Decentralized Detection of SLA Violations Using P2P Technology

Jéferson C. Nobre, Lisandro Z. Granville
Institute of Informatics
Federal University of Rio Grande do Sul - Brazil
Email: {jcnobre, granville}@inf.ufrgs.br

Alexander Clemm, Alberto Gonzalez Prieto
Cisco Systems
San Jose, USA
Email: {alex, albertgo}@cisco.com

***Abstract*—Critical networked services enable significant revenue for network operators and, in turn, are regulated by Service Level Agreements (SLAs). In order to ensure SLAs are being met, service levels need to be monitored. One technique for this involves active measurements, such as IPSLA. However, active measurements are expensive in terms of CPU consumption on network devices. As a result, active measurements usually can cover only a fraction of what could be measured, which can lead to SLA violations being missed. The definition of which subsets of service paths to measure and to configure corresponding measurement probes is a practice that does not scale well and results in fairly static measurement setups that do not adapt well to shifting networking patterns. We propose a solution to increase the detection rate of SLA violations in which devices in a network autonomously and dynamically determine how to place probes in order to detect service level violations. It does not require human intervention, is adaptive to changes in network conditions, resilient to networking faults, and independent of the underlying active measurement technology. Our solution is based on peer-to-peer principles and is characterized by a high degree of decentralized decision making across a network using a self-organizing overlay. In these experiments it is possible to observe that an increase in the information used in probe placement decisions decreases the number of SLA violations missed.**

I. Introduction

Critical networked services are expected to operate respecting associated Service Level Agreements (SLAs), established between service provider and customer. To ensure that SLAs are not being violated, which would usually incur in costly penalties, service levels need to be constantly monitored at the network infrastructure layer. To that end, either active or passive network measurements must take place.

In passive measurement, network conditions are said to be checked in a non intrusive way because no monitoring traffic is created by the measurement process itself. Passive measurement is realized, for example, inside network devices when they observe the passing traffic flows. Active measurement, on the other hand, is intrusive because it injects synthetic traffic into the network to measure the network performance. Probes distributed along the network are the elements that, in active measurement solutions, inject such a synthetic traffic and compute the network performance.

Active measurement mechanisms (e.g., Cisco's Service Level Assurance Protocol [2], and IETF's One-Way Active Measurement Protocol (OWAMP) [10] and Two-Way Active Measurement Protocol (TWAMP) [6]) usually offer better accuracy and privacy than passive measurements. As a result, active is preferred over passive measurement in several scenarios. However, better accuracy and privacy come at a price: they turn active measurement solutions more expensive in terms of resource consumption (e.g., CPU cycle and memory footprint required by measurement probes) inside network devices, in addition of increasing the network load because of the injected traffic. Furthermore, because the increasing size and complexity of modern networks lead to a combinatorial explosion of possible network paths, monitoring all network flows would requires a too large and costly prohibitive number of probes.

In practice, the common approach in feasible deployments of active measurement solutions consists in having the network operator activating only a subset of all available probes, thus enabling the observation of just a subset of all network flows, i.e., the set of all network flows is never covered entirely. This approach, however, does not scale well because it is still too difficult and labor intensive for the network operator to compute which probes should be activated given the set of critical flows that needs to be measured. Even worse, this practice completely fails in networks whose critical flows are too short in time and dynamic in terms of traversing network paths, like in modern cloud environments. That is so because human operators are not just enough in computing and activating the new set of probes required every time the network traffic pattern changes. In this context, in addition of being labor intensive, the current active measurements practice usually covers only a fraction of the network flows that should be observed, which invariably leads to the damaging consequence of undetected SLA violations.

In this paper, we present an autonomic peer-to-peer (P2P) active measurement solution that increases the number of detected network SLA violations. Our solution is adaptive to changes in network conditions, resilient to networking faults, independent of the underlying active measurement technology, and requires no human intervention. Through the employment

Jéferson C. Nobre is a PhD student at the Federal University of Rio Grande do Sul, Brazil. He conducted this work during a partial doctoral fellowship at Cisco Systems, funded by CAPES Foundation (Ministry of Education of Brazil).

