Towards Deep Network & Application Integration: Possibilities, Challenges, and Research Directions

Danny Lachos
University of Campinas

Qiao Xiang
Yale University

Christian Rothenberg
University of Campinas

Sabine Randriamasy
Nokia Bell Labs

Luis M. Contreras
Telefonica

Börje Ohlman
Ericsson

ABSTRACT

The collaboration between networks and applications provides opportunities to both applications to improve their performances and network service providers to increase business offering. Although many systems are proposed to support such collaborations, they are point or incremental solutions. In this paper, we propose the exploration of a more integrated architecture with huge possibilities taking a network-application integration (NAI) approach. Specifically, we explore the NAI possibilities in two concrete aspects: application-aware networking and network-aware applications. We review recent progress in these two aspects, and identify the key challenges in systematically realizing such a deep integration. To address these challenges, we present the initial design of PED, a generic NAI possibilities exposure and discovery framework based on satisfiability modulo theories (SMT). The key components of PED include a unified, abstract representation of network information using mathematical programming constraints, a declarative language for applications to express their intents on discovering network information, and an efficient compiler to translate application intents to constraint programming problems and discover corresponding network information. Preliminary evaluation results demonstrate the potentials of the PED framework. At the end of the paper, we also discuss a series of key future research directions toward deep NAI.

1 INTRODUCTION

The collaboration between networks and applications increases the quality and hence the business offering of the former, and the performance of the latter [21, 26]. Data-intensive science applications (*e.g.*, the large hadron collider, telescopes, and light sources), for instance, rely on networks as one of the key components of their infrastructure for local and global interconnection of laboratories, sites, and data centres [4]. Another unexpected but evident example is the current COVID-19 pandemic, with many institutional applications taking advantage of the network infrastructure to share data quickly and support collaborative efforts from multiple communities and disciplines such as medicine, health, genomics, and disaster mitigation [17]. Flexible inter-domain routing and End-to-End (E2E) network services are also emerging applications that construct complex data flows between users in the network [10, 25].

Different systems and mechanisms have been proposed to support such collaboration. However, they are point or incremental solutions with various limitations. For example, network providers and applications have considered different nash equilibrium solutions (See Fig. 1a). ISPs, for example, attempt to improve the application issues through an infrastructure upgrade, usage-based charging model, rate limiting, or termination of services [16]. Meanwhile, applications attempt to improve the network efficiency having flexibility in shaping communications patterns as well as having flexibility to adapt to network topologies and conditions [1, 5, 8, 15]. However, such solutions are largely application/network-oblivious, making the interaction between them inefficient.

In addition, solutions adopting either a "best-effort" [3, 22] or "blackbox-request" [5, 27] approach (See Fig. 1b) are also proposed. In the first one, applications submit complete network requirements, and the network computes and enforces the optimal resource allocation for applications. In the second one, applications submit the amount of network resources needed, and the network returns success or failure based on the resource availability. These solutions either have limitations on the privacy of applications and scalability, or have inefficiency in finding the optimal resource allocation for applications, respectively.

In this paper, we propose to explore a more integrated and coherent architecture that takes a deep network-application integration (NAI) approach (See Fig. 1c). Specifically, we explore the possibilities of NAI in two concrete aspects: application-aware networking and network-aware applications. The first one allows applications to specify diverse requirements for the network infrastructure. The second one allows networks to expose underlying network information available to applications.

Despite the huge possibilities of NAI, systematically realizing it is non-trivial. The key challenge is the lacking of generic mechanisms for exposure and discovery of NAI possibilities. Existing solutions either fail to provide accurate

