Optimal Zero-Queue Congestion Control using ADMM

Nikolai Matni

Abstract— **We propose an alternating direction method of multipliers (ADMM) based solution to a network utility maximization (NUM) problem, and show that it leads to a protocol for congestion control that converges rapidly to utility maximizing transmission rates while guaranteeing zero congestion throughout the network. Our approach hinges on a novel decomposition of the NUM problem that leads to easily solvable iterate update subproblems – we provide closed form solutions for these subproblems for proportional fairness and minimum delay fairness utility functions. We further show that this decomposition lends itself to a distributed implementation that naturally allows for the generation of a set of feasible transmission rates at each iteration of the algorithm, thus ensuring that queues remain empty throughout the network. We formalize these notions in the form of a congestion control protocol, and comment on their usefulness and implementation in the context of recent developments in software defined networking. Finally we compare the performance of our approach to state-of-the-art algorithms from the networking and optimization communities via a datacenter network simulation.**

I. Introduction

A long standing challenge in packet-switched networks is the rapid convergence of resource allocations. For instance, in the context of congestion control, a complicating factor is that decisions and computation are distributed among endpoints which then vary their transmission rates based on packet-by-packet feedback from the network (e.g., by measuring loss, average queue length or round trip time). This approach to congestion control was given a theoretical footing via the Network Utility Maximization (NUM) framework [1], [2], which interpreted these protocols as a primal, dual, or primal-dual gradient ascent algorithm which converges to optimal transmission rates which respect the capacity constraints of the network. However, inherent to this framework is that capacity constraints must be violated in order to provide feedback to end-hosts.

With the introduction of software defined networking [3], which allows for centralized programmatic control of networks, and software defined switches [4], which allow for in-network components to perform non-trivial computation, the end-point only approach to congestion control is no longer necessary. In the context of datacenter networks (DCNs), where physical components are co-located, the end-to-end principle that proved so powerful in wide are networks (WANs) seems particularly unnecessary and dated.

Inspired by these developments and observations, as well as recently proposed "zero-queue" centralized control schemes for DCNs [5], [6], we propose a zero-queue congestion control algorithm based on the Alternating Direction Method of Multipliers (ADMM) [7]. As we discuss in Section II-C, ADMM based algorithms are particularly well suited for congestion control tasks, and as we show in Section IV, the protocol that we develop is as equally well suited to distributed implementations in SDN enabled WANs as it is in DCNs with a centralized controller utilizing multicore processors.

N. Matni is with the Department of Control and Dynamical Systems, California Institute of Technology, Pasadena, CA. nmatni@caltech.edu.

A. Contributions

We develop a distributed ADMM based algorithm for congestion control, and show that it converges rapidly to utility maximizing transmission rates, and guarantees zero congestion throughout the network. We we provide closed form solutions to the ADMM iterate update subproblems for proportional fairness and minimum delay fairness utility functions, thus allowing for a computationally efficient implementation of the proposed scheme. Further, we show that the algorithm admits a distributed implementation, making it well suited to WANs, and that it further allows us to generate a set of feasible transmission rates at each iteration of the algorithm, thus ensuring that queues remain empty throughout the network. We formalize these notions in the form of a congestion control protocol, and comment on this protocol's usefulness and implementation in the context of recent developments in software defined networking.

B. Comparison to prior work

Augmented Lagrangian inspired congestion control algorithms are not unique to this work. For instance, in [8], the authors introduce extra dynamics to algorithms resulting from traditional primal-dual methods to improve their performance and robustness to in-network delays. The algorithm developed in this paper is most similar in spirit to the solution described in [9], where the authors define the Cluster-ADMM algorithm, and apply it to solving a NUM problem. Whereas the authors of [9] suggest applying the Cluster-ADMM algorithm to the the dual of the NUM problem, we directly tackle the primal problem, which as we later show, is essential to the zero-queue distributed protocol that we develop.

C. Paper organization

We introduce and define notation in Section II, as well as provide an overview of the NUM framework and ADMM. In Section III, we describe the proposed decomposition of the NUM problem, and provide closed form solutions to the iterate update subproblems when utility functions are

chosen to achieve proportional fairness or minimum delay fairness. We further provide guarantees of convergence of the algorithm, and propose a simple heuristic for generating a set of feasible transmission rates at each iteration of the ADMM algorithm. In Section IV, we discuss how to implement the proposed algorithm as a distributed congestion control protocol in SDN enabled WANs, and comment on how one would adapt this protocol to a centralized controller utilizing multicore processors, such as those used in SDN enabled DCNs. In Section V, we compare the transient behavior of our algorithm to state of the art approaches from the optimization and networking communities [1], [2], [6], [10], [11] via a DCN inspired topology in which sources enter and leave the network stochastically. We end with conclusions in Section VI.