978-3-901882-48-7

of a self-organizing, in-network P2P measurement overlay, our solution presents a high degree of decentralization in the decision making process of determining which measurement probes must be activated/deactivated to cope with the network dynamics. Our solution has been evaluated using an event-based P2P simulator. In our experiments, we deployed our peer code in different topological settings, varying from synthetic HOT-like topologies [7] to topologies obtained from the Rocketfuel project [11]. Results show that our solution performs substantially better than the random placement, increasing the the number of detected network SLA violations as well as respecting resource constraints.

The remaining of this paper is organized as follows. In Section II we review the background on active measurement and P2P technology for network management. In Section III, our proposed solution is introduced and its associated concepts are described. The experimental evaluation, encompassing simulation setups, is presented in Section IV. Related work is discussed in Section V. Finally, we close this paper in Section VI, where concluding remarks and future directions are presented.

II. Background

In this section we first cover some of the most prominent active measurement mechanisms and their main concepts. After that, some properties found on the employment of P2P technology in network management are discussed.

A. Active measurement mechanisms

Active measurement mechanisms are an important tool to monitor Service Level Objectives (SLO) and the health of a network as a whole. These mechanisms can be employed in different contexts, such as pre-deployment validation or measurement of in-band live network performance characteristics [2]. Active measurement mechanisms inject synthetic traffic into specific network paths to measure the network performance which can lead to either one-way or two-way measurements. The generation of synthetic traffic and its computation to provide measurements is usually performed by an architecture comprised of two hosts with specific roles, a sender and a responder. These hosts are also collectively known as measurement probes. There are several protocols used to enable active measurement, however, for the sake of simplicity, we will focus active in the proposals from IETF and Cisco Systems.

The Internet Engineering Task Force (IETF) IP Performance Metrics (IPPM) Working Group has proposed open mechanisms that permit the exchange of packets to collect metrics for one-way packet delay and loss across Internet paths in an interoperable manner. The One-way Active Measurement Protocol (OWAMP) [10] is aimed at the control and collection of one-way measurements. OWAMP consists of two inter-related protocols: OWAMP-Control (used to send single measurement packets along the Internet path under test) and OWAMP-Test.(used to initiate and control measurement sessions and to fetch their result). In order to accommodate two-way measurements, the Two-Way Active Measurement Protocol (TWAMP) [6] was proposed. TWAMP uses the methodology and architecture of OWAMP for measurement of two-way metrics in addition to the one-way metrics of OWAMP. From the industry side, Cisco Service Level Assurance (SLA) protocol [2] is a widely deployed protocol used to measure service level parameters. Cisco SLA protocol measures service levels in L2 and L3 networks and applications running on top of L3, both considering one-way and two-way metrics. Besides delay and loss, Cisco SLA protocol provides support to a greater variety of statistics including network jitter and packet/frame loss.

Active measurement mechanisms have good characteristics in terms of accuracy, since they can simulate actual protocol exchanges; and privacy, because they are hard to detect and it is difficult to make inferences about measurements by intermediaries in the middle of the network [10]. However, these mechanisms are expensive in terms of computational resources consumed on network devices and the network bandwidth required to carry synthetic traffic. The reason for this is the inherent cost related to the generation and forwarding of synthetic test packets and the further analysis of those packets and their responses. This cost explodes with the size and complexity of current network infrastructures, which avoids monitoring all network flows. In some settings even dedicated routers (also known as "shadow routers") are deployed to handle active measurement mechanisms and save resources for primary network functions (e.g., routing and switching).

B. The employment of P2P technology in network management

The use of P2P technology in network management, also known as P2P-based Network Management (P2PBNM), extends Distributed Network Management (DNM) by merging characteristics of DNM and P2P overlays. In P2PBNM, peers have to perform management tasks and their related provisioning details (e.g., overlay organization). From the user perspective, however, the overlay provisioning details must be transparent, requiring no knowledge about the implementation or architectural organization of nodes in the overlay topology.

