Technical Report: A Case Study in Network Architecture Tradeoffs

Nikolai Matni (Caltech), Ao Tang (Cornell) and John C. Doyle (Caltech)

March 18, 2015

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 however 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.

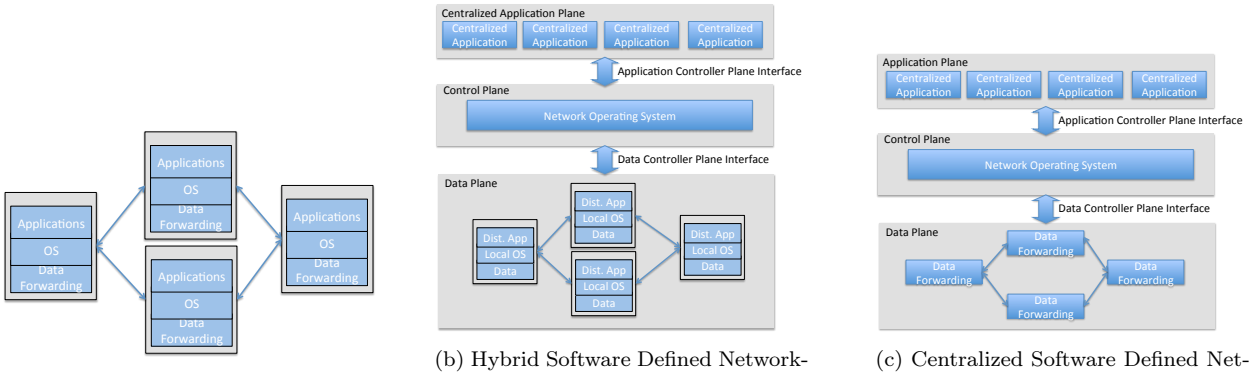
Using admission control as a case study and leveraging recent developments in distributed optimal control, this paper provides network designers with tools to quantitatively compare different architectures, allowing them to explore the relevant HySDN design space in a principled manner. In particular, 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 philosophy and approach proposed in this paper.

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., [4, 5, 14]). However, such centralized solutions are not always possible or desirable due to latency and scalability constraints – in particular the delay inherent in communicating the global state of the network, computing a solution to a global optimization problem and redistributing this 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]).

Network designers currently make architectural decisions based on qualitative reasoning and best practices gleaned from experience, often leading them to commit to one extreme of the network design space or the other, i.e., they often commit to either a completely centralized or a completely distributed network solution. The appropriateness of such an architectural decision is then only confirmed once a suitable algorithm has been developed and tested. 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 there is a broad class of network architectures that can be quantitatively compared in a principled manner. In particular, we define the broader *Hybrid Software Defined Networking*



(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.

(HySDN) design space (§2) of network architectures, and claim that a meaningful way to quantify the appropriateness of an architecture is to determine the optimal performance achievable by any algorithm implemented on that architecture. This metric thus allows network designers 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.

Of course, this approach is only practical if the optimal performance achievable by an algorithm implemented on a network architecture can in fact be computed efficiently, and indeed, up until recently, only a limited subset of centralized architectures admitted such an analysis. In §2, we further explain how recent advances in distributed optimal control theory allow us to expand our analysis to a much larger subset of the HySDN design space, and to in fact reason quantitatively (if not exactly) about architectures for which the optimal achievable performance is still computationally intractable to identify. In §4, 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 for 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 make quantitative decisions about architectural choices early in the design process.

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*, 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. Our approach to exploring

these tradeoffs is simple: we compare network architectures by comparing the optimal performance achievable by any algorithm implemented using them. If an optimal algorithm cannot be computed efficiently for a given architecture then we bound the optimal performance using a "nearby" architecture for which an optimal cost can be obtained.

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 §4, we consider admission control algorithms that aim to keep the link flow rates at a user-specified set-point 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 this problem in future work.

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 [21]. Further, it is known that myopic distributed optimal controllers are in general NP-hard to compute [20, 13]. 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 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 [16, 15]. 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 [16, 15]. This architecture takes both rapid action based on timely local information, and slower scale action based

[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 [6, 3].

[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 [15].

on 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.

By computing the performance of each of these architectures, the network designer can then quantify tradeoffs in implementation complexity and performance in a computationally efficient and inexpensive manner. We demonstrate the usefulness of this approach on an admission control case study in the next section.

3 Control Theory in Networking

Network applications use the architectural resources allocated to them to implement network logic (e.g., traffic engineering, network virtualization, processor/storage optimization, etc.): the objective of this network logic is to bring the network to a state that is optimal with respect to a performance metric by appropriately allocating network resources. It is therefore natural to view such network applications as *resource allocation controllers* – the resource allocation controller determines the optimal distribution of resources within a network so as to maximize a specified performance metric, and if applicable, a suitable means of distributing said resources.