II. Preliminaries

A. Notation & preliminaries

We consider a network composed of S source-destination pairs (s, d) connected via a directed graph. We refer to the edges of this graph as links ℓ and call intermediary nodes (i.e., nodes that are not sources or destinations) routers. We summarize the notation used for other parameters of the problem in Table I.

TABLE I: Notation used to specify NUM problems

Parameter	Notation
source	s
destination	d
link	ℓ
capacity of link ℓ	c_ℓ
tx rate at source s	r_s
utility function at source s	$U_s(r_s)$
set of sources using link ℓ	$S(\ell)$
set of links used by source s	$L(s)$

B. The Network Utility Maximization Framework

The following is modified from [1], [2]. The NUM framework is a convex optimization based approach to solving network resource allocation problems. Many network resource allocation problems can be formulated as the optimization of some utility function subject to constraints imposed by the capacity of the network. In the context of congestion control, which is the main focus of this paper, one considers the optimization problem

$$\begin{array}{ll} \text{maximize}_{r\geq 0} & \sum_{s=1}^{S} U_s(r_s) \\ \text{s.t.} & \mathbf{1}^\top r_{S(\ell)} \leq c_\ell, \ \forall \ell, \end{array} \tag{1}$$

where we recall that U_s is a (strictly) concave utility function, r_s is the rate of source s, $S(\ell)$ is the set of sources using link ℓ, and c_ℓ denotes the capacity of link ℓ.

A typical choice for the utility function U_s is the weighted α-fairness utility function given by

$$U_s^\alpha(r_s) = w_s(1-\alpha)^{-1} r_s^{1-\alpha}, \tag{2}$$

where $w_s > 0$ is a weight, and $\alpha \in [1, \infty]$ is the fairness parameter. Three values of α are of particular interest, as they recover well studied notions of fairness in networks [12]:

- $\alpha \to 1$: in this case, it can be shown that $U_s^1(r_s) = w_s \log(r_s)$, which leads to solutions that satisfy *proportional fairness*;
- $\alpha = 2$: in this case, $U_s^2(r_s) = -w_S/r_s$, which leads to solutions that satisfy *minimum delay fairness*;
- $\alpha \to \infty$: choosing a large (infinite) value of α leads to solutions that approximately (exactly) satisfy *max-min fairness*.

A driving motivation behind the NUM framework is that network structure often leads to decomposability of the resource allocation problem, allowing for distributed (and often iterative) algorithms to be used that provably converge to a globally optimal network state. For instance it was shown that existing variants of TCP can be viewed as implementing a distributed gradient ascent algorithm for solving the NUM problem (1), with various network metrics (delay, queue length, packet drops) playing the role of lagrange multipliers. A feature of this approach is that capacity constraints must be relaxed in order to implement the distributed algorithm – hence congestion can, and in fact must, occur (i.e., capacity constraints must be violated) while the algorithm is converging to the optimal value. While this was a necessary evil at the time due to the rigidity of network architectures, with the recent introduction of SDN, more sophisticated control algorithms can be implemented: as we show in this paper, by constructing a congestion control protocol around an ADMM solution to the NUM problem (1), one preserves the decomposability needed to scale such solutions to large-scale systems and gains additional features such as improved convergence properties and congestion free convergence.

C. Alternating Direction Method of Multipliers

The following is adapted from [7]. ADMM is a "meta"-optimization scheme, where each step is carried out by solving a convex optimization problem. Consider the optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array} \tag{3}$$

over the variables x and z and convex functions f and g. Define an augmented Lagrangian

$$L_\rho = f(x) + g(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2,$$

where $\rho > 0$ is an algorithm parameter, and λ is the lagrange multiplier associated with the equality constraint $Ax + Bz = c$. The constrained optimization problem is solved through alternately minimizing the augmented Lagrangian over the primal variables x, z, and updating the dual variable λ,

$$\begin{array}{lcl} x^{k+1} & := & \text{argmin}_x L_\rho(x, z^k, \lambda^k) \\ z^{k+1} & := & \text{argmin}_z L_\rho(x^{k+1}, z, \lambda^k) \\ \lambda^{k+1} & := & \lambda^k + \rho\left(Ax^{k+1} + Bz^{k+1} - c\right). \end{array} \tag{4}$$

Suppose that optimization problem (3) satisfies the following two assumptions:

Assumption 1: The (extended real valued) functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, proper, and convex.

Assumption 2: The unaugmented Lagrangian has a saddle point.

Then the following general theorem applies.

