2}$ if $i = \beta_{f_1}(n)$, $j = \beta_{f_2}(n)$, or $-\sum_{l\in\Gamma(n_1,n_2)} \frac{(x_l^{(f_1)} x_l^{(f_2)})^2}{\|\widehat{\mathbf{x}}_l\|^2}$ if $i = \beta_{f_1}(n_1)$, $j = \beta_{f_2}(n_2)$.

With Theorem 7, we are now in a position to design a distributed scheme to compute $\widetilde{\mathbf{w}}_k$ using the matrix splitting technique. Historically, the idea of matrix splitting originates from designing iterative schemes to solve linear equation systems [9]. Consider a consistent linear equation system $\mathbf{F}\mathbf{z} = \mathbf{d}$, where $\mathbf{F} \in \mathbb{R}^{n\times n}$ is a nonsingular matrix and $\mathbf{z}, \mathbf{d} \in \mathbb{R}^n$. Let $\mathbf{F}$ be split into a nonsingular matrix $\mathbf{F}_1$ and another matrix $\mathbf{F}_2$ as $\mathbf{F} = \mathbf{F}_1 - \mathbf{F}_2$. Also, let $\mathbf{z}^0$ be an arbitrary starting vector. Then, a sequence of approximate solutions can be generated by the following iterative scheme [9]:

$$\mathbf{z}^{k+1} = (\mathbf{F}_1^{-1}\mathbf{F}_2)\mathbf{z}^k + \mathbf{F}_1^{-1}\mathbf{d}, \quad k \ge 0. \tag{20}$$

Generally, $\mathbf{F}_1$ should be easier to invert than $\mathbf{F}$. The iterative scheme in (20) is convergent to the unique solution $\mathbf{z} = \mathbf{F}^{-1}\mathbf{b}$ if and only if $\rho(\mathbf{F}_1^{-1}\mathbf{F}_2) < 1$, where $\rho(\cdot)$ represents the spectral radius of a matrix.

We now adopt the matrix splitting scheme in (20) to compute $\widetilde{\mathbf{w}}^k$. First, we let $\mathbf{\Lambda}_k$ be the diagonal matrix having the same main diagonal of $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$, i.e., $\mathbf{\Lambda}_k = \mathrm{Diag}\left\{\mathrm{diag}\left\{\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T\right\}\right\}$. We let $\mathbf{\Omega}_k$ denote the matrix containing the remaining entries after subtracting $\mathbf{\Lambda}_k$ from $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$, i.e., $\mathbf{\Omega}_k = \widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T - \mathbf{\Lambda}_k$. Further, we define a diagonal matrix $\overline{\mathbf{\Omega}}_k$ where the diagonal entries are given by $(\overline{\mathbf{\Omega}}_k)_{ii} = \sum_j |(\mathbf{\Omega}_k)_{ij}|$. Then, we have the following result:

Lemma 8. *Let the matrix $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$ be split as $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T = (\mathbf{\Lambda}_k + \alpha\overline{\mathbf{\Omega}}_k) - (\alpha\overline{\mathbf{\Omega}}_k - \mathbf{\Omega}_k)$, where $\alpha > \frac{1}{2}$ is a given parameter. Then, the following sequence $\{\widetilde{\mathbf{w}}^k\}$ generated by*

$$\widetilde{\mathbf{w}}^{k+1} = (\mathbf{\Lambda}_k + \alpha\overline{\mathbf{\Omega}}_k)^{-1}(\alpha\overline{\mathbf{\Omega}}_k - \mathbf{\Omega}_k)\widetilde{\mathbf{w}}^k + (\mathbf{\Lambda}_k + \alpha\overline{\mathbf{\Omega}}_k)^{-1}(-\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\nabla f(\widetilde{\mathbf{y}}^k)) \quad (21)$$

converges to the solution of (15) as $k \to \infty$.

To prove Lemma 8, we use the definitions of $\mathbf{\Lambda}_k$, $\mathbf{\Omega}_k$, and $\overline{\mathbf{\Omega}}_k$ to check that both the sum and difference of $\mathbf{\Lambda}_k + \alpha\overline{\mathbf{\Omega}}_k$ and $\alpha\overline{\mathbf{\Omega}}_k - \mathbf{\Omega}_k$ are strictly diagonally dominant. Then, according to [21, Corollary 7.2.3], the diagonal dominance result implies the positive definiteness of the sum and difference, which in turn implies Lemma 8 according to [22, Theorem 2.5.3] (see [5, Appendix D] for the details of the proof).

