A P2P Overlay System for Network Anomaly Detection in ISP Infrastructures

Miguel Silva, Ricardo Mendonça
Department of Informatics
University of Minho
Braga, Portugal
mikedvlp@gmail.com, zepmend@hotmail.com

Pedro Sousa
Centro Algoritmi, Department of Informatics
University of Minho
Braga, Portugal
pns@di.uminho.pt

***Abstract* — This paper presents a P2P Overlay system for network anomaly detection, providing ISP network administrators with a versatile, easily configurable and useful tool to monitor the network infrastructures. In this context, this document describes the system general architecture and the main entities that integrate the proposed overlay system. Some technical details of the system will be provided, along with particular aspects of a real implementation of a system prototype, which was made using the JAVA language. The robustness and usefulness of the devised solution is proven resorting to the CORE network emulator platform. Such platform allows emulating a real ISP network infrastructure, over which a prototype of the proposed overlay system and integrating entities is tested and the corresponding results analysed. Some illustrative use cases of the system operation will be also presented and analysed.**

Keywords - P2P and Overlay Networks, Network Anomaly Detection, ISP, Monitoring.

I. INTRODUCTION

Internet Service Provider (ISP) networks are sometimes faced with anomalies that affect the service provided to the final users. Some of those anomalies are related with temporary or permanent faults of physical equipment of the network (e.g. links, routers, etc.), which lead to a degradation of the level of service provided to end users. Other anomalies come from situations of congestion originated from excessive levels of traffic in some critical points of the network. In such context, this article aims to devise and develop a distributed P2P based overlay system able to detect network anomalies and that can be easily configured and used by ISP network administrators.

P2P overlay networks differ in the way they work, and distinct benefits result from assuming centralized or decentralized approaches, the distinct protocols chosen, system parameters, and others related alternatives [4]. Overlay systems and P2P networks are widely used in several and heterogeneous scenarios. As simple examples, they can be used to handle network malicious attacks [1], or to provide solutions in the area of application level multicast [2,8], or even in the construction of overlay network simulators [3]. In the similar context of the research area here explored, and despite constituting distinct approaches from the proposed one, some other projects also take advantage of having a distributed probing infrastructure to provide network related measuring metrics, as the examples of the *Ripe Atlas* [9], the *perfSonar* [10], the *SameKnows* [11] and the *Nlnog Ring* [12] projects.

The system proposed in this paper is based on a highly reconfigurable overlay network integrating several peers, being able to perform real-time monitoring of the network infrastructure, storing the results into a database and providing an user-friendly interface with the ISP network administrator. The system also encompasses a network anomaly alarm module that can detect several network anomalies related to temporary or permanent faults, which may lead to a degradation of the Quality of Service (QoS) provided by the ISP. The overlay network is composed by a set of peers that establish P2P relations with other peers of the overlay network and, after receiving appropriate commands from the overlay coordinator, perform several measurements of the network infrastructure (e.g. packet loss, RTT, jitter, changes of routing paths, among many others).

This article is organised as follows: Section II presents the overlay system architecture and the main integrating entities. Section III focuses on the overlay internal communication processes, highlighting the probing commands generated by the overlay coordinator and the structure of some of the exchanged packets. After that, Section IV will give some details about the technologies used for the system implementation. Section V presents some illustrative results and use cases of the overlay system operation. Finally, in Section VI some conclusions are presented along with possible future work in this project.

II. SYSTEM GENERAL ARCHITECTURE AND ENTITIES

Figure 1 depicts in a simplified way the architecture and the main entities integrating the proposed system. As observed, the monitoring overlay system is composed by a central node, which acts as the overlay coordinator and a set of peers, distributed along the ISP infrastructure, responsible for probing the network infrastructure. The overlay peers are activated or deactivated by the coordinator, also receiving control commands determining which type of P2P relations to establish and which type of probing processes they should perform, using a control communication port for that purpose.

In the simple illustrative example of Figure 1, the monitoring overlay network is solely composed by two probing processes, between *peerA* and *peerB* and between

peerA and *peerC*. The results of the probing processes are then sent to the overlay coordinator through data transfer ports.