Theorem 1 (§3.2.1 of [7]): Let $p_\star$ denote the optimal value of optimization problem (3). Given Assumptions 1, 2 then the ADMM iterates satisfy the following:

- **Residual convergence**: $Ax^k + Bz^k - c \to 0$ as $k \to \infty$, i.e. the iterates approach feasibility;
- **Objective convergence**: $f(x^k) + g(z^k) \to p_\star$ as $k \to \infty$, i.e. the objective function of the iterates converges to the optimal value;
- **Dual variable convergence**: $\lambda^k \to \lambda_\star$ as $t \to \infty$, where $y_\star$ is a dual optimal point.

Why use ADMM for NUM?: In addition to the strong convergence guarantees that ADMM provides, it has other desirable properties when used as a congestion control algorithm. As described in [7], ADMM blends the decomposability of dual ascent with the superior convergence of the method of multipliers. Further, in practice, the ADMM algorithm is often seen to converge to modest accuracy within a few tens of iterations.

Decomposability is an obvious necessary property for applying such techniques to congestion control. Further, rapid approximate optimality is precisely the behavior needed in an environment where sources enter and leave the network frequently – in particular, ADMM is particularly well suited for dynamic environments where obtaining good solutions quickly is more important than obtaining optimal solutions slowly. Finally, as we show in §III, the approach that we propose allows for a set feasible source rates to be rapidly generated at each iteration of the algorithm, even in a distributed setting – this in turns allows us to guarantee zero congestion in the network, which is a desirable property in DCNs, or when real-time applications (such as video conferencing) are being run across WANs.

Finally, from an implementation perspective, the introduction of software defined networks (SDNs) [3], which allow for centralized real-time programmatic control of a network (e.g., [5], [6]), as well as software defined switches (SDSs), such as those that are OpenFlow enable [4], make using such a protocol a practically meaningful proposition (c.f., Section IV).

III. Using ADMM to Solve a NUM Problem

We begin by rewriting the NUM problem (1) in the following ADMM amenable form:

$$\begin{array}{rl} \text{maximize}_{x\geq 0,\{z_\ell\}} & \sum_{s=1}^{S} U_s(x_s) \\ \text{s.t.} & \mathbf{1}^\top z_\ell \leq c_\ell,\ \forall \ell, \\ & x_{S(\ell)} = z_\ell, \forall \ell. \end{array} \tag{5}$$

Note that we no longer use r to denote the rate vector, as we wish to reserve this variable for the actual transmission rates applied at the sources. As we shortly show, both the x and z variables consist of internal variables to the ADMM algorithm that need to be suitably manipulated to generate a set of feasible source rates at each iteration of the protocol.

Thus the iterate update subproblems become:[1]

$$x_s^{k+1} = \arg\max_{x_s \geq 0} U_s(x_s) + x_s\Big(\sum_{\ell \in L(s)} \lambda_{\ell,s}^k\Big) - \frac{\rho}{2}\sum_{\ell \in L(s)} (x_s - z_{\ell,s})^2 \tag{6a}$$

$$z_\ell^{k+1} = \arg\max_{\{z_\ell : \mathbf{1}^\top z_\ell \leq c_\ell\}} -(\lambda_\ell^k)^\top z_\ell - \frac{\rho}{2}\|x_{S(\ell)}^{k+1} - z_\ell\|_2^2 \tag{6b}$$

$$\lambda_\ell^{k+1} = \lambda_\ell^k - \rho(x_{S(\ell)}^{k+1} - z_\ell^{k+1}). \tag{6c}$$

Rewriting the problem as in (5) allows for an intuitive interpretation of the update subproblems and their corresponding iterates x_s, z_ℓ and λ_ℓ. The x variables are selected to optimize performance, as is made clear by the objective function composed of the utility function $U_s(x_s)$ augmented with additional terms from the Lagrangian. In contrast, the z_ℓ variables are selected to ensure local feasibility on link ℓ, as is made clear by the constraint $\mathbf{1}^\top z_\ell \leq c_\ell$. Finally the Lagrange multipliers $\{\lambda_\ell\}$ can be interpreted as the price of disagreement between the performance variables x and the feasibility variables $\{z_\ell\}$, and hence appear as correction factors in the objective functions of the x and $\{z_\ell\}$ update subproblems to ensure that at optimality these values coincide. Note that the interpretation of the multipliers $\{\lambda_\ell\}$ is different than that found in the traditional NUM literature [1], [2] and in [9], where they are interpreted as the cost of using a link.

A. Convergence properties

Any reasonable choice of utility function $U_s(x_s)$, such as α-fair utilities, leads to the NUM problem (1) satisfying Assumption 1. Further it is easily seen that Slater's conditions are satisfied for the NUM problem (1), and hence Assumption 2 is trivially satisfied as well. Thus all of the guarantees of Theorem 1 – namely residual, objective and dual variable convergence – hold.

