Sextant: Enabling Automated Network-aware Application Optimization in Carrier Networks

Jingxuan Zhang*, Luis Contreras†, Kai Gao‡, Francisco Cano†, Patricia Cano†, Anais Escribano§
Y. Richard Yang¶
* Tongji University; † Telefonica University; ‡ Sichuan University; § Alten; ¶ Yale University

***Abstract*—A growing trend is that network service providers and applications are integrated more closely through open APIs for better end-to-end performance. In particular, the provided-aided network information can help applications orchestrate their traffic for a better quality of experience, and is gaining a lot of attention from both the academia and the industry. However, enabling automated network-aware application optimization in today's Internet is difficult because of the multi-vendor, multi-domain nature of carrier networks. In this paper, we introduce Sextant, a novel network information exposure system, taking a solid step towards enabling network-aware application optimization in carrier networks. Our work is driven by a real requirement from an international ISP, where timely, accurate distance information is critical to fully utilize its CDN caches deployed at different Point-of-Presence (PoP). We design a novel, flexible requirement model, allowing applications to specify their interests. The queries are efficiently carried out using our aggregation and incremental update algorithms. Evaluations demonstrate that our prototype system operates on top of real router software images, and can efficiently handle dynamics from multiple ASes.**

***Index Terms*—network-awareness, ISP-CDN collaboration**

I. Introduction

A foundation of intelligent network management is network openness, where the network provider and its customers exchange information through automated, open APIs. Thus, the network and overlay applications can potentially be better integrated to provide better end-to-end performance. Network openness can substantially improve the QoE of web-based applications, as reported by several researches [1–9].

In particular, an important feature of network openness is that network service providers facilitate guidance (e.g., abstract network view [10]) to the overlay applications for mutual benefits over the measurement-based, fully end-to-end solutions.

As the network service provider has full visibility and control over the network, it can easily identify and even foresee potential network dynamics including failures and policy updates. Thus, those changes can be timely reflected in the abstract network view and help the applications take actions accordingly.

Several systems dynamics may effect the quality of service (QoS) of an application, like network failure events or some capacity upgrades in the application side (e.g., through the deployment of new servers). In consequence, it is crucial for the application to quickly obtain the performance metrics between end hosts and these newly deployed servers to better steer the traffic. However, this cannot be achieved using a measurement-based solution. In the meantime, the network service provider can quickly adopt the changes in the abstract network view. It may even provide the information without actually deploying the new servers. Thus, the application can estimate the expected performance before the real deployment, and decide whether to invest the new servers or not.

Pioneering work such as ALTO [10] has identified this problem and provided automated ways to distribute an abstracted network view to applications. However, realizing it in carrier networks [1] as we target in this paper, is not trivial because of the following practical challenges:

- How to collect and disseminate the network information with the practical constraints posed by carrier networks.
- How to efficiently obtain fine-grained global abstract network view from multiple networks.
- How to efficiently reconstruct and disseminate the abstract network view upon dynamics

Existing researches [1, 11–13] is not practical to address those challenges in our settings. Therefore, in this paper, we introduce Sextant, a novel framework to enable Automated Network-aware Application (ANA) optimization leveraging on the structure of the real carrier network (§ II). Sextant uses BGP [14] and BGP-LS [15] to aggregate network information, and provides the abstract network to applications through the Application-Layer Traffic Optimization (ALTO) protocol [10]. The information collected from every AS is aggregated using a novel algorithm that aggregates prefixes based on the first hop and provides hop-by-hop cost values. We also introduce how the networks can be properly configured to make sure the aggregator obtains the accurate and complete global network routing information. To enable fast responses to dynamics, we develop a novel algorithm to efficiently reconstruct abstract network views from incremental updates. The prototype system is now running on a testbed using real router software images as a previous validation step before being deployed on an actual operational network (§ III).

II. Sextant

In this section, we first give an overview of Sextant and introduces how it interacts with applications. Then, we present

[1] By carrier networks, we are referring to the networks owned and operated by a single Internet Service Provider (ISP) which spans over a wide geographical area, and providing multiple services (e.g. fixed and mobile).