Granville et al. [5] describe P2PBNM as an extension of the management by delegation (MbD) model. In MbD, managers delegate the execution of tasks to mid-level managers (MLMs) located closer to agents (e.g., transferring management scripts), which reduces network bandwidth consumption and decentralizes the execution of management tasks. P2PBNM merges the MbD model and the services introduced by P2P networks (e.g., robust connectivity).

P2PBNM systems usually present a high degree of decentralization concerning the execution of management tasks. Thus, the bulk of the computation, bandwidth, and storage needed to operate P2PBNM systems is contributed by the management peers. This decentralization also pushes the peers' local autonomy, increasing the use of local data and logic to make management decisions [4]. Furthermore, the decentralization of P2PBNM systems makes their growth more "organic" since they can grow without requiring a fork-lift up-

grade, i.e., they can grow just adding more peers. For example, decentralized techniques have been successfully applied to the detection of threshold crossings. Wuhib et al. [12] show that such techniques can enable the design of efficient solutions that scale with the size of the network.

III. Proposed Solution

Active measurement mechanisms are an effective technique for monitoring Service Level Objectives (SLOs). However, these mechanisms are expensive in terms of resource consumption, both on network devices and bandwidth. Concerning the utilization of active measurement mechanisms to detect SLA violation, there is an inherent trade-off between attempting to maximize SLA coverage over end-to-end paths and minimizing the resource consumption due to the deployment of active measurement probes. Intuitively, two extreme strategies can be described: maximum coverage, increasing the number of deployed probes without considering resource consumption and possibly leading to network devices starvation, and minimum resource consumption, decreasing network coverage and, probably, missing SLA violations. In order to balance between these strategies, we devise the utilization of past service level measurement results and resource constraints to steer autonomically probe placement decisions, i.e., to decide which paths must be probed observing resource constraints.

Network devices in our solution determine a subset of end-to-end paths in which probes must be deployed. Our solution attempts to build a subset that maximizes the fraction of detected SLA violations. In order to build this subset, paths are ranked by network devices what leads to a distributed path rank. This rank is determined by taking into account a variety of information from the network itself to steer probe placement decisions. This information can be generated locally (e.g., local historical measurement data) or received from other network devices. After that, this rank and resource constraints are used to select a subset of end-to-end paths. Resources constraints are used in our solution to assure that local and global upper bounds for the number of deployed probes are respected. Once this subset has been selected, probes are deploy along these paths over a given time interval.

Our proposed solution has some assumptions as follows. We consider an underlying probe infrastructure that permits end-to-end SLA monitoring. Probes in this infrastructure are able of both initiate and respond synthetic traffic. In this infrastructure, deviations from SLOs (i.e., detection of SLA violations) can be identified by single network devices using the chosen active measurement mechanism. Thus, the solution aims at increasing the efficiency of the detection of SLA violations solely through probe placement decisions. In this context, the chosen active measurement mechanism does not need any modification to be deployed according these decisions. In principle, knowledge of the routing information as well as of the network topology is not necessary to rank paths. Furthermore, security issues are considered orthogonal to the present proposed solution.

The two main concepts of our proposed solution, the utilization of past service level measurement results and resource constraints and the path rank and its heuristic algorithm, are discussed in the following subsections.

A. The utilization of past service level measurement results and resource constraints

The utilization of past service level measurement results is our approach to develop a method for establishing if a path is likely to disrespect SLOs. In order to establish that, we use the average of past service level measurement results for a given path considering a sliding window. If the past measurements results for a given path are close to a SLO, then the probability of deploying a probe in this path is increased. Besides that, even if a path is not close to the SLOs, it is important that it be measured frequently. In order to induce frequent probing on all paths, we also use the time elapsed from the last measurement for a given path to increase the probability of this path to be probed. Clearly, if a path had not been measured recently, then it should be more likely to be selected in the next interactions. The joint use of the past service level measurement results and the time elapsed from the last measurement for a given path enables the network devices to determine how to place probes in an autonomic manner.