Finally, we note that if $U_s(x_s)$ is strictly concave over the feasible set of problem (1) then the NUM problem has a unique optimal solution. If we further assume them to be continuous (both properties hold for α-fair utilities), then Theorem 1 implies that the primal iterates $\{x_s^k\}$ and $\{z_s^k\}$ converge to their optimal values.

B. Closed form solutions to update subproblems

1) Performance variable x_s update: Here we show that for the case of α-fair utility functions, with $\alpha \geq 1$ an integer, the x_s iterate update (6a) can be computed by finding the real positive roots of a polynomial equation, should they exist. We further show that in the case of $\alpha \to 1$

[1]Note that the constrained subproblems (6) can be converted to unconstrained ones of the form (4) by adding a suitable indicator function to their objectives.

(proportional fairness) and $\alpha = 2$ (minimum delay fairness), the iterate update admits a closed form solution, as in this case the corresponding polynomial is a quadratic and a cubic, respectively, which can both be shown to always have a real positive root.

Lemma 1: Assume that $U_s(x_s) = U_s^\alpha(x_s)$ is an α-fair utility function, with $\alpha \geq 1$ an integer. If the polynomial equation

$$x_s^{\alpha+1} - \frac{1}{|L(s)|}\left(\sum_{\ell \in L(s)} z_{\ell,s}^k + \frac{1}{\rho}\lambda_{\ell,s}^k\right) x_s^\alpha - \frac{w_s}{\rho|L(s)|} = 0, \tag{7}$$

has at least one real positive root σ, then it is the only such real positive root, and $x_s^{k+1} = \sigma$.

Proof: We first note that for $\alpha \geq 1$, a value of $x_s^{k+1} = 0$ leads to an objective value of $-\infty$ for the update problem (6a), hence we can restrict our attention to positive values $x_s^{k+1} > 0$. We drop the constraint $x_s \geq 0$, and differentiate the objective function of update problem (6a) with respect to x_s, and multiply through by $\frac{x_s^\alpha}{\rho|L(s)|}$ to obtain the polynomial (7).

Assume now that polynomial (7) has at least one real positive root σ, and let $\mathcal{R}$ denote the set of such roots. Seeking a contradiction, assume that $|\mathcal{R}| > 1$, and notice that any root drawn from this set can be shown to satisfy the optimality conditions of subproblem (6a). However, the objective function of subproblem (6a) is strongly concave over its domain, and hence this subproblem has a unique optimal solution, leading to a contradiction. Hence if the set $\mathcal{R}$ is non-empty, it must be a singleton given by $\{\sigma\}$, from which the claim follows. ∎

For the cases of $\alpha \to 1$ and $\alpha = 2$, the following corollaries are immediate.

Corollary 1 (Proportional fairness update): For $U_s(x_s) = w_s \log(x_s)$, the solution to the x-update subproblem (6a) is given by:

$$x_s^{k+1} = \frac{1}{2}\left(\sqrt{(b_s^k)^2 + 4\frac{w_s}{\rho|L(s)|}} - b_s^k\right), \tag{8}$$

for

$$b_s^k := \frac{1}{|L(s)|}\sum_{\ell \in L(s)}\left[z_{\ell,s}^k - \frac{1}{\rho}\lambda_{\ell,s}^k\right].$$

Proof: Substituting $\alpha = 1$ into polynomial (7), we obtain a quadratic polynomial with negative constant term $-\frac{w_s}{\rho|L(s)|}$. Hence we know that both roots are real, with one positive and one negative: the update is then obtained by applying the quadratic formula. ∎

Corollary 2 (Minimum-delay fairness update): For $U_s(x_s) = -\frac{w_s}{x_s}$, the solution to the x-update subproblem is given by the unique positive root to a cubic polynomial, which can be computed in closed form using standard methods.[2]

[2]We omit the explicit closed form expression as it is standard but cumbersome to write.

Proof: Substituting $\alpha = 2$ into polynomial (7), we obtain a cubic polynomial with negative constant term $-\frac{w_s}{\rho|L(s)|}$. Hence we know that at least one root is real and positive (follows from the fact that the leading coefficient is 1, the constant term is negative, and applying the intermediate value theorem); from Lemma 1 we conclude that it is unique. The update is then obtained by applying the cubic formula to determine this positive real root. ∎

2) Feasibility variable z_ℓ update: The z_ℓ-update subproblem (6b) is independent of the objective function, and as we show below, admits a closed form solution.