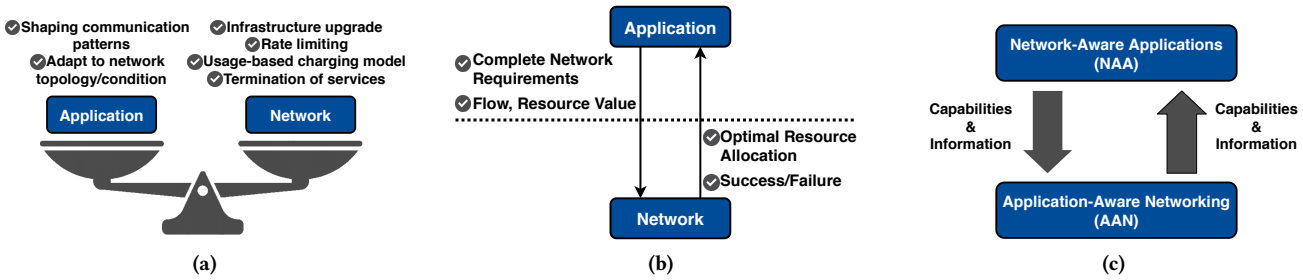


Figure 1: Different approaches for the interaction of networks and applications: (a) nash equilibrium point, (b) best-effort/black-box approach, and (c) network-application integration (NAI) Approach.

resource sharing information [12, 26], or expose the complete information of the network [11, 24], raising scalability and security concerns.

To fill this gap, we develop PED, a generic NAI possibilities exposure and discovery framework based on satisfiability modulo theories (SMT). The core of PED is the use of mathematical programming constraints as a unified abstract representation of network information. Second, PED provides a declarative language for applications to express intents on discovering network information. An efficient compiler is also developed to translate an application intent to a constraint programming problem and discover the corresponding network information.

The **main** contributions of this paper are as follows:

- We conduct a systematic review of the large variety of possibilities in designing and implementing NAI by application-aware networking and network-aware applications (Sections 2 and 3);
- We identify the key challenges for systematically realizing NAI, and present the initial design and evaluation of PED, a generic NAI possibilities exposure and discovery framework based on SMT (Section 4);
- We discuss a series of future research directions toward deep NAI (Section 5).

2 POSSIBILITIES OF NAI: APPLICATION-AWARE NETWORKING

Applications have varying needs for network latency, bandwidth, packet loss, etc. However, such applications' requirements are often unknown to the network due to applications and networks are decoupled. Thus, one concrete aspect of NAI is adding application knowledge to the network so that applications can express finer granularity requirements.

There are substantial possibilities in designing and implementing NAI by application-aware networking. For example, the network infrastructure can provide better support for applications introducing different capabilities. Table 1 shows a set of transport differentiation capabilities for applications and the newer trend where applications can also provide in-network computation or in-network storage.

Research contributions. Several research activities have been proposed exploring the possibilities of adding application knowledge to the network layer [7, 14, 21]. Magellan [7], for instance, is a programming environment for users to specify a global packet/in-network processing logic which is expressed in a general-purpose language. Then, Magellan automatically generates both datapaths in every single network device and runtime for control plane. Schmidt *et al.* [21] introduce *Socket Intents* as a proactive, application-expressed approach for multi-access network connectivity. *Socket Intents* allow applications to share information, in a generic way, about their communication patterns such as preferences (*e.g.*, bandwidth optimization), characteristics (*e.g.*, expected packet rates), expectations (*e.g.*, paths availability), and resiliences (*e.g.*, handle certain error cases). Application-aware IPv6 Networking (APN6) [14] proposes a framework for using IPv6 extensions header to convey the service/application requirements along with the packet to the network. The application awareness introduced by APN6 can benefit different use cases, such as application-aware SLA guarantee, application-aware network slicing, and application-aware network measurement.

Real deployment examples. BigData Express [6] is a data transfer service for big data science. It provides an application-aware SDN-enabled network service to program networks with fast provisioning of multi-domain E2E network paths at run-time and with guaranteed QoS. BigData Express is currently deployed in several research institutions, including UMD, FNAL, StarLight, KISTI, KSTAR, and Ciena. The SDN for E2E Networked Science at the Exascale (SENSE) [2] is another system providing an intuitive intent-based interface to allow applications to express high-level service requirements. A multi-institution testbed has been deployed at DOE Laboratories and Universities facilities, including Caltech, Fermilab, UMD, NERSC, among others.

