INT-filter: Mitigating Data Collection Overhead for High-Resolution In-band Network Telemetry

Enge Song*, Tian Pan*†, Chenhao Jia*, Wendi Cao‡, Jiao Zhang*, Tao Huang*, Yunjie Liu*
*State Key Laboratory of Networking and Switching Technology, BUPT, Beijing 100876, China
†Purple Mountain Laboratories, Nanjing 211111, China ‡Peking University, Beijing 100871, China
*{songenge, pan, 564822241, jiaozhang, htao, liuyj}@bupt.edu.cn ‡caowendi@pku.edu.cn

***Abstract*—In-band Network Telemetry (INT) enables fine-grained network monitoring to ease the management of large-scale networks, which, however, relies on the real-time collection of a huge amount of telemetry data through the southbound interface. For example, the INT telemetry data upload rate of a 28-pod FatTree topology reaches 3Tbps under a probe frequency of 100 times/s, which is rather unacceptable since the controller-switch link bandwidth is limited. To mitigate the telemetry data collection overhead, in this work, we propose *INT-filter*, a novel measurement architecture that deploys the same prediction algorithm on both the data plane and the control plane to predict the traffic state in the near future instead of uploading all the telemetry data. Such prediction-based approach leverages the observation that there is considerable redundancy in the telemetry data sequence. In addition, we design an integration mechanism that conducts predictions using multiple methods simultaneously and uploads the predicted result from the least-error method to further decrease the upload volume. Extensive evaluation suggests that INT-filter can achieve at least 33.6% data collection decrease under a 10ms probe interval. With prediction integration, the upload reduction can further reach 58.5%.**

I. Introduction

At present, conducting fine-grained, network-wide traffic monitoring plays an increasingly significant role in maintaining large-scale computer networks [1]. It enables fine-grained network-wide visibility, allowing the fast detection and localization of network gray failures [2, 3]. It also helps improve the network traffic load balancing with the prior knowledge of link congestion [4]. The network-wide traffic monitoring can be well applied to all types of networks, especially the data center networks, where traffic is highly aggregated and dynamic, with occasional and silent network failures, while user perception of network latency is expected to be persistently guaranteed [5]. With high-resolution network-wide telemetry, any port failure or traffic choke point can be spotted immediately before taking rapid response measures.

In traditional network monitoring, management protocols, such as SNMP [6], are coarse-grained and involve a large device query latency due to the frequent interactions between the control plane and the data plane of each underlying network device. To ameliorate the performance issue, In-band Network Telemetry (INT) [7] is proposed to achieve fine-grained monitoring. INT allows packets to query device-internal states, such as queue depth and queuing latency, when they pass through the data plane pipeline, without requiring additional intervention from the control plane CPU. At the last hop of the path, the packet containing the end-to-end monitoring data will be uploaded to a remote controller for further data analysis. The ability to arbitrarily write device states into probe packet headers in INT is essentially supported by the protocol-independent forwarding architecture, *i.e.*, P4 [8]. Compared to SNMP, INT interacts with the controller only at the last hop of the monitoring path, thereby reducing a large amount of device interruption on the controller. The "in-band" state collection paradigm also allows INT to achieve high-resolution network monitoring with a high probe frequency, which is extremely helpful when monitoring microbursts in data center networks [9].

As a device-level primitive, INT simply defines the interaction between the device-internal states and the incoming query packets, which is, however, insufficient for completely monitoring every link of the network graph. Further orchestration on INT to generate multiple monitoring paths to cover the entire network is essential for network-wide telemetry. For example, HULA [4] adopts the ToR switches to pour INT probes into the multi-root topology to achieve measurement coverage. INT-path [10] collects the network topology and uses an algorithm based on Euler trail to generate non-overlapped probing paths covering the whole network with a minimum number of paths. For either method, at the last hop of each monitoring path, the INT telemetry data needs to be uploaded to the controller through the *southbound interface* for analysis, the bandwidth of which, however, is not limitless. For high-resolution monitoring (*i.e.*, high probe frequency) at mega-scale network topology (*i.e.*, more probing paths), the southbound bandwidth occupation by the INT telemetry data will become much graver. In addition, the southbound interface is also at the service of other functions such as Packet-In messages and flow entry insertions [11], which need sufficient bandwidth as well. Since both HULA and INT-path upload all the telemetry data directly to the controller, according to our measurement and analysis in §II, HULA's upload rate even reaches 3.03Tbps under a massive-scale data center FatTree topology, which is a quite terrible number for the bandwidth occupation of probe packets. Moreover, the upload rate of INT-path also reaches 702Mbps despite its path optimization.