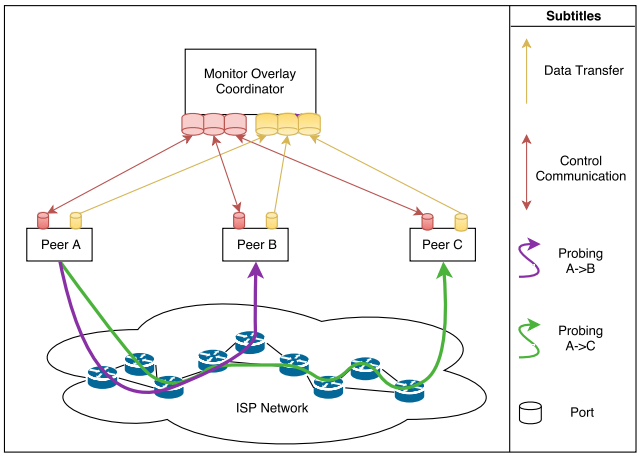

Figure 1. General Architecture of the Overlay System

A. Monitor Overlay Coordinator

The monitor overlay coordinator architecture is further detailed in Figure 2. As observed, it has multiple control communication ports to allow it to send several commands to the multiple existing peers in the overlay network, and it also has multiple data transfer ports to allow it to receive several measurement data reports collected by the overlay peers. Such probing related information will be stored it in an internal database module within the coordinator node. As also depicted in Figure 2, the coordinator will also have an interface with the administrator, allowing him to activate the overlay peers, to issue probing commands, to receive alarm notifications and several options to check the status of the ISP network, as will be latter detailed in this document.

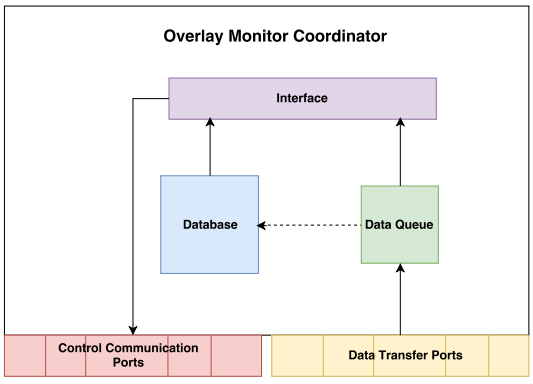

Figure 2. Monitor Overlay Coordinator Architecture

B. Overlay Peer

The overlay peer is the entity responsible for performing the probing processes in the ISP network. As visible in Figure 3, it has two distinct channels, one for control commands, which is needed to receive commands from the coordinator to be run on the peer, and send response packets confirmation back to the coordinator, thus requiring a control queue which works in both directions. The other channel is used for data transfer communications with the coordinator, being used for sending constant monitoring feedback to the coordinator. A data buffer is required for large data transfers, to temporarily store data while it is being transferred from the peer to the overlay coordinator. Additionally, as observed in Figure 3, each peer maintains a peer port list. This list has one entry for each peer in the overlay network for which the peer has established a probing process, previously triggered by the overlay controller. The peers will also have a controller module allowing the peer to know what measurements needs to be made and compute some statistical data, when that functionality is requested by the overlay coordinator.

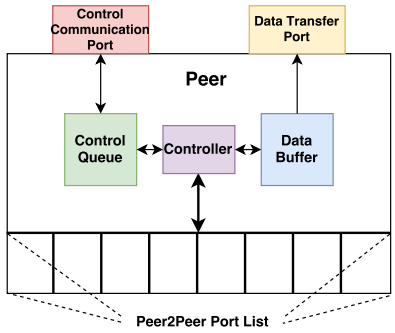

Figure 3. Overlay Peer Architecture

III. OVERLAY COMMUNICATION PROCESSES, PROBING COMMANDS AND PACKET FORMATS

This section highlights the overlay internal communication processes, focusing on the probing commands generated by the overlay coordinator node and the structure of some of the exchanged packets.

A. Coordinator to Overlay Peer Communications

The transport layer chosen for the control commands and the data transfer communications between the coordinator node and the overlay peers is the TCP protocol, in order to assure the reliability for the communications between these two entities. The number of control and data communication ports used by the overlay coordinator is the same as the number of the peers active in the overlay network, i.e. there is an independent communication port within the coordinator node for exchanging commands and data reports with each specific overlay node.