Table 1: Application-aware networking: possibilities.

Example Capability	Network Support
Provide Transport Differentiation	
• At app-level granularity	
Create different networks/slices/QoS	Classification; Scheduling
• At sub-app granularity	
Scheduling each packet according to app-level deadline [18]; Distinguish application-level structures (*e.g.*, I frame vs P frame); Co-flow schedule	Classification; Network State; Scheduling
• Cross-app/protocol dependency	
Identify full dependency (*e.g.*, DNS->handshake->…)	Classification; Network state; Scheduling
Provide In-Network Storage/Compute	
• Application state inside the network	
Key-Value Store	Programmable networking
• Application compute inside the network	
Paxos algorithms	Programmable networking

Table 2: Network-aware applications: possibilities

Example Capability	Network Support
Conduct Transport Selection	
• Time adaption	
Bandwidth time window	Network state; Capability information
• Server/Path adaption	
e.g., Which servers to use in multiple replicas	Network state; Capability information
• Rate adaption	
Congestion control (reacting to packet loss/delay/ECN bitdelay, ECN bit [20]/ INT [13]); Adaptive streaming; Lower-than-best-effort (*e.g.*, LEDBAT); Multi-path TCP.	Network state; Capability information
Conduct Network Compute Selection	
• Network function instantiation and invocation	
e.g., Function as a service (FaaS)	Programmable networking

3 POSSIBILITIES OF NAI: NETWORK-AWARE APPLICATIONS

Applications running over networks face challenges due to the lack of network state and information. Applications can benefit from network information exposure to make them more flexible in terms of rate adaptation, transmission time, server/path selection, among others. Therefore, the other side of designing and implementing NAI is network-aware applications, and there are many possibilities as well.

Table 2, for instance, illustrates that applications have possibilities to conduct transport selection capabilities based on network state (*e.g.*, packet loss, INT), performance metrics (*e.g.*, throughput, max reservable Bandwidth), capability information (*e.g.*, delivery/acquisition protocol), and locality (*e.g.*, servers location and paths). Besides, if network can provide programmability support, then applications can also use that support to conduct network compute selection.

Research contributions. There are different proposals introducing the benefits of network awareness for applications. For example, P4P (Provider Portal for Applications) [26] is a framework to enable a better cooperation between network providers and network applications. P4P iTrackers accelerate the content distribution and optimize the utilization of ISP network resources. Another maturing example of NAI protocols is ALTO [12]. ALTO exposes network state and capabilities to support efficient construction of diverse network-aware applications models, such as CDN model, swarm model, dataflow/streaming model, etc. Network information is exposed as abstractions (*e.g.*, network/cost maps) to protect the information privacy and improve the scalability.

Real Deployment Examples. Comcast, a large cable broadband Internet Service Provider (ISP) in the U.S., deployed a P4P-based open framework [9]. Specifically, P4P iTrackers are used to allow P2P networks to optimize traffic within each ISP while improve P2P download performance for P2P clients. Another much larger deployment is Flow Director [19], the first-ever ISP-hyper-giant collaboration system. Flow Director starts with the ALTO protocol but goes further, designing, building, rolling-out, and operating a large scale system that enables automated cooperation between one of the largest eyeball networks and a leading hyper-giant.

4 APPROACH

The preceding discussion exposes huge possibilities of NAI. However, there still exists a major lacking in systematically realizing such a deep integration. Different NAI possibilities are not uniformly deployed due to economy, autonomy, and architecture evolution concerns. Different possibilities have heterogeneous requirements on information exposure, manipulation, and interaction. As such, existing realizations are complex point solutions, and could raise scalability and security concerns. In this section, we first review the key challenges for systematically realizing NAI possibilities. Next, we present the initial design of a systematic framework for NAI possibilities exposure and discovery (PED).