This work is supported by the National Key Research and Development Program of China (No. 2019YFB1802600), the National Natural Science Foundation of China (NSFC) (No. 61702049), and the Fundamental Research Funds for the Central Universities. Corresponding author: Tian Pan.

To tackle the above-mentioned problems, in this work, we propose *INT-filter*, which aims to decrease the southbound bandwidth occupation for high-resolution In-band Network Telemetry. Unlike previous work [10] to minimize the probing path number, INT-filter is designed to reduce unnecessary uploaded INT telemetry data via conducting historical data-based prediction. Such design relies on our observation that the most common device states, such as queue depth, do not always change drastically or irregularly during a short period of time, which can be well predicted with some linear or nonlinear regression methods according to a window of historical data. Specifically, we deploy the same prediction algorithm on the data plane device (the data plane device can be either a switch in the DCN or a router in the WAN) and on the controller to predict current values of the device states based on historical data. If there is minor difference between the current value and the predicted value at the data plane, the data plane will notify the controller to use the predicted value at the controller side by uploading a 1-bit notification flag. Compared with the 32-Byte INT telemetry data, the 1-bit flag greatly mitigates the uploaded data volume. In essence, INT-filter can well co-work with other methods [10, 12–14] to reduce the bandwidth overhead of INT telemetry data collection by filtering out the redundant collected data. Furthermore, we introduce an prediction integration mechanism that uses the first-, second-, and third-order polynomial fitting for prediction, and uploads the least-error method to the controller. With prediction integration, the reduction of upload rate further increases from 33.6% to 58.5%.

The main contributions are summarized as follows:

- We investigate the southbound bandwidth overhead of collecting INT telemetry data using HULA and INT-path, two existing network-wide telemetry proposals, in a data center FatTree topology of different scales. Via both measurement and theoretical analysis, we have come to the conclusion that without dedicated optimization measures, the southbound interface bandwidth cannot meet the needs of high-resolution telemetry data collection.
- We propose INT-filter, which enables the data plane to opportunistically upload only a 1-bit flag instead of the 32-Byte INT telemetry data via prediction, thus depleting unnecessary uploads as well as the southbound bandwidth occupation. We design the packet format for INT collection and INT upload and use software P4 switches [15] to implement an INT-filter prototype, which is open sourced at out git repository [16]. Extensive evaluation indicates that INT-filter can achieve 58.5% data collection decrease under a probe interval of 100 times per second.
- In order to further improve the prediction accuracy, we proposed prediction integration with simultaneous employment of first-, second-, and third-order polynomial fitting. With different methods dedicatedly handling different types of data, the prediction accuracy is well improved and the uploads are further decreased.

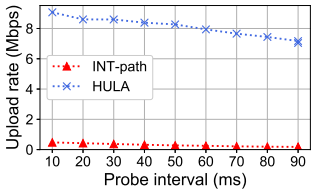

Fig. 1. Upload rates under different probe intervals (INT-path vs. HULA).

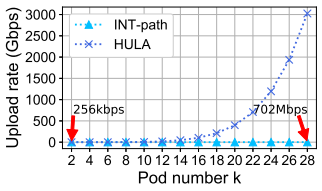

Fig. 2. Upload rates under different network sizes of FatTree topology.

II. Telemetry Data Collection Overhead

In order to obtain real-time device-internal states, INT probe packets are injected into the network and collected through the southbound interface. These probes will occupy the network bandwidth of not only the data plane, but also the interaction channel between the data plane and the central controller. The upload rates of INT-path [10] and HULA [4], two network-wide telemetry approaches built upon INT, are measured and analyzed at different probe frequencies as well as different topological scales for understanding the impact in depth.

