A Case Study in Network Architecture Tradeoffs*

Nikolai Matni
California Institute of Technology
1200 E California Blvd
Pasadena, California
nmatni@caltech.edu

Ao Tang
Cornell University
337 Frank H. T. Rhodes Hall
Ithaca, NY
atang@ece.cornell.edu

John C. Doyle
California Institute of Technology
1200 E California Blvd
Pasadena, California
doyle@caltech.edu

ABSTRACT

Software defined networking (SDN) establishes a separation between the control plane and the data plane, allowing network intelligence and state to be centralized – in this way the underlying network infrastructure is hidden from the applications. This is in stark contrast to existing distributed networking architectures, in which the control and data planes are vertically combined, and network intelligence and state, as well as applications, are distributed throughout the network. It is also conceivable that some elements of network functionality be implemented in a centralized manner via SDN, and that other components be implemented in a distributed manner. Further, distributed implementations can have varying levels of decentralization, ranging from myopic (in which local algorithms use only local information) to coordinated (in which local algorithms use both local and shared information). In this way, myopic distributed architectures and fully centralized architectures lie at the two extremes of a broader hybrid software defined networking (HySDN) design space.

Using admission control as a case study, we leverage recent developments in distributed optimal control to provide network designers with tools to quantitatively compare different architectures, allowing them to explore the relevant HySDN design space in a principled manner. In particular, we assume that routing is done at a slower timescale, and seek to stabilize the network around a desirable operating point despite physical communication delays imposed by the network and rapidly varying traffic demand. We show that there exist scenarios for which one architecture allows for fundamentally better performance than another, thus highlighting the usefulness of the approach proposed in this paper.

*N. Matni & J. C. Doyle were in part supported by the AFOSR and the Institute for Collaborative Biotechnologies through grant W911NF-09-0001 from the U.S. Army Research Office. A. Tang was in part supported by the ONR under grant N00014-12-1- 1055.

SOSR2015, June 17 - 18, 2015, Santa Clara, CA, USA

DOI: http://dx.doi.org/10.1145/2774993.2775011.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Miscellaneous; C.4 [**Performance of Systems**]: Miscellaneous.

General Terms

Design, Theory, Performance

1. INTRODUCTION

A common challenge that arises in the design of networks is that of achieving globally optimal behavior subject to the latency, scalability and implementation requirements of the system. Many system properties and protocols – such as network throughput, resource allocation and congestion avoidance – are inherently global in scope, and hence benefit from centralized solutions implemented through Software Defined Networking (SDN) (e.g., [3, 4, 9]). However, such centralized solutions are not always desirable due to latency and scalability constraints – in particular the delay inherent in communicating the global state of the network, solving a global optimization problem and redistributing the solution to the network, can often negate the benefits achieved from this holistic strategy. In these cases, distributed networking solutions can be preferable (e.g., [1]).

Traditionally, there has been very little in the way of theoretical tools at the disposal of network designers to quantitatively compare different architectures. Because of this, the appropriateness of an architectural decision could only be confirmed once a suitable algorithm had been prototyped and validated via experiment. This process is time consuming and expensive, and even worse, can be inconclusive. In particular, if an algorithm performs poorly, it is not clear if this poor performance is due to an inherent limitation of the chosen architecture, or simply due to a poorly designed algorithm, making it very difficult to quantitatively compare network architectures.

In this paper, we argue that due to both the increased flexibility afforded to network designers by SDN and to recent advances in distributed optimal control, the gap between theory and practice has closed significantly. Although the gap has not yet completely closed, we show that in the context of certain network applications, existing theory can indeed be used to quantitatively compare the fundamental performance limits of different network architectures. In particular, we leverage the fact that new theory can now explicitly take into account the effect of communication delays inherent to coordinating control actions in a networked setting.

Towards this end, we define the broader *Hybrid Software Defined Networking* (HySDN) design space (§2) of network architectures, and argue that an important metric in determining the appropriateness of an architecture is the optimal performance achievable by any algorithm implemented on that architecture. This additional metric can then be used by a network designer to quantify the performance tradeoffs associated with using a simpler or more complex architecture, allowing for more informed decisions about architecture early in the design process.