Past service level measurement results can be locally collect or received by other network devices. However, it is necessary to assure that the received results have local relevancy. For example, if the network paths used by two endpoints, node *a* and node *b*, to reach a third endpoint, node *c*, are completely disjoint, the contribution due to network transmission for measurement results would be probably different. In this context, the use of measurement results exchanged by the node *a* and node *b* concerning node *c* could lead to undesirable results (i.e., decrease the number of detected SLA violations). In order to avoid that, we use the concept of *correlated peers*. Two nodes are considered as correlated peers if the results of their measurements for a given destination are correlated. Since this concept determines with which nodes exchange information, it helps decreasing the amount of traffic generated and the node resources needed to process measurement results exchange.

The resource constraints are defined according to the maximum number of probes expected to be deployed in a given time. This number is used as an abstraction for the resources available for active measurement mechanisms. The rationale behind this strategy is that nodes should add probes when the load allows since an increase in the number of probed paths should also increase the number of detected SLA violations. The maximum number of deployed probes must be enforced locally (i.e., in a specific node) and globally (i.e., concerning nodes that deploy our solution). This leads to two resource constraints, α and β. The constraint α is the global upper bound for deployed probes and β is the local upper bound for deployed probes. These resource constraints may vary over time. Nodes can easily ensure that β is respected since this can be done just checking the local number of deployed probes. However, as α considers information from different nodes,

now the amount of remote information that is exchanged by nodes influences how α is computed. For example, using only the number of nodes exchanging information and considering a homogeneous distribution, the local contribution to α can be infer to be at most α divided by the number of nodes exchanging information.

B. Path ranks and probe placement algorithms

We now present how the path ranks are composed and probe placement algorithms are used to determine which paths to probe. In principle, as more information from the network is used, the probe placement decisions capture better the service level violations. However, the network conditions may change over time and probe placement decisions must cope with these changes. Hence some mechanism is needed to dynamically adapt the probe placement decisions to network conditions. We use an approach which is opposite to heavyweight optimization. In fact, we employ a computationally simple algorithm that tries to capture the common sense used by network administrators. This common sense is abstracted through path weights which in turn compose path ranks in a per node basis.

Path weights are composed by information from past service level measurement results (previously explained). As our algorithm proceeds, path weights are calculated for each path and they are normalized using the sum of the same kind of weights for the possible paths. Thus, a specific (normalized) weight of a path varies between 0 (minimum) and 1 (maximum). The path rank is composed by the total weight of paths, i.e., each path in the set of paths is ranked according to the sum of the several weights that comprise it. After that, we start by sorting the paths according to their total weight, and then probe on the top k paths with the largest weights. In other words, at each time interval we greedily choose the k highest weighted paths for deploying probes. The k is the minimum between β and the local inferred α (observing the number of available paths to probe). Once a set of paths is chosen, we deploy probes in those paths as an effort to maximize the number of detected SLA violations.

The path rank (and the probe placement algorithms) can be computed using either only information available locally or information exchange by correlated peers. The main difference is the set of weights used during the definition of the path rank. Initially, we define 2 possible approaches to use information to define the path rank, and, therefore, to choose which paths will be probed: probe placement based on local information and probe placement based on local and remote information. We also define a random probe placement in order to provide a baseline. Now we describe the different probe placement algorithms.

Random probe placement. Probe placement is done through a random placement over the possible end-to-end paths. Random probe placement is shown on Algorithm 1. Thus, if there are k possible paths to choose from, they would be chosen with the same probability $(1/k)$. However, even in the simplest algorithm, the nodes also control the resource consumption using the α and β constraints. The algorithm shuffles the list of possible paths ($path[]$) and then uses as the number of paths to be probed the minimum among the number of possible paths ($sizeOf(path[])$), the locally inferred global bound ($\alpha/sizeOf(path[])$), and β.