The coordinator can activate or deactivate several peers of the overlay network. Also, the coordinator is able to send several commands requiring the execution of specific probing related processes between specific overlay peers that are currently active. Such commands issued by the coordinator have multiple parameters. Due to space constraints, only some of the available commands allowed in the overlay operation are described below, being the corresponding function parameters further described in detail in Table 1.

PacketLoss *Number of Packets, Size, Timeout, Loop, Feedback(Feedback Time) or Statistic Options, Peer Y, Alarm(Alarm Condition)*

This command triggers the activation of packet loss measurements along the path connecting the peer receiving this command and *PeerY*. This measurement will be calculated considering the total amount of packets sent and the number of packets that effectively reached the destination.

Round Trip Time *Number of Packets, Size, ICMP, Timeout, Loop, Feedback(Feedback Time) or Statistic Options, Peer Y, Alarm(Alarm Condition)*

This command activates the measurement of the round trip time between the peer receiving this command and *PeerY*. The *ICMP* flag by default is activated (1), but the administrator can deactivate it by setting it to 0. When activated, this measurement will use an ICMP *echo_request* functionality to measure the round trip time. If the *ICMP* flag is deactivated (0), the peer will use a proprietary implementation of round trip time estimation made in Java using UDP packets.

Jitter *Number of Packets, Size, Loop, Timeout, ICMP, Feedback(Feedback Time) or Statistic Options, Peer Y, Alarm(Alarm Condition)*

This command measures the variation of the latency between successive data packets sent between the source peer and *PeerY*. If the *ICMP* flag is activated (1), which is 1 by default, it will be used an ICMP *echo_request* to obtain average values of the RTT variation. If the flag is deactivated (0) the one-way jitter will be measured with a proprietary implementation of the jitter time estimation made in Java using UDP packets.

RoutePath *Timeout, Loop, Feedback(Feedback Time) or Statistic Options, PeerY, Alarm(Alarm Condition)*

This command allows the coordinator node to obtain the complete route path between the peer receiving this command and *PeerY*. A system call is used to obtain the network path. This makes possible to verify when changes in specific ISP routing paths occur, and how much time such route changes persist, which may indicate a possible network anomaly.

Activate/Deactivate *PeerX*

The *activate* command activates a peer, making it start listening to various commands or return to the previous state of measurements if it had any before. Conversely, the *deactivate* command stops all peer functionalities and communications taking place with the coordinator node.

Cancel *Probe ID, Probe Type, Timeout, PeerY*

This command allows the coordinator node to cancel a measurement process that is running on a peer indefinitely. After receiving this command, and when the information is ready to be sent, the peer will send the corresponding measurements results to the overlay coordinator node.

All the commands generated by the overlay coordinator have a set of additional parameters, which assigned values may influence the way that the probing processes are made, and the way that the results are transmitted to the overlay coordinator. Table I summarizes some of the supported commands parameters and their general description.

Parameter	***Command Description***
Number of Packets	A numeric value indicating the number of data packets that should be generated by the peer receiving a specific command, the default value is 4.
Size	The packet size that should be used (only useful for some commands).
Loop	A value that indicates if the measurement will be run for an indefinite period or at an instant time, which by default is set at instant time *(0)*. If the measurement is set to run at indefinite time, the administrator may choose between: *Time Feedback (1)*, where the administrator sets the time between the peer responses to the coordinator; or *Statistical Feedback (2)*, which permits the administrator to choose a set of statistic options to be evaluated when the administrator cancels a measurement command being processed by the peers.
Timeout	A numeric value indicating when the peers should discard an issued command in the presence of problems, the default value is 5 seconds.
Probe ID + Probe Type	The coordinator identifies the command that he wants to cancel. Only possible in *Cancel* command.
Statistic Options	With this parameter the administrator defines the types of statistics he wants to analyse within a given measured metric (e.g. average, mean, maximum value, minimum value, mode, median). This option is only effective when using the *cancel* command on a given command that used the flag *Loop* (2), otherwise the result will be instantly committed to the coordinator
ICMP	This flag value by default is activated (1). With this flag the *ICMP* protocol is used on specific measurement commands. If the coordinator deactivates this flag, proprietary implementations to execute the commands without the ICMP protocol are then used.
Feedback	If the administrator wants to observe the probe results that are sent by peers, he can define the time interval that the probes are done and sent to the administrator in the *Feedback Time* parameter (in milliseconds). By default this option is activated. This is only possible with the *Time Feedback Loop* flag activated, otherwise the result will be instantly committed to the coordinator if the *Loop* is atomic, or the result will be committed only if the probing is cancelled if the *Loop* is *Statistical Feedback*
Alarm Condition	If the Alarm flag is activated *(1)*, the administrator needs to choose a logical condition (e.g. *bigger, lower, equal than* <value>) to trigger and alarm when a given measurement is taking place in the network and such condition occurs.