We further explain how recent advances in distributed optimal control theory allow us to apply this approach to a class of network control problems in which the objective is to regulate network state around a pre-specified desired set-point. In §3, we illustrate the usefulness of this approach with an admission control case study. Perhaps surprisingly, we show that for two nearly identical routing topologies, there can be significant differences in the performance achievable by centralized and distributed network architectures.

We emphasize that we are not arguing that a specific implementation, algorithm or architecture is best suited for a given application – rather, we are illustrating the usefulness of a methodology that allows network designers to make quantitative decisions about architectural choices early in the design process. We are also not proposing that this method replace the important steps of simulation, prototyping and experiments, but rather as a complement to it. Indeed, in order to achieve theoretical tractability, simplifying assumptions such as linearized flow models, constant delays, reliable control packet communication and negligible computation time are made – the validity of these assumptions can only be confirmed through experiments. We do however believe that the proposed tools can help network designers streamline the prototyping process by allowing them to narrow down the range of potential architectures that need to be explored.

2. ARCHITECTURAL TRADEOFFS

Completely distributed network architectures, in which local algorithms take local actions using only locally available information, and centralized network architectures, in which a centralized algorithm takes global action using global information, can be viewed as lying at the extremes of a much richer design space. It is possible to build a network architecture in which certain network logic elements are implemented in a centralized fashion via SDN, and in which other network logic elements are implemented in a distributed fashion. Further, distributed architectures can have varying levels of decentralization, ranging from completely distributed (as described above), or *myopic*, architectures to *coordinated* distributed architectures, in which local algorithms take local actions using both locally available information and shared subsets of global state information. We call this broad space of architectures the Hybrid Software Defined Networking (HySDN) design space (illustrated in Figure 1), as its constituent architectures are naturally viewed as hybrids of distributed and software defined networks.

The question then becomes how to explore this even larger design space in a systematic way. As we have already alluded to, there are inherent tradeoffs associated with any architecture: algorithms running on centralized architectures typically achieve better steady state performance, but often react with higher latency than those implemented on a distributed architecture – conversely distributed algorithms are often simpler to implement but can lead to less predictable steady state performance.

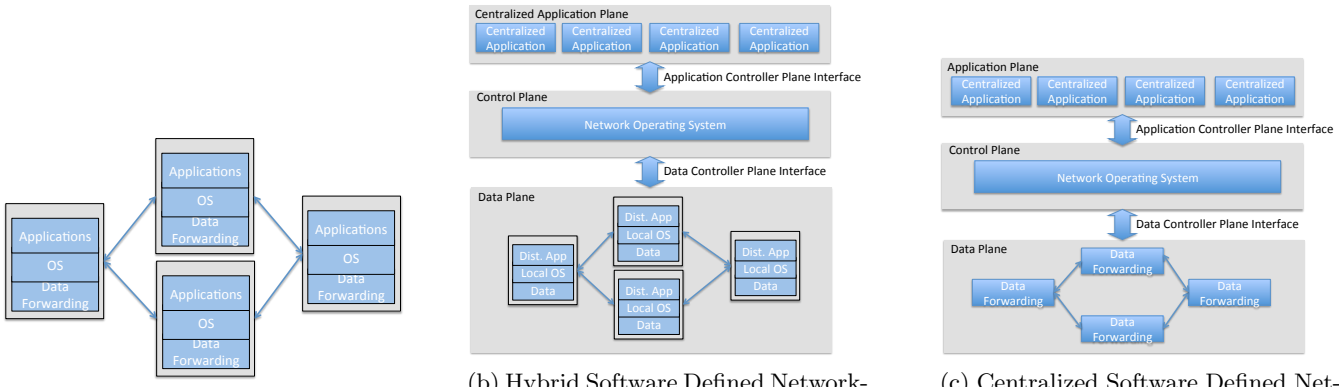
We pause here to recognize that network architectures and algorithms are judged by many different metrics, including but not limited to performance, scalability, ease of deployment and troubleshooting, and flexibility. We argue that as much as possible, each of these different metrics should be traded off against each other in a quantitative way – for example, it is important for a network designer to be able to quantify the performance degradation suffered by using a network architecture or algorithm that is simpler to deploy and troubleshoot. Our approach to exploring these tradeoffs is simple: we compare network architectures by comparing the optimal performance achievable by any algorithm implemented using them. This approach allows for a network designer to compare fundamental limits of achievable performance across different architectures. The approach also allows for the comparison of the performance of more scalable, flexible, or easier to deploy algorithms implemented using a given network architecture to that network architecture's best possible performance, allowing for a quantifiable tradeoff between these metrics of interest.

