Innovative network monitoring techniques through In-band Inter Packet Gap Telemetry (IPGNET)

Francisco Germano Vogt
University of Campinas (Unicamp)

Fabricio Rodriguez
University of Campinas (Unicamp)

Christian Rothenberg
University of Campinas (Unicamp)

Gergely Pongrácz
Ericsson Research

ABSTRACT

Network monitoring is a fundamental task for proper network troubleshooting and performance management. Recently, in-band Network Telemetry (INT) has been demonstrated as a powerful and efficient network monitoring framework. Using INT, network information hop-by-hop can be collected directly from the data plane by gathering this information in the production traffic. However, INT data collection is limited by available packet size and processing overhead, making it critical to choose what data to collect and when to collect it. In this demo, we propose the In-band Inter Packet Gap Network Telemetry (IPGNET) per-hop monitoring. We argue that by monitoring the IPG hop-by-hop, it is possible to correlate the data and identify: (i) Network problems like congestion and delays, finding their root cause, and (ii) Microbursts and their contributing flows. Our preliminary results show that IPGNET can detect microbursts on multiple queues and report all the contributing flows with high efficiency in terms of control/data plane overhead.

CCS CONCEPTS

• **Networks** → **Network manageability**; **Programming interfaces**; • **Computer systems organization** → *Maintainability and maintenance.*

KEYWORDS

P4, Software Defined Networking, Network Management

ACM Reference Format:
Francisco Germano Vogt, Fabricio Rodriguez, Christian Rothenberg, and Gergely Pongrácz. 2022. Innovative network monitoring techniques through In-band Inter Packet Gap Telemetry (IPGNET). In *P4 Workshop in Europe (EuroP4 '22), December 9, 2022, Roma, Italy.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3565475.3569077

1 INTRODUCTION

In-band Network Telemetry (INT) [24] is an emerging technique for network monitoring. Different from the traditional network monitoring techniques like ICMP [9], Traceroute [1], NetFlow [23], and SNMP [10], INT can provide finer-grained network monitoring with better network visibility and coverage [15]. Through programmable devices and high-level languages like P4 [3], INT can collect a large and flexible set of information (e.g., ingress timestamp, queue length) by the network. This information can be used to detect and mitigate several network problems, like network congestion, packet loss, and delay. Each application used to detect and mitigate these problems requires a specific set of network information. However, INT data collection is limited by some network factors (e.g., bandwidth and latency of INT processing [14]), making it too costly to collect all network information every time due to the associated overheads.

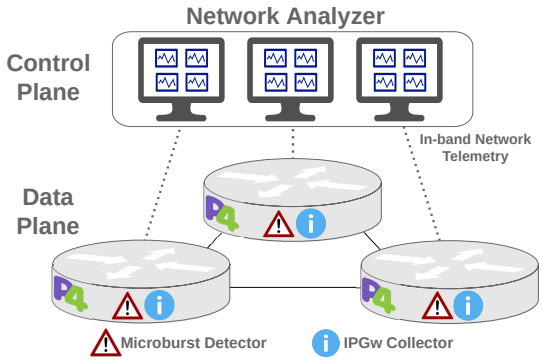

Figure 1: IPGNET architecture.

Due to scalability and overhead concerns when running INT, the collected information must be chosen wisely. Recent research efforts have focused on INT orchestration, maximizing, and optimizing information collection [11] [4]. In this work, we argue that instead of collecting more network information, it could be better to collect information that can be used for more multiple use cases. In this context, the Inter Packet Gap (IPG) is a promising network measurement and management metric because it can be used to identify with a high accuracy larger set of events than the traditional metrics in addition to being a small size metric to collect (e.g., 16 bits of IPG size to heavy hitters detection [22]). In the literature, IPG has proven to be efficient for problems like packet loss [21], delay measurement [18], and heavy hitters detection [22]. Furthermore, we argue that by collecting the IPG hop-by-hop using INT, we can detect and report additional events like network congestion and microbursts, besides finding their root causes.