Lemma 2: The solution to the z_ℓ-update subproblem (6b) is given by:

$$z_\ell^{k+1} = x_{S(\ell)} - \frac{1}{\rho}\left(p_\ell^{k+1}\mathbf{1} + \lambda_\ell^k\right), \tag{9}$$

where

$$p_\ell^{k+1} := \frac{\rho}{|S(\ell)|}\left[\mathbf{1}^\top\left(x_{S(\ell)}^{k+1}\frac{1}{\rho}\lambda_\ell^k\right) - c_\ell\right]_+. \tag{10}$$

Proof: We write the Lagrangian for the z_ℓ-update subproblem (6b):

$$L(z_\ell, p_\ell^{k+1}) = -(\lambda_\ell^k)^\top z_\ell - \frac{\rho}{2}\|x_{S(\ell)}^{k+1} - z_\ell\|_2^2 + p_\ell^{k+1}\left(\mathbf{1}^\top z_\ell - c_\ell\right) \tag{11}$$

for $p_\ell^{k+1} \geq 0$ the Lagrange multiplier corresponding to the capacity constraint $\mathbf{1}^\top z_\ell \leq c_\ell$.

Solving $\nabla_{z_\ell} L(z_\ell, p_\ell^{k+1}) = 0$ for z_ℓ yields the update expression (9). Substituting this expression into the capacity constraint $\mathbf{1}^\top z_\ell \leq c_\ell$, we see that it is satisfied if and only if

$$p_\ell^{k+1} \geq \frac{\rho}{|S(\ell)|}\left[\mathbf{1}^\top\left(x_{S(\ell)}^{k+1}\frac{1}{\rho}\lambda_\ell^k\right) - c_\ell\right]. \tag{12}$$

It can further be checked via direct substitution of (9) into the objective function of the z_ℓ-update subproblem (6b) that one should select $p_\ell^{k+1} \geq 0$ as small as possible. This fact combined with inequality (12) yields the expression (10). ∎

C. Feasible set of transmission rates at each iteration

We now show that the chosen problem decomposition (5) naturally allows for a feasible set of transmission rates $\{r_s\}$ to be generated at each iteration of the algorithm. This is particularly appealing in both datacenter networks (DCN) and real-time applications (such as video streaming), as it guarantees that all queues within the network are empty under normal operating conditions (i.e., barring any faults within the network).

In particular, for each link ℓ at iteration k, we define the modified locally feasible variable $\tilde{z}_\ell^k$ as follows:

$$\tilde{z}_\ell^k := \begin{cases} z_\ell^k & \text{if } z_\ell^k \geq 0 \\ \frac{c_\ell}{|S(\ell)|}\mathbf{1} & \text{otherwise} \end{cases} \tag{13}$$

We require this auxiliary variable because we cannot guarantee that the variable z_ℓ^k is non-negative at each iteration k. Note that when z_ℓ^k is not non-negative, any feasible value

$\tilde{z}_\ell$ can be chosen – that proposed in (13) is just one such choice.

We then have the following useful property:

Lemma 3: At iteration k, define the transmission rate r_s at source s to be

$$r_s^k := \min_{\ell \in L(s)} \tilde{z}_{\ell,s}^k. \tag{14}$$

We then have that $r^k = (r_s^k)_{s=1}^S$ defines a set of transmission rates that are a feasible solution to the NUM optimization problem (1).

Proof: We first note that $r^k \geq 0$ by construction. Thus it suffices to show that for any link ℓ, it holds that $\mathbf{1}^\top r_{S(l)}^k \leq c_\ell$. This follows immediately from the definition of r_s^k given in (14), which implies that $\mathbf{1}^\top r_{S(l)}^k \leq \mathbf{1}^\top \tilde{z}_\ell^k \leq c_\ell$. ∎

IV. Optimal Zero-Queue Congestion Control

In this section we propose a distributed congestion control algorithm, based on the ADMM approach developed in the previous Section, that satisfies the following properties:

1) The source transmission rates $\{r_s\}$ converge to the optimal solution to the NUM problem (1);
2) All iterates $\{r_s^k\}$ define a set of feasible transmission rates – hence the congestion control algorithm leads to zero-queuing in the network.

This section assumes familiarity with the basics of TCP implementation at the packet-level.

We assume the following distribution of computation throughout the network: each source s is responsible for computing its x_s iterate, and each router is responsible for computing the z_ℓ and λ_ℓ iterates for all of its outgoing links ℓ. We outline below components of our distributed protocol, and summarize the discussion in pseudo-code – in all of the following we assume that an initial SYN, SYN/ACK, ACK handshake has been performed between each source-destination pair.

