# Dynamics, Control and Optimization in Layered Network Architectures

***Submitted to the 2016 ACM Symposium on SDN Research (SOSR 2016)***

Nikolai Matni[*], Dimitar Ho[†], Ao Tang[‡], David Meyer[§], John C. Doyle[¶]

**Abstract**

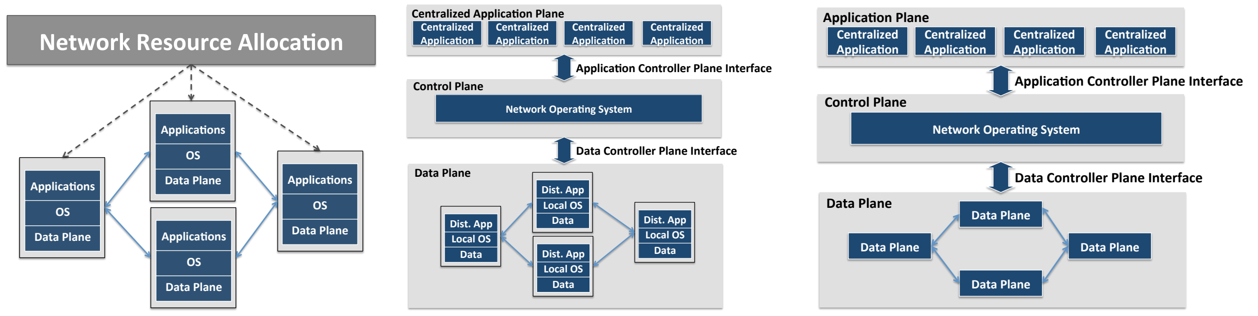
Software defined networking (SDN) allows designers to increase the flexibility, elasticity and responsiveness of network applications by vertically distributing their functionality across the application, control and data planes – however, the sheer amount of options for a given network solution can be overwhelming. In this paper we introduce a new more complete theory for SDN based on Layering as Optimization (LAO) decomposition and Network Utility Maximization (NUM) that explicitly incorporates dynamics and distributed control.

## 1 Introduction

Network control algorithms are often faced with two complementary tasks: (i) finding an optimal operating-point with respect to a functional utility, and (ii) efficiently bringing the state of the network to this optimal operating-point despite model uncertainty, fast-time scale dynamics and information sharing constraints. The Layering as Optimization (LAO) decomposition [1] technique applied to Network Utility Maximization (NUM) [2] problems has proven to be an effective means of addressing task (i) – these techniques focus on network resource allocation problems, and offer a formal approach to designing layered architectures (via vertical decompositions of a NUM problem) and distributed algorithms (via horizontal decompositions of a NUM problem) that guarantee convergence to a functionally optimal network operating-point (e.g., as specified by a congestion control or traffic engineering problem). We note however that the LAO/NUM frameworks are built around static optimization problems that do not explicitly incorporate network dynamics or information sharing constraints. In contrast, we showed in previous work [3] that task (ii) can be addressed using tools from distributed optimal control theory. We also argued that these tools provided a principled means of exploring the effects of different distributed control architectures, ranging from fully myopic (with no coordination between network elements) to fully centralized (cf. Fig 1), on network performance and demonstrated these ideas with an admission control (AC) case study.

In this paper we significantly extend existing results by proposing a method that jointly addresses tasks (i) and (ii) in a unified optimization based framework [4] that can be viewed as a natural extension of the LAO/NUM methodologies to the SDN paradigm. In particular our newly developed theory extends the optimization based frameworks of LAO/NUM to explicitly incorporate fast-time scale network dynamics and distributed information sharing constraints, allowing

[*]N. Matni is with the Department of Control and Dynamical Systems, Caltech, Pasadena, CA.
[†]D. Ho is with the Department of Control and Dynamical Systems, Caltech, Pasadena, CA.
[‡]A. Tang is with the Department of Electrical and Computer Engineering, Cornell, Ithaca, NY.
[§]D. Meyer is with Brocade Communication Systems, Inc., San Jose, CA.
[¶]J. C. Doyle is with the Department of Control and Dynamical Systems, Caltech, Pasadena, CA.



(a) Traditional Architecture (b) Hybrid SDN Architecture (c) Extreme SDN Architecture

Figure 1: The different layered architectures considered for the joint TE/AC problem studied in §3, ranging from traditional (Fig 1a) to extreme SDN (Fig 1c) network structures.

the network designer to fully exploit the flexibility, elasticity and responsiveness introduced by the SDN paradigm. This new flexibility does however lead to new design challenges, such as deciding which aspects of network functionality should be implemented in the application plane (e.g., as in [5, 6, 7]), which components of network structure should be virtualized by the control plane, and which elements of network control should remain in the data plane (e.g., as in [8, 1, 2]). In principle SDN allows for any combination of centralized, virtualized and decentralized functionality to be vertically distributed across the application, control and data planes. This paper introduces quantitative tools, firmly rooted in distributed optimization and control, that aim to inform these architectural decisions.

In this paper we summarize our recently developed extension [4] of the LAO framework to the SDN setting and use it to explore the newly opened up *Hybrid SDN* design space. In particular, we show that the increased responsiveness in network control afforded by SDN allows for joint network resource allocation and disturbance rejection problems to be posed and solved using novel network control architectures. We show how both traditional and centralized (or as we label them, extreme) SDN architectures can be recovered as special cases of our framework, and also define novel hybrid SDN architectures that divide network functionality between the application, control and data plane using vertical decompositions that are akin to those used in the LAO/NUM framework. We illustrate the usefulness of our framework by applying it to a joint traffic engineering (TE) and AC problem, for which we define and explore the corresponding HySDN design space.