For example, in the case of traffic engineering, given a set of source-destination pairs and associated rate demands, the resource allocation controller must first solve a multi-commodity flow optimization problem to determine how the link capacities should be allocated among the source-destination pairs. It must then further solve a suitable problem to determine routing protocols (be they path, source-destination or destination based) to achieve these optimal flow rates in the network.

Suppose now that the source-destination rate demands fluctuate in an unpredictable manner around the value used by the resource allocation controller to compute the optimal flow rates – it is desirable to be able to handle such fluctuations in a principled and practical manner. One such approach is to introduce admission controllers in between each source and the rest of the network, and to use these admission controllers to suppress the effect of these un-modeled rate demand perturbations on the flow rates of the network. These admission controllers are inherently *feedback based*, deciding their actions based on the measurements of the state of the network (namely the flow rates and size of admission control buffers) and their model of the network dynamics. In other words, we introduce fast time-scale feedback control in order to keep the link flow rates as near as possible to the optimal flow rates determined by the resource allocation controller, despite fluctuations in the demand rates.

This example serves to illustrate a general principle: the network applications determine an *optimal set point* at which the network should operate under simplifying assumptions. As these simplifying assumptions are often violated by the actual system, fast time scale perturbation controllers should then be introduced to maintain the state of the network at the determined optimal set point – as these perturbations are inherently unknown, feedback control is necessary.

3.1 Distributed Optimal Perturbation Control

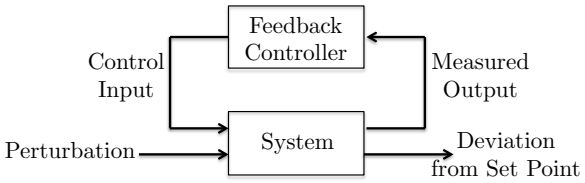


Figure 2: A centralized control problem

Figure 2 illustrates a standard, or centralized, optimal perturbation control problem: the objective is to minimize the deviations of the system from a desired set-point using control inputs that are computed using measurements of the state of the system. Mathematically, we seek the feedback policy K that minimizes the norm of the closed loop map from perturbations to deviations, i.e., the map from perturbation to deviation when the controller is connected to the system

$$\begin{aligned} \underset{\text{Feedback Policy } K}{\text{minimize}} \quad & \|\text{Closed Loop Response}\| \\ \text{s.t.} \quad & \text{System Dynamics.} \end{aligned} \tag{1}$$

Typical objective functions minimize the average effect of the perturbation on the deviations (known as $\mathcal{H}_2$ or LQG optimal control) or the worst case effect of the perturbation on the deviation (known as $\mathcal{H}_\infty$ optimal control). If the system is described by linear dynamics, then for such objective functions, it is known that the optimal control policies are *linear* in the measurements available to the controller and are specified by the solutions to *finite dimensional convex optimization problems*.

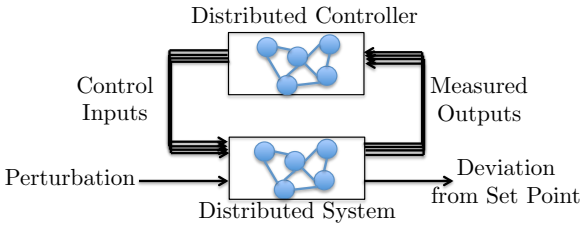


Figure 3: A distributed control problem

Notice however that Figure 2 does not fully capture the nature of the types of perturbation control problems that arise in a networking setting – in particular, both the system and the controller are spatially *distributed*, as illustrated in Figure 3 For example, in the admission control problem formulated in the next section, the feedback controller is actually comprised of several sub-controllers (each admission control buffer is such a sub-controller), and further, each sub-controller has access to a different subset of the measurements of the system. Even if sub-controllers are allowed to exchange their respective measurements, due to the communication delays inherent in such an information exchange, there is an *asymmetry* in the information available at each sub-controller.

This asymmetry manifests as additional constraints on the feedback policy in the optimal control problem (1), leading to the distributed optimal control problem

$$\begin{aligned} \underset{\text{Feedback Policy } K}{\text{minimize}} \quad & \|\text{Closed Loop Response}\| \\ \text{s.t.} \quad & \text{System Dynamics} \\ & \text{Information Exchange Constraints.} \end{aligned} \tag{2}$$

As described in the previous section, this seemingly innocuous modification to optimization problem causes traditional methods to fail, and in fact makes the optimal control policy NP-hard to compute in general [20, 13].