Impact of the probe interval. Fig. 1 shows the upload rates of INT-path and HULA under different probe intervals in a small FatTree [17] topology (the same as the topology in the HULA paper [4]). HULA's basic idea is that the ToR switches will broadcast probes for link state detection. Such mechanism will cause multiple probes of a switch port in a short period time, although the most common link state does not change much in such a short time (such as queue depth), *i.e.*, conducting redundant probing. In contrast, with INT-path, we can calculate eight distinct paths that can probe all ports without redundancy, *i.e.*, no probing path overlapping. As shown in Fig. 1, HULA's probe upload rate maximally reaches 9.07Mbps while its upload rate decreases with the increase of the probe interval. In contrast, the upload rate of INT-path is around 0.4Mbps. We can draw a conclusion that for a small-scale topology, the non-redundant probing is practicable by drastically reducing the bandwidth occupation of the southbound interface. But in the face of today's mega-scale data center topologies, will such mechanism still work?

Impact of the topological scale. We theoretically analyze the upload rates of a k-ary FatTree as introduced in [17]. The k indicates its pod number, and each pod contains two layers of $k/2$ switches. We assume that each switch in the lowest layer is directly connected with $k/2$ hosts. We calculate that the total number of hosts (*i.e.*, odd vertices in the graph) is $k^3/4$. Therefore, there are $k^3/4$ non-overlapped directed probing paths obtained through INT-path [10]. The probe frequency is fixed to 100 times per second. The size of the INT telemetry data for each hop is 32B. According to INT-path, each host sends a probe packet to another host that is not in the same pod, and the size of the end-to-end INT telemetry data is always 160B (*i.e.*, with 5 hops). So the upload rate under the INT-path architecture is $k^3/4 * 160 * 100 * 8$bps. In the

According to INT-path, the minimized non-overlapped undirected path number is the same as the pair of odd vertices. In this paper, we consider the directed path number, which doubles the undirected path number.

HULA architecture, each switch in the lowest layer broadcasts probe packets, and $k/2$ switches in the higher layer in the same pod will receive its broadcast probes. Each switch in the higher layer will further send broadcast probes to the other $(k/2-1)$ lowest-layer switches in the same pod, so each time there will be $(k/2)*(k/2-1)$ probes (each with 64B (2*32B) INT telemetry data) generated and uploaded to the controller. In addition, each switch in the higher layer will also send broadcast probes to $k/2$ spine switches in the highest layer. Then, each spine switch sends broadcast probes to $(k-1)$ other higher-layer switches. These higher-layer switches will further send broadcast probes to the $k/2$ lowest-layer switches in the same pod. These lowest-layer switches will receive probe packets with a size of 128B (4*32B) INT telemetry data and upload them to the controller. Since the k-ary FatTree topology has $k*(k/2)$ lowest-layer switches, the upload rate under the HULA architecture is $((k/2)*(k/2-1)*64+(k/2)^3*(k-1)*128)*(k*(k/2))*100*8$bps. In Fig. 2, HULA's upload rate increases rapidly along with the network size, terribly reaching 3.03Tbps at $k=28$. The upload rate of INT-path also increases from 256kbps to 702Mbps. The evaluation result is the proof of the tremendous bandwidth required to transmit telemetry data between the data plane and the controller, especially in the case of large network topology, which, however, is very common in modern data centers.

III. Design and Implementation of INT-filter

Design Overview. The result in §II demonstrates that the communication of INT telemetry data between the data plane and the controller requires huge network bandwidth occupation. The most common device-internal states, such as queue depth, however, do not always change randomly in a short time. With simple prediction algorithms, the states not changing drastically or changing regularly can be predicted according to their historical values at the controller. This will help significantly reduce the volume of data uploaded from the data plane to the controller. With this basic idea, we deploy the same prediction on the data plane and on the controller to predict current values of the device-internal states based on historical values. If there is minor difference between the current value and the predicted value at the data plane, the data plane will notify the controller to use the predicted value at the controller side by uploading a 1-bit notification flag. Compared to the 32-Byte INT telemetry data, the 1-bit flag greatly reduces the uploaded data volume. Since INT-path is more lightweight due to its non-overlapped probing path coverage, the proposed INT-filter is built upon the underlying INT-path architecture to achieve a best-effort mitigation of the switch-controller channel bandwidth consumption.