Updating x_s iterates at source s: In order to compute its x_s^{k+1} update, each source s must solve the polynomial equation (7). In order to do so, it needs access to the sum $\left(\sum_{\ell \in L(s)} z_{\ell,s}^k + \frac{1}{\rho}\lambda_{\ell,s}^k\right)$ – this sum can be computed along the path used by source s by having each router increment a counter included in a control packet transmitted by the source[3] that is then returned to the source by the destination host via an acknowledgement packet containing the computed sum.

Updating z_ℓ and $\tilde{z}_\ell$ iterates at router: In order to compute its z_ℓ^{k+1} update, each router must evaluate expressions (9) and (10). To do so, each router must maintain variables z_ℓ^k and λ_ℓ^k for each of its outgoing links, as well as a local copy of $x_{S(\ell)}^{k+1}$ for each outgoing link ℓ, that is a vector of the x_s values for each source s using one of its outgoing links ℓ. Therefore we propose that each source s include its iterate update x_s^{k+1} in a field of the control packets that it transmits, allowing the corresponding local record $x_{S(\ell)}^{k+1}$ to be populated and/or updated . Once z_ℓ^{k+1} has been computed, it is trivial to update $\tilde{z}_\ell^{k+1}$ according to equation (13).

Updating λ_ℓ iterates at router: In order to compute its λ_ℓ^{k+1} update, each router must evaluate the update equation (6c). Given the suggested protocol for the z_ℓ update, the necessary elements z_ℓ^{k+1}, $x_{S(\ell)}^{k+1}$ and λ_ℓ^k are locally available.

Updating feasible rate r_s: Recall that r_s^{k+1} is specified by the minimum value of $\tilde{z}_{\ell,s}^{k+1}$ along the path of source s, as specified in expression (14). This minimum can be computed

[3]This can either be included as a field in the header of a data packet, or as a field in a control packet – in the data packet case, this header is left empty unless the ADMM algorithm is beginning its next iteration. In the interest of clarity, we restrict our discussion to the control packet case.

Algorithm 1 Optimal Zero-Queue Congestion Control

initialize: iterate $k = 1$;
values $\{r_s^1 = 0\}$, $\{z_\ell^1 = 0\}$, $\{\lambda_\ell^1 = 0\}$, $\{c_s^1 = 0\}$
while ***true*** **do**
  foreach *source* s **do**
    update x_s^{k+1} **and** r_s^{k+1}
      receive ack packet and transmit at feasible rate r_s^k;
      read sum counter $c_s^k = \left(\sum_{\ell \in L(s)} z_{\ell,s}^k + \frac{1}{\rho}\lambda_{\ell,s}^k\right)$;
      use c_s^k to compute solution to x_s-update subproblem (6a), and update x_s^{k+1};
    end
    transmit control packet
      set the following fields in control packet:
      - counter $c_s^{k+1} = 0$;
      - feasible rate $r_s^{k+1} = +\infty$;
      - x_s-field $= x_s^{k+1}$;

      and transmit to destination d;
    end
  end
  foreach *link* ℓ **do**
    wait until $x_{S(l)}^{k+1}$ received;
    update z_ℓ^{k+1}**,** $\tilde{z}_\ell^{k+1}$ **and** λ_ℓ^{k+1}
      update z_ℓ^{k+1} and $\tilde{z}_\ell^{k+1}$ using expressions (9), (10) and (13);
      update λ_ℓ^{k+1} according to expression (6c);
    end
    update packet fields
      foreach *source* $s \in S(\ell)$ **do**
        increment counter $c_s^{k+1} = c_s^{k+1} + z_{\ell,s}^{k+1} + \frac{1}{\rho}\lambda_{\ell,s}^{k+1}$;
        set $r_s^{k+1} = \min\left[r_s^{k+1}, \tilde{z}_{\ell,s}^{k+1}\right]$;
      end
    end
  end
  foreach *destination* d **do**
    wait to receive control packet from source s;
    transmit ack packet containing counter c_s^{k+1} and feasible source rate r_s^{k+1};
  end
  increment $k = k + 1$;
end

by including a field in the control packets transmitted by each source s, with an initial value of infinity, and updating it at each hop by taking the minimum of the value of this field and the local feasible copy $z_{\ell,s}$. The final feasible rate update r_s^{k+1} value is then returned by the destination via the aforementioned acknowledgement packet.

Algorithm 1 presents pseudo-code for one iteration of the proposed protocol.

A. Asynchrony and centralized implementations

We note that implicit in the above algorithm is that all source rates can synchronize their transmission rate updates, as otherwise, we cannot guarantee the desired zero-queue property. Whereas such synchronization in source rate updates can be challenging in a distributed setting, it is not unreasonable in a SDN setting. In particular, consider an architecture wherein the x_s and r_s updates are computed by a centralized controller and then transmitted to sources – exactly such an architecture has been successfully applied in the context of datacenter networks [5], [6]. Furthermore, the distributed implementation discussed above also suggests a natural parallelization scheme for a centralized implementation on a multi-core CPU or GPU. Finally, if we are willing to relax the zero-queue property, it will be interesting to explore the application of asynchronous ADMM algorithms (e.g., those described in [13]–[15]) in this context.