In this demo, we present In-band Inter Packet Gap Network Telemetry (IPGNET)[1], a system to collect, correlate, and analyze the IPG metric hop-by-hop. The main idea of IPGNET is to gather the IPG per-hop using INT to analyze and identify different network scenarios like network delay and microbursts. Additionally to detecting

[1]Public available at: https://github.com/intrig-unicamp/IPGNET

EuroP4 '22, December 9, 2022, Roma, Italy

ACM ISBN 978-1-4503-9935-7/22/12…$15.00
https://doi.org/10.1145/3565475.3569077

these events, IPGNET can find the root cause of the problems and identify the contributing network flows in a microburst scenario. The ability to detect and report these problems with low overhead can help ensure the performance of latency-sensitive applications [2] and help traditional network measurement systems [6][16] that cannot detect fine-grained problems such as microbursts.

Figure 1 shows the IPGNET architecture with an IPG weighted (IPGw) collector and a microburst detector running on the data plane while a network analyzer runs on the control plane. In our system, the IPG collector is responsible for calculating and reporting the IPG weighted metric, while the microburst detector is responsible for detecting the microbursts and reporting the contributing flows. Finally, the network analyzer is responsible for receiving and correlating the data to identify the problems and their causes.

2 IPGNET: CONCEPTS & DESIGN

2.1 In-band network telemetry (INT)

Network programmability enables several benefits related to the management and operation of network infrastructures. More specifically, the flexibility offered by high-level data plane programming languages like P4. In this context, INT emerged as a network monitoring mechanism that provides higher network-wide visibility to network operators.

INT enables the collection of network statistics with higher granularity and flexibility. INT can collect several network information directly from the data plane (e.g., ingress timestamp, queue length, process time) in two ways: using active network flows or generating probe packets:

- **Active flows:** Telemetry data is embedded in user packets by including an instruction header in the network packet that enables the information collection and defines what information should be collected on which devices.
- **Probe packets:** Are crafted and sent on a predefined route specifically to carry INT information. Similar to the previous strategy, probes carry INT headers that specify the information that should be collected. However, these probes are specifically created to load INT information and therefore do not carry user data adding extra traffic to the network.

In this demo, we use the active flows strategy to embed INT information, using the IPV4 options field to include the information. In IPGNET, instead of the traditional INT information defined by INT specification, we collect the IPGw metric as discussed next.

2.2 Inter-packet gap metric (IPG)

The IPG is a network metric that calculates the difference between the arrival time of two network packets. If measured between any two random network packets, this metric may not be as relevant information as a queue size, for example. However, if correctly measured and correlated can be used to detect several network events. Recent research efforts have demonstrated the IPG metric used to detect events such as packet losses [21], delay measurement [18], and most recently, in the SDN context, heavy hitters detection [22].

In this demo, we argue that we can detect several network events by collecting the IPG hop-by-hop and correlating this information. For this, we propose an IPG calculation per network flow, where a network flow is a set of packets with features in common (e.g., IP address, port number). In our scenario, we define a network flow as a 5-tuple (i.e., source IP, destination IP, protocol, source port, destination port). Therefore, the idea is to collect the hop-by-hop IPG from a network flow to analyze and correlate this information with other network flows to detect network events.

The proposed IPG metric does not represent just a measure between the last two packets received because IPGNET uses an Exponentially Weighted Moving Average (EWMA) for the IPG calculation, as discussed in [22]. When a packet arrives on the switch, the difference between it and the last packet received is calculated by Equation 1, where TS_c is the current timestamp, and TS_l is the timestamp of the last packet.

$$IPG_c = TS_c - TS_l \tag{1}$$

Then, the IPGw is calculated with the EWMA function by Equation 2, where IPG^{w-1} is the last weighted IPG. In this demo, we fixed the $\alpha = 0.99$ as better described in [22].

$$IPG^w = \alpha \cdot IPG^{w-1} + (1 - \alpha) \cdot IPG_c \tag{2}$$

2.3 Problem detection and root cause analysis.

Considering a physical network infrastructure $G = (D, L)$ and a set of active network flows F. The set D in the network G represents all programmable forwarding devices, then $D = \{1, ..., |D|\}$, while set L represents the links interconnecting pair of devices $(d_1, d_2) \in (D \times D)$.