Fortunately, a class of systems for which the distributed optimal control problem can be solved via finite dimensional convex optimization has recently been identified [16]. Such systems are characterized by a particularly intuitive property: if sub-controllers can exchange information with each other at least as fast as their control actions propagate through the system, then the optimal control problem is convex [15]. This property ensures that control inputs are used only to control the system, and not as a means of communication between sub-controllers.

Although this delay based condition may seem restrictive, it can be satisfied in a networking setting by giving control dedicated packets priority – in this way, the communication delay between two sub-controllers is specified by the propagation delay associated with the shortest path between them.

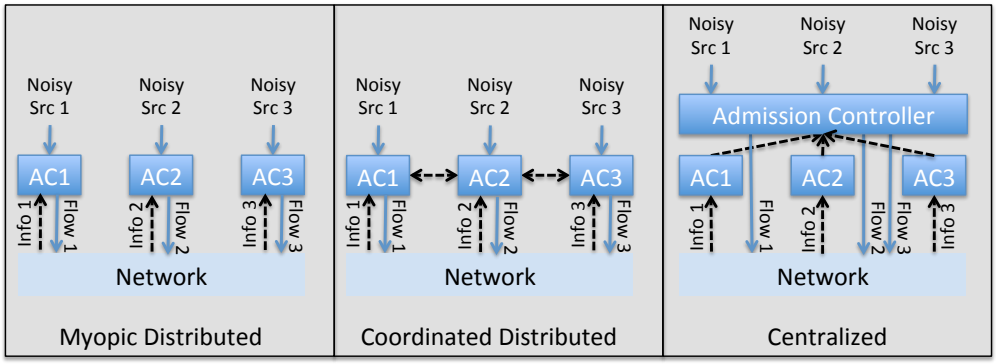


Figure 4: The HySDN design space for an admission control problem with three sources. Blue arrows denote the flow of traffic, whereas dashed black lines denote the flow of admission control related information. Note that the dotted lines correspond to virtual connections, and can be implemented using either control dedicated communication links, or using control dedicated tunnel overlays on the network.

4 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 §A of the Appendix for technical details.

4.1 Problem

We consider the following admission control task: given a set of source-destination pairs (s, d), a set of desired flow rates $\left(f_\ell^{s,d}\right)^\star$ 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 $\left(f_\ell^{s,d}\right)^\star$ 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 4 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 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

$$\sum_{t=1}^{N} \sum_{\ell}^{s,d} \left(f_\ell^{s,d}(t) - \left(f_\ell^{s,d}\right)^\star \right)^2 + \lambda \|A(t)\|_2^2, \tag{3}$$

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

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 6, whereas Source 3 either has an edge admission controller or an internal admission controller, as depicted in Figure 7, depending on the scenario considered.

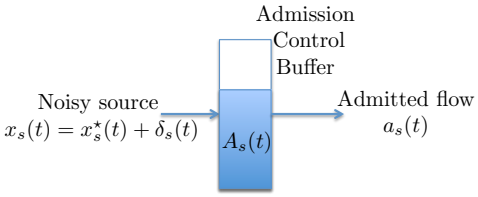


Figure 6: Diagram of an edge admission controller.

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.

4.2 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 6) and one in which edge and internal admission control is allowed (cf. Figure 7). We model the dynamics of the system using a flow based model and solve the optimal control problem with the cost (3) taken to be the infinite horizon LQG cost using the methods described in [21] and [7] – we refer the reader to Section A 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 $\left(f_\ell^{s,d}\right)^\star$ 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 4, 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 (recall that optimal myopic controllers are in general computationally intractable to compute).

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

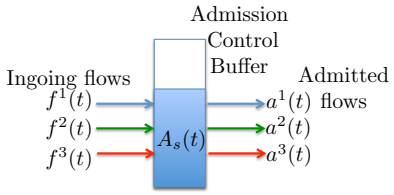


Figure 7: Diagram of an internal admission controller.

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 6, 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 7. 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 to be specified by the solution to a convex optimization problem, and can be computed efficiently using the methods in [7]. 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 coordinated 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 8, with the driving source rate deviations found in Figure 9. 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 adopting either a myopic or coordinated distributed architecture over a centralized architecture. 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