This paper is organized as follows: in §2 we recall the LAO/NUM and optimal control frameworks, and show how they can be naturally combined into a framework that leverages the use of dynamics, control and optimization in layered architectures. In §3, we specialize the discussion to a joint traffic engineering (TE) and admission control (AC) problem, and define the corresponding traditional, extreme SDN and hybrid SDN architectures to be studied. We present a simple case study in §4 and demonstrate that the interplay between network dynamics and computational delay can significantly affect performance. We conclude with a summary of our results and directions for future work in §5.

Just as in our previous paper [3], we emphasize that we are not arguing that a specific implementation, algorithm or architecture is best suited for a specific application – rather, we are proposing a methodology that network designers can use to explore the HySDN architectural design space efficiently.

## 2 LAO/NUM, Optimal Control & SDN

In this section we first recall (i) the LAO decomposition philosophy as applied to NUM problems and (ii) the distributed optimal control paradigm as applied to disturbance rejection problems. As argued in the introduction, the LAO/NUM paradigm should be viewed as a principled approach to designing layered network architectures that optimally allocate resources within the network – i.e., LAO/NUM yields a network resource "planner" – whereas the distributed control paradigm should be viewed as a network "controller" that ensures that the network state remains close to a value specified by the resource planner. Traditionally the network resource allocation and network control tasks were either (i) vertically combined in the data plane (e.g., [2, 9]) due to limitations in network controller architecture flexibility (in the pre-SDN era), or (ii) independently designed and implemented. We argue that in some cases, these extremes in layered architecture can lead to degradations in performance. Fortunately we are no longer constrained to these extremes thanks to the flexibility and elasticity afforded by SDN. We conclude this section with a summary of our newly developed theory [4], which allows for network resource allocation and network control to be analyzed and synthesized in a unified framework, and argue that this approach is a natural generalization of the NUM framework to incorporate the flexibility of SDN controllers. In §4, we build on our previous results [3] and study a joint TE/AC problem using our new framework.

### 2.1 Layering as Optimization Decomposition

The following is modified from [1]. The LAO/NUM framework is a convex optimization based approach to solving network resource allocation problems. Indeed, 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. The driving motivation behind the LAO 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. As should be clear, distributed solutions are necessary in large-scale networks as centralized approaches can often be infeasible, non-scalable, too costly or too fragile. However, as this approach is built around a *static* optimization problem, it only addresses steady-state performance of the system – in particular, this requires assuming static nominal values for relevant network parameters, and does not allow the designer to explicitly penalize undesirable transient behavior in the system.

As it is impossible for us to review the rich literature pertaining to the LAO/NUM framework we refer the reader to the review papers [1, 2] for an in depth exploration of this material, and defer a focused discussion of applying this methodology to a traffic engineering (TE) problem to §4.

### 2.2 Distributed Disturbance Rejection

For all of its favorable properties, the LAO/NUM framework is not naturally robust to fast-time scale disturbances as dynamics are not explicitly incorporated into the problem formulation. In contrast, the distributed disturbance rejection problem assumes that a nominal network state has already been specified (e.g., from the solution to a NUM problem), and aims to maintain the network state at these values despite network dynamics, un-modeled disturbances and information sharing constraints. As we argued in [3], the appropriate algorithmic tool for solving the disturbance rejection problem is distributed optimal control theory [10, 11], which allows for fast time-scale dynamics and information sharing constraints between local controllers to be explicitly incorporated into the algorithm synthesis procedure. Once again, it is impossible for us to review the increasingly rich literature on this subject – rather we refer the reader to the survey paper [11] and the references therein, as well as the discussion in [12], for a more detailed exposition and defer a focussed review of using distributed optimal control to design an admission controller to §4.

### 2.3 A Unified Optimization Based Approach

As described, LAO/NUM addresses steady-state network resource allocation problems, whereas the disturbance rejection problem ensures that the system efficiently remains at some nominal state despite fast time-scale fluctuations in the dynamics. These two algorithms (NUM and disturbance rejection) are often designed independently of each other by appealing to a time-scale separation between the planning and control problems, or are vertically combined in the data plane leading to stabilizing, rather than optimal/robust, distributed controllers. We argue that with the introduction of SDN and its explicit separation of the application, control and data planes, it is now possible to have the "planning" of network resource allocation explicitly account for and react to fast time-scale dynamics.