Table 1 – Parameters of the commands sent by the coordinator

B. Overlay Peer to Peer Communications

The communications processes between active peers in the overlay network are a consequence of the commands received from the overlay coordinator, requiring that specific measurements be made in the underlying ISP network.

Depending on the received commands, distinct UDP packets with distinct formats and contents are generated from the sending peers, which, in turn, trigger some response UDP packets from the receiving peers. Based on the exchanged packets, and corresponding contents, several measurement based data is stored in the peers for subsequently transmission to the overlay coordinator and to be stored in the database.

C. Packets Structure

There are several packet formats associated with the operation of the proposed P2P overlay system. Due to space constraints, this section only briefly refers to two types of packets exchanged between the overlay coordinator and the overlay peers entities: *i)* the control packets sent by the overlay coordinator to the overlay nodes, containing the probing commands that the coordinator intends to trigger in the peers and *ii)* the data packets sent by the overlay nodes to the coordinator, containing measurement related information that will be stored, analysed and presented to the ISP network administrator using appropriate interfaces. The packet format of the control messages sent by the overlay coordinator is presented in Figure 4. The packet has some fields allowing the identification of the packet and corresponding measurement process (*pid* and *pckid* packet fields), followed by an identification of the commands described in the previous section (the *type* packet field with values "A" *Activate*, "D" *Deactivate*, "P" *Packet Loss*, "R" *Round Trip Time*, "r" *Route Path*, "C" *Cancel*, "J" *Jitter*, "I" *Packet Injection*, etc.). After the identification of the peers that should participate in the probing process (*source* and *destination peer*), the following fields included in the packet format of Figure 4 are related with the function parameters presented in Table 1, which vary depending on the considered command issued by the overlay coordinator.

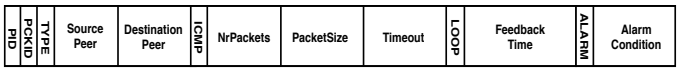

Figure 4. Control Communication packet format (sent by overlay coordinator to the overlay peer)

Similarly to the control communication packet format, Figure 5 shows the packet format associated with the data transfer processes between the overlay peers and the coordinator node. As visible, the monitoring results (or the statistical related data) are transmitted in the *results/statistical data* field, whereas the *alarm message* packet field carries the alarm related information that may be triggered during a probing process. Such alarm information will have a high processing priority, and should be presented to the ISP network administrator as soon as possible. As explained before, some of the information transmitted in these packets will be stored in an internal database at the coordinator node.

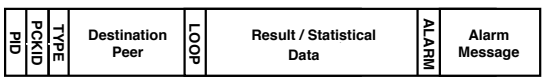

Figure 5. Data Transfer packet format (sent by the overlay peer to the overlay coordinator)

IV. System Implementation Technologies

In this section some implementation details will be explained, related to the developed system entities, the used graphical user interface and the database used to store probing information collected from the ISP network.

A. Overlay Entities and Communication Processes

All the entities of the proposed overlay system (Figures 1-3) were implemented in *Java*. The communication processes between such entities were implemented using TCP/UDP *Java* sockets, over which all the mentioned control messages, data transfer packets and general probing related processes take place. Moreover, in order to enrich the ISP administrator user experience and improve the overall system performance, specific graphical and database technologies were used, being discussed in the following sections.