In order to make the discussion concrete, we focus on algorithms that can be viewed as controllers that aim to keep the state of the network as close to a nominal operating point as possible, while exchanging and collecting information subject to the communication delays imposed by the network. For example, in §3, we consider admission control algorithms that aim to keep the link flow rates at a user-specified setpoint while minimizing the admission buffer size, despite physically imposed communication delays and rapidly varying source rates. In particular, we are not addressing the problem of determining what nominal operating point the controllers should attempt to bring the network state to – we aim to extend our analysis to such problems in future work. We recognize that this measure of performance is not standard in the networking community, but note that it is a natural one to consider when explicitly taking into account rapidly varying and unpredictable source rates.

By restricting ourselves to problems of this nature, we can leverage recent results in distributed optimal control theory to classify those architectures for which the optimal algorithm and achievable performance can be computed efficiently.[1] It is well known that the optimal centralized controller, delayed or not, can be computed efficiently via convex optimization [13]. Further, it is known that myopic distributed optimal controllers are in general NP-hard to compute [12, 8]. Up until recently however, it was unclear if and when coordinated distributed optimal controllers could be specified as the solution to a convex optimization problem.

The challenge inherent in optimizing distributed control algorithms is that control actions (e.g., local admission control decisions) can potentially serve two purposes: actions taken by local controllers can be used to both control the state of the system in a manner consistent with performance

[1]Throughout this discussion, we assume that the dynamics of the network are linear around a neighborhood of the nominal operating point. This assumption holds true for many commonly used network flow models [5, 2].



(a) Distributed Networking

(b) Hybrid Software Defined Networking

(c) Centralized Software Defined Networking

Figure 1: The Hybrid Software Defined Networking Design space, ranging from distributed (Fig 1a) to centralized (Fig 1c) network protocols.

objectives, and to signal to other local controllers, allowing for implicit communication and coordination. Intuitively, it is this attempt to both control the system and to implicitly communicate that makes the problem difficult to solve computationally. However, if local controllers are able to coordinate their actions via explicit communication, rather than by implicit signaling through the system, then the optimal controller synthesis problem becomes computationally tractable [11, 10]. Further it is not difficult to argue that distributed controllers using explicit communication to coordinate will outperform those relying on implicit signaling through the system. Removing the incentive to signal through the system can be done in a network control setting by giving control dedicated packets, i.e., packets containing the information exchanged between local algorithms, priority in the network.[2]

These theoretical developments thus provide the necessary tools to explore a much larger section of the HySDN design space in a principled manner. We propose leveraging these results to compare the performance achievable by algorithms implemented on four different classes of architectures, described below:

1. **The GOD architecture**: in order to quantify the fundamental limits on achievable performance, we propose computing the optimal controller implemented using the Globally Optimal Delay-free (GOD) architecture. This architecture assumes instantaneous communication to and from a central decision maker – although not possible to implement, the performance achieved by this architecture cannot be beaten, and as such represents the standard against which other architectures should be compared.

2. **The centralized architecture**: this architecture corresponds to the SDN approach, in which a centralized decision maker collects global information, computes a global control action to be taken, and broadcasts it to the network. Although global in scope, the latency of algorithms implemented using this architecture is determined by the communication delays inherent in collecting the global network state and broadcasting global actions.

3. **The coordinated architecture**: this architecture is distributed, but allows for sufficient coordination between local controllers so that the optimal control law can be computed efficiently [11, 10]. This architecture takes both rapid action based on timely local information, and slower scale action based on global but delayed shared information, and can thus be viewed as an intermediate between centralized and myopic architectures.

4. **The myopic architecture**: this architecture is one in which local controllers take action based on local information. Although the optimal controller cannot be computed, the performance achieved by any myopic controller can be compared with the performance achieved by an optimal coordinated controller, thus providing a bound on the performance difference between the two architectures.