In our recently developed framework [4] we extend the NUM framework to explicitly incorporate fast time-scale dynamics. We do so by extending defining a problem that optimizes a *system trajectory* with respect to a functional utility, subject to both network dynamics and information sharing constraints. We then show, through a suitable vertical decomposition of the problem, that a layered architecture composed of a low-level fast-time scale tracking layer (i.e., a data plane controller) and top-level slow-time scale planning layer (i.e., an application plane layer) naturally emerges. The low-level data plane controller consists of a distributed optimal controller that takes as an input a virtual state trajectory generated by the top-level application plane, where this top-level layer consists of a resource allocation problem that optimizes a weighted sum of a network utility function and a "tracking penalty" regularizer. This latter penalty can be interpreted as a model of the data plane controller's ability to follow the virtual reference trajectory issued by the application plane – in this way there is now an explicit, simple, but exact model of the fast-time scale dynamics incorporated into the network resource allocation problem. The control plane is thus tasked with sending state measurements up from the data plane to the application plane to be used by the "planner," as well as transmitting virtual state trajectories down from the application plane to the data plane to be used by the distributed controller to bring the true state of the system to these nominal values. Thus, this new framework should be viewed as an organic extension of the NUM framework to incorporate the additional flexibility and elasticity afforded to the network controller by the SDN framework. Finally we note that our proposed extension to LAO/NUM is as scalable as the underlying optimization and control techniques used in its realization – a detailed discussion of such synthesis and implementation issues are beyond the scope of this paper, and will be addressed in future work.

In the next section, we make the discussion concrete and show how this extension of the LAO/NUM frameworks can be used to explore the Hybrid SDN design space related to a *joint* TE/AC problem.

## 3 A Joint TE/AC Problem

In this section we formulate a joint TE/AC problem, and define three different layered architectures for solving it. The first is a "traditional" architecture (TA) in which the TE problem is updated at a slow time-scale using nominal models of user traffic demands, and the resulting nominal flows are then tracked using an AC. The second is an "extreme" SDN (xSDN) architecture, in which the joint

TE/AC problem is tackled using a model predictive control (MPC) approach – we call this an "extreme" SDN architecture because the network resource allocation TE problem and the disturbance rejection AC problem are jointly solved in the application plane. Finally, the third architecture that we propose is a Hybrid SDN (HySDN) architecture: by applying our novel framework [4], a modified TE problem (that incorporates a suitable tracking penalty) is solved in the application plane using MPC, and the resulting virtual state trajectories are then transmitted to and tracked by an AC embedded in the data plane. In what follows we take care to explicitly model the communication and computation delays needed to implement each of these architectures.

This section is organized as follows: we first recall the TE and AC problems and then use them to define a joint TE/AC problem. We then use these problems to define the three architectures that we intend to study, namely the TA, xSDN and HySDN architectures.

## 3.1 Traffic Engineering

Under the quasi-static model, the traffic engineering problem can be cast as a Multi-Commodity Flow (MCF) problem [9, 13]. Consider a wireline network as a directed graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of links, and link $(u, v)$ has capacity $c_{u,v}$. The offered traffic is represented by a traffic matrix $D_{s,d}$ for source-destination pairs indexed by $(s, d)$. The load $f_{u,v}$ on each link $(u, v)$ depends on how the network decides to route the traffic. TE usually considers a link-cost function $\phi(f_{u,v}, c_{u,v})$ that is an increasing function of $f_{u,v}$. For this paper, we restrict ourselves to $\phi(f_{u,v}, c_{u,v}) = {}^{f_{u,v}}/{}_{c_{u,v}}$ and define the global objective as $\Phi(\{f_{u,v}, c_{u,v}\}) = \max_{u,v\in\mathcal{E}} \phi(f_{u,v}, c_{u,v})$, although more general functions could also be used (such as a piecewise linear approximation to the M/M/1 delay formula).

In order to determine the optimal flows that should be assigned to the links of the network, the following multi-commodity flow problem is solved:

$$\begin{aligned} \min_{\{f_{u,v}^{(s,d)}\}} \quad & \Phi(\{f_{u,v}, c_{u,v}\}) \\ \text{s.t.} \quad & \sum_{v:(s,v)\in\mathcal{E}} f_{s,v}^{(s,d)} - \sum_{u:(u,s)\in\mathcal{E}} f_{u,s}^{(s,d)} = D_{s,d} \ \forall s \neq d \\ & f_{u,v} := \textstyle\sum_{(s,d)} f_{u,v}^{(s,d)} \leq c_{u,v} \ \forall (u,v) \in \mathcal{E} \\ & f_{u,v}^{(s,d)} \geq 0, \end{aligned} \tag{MCF}$$

where $f_{u,v}^{(s,d)}$ corresponds to the flow on link $(u, v)$ from source $s$ to destination $d$.[1] Leveraging the LAO approach, many distributed solutions (with varying degrees of implementation complexity and global optimality) have been developed (cf. [9, 13] and references therein). However, for the purposes of the case study considered in §4, we assume that the TE problem is small enough to be solved by a centralized decision maker.

## 3.2 Admission Control

We recall the admission control task that we defined in [3]: given a set of source-destination pairs $(s, d)$, a set of desired flow rates $f_{(u,v),\star}^{(s,d)}$ on each link $(u, v)$ for said source-destination pairs, and a fixed routing strategy that achieves these flow rates under nominal operating conditions, design an admission control policy that maintains the link flow rates $f_{(u,v)}^{(s,d)}(t)$ as close as possible to $f_{(u,v),\star}^{(s,d)}$ while

[1]Traditionally the MCF problem is formulated in terms of commodities labeled by their destination $d$ only (i.e., $f_{u,v}^{d}$) – we adopt this slightly non-standard formulation in order to facilitate integration with the AC problem we posed and solved in [3].