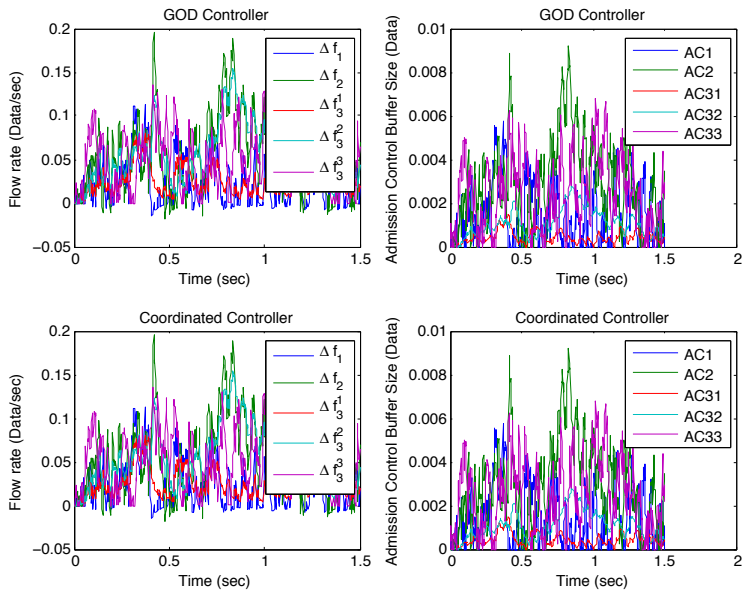


Figure 8: Sample flow deviations and admission buffer length evolution under the GOD and coordinated distributed controllers when the system is subject to the source rate fluctuations illustrated in Figure 9.

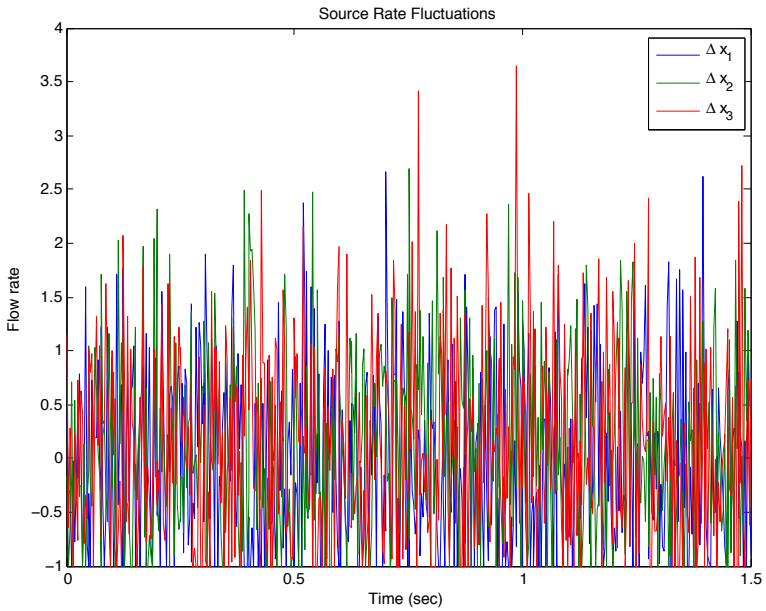


Figure 9: Sample source rate fluctuations – these are lower bounded by -1 as the nominal source rates are all assumed to be 1.

	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.

3% performance gain over the proposed myopic algorithm.

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 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.

5 Discussion

Localized control: We have not discussed scalability issues that arise in designing and implementing centralized and coordinated control strategies. As we argue in [19, 18, 17], both centralized and coordinated distributed optimal controllers require sharing global state information among decision makers – this can quickly become impractical for even networks of moderate size. To address this issue, we have developed the *localized optimal control* framework [19, 18, 17], in which the information collected by a local controller is limited to a local neighborhood. Although the details are beyond the scope of this paper, we believe that this theory will play an integral part in allowing optimal control methods to scale to networks comprised of thousands of links.

Determining nominal operating points: This paper focused on the architectural tradeoffs in designing controllers for tracking a nominal set point. However, an equally important problem is that of determining the optimal set points themselves in a rapid, scalable and optimal manner. The challenge inherent to this setting are the same as those faced in the tracking setting: one must trade off globally optimal performance with scalability and latency constraints. Much as recent progress in distributed optimal control theory [16, 15, 18, 11] has allowed us to explore different controller architectures in a principled manner, we believe that recent advances in distributed optimization [2, 8, 12] will allow for the research program initiated by the seminal work [3] to be further expanded, allowing for the relevant architectural space to be explored in a principled manner as well. Our ultimate aim is to combine the nominal set point and controller synthesis tasks into one integrated framework: we believe that this once incredibly daunting task is now within the reach of current theory.

Additional architecture design questions: Within the realm of controller design for networks, there natural architecture questions that we did not address in this paper. For example, how coordinated should a coordinated distributed controller be – i.e., how much information should be shared among local algorithms, and how quickly. Alternatively, one can ask where edge and internal admission controllers need to be placed to achieve a desired performance level – i.e., how much actuation needs to be present in the system, and where. Although not touched upon in this paper, the recently developed Regularization for Design [10, 9] framework provides the controller designer with a computationally efficient means of answering such inherently combinatorial questions in a principled manner.