It should be noted that the coordinated architecture will always perform at least as well as the myopic and centralized architectures, as any algorithm implemented on the latter architectures can also be implemented on the former. It is clear why this holds for myopic algorithms, but it is worth emphasizing why this holds for centralized algorithms. This is true because the delays faced by a coordinated distributed algorithm in collecting and sharing state information are also faced by a centralized algorithm. Whereas a centralized algorithm waits until the global state has been collected and processed to react to make changes to the system, a coordinated distributed algorithm takes both local timely actions based on local information and delayed actions based on shared information.

What we seek to understand is how large of a gap in performance exists between these different architectures. By computing the performance of each of these architectures, the network designer can then quantify tradeoffs in implementation complexity and performance in a computationally

[2]Specifically, if local controllers can communicate with each other as quickly as the effect of their actions propagate through the network, then the resulting optimal control problem is convex. By giving such communication packets priority in the network and ensuring that they are routed along suitably defined shortest paths, this property is guaranteed to be satisfied [10].

efficient and inexpensive manner. We demonstrate the usefulness of this approach on an admission control case study in the next section.

3. ADMISSION CONTROL DESIGN

In this section we pose an admission control problem, define the relevant HySDN design space and show that it can be explored in a principled and quantitative manner using tools from distributed optimal control theory. We discuss the problem at a conceptual level in this section, and refer the interested reader to [7] for the technical details.

3.1 Problem

We consider the following admission control task: given a set of source-destination pairs (s, d), a set of desired flow rates $f^{\star}_{\ell,(s,d)}$ on each link ℓ for said source-destination pairs, and a fixed routing strategy that achieves these flow rates, design an admission control policy that maintains the link flow rates $f_{\ell,(s,d)}(t)$ as close as possible to $f^{\star}_{\ell,(s,d)}$ while minimizing the amount of data stored in each of the admission control buffers, despite fluctuations in the source rates $x_s(t)$.

The architectural decision that the network designer is faced with is whether to implement the admission control policy in a myopic, coordinated, or centralized manner – representative examples of these possible architectures are illustrated in Figure 2 for the case of three sources. In the myopic distributed architecture, local admission controllers $AC1$, $AC2$ and $AC3$ have policies that depend solely on their local information – in what follows, we define the local information available to a local algorithm for the specific case studies that we consider. In the coordinated distributed architecture, the local admission controllers take action based on both locally available information and on information shared amongst themselves – this shared information is delayed, as it must be communicated across the network and is therefore subject to propagation delays. Finally, in the centralized architecture, a central decision maker collects the admission control buffer and link flow rate states subject to appropriate delays, determines a global admission control strategy to be implemented and broadcasts it to the local AC controllers – this strategy also suffers from delay due to the need to collect global state information and to broadcast the global policy to each AC controller.

As mentioned in the previous section, we know *a priori* that the coordinated distributed optimal control architecture will perform no worse than either the myopic or centralized architecture. Conversely, the myopic and centralized schemes are significantly easier to deploy, and the centralized scheme is significantly easier to troubleshoot. Thus, in quantifying the gaps in performance between the centralized, myopic and distributed architectures, the network designer is able to make an informed decision as to whether the added complexity of the coordinated distributed scheme is warranted.

As described in §2, our approach to exploring the HySDN design space is to compute optimal admission controllers implemented on each of these different architectures. In particular we compute admission controllers that minimize a performance metric of the form

$$\sum_{t=1}^{N} \sum_{\ell,(s,d)} \left(f_{\ell,(s,d)}(t) - f^{\star}_{\ell,(s,d)}\right)^2 + \lambda \|A(t)\|_2^2, \qquad (1)$$

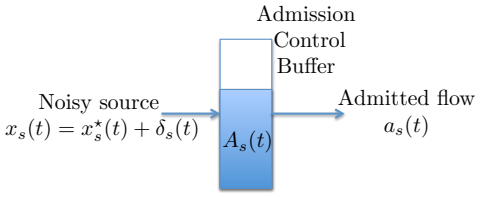


Figure 3: Diagram of an edge admission controller.