minimizing the amount of data stored in each of the admission control buffers, despite fluctuations in the source rates $x_s(t)$. In [12], we showed that a fluid model of network traffic can be used to write the network dynamics as

$$\begin{aligned}\mathbf{f}(t+1) &= R\mathbf{f}(t) + S\mathbf{a}(t)\\ \mathbf{A}(t+1) &= \mathbf{A}(t) + \tau\left(M_1\mathbf{x}(t) + M_2\mathbf{f}(t) - \mathbf{a}(t)\right)\end{aligned} \tag{1}$$

where $\mathbf{f}$, $\mathbf{A}$, $\mathbf{a}$ and $\mathbf{x}$ are stacked vectors of the flow rates, admission control buffers, admission control released flow rates, and source rates, respectively, and $\tau$ is the discrete-time sampling rate. The matrices $R$, $S$, $M_1$ and $M_2$ are uniquely specified by the network topology (which we assume to be static for simplicity) and ensure that flow is conserved. For a horizon $N$, our objective is to compute admission controllers that minimize a performance metric of the form

$$\textstyle\sum_{t=1}^{N}\sum_{(u,v),(s,d)}\left(f_{(u,v)}^{(s,d)}(t) - f_{(u,v),\star}^{(s,d)}\right)^2 + \lambda\|\mathbf{A}(t)\|_2^2, \tag{2}$$

subject to the dynamics (1) and the information sharing constraints imposed on the controller. Thus the controllers aim to minimize a weighted sum of flow rate deviations and admission queue lengths over time, where $\lambda > 0$ determines the relative weighting assigned to each of these two terms in the final cost.

### 3.3 The joint TE/AC Problem

We begin by modifying the TE problem (MCF) to incorporate the system dynamics (1), where we now set $x_s(t) = D_{s,d}(t) + w_s(t)$, for $w_s(t)$ some fast-time scale fluctuations around the nominal source rate $D_{s,d}(t)$. We propose the following dynamic extension to (MCF):

$$\begin{array}{rl}\displaystyle\min_{\{f_{u,v}^{(s,d)}\},b,u} & \max_t \Phi\left(\{f_{u,v}(t), c_{u,v}(t)\}\right) + \lambda\Psi(\mathbf{A})\\ \text{s.t.} & D_{s,d}(t) \leq \displaystyle\sum_{v:(s,v)\in\mathcal{E}} f_{s,v}^{(s,d)}(t+1) -\\ & \quad \textstyle\sum_{u:(u,s)\in\mathcal{E}} f_{u,s}^{(s,d)}(t+1) \leq D_{s,d}(t) + \frac{A_{s,v}^{(s,d)}(t)}{\tau}\\ & \quad \forall s \neq d\\ & f_{u,v}(t) := \textstyle\sum_{(s,d)} f_{u,v}^{(s,d)}(t) \leq c_{u,v}(t)\ \forall (u,v)\in\mathcal{E}\\ & f_{u,v}^{(s,d)}(t) \geq 0.\\ & \text{dynamics (1)}\\ & \text{initial conditions } \mathbf{f}(0),\ \mathbf{A}(0).\end{array} \tag{Dyn. MCF}$$

There are two components of note in the dynamic TE problem (Dyn. MCF): the first is the addition of the penalty $\lambda\Psi(\mathbf{A})$ on the buffer terms $\mathbf{A}$ (here we use $\mathbf{A}$ to denote $\mathbf{A}(1), \ldots, \mathbf{A}(N)$, and each $\mathbf{A}(t)$ is a stacked vector of the buffers $A_v^{(s,d)}(t)$). This penalty, which should be chosen to be increasing in $\mathbf{A}$ and convex, ensures that all deviations in flow are not simply stored in the admission control buffers. The second element of note is the modification of the first set of constraints. This constraint now lower bounds the throughput to be at least as large as the nominal value $D_{s,d}(t)$, and limits how much outwards flow is possible at the next time-step as a function of the nominal flow and what is contained in the buffer. In this way we expect the nominal flows to be such that they always meet the demands $D_{s,d}(t)$, and do not exceed the system's capacity to increase flows based on stored data in the buffers. Finally, we note that the dynamics are also incorporated by defining the optimization problem over a horizon of of $t = 0, 1, \ldots, N$ and constraining the flow and buffer values to obey equation (1).

### 3.4 Three Architectures

#### 3.4.1 The Traditional Architecture

The TA is a "decoupled" layered approach, as illustrated in Figure 1a – we say that this approach is decoupled because a static TE problem is solved that does not explicitly take into account the underlying network dynamics. Specifically, the TE problem (MCF) is solved in the application plane every $T$ time-steps using nominal values for $D_{s,d}$. This approach then correspondingly updates the desired flow rates $f^{(s,d)}_{(u,v),\star}$ tracked by the AC. One expects such an approach to work well if the source rates are well modeled by the static traffic demands $D_{s,d}$ over the update window $T$.

#### 3.4.2 The xSDN Architecture

The xSDN approach consists of incorporating optimization problem (Dyn. MCF) into a MPC framework. In particular, at time $k$, the application plane measures the network state $(\mathbf{f}(k-\delta), \mathbf{A}(k-\delta))$, where $\delta$ is the communication delay incurred in measuring the global network state, and uses this network state as the initial condition to solve (Dyn. MCF). We assume that solving this optimization problem takes $\kappa$ time steps – as such, the resulting control sequence $\mathbf{a}(k-\delta), \ldots, \mathbf{a}(k-\delta+N)$ is only available at time $k+\kappa$. The application plane controller then transmits the sequence $\mathbf{a}(k+\kappa), \ldots, \mathbf{a}(k+2\kappa)$ to the admission control buffers in the data plane, which then implement this sequence of control actions. The entire process is then repeated. See Figure 1c for a diagram of this control architecture.