B. System Graphical User Interface

A graphical user interface for the ISP administrator was developed, using the GUI library *JavaFX* [5] that helps the administrator to monitor the network and setup the overlay P2P network using an intuitive interface. Figure 6 presents the interface provided to the network administrator. As observed, the interface integrates a view of the network topology, over which the overlay network operates, and the links and routers which compose it. At the bottom of the interface two text display areas are available for the probing reports, one for results and another for the generated alarm messages. To check the results of the measurements in real time, the administrator may choose some filters to reduce the information being displayed and target the specific probing that he wants. A text area on the right side of the interface displays all the current measurements being made.

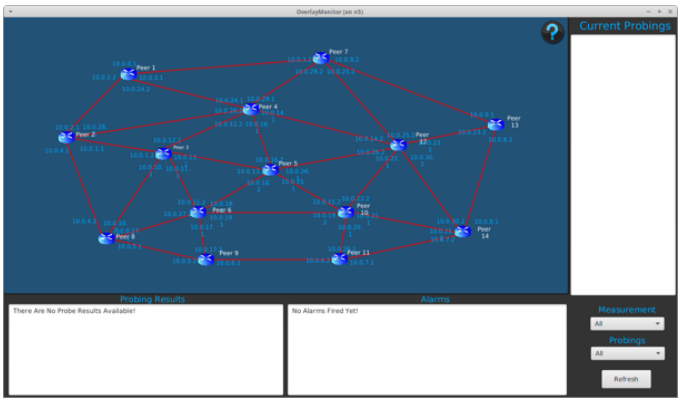

Figure 6. The interface developed for the P2P overlay monitoring system

The interface of Figure 6 allows the administrator to mouse over the links to see the respective probing processes that they are being subject to. The interface also highlights all the links associated with those probes with a bloom effect to help to visualize the probing path. Each link can have multiple colours to help the administrator to associate the probes with their type. The information about all the possible colours and some other helps for the administrator is displayed when the *'?'* button is pressed. Using the interface depicted in Figure 6,

to start a probing process the administrator needs to left click a peer and then select the destination interface, specifying afterwards other information (e.g. the type of measurement, alarms, type of loop, ICMP flag, etc.) as observed in Figure 7. The administrator also has the ability to cancel the probing, by left clicking a peer, choosing *cancel* and selecting the respective probe from the current running probes.

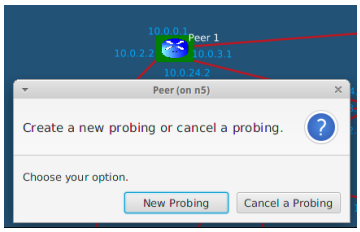

Figure 7. Probing creation interface

The developed interface also allows the administrator to check the probing history of the ISP network, as depicted in Figure 8. Using such interface, the administrator can filter the results by probing type, peer destination, data that the probe was created, display only results or alarms that were triggered, among many other options.

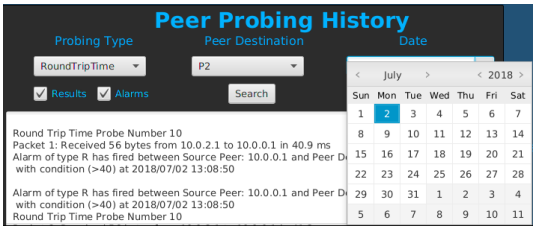

Figure 8. Peer probing history interface

C. *Graph Database of the P2P Overlay System*

The developed monitoring system can be used in ISP networks with a large number of routers and links. Moreover, it is also possible to configure a large number of probing processes, collecting a huge amount of related data that is stored at the overlay coordinator. Thus, the database technology should be carefully selected.

In this context, the *Neo4j*, a graph database management system [6], was used because it allows highly performing read and write scalable operations, also allowing performing reliably fast transactions with ultra-high parallelized throughput even as if the data grows. It also uses a very powerful and productive graph query language, *Cypher*. In Neo4j, everything is stored in the form of an edge, node, or attribute. Each node and edge can have any number of attributes. Additionally, nodes and edges can be labeled, allowing to narrow searches. A very simple example is provided in Figure 9, showing the node and edge visualization of the storage of two probing processes (*probing 1* and *probing 2*) made from *peer1* to the destination *peer2*.