Remark 2. *The matrix splitting scheme in Lemma 8 is inspired by, and is a generalization of, the scheme in [13]. The goal of both schemes is to construct a diagonal matrix for which the inverse can be separated and computed by each node (as in our case) or each link (as in [13]). However, our scheme differs from that of [13] in the following two aspects. First, the matrix $\widetilde{\mathbf{M}}\widetilde{\mathbf{H}}_k^{-1}\widetilde{\mathbf{M}}^T$ in our paper is not element-wise non-negative (cf. [13]). Hence, the definition of matrix $\overline{\mathbf{\Omega}}_k$ is different from the counterpart in [13] and leads to a different proof. Second, we parameterize the splitting scheme (using α) to allow for tuning the convergence speed in (21), where the scheme in [13] is a special case of our scheme when $\alpha = 1$.*

Some comments on the parameter α are in order at this point. From (20), it can be seen that the solution error decreases in magnitude approximately by a factor of $\rho(\mathbf{F}_1^{-1}\mathbf{F}_2)$. Thus, the smaller $\rho(\mathbf{F}_1^{-1}\mathbf{F}_2)$ is, the faster the convergence we might expect of the matrix splitting scheme. Using the comparison theorem in [9, Theorem 2.3], we can derive the following result for the selection of α (see [5, Appendix E] for the details of the proof):

Proposition 9. *Consider two alternative matrix splitting schemes with parameters α_1 and α_2, respectively, satisfying $\frac{1}{2} < \alpha_1 \leq \alpha_2$. Let ρ_{α_1} and ρ_{α_2} be their respective spectral radii. Then, $\rho_{\alpha_1} \leq \rho_{\alpha_2}$.*

Proposition 9 implies that in order to make the matrix splitting scheme converge faster, we should choose a smaller α, i.e., we can let $\alpha = \frac{1}{2} + \epsilon$, where $\epsilon > 0$ is small.

Next, we will show that the matrix splitting scheme in Lemma 8 can indeed be implemented in a distributed fashion. For convenience, we define a link set as follows: $\Psi(n, f) \triangleq \{l \in \mathcal{I}(n) \cup \mathcal{O}(n) : \mathrm{Tx}(l) = \mathrm{Dst}(f) \text{ or } \mathrm{Rx}(l) = \mathrm{Dst}(f)\}$. We also let $_S(a)$ denote the set indicator function, which takes value 1 if $a \in S$ and 0 otherwise. Then, we have the following result:

Theorem 10. *Given a primal solution $\widetilde{\mathbf{y}}^k$, the update of the dual variable $w_n^{(f)}$ can be computed using local information at each node as follows:*

$$w_n^{(f)}(k+1) = \frac{1}{U_n^f(k)}(V_{n,1}^{(f)}(k) + V_{n,2}^{(f)}(k) - W_n^f(k)), \quad (22)$$

where $U_n^{(f)}(k)$, $V_{n,1}^{(f)}(k)$, $V_{n,2}^{(f)}(k)$, and $W_n^{(f)}(k)$ are, respectively, defined as

$$U_n^{(f)}(k) \triangleq \sum_{l\in\Phi(n)} [1 + \alpha(1 - \ _{\Psi(n,f)}(l))](x_l^{(f)})^2\Big(1 - \frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}\Big) + \sum_{f'=1,\neq f}^{F} \sum_{l\in\Psi(n,f')} (1 + \ _{\Psi(n,f')}(l)) \frac{\alpha(x_l^{(f)}x_l^{(f')})^2}{\|\widehat{\mathbf{x}}_l\|^2} + \frac{\ _{\mathrm{Src}(f)}(n)}{-tU_f''(s_f) + \frac{1}{(s_f)^2}}, \quad (23)$$