Packet encapsulation. The INT-based data plane state collection can be partitioned into two phases. The first phase is hop-by-hop device-internal state collection when the probe packet traverses through multiple devices. The second phase is end-to-end telemetry data uploading to the controller at the last hop. Since INT-filter relies on the underlying INT-path for probing path planning, the packet format of the first phase

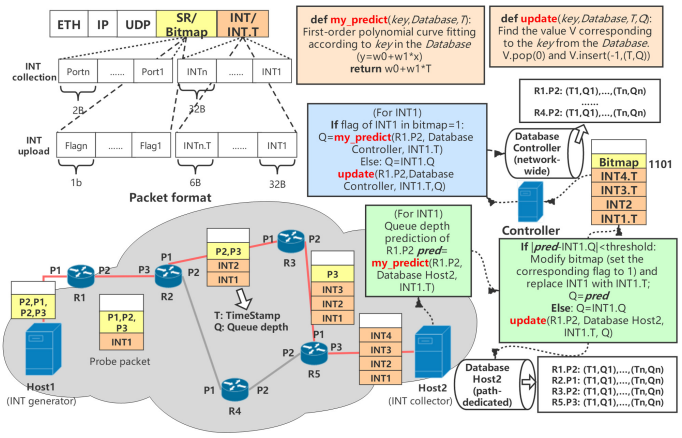

Fig. 3. In-band Network Telemetry with state collection filtering (INT-filter).

is exactly the same as that of INT-path, with source routing (SR) label stack for probe forwarding and INT telemetry data stack for device state collection. The difference lies in the second phase. INT-path uploads the probe packet with the complete end-to-end INT telemetry data received at the last-hop device to the controller. By contrast, INT-filter reduces the bandwidth consumption of the switch-controller channel by conducting "probe filtering" and uploading a rather slim packet, which is composed of a bitmap with flags indicating whether the predicted values in the controller can be directly used. Essentially, there is a one-to-one correspondence between the flags and the INT telemetry data collected from all devices along the end-to-end path. If the data plane expects the controller to use the predicted value for a specific device, it will replace the 32B INT telemetry data collected from that device with a 6B TimeStamp and set the corresponding flag for notification. Such filtering behavior will reduce the traffic uploaded since some telemetry data is predicted directly at the controller rather than being transmitted (as shown in Fig. 3).

Forwarding behavior. The probe packet forwarding behavior is exactly the same as that of INT-path. At each hop, the probe is forwarded according to the port ID embedded in the SR field and the device-internal states of the port will be recorded into the corresponding INT telemetry data field.

Telemetry result collection. The key of INT-filter is that the data plane and the controller use the same prediction algorithm and the same historical data for prediction, so as to ensure that the same predicted value can be obtained. As shown in Fig. 3, the *first-order polynomial fitting* ($y=w_0+w_1*x$) is taken as an example for illustration purpose. Generally, such prediction method using curve fitting is based on the historical data over a period of time which we call the *prediction windows size (m)*. In INT-filter, each probe packet is originated from the *INT generator* and collected by the *INT collector* at the end of the probing path before being forwarded to the controller. Each *INT collector* maintains a database that stores historical records of each device port. As depicted in Fig. 3, the Database Host2 stores the latest n records of the 4 ports that the probe packets sent by Host1 pass through, with a record format *(TimeStamp, Value)*. Correspondingly, the controller also has

a database for storing historical records of all ports on the data plane (*i.e.*, the predicted value at the controller using the same prediction algorithm is very close to the telemetry data collected at the data plane). In our experiment, the queue depth of the port serves as the recorded data plane state. The controller will either collect the uploaded real queue depth or just use the local predicted value to reduce the upload traffic.