978-3-903176-32-4

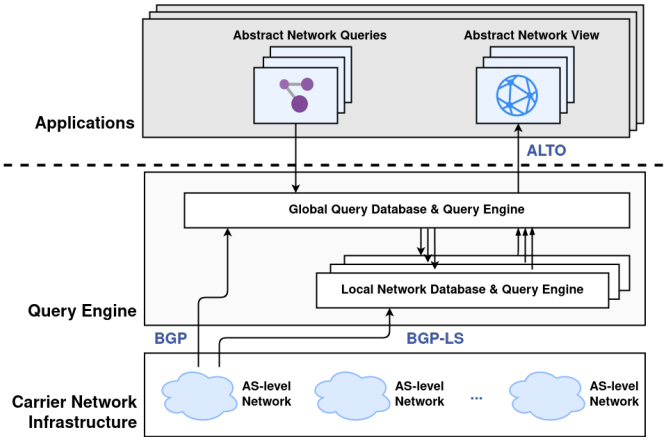


Fig. 1: Overview of Sextant.

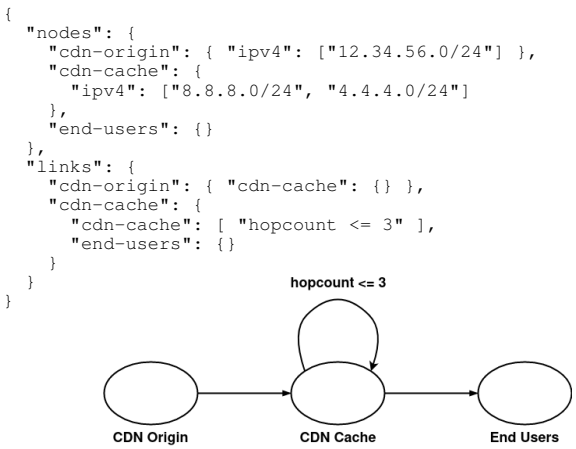


Fig. 2: An Example Abstract Network Query and its Graphical Representation. Node Filters are Omitted.

the basic idea of each design component and discuss how it can enable automated network-aware application optimization in carrier networks. Given the limitation of pages, we put more details of Sextant in an extended technical report [16].

A. Overview

Fig. 1 gives a high-level overview of Sextant. In a nutshell, Sextant 1) continuously collects AS-level inter-domain connectivity and intra-domain link state using BGP [14] and BGP-LS [15], and stores them in local network databases, and 2) accepts abstract network queries from multiple applications, and returns an abstract network view for each query using the ALTO protocol [10].

The hierarchical query engine monitors changes from both the application (e.g., new queries or modifications on existing queries) and the network (e.g., link state changes caused by network dynamics). When necessary, as we discuss in § II-D, it automatically updates the abstract network view to provide the applications with the most up-to-date fine-grained network information.

a) Abstract Network Query: Sextant allows applications to specify the information they are interested in, using abstract network queries. Each abstract network query is modeled as an abstract directed graph. Each node represents a specific type of entities, each link represents that the application is interested in the distance between the source entities and the destination entities, and a self-loop represents the interest to know the distance between the corresponding entities.

Nodes and links in an abstract network query can be annotated with filters to better express the interests of an application. Entities or entity pairs that do not satisfy the filter conditions will not be included in the abstract network view. This approach has two benefits: first, it helps reduce the data transmission between the network provider and the applications; second, it allows the Sextant query engine conduct optimizations to avoid unnecessary computation, speeding up the query execution.

Right now we only support filters on IPv4 addresses for the nodes, and hop-count constraints between entities. However, the design is very extensive and can be adopted to support constraints on other properties as well. We also enforce that there can be at most one wildcard node, i.e., a node without any IPv4 filter. Also, we forbid self-loops on the wildcard node. It is the case where an application is querying the end-to-end distance between all users of the ISP network. This type of query requires a lot of storage and computation power, posing a heavy burden both on the network and the application's scheduler. Given the circumstance, it is unlikely that the application can take real advantage of the information. Thus, we forbid this undesired behavior.

b) Abstract Network View: We reuse the abstractions provided by the ALTO protocol [10]. For each abstract network query, Sextant automatically generates two ALTO resources: a *network map* which aggregates IPv4 prefixes to different PIDs, and a *cost map* which contains the distances between PID pairs, measured by the number of hops.