$$V_{n,1}^{(f)}(k) \triangleq \sum_{l\in\mathcal{I}(n)\setminus\Psi(n,f)} (x_l^{(f)})^2\Big(1 - \frac{((x_l^{(f)})^2)}{\|\widehat{\mathbf{x}}_l\|^2}\Big)(\widetilde{w}_{\mathrm{Tx}(l)}^{(f)} + \alpha\widetilde{w}_{\mathrm{Rx}(l)}^{(f)}) + \sum_{l\in\mathcal{O}(n)\setminus\Psi(n,f)} (x_l^{(f)})^2\Big(1 - \frac{((x_l^{(f)})^2)}{\|\widehat{\mathbf{x}}_l\|^2}\Big)(\widetilde{w}_{\mathrm{Rx}(l)}^{(f)} + \alpha\widetilde{w}_{\mathrm{Tx}(l)}^{(f)}) - \sum_{f'=1,\neq f}^{F}\Big(\sum_{l\in\Phi(n)} (1 + \ _{\Psi(n,f')}(l)) \frac{\alpha(x_l^{(f)}x_l^{(f')})^2}{\|\widehat{\mathbf{x}}_l\|^2}\Big)\widetilde{w}_n^{(f)}, \quad (24)$$

$$V_{n,2}^{(f)}(k) \triangleq \sum_{f'=1,\neq f}^{F}\Big(\Big(\sum_{l\in\mathcal{O}(n)} \frac{(x_l^{(f)}x_l^{(f')})^2}{\|\widehat{\mathbf{x}}_l\|^2} - \sum_{l\in\mathcal{I}(n)} \frac{(x_l^{(f)}x_l^{(f')})^2}{\|\widehat{\mathbf{x}}_l\|^2}\Big) \times (\widetilde{w}_{\mathrm{Tx}(l)}^{(f')} - \widetilde{w}_{\mathrm{Rx}(l)}^{(f')})\Big), \quad (25)$$

$$W_n^{(f)}(k) \triangleq \begin{cases} \Big(1 - \frac{x_l^{(f)}}{\delta_l}\Big)\Big[\sum_{l\in\mathcal{O}(n)}\Big(1 - \sum_{f'=1}^{F}\frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}x_l^{(f')}\Big) - \sum_{l\in\mathcal{I}(n)}\Big(1 - \sum_{f'=1}^{F}\frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}x_l^{(f')}\Big)\Big], & \text{if } n \neq \mathrm{Src}(f), \\ \sum_{l\in\mathcal{O}(n)}\Big(1 - \sum_{f'=1}^{F}\frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}x_l^{(f')}\Big) - \sum_{l\in\mathcal{I}(n)}\Big(1 - \sum_{f'=1}^{F}\frac{(x_l^{(f)})^2}{\|\widehat{\mathbf{x}}_l\|^2}x_l^{(f')}\Big)\Big] + \frac{s_f(1+ts_fU_f'(s_f))}{ts_f^2U_f''(s_f)-1}, & \text{otherwise}. \end{cases} \quad (26)$$

Theorem 10 can be proved by computing element-wise expansions of (21) based on Theorem 7. Due to limited space, we relegate the lengthy proof to [5, Appendix F].

Remark 3. *There are several interesting remarks pertaining to Theorem 10. First, from (23), (24), (25), and (26), we can see that all the information needed to update $w_n^{(f)}$ are either locally available at node n or at links that touch node n. This shows that the splitting scheme can be distributedly implemented. Second, it can be verified that $V_{n,1}^{(f)}$ involves "same-session" as well as "cross-session" quadratic terms of $x_l^{(f)}$ coming into and going out of node n. This is starkly different*

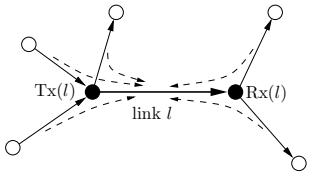

Fig. 1. Computing $\Delta x_l^{(f)}$ only requires information exchanges from links one-hop away from link l.

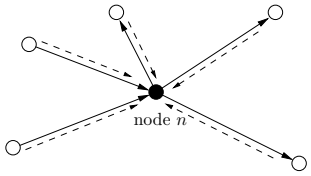

Fig. 2. Computing $\widetilde{w}_n^{(f)}$ only requires information exchanges from nodes one-hop away from node n.

from the dual update scheme in the subgradient method (cf. [5, Section 4]) in that all flow-related quantities here are of second-order and weighted by $\widetilde{w}_{\mathrm{Tx}(l)}^{(f)} + \widetilde{w}_{\mathrm{Rx}(l)}^{(f)}$ (when $\alpha = 1$), which can be loosely interpreted as "total queue length" on both sides of l (see Remark 1). $V_{n,2}^{(f)}$ involves a "back-pressure" ($\widetilde{w}_{\mathrm{Tx}(l)}^{(f)} - \widetilde{w}_{\mathrm{Rx}(l)}^{(f)}$) weighting mechanism . But unlike $V_{n,1}^{(f)}$, the second-order quantities in $V_{n,2}^{(f)}$ are only related to "cross-session" flow products $x_l^{(f)} x_l^{(f')}$. Third, although the dual update scheme within a second-order method is more complex at each node, the more rapid convergence rate of a second-order method, with its accompanying less information exchange, outweigh this local computational cost increase.