V. A Datacenter Network Example

Setup: We follow the simulation setup described in [6]. The topology is a two-tier full-bisection topology with 4 spine switches connected to 9 racks of 16 servers each, where server are connected with a 10 Gbits/s link. To model micro-bursts, flows follow a Poisson arrival process, with job sizes distributed according to the Web workloads published by Facebook [16]. The Poisson rate at which flows enter the system is chosen to reach a 70% average server load, where 100% load is when the rate equals server link capacity divided by the mean flow size. Sources and destinations are chosen uniformly at random, and we assume that there are initially 512 flows in the network. We pick a proportional fairness utility function with uniform weight, i.e., $U_s(r_s) = \log r_s$ for all sources s. We use a constant value of $\rho = .1$ throughout.

Results: Illustrated in Figure 1 are results from a typical trace generated by the algorithm. We compare the ADMM based algorithm to Gradient Ascent (Grad) [1], [2], Fast Weighted Gradient Method (FGM) [10] and Netwon Exact Diagonal [6], which is a modification of the Netwon-like method proposed in [11].

Figures 1a) and 1b) show the number of congested links and the maximum ratio r_s/c_ℓ of transmission rate over link capacity achieved by each algorithm, respectively As expected, the ADMM algorithm has zero congested links throughout, and interestingly, always fully utilizes the network in the sense that there always exists a link ℓ such that $\sum_{s\in S(\ell)} r_s/c_\ell = 1$.

In order to fairly compare the transient performance of these algorithms, we adopt the methodology proposed in [6], wherein at each iteration the set of rates specified by an algorithm is renormalized to be feasible, while ensuring that there exists at least one link ℓ such that $\sum_{s\in S(\ell)} r_s/c_\ell = 1$. We note that in this case, the transmission rates generated by the ADMM algorithm are not renormalized because they are by construction feasible. In Figures 1c) and 1d) we compare the utility and total throughput achieved by the renormalized outputs of each algorithm (note that we omit Grad from Fig. 1c) because it is significantly outperformed by the other algorithms). As can be seen, the ADMM algorithm leads to significantly higher proportional fairness while simultaneously maintaining a higher total throughput.

VI. Conclusion

This paper proposed a protocol for congestion control based on ADMM, and showed that it converges rapidly to utility maximizing transmission rates while guaranteeing zero congestion throughout the network. Our approach hinges on a novel decomposition of the NUM problem that leads to ADMM subproblems with closed form solutions and that naturally lends itself to a distributed implementation. We described a distributed implementation of this congestion control protocol and showed that it is well suited to both SDN enabled WANs as well as SDN controlled DCNs. Finally we demonstrated via a DCN simulation that the ADMM based algorithm significantly outperforms the state-of-the-art.

References

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

[2] D. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 8, pp. 1439–1451, Aug 2006.

[3] B. A. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1617–1634, Third 2014.

[4] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[5] J. Perry, A. Ousterhout, H. Balakrishnan, D. Shah, and H. Fugal, "Fastpass: A centralized "zero-queue" datacenter network," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 307–318. [Online]. Available: http://doi.acm.org/10.1145/2619239.2626309

[6] J. Perry, H. Balakrishnan, and D. Shah, "Flowtune: Flowlet control for datacenter networks," *MIT CSAIL Technical Report*, 2016.

[7] 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.

[8] X. Zhang and A. Papachristodoulou, "Improving the performance of network congestion control algorithms," *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 522–527, Feb 2015.

[9] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, "Distributed ADMM for model predictive control and congestion control," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, pp. 5110–5115.

[10] A. Beck, A. Nedic, A. Ozdaglar, and M. Teboulle, "An o(1/k) gradient method for network resource allocation problems," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 64–73, March 2014.