The PIDs are constructed based on the nodes in the corresponding abstract network query. If two IP prefixes belong to different types of entities, they will be put into two different PIDs even if they may be located at the same PoP. The PIDNames in the network map are derived from the type of entities. Specifically, it appends a suffix to the type of entity, where the suffix is a hash of where the prefixes are attached. Distances are only computed for PID pairs whose corresponding link type is specified in the abstract network query. Also, distances that violate the link filters are omitted in the cost map.

c) Network Information Collection: The fine-grained network information is collected using BGP and BGP-LS. Specifically, to obtain the AS-level connectivity for each prefix, we configure a border router running eBGP as a route reflector in each AS.

B. Sextant in Practice

A common abstract network query, derived from the real requirements of a CDN operator, is as shown in Fig. 2, alongside its graphical representation. It specifies three types

```
{
  "meta": { ... },
  "network-map": {
    "cdn-origin-793bb1": { "ipv4": ["12.34.56.0/24"] },
    "cdn-cache-6314f2": { "ipv4": ["8.8.8.0/24"] },
    "cdn-cache-0d45ea": { "ipv4": ["4.4.4.0/24"] },
    "end-users-30e0ac": { "ipv4": ["5.5.5.0/24"] },
    "end-users-b9cda9": { "ipv4": ["3.3.3.0/24"] }
    ...
  }
}

{
  "meta": { ... },
  "cost-map": {
    "cdn-origin-793bb1": {
      "cdn-cache-6314f2": 2,
      "cdn-cache-0d45ea": 2
    },
    ...
  }
}
```

Fig. 3: The Generated ALTO Maps for the Query in Fig. 2.

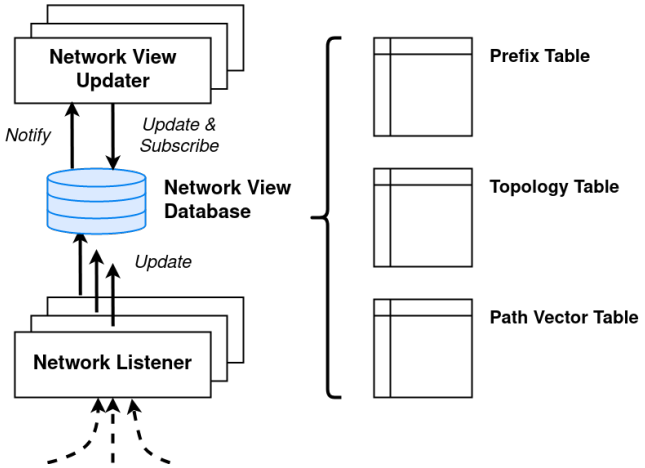


Fig. 4: A Closer Look at Sextant's Query Engine.

of entities, whose meanings are quite straight-forward: *cdn-origin*, *cdn-cache*, and *end-users*. It contains 3 links including a self-loop. The links from cdn-origin to cdn-cache and from cdn-cache to end-users are unidirectional since the application is mostly interested in the downloading performance, which is the major QoS metric for video streaming. Since the CDN allows caches to fetch data from neighbors, it is also interested in the distances between different caches, resulting a self-loop on the cdn-cache node. However, if two caches are too distant, for example, more than 3 hops away, the cache-to-cache data transfer is not economic. Thus, it puts a constraint on the self-loop link, specifying interests only in caches no more than 3 hops away.

The automated-generated ALTO maps for the example query are illustrated in Fig. 3. Given the limited space, we only include two prefixes for the end users in the network map, only include the cost values between cdn-origin nodes and cdn-cache nodes in the cost map, and omit the *meta* field in both maps. The hash values are also truncated for better readability. As we can see, the network map re-groups the prefixes into different PIDs, and the cost map only returns values for PID pairs whose link type appears in the query.

The application (e.g., the CDN request router) can periodically query the generated maps, or subscribe to the resources using ALTO's incremental update mechanism [17]. The latter is recommended as it allows an application to automatically receive the most up-to-date network information.

C. System Design

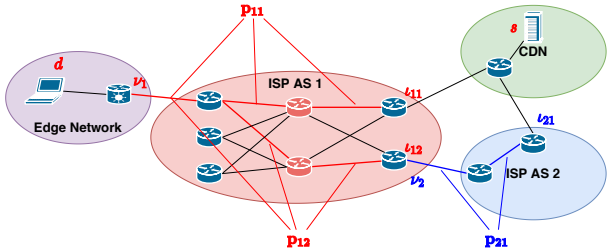
Now we introduce the design details of the core query system, as illustrated in Fig. 4. It includes three key components: 1) *network listener* that collects the low-level network information (e.g., LSDBs, RIBs) using southbound interfaces (e.g., BGP, OSPF, ISIS), 2) *network view database* that maintains database tables representing Sextant network abstraction, and 3) *network view updater* that transfers the low-level network information into Sextant network abstraction with user demands in real-time.