4.1 Key Challenges of NAI

Network information exposure. The first challenge of NAI is that applications are lacking of visibility of available and shared network resources (*e.g.*, bandwidth of shared

resources for a set of flows), resulting in poor performance. Existing network resource exposure mechanisms, including graph-based abstractions [11, 24] and the one-big-switch abstractions [12, 26], either expose all sensitive information, or fail to capture the resource sharing between virtual flow requests. How to expose network information to applications in a unified, abstract representation is still an open challenge.

Network information discovery. The second challenge is the lacking of a generic, flexible mechanism for applications to specify and discover the network information they need for NAI, from the network. Existing solutions (*e.g.*, [12, 19, 26]) provide application interfaces to discover E2E cost information of different packet spaces. However, this information is derived from the network's fixed resource allocation (*e.g.*, fixed route assignment) to the corresponding packet spaces, and applications are not provided the flexibility to discover additional network resources (*e.g.*, on-demand routing) that can satisfy their needs (*e.g.*, waypoint routing).

Security and privacy. Network information exposure and discovery play an important role in realizing NAI, but can also raise security and privacy concerns. For example, application's queries may expose proprietary information (*e.g.*, internal data flow policies) or may reveal too much information about individual clients. From the network side, too much network information may be exposed if aggregation or transformation mechanisms are not considered.

4.2 An SMT-based PED Framework

To address the aforementioned challenges, we propose the initial design of a generic PED framework based on satisfiability modulo theories (SMT). First, the PED framework uses generic mathematical programming constraints as a unified, compact representation of network information. Second, PED utilizes the equivalence between relational algebra and first order logic to provide a SQL-style language for application to express their intents on discovering resources in the network. Third, PED develops a compiler to translate an application's resource discovery intent into a constraint programming problem with a set of logical constraints, whose feasible solutions correspond to qualified configurations for application. In addition, to improve the efficiency of finding qualified configurations, PED also develops a search space decomposition (SSD) algorithm that decomposes the compiled constraint programming problem into a series of subproblems with smaller, disjoint search space. Fig. 2 presents the architecture and workflow of the PED framework.

Mathematical programming constraints as a unified, compact resource representation. In PED, when the network needs to expose the information for a set of flows to applications, it uses mathematical programming constraints

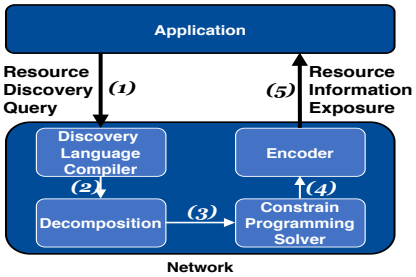


Figure 2: Architecture/Workflow of SMT-based PED framework.

to capture the resource availability and sharing information of these flows, providing a unified resource representation.

Specifically, suppose PED receives the resource discovery request of a set of flow F. For each flow $f_j \in F$, we use x_j to denote an available resource (*e.g.*, bandwidth) the application can reserve for this flow. Upon receiving this request, PED first checks the routes – computed by the underlying routing protocol – for each flow f_j. Then all the links are enumerated. For each link l_u, it generates a linear inequality:

$\sum x_j \leq l_u.resource, \forall f_j$ that uses link l_u in its route.

To illustrate this formulation, consider the network topology in Figure 3, where an application wants to reserve bandwidth for two flows $f_1 : (S_1, D_1)$ and $f_2 : (S_2, D_2)$. The routes for the two flows share common links, *i.e.*, l_3 and l_4, hence it infeasible for both circuits to each reserve a 100 Mbps bandwidth. Therefore, the PED framework will generate the following set of linear inequalities:

$$\begin{aligned} x_1 &\leq 100, \forall l_u \in \{l_1, l_2, l_5, l_6\}, \\ x_2 &\leq 100, \forall l_u \in \{l_7, l_8, l_{11}, l_{12}\}, \\ x_1 + x_2 &\leq 100, \forall l_u \in \{l_3, l_4\}. \end{aligned} \tag{1}$$

Where x_1 and x_2 represent the available bandwidth that can be reserved for $(S1, D1)$, and $(S2, D2)$, respectively. Each linear inequality represents a constraint on the reservable bandwidths over different shared resources by the two flows.