The network flows F are responsible for collecting their IPGw metric (using INT) in all devices in their paths. After the collection, we have a set M of IPGs measured for each flow in F. We can define the set of IPGs collected by a flow f as $M(f) = \{(IPG_1, d_1), ..., |M|\}$ corresponding the all pairs of IPGs and switches IDs on the path. IPGNET performs IPG variance calculation for all sets $M(f)$ with $f \in F$ for network problem detection. This variance calculation is defined by Equation 3.

$$\sigma_f^2 = \frac{\sum\limits_{m \in M(f)} (m - \overline{M})^2}{|M(f)|} \tag{3}$$

With this variance calculation, IPGNET can detect any change in the network behavior. Note that regardless of whether the throughput of the flows are different or may vary over time (e.g., flow sending more or less traffic at any given time), the IPG between each flow hop will not change unless there is a "problem" with the network (e.g., queuing delays).

Figure 2 illustrates an IPG variance distribution for three flows with different throughputs. These distributions represent 10k IPGs reports measured with our strategy in two Barefoot Tofino switches [17]. Note that even though the flows have different throughputs, the variation of the IPGw between the two devices tends to be small ($\approx$ less than $5\mu s$ in all cases) and, in most cases, close to zero. This behavior is because we do not have any network problems in our tests, causing the variance unaffected. To detect a network problem, we can define a variance threshold (e.g., latency-increase tolerance), and when the variance exceeds this threshold, we detect the problem. Then, to find the root cause of the problem, we can analyze the INT reports finding the first hop where the variance occurred, which signifies the source of the problem.

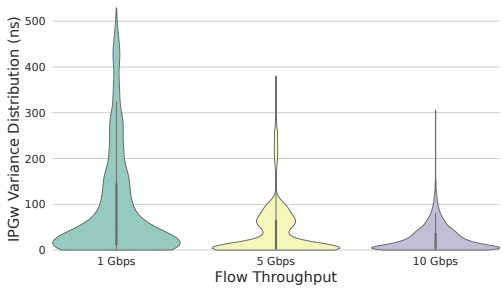

Figure 2: Example of IPGw variance

Algorithm 1 Microbursts detection and report

Input: $threshold$

if $deqDepth > threshold$ **then**
  $slotsRemain[qID] \leftarrow deqDepth$
  if $IPG_w^f < IPG_w^{avg}$ **then**
    ReportContributingFlow(pkt)
else
  if $slotsRemain[qID] > 0$ **then**
    $slotsRemain[qID] \leftarrow slotsRemain[qID] - pktLen/slotSize$
    if $IPG_w^f < IPG_w^{avg}$ **then**
      ReportContributingFlow(pkt)

2.4 Microburst detection and report

For microburst detection, IPGNET uses a full-data plane approach to detect the microburst and report the contributing flows. The main idea of this strategy is to detect the microbursts based on the queue depth and then report the contributing flows for the control plane based on their IPGs. So, IPGNET uses the same IPGw calculation (used for problem detection and root cause analysis) to decide whether a flow contributes to the burst or not. This decision is based on the moving average of flow IPGs that pass through the switch, and when the flow has an IPGw lower than this average, it is considered a contributing flow.

The concept of contributing flows, also covered in [8] and [5], refers to network flows with greater throughput than the other flows. However, [8] uses a hybrid strategy, which combines control and data plane to detect microbursts (and consequently includes a time overhead), and [5] uses a probabilistic technique which needs a large number of pipeline stages to get a decent recall. More details of IPGNET strategy in Algorithm 1.