Algorithm 1 Data plane filtering algorithm

Require: $probe\ packet$
Ensure: $modified\ probe\ packet\ after\ filtering$
while Receive an INT probe packet **do**
  Parse the probe packet into INT telemetry data
  for Each INT telemetry data INT_i collected from router R_j **do**
    Queue depth prediction of a specific router port $R_j.P$
    i.e., $pred$ = $my\_predict(R_j.P, Database\ Host, INT_i.T)$
    if $|pred - INT_i.Q| < threshold$ **then**
      Set the corresponding flag to 1 in the bitmap
      $Q = pred$
    else
      $Q = INT_i.Q$
    end if
    $update(R_j.P, Database\ Host, INT_i.T, Q)$
  end for
end while

Redundant information filtering. First, we expound on how the *INT collector* processes the received probe packet with end-to-end INT telemetry data to generate the compressed packet with a bitmap of flags. When receiving the probe packet, Host2 reads the historical value in the Database Host2 and applies the *first-order polynomial fitting* for curve fitting. Mathematically, the least squares are used for solving the n-order polynomial fitting, that is to say, the corresponding coefficients are solved with the minimum mean square error. Specifically, when we use a set of historical data $(x_i, y_i), i = 1, 2, \cdots, m$ to fit the curve $y = w_0 + w_1 * x + \cdots + w_n * x^n$, the coefficients $w_0, w_1, \cdots, w_n$ corresponding to the minimum value of $Q(w_0, w_1, \cdots, w_n) = \sum_{i=1}^{m}(w_0 + w_1 * x_i + \cdots + w_n * x_i^n - y_i)^2$ are solved for best curve fitting. Next, the TimeStamp as the independent variable for prediction is read out from the probe packet as the input of the fitted curve to obtain the predicted value (as implemented in the algorithm $my\_predict()$ in Fig.3). If the error between the true value and the predicted value is less than a predefined threshold, the predicted value and the corresponding TimeStamp will be stored in the Database Host2. Otherwise, Host2 will store the true value and its TimeStamp into the database. During the database update, the oldest record will be eliminated to ensure the fixed-size database always maintains the most recent historical data. The above procedure is performed on each piece of INT telemetry data. If the predicted value is stored, the corresponding flag in the Bitmap will be set to 1 and the INT data will be replaced with the corresponding TimeStamp. Otherwise, the flag will be set to 0. The modified probe packet is then uploaded to the controller. It should be noted that our prediction is based on each port's own data, so there is no need for time synchronization of each switch. Algorithm 1 formulates the data plane filtering process.

Second, we elaborate on how the controller handles the compressed packet and updates the local database. When the controller receives the uploaded probe packet, it first reads the Bitmap. For a particular INT telemetry data, if the corresponding flag in the Bitmap = 1, the controller will read the local historical value in the database, and predict the queue depth of the TimeStamp in the probe packet using the simple first-order polynomial fitting. Then, the controller will store the predicted value and its TimeStamp in the local database. Otherwise, the true queue depth and the corresponding TimeStamp will be stored in the database. This approach provides a significant reduction of switch-controller channel bandwidth consumption, since for each piece of well-predicted INT telemetry data, only 1-bit flag and 6B TimeStamp are uploaded instead of the original 32B telemetry data. Algorithm 2 formulates the control plane telemetry data recovering process.

Algorithm 2 Control plane recovering algorithm

Require: $modified\ probe\ packet\ after\ filtering$
while Receive an INT probe packet **do**
  Get the bitmap and parse the probe packet into INT telemetry data
  for Each INT telemetry data INT_i collected from router R_j **do**
    if flag of INT_i in bitmap = 1 **then**
      $Q = my\_predict(R_j.P, Database\ Controller, INT_i.T)$
    else
      $Q = INT_i.Q$
    end if
    $update(R_j.P, Database\ Controller, INT_i.T, Q)$
  end for
end while