Resource discovery language. PED introduces a declarative language that allows applications to express flexible resource discovery intents. Specifically, the language uses a resource-filtering design, which allows applications to define predicates on packet spaces (*i.e.*, different sets of flows), and predicates on resources (*i.e.*, particular resource attributes that applications are interested in discovering). Leveraging the equivalence between relational algebra and first-order logic, the language uses SQL-style semantics, which are familiar to both application and network engineers. Figure 4 gives an example to discover the bandwidth information of two flows (based on the configuration in Fig. 3), where the bandwidth of both flows must be at least 100 Mbps.

From resource discovery query to network information exposure. Given a resource query (Step 1 of Fig 2), PED first compiles the requirement filter predicates into a

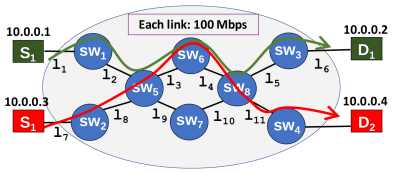


Figure 3: An example where an application tries to discover information of tow flows from the network.

```
flow_1: {src_ip = 10.0.0.1 and dst_ip = 10.0.0.2};
flow_2: {src_ip = 10.0.0.3 and dst_ip = 10.0.0.4};
flow_set: {flow_1, flow_2};
req_1: flow_2.bandwidth >= 100 Mbps;
req_2: flow_2.bandwidth >= 100 Mbps;
select bandwidth from flow_set
        where req_1 and req_2;
```

Figure 4: An example resource discovery query.

set of logical constraints (Step 2), and then leverages state-of-the-art solvers (*e.g.*, the Z3 SMT solver) to search for qualified configurations that satisfy the application requirements. Specifically, given a resource query, a *qualified configuration* is defined as the network paths (i) that can be used to route the traffic of the packet space specified in the query, and (ii) along which available resources to the specified packet space satisfy the resource predicates in the query.

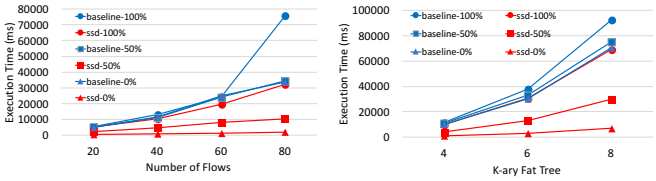
To preserve the privacy of network, in PED the network has the flexibility of deciding how many qualified configurations to search for and how many of them can be returned to the application. This can be achieved by tuning corresponding options in problem solvers.

Topologies of networks, especially data center networks, have a large number of possible paths for each source-destination pair, resulting in a large search space and a higher latency for finding qualified configurations. As such, we develop the SSD algorithm to decompose the compiled constraint programming problem on the whole network into multiple sub problems on smaller, disjoint partitions of the network (Step 3). These sub problems can be solved efficiently and in parallel. In this way, the efficiency of finding qualified configurations for application is substantially improved.

After qualified configurations are found (Step 4), PED encodes the resource information of the configurations (*e.g.*, bandwidth sharing) in a set of mathematical programming constraints and sends to the application (Step 5). For example, from the bandwidth availability query in Figure 4, PED will provide the linear inequality $x_1 + x_2 \leq 100$ indicating that both flows share a common resource and thus, the sum of their bandwidths can not exceed 100 Mbps.

After receiving the network information, the application can then optimize resource allocation for its flows using the retrieved information, together with its private constraints.

Preliminary evaluation. We evaluate our design proposal in different scales of fat-tree topologies [23] using different



(a) (b)

Figure 5: The resource discovery latency of PED w/ and w/o SSD: (a) 4-ary fat tree with varying numbers of flows; (b) 40 flows with varying fat tree sizes.

workloads. Specifically, we study the performance of two versions of our design: (i) SSD: the full version of PED where the SSD algorithm is enabled; (ii) Baseline: a simplified version of PED where the SSD algorithm is disabled. We measure the resource discovery latency of both versions as the elapsed time from the time when the compiler finishes the compilation to the time when a feasible configuration of the original search problem is found.