3 OVERHEAD EVALUATION

3.1 Data plane

The data plane overhead is measured in terms of resource utilization of IPGNET P4 code compiled for the Tofino switch [17]. Table 1 compares the resource utilization of IPGNET with the switch.p4 [7] (baseline P4 program), and BurstRadar [13] state-of-the-art in microbursts detection in the data plane. In the table, we consider the BurstRadar with a ring buffer size of 1k entries, while IPGNET is computing the top-2k flows. Besides, in the case of IPGNET and BurstRadar, we consider the cost of strategies added to the switch.p4 cost. Overall, IPGNET uses fewer hardware resources than BurstRadar in all the analyzed resources.

Table 1: Hardware resource utilization

Resource	Switch.p4	BurstRadar	**IPGNET**
Hash Bits	32.3%	37.2%	34.4%
SRAM	29.8%	33.9%	31.1%
TCAM	28.4%	29.2%	28.4%
VLIW Actions	34.6%	39.3%	38.0%
Stateful ALUs	15.6%	28.2%	15.6%

3.2 Control plane

For the control plane overhead, we analyze the number of flows reported by IPGNET compared to BurstRadar [13] in a burst situation. A burst situation is considered when the occupancy of the switch queue is greater than the threshold defined by the network operator. To generate the experiment, we send bursts at a rate of 3 Gbps to a link with a capacity of 1GB, following the burst data distribution available in [25] and generated by [19]. For the background traffic (or normal traffic), we use the IMIX traffic distribution [12] at 100 Mbps to keep link occupancy at 10%, as described in [25] and totaling 557 flows (310 normal and 247 bursts).

When comparing our results to [13] (Table 2), it reports 100% of burst flows, but IPGNET only reports 9.03% of the non burst flows. Even though it manages to report all burst flows, IPGNET reports 50.63% fewer flows than [13], reducing the control plane overhead.

Table 2: Flows reported to the controller in a burst

Flows	BurstRadar	**IPGNET**
Burst Flows Reported	100%	100%
Non Burst Flows Reported	100%	9.03%
Total Flows Reported	100%	49.37%

4 DEMONSTRATION AND FUTURE WORK

During the demo. Attendees will be able to see how IPGNET performs when detecting microbursts and reporting contributing flows. They will be able to run IPGNET P4 code on a remote Tofino, configure network and microburst detection parameters, and send normal/burst traffic to see IPGNET in action. We will showcase the ability of IPGNET to report the flows contributing to bursts by inspecting the reports received by the remote monitoring server.
Future work. As a continuation of our work, we expect to finalize the complete implementation of IPGNET, evaluating further use cases by comparing them with state-of-the-art monitoring strategies. Furthermore, we plan to integrate with other platforms such as the P7 [20] emulator to validate IPGNET in large topologies, evaluating its accuracy, overhead and scalability.

ACKNOWLEDGMENTS

This work was supported by the Innovation Center, Ericsson Telecomunicações S.A., Brazil, under grant agreement UNI.70. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nivel Superior - Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] Kanak Agarwal, Eric Rozner, Colin Dixon, and John Carter. 2014. SDN traceroute: Tracing SDN forwarding without changing network behavior. In *Proceedings of the third workshop on Hot topics in software defined networking*. 145–150.

[2] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2010. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM 2010 Conference*. 63–74.

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* (2014).

[4] Ronaldo Canofre, A.G Castro, A.F Lorenzon, F.D Rossi, and M.C Luizelli. 2022. Towards Efficient Selective In-Band Network Telemetry Report Using SmartNICs. In *International Conference on Advanced Information Networking and Applications*. Springer, 271–284.

[5] Xiaoqi Chen, Shir Landau Feibish, Yaron Koral, Jennifer Rexford, and Ori Rottenstreich. 2018. Catching the microburst culprits with snappy. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*. 22–28.

[6] Benoit Claise. 2004. *Cisco systems netflow services export version 9.* Technical Report.

[7] P4 Language Consortium. 2018. Baseline switch.p4. https://github.com/p4lang/switch.