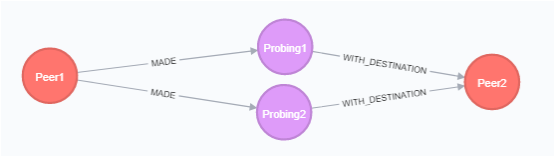

Figure 9. Neo4J probing example storage

V. Illustrative Results of the System Operation

The test of the developed was made using the *CORE* network emulator [7], where is possible to create ISP network topologies and run applications. Using this emulator real applications can run in the network nodes and hosts, which are Linux based systems. In this case, after creating an illustrative ISP network, the implemented overlay coordinator and several overlay peers were setup in the topology and the interface with the administrator was activated. Figure 10 (left side) shows an illustrative ISP network topology with routers and links created in the CORE emulator, and Figure 10 (right side) depicts the previously explained interface where the ISP administrator visualizes the ISP network and is able to command and interact with the P2P overlay monitor network.

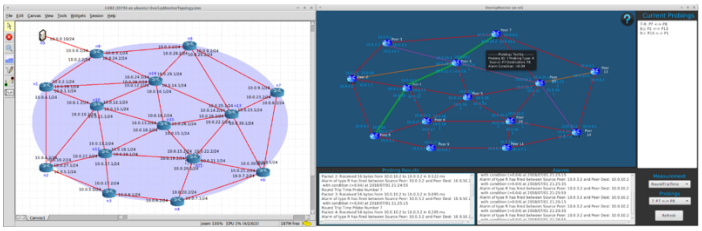

Figure 10. Network topology emulated in CORE (left side) and the graphical user interface of the developed P2P overlay system (right side)

In the following subsections, three different test cases are presented, using distinct *loop* flags and distinct measurement probes types (*round trip time, jitter and route path*). That commands were activated by the administrator using the mentioned application interface (as in Figure 11).

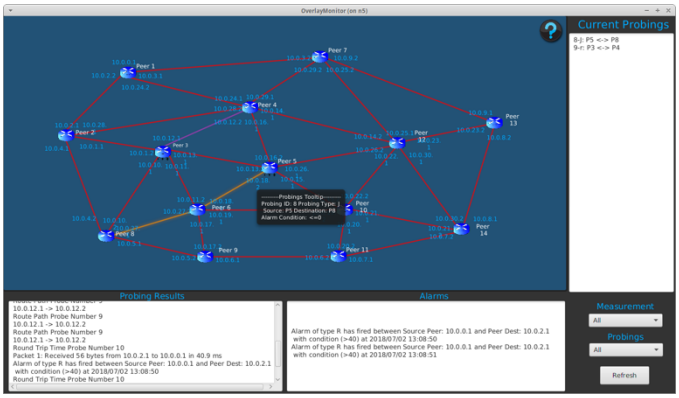

Figure 11. Activation of all the test cases in the administrator interface

A. *Test Case 1: Atomic Probing*

This test case shows an atomic *round trip time* probing activated by the administrator between the *peer1* and *peer2* and with an alarm condition of *">40" ms*. In Figure 12

it can be observed the output displayed in the interface text display, denoting the measured round trip time, and the triggered alarm when the RTT surpasses 40 milliseconds.

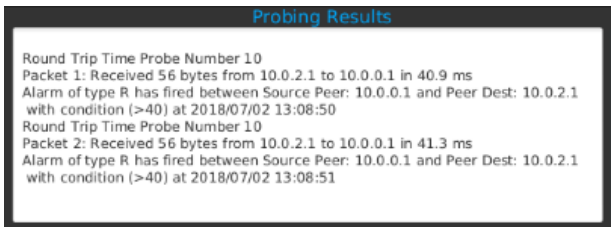

Figure 12. Atomic probing of RTT between *peer1* and *peer2*

B. *Test Case 2: Feedback Loop Probing*

The second test case shows the case of a loop probe with constant feedback to the administrator (*feedback time of* 5sec), and with the measurement type of *route path*. This probing was setup in the link between *peer3* and *peer4*, and with the alarm configured to be triggered when a route path change occurs. To force this scenario, in the *CORE* emulator the link between the peers will be shutdown forcing a new route between the peers. Figure 13 clearly shows that the route path change has been detected by the triggered alarms, indicating the change of the path generated by the link that was shutdown in the network topology.