E. Implementation of the Distributed Newton's Method

Here, we discuss some other relevant implementation issues pertaining to the distributed Newton's method.

Information Exchange Scale Analysis: First, consider the primal Newton direction update in Theorem 6. We can see that to compute Δs_f, we need s_f and $\widetilde{w}_{\mathrm{Src}(f)}^{(f)}$. Since s_f is available at $\mathrm{Src}(f)$ and $\widetilde{w}_{\mathrm{Src}(f)}^{(f)}$ can also be computed at $\mathrm{Src}(f)$ (cf. Theorem 10), there is *no* need for any information exchange in computing Δs_f. To compute $\Delta x_l^{(f)}$, we can see from (18) that we need $x_l^{f'}$, $f' = 1, \ldots, F$, $\widetilde{w}_{\mathrm{Tx}(l)}^{(f')}$, and $\widetilde{w}_{\mathrm{Rx}(l)}^{(f')}$. Clearly, $x_l^{f'}$, $f' = 1, \ldots, F$ are already available at link l. From Theorem 10, we can see that $\widetilde{w}_{\mathrm{Tx}(l)}^{(f')}$ and $\widetilde{w}_{\mathrm{Rx}(l)}^{(f')}$ can be computed using flow and dual information with respect to links that share $\mathrm{Tx}(l)$ and $\mathrm{Rx}(l)$. This implies that computing $\Delta x_l^{(f)}$ only requires exchanging information *one-hop* away from link l, as shown in Fig. 1.

Next, consider the updating of dual variables. From Theorem 10, we can see that the computation of $\widetilde{w}_n^{(f)}$ requires $x_l^{(f')}$, $\widetilde{w}_{\mathrm{Tx}(l)}^{(f')}$, and $\widetilde{w}_{\mathrm{Rx}(l)}^{(f')}$, where $l \in \Phi(n)$, $f' = 1, \ldots, F$. Clearly, $x_l^{(f')}$ and $l \in \Phi(n)$ are available at node n. Also, $\widetilde{w}_{\mathrm{Tx}(l)}^{(f')}$ and $\widetilde{w}_{\mathrm{Rx}(l)}^{(f')}$ are either available at node n itself or are available at nodes one-hop away from node n. This implies that computing $\widetilde{w}_n^{(f)}$ only requires exchanging information from nodes *one-hop* away from node n, as shown in Fig. 2.

Algorithm Initialization: One simple initialization scheme is to let each $\mathrm{Src}(f)$ choose an initial value ϵ_f and equally distribute ϵ_f along all of its outgoing links. Also, each intermediate node equally distributes the sum of its incoming traffic along each outgoing link as well. Clearly, if ϵ_f, $\forall f$, are small enough, then the constraint $\sum_{f=1}^{F} x_l^{(f)} \leq C_l$ can be satisfied. The initial values of dual variables can be chosen arbitrarily since they are unrestricted (e.g., a simple choice is to set $\widetilde{w}_n^{(f)} = 1$ if $n \neq \mathrm{Dst}(f)$ and $\widetilde{w}_n^{(f)} = 0$ if $n = \mathrm{Dst}(f)$).

Algorithm 1 Distributed Newton's Method for Solving MRFC

Initialization:

1. Choose appropriate values of s_f and $x_l^{(f)}$, $\forall f$, for each source and link.
2. Choose appropriate values of dual variables $\widetilde{w}_n^{(f)}$, $\forall f$, for each node.

Main Iteration:

3. The primal Newton directions Δs_f and $\Delta x_l^{(f)}$ are updated using (17) and (18) at each source node and link, respectively.
4. The dual variables $\widetilde{w}_n^{(f)}$ are updated using (22) at each node.
5. Stop the algorithm if some predefined run-time limit is reached or if the Newton decrement criterion is satisfied. Otherwise, go to Step 3.