[8] Kaihui Gao, Dan Li, and Shuai Wang. 2022. Bandwidth-efficient Microburst Measurement in Large-scale Datacenter Networks. (2022).

[9] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. 2015. 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*. 139–152.

[10] David Harrington, Randy Presuhn, and Bert Wijnen. 2002. *An architecture for describing simple network management protocol (SNMP) management frameworks.* Technical Report.

[11] Rumenigue Hohemberger, A.G Castro, F.G Vogt, R.B Mansilha, A.F Lorenzon, F.D Rossi, and M.C Luizelli. 2019. Orchestrating in-band data plane telemetry with machine learning. *IEEE Communications Letters* 23, 12 (2019), 2247–2251.

[12] IMIX. 2021. IMIX Traffic. ttps://en.wikipedia.org/wiki/Internet_Mix.

[13] Raj Joshi, Ting Qu, Mun Choon Chan, Ben Leong, and Boon Thau Loo. 2018. BurstRadar: Practical real-time microburst monitoring for datacenter networks. In *Proceedings of the 9th Asia-Pacific Workshop on Systems*. 1–8.

[14] Youngho Kim, Dongeun Suh, and Sangheon Pack. 2018. Selective in-band network telemetry for overhead reduction. In *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*. IEEE, 1–3.

[15] Zhengzheng Liu, Jun Bi, Yu Zhou, Yangyang Wang, and Yunsenxiao Lin. 2018. Netvision: Towards network telemetry as a service. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 247–248.

[16] Arista Networks. 2015. Latency Analyzer (LANZ) Architectures and Configuration. https://goo.gl/LrRNi4.

[17] Barefoot Networks. 2018. Tofino Switch. https://goo.gl/cdEK1E.

[18] N.M Piratla, A.P Jayasumana, and H Smith. 2004. Overcoming the effects of correlation in packet delay measurements using inter-packet gaps. In *Proceedings. 2004 12th IEEE International Conference on Networks*. IEEE.

[19] Fabricio Rodriguez, P. Gyanesh Kumar Patra, Levente Csikor, Christian Rothenberg, Péter Vörös Sándor Laki, and Gergely Pongrácz. 2018. BB-Gen: A Packet Crafter for P4 Target Evaluation. In *Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos* (Budapest, Hungary) *(SIGCOMM '18)*. Association for Computing Machinery, New York, NY, USA, 111–113. https://doi.org/10.1145/3234200.3234229

[20] Fabricio Rodriguez, Francisco Germano Vogt, Ariel Góes De Castro, Marcos Felipe Schwarz, and Christian Rothenberg. 2022. P4 Programmable Patch Panel (P7): An Instant 100g Emulated Network on Your Tofino-Based Pizza Box. In *Proceedings of the SIGCOMM '22 Poster and Demo Sessions* (Amsterdam, Netherlands) *(SIGCOMM '22)*. Association for Computing Machinery, New York, NY, USA, 4–6. https://doi.org/10.1145/3546037.3546046

[21] Jirapon Sa-Ingthong, Anan Phonphoem, A. Jansang, and C Jaikaeo. 2021. Probabilistic Analysis of Packet Losses in Dense LoRa Networks. In *2021 13th International Conference on Knowledge and Smart Technology*. IEEE.

[22] Suneet Singh, Christian Rothenberg, Marcelo Luizelli, Gianni Antichi, and Gergely Pongracz. 2020. Revisiting heavy-hitters: don't count packets, compute flow inter-packet metrics in the data plane. In *Proceedings of the SIGCOMM'20 Poster and Demo Sessions*. 49–51.

[23] Robin Sommer and Anja Feldmann. 2002. NetFlow: Information loss or win?. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*. 173–174.

[24] Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. 2021. In-band network telemetry: A survey. *Computer Networks* 186 (2021), 107763.

[25] Qiao Zhang, Vincent Liu, Hongyi Zeng, and Arvind Krishnamurthy. 2017. High-resolution measurement of data center microbursts. In *Proceedings of the 2017 Internet Measurement Conference*. 78–85.