Eliminating Adverse Control Plane Interactions in Independent Network Systems

Matthew K. Mukerjee

Overview

Network system operation is typically divided into control and data planes—while the data plane is responsible for processing individual messages or packets, the control plane computes the configuration of devices and optimizes system-wide performance. Unfortunately, the control plane of each network system or protocol layer typically operates independently (e.g., CDNs selecting servers without coordinating with ISPs), resulting in poor interactions between control planes across systems. I categorize existing systems into one of four general control plane coordination mechanisms that overcome these problems, based on how much information can be shared between control planes. If no information can be shared, control planes simply react to data plane changes as a rudimentary form of coordination (e.g., CDN server selection + ISP traffic engineering). If all information can be shared, transparency in decision making can remove most poor interactions (e.g., Coflow datacenter flow scheduling). In many scenarios, however, only some information can be shared (e.g., between control planes running in different companies). Coordination in these scenarios is more specialized; control planes with separate data plane resources can use priority ranking (i.e., providing a list of preferences for resources without needing to show how these preferences were decided; e.g., BGP routing between ISPs), and control planes with shared data plane resources can use hierarchical partitioning (i.e., making coarse-grained decisions globally, and fine-grained decisions locally; e.g., internet-wide BGP + OSPF routing).

While systems utilizing control plane coordination exist today, they have been designed in an ad hoc fashion. We propose a set of recipes that show when it’s appropriate to use different coordination mechanisms, based on key properties (information sharing and shared resources) in varied scenarios (layering, administrative separation, and internet-scale systems).

We use these recipes to guide system design in a variety of contexts, as a case study in control plane coordination. First, many systems use layer separation for modularity, but in doing so trade performance for generality. As layered systems have no information sharing constraints, we argue that transparency (i.e., cross-layer optimization; allowing layers to run specialized code to better use other layers) is the correct technique for regaining performance. We explore this with our emulator Etalon [8], in the context of reconfigurable datacenters. Second, some systems are administratively separate (i.e., are split across different companies), limiting information sharing for business reasons. These systems may overcome adverse interactions using priority ranking. We explore this with VDX [5, 6], in the context of content brokering. Finally, systems needing internet scalability are increasingly combining a slow centralized control plane with a fast distributed control plane. Complete information sharing (and thus, transparency) would appear possible, but timescale separation makes this fundamentally impractical. Instead, hierarchical partitioning can overcome the challenges present in this scenario. We explore this with VDN [9], in the context of live video delivery. Through this case study we find that these coordination mechanisms not only solve a variety of problems, but can be efficiently implemented.
1 Thesis Work: Overcoming Competition across Control Planes

Competing Control Planes in Different Layers: Etalon (reconfigurable datacenters) [in submission]

As datacenter (DC) networking demands have increased, CMOS manufactures have struggled to build switches with simultaneously higher bandwidth and port count. Thus, researchers have proposed augmenting DCs with very high bandwidth reconfigurable circuit technologies to add bandwidth on demand. While prior work have mainly focused on switch design or network scheduling, little focus has been paid to end-to-end challenges and their solutions.

In this work, currently in submission [8], I identify three key end-to-end challenges: 1) poor TCP performance caused by rapid bandwidth fluctuation, 2) inefficient schedules caused by poor demand estimation, and 3) fundamentally difficult-to-schedule workloads caused by application demand. These challenges arise from transport-/application-layer control planes making assumptions about the network-layer control plane (e.g., network scheduling), which do not hold in reconfigurable DCs. There is nothing stopping system designers from making these assumptions explicit using 
transparency, thereby trading the modularity of strict layering for the high performance of cross-layer optimization.

Solution Using cross-layer optimization, I solve these problems directly with: 1) dynamic in-network queue resizing to mask bandwidth fluctuations, 2) proper demand estimation using endhost stack ADUs, and 3) rewriting application logic for easier-to-schedule demand. I build Etalon1 an open-source reconfigurable DC emulator, to evaluate these solutions, and find they improve transport-/application-layer metrics by as much as 9x.