6 Summary

In this paper, we propose a methodology for quantitatively comparing network architectures. In particular, we focused on network control problems in which the task is to keep the network state as close as possible to 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 that we considered, the GOD admission control policy can be implemented using a myopic distributed architecture when only edge admission control is allowed. However, if internal admission control is also allowed, a coordinated distributed architecture leads to the best performance.

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] 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*, 3(1):1–122, 2011.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *arXiv preprint arXiv:1408.3595*, 2014.

[9] N. Matni. Communication delay co-design in $\mathcal{H}_2$ distributed control using atomic norm minimization. *IEEE Transactions on Networked Control Systems, Submitted to the*, arXiv:1404.4911, 2015.

[10] N. Matni and V. Chandrasekaran. Regularization for design. *Submitted to the IEEE Transactions on Automatic Control*, 2015.

[11] N. Matni, A. Lamperski, and J. C. Doyle. Optimal two player LQR state feedback with varying delay. In *IFAC 2014, To appear.*, 2014.

[12] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A general analysis of the convergence of admm. *arXiv preprint arXiv:1502.02009*, 2015.

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

[14] 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.

[15] 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.

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

[17] Y.-S. Wang and N. Matni. Localized distributed optimal control with output feedback and communication delays. In *Communication, Control, and Computing, IEEE 52nd Annual Allerton Conference on*, 2014.

[18] Y.-S. Wang, N. Matni, and J. C. Doyle. Localized LQR optimal control. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 2014.

[19] Y.-S. Wang, N. Matni, S. You, and J. C. Doyle. Localized distributed state feedback control with communication delays. In *American Control Conference (ACC), 2014*, pages 5748–5755. IEEE, 2014.

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

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

A Model and Problem Formulation

This section formally defines the model used §4, and formulates the admission control problem as an optimal control problem subject to information constraints imposed by the chosen architecture (myopic, coordinated, centralized or GOD).

A.1 Modeling

We model the network as an interconnection of a set of sources $\mathcal{S}$, a set of links $\mathcal{L}$ and a set of switches $\mathcal{V}$. We associate to each source $s \in \mathcal{S}$ a unique source-destination pair (s,d) and a corresponding transmission rate $x_s(t)$ at time t. We denote the incoming flow on a link $\ell \in \mathcal{L}$ due to a source destination pair (s,d) at time t by $f_\ell^{s,d}(t)$, and write $\mathbf{f}_\ell(t)$ for the vector of such flows on a link ℓ, i.e., $\mathbf{f}_\ell(t) = \left(f_\ell^{s,d}(t)\right)_{(s,d):s\in\mathcal{S}}$. Under this convention, it follows that the total incoming flow on a given link ℓ at time t is given by $\mathbf{1}^\top \mathbf{f}_\ell(t)$: we denote this quantity by $\psi_\ell(t)$. Similarly, we denote the outgoing flow on a link $\ell \in \mathcal{L}$ due to a source destination pair (s,d) at time t by $g_\ell^{s,d}(t)$, we write $\mathbf{g}_\ell(t)$ for the vector of such flows on a link ℓ, and set $\phi_\ell(t) = \mathbf{1}^\top \mathbf{g}_\ell(t)$.

Let $\mathcal{L}^{\text{edge}} \subset \mathcal{L}$ denote the set of $|\mathcal{S}|$ links for which the incoming flow of a link $e \in \mathcal{L}^{\text{edge}}$ is determined directly by a source rate. Then for each edge link $e \in \mathcal{L}^{\text{edge}}$, it holds that

$$f_e^d(t) = \begin{cases} x_s(t) & \text{if } (s,d) \text{ is a source destination pair} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Let $\mathcal{L}_v^{\text{in}}$ and $\mathcal{L}_v^{\text{out}}$ denote the set of incoming and outgoing links, respectively, of a switch $v \in \mathcal{V}$. Each switch v is equipped with a set of non-negative and possibly time varying path routing split ratios $\alpha_{v,\ell}^{s,d}(t)$ satisfying

$$\sum_{\ell \in \mathcal{L}_v^{\text{out}}} \alpha_{v,\ell}^{s,d}(t) = 1.$$

Under this routing protocol, for every incoming link $\ell \in \mathcal{L}_v^{\text{out}}$ and source-destination pair (s,d), it holds that

$$f_\ell^{s,d}(t) = \alpha_{v,\ell}^{s,d}(t) \left[\sum_{k \in \mathcal{L}_v^{\text{in}}} g_k^{s,d}(t - \delta_v - \delta_k) \right], \tag{5}$$

where δ_v is the computation delay for switch v, and δ_k is the propagation delay for link k.