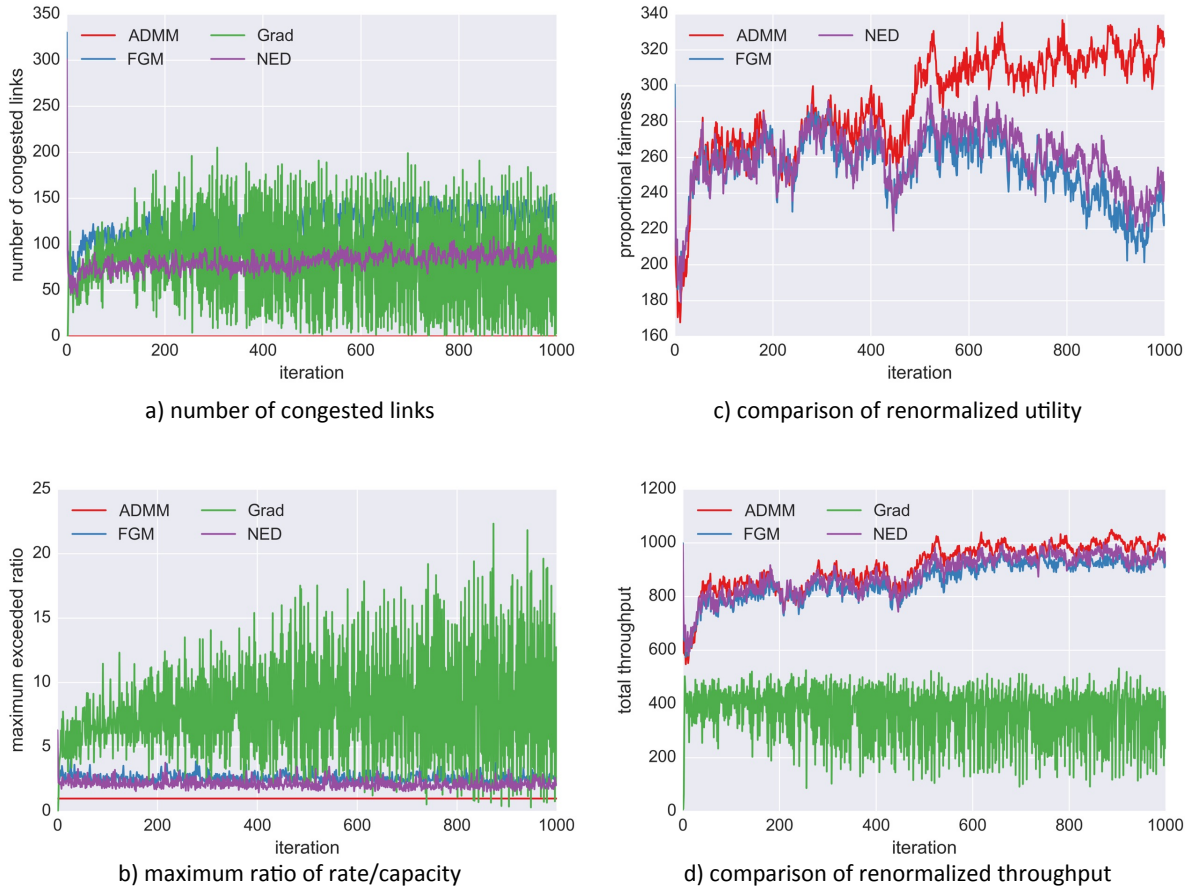

Fig. 1: Results for the simulation example described in Section V. Figures 1a) and 1b) respectively show the number of congested links and the maximum exceeded capacity ratio $\sum_{s \in S(\ell)} {}^{r_s}/_{c_\ell}$, respectively. As can be seen, the ADMM algorithm leads to zero congested links, and full network utilization, i.e., $\max_\ell \sum_{s \in S(\ell)} {}^{r_s}/_{c_\ell = 1}/_{c_\ell} = 1$. Figures 1c) and 1d) compare the utility and total throughput achieved by each algorithm (we omit Grad from Fig. 1c) because it is significantly outperformed by the other algorithms), where all non-ADMM based algorithms have their throughputs renormalized to ensure a feasible solution (c.f., [6] for more details). As can be seen, the ADMM algorithm leads to significantly higher proportional fairness while maintaining a higher total throughput.

[11] S. Athuraliya and S. H. Low, "Optimization flow control with Newton-like algorithm," *Telecommunication Systems*, vol. 15, no. 3, pp. 345–358, 2000. [Online]. Available: http://dx.doi.org/10.1023/A:1019155231293

[12] R. Srikant, *The mathematics of Internet congestion control*. Springer Science & Business Media, 2012.

[13] R. Zhang and J. T. Kwok, "Asynchronous distributed ADMM for consensus optimization." in *ICML*, 2014, pp. 1701–1709.

[14] T. H. Chang, M. Hong, W. C. Liao, and X. Wang, "Asynchronous distributed alternating direction method of multipliers: Algorithm and convergence analysis," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 4781–4785.

[15] E. Wei and A. Ozdaglar, "On the O(1=k) convergence of asynchronous distributed alternating direction method of multipliers," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, Dec 2013, pp. 551–554.

[16] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 123–137, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787472