Algorithm 1 randomDecision($\alpha, \beta, path[]$)

$M \leftarrow min((\beta, \alpha/sizeOf(path[]), sizeOf(path[]))$
shuffle($path[]$)
for $i = 1 \rightarrow M$ **do**
  deployProbe($path[i]$)
  $i \leftarrow i + 1$
end for

Probe placement based on local information. Probe placement is performed using local information about past service level measurement results. Probe placement based on local information is shown on Algorithm 2. Now the probability of the path to be chosen is related with the past measurement results ($rankPast[]$) in a sliding window ($windowSize$) and the time elapsed from the last measurement ($rankLast[]$) for a given path (t). It is possible to "tune" the contribution of $rankPast[]$ and $rankLast[]$ using the constants A and B, respectively. The nodes also control the resource consumption using the α and β constraints. The algorithm shuffles the list of possible paths ($path[]$) and updates $rankPast[]$ and $rankLast[]$ for these paths. This information is used to define the current weights for each path, and after that, paths are sorted in descending order. The algorithm then uses as the number of paths to be probed (M) the minimum among the number of possible paths ($sizeOf(path[])$), the locallly inferred global bound ($\alpha/sizeOf(path[])$), and β. Then, we greedily choose the top M paths.

Algorithm 2 localDecision($\alpha, \beta, A, B, windowSize, path[]$)

$M \leftarrow min((\beta, \alpha/sizeOf(path[]), sizeOf(path[]))$
shuffle($path[]$)
for $t = 1 \rightarrow sizeOf(path[])$ **do**
  $rankLast[t] \leftarrow getLastLocal(path[t])$
  $rankPast[t] \leftarrow getPastLocal(path[t], windowSize)$
  $t \leftarrow t + 1$
end for
sortDesc($path[], key \leftarrow A * (rankLast[]/\Sigma rankLast[]) + B * (rankPast[]/\Sigma rankPast[])$)
for $i = 1 \rightarrow M$ **do**
  deployProbe($path[i]$)
  $i \leftarrow i + 1$
end for

Probe placement based on local and remote information. Probe placement is performed using information about past service level measurement results, both locally collected and obtained from correlated peers. Probe placement based on local and remote information is shown on Algorithm 3. Now, in each interaction, there are two distinguished phases: peer topology phase and probe placement. The probe placement

phase resembles the previous described algorithm (probe placement based on local information), however, the probe rank now is also populated using information from correlated peers. Thus, the current path weight is composed of one more item, the summary of received remote measurement results ($rankRemote[]$) for a given path (t). It is also possible to "tune" the contribution of $rankRemote[]$ using the constant C. The resource consumption is controlled in a similar fashion using the α and β constraints.

Algorithm 3 localRemoteDecision($\alpha, \beta, A, B, C, confidence, windowSize, path[], correlatedPeers[]$)

$M \leftarrow min((\beta, \alpha/sizeOf(path[]), sizeOf(path[]))$
if $correlatedPeers[] == null$ **then**
  $correlatedPeers[] \leftarrow getEndpoints(path[])$
else
  $correlatedPeers[] \leftarrow getCorrelatedPeers(path[])$
end if
sendMeasurementSummary($correlatedPeers[]$)
sendCorrelatedPeers($correlatedPeers[]$)
$M \leftarrow min((\beta, \alpha/sizeOf(path[]), sizeOf(path[]))$
shuffle($path[]$)
for $t = 1 \rightarrow sizeOf(path[])$ **do**
  $rankLast[t] \leftarrow getLastLocal(path[t])$
  $rankPast[t] \leftarrow getPastLocal(path[t], windowSize)$
  $rankRemote[t] \leftarrow getPastRemote(path[t])$
  $t \leftarrow t + 1$
end for
sortDesc($path[], key \leftarrow A * (rankLastLocal[]/\Sigma rankLastLocal[]) + B * (rankPastLocal[]/\Sigma rankPastLocal[]) + C * (rankPastRemote[]/\Sigma rankPastRemote[])$
for $i = 1 \rightarrow M$ **do**
  deployProbe($path[i]$)
  $i \leftarrow i + 1$
end for