Each link $\ell \in \mathcal{L}$ has a specific capacity c_ℓ, and is further equipped with a buffer: we denote the buffer state at a link ℓ at time t by $B_\ell(t)$. The buffer state evolves according to the dynamics given by

$$\dot{B}_\ell(t) = \begin{cases} \psi_\ell(t) - c_\ell & \text{if } B_\ell(t) > 0 \\ [\psi_\ell(t) - c_\ell]_+ & \text{if } B(t) = 0. \end{cases} \tag{6}$$

When the buffer state is empty, the incoming and outgoing flows on a link ℓ are equal, i.e., $\mathbf{g}_\ell(t) = \mathbf{f}_\ell(t)$ if $B(t) = 0$. When the buffer is non-empty, then the relationship is dependent on the buffering protocol in place – without loss of generality we write $\mathbf{g}_\ell(t) = \gamma_\ell(\mathbf{f}_\ell(t), B_\ell(t))$ when $B_\ell(t) > 0$, for some function γ_ℓ that

encodes the buffering protocol. Thus the dynamics of the network are fully specified by equations (4), (5), (6) and

$$\mathbf{g}_\ell(t) = \begin{cases} \gamma_\ell(\mathbf{f}_\ell(t), B_\ell(t)) & \text{if } B_\ell(t) > 0 \\ \mathbf{f}_\ell(t) & \text{if } B(t) = 0. \end{cases} \tag{7}$$

A.2 Admission Control Problem Formulation

We consider two types of admission controllers, edge admission controllers and internal admission controllers.

A.2.1 Edge admission control

To a link $e \in \mathcal{L}^{\text{edge}}$ we can associate an edge admission control buffer (as illustrated in Figure 6) $A_e(t)$ and admission rate $a_e(t)$ such that the admission buffer and edge link flow rate are specified by

$$\begin{aligned} \dot{A}_e(t) &= \begin{cases} x_s(t) - a_e(t) & \text{if } A_e(t) > 0 \\ [x_s(t) - a_e(t)]_+ & \text{if } A_e(t) = 0 \end{cases} \\ f_e(t) &= a_e(t). \end{aligned} \tag{8}$$

Notice in particular that if $a_e(t) = x_s(t)$ for all t then the edge admission controller is "invisible" to the network and we revert to the dynamics described in the previous section.

A.2.2 Internal admission control

To a switch v we can associate an internal admission control buffer (as illustrated in Figure 7). Let $f_v^{s,d}(t) = \sum_{\ell \in \mathcal{L}_v^{\text{in}}} f_\ell^{s,d}(t)$. We model an internal admission control buffer as having a dedicated internal state and control action for each source-destination pair flowing through it. In particular, for each (s,d) pair that utilizes a link $\ell \in \mathcal{L}_v^{\text{in}}$, we define the following admission control buffering dynamics

$$\dot{A}_v^{s,d}(t) = \begin{cases} f_v^{s,d}(t) - a_v^{s,d}(t) & \text{if } A_v^{s,d}(t) > 0 \\ \left[f_v^{s,d}(t) - a_v^{s,d}(t)\right]_+ & \text{if } A_v^{s,d}(t) = 0. \end{cases} \tag{9}$$

For each $k \in \mathcal{L}_v^{\text{out}}$ it then follows that

$$f_k^{s,d}(t) = \alpha_{v,k}^{s,d}(t) a_v^{s,d}(t). \tag{10}$$

Analogously, we note that if $a_v^{s,d}(t) = f_v^{s,d}(t)$ for all t, then the internal admission controller is also "invisible" to the network, and we revert to the dynamics described in the previous section.

A.2.3 Discrete Time Linearized Model

In what follows, we show how the dynamics described above can be appropriately linearized and used to formulate optimal control problems. For simplicity we describe the setting in which all admission controllers are edge controllers – an analogous argument applies to when there are both edge and internal admission controllers.

We impose that measurements of flow rates and admission control buffers be available to controllers only after suitable delays as specified by the topology of the system. By allowing control dedicated packets, i.e., packets used to communicate with the admission buffer controllers, priority in the network, we ensure that the resulting distributed optimal control problems are tractable to solve.

We assume that we are given nominal source rates $x_s^\star$, nominal flow rates $\mathbf{f}_\ell^\star$ (which we assume to satisfy the capacity constraints of each link), and suitable routing split ratio parameters $\alpha_{v,\ell}^{s,d}$ to achieve such flows. Our objective is to determine a feedback control law for each admission control buffer so as to minimize deviations from the optimal flow rates $\mathbf{f}_\ell^\star(t)$ and the size of the admission control queues. Note that as these nominal flow rates are assumed to satisfy the capacity constraints of each link, this in turn prevents congestion in the network.