#### 3.4.3 The HySDN Architecture

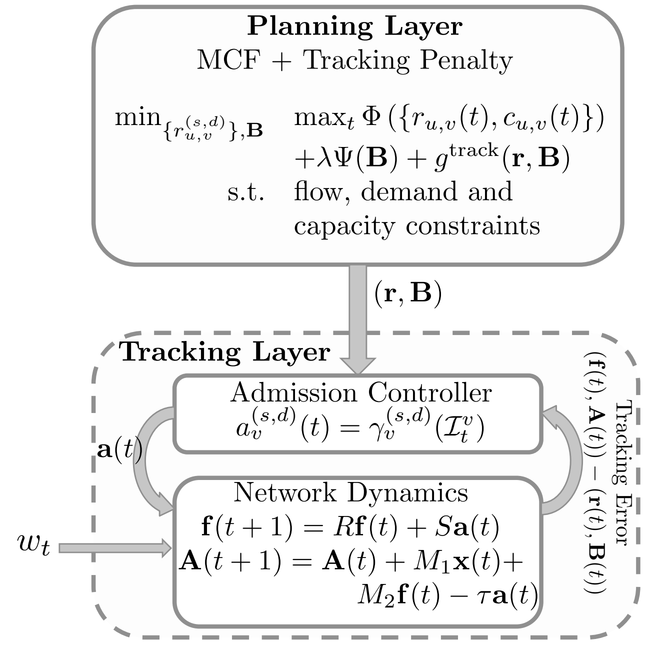

Planning Layer
MCF + Tracking Penalty
$\min_{\{r^{(s,d)}_{u,v}\},\mathbf{B}}$ $\max_t \Phi(\{r_{u,v}(t), c_{u,v}(t)\})$
$+\lambda\Psi(\mathbf{B}) + g^{\text{track}}(\mathbf{r},\mathbf{B})$
s.t. flow, demand and capacity constraints
$(\mathbf{r},\mathbf{B})$
Tracking Layer
Admission Controller
$a^{(s,d)}_v(t) = \gamma^{(s,d)}_v(\mathcal{I}^v_t)$
$\mathbf{a}(t)$
Network Dynamics
$\mathbf{f}(t+1) = R\mathbf{f}(t) + S\mathbf{a}(t)$
$\mathbf{A}(t+1) = \mathbf{A}(t) + M_1\mathbf{x}(t) + M_2\mathbf{f}(t) - \tau\mathbf{a}(t)$
$w_t$
Tracking Error
$(\mathbf{f}(t), \mathbf{A}(t)) - (\mathbf{r}(t), \mathbf{B}(t))$

Figure 2: Diagram of HySDN architecture for joint TE/AC. Note that to reduce clutter, we we use $\mathbf{r}$ to denote $\{r^{(s,d)}_{u,v}\}$ in the diagram.

Architecturally, the HySDN approach can be viewed as an intermediate between the TA and xSDN architectures: some functionality is included in the application layer, and some is included in the data plane layer (cf. Fig. 1b). A novel feature is the inclusion of a model of data plane control capabilities (via a tracking penalty) in the application plane, thus allowing the HySDN controller to inherit the strengths of both the TA and the xSDN control schemes, i.e., data plane feedback control and responsive dynamic TE.

In particular, as proposed in [4], we introduce virtual reference variables $r_{u,v}^{(s,d)}$ and $\mathbf{B}$ (where $\{r_{u,v}^{(s,d)}\}$ are the virtual flow rate references for the true flows $\{f_{u,v}^{(s,d)}\}$, and $\mathbf{B}$ is the virtual buffer reference for the true buffer values $\mathbf{A}$). The use of these virtual reference variables functionally separates the tasks of planning the optimal flow rates and buffer values and maintaining the network state at these values. In particular, this results in a "planning layer" in the application plane that computes virtual state trajectories $\{r_{u,v}^{(s,d)}\}$ and $\mathbf{B}$, and a "tracking layer" in the data plane that takes these virtual state trajectories as an input and computes control actions $\mathbf{a}$ such that $\{r_{u,v}^{(s,d)}\} \approx \{f_{u,v}^{(s,d)}\}$ and $\mathbf{B} \approx \mathbf{A}$ (cf. Figure 2). An important feature of this methodology is that the fast-time scale tracking-level control policies can be precomputed offline, allowing for real-time implementation of the AC as *feedback* controllers.