The designs of these components make it possible to fulfill the requirements of enabling automated network-aware application optimization in a carrier network. First, the southbound interfaces use standard protocols, which ensures the interoperability between devices from multiple vendors. Second, the network view database forms a hierarchical structure, where the global view database aggregates local forwarding views. Third, the network view updater performs incremental updates in the network view database.

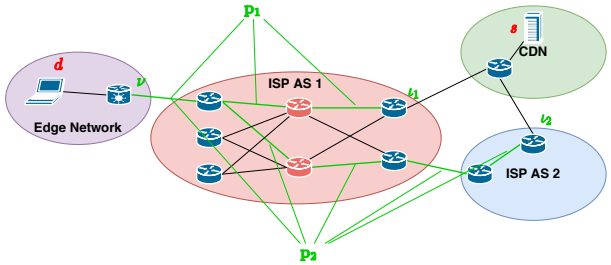
a) Network Listener: This component is to monitor the control plane of the routing elements in a network domain and to collect the low-level network information via the southbound interfaces. In particular, a network listener collects subnets and the routing information in intra-domain areas from the OSPF and ISIS routers, and external subnets and the inter-domain routing information from the BGP border routers. To reduce the southbound connections and simplify the collection, Sextant uses BGP-LS as a unified southbound interface to collect the link-state database of OSPF or ISIS areas.

b) Network View Database: The main function of this component is to maintain the network information collected by the network listener and the Sextant network view generated by the network information. Each network view database contains three database tables: 1) prefix table that records the internal and external IP prefixes that may send or receive traffic going through the current network domain, 2) topology table that records the network graph annotated with properties of each link and routing element, and 3) path vector table that caches the path vector of each potential network flow in the scope of the current network domain. Each network domain can maintain its own local network view database. The ISP and the higher-level network administrator can build network view databases hierarchically. All the network view databases organize the network view in the same format.

c) Network View Updater: This is a component to convert the low-level network information into the network view database, and update the tables in the existing network view database.



(a) Local network forwarding views of two ASes.



(b) Aggregated network forwarding view of the ISP.

Fig. 5: Hierarchical Forwarding View in Sextant.

Each network domain can have its own local network view updater to generate and update its local network view database. The higher level aggregated network (e.g., ISP, collaborative network) can set up the aggregated network view updater to build the aggregated network view database based on the network view databases of its member networks hierarchically. Thus, a major responsibility of the network view updater is to aggregate local network forwarding views into a global view, as illustrated in Fig. 5. A local forwarding view represents the network information of potential network flows in its own local scope. In particular, it includes a set of forwarding entries. Each forwarding entry records the path vector for a potential network flow going through this network domain. A network flow is identified by the IP prefix of the traffic sender (s), the IP prefix of the traffic receiver (d), and the ingress point where the traffic enters this network domain (ι).

When the low-level network information changes (e.g., link failures, demand changes), the corresponding local network view updater will reconstruct the network view and update the local network view database dynamically. The updates of local network view databases will trigger the aggregation process and then be aggregated into the global network view database dynamically.

D. Efficient Network View Computation

The main difficulty is how to efficiently compute the network view. To achieve this practically, Sextant develops two techniques:

a) Compact View Database Aggregation: For a carrier network involving multiple network domains, the most time-consuming process is the aggregation of databases, in particular, the aggregation of the path vector tables from different local network view databases. The traditional approach is to store path vector tables in standard relational databases, and simply use the union operation to aggregate them. It is very inefficient as the application has to lookup the union of tables recursively for multiple times to get the complete forwarding entries (i.e., path vectors). To reduce the lookup time of the path vector table, Sextant maintains the *compact path vector table* in both local and global query manager.

b) Incremental View Database Recomputation: The network view database has to be updated once the network information (e.g., DHCP, LSDB, BGP RIB, etc.) changes. However, recomputing the whole network view database from scratch whenever the network changes is very inefficient. From our practical experience, we find that, in each given time window, only a limited number of low-level network information entries may change, which may affect very few table entries in the network view database to be updated. To avoid recomputing the whole network view database everytime, we apply *incremental updates* to the network view database tables.