The peer topology phase uses the concept of correlated peers. Two peers are considered as correlated if their measurements for a given destination are correlated. The peer topology phase proceeds as follows. To bootstrap peer selection, each peer uses their known endpoints neighbors as the initial seed to get candidate peers, i.e., peers that may be evaluated for correlation purposes. Then, peers send information about their measurements (collected using a sliding window) for their candidate peers. Each peer compares this information with their own measurements in order to extract a subset of candidate peers which in turn become correlated peers. Candidates are then ranked by their comparison scores and those which have higher scores are then chosen. Eventually, peers also spread their correlated peers in order to permit evaluation of "peers of peers". The number of correlated peers of a peer in a given time is controlled by a upper bound constraint.

There are different ways to compare the local measurements and measurement received by other peers to define correlated peers. Initially, we chose to perform this comparison using a *Student's t-test*. Since this test can be done using only summary information (average, variance and the number of samples), the information that needs to be exchanged by peers is substantially decreased. The confidence of Student's t-test is used twofold: it must be provided as a system parameter for the lower bound to consider peers as correlated ones and it is also used by peers to rank their candidate peers. Probe placement phase is taken place after the peer topology phase.

TABLE I
SUMMARY OF SELECTED TOPOLOGIES USED IN THE EXPERIMENTS

Topology	Interior nodes	Leaf nodes	Total nodes	Interior/Leaf nodes ratio
Hot-like A	8	19	27	0.42
Hot-like B	17	34	51	0.5
Rocketfuel-derived A	21	19	40	1.11

IV. EVALUATION

We studied the performance of our proposed solution by defining and implementing simulation experiments. These experiments were implemented in Java using PeerSim [8], an open source event-based simulator of P2P systems. The simulator provides the basic node communication infrastructure as well as transport layer models, which can emulate some characteristics of IP networks (e.g., packet loss and delay). We implemented the probe placement decision algorithms as well as a simple active measurement mechanism. This mechanism shares the basic features found on commercial implementations, such as IPSLA[1]. Furthermore, we introduced some changes in the transport layer models of PeerSim in order to allow explicitly modifications in network conditions in a per link basis. Thus it is possible to have an almost complete control of the simulation environment.

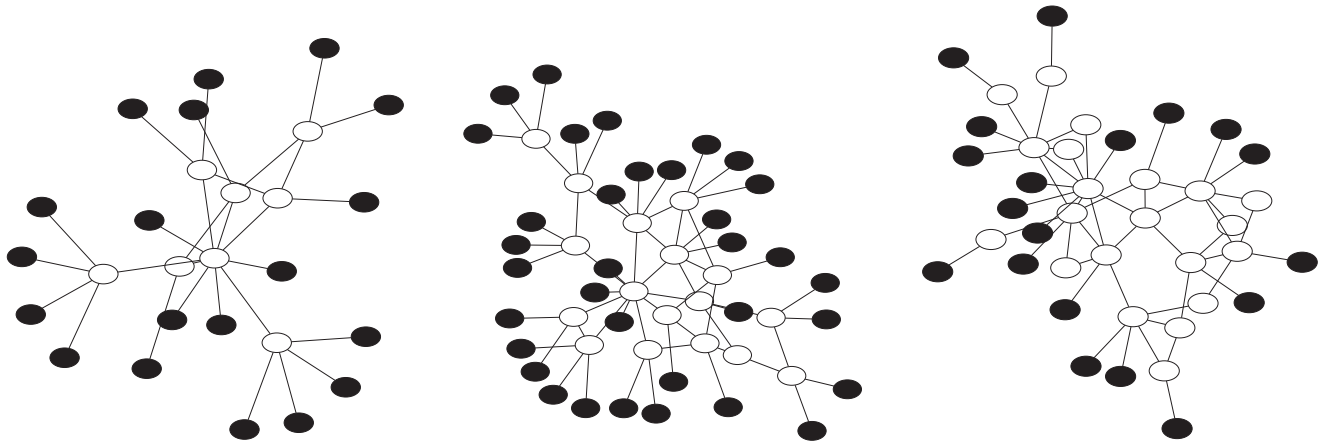
The focus of the simulation experiments is to evaluate the detection rate of SLA violations as well as the properties of the probe placement decision algorithms and the path rank algorithm per se. We explicitly do not focus on the accuracy characteristics of the measurement mechanism itself, since we believe our approach can be used with different active measurement mechanisms. For this reason, we did not experiment with either a variety of network conditions or network metrics. In the next subsection we describe the experimental setup used in the simulation experiments and the obtained results.