Prediction accuracy enhancement (prediction integration). In addition, polynomial curve fitting of different orders is suitable for predicting different types of data, such as first-order polynomial fitting for linear data and second-order polynomial fitting for simple-curve data. Selecting the most appropriate order in polynomial fitting will greatly improve the prediction accuracy. Since the true value is known when making predictions at the data plane, multiple polynomial fitting methods with different orders can be used simultaneously to report the least error method of the true value to the controller. In our implementation, first-order, second-order and third-order polynomial fitting are integrated for prediction in order to improve the prediction accuracy. Accordingly, the size of the flag in the Bitmap becomes 2 bits, where 00, 01, 10 and 11 represent the case of not using predicted values, fitting with first-, second- and third-order polynomial fitting, respectively. In the data plane, the least error method is selected from the prediction methods with error lower than the threshold, and then the corresponding flag is updated in the Bitmap. The controller selects the corresponding action based on the flag after receiving the modified probe packet.

IV. Evaluation

A. Experiment Setup

We build an emulation-based network testbed to demonstrate the performance of INT-filter. The hardware is a 40-core server (Lenovo System x3850 X6) with four E7-4800@2.0GHz CPUs and 512GB DRAM. The OS is Ubuntu 16.04. The testbed, composed of 1 controller, 4 spine switches, 4 leaf switches, 4 ToR switches and 8 servers, is on the

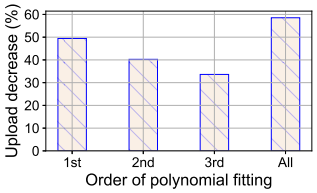

Fig. 4. The impact of order of polynomial fitting on upload decrease.

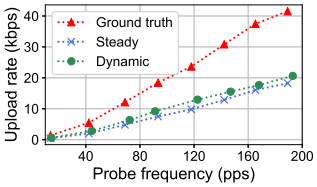

Fig. 5. The impact of data plane probe frequency on upload rate.

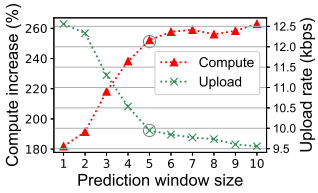

Fig. 6. The impact of prediction window size on the computational complexity and upload rate.

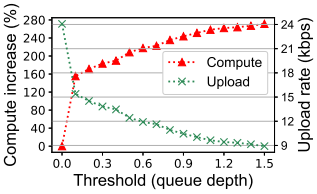

Fig. 7. The impact of threshold on the computational complexity and upload rate.

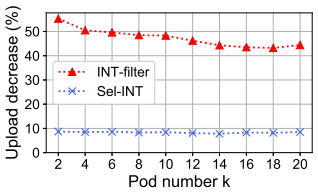

Fig. 8. Upload decrease under different network sizes of FatTree topology (INT-filter vs. Sel-INT).

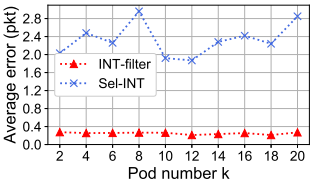

Fig. 9. Average error under different network sizes of FatTree topology (INT-filter vs. Sel-INT).

basis of Mininet [18], organized into the same FatTree topology with the HULA paper [4]. Among these eight servers, server1, server3, server6 and server8 send randomly-generated traffic (packet size = 1kB) to the other servers. The default background traffic rate is around 4.8Mbps. The INT-path algorithm [10] is employed for probing path planning. The length of INT header is 160B (32B * 5 hop) and SR header is 10B (2B *5 hop). *Prediction window size* is the amount of historical data used for prediction, with a default value of 5. *Threshold* is the limit used for determining whether or not to use the predicted value, with a default value of 1. If the error is less than or equal to the threshold, the INT collector will upload the flag corresponding to the least-error prediction method. We write 1323 lines of Python and 282 lines of P4 for the testbed, which are available in our git repository [16].

B. Experimental Results

Impact of order of polynomial fitting. Fig. 4 shows the impact of different prediction methods on upload decrease. First-order polynomial fitting has the best performance among the three basic methods. It can be seen that the maximum decline of upload rate is about 58.5% when the three fitting methods are used simultaneously. It suggests that the integration of multiple methods can indeed achieve the effect of selecting the most suitable method according to different data. We can also come to the conclusion that only 1-bit increase of the flag size decreases the upload rate by 25%. In subsequent experiments, the integration mechanism is adopted by default.