III. Evaluations

In this section, we evaluate Sextant in both real testbed and microbenchmark scenarios.

A. Evaluation on Real Testbed

In this evaluation, we set up a real testbed as a shortcut version of the real ISP topology to answer the first two questions. We deployed Sextant in our testbed to demonstrate the usability of Sextant and performance of the application using Sextant. The experiment includes two types of network dynamics: (1) network failure, and (2) demand change.

a) Testbed setup: The testbed setup used in this evaluation is as reflected in Fig. 6. It has been built in leveraging on virtual images of commercial routers. In our case all the routers are based on Cisco IOS XRv 9000 Router, version 6.6.2. The setup represents a partial view of the real topology pattern of an operational network, including backbone (hierarchical level 2, HL2, formed by router R9 and R10), distribution (HL3, formed by R8, R4 and R2) and aggregation (HL4, formed by R1, R5 and R6) layers. On this topology a set of residential users are emulated as connected to HL4 routers with different /24 prefix ranges (e.g., 5.5.5.0 /24 attached to R5). Similarly, a couple of CDN end-points are emulated as connected to R4 and R8 to illustrate different deployment options. Specific virtual machines have been instantiated for emulating both hosts and CDN end-points, using iperf tool for generating traffic among them. Finally, two specific routers (R3 and R7) play the role of route reflectors, one aggregating BGP sessions (for collecting network topology) and the other maintaining BGP-LS sessions (for the link states).

The route reflectors maintain those BGP sessions with all the routers in the setup, and also with the OpenDayLight (ODL) controller in its version Oxygen-SR4, including an operational ALTO implementation, in which on top of that a specific plugin has been developed for enabling the interaction between ALTO and BGP modules, not existing up to now.

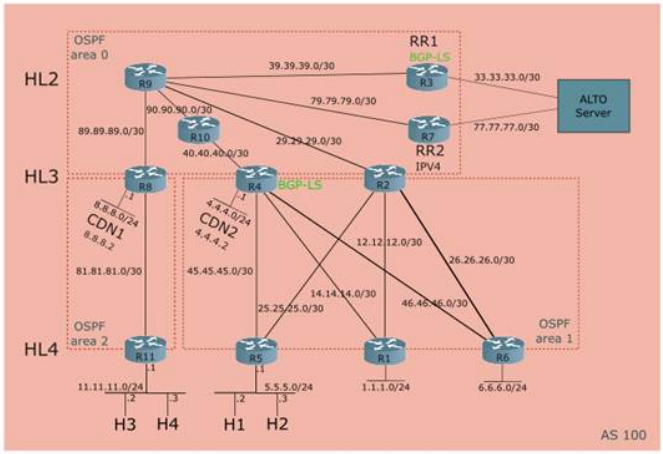


Fig. 6: Topology of testbed

All this setup physically runs on top of three Dell Power Edge R740XD compute nodes with 128 GB of RAM, 80 CPUs, 1,8 TB HDD and 240 GB SSD, and CentOS 7.6.1810, involving one additional node as controller based on a Dell PowerEdge R730 server hosting a KVM Linux VM with 8 GB RAM, 8 CPUs, 150 GB HDD and CentOS 7.6.1810. The OpenStack version used is Queens.

b) Response time for network failure: We first evaluate the Sextant response for network failure.

We set up a simple CDN mapping using the hop-count based ALTO cost map to allocate the CDN replica server for the client. The allocated server IP is mapped by the domain name `cdn.optimal`. During the evaluation, we continuously measure the RTT between the client (H1) to the allocated CDN replica server `cdn.optimal`.

To illustrate the network failure, we manually shut down the port of R4 that is connected to R5.

Fig. 7 shows the RTT measure between the client H1 and the CDN servers (CDN1, CDN2, and `cdn.optimal`). We shutdown the link R4-R5 at 25s and recover the link at 105s. As we set the link-state advertisement interval as 1 second, the link-state updates can be received by Sextant very quickly. The overall QoS is better than any fixed CDN mapping. Note that although we use RTT to estimate the QoS of the CDN, the optimization of the CDN mapping is based on the hop count information, which may lead to suboptimal allocation sometimes.

c) Response time for demand change: Then we evaluate the Sextant response for the demand change.

To illustrate the demand change, we add/remove a CDN node manually. We use the same network and CDN mapping setup in this evaluation. We continuously measure the RTT between the CDN users (H1, H3) and the allocated CDN replica server `cdn.optimal`.