A. Experimental Setup

The simulation experiments are performed in increasingly complex topological settings. These topologies are both synthetic ones and inferred from real network environments. The synthetic HOT-link topologies were created using the Orbis topology generator [7]. On th other hand, the inferred topology is obtained using available data from the Rocketfuel project [11]. The selected topologies are shown in Figure 1 and some characteristics of these topologies are presented in Table I.

The topologies vary significantly regarding the number of interior and leaf nodes as can be noticed. We consider that

[1]Cisco IOS IP Service Level Agreements - http://www.cisco.com/go/ipsla



(a) Hot-like A topology (b) Hot-like B topology (c) Rocket-derived A topology

Fig. 1. Selected topologies used for simulation experiments.

all leaf nodes can deploy active measurement probes (both as senders and responders) as well as these probes always reply to measurement requests. The assumption of probes being located on leaf nodes is related to the investigation focus: detection of *end-to-end* SLA violations. These assumptions also holds considering the common practices on field deployments.

In the experiments, the end-to-end paths are calculated using shortest paths, considering only the number of hops. All end-to-end paths are considered as candidates to be probed in each simulation cycle. For simplicity, we used a network-wide SLO for detecting SLA violations. However, it is possible to operate with multiple SLAs as well as considering SLOs in a per node basis. All network modifications are injected in one direction for a given link and are defined in terms of simulation cycles. Besides that, all nodes run exactly the same probe placement decision algorithm in a given experiment. We include results for random probe placement decision as a baseline.

In the experiments, we used series of 10 simulation experiments started using different random seeds. The graphics present the mean values; the observed variance in the experiments was low.

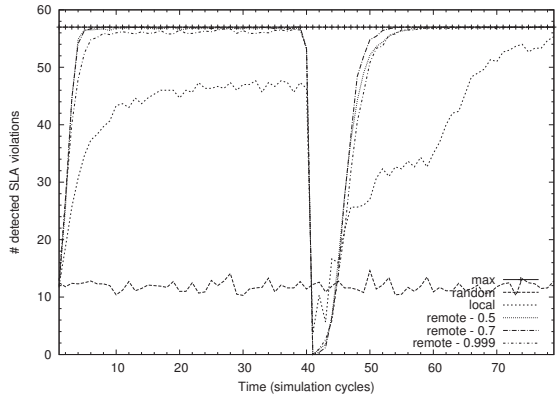
B. Results

In the first experiment (β = 3), we aim at determining the adaptation features of the proposed probe placement decisions approaches. In order to accomplish that, we collected the total number of SLA violations detected by nodes regarding a specific network environment setup (for the mentioned topologies). In this setup, we increased the one-way delay on 4 access links for 40 cycles, and then we changed for other 4 links for the same amount of cycles. This increase makes the end-to-end paths that traverse the changed links to appear as SLA violators for the simulated active measurement mechanism. We chose the number of cycles in which the experimental setup is changed in order to permit that the proposed approaches go through their permanent response. As we primarily focus upon the local resource constraint, β, we do not consider α for this experiment.