We generate different application workloads by randomly selecting different amounts of end host pairs to compose different flow sets, and divide them 3 different cases, each of which has a different ratio of resource sharing requirements (*e.g.*, QoS metrics) in the original logical constraints. For example, baseline-100% indicates that *all* original logical constraints are QoS metric requirements for *all* flows.

Figure 5a plots the resource discovery latency of two PED versions in a 4-ary fat tree topology as the number of flows changes. Results show that the PED with SSD enabled always has a lower discovery latency than the one with SSD disabled, regardless of the ratio of resource sharing requirements. When this ratio is fixed, the improvement of discovery latency increases as the number of flows does. Specifically, when there is no resource sharing requirement, SSD improves the discovery latency by up to 15 times (*i.e.*, 30000 ms vs. 2000 ms with 80 flows), demonstrating the efficiency of the first phase of search space decomposing in SSD. When all constraints are resource sharing ones, SSD improves the latency by 2-4 times, demonstrating the efficiency of the second phase of search space decomposing. Similar results can also be observed in Fig. 5b.

5 FUTURE RESEARCH DIRECTIONS

Multi-domain information exposure. Many novel applications require the orchestration of multiple resources across multiple domains (technological or administrative) where dynamics and topologies are completely different. Exposing network information for a multi-domain setting introduces a basic challenge because each domain can have its own representation of the same network infrastructure. To fully benefit from the network awareness in applications, it is necessary to design multi-domain composition mechanisms, so

that network information in multiple domains are adapted together to a single and consistent "virtual" abstraction.

Control exposure for NAI. A programmable network can provide opportunities to both applications to optimize E2E routing control and network service providers to increase business offering. However, traditional inter-domain routing protocols (e.g., using the traditional BGP model) provide very limited mechanisms for network operators and applications to achieve flexible, E2E route control. Thus, more research efforts in this direction are required to expose more control (ultimately programmability) beyond just information.

Computation complexity optimization. As already mentioned, resource discovery techniques are characterized by increased optimal resource allocation but at the cost of communication and computation overhead of resource discovery. Therefore, solutions to reduce the delay as well as number of messages for resource discovery are necessary. A possible optimization consists in proactively discover the resource information. Another alternative to explore is to use those pre-computed abstractions to quickly project to get the resource abstraction for application's requests.

Security/privacy preserving. PED may rise to privacy and security issues. Therefore, it is necessary to ensure that queries to the network can provide enough information without compromising the privacy of clients/applications. To deal with the network information exposure issues, mechanisms to ensure that information is transformed and aggregated need to be also developed.

6 CONCLUSIONS

The collaboration between networks and applications brings benefits to both parties, yet realizing it is non-trivial. In this paper, we review huge possibilities in designing and implementing NAI by application-aware networking and network-aware applications. We design PED, an NAI possibilities discovery and exposure framework to address the key challenges of systematically realizing NAI. Preliminary experiments show the potentials of this framework. Besides, we also discuss future research directions toward deep NAI.

REFERENCES

[1] Aleksandar Kuzmanovic et al. 2006. TCP-LP: low-priority service via end-point congestion control. *IEEE/ACM ToN* (2006).

[2] Inder Monga et al. 2018. SDN for End-to-end Networked Science at the Exascale (SENSE). In *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*. IEEE, 33–44.

[3] Jeongkeun Lee et al. 2014. Application-driven bandwidth guarantees in datacenters. In *Proceedings of the 2014 ACM conference on SIGCOMM*.

[4] Marian Babik et al. 2020. HEPiX Network Functions Virtualisation Working Group Report. (April 2020). https://tinyurl.com/ydaptoed

[5] Mauro Campanella et al. 2006. Bandwidth on demand services for european research and education networks. In *2006 IEEE First International Workshop on Bandwidth on Demand*. IEEE, 65–72.

[6] Qiming Lu et al. 2018. BigData Express: Toward Schedulable, Predictable, and High-Performance Data Transfer. In *2018 IEEE/ACM Innovating the Network for Data-Intensive Science (INDIS)*. IEEE, 75–84.