Impact of probe frequency. Fig. 5 shows the impact of probe frequency on upload rate. Here, *Ground truth* is the original upload without applying INT-filter. *Steady* corresponds to the stable background traffic while *Dynamic* corresponds to the background traffic with frequent bursts. It can be seen that the upload decrease grows gradually with the increase of probe frequency as more device-internal state predictions are uploaded instead of the original telemetry data. Furthermore, when the background traffic is stable, it is easy to predict, so the upload rate is lower than the dynamic case.

Impact of prediction window size. As shown in Fig. 6, the larger the prediction window is, the more accurate the prediction will be, and thus the lower the upload rate will be. But at the same time, the computational complexity increases as well with the increasing use of predictions. The computational complexity is quantified by the average time required to process the INT probe (*e.g.*, make predictions) and store the telemetry data into the database. INT-filter has a much higher computational complexity compared to not using it. We set the default prediction window size to 5, since in Fig. 6, 5 is a sweet spot for both accuracy and computational complexity.

Impact of threshold. The prediction accuracy is determined by the threshold. A large threshold value means that even in the case of insufficient prediction accuracy, the data plane can upload the prediction method rather than the original INT telemetry data. As shown in Fig. 7, with the increase of the threshold, the upload rate drops gradually, thus increasing the computational complexity at the controller. The default threshold is set to 1, a relatively appropriate value compared to the average queue depth of 50 measured in our experiment. It suggests that the error between the true queue depth and the value written in the database must be less than or equal to 1. The error is too small to affect controller's network applications, such as load balancing and failure localization.

Comparison of upload decrease (INT-filter vs. Sel-INT). Sel-INT [14], a selective INT telemetry system, is composed of a selective INT header insertion module and a *Data analyzer*. The *Data analyzer* of Sel-INT compares the newly acquired device-internal state with the latest value recorded in the database, and does not upload the data with minor change, thereby reducing the upload volume. In order to compare the performance of INT-filter and Sel-INT in filtering redundant telemetry data, we perform a simple simulation in Python. The variable-scale topology is the same as the one mentioned in §II. As shown in Fig. 8, the upload decrease rate of INT-filter gradually declines as the network size increases. Since there are more packet generators in the network, the number of packets passing through each switch per second varies more greatly, making it harder to predict. However, it can be seen that the upload decrease of INT-filter is invariably much higher than that of Sel-INT (about 6 times). This indicates that our algorithm performs better in redundant information filtering.

Comparison of average error (INT-filter vs. Sel-INT). Since INT-filter invokes predictions rather than uploading true values and Sel-INT [14] does not upload telemetry data with minor variation, there will be a gap between the controller-side

network telemetry results and the underlying real device states. We calculate the average error of 100 pieces of INT telemetry data processed by INT-filter and Sel-INT, respectively. The error threshold of INT-filter is set to 1 by default, which means that the error caused by each piece of INT telemetry data is a maximum of 1. Moreover, because only part of the INT telemetry data is filtered out, the average error rate is far less than 1. As shown in Fig. 9, the average error rate of INT-filter is mere 1/10 of that of Sel-INT, and neither of them changes rapidly with the network size. Through the analysis of Fig. 8 and Fig. 9, it can be concluded that compared with Sel-INT, INT-filter not only filters out more redundant INT telemetry data but also has a much lower error rate.