Stopping Criterion: Since the Newton's method enjoys a quadratic convergence rate, a simple stopping rule is to let all sources and links run the algorithm for a fixed amount of time. If the time duration is long enough, then due to the rapid convergence speed, by the time the clock expires, it is with high probability that the algorithm will have converged to a near-optimal solution. Another more accurate termination time can be calculated based on the so-called Newton decrement [7], but at the expense of a larger scale of information exchange across the network. Due to limited space, we refer readers to [5, Section 6.5.3] for more details.

Step-Size Selection: As in the classical Newton's method [6], [7], when the iterates $\{\widetilde{\mathbf{y}}^k\}$ approach a close neighborhood of an optimal solution, a fixed step-size $\pi^k = 1$ gives us a quadratic rate of convergence. If the iterates $\{\widetilde{\mathbf{y}}^k\}$ are far from an optimal solution (which is also referred to as "damped Newton phase" [7]), then inexact line search methods (e.g., the "Armijo rule" [6], also known as "backtracking line search" [7]) or the step-size rule in [13] can be used. Due to the inexactness of these line search methods, the theoretical convergence rate would be sub-quadratic, but still, in theory and practice, superlinear. So, it remains much faster than subgradient-type methods.

To conclude this section, we summarize our distributed Newton's method for the MRFC problem in Algorithm 1.

VI. Numerical Results

In this section, we present numerical results for our proposed distributed Newton's method. First, we examine the convergence speed of the parameterized matrix splitting scheme. We use a 10-node 3-session network as an example. The initial value of $\widetilde{w}_n^{(f)}$ is set to 1 if $n \neq \mathrm{Dst}(f)$ and 0 otherwise. We vary α from 0.1 to 1. The iterative scheme is stopped when the error with respect to the true value $\mathbf{w}^*$ (measured by $\|\mathbf{w}^k - \mathbf{w}^*\|_2$) is less than 10^{-6}. The decrease of error is displayed in Fig. 3 (in log scale). For all values of α, the error decreases exponentially fast. As predicted by Proposition 9, the smaller the value of α, the faster the error decreases. For example, when $\alpha = 0.1$, the number of iterations is approximately half of that when $\alpha = 1$ (115 vs. 232). To exhibit the convergence behavior of our distributed Newton's method, we use a network as shown in Fig. 4, where five nodes are distributed in a square region of $800m \times 800m$.

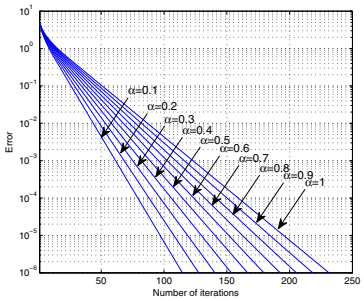

Fig. 3. The error between the true solution of $\mathbf{w}^k$ in Eq. (15) and the matrix-splitting scheme.

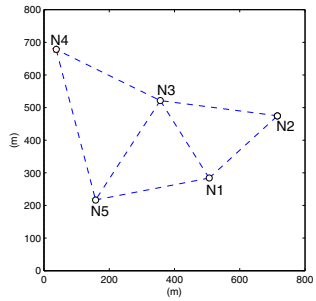

Fig. 4. A five-node two-session network.

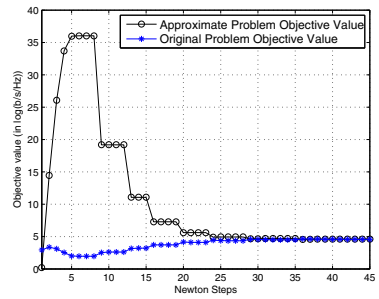

Fig. 5. Convergence behavior for the network shown in Fig. 4.

There are two sessions in the network: N4 to N1 and N5 to N3. We adopt $\log(s_f)$ as our utility function, which represents the proportional fairness [2]. The convergence behavior is illustrated in Fig. 5, which shows both the objective values of the approximating and the original problems. It can be seen that our proposed algorithm only takes 45 Newton steps to converge. To compare our algorithm with the subgradient method, we randomly generate 50 networks with 30 nodes and six sessions. For these 50 examples, the mean numbers of iterations for our method and the subgradient method are 779.3 and 61115.26, respectively, which shows that our algorithm converges almost two orders of magnitude faster.