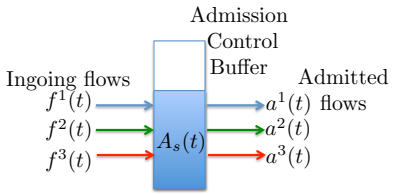


Figure 4: Diagram of an internal admission controller.

where $A(t)$ is a vector containing the size of the admission control buffers and N is the optimization horizon. 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. By solving the corresponding optimal control problems, we obtain two parameters: an optimal cost and an admission control policy that achieves it. These optimal costs thus serve as a quantitative measure of the performance of a given architecture, as by definition, they correspond to the best performance achievable by any admission control policy implemented on that architecture.

3.2 Case Study

We consider a simple routing topology overlaid onto the abilene network, and two different admission control scenarios: one in which only edge admission control is allowed (cf. Figure 3) and one in which edge and internal admission control is allowed (cf. Figure 4). We model the dynamics of the system using a flow based model and solve the optimal control problem with the cost (1) taken to be the infinite horizon LQG cost using the methods described in [13] and [6] – we refer the reader to [7] for the technical details. Intuitively, this cost measures the amount of "energy" transferred from the source rate deviations to the flow rate deviations and buffer sizes. We assume that the nominal source rates $x_s^{\star}(t)$ and the nominal flow rates $f^{\star}_{\ell,(s,d)}$ are all equal to 1 (this is without loss of generality through appropriate normalization of units), and empirically choose $\lambda = 50$ based on the observed responses of the synthesized controllers.

We compute three optimal controllers for each of the scenarios considered: a coordinated distributed optimal controller in which local admission controllers are able to exchange information via the network in order to coordinate their actions, as illustrated in the middle pane of Figure 2, a centralized optimal controller subject to the delays induced by collecting global state information and broadcasting a global control action, and the GOD controller. We also compare the performance of these controllers with the performance achieved by the best myopic distributed controller we are able to compute via non-linear optimization (recall that optimal myopic controllers are in general computationally intractable to compute).

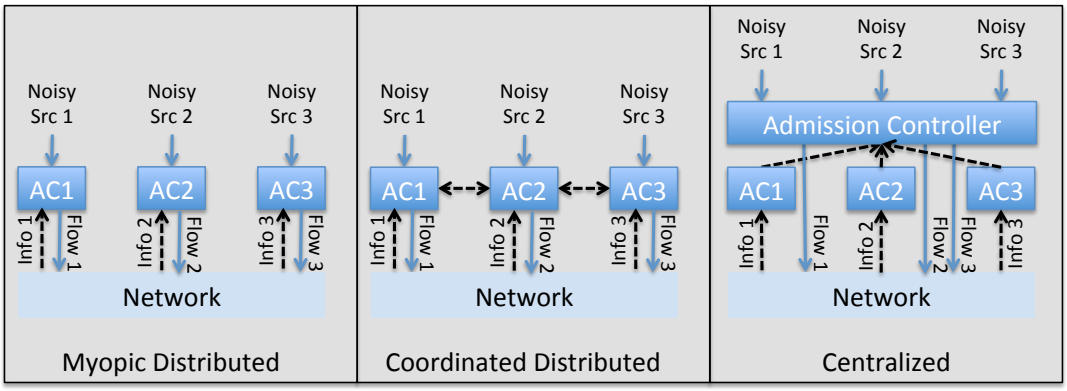


Figure 2: The HySDN design space for an admission control problem with three sources. Blue arrows denote the flow of traffic, whereas dashed black lines denote the flow of admission control related information. Note that the dotted lines correspond to virtual connections, and can be implemented using either control dedicated communication links, or using control dedicated tunnel overlays on the network.

Figure 5: Routing topology used for case study: each source-destination path is denoted by a dashed line. Sources 1 and 2 have edge admission controllers as depicted in Figure 3, whereas Source 3 either has an edge admission controller or an internal admission controller, as depicted in Figure 4, depending on the scenario considered.

We present two different settings to illustrate the benefit of our approach to exploring the HySDN design space: one in which using a coordinated distributed algorithm leads to a significant improvement over a centralized algorithm and a slight improvement over a myopic distributed algorithm, and one for which the optimal GOD algorithm is inherently myopic in nature. In the former case, the significant improvement over a centralized implementation justifies the use of a coordinated or myopic architecture, whereas in the latter case, the myopic distributed architecture is the clear choice as it achieves the same performance as a controller implemented using the GOD architecture.