Competing Control Planes in Different Companies: VDX (content brokering) [HotNets '16; CoNEXT '17 (best paper)]

Internet video delivery is undergoing a fundamental change; content providers are shifting delivery from a single CDN to multiple CDNs, using a content broker. While watching a video, client video players periodically contact the broker to find the “best” CDN for a user based on ISP, device type, geographic location, etc.

In joint work with a broker (Conviva) and a CDN (Akamai) at Hotnets ’16 [5] and CoNEXT ’17 (best paper winner) [6], I provide data-driven analysis of problems CDNs and brokers cause one another due to a lack of communication. The best way to solve the problem would be transparency, but CDNs and brokers wish to avoid sharing as much information as possible, as they are separate companies. As CDNs and brokers make decisions for different data plane resources, providing a priority ranked list of options (a type of incomplete information sharing) for the other company provides the ideal middle ground, giving flexibility without needing companies to explain why they made their decisions.

Solution I address the problems seen by designing a CDN-broker interface, Video Delivery eXchange (VDX), inspired by online bidding (i.e., advertising exchanges). Only minimal information is shared between CDN and broker control planes; CDNs provide a set of “bids” for groups of clients in priority ranking to the broker, which decides which bids to accept. I show the efficacy of VDX through CDN-scale data-driven simulations, finding VDX can decrease CDN cost by 32%, while assigning clients to servers that are 40% closer. More importantly, VDX provides CDNs of varying sizes or deployments more opportunities to profit on brokered video delivery.

Competing Control Planes at Internet Scale: VDN (live video) [SIGCOMM ’15]

CDN-based internet-scale live video streaming requires high bandwidth, synchronized real-time delivery. Slow centralized control over the distribution has been deemed impractical due to the need for quick responses

1 http://www.github.com/mukerjee/etalon

2
to new user video startup and failures, given internet-scale latency. Fast distributed control, while amenable
to many of these needs, does not provide the user performance of a centralized approach. CDNs currently
choose availability over quality, using a distributed approach. Using both approaches together has not been
previously considered, as competition between them (i.e., which decision should be used if they disagree)
make this challenging. Although both control planes are run by the same company, timescale separation (due
to the slow centralized optimization and internet latency) limits what information can be shared between the
control planes, making transparency impossible. As both control planes make decisions for the same data
plane resources, hierarchical partitioning (a type of incomplete information sharing) can be used.

Solution I design a system, Video Delivery Network (VDN), appearing in SIGCOMM ’15 [9], that provides
the best of both centralized and distributed control using hierarchical partitioning. At a long timescale,
centralized control optimizes for user performance, while (simultaneously) at a short timescale, distributed
control handles failures and new user joins. The centralized control decision has priority over the distributed
control decision (i.e., “hierarchy”), but slack in the decision allows distributed control to react to changes
in its local region (i.e., its “partition”). Using large-scale simulation and a wide-area testbed, I show that
VDN can offer ∼2× improvement in quality over today’s distributed system and ∼100ms join times, while
providing CDN operators expressive policy management.

2 Other Work

I believe that working with many collaborators provides insights into different research styles, and ultimately
helps one synthesize their own. To this end, I’ve worked closely with as many faculty members as possible
on different papers, ranging from datacenter networking [8, 4, 3], network architecture [11, 7, 10], content
delivery [6, 5, 9], mobile networking for robotics [12], EEG use for mobile [1], network emulation [2], and
heterogeneous wide-area TCP [13]. I describe some of these efforts below:

Reconfigurable Datacenters

Solstice [CoNEXT ’15, (best paper nominee)]: Prior work on reconfigurable datacenter (DC) net-
works focused on hybrid switch design, assuming an oracle perfectly schedules demand to circuits. A proper
scheduling algorithm is needed to build a working reconfigurable DC network. Traditional scheduling al-
grithms like Birkoff-von Neumann decomposition, however, do not take circuit reconfiguration delay into
account and thus produce poor schedules. I develop a heuristic-based scheduling algorithm, Solstice [4], that
uses the skew and sparsity commonly found in DC workloads to provide 3× more circuit utilization than
traditional algorithms while being within 14% of optimal, at scale.

Albedo [ANCS ’17]: Given a scheduling algorithm for reconfigurable DC networks like Solstice, how does
one recover from errors in demand estimation? Albedo [3] uses indirect routing (i.e., sending data to the
wrong destination before eventually reaching the right destination) to recover about half of the error.