Fig. 8 shows the measured RTTs between users and the CDN. In the beginning, we do not include CDN2 in the network map. Both H1 and H3 select CDN1. At 75s, we add CDN2. For H1, it still selects CDN1. But for H3, it change the replica server to CDN2. At 175s, we remove CDN1. Both H1 and H3 select CDN2.

B. Evaluation on Mirco-benchmark

To answer the last two questions, we set up a micro-benchmark and run simulated networks and queries on it so that we can evaluate the efficiency of our solution in larger scale networks. In particular, we evaluate (i) the lookup efficiency of the compact network view database, and (ii) the update efficiency of the incremental network view database computation.

a) Benchmark Setup: We use the sample networks in topologyzoo [18] to simulate the AS-level inter-domain network, and use the mrinfo [19] dataset to simulate the intra-domain networks. We use a custom network simulator to set up the whole network in a workstation (Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz, 64GB RAM). We run BGP on each border router of each AS, and link-state protocol inside each AS. We attach the IP prefixes reported by CIDR Report [20] to the routers of ASes.

b) Efficiency of compact view database aggregation: In this experiment, we compare the lookup efficiency between using (i) the compact network view database and (ii) the simple union network view database.

To evaluate the lookup efficiency, we simulate an ALTO client to send abstract network queries to the query engine, and estimate the number of database queries that the query engine does to respond to the ALTO client. We simulate 30 abstract network queries.

Fig. 9 shows the nummber of database queries that the query engine does, and the number of table entries queried from the database. Using the simple union view aggregation, the query engine has to query the database for multiple times to respond a single application query request. In the experimental network, the query engine has to execute 6-7 database queries. However, using the compact view aggregation, the query engine can always finish the asbtract network view computation in one database query. It can reduce both database queries and queried table entries by 5-6 times.

c) Efficiency of incremental view database computation: We also evaluate the efficiency of applying incremental network view database computation, to compare with the fully recomputation.

We consider the number of recomputed entries in the path vector table as the performance metric, and trigger three kinds of network changes: (i) IP prefix changes, (ii) intra-domain link state changes, and (iii) inter-domain routing changes.

To evaluate the performance of incremental network view database computation, we trigger 10 times of network changes in each AS. Fig. 10 demonstrates the fraction of table entries recomputed by the local query engine of each AS. From the result, for most of the network changes, the incremental computation even don't have to recompute any table entries. In average, the incremental computation only needs to recompute the table entries less than 5%. For very few (less than 0.5%) extrame cases, the incremental computation recomputes over

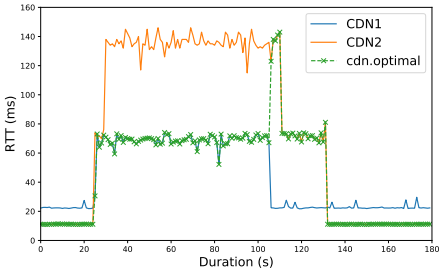


Fig. 7: Response for network failure

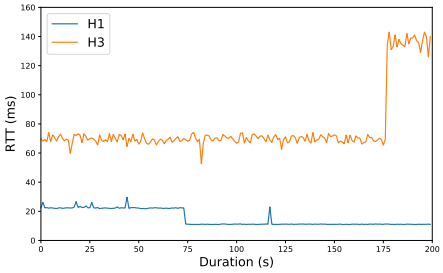


Fig. 8: Response for demand change

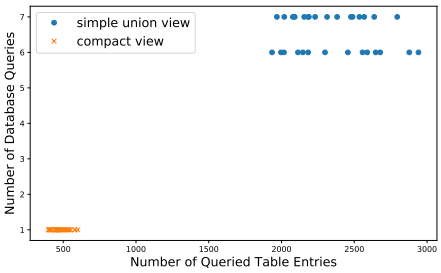


Fig. 9: Lookup efficiency of aggregation.