The topology that we consider is illustrated in Figure 5 – source-destination pairs are illustrated with dashed lines, and each source has an admission controller. We first consider the edge only admission control scenario, where each admission controller is as depicted in Figure 3, and compute the optimal controller implemented using the GOD architecture. Perhaps surprisingly, the optimal control policy is naturally myopic, i.e., the admitted flow $a_s(t)$ at admission control buffer s is strictly a function of $A_s(t)$. In other words, there is *no loss in performance* in using a myopic distributed architecture so long as the local control actions are appropriately specified.

We next consider a scenario where Sources 1 and 2 have edge admission controllers, but now Source 3 has an internal admission controller as depicted in Figure 4. In particular, the internal admission controller takes as inputs both the incoming flows due to Sources 1 and 2 arriving at the Denver switch, as well as the contribution from Source 3. We once again begin by computing the optimal controller implemented using the GOD architecture. In this case, the GOD policy requires instantaneous sharing of information between different admission controllers and hence cannot be implemented. We then compute a centralized optimal controller, in which we assume that the central decision maker is located at the Denver switch. At this location, the largest round trip time between Denver and an admission controller is 18ms: we assume that computation time is negligible, and thus, it takes 18ms for the centralized decision maker to react to local changes.

We also compute the optimal coordinated distributed controller, in which we assume that admission controllers have access to flow rate and admission control buffer information with delay specified by the routing topology. For example, the admission control buffer at Source 1 has access to the admission control buffer state at Source 2 with a delay of 3ms and to the admission control buffer state at the Denver switch with a delay of 6ms. This information sharing protocol is sufficient for the optimal coordinated distributed controller to be specified by the solution to a convex optimization problem, and can be computed efficiently using the methods in [6]. Finally, we bound the performance of the myopic architecture using the best myopic distributed algorithm that we are able to generate via non-linear optimization.

We normalize the costs achieved by each of the architectures by the performance achieved by the GOD architecture: in this way, a ratio of 1 corresponds to the best possible performance achievable. The optimal algorithm implemented using the centralized architecture achieved a cost ratio of 1.13 – thus the delay needed to take centralized decisions leads to a 13% performance degradation over the GOD architecture performance. We then computed the optimal co-

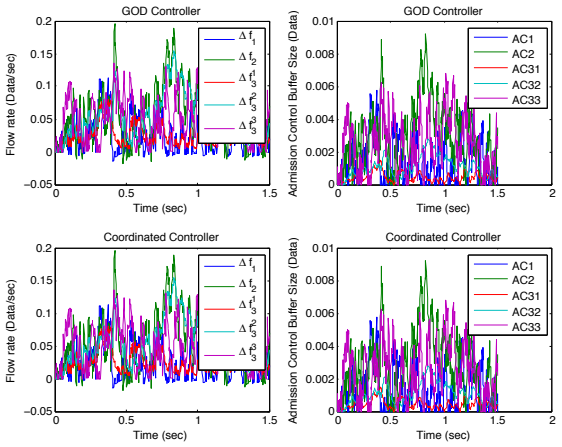


Figure 6: Sample flow deviations and admission buffer length evolution under the GOD and coordinated distributed controllers when the system is subject to the random source rate fluctuations, illustrated in Figure 7.

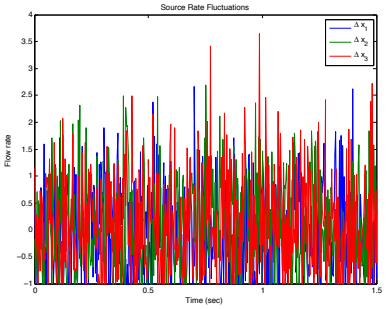


Figure 7: Sample source rate fluctuations – these are lower bounded by -1 as the nominal source rates are all assumed to be 1.

ordinated distributed controller and obtained a cost ratio of 1.01 – thus, with a mild amount of coordination, a realistic controller implementation can achieve performance nearly identical to that of a controller using the GOD architecture. Finally, the best myopic distributed controller we were able to synthesize achieved a cost ratio of 1.04 – these results are summarized in Table 1.