V. Related Work

Openflow [11], completely decoupling the control plane from the data plane, requires frequent interactions of various information between the switch and the controller through controller's southbound interface, which is the bottleneck of efficient network management. Gao et al. [19] reduce the southbound bandwidth occupation by an effective detection method that defeats the Packet-In messages flooding attack with low overhead and high accuracy. Nevertheless, their method cannot reduce the INT telemetry data bandwidth occupation of the southbound interface since it is not only an excessive consumption of flow table resources, but also cannot deal with the changing device-internal states. [12] and [13] reduce the uploaded INT telemetry data by dynamically adjusting the detection frequency of the data plane to reduce unnecessary detection. The basic idea behind their architecture is to reduce the detection frequency when the telemetry data does not change significantly, such as queue depth. They can only reduce the probe frequency of state-stable nodes. However, it is impossible to handle situations where the node state is changing regularly but still predictable, such as the queue depth increasing/decreasing steadily. Tang et al. [14] propose Sel-INT, which can not only reduce the telemetry frequency of the INT data collection through the network, but also reduce the upload volume of the INT telemetry data through the southbound interface. In the Sel-INT architecture, the SDN controller will analyze historical INT data with the Fourier transform and set the sampling frequency to twice the lowest point frequency. The last-hop switch duplicates the INT packet to the Data Analyzer for network monitoring and further processing. The Data Analyzer then reports most significant information to the controller. In the same way, their method cannot deal with situations where the state change of the node is regular and can be predicted. The INT-filter proposed in this paper uses the idea of predicting through historical data, which can reduce the uploads of not only nodes with stable status, but also nodes with predictable status. In addition, as a standalone module, our method can co-work with other methods, such as [12–14].

VI. Conclusion

In this work, we propose *INT-filter*, a highly efficient data collection architecture for high-resolution In-band Network Telemetry. Instead of uploading the original telemetry data, INT-filter uploads a flag indicating whether or not to use the predicted value, thus reducing redundant uploads. In addition, the combination of different prediction methods significantly improves the prediction accuracy, further reducing the amount of telemetry data collected. The saved bandwidth can further be occupied by other critical control messages that need to be timely transmitted through the southbound interface.

References

[1] T. Pan, E. Song, C. Jia, W. Cao, T. Huang, and B. Liu, "Lightweight network-wide telemetry without explicitly using probe packets," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 1354–1355.

[2] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 139–152.

[3] C. Jia, T. Pan, Z. Bian, X. Lin, E. Song, C. Xu, T. Huang, and Y. Liu, "Rapid detection and localization of gray failures in data centers via in-band network telemetry," in *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, pp. 1–9.

[4] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*. ACM, 2016, p. 10.

[5] J. R. Hearn, P. Connor, K. Sood, S. P. Dubal, and A. J. Herdrich, "Real-time local and global datacenter network optimizations based on platform telemetry data," Mar. 30 2017, uS Patent App. 14/866,869.

[6] J. Case, M. Fedor, M. L. Schoffstall, and J. Davin, "Rfc1157: Simple network management protocol (snmp)," 1990.

[7] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in *ACM SIGCOMM*, 2015.

[8] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[9] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proceedings of the 2017 Internet Measurement Conference*, 2017, pp. 78–85.

[10] T. Pan, E. Song, Z. Bian, X. Lin, X. Peng, J. Zhang, T. Huang, B. Liu, and Y. Liu, "Int-path: Towards optimal path planning for in-band network-wide telemetry," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 487–495.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[12] Y. Kim, D. Suh, and S. Pack, "Selective in-band network telemetry for overhead reduction," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE, 2018, pp. 1–3.

[13] D. Suh, S. Jang, S. Han, S. Pack, and X. Wang, "Flexible sampling-based in-band network telemetry in programmable data plane," *ICT Express*, vol. 6, no. 1, pp. 62–65, 2020.

[14] S. Tang, D. Li, B. Niu, J. Peng, and Z. Zhu, "Sel-int: A runtime-programmable selective in-band network telemetry system," *IEEE Transactions on Network and Service Management*, 2019.

[15] "P4 switch behavioral model," https://github.com/p4lang/behavioral-model, 2018.

[16] "Int-filter repository," https://github.com/graytower/INT_FILTER, 2020.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4. ACM, 2008, pp. 63–74.

[18] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[19] D. Gao, Z. Liu, Y. Liu, C. H. Foh, T. Zhi, and H.-C. Chao, "Defending against packet-in messages flooding attack under sdn context," *Soft Computing*, vol. 22, no. 20, pp. 6797–6809, 2018.