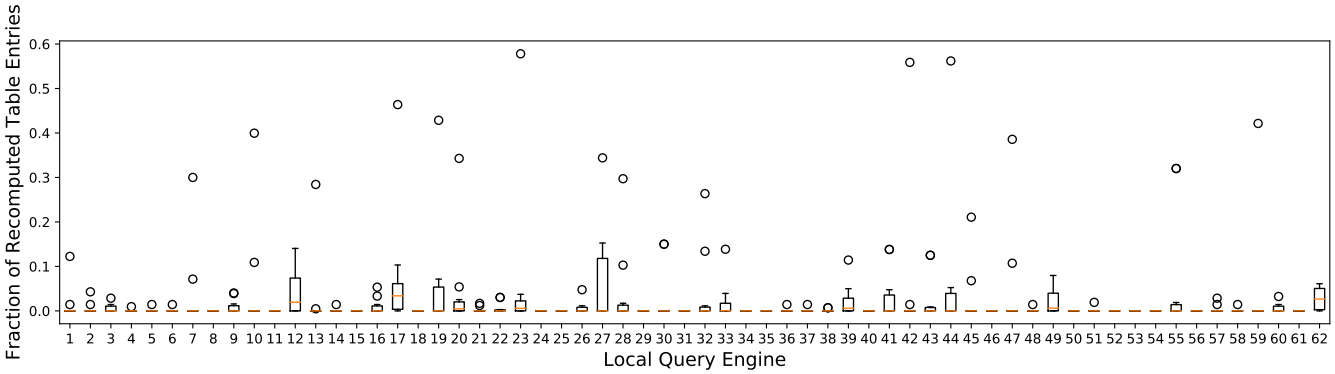


Fig. 10: Update effiency of incremental view database computation.

half of table entries, but the fraction is still less than 60%, which is much better than the fully recomputation.

IV. Related Work

Network Information Exposure. An abstract network view is a fundamental feature in Software-Defined Networking [21] (SDN). Thus, SDN controllers [22–25] provide the network topology to its applications. However, for confidentiality, the information is usually not directly exposed to end users. Some researches [7, 26–28] are built on top of SDN and can provide more abstracted network views. However, they depend on devices that are not yet widely deployed and cannot be applied directly in carrier networks.

ALTO (Application Layer Traffic Optimization) [10] is a protocol that defines standard interfaces for network providers to expose network information to overlay applications. We use ALTO as our northbound interface and propose a concrete implementation on top of carrier networks.

Application-Network Collaboration and Integration. Since traffic optimization is beneficial for both the network and the overlay applications, several studies have proposed to deepen the application-network integration and collaboratively optimize the traffic, such as P4P [1], Joint TE [11] and Unison [12]. In this paper, given the real requirements from the customers, we focus on a loosely coupled application-network integration approach which only gives guidance and allows applications to steer their traffic based on their own objectives.

FlowDirector [8] is the most relevant study to this paper. It also collects network information and exposes it to large content providers, which are referred to as hyper-giants. We are different from FlowDirector in several ways. First, the ISP in FlowDirector forbids caches inside the network but cache placement is a major motivation in our paper. Second, we use BGP-LS protocol for topology collection in carrier-grade application-network integration, which simplifies the configuration of the system. Last, we give details of key algorithms which are not fully described in FlowDirector.

V. Conclusions

In this paper, we introduce Sextant, a novel automated network information exposure system that is workable in realistic carrier-grade ISP networks. Sextant allows overlay applications to express their need with a novel, flexible requirement model, and automatically generates ALTO information resources for each application based on routing information collected by technologies that are already widely deployed, i.e., BGP and BGP-LS. Evaluations demonstrate Sextant can work on real router software images, and can efficiently handle dynamics in multiple ASes. We include more details and further discussions in an extended technical report [16] due to the limited space.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments. We also would like to thank Telefonica and IETF ALTO working group for collaborations and substantial contributions.

This work was supported in part by the National Natural Science Foundation of China under Grant 61902266, Grant 61702373, Grant 61672385, and Grant 61701347.

References

[1] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: Provider Portal for Applications," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '08. ACM, 2008, pp. 351–362.

[2] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *INFOCOM, 2013 Proceedings IEEE*. IEEE, 2013, pp. 854–862.

[3] K. LaCurts, S. Deng, A. Goyal, and H. Balakrishnan, "Choreo: Network-aware Task Placement for Cloud Applications," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. ACM, 2013, pp. 191–204.

[4] A. Ganjam, J. Jiang, X. Liu, V. Sekar, F. Siddiqi, I. Stoica, J. Zhan, and H. Zhang, "C3: Internet-scale Control Plane for Video Quality Optimization," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'15. USENIX Association, 2015, pp. 131–144.