[7] R. Yang et al. 2017. Magellan: Toward Automatic Mapping from High-Level SDN Programs to Low-Level , Optimized Datapath Pipelines.

[8] Thomas Karagiannis et al. 2005. Should internet service providers fear peer-assisted content distribution?. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. 6–6.

[9] Y. Richard Yang et al. 2009. Comcast's ISP Experiences in a Proactive Network Provider Participation for P2P (P4P) Technical Trial. RFC 5632. (Sept. 2009). https://doi.org/10.17487/RFC5632

[10] ETSI. 2020. Zero touch network & Service Management. https://tinyurl.com/yatpnnku. (2020).

[11] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, R. Katz, S. Shenker, and I. Stoica. 2011. Mesos: A platform for fine-grained resource sharing in the data center. In *NSDI*.

[12] S. Kiesel, W. Roome, R. Woundy, S. Previdi, S. Shalunov, R. Alimi, R. Penno, and R. Yang. 2014. Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285. (2014). https://doi.org/10.17487/RFC7285

[13] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. 2015. In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM*.

[14] Zhenbin Li, Shuping Peng, C. Xie, and Li Cong. 2019. *Application-aware IPv6 Networking*. Internet-Draft. IETF Secretariat. http://www.ietf.org/internet-drafts/draft-li-6man-app-aware-ipv6-network-00.txt

[15] Harsha et al. Madhyastha. 2006. iPlane: An information plane for distributed services. In *Proceedings of OSDI'06*.

[16] Michael Ott. 2005. Intelligent network based application recognition. (Nov. 1 2005). US Patent 6,961,770.

[17] Pacific Wave. 2020. R&E Networks in CA, OR, and WA Announce Collaborative Support for COVID-19 Wester States Pact. (April 2020). https://tinyurl.com/ydgfa7kf Accessed 2020-05-03.

[18] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: a centralized" zero-queue" datacenter network. In *Proceedings of the 2014 ACM conference on SIGCOMM*.

[19] Enric Pujol, Ingmar Poese, Johannes Zerwas, Georgios Smaragdakis, and Anja Feldmann. 2019. Steering Hyper-Giants' Traffic at Scale. In *Proceedings of CoNEXT '19*. ACM, New York, NY, USA.

[20] K. Ramakrishnan, S. Floyd, and D. Black. 2001. *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC 3168. RFC Editor. http://www.rfc-editor.org/rfc/rfc3168.txt http://www.rfc-editor.org/rfc/rfc3168.txt.

[21] Philipp S. Schmidt, T. Enghardt, R. Khalili, and A. Feldmann. 2013. Socket Intents: Leveraging Application Awareness for Multi-Access Connectivity. In *Proceedings of CoNEXT '13*. ACM, New York, NY, USA.

[22] Robert Soulé, Shrutarshi Basu, Parisa J. Marandi, F. Pedone, Robert Kleinberg, Emin Gun Sirer, and N. Foster. 2014. Merlin: A language for provisioning network resources. In *Proceedings of CoNEXT '14*.

[23] Amin Vahdat, M. Al-Fares, and A. Loukissas. 2013. Scalable commodity data center network architecture. (July 9 2013). US Patent 8,483,096.

[24] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems*.

[25] Qiao Xiang, C. Guok, F. Le, J. MacAuley, H. Newman, and R. Yang. 2018. SFP: Toward Interdomain Routing for SDN Networks. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos*. ACM.

[26] Haiyong Xie, Y Richard Yang, Arvind Krishnamurthy, Yanbin Grace Liu, and Abraham Silberschatz. 2008. P4P: Provider portal for applications. *ACM SIGCOMM Computer Communication Review* 38, 4 (2008).

[27] Xuan Zheng, Malathi Veeraraghavan, Nageswara SV Rao, Qishi Wu, and M. Zhu. 2005. CHEETAH: Circuit-switched high-speed end-to-end transport architecture testbed. *IEEE Communications Magazine* (2005).