To that end, we linearize the system around the buffers being empty (i.e., around $B_k(t) = 0$ for all $k \in \mathcal{L}$), and around the nominal flow and source rates $\mathbf{f}^\star_\ell(t)$ and $x^\star_s(t)$. As the these flows/rates are feasible, there is no need for admission control and hence we linearize around $a^\star_e(t) = x^\star_s(t)$ and $A^\star_e(t) = 0$. Letting $\Delta f^{s,d}_\ell(t) := f^{s,d}_\ell(t) - \left(f^{s,d}_\ell\right)^\star(t)$, $\Delta x_s(t) = x_s(t) - x^\star_s(t)$ and $\Delta a_e(t) = a^\star_e(t) - a_e(t)$, the resulting dynamics are then given by

$$\begin{aligned} \Delta f^{s,d}_\ell(t) &= \begin{cases} \alpha^{s,d}_{v,\ell}(t) \sum_{k \in \mathcal{L}^{\text{in}}_v} \Delta f^{s,d}_k(t - \delta_v - \delta_k) & \\ & \text{if } \ell \notin \mathcal{L}^{\text{edge}} \\ a_e(t) \text{ if } \ell \in \mathcal{L}^{\text{edge}} & \end{cases} \\ \dot{A}_e(t) &= \Delta x_s(t) - \Delta a_e(t). \end{aligned} \tag{11}$$

In order to apply the relevant tools from distributed optimal control theory [7], we require a discrete time model – this is consistent with practical considerations as well, as control laws are ultimately implemented using digital devices. To that end, let τ be the largest number such there exists a natural number n_{kv} satisfying

$$n_{kv}\tau = \delta_v + \delta_k \tag{12}$$

for every incoming link switch pair (k, v). The number τ then corresponds to the largest sampling time with which we can discretize the dynamics of the system while still exactly preserving its delay characteristics. We use a simple first order hold method to discretize the admission control buffer dynamics (this is indeed a sound approach for small sampling times) – the resulting discrete time linearized system is then governed by the difference equations:

$$\begin{aligned} \Delta f^d_\ell(n) &= \begin{cases} \alpha^d_\ell \sum_{k \in \mathcal{L}^{\text{in}}_v} \Delta f^d_k(n - n_{kv}) & \text{if } \ell \notin \mathcal{L}^{\text{edge}} \\ a_e(n) & \text{if } \ell \in \mathcal{L}^{\text{edge}} \end{cases} \\ A_e(n+1) &= A_e(n) + \tau \left(\Delta x_s(n) - \Delta a_e(n)\right), \end{aligned} \tag{13}$$

where n is the discrete time index satisfying $t = \tau n$.

Let $\Delta \mathbf{f}^{s,d}_\ell(n) = \left(f^{s,d}_\ell(n), f^{s,d}_\ell(n-1), \ldots, f^{s,d}_\ell(n - n_{\ell v})\right)^\top$, $\Delta \mathbf{f}_\ell(n) = \left(\Delta \mathbf{f}^{s,d}_\ell(n)\right)_{(s,d)}$ and $\Delta \mathbf{f}(n) = (\Delta \mathbf{f}_\ell(n))_\ell$. Further let $\mathbf{A}_e(n) = (A_e(n))_e$, $\Delta \mathbf{a}(n) = (\Delta a_e(n))_e$, and $\Delta \mathbf{x}(n) = (\Delta x_s(n))_s$. Then there exists a unique routing matrix $R(t)$ and source matrix S such that

$$\begin{aligned} \Delta \mathbf{f}(n+1) &= R(t)\Delta \mathbf{f}(n) + S\Delta \mathbf{a}(n) \\ \mathbf{A}_e(n+1) &= \mathbf{A}_e(n) + \tau \left(\Delta \mathbf{x}(n) - \Delta \mathbf{a}(n)\right) \end{aligned} \tag{14}$$

is compatible with the dynamics described in equation (13).

One can then formulate an optimal control problem that aims to minimize these perturbations around the desired set point as

$$\begin{aligned} \underset{\Delta \mathbf{a}(t)}{\text{minimize}} \quad & \textstyle\sum_{n=1}^{N} \text{cost}\left(\Delta \mathbf{f}(n), \mathbf{A}(n), \mathbf{\Delta a}(n)\right) \\ \text{s.t.} \quad & \text{dynamics (14)} \\ & \text{information exchange constraints} \end{aligned} \tag{15}$$

where N is the horizon of the optimization, $\text{cost}(\cdot)$ is a suitably chosen cost function, and the information exchange constraints enforce the distributed nature of the controller.

In order to leverage recently developed distributed optimal control techniques [7], we consider quadratic costs of the form

$$\text{cost}\left(\Delta \mathbf{f}(n), \mathbf{A}(n), \mathbf{\Delta} a(n)\right) = \Delta \mathbf{f}(n)^\top Q_f \Delta \mathbf{f}(n) + \mathbf{A}(n)^\top Q_A \mathbf{A}(n) + \mathbf{\Delta a}(n)^\top R \mathbf{\Delta a}(n), \tag{16}$$

where Q_f, Q_A and R are symmetric and positive-semidefinite matrices, and at least one of Q_A or R are strictly positive definite (this latter condition is required for the optimization problem to be well posed). We typically enforce that Q_A be positive definite and set $R \approx 0$, as there is no natural interpretation to the cost of non-zero $\Delta \mathbf{a}(n)$.