We omit the details of the derivation (as these are a straightforward application of the methods described in [4]), and rather present the resulting optimization problem that the "planning layer" must solve.

$$\begin{array}{rl} \min\limits_{\{r_{u,v}^{(s,d)}\},\mathbf{B}} & \max_t \Phi\left(\{r_{u,v}(t), c_{u,v}(t)\}\right) + \lambda\Psi(\mathbf{B}) + g^{\text{track}}(\{r_{u,v}^{(s,d)}\},\mathbf{B}) \\ \text{s.t.} & D_{s,d}(t) \leq \sum\limits_{v:(s,v)\in\mathcal{E}} r_{s,v}^{(s,d)}(t+1) - \\ & \sum_{u:(u,s)\in\mathcal{E}} r_{u,s}^{(s,d)}(t+1) \leq D_{s,d}(t) + \frac{B_{s,v}^{(s,d)}(t)}{\tau} \\ & \quad \forall s \neq d \\ & r_{u,v}(t) := \sum_{(s,d)} r_{u,v}^{(s,d)}(t) \leq c_{u,v}(t)\ \forall (u,v)\in\mathcal{E} \\ & r_{u,v}^{(s,d)}(t) \geq 0. \end{array} \qquad \text{(Planning MCF)}$$

Notice in particular that the problem (Planning MCF) no longer needs to solve for the fast-time scale admission control actions $\mathbf{a}$ (these are pre-solved for in the tracking problem, cf. [4]), and that the dynamic constraints are no longer present. Rather the objective function is augmented with an additional penalty function $g^{\text{track}}(\{r_{u,v}^{(s,d)}\},\mathbf{B})$ which, as we describe in [4], captures the tracking layer's ability to guide the true state of the system to the match the reference values $(\{r_{u,v}^{(s,d)}\},\mathbf{B})$. Thus this problem is now only subject to the simple constraints that are a natural generalization of those present in (MCF). Much as in the xSDN architecture, we incorporate the planning problem (Planning MCF) into a MPC framework – however, now rather than transmit explicit control actions $\mathbf{a}(k+\kappa),\ldots,\mathbf{a}(k+2\kappa)$ down to the tracking layer, the application plane transmits its virtual reference trajectories $(\{r_{u,v}^{(s,d)}\},\mathbf{B})$ to the AC in the data plane, which then uses *feedback control* to bring the state of the system to these nominal values. In the next section, we study a simple example of the joint TE/AC problem and explore the effects of computational delay on the performance achieved by the different architectures.

## 4 An Illustrative Case Study

We consider the simple network topology illustrated in Figure 3: we assume that there is an admission controller present at nodes 1 and 2. We chose our case study topology and parameters to highlight differences in performance between the three architectures using a simple and easily visualized system. Because of the simple topology that we consider, it is necessary to set some

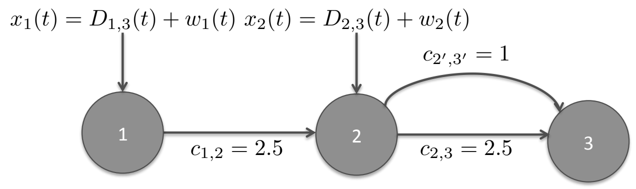

$x_1(t) = D_{1,3}(t) + w_1(t)$
$x_2(t) = D_{2,3}(t) + w_2(t)$
$c_{2',3'} = 1$
1
2
3
$c_{1,2} = 2.5$
$c_{2,3} = 2.5$

Figure 3: Case study topology.

parameters at slightly unrealistic values to obtain noticeable gaps in performance between the different architectures – we expect more dramatic differences to become apparent for more complex topologies and coordination problems.

| | $\kappa = 1$ | $\kappa = 2$ | $\kappa = 3$ | $\kappa = 4$ |
|---|---|---|---|---|
| Avg. $\Phi$ | 0.9049 | 0.9032 | 0.9005 | 0.8997 |
| Avg $A_1$ | 0.0447 | 0.0587 | 0.0728 | 0.0902 |
| Avg $A_2$ | 0.0362 | 0.0377 | 0.0427 | 0.0473 |

Table 1: HySDN simulation results

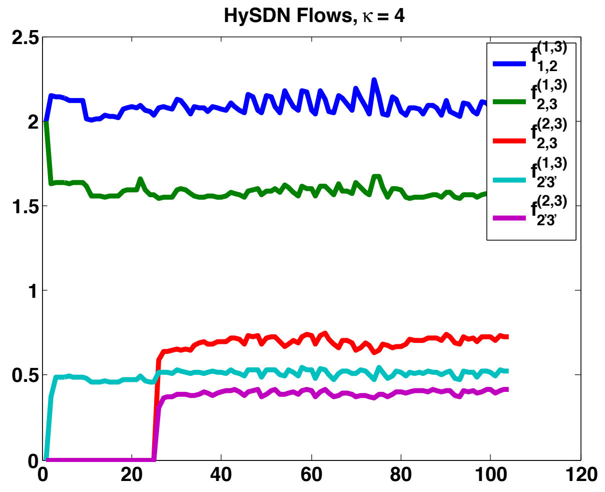
| | $\kappa = 1$ | $\kappa = 2$ | $\kappa = 3$ | $\kappa = 4$ |
|---|---|---|---|---|
| Avg. $\Phi$ | 0.9005 | 0.8983 | 0.8992 | 0.9106 |
| Avg $A_1$ | 0.0513 | 0.0653 | 0.0785 | 0.1533 |
| Avg $A_2$ | 0.0477 | 0.0519 | 0.0567 | 0.0604 |