A representative example of flow deviation and admission buffer length evolution under the GOD and coordinated distributed controllers can be found in Figure 6 – the driving source rate deviations are illustrated in Figure 7. As there is very little quantitative difference in the performance of these two controllers, one does not expect to see a qualitative difference in the state trajectories.

Thus, in this scenario there is a significant quantifiable advantage in performance when adopting either a myopic or coordinated distributed architecture over a centralized architecture, indicating that the increased complexity in deployment and troubleshooting may be worthwhile. The difference in performance between the two distributed architectures considered is much less significant – in this case, it is up to the network designer to choose whether the additional complexity of implementing a coordinated algorithm is worth the 3% performance gain over the proposed myopic algorithm.

	GOD	Myopic	Coordinated	Centralized
Ratio	1	1.04	1.01	1.13

Table 1: Summary of admission control case study results for edge and internal admission control scenario.

Although the differences in performance in this case study ranged from 1% to 13%, in general, the gap between centralized and distributed architectures can become arbitrarily large as the network size (as measured in terms of end to end communication delays) increases. In particular, as the network size increases, the centralized approach requires an increasingly larger delay to collect and take global actions, leading to a corresponding degradation in performance. However, this is not true for the coordinated distributed architecture, as local actions are taken without delay, and the delay needed for two controllers to exchange information to coordinate their control actions is independent of the rest of the network.

4. SUMMARY

In this paper, we propose a methodology for quantifying the achievable performance of network architectures. We focused on problems in which the task is to keep the network state near a nominal operating point despite physically imposed delays and varying operating conditions. We showed how recent results in distributed optimal control theory allow for the efficient computation of *optimal* algorithms implemented using the GOD, centralized and coordinated distributed architectures, and proposed using the performance achieved by these controllers, as well as the performance achieved by a candidate myopic distributed algorithm, as a means of comparing the respective network architectures. We applied this approach to an admission control case study, and in particular, discovered that for the topology considered, the GOD policy can be implemented using a myopic distributed architecture when only edge admission control is allowed. However, if internal admission control is used, a coordinated distributed architecture leads to the best performance.

We believe that our proposed approach could play an important and complementary role to the traditional experimental validation of network architectures and algorithms by allowing network designers to narrow the range of promising architectures earlier in the design process, and by letting them quantifiably measure the effects on performance of other metrics such as scalability, ease of deployment and troubleshooting, and flexibility.

5. REFERENCES

[1] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, F. Matus, R. Pan, N. Yadav, G. Varghese, et al. CONGA: Distributed congestion-aware load balancing for datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 503–514. ACM, 2014.

[2] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE*, 95(1):255–312, 2007.

[3] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 15–26. ACM, 2013.

[4] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al. B4: Experience with a globally-deployed software defined WAN. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 3–14. ACM, 2013.

[5] F. Kelly and R. Williams. Fluid model for a network operating under a fair bandwidth-sharing policy. *Annals of Applied Probability*, pages 1055–1083, 2004.

[6] A. Lamperski and L. Lessard. Optimal state-feedback control under sparsity and delay constraints. In *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, pages 204–209, 2012.

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

[8] C. H. Papadimitriou and J. Tsitsiklis. Intractable problems in control theory. *SIAM Journal on Control and Optimization*, 24(4):639–654, 1986.

[9] 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*, pages 307–318. ACM, 2014.

[10] M. Rotkowitz, R. Cogill, and S. Lall. Convexity of optimal control over networks with delays and arbitrary topology. *Int. J. Syst., Control Commun.*, 2(1/2/3):30–54, Jan. 2010.

[11] M. Rotkowitz and S. Lall. A characterization of convex problems in decentralized control. *Automatic Control, IEEE Transactions on*, 51(2):274–286, 2006.

[12] H. S. Witsenhausen. A counterexample in stochastic optimum control. *SIAM Journal on Control*, 6(1):131–147, 1968.

[13] K. Zhou, J. C. Doyle, K. Glover, et al. *Robust and optimal control*, volume 40. Prentice Hall New Jersey, 1996.