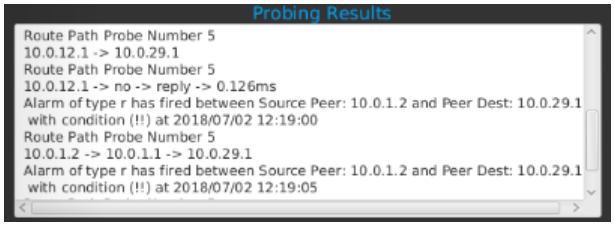

Figure 13. Feedback loop probing of route path between *peer3* and *peer4*

C. *Test Case 3: Statistical Loop Probing*

In this third test case it is presented the statistical loop probing with the *jitter* metric. As mentioned, this type of loop probing does not provide constant results to the administrator and if the administrator wants to actually see the statistics of the probing being made, he needs to finish the probing process with a *cancel* command. This probing runs between *peer5* and *peer8* and the alarm was set to be triggered with the condition *"<=0"*, which in this case means when a decrease in the delay is detected. As in the previous examples, Figure 14 shows the generated outputs for this test case.

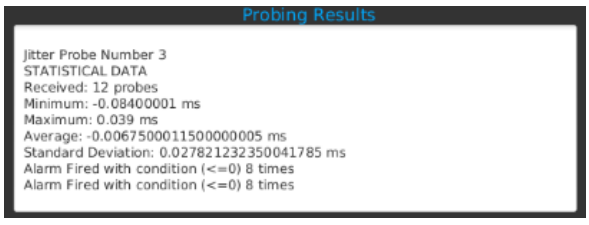

Figure 14. Statistical loop probing of jitter between *peer5* and *peer6*

VI. CONCLUSIONS

This article presented an overlay P2P system for network anomaly detection, being able to assist ISP network administrators. The developed system provides the administrator with different types of probing and alarms able to be triggered in real-time, for monitoring, congestion control, fault detection, or other network related purposes. After presenting the overall architecture and entities of the system, some details were given regarding the communication processes, commands and packets associated with the internal operation of the devised system. The development technologies were also focused, mentioning the implementation language, the database and the developed interface to interact with the ISP administrator. Finally, some illustrative system use cases were also presented.

As possible future work topics, more types of probing processes could be added to the prototype. Also, it is also interesting to study the expansion and test of the developed system for scenarios involving multi-ISP environments where several administrators may cooperate in the network management operations and anomaly detection efforts. The interface can be also enriched with a graphical visualization of the measured metrics and also an improved highlight of the alarms triggered during the overlay operation.

ACKNOWLEDGMENT

This work has been supported by FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UID/CEC/00319/2019.

REFERENCES

[1] Stone, R. (2000). CenterTrack: An IP Overlay Network for Tracking DoS Floods. Proc. 9th Conference on USENIX Security Symposium Vol. 9.

[2] Jannotti, J., et al. (2000). Overcast: Reliable Multicasting with an Overlay Network. 4th conference on Symposium on Operating System Design & Implementation - Volume 4. pp. 197-212.

[3] Baumgart, I., Heep, B., & Krause, S. (2007). OverSim: A Flexible Overlay Network Simulation Framework. *2007 IEEE Global Internet Symposium*, pp. 79-84.

[4] Lua, E.K., Crowcroft, J.A., Pias, M., Sharma, R., & Lim, S. (2005). A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *IEEE Communications Surveys & Tutorials, 7*, 72-93.

[5] JavaFX, https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

[6] Neo4J , https://neo4j.com/

[7] CORE Emulator, https://www.nrl.navy.mil/itd/ncs/products/core

[8] Sampaio, A., Sousa, P. (2018). An adaptable and ISP-friendly multicast overlay network. Peer-to-Peer Networking and Applications, Springer. https://doi.org/10.1007/s12083-018-0680-y

[9] RIPE Atlas. https://atlas.ripe.net/

[10] perfSONAR. https://www.perfsonar.net/

[11] Sam Knows. https://www.samknows.com/

[12] NLNOG RING. https://ring.nlnog.net/