Network Architecture

Understanding Incremental Deployment of Network Architectures [CoNEXT ’13]: Many pro-
posed network architectures are fundamentally incompatible with IPv4, requiring a “flag day” where all
endhosts and routers are upgraded simultaneously. This requirement is untenable; incremental deployment
is a necessity if an architecture is going to be feasible in the real world. In this work [7], I provide a frame-
work for network architecture designs by distilling incremental deployment down to four key problems, and
examine a variety of mechanisms that solve these problems.

eXpressive Internet Architecture [CCR ’14]: The eXpressive Internet Architecture (XIA) project [10]
is part of a large-scale NSF initiative to design, implement, and evaluate a clean-slate internet architecture.
XIA focuses on evolvability at the network layer by providing an extensible set of communication primitives
(e.g., hosts, services, content), in addition to flexible routing and intrinsic security.
Accountable and Private Internet Protocol [SIGCOMM ’14, (best paper)]: In the current internet it is very difficult to keep communication private (i.e., prevent on-path ISPs from knowing who talked to whom), while simultaneously providing accountability (i.e., knowing which machine launched an attack). I develop a new network architecture, APIP [11] to address this problem. The key insight is that network source IP addresses are overloaded, serving both as a return address as well as an accountability address. By separating these into individual return addresses (which can be encrypted) and accountability addresses (opaque identifiers known only by third-party trusted accountability delegates, e.g., ISPs, Symantec), both properties can be provided.

3 Future Work

While I am interested in many areas of networking, as evidenced by my various publications, I am currently most interested furthering work in reconfigurable datacenters (DCs), as well as generalizing ideas in this space to the wider problem of network / endhost co-design.

Reconfigurable datacenters: While fairly well studied over the past decade, there are still fairly fundamental open problems in reconfigurable DCs. There has yet to be an “apples-to-apples” comparison of different switch designs in terms of technologies (e.g., optical, 60GHz, free-space optics), features/limitations (e.g., circuits can be perfect matchings only, indirection routing, multicast), and timescales (e.g., millisecond reconfiguration delay, microsecond, future nanosecond). With Etalon [8] I’ve shown that the nature of end-to-end challenges change based on these features; investigating which combinations provide optimal properties, using my open-source emulator, can provide a principled approach to reconfigurable switch design, rather than just applying trendy new technology. More specific open problems are also interesting, e.g., remote direct memory access (RDMA) has greatly improved DC performance by removing the remote CPU from the critical path. What changes are needed in reconfigurable DCs to support RDMA or RDMA-like primitives? Can machine learning-based scheduling algorithms seen at the application-layer (e.g., for MapReduce tasks) apply to circuit scheduling in reconfigurable DCs? What challenges arise from the 6 orders of magnitude shift in timescale, and does the resulting scheduler look similar to prior (hand-built) schedulers like my work Solstice [4]?

Network / endhost co-design: My work on Etalon [8] provides a cursory glance at end-to-end challenges that arise if fundamental assumptions about the network change without making changes to the endhost stacks and applications. This need for network / endhost co-design appear in a variety of scenarios. A simple example would be adding multicast to DC networks (as has been proposed for reconfigurable DCs for example); not only does this break assumptions for in-network packet / flow scheduling algorithms, it also provides new performance opportunities for low-level scheduling and for applications (e.g., HDFS could cut replication traffic in half using multicast). I point out with Etalon that making applications “network-aware” can greatly improve performance, but this currently requires manual application code modification; are there better network primitives (e.g., anycast, multicast) that developers can build into applications to help automate network-awareness? Furthermore, some DC network designs propose combining multiple heterogeneous networks. While multipath TCP has been explored with heterogeneous paths in the wide-area, its impact has yet to be explored over heterogeneous paths at DC timescales. Finally, the advent of re-programmable FPGA-based NICs and re-programmable switches (e.g., P4-based or Mellanox Spectrum Linux-based) provides a strong indicator that changes to network functionality is imminent, thus a strong understanding of network / endhost co-design will be required to overcome the challenges laid out in my thesis.
References