Table 2: xSDN simulation results

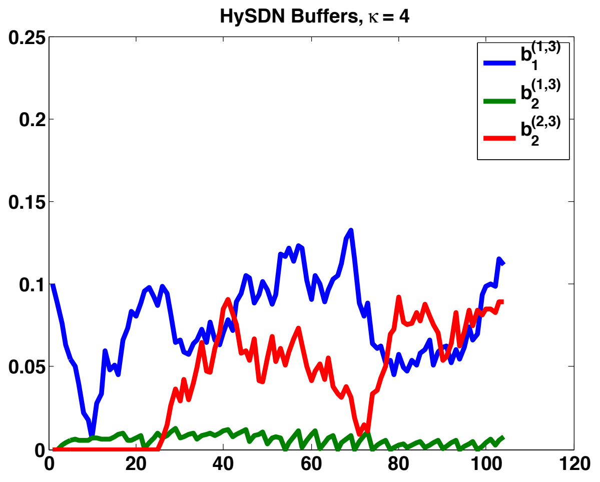
With this in mind, we set the sampling time $\tau$ to be .1s, and make the following simplifying assumption: we set the communication delay $\delta$ to 0 (as this delay would be identical across all architectures) and assume that the Globally Optimal Delay free (GOD) AC is implemented by the TA and HySDN architectures (cf. [3] for more details). We emphasize that both of these assumptions are made to highlight the differences in performance caused by different types of vertical layering – our approach can explicitly accommodate both nonzero communication delays and more complex information sharing patterns in the AC, as defined in [3]. Likewise we assume that in the xSDN architecture, the admission control actions $\mathbf{a}$, once computed, can also be instantaneously shared and implemented by the AC. In this section we study the effects of increasing the computation delay $\kappa$ on the performances of the xSDN and HySDN architectures, and compare these two schemes to the TA approach of solving a static TE problem every $T$ time-steps and using an AC to maintain the computed nominal flow rates.

We consider the following source rates: $x_1(t) = 2 + w_1(t)$ and $x_2(t) = \text{step}_{25}(t) + w_2(t)$, where the $w_i(t) \sim \mathcal{N}(0.1, .01)$ are i.i.d. additive white Gaussian noise (AWGN) with a slight positive bias (mean of .1) and small variance ($\sigma^2 = .01$), and $\text{step}_k(t) = 0$ for $t < k$ and 1 for $t \geq k$ (i.e., $D_{2,3}(t)$ is 0 from times 0 to 24, and then steps to 1 afterwards). The positive bias can be seen as modeling an underestimate of the nominal traffic demands $D_{s,d}(t)$ by the TE layer.

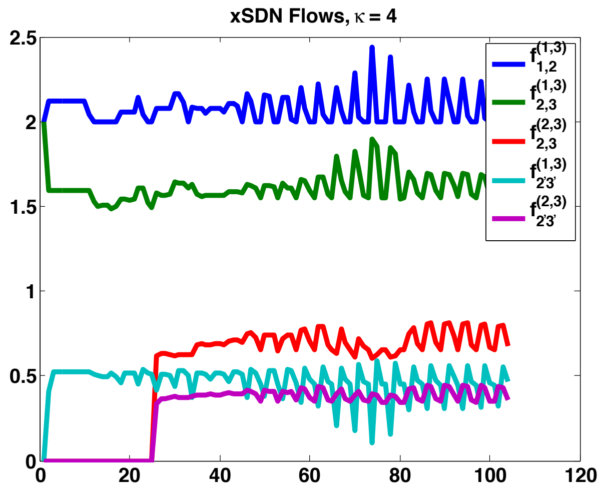
Summarized in Tables 1 and 2 are the average link utilizations (Avg. $\Phi$) and average buffer sizes (Avg. $A_i$) for the HySDN and xSDN architectures as a function of computation delay $\kappa$. Those values are computed across ten simulations beginning from initial conditions specified by $A_1^{(1,3)}(0) = .1$, $f_{1,2}^{(1,3)}(0) = f_{2,3}^{(1,3)}(0) = 2$, and all other states set to 0. For each simulation iterate, both architectures were driven by the same noise sequence, thus ensuring a fair comparison. Further representative state trajectories are shown in Figures 4a-4d for the HySDN and xSDN architectures



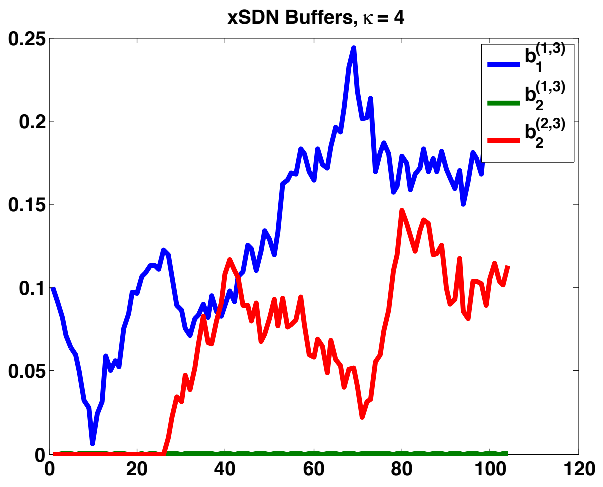
(a) HySDN Flows, $\kappa = 4$.



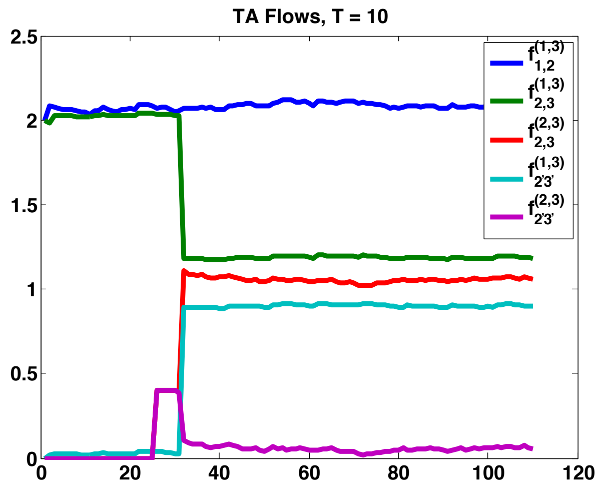
(b) HySDN Buffers, $\kappa = 4$.