[5] N. Zhang, Y. Lee, M. Radhakrishnan, and R. K. Balan, "GameOn: P2p Gaming On Public Transport," in *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM Press, 2015, pp. 105–119.

[6] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "CLARINET: WAN-Aware Optimization for Analytics Queries," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. USENIX Association, 2016, pp. 435–450.

[7] K. Gao, Q. Xiang, X. Wang, Y. R. Yang, and J. Bi, "NOVA: Towards on-demand equivalent network view abstraction for network optimization," in *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, 2017, pp. 1–10.

[8] E. Pujol, I. Poese, J. Zerwas, G. Smaragdakis, and A. Feldmann, "Steering hyper-giants' traffic at scale," in *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '19. Association for Computing Machinery, Inc, 2019, pp. 82–95.

[9] Y. Zhang, G. Li, C. Xiong, Y. Lei, W. Huang, Y. Han, A. Walid, Y. R. Yang, and Z.-L. Zhang, "Mowie: Toward systematic, adaptive network information exposure as an enabling technique for cloud-based applications over 5g and beyond (invited paper)," in *Proceedings of the Workshop on Network Application Integration/CoDesign*, ser. NAI '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 20–27.

[10] S. Kiesel, W. Roome, R. Woundy, S. Previdi, S. Shalunov, R. Alimi, R. Penno, and Y. R. Yang, "Application-Layer Traffic Optimization (ALTO) Protocol," RFC 7285, Sep. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7285.txt

[11] W. Jiang, Z. S. Rui, J. Rexford, and M. Chiang, "Cooperative content distribution and traffic engineering in an ISP network," in *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '09, vol. 37. ACM Press, 2009, pp. 239–250.

[12] Y. Zhao, A. Saeed, M. Ammar, and E. Zegura, "Unison: Enabling Content Provider/ISP Collaboration using a vSwitch Abstraction," in *Proceedings the 27th International Conference on Network Protocols (ICNP)*. IEEE, 2019, pp. 1–11.

[13] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying Software-Defined Network Optimization Using SOL," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 223–237.

[14] Y. Rekhter, S. Hares, and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4271.txt

[15] H. Gredler, J. Medved, S. Previdi, A. Farrel, and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP," RFC 7752, Mar. 2016. [Online]. Available: https://rfc-editor.org/rfc/rfc7752.txt

[16] J. Zhang, L. Contreras, K. Gao, F. Cano, P. Cano, A. Escribano, and Y. R. Yang. (2021) Sextant Technical Report. [Online]. Available: https://www.dropbox.com/s/r0s7kbsbcwudmwu/sextant-tech-report.pdf?dl=0

[17] W. Roome and Y. R. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)," RFC 8895, Nov. 2020. [Online]. Available: https://rfc-editor.org/rfc/rfc8895.txt

[18] S. Knight, H. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 9, pp. 1765 –1775, october 2011.

[19] J.-J. Pansiot, P. Mérindol, B. Donnet, and O. Bonaventure, "Extracting intra-domain topology from mrinfo probing," in *Proceedings of the 11th International Conference on Passive and Active Measurement*, ser. PAM'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 81–90.

[20] T. Bates, P. Smith, and G. Huston. (2020) Cidr report. [Online]. Available: https://www.cidr-report.org/as2.0/

[21] S. Shenker, M. Casado, T. Koponen, N. McKeown *et al.*, "The Future of Networking, and the Past of Protocols," *Open Networking Summit*, vol. 20, pp. 1–30, 2011.

[22] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.

[23] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A Distributed Control Platform for Large-scale Production Networks." in *OSDI '10*, vol. 10, 2010, pp. 1–6.

[24] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Openday-

light: Towards a model-driven SDN controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2014.

[25] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an Open, Distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14. ACM, 2014, pp. 1–6.

[26] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A Software Defined Internet Exchange," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 551–562, 2015.

[27] A. Wang, X. Mei, J. Croft, M. Caesar, and B. Godfrey, "Ravel: A Database-Defined Network," in *Proceedings of the Symposium on SDN Research*, ser. SOSR '16. ACM, 2016, pp. 5:1–5:7.

[28] Q. Xiang, S. Chen, K. Gao, H. Newman, I. Taylor, J. Zhang, and Y. R. Yang, "Unicorn: Unified resource orchestration for multi-domain, geo-distributed data analytics," in *IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2017, pp. 1–6.