Information exchange constraints dictate what information each admission control buffer has access to when deciding what action it should take. As mentioned in §4, we are concerned with myopic, coordinated,

centralized and GOD controllers. We briefly summarize the mathematical implications of each of these architectures, and refer the reader to §4 for intuitive explanations for how these constraints arise. In order to lighten notational burden, we let $\mathbf{z}(n) = (\Delta\mathbf{f}(n), \mathbf{A}(n))^\top$ denote the *state* of the system (14), $\mathbf{z}(0:n)$ denote the history of the state from time 0 to n, i.e., $\mathbf{z}(0:n) = (\mathbf{z}(0), \ldots, \mathbf{z}(n))$, and β be a generic control policy that maps measurements to control actions.

1. **GOD:** The GOD controller has instantaneous access to the state of the system, and hence its control actions take the form
$$\Delta\mathbf{a}(n) = \beta(\mathbf{z}(0:n)), \tag{17}$$
for some function β.

2. **Centralized:** Let d be the delay induced by a centralized implementation. The corresponding constraint on the admission control policies in the optimal control problem (15) is given by
$$\Delta\mathbf{a}(n) = \beta(\mathbf{z}(0:n-d)), \tag{18}$$
for some function β.

3. **Coordinated:** We treat each admission control buffer A_e as a local controller that seeks to compute an admission policy of the form
$$\Delta a_e(n) = \beta_e(\Delta\mathbf{f}_1(0:n-n_{e1}), \ldots, \Delta\mathbf{f}_L(0:n-n_{e|\mathcal{L}|}), A_1(0:n-m_{e1}), \ldots, A_A(0:n-m_{e\alpha})) \tag{19}$$
where $L = |\mathcal{L}|$, α is the number of admission control buffers, $n_{e\ell}$ specifies the delay in communicating the flow rate $\mathbf{f}_\ell$ on link ℓ to admission control buffer A_e, and m_{ef} specifies the delay in communicating the admission control buffer state A_f to admission control buffer A_e.

A.3 Case study data

For the case study used in §4, a sampling time of $\tau = .003$ seconds was used. The source noise terms $\Delta x_s(t)$ were taken to be independently and identically distributed, with each $\Delta x_s(t) \sim \mathcal{N}(0, .1)$. The cost was taken to be
$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N} \mathbb{E}\left[\|\Delta\mathbf{f}(n)\|_2^2 + 50\|\mathbf{A}(n)\|_2^2\right], \tag{20}$$
and the information constraints for the respective architectures were taken to be consistent with the propagation delays imposed by the network topology. In particular,

1. **GOD:**$\Delta\mathbf{a}(n) = \beta(\mathbf{z}(0:n-1))$, for some linear map β.

2. **Centralized:** $\Delta\mathbf{a}(n) = \beta(\mathbf{z}(0:n-6))$, for some linear map β.

3. **Coordinated:**
$$\begin{aligned}
\Delta a_1(n) &= \beta_1(\Delta\mathbf{f}_1(0:n-1), \Delta\mathbf{f}_2(0:n-2), \Delta\mathbf{f}_2(0:n-3), A_1(0:n-1), A_2(0:n-2), A_3(0:n-3)) \\
\Delta a_2(n) &= \beta_2(\Delta\mathbf{f}_1(0:n-2), \Delta\mathbf{f}_2(0:n-1), \Delta\mathbf{f}_2(0:n-2), A_1(0:n-2), A_2(0:n-1), A_3(0:n-2)) \\
\Delta a_3(n) &= \beta_3(\Delta\mathbf{f}_1(0:n-3), \Delta\mathbf{f}_2(0:n-2), \Delta\mathbf{f}_2(0:n-1), A_1(0:n-3), A_2(0:n-2), A_3(0:n-1))
\end{aligned}$$
for some suitable linear maps β_1, β_2 and β_3.

Note that we have assumed a local computation delay of 1 discrete time-step, implying that all controllers are strictly proper – this assumption can be suitably modified or removed, and qualitatively analogous results are still obtained. The resulting optimal control problems are then all of a form amenable to the methods presented in [7].

We assume that the link capacities are sufficiently large that congestion is not an issue, and hence are able to ignore the buffer dynamics (6). For the simulations presented in Figure 8, the linear controller is applied to the non-linear model described by equations (4), (5), (6) and (7) using appropriate admission control dynamics, as specified in (8) and (9).