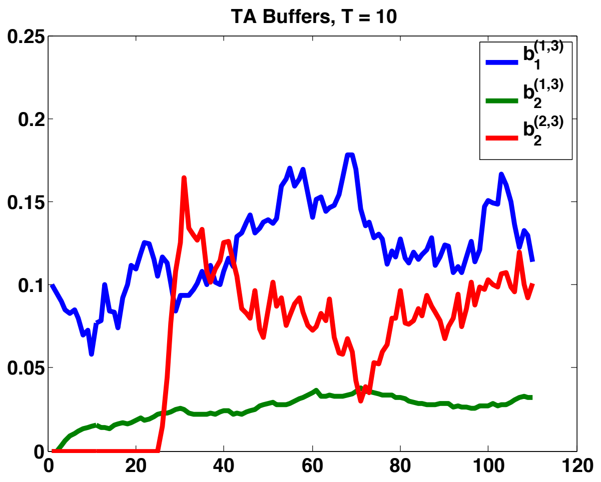
(c) xSDN Flows, $\kappa = 4$.



(d) xSDN Buffers, $\kappa = 4$.



(e) TA Flows, $T = 10$.



(f) TA Buffers, $T = 10$.

Figure 4: Representative time series plots.

for computation delay $\kappa = 4$.

As can be seen, as $\kappa$ increases, the gap in performance between xSDN and HySDN increases. This is because the HySDN architecture incorporates explicit feedback in the AC layer to compute its control actions, thus providing robustness to computational delay – in contrast, the xSDN control is computed ahead of time and hence applied in "open-loop" in between computation delay limited updates. Finally we show representative trajectories for the TA architecture in Figure 4e and 4f. We investigated this scheme for a TE update period $T = 10$. The resulting performance metrics, again computed across 10 simulations, are: Avg.$\Phi = 0.9282$, Avg.$A_1 = 0.1217$, Avg.$A_2 = 0.0924$. The TA architecture is outperformed by the HySDN architecture for all values of $\kappa$, and has comparable performance to the xSDN architecture only for larger $\kappa$.

## 5 Summary & Future Work

In this paper we introduced a dynamic version of the LAO/NUM framework, and argued that it is a natural extension of these well established techniques to SDN. Through the use of a joint TE/AC problem, we defined and explored the corresponding HySDN design space which was composed of traditional, extreme SDN and hybrid SDN architectures – these different architectures are distinguished by the different types of vertical layering that they employ.

Going forward, we aim to apply these ideas to other joint planning/control problems in the context of SDN controlled wide area networks and data-centers, and to validate these novel layered control architectures in experimental testbeds. We also believe that the ideas introduced in this paper may prove fruitful in designing *elastic* architecture control paradigms that jointly exploit SDN and NFV to allow for virtual architectural resources to be allocated and distributed in real-time.

## References

[1] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J.Sel. A. Commun.*, vol. 24, no. 8, pp. 1439–1451, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2006.879350

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

[3] N. Matni, A. Tang, and J. C. Doyle, "A case study in network architecture tradeoffs," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, ser. SOSR '15. New York, NY, USA: ACM, 2015, pp. 18:1–18:7. [Online]. Available: http://doi.acm.org/10.1145/2774993.2775011

[4] N. Matni and J. C. Doyle, "A theory of dynamics, control and optimization for layered architectures," in *2016 IEEE American Control Conference (ACC), Submitted to*, ser. ACC '16, 2016.

[5] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486012

[6] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, “B4: Experience with a globally-deployed software defined wan,” *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, Aug. 2013. [Online]. Available: http://doi.acm.org/10.1145/2534169.2486019

[7] 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

[8] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, “Conga: Distributed congestion-aware load balancing for datacenters,” *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 503–514, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2740070.2626316

[9] D. Xu, M. Chiang, and J. Rexford, “Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering,” *Networking, IEEE/ACM Transactions on*, vol. 19, no. 6, pp. 1717–1730, Dec 2011.

[10] M. Rotkowitz and S. Lall, “A characterization of convex problems in decentralized control,” *Automatic Control, IEEE Transactions on*, vol. 51, no. 2, pp. 274–286, 2006.

[11] A. Mahajan, N. Martins, M. Rotkowitz, and S. Yuksel, “Information structures in optimal decentralized control,” in *Decision and Control (CDC), IEEE 51st Annual Conf. on*, Dec 2012, pp. 1291–1306.

[12] N. Matni, A. Tang, and J. C. Doyle, “Technical report: A case study in network architecture tradeoffs,” *Technical Report*, 2015.

[13] N. Michael and A. Tang, “Halo: Hop-by-hop adaptive link-state optimal routing,” *Networking, IEEE/ACM Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[14] N. Matni, D. Ho, A. Tang, D. Meyer, and J. C. Doyle, “Technical report: Dynamics, control and optimization in layered network architectures,” *Technical Report*, 2016.