VII. Conclusion

In this paper, we developed a novel and fully distributed Newton's method for joint multi-path routing and flow control (MRFC). We showed that, by appropriately exploiting the special problem structure, both the primal Newton direction and dual variable updates can be computed without the need for global information. Our proposed algorithm requires almost the same scale of local information exchange as in first-order methods, while achieving a quadratic convergence rate as in centralized Newton methods. We conducted extensive numerical studies to demonstrate the efficacy of our algorithm. Our work offers important and interesting "back-pressure" insights and contributes to the advanced paradigm shift in cross-layer network design that is evolving from first-order to second-order methods. Cross-layer design with distributed second-order methods is an important and yet under-explored area. Future research in this area may include to incorporate possibly nonconvex link layer models from wireless networks, to analyze the impact of inexact line search on convergence, and to consider stochastic traffic models.

References

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

[2] F. P. Kelly, A. K. Malullo, and D. K. H. Tan, "Rate control in communications networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, 1998.

[3] S. Low and R. Srikant, "A mathematical framework for designing a low-loss low-delay internet," *Network and Spatial Economics*, vol. 4, no. 1, pp. 75–101, 2004.

[4] L. Tassiulas and A. Ephremides, "Stability properties of constrained queuing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 2, pp. 466–478, Mar. 1993.

[5] J. Liu and H. D. Sherali, "A distributed Newton's method for joint multi-hop routing and flow control: Theory and algorithm," *Technical Report, Dept. of ECE, Ohio State University*, Jul. 2011. [Online]. Available: http://www2.ece.ohio-state.edu/~liu/publications/DNewton.pdf

[6] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. New York, NY: John Wiley & Sons Inc., 2006.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, UK: Cambridge University Press, 2004.

[8] D. Bickson, "Gaussian belief propagation: Theory and application," Ph.D. dissertation, Hebrew University of Jerusalem, 2009.

[9] Z. I. Woznicki, "Matrix splitting principles," *International Journal of Mathematics and Mathematical Sciences*, vol. 28, no. 5, pp. 251–284, May 2001.

[10] D. Bickson, Y. Tock, O. Shental, and D. Dolev, "Polynomial linear programming with Gaussian belief propagation," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 23-26, 2008, pp. 895–901.

[11] D. Bickson, Y. Tock, A. Zymnis, S. Boyd, and D. Dolev, "Distributed large scale network utility maximization," in *Proc. IEEE International Symposium on Information Theory (ISIT)*, Seoul, Korea, Jun.28–Jul.3, 2009, pp. 829–833.

[12] A. Jadbabaie, A. Ozdaglar, and M. Zargham, "A distirbuted Newton method for network optimization," in *Proc. IEEE Conference on Decision and Control (CDC)*, Shanghai, China, Dec. 16-18, 2009.

[13] E. Wei, A. Ozdaglar, and A. Jadbabaie, "A distirbuted Newton method for network utitlity maximization," in *Proc. IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, Dec. 15-17, 2010.

[14] D. P. Bertsekas and E. M. Gafni, "Projected Newton methods and optimization of multi-commodity flows," *IEEE Trans. Autom. Control*, vol. 28, no. 12, pp. 1090–1096, Dec. 1983.

[15] J. G. Klincewicz, "A Newton method for convex separable network flow problems," *Networks*, vol. 13, no. 3, pp. 427–442, Mar. 1983.

[16] S. Bolognani and S. Zampieri, "Distirbuted quasi-Newton method and its applications to the optimal reactive power flow problem," in *Proc. 2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Annecy, France, Sep. 13-14, 2010, pp. 305–310.

[17] A. Zymnis, N. Trichakis, S. Boyd, and D. ONeill, "An interior-point method for large scale network utility maximization," in *Proc. Allerton Conference on Communication, Control, and Computing*, Monticello, IL, Sep. 26-28, 2007.

[18] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*, 3rd ed. Philadelphia, PA: SIAM, 2001.

[19] F. R. K. Chung, *Spectral Graph Theory*. Providence, RI: American Mathematical Society, 1994.

[20] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali, *Linear Programming and Network Flows*, 4th ed. New York: John Wiley & Sons Inc., 2010.

[21] R. A. Horn and C. R. Johnson, *Matrix Analysis*. New York, NY: Cambridge University Press, 1990.

[22] R. Cottle, J. Pang, and R. Stone, *The Linear Complementarity Problems*. San Diego, CA: Academic Press, 1992.