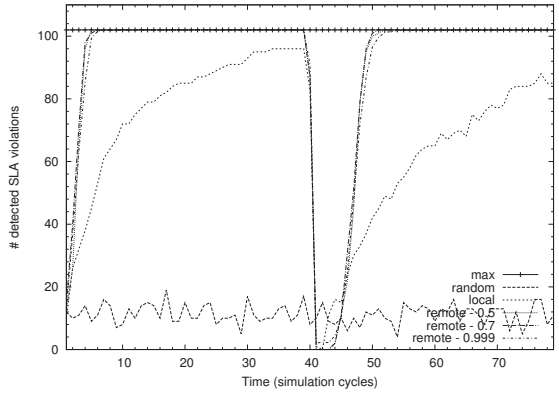
The results for the first experiment are shown on Figure 2, considering the aforementioned topologies. The curves depicted on this Figure represent the mean number of detected SLA violations as a function of simulation cycles regarding the next probe placements strategies: the use of local information ("local" on Figure 2) and the use of local and remote information using the following values of confidence on the *Student's t-test*: 0.5, 0.7, and 0.999 ("remote - 0.5", "remote - 0.7", and "remote - 0.999" on Figure 2). Results for the random placement ("random" on Figure 2 are depicted as a baseline. Besides the curves for the proposed approaches, we also present the maximum number of SLA violations that can be detected ("max" on Figure 2), i.e., every probe deployed detects a SLA violation, considering the number of probes that can be deployed by each node and the total number of nodes that can deploy probes (leaf nodes).

The experiment shows that the proposed approaches behave as we expected, without stability and convergence problems. As can be seen in Figure 2, the utilization of both local and remote information on probe placement decision increases significantly the number of detected SLA violations in the experimental setup. Clearly, even the unique utilization of local information (which can be view also as a baseline for the P2P strategy) has a better performance than the random placement. Moreover, we approach the maximum number of detected SLA violation for the resource constraints when using both local node and remote information.

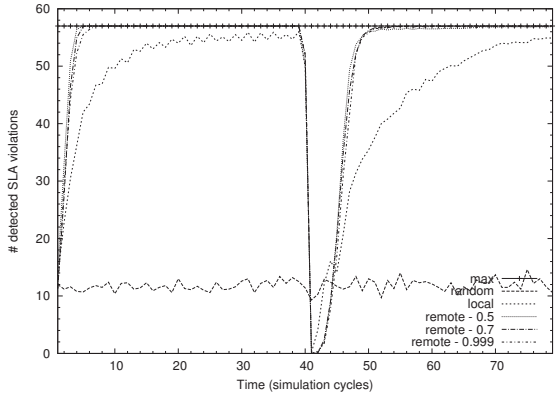
In the second experiment, we evaluated the number of detected SLA violation as a function of the number of deployed probes (β), considering the proposed probe placement decisions approaches. In this experiment, we used the same delay function from the previous experiment (one-way delay increased on 4 access links for 40 cycles, then change for other 4 access links for 40 cycles), however the number of detected



(a) Hot A



(b) Hot B



(c) Rocket A

Fig. 2. Number of detected SLA Violations.

SLA violations is now consolidated for each experiment. We show on Figure 3 mean results for the Rocket A topology. The β values were chosen to permit that the approaches be evaluated according to different available resources, from 1 to 18 (the number of possible end-to-end paths in this topology).

The experiment shows how efficiently the proposed probe placement decisions approaches use the available resources. As can be seen from the results in Figure 3, more available resource (higher β values) evidently lead to a higher number of detected SLA violations for all approaches. However, the proposed probe placement decisions approaches perform significantly better than the random placement for most values of the β range. In fact, excluding the situation where the available resource are sufficient to probe all possible end-to-end paths (β = 18) and obviously all approaches have the same performance, the use of past measurement results leads to a smarter employment of resources and, consequently, a higher number of detected SLA violations. Besides that, it is possible to note that the use of both local node and remote information even improves the detection of SLA violations.

V. Related Work

The problem of probe assignment was tackled in some works over the past years. Some of these works are discussed as follows.

Sekar et al [9] proposed CSAMP, a centralized optimization engine for system-wide flow monitoring. The main features of CSAMP are the use of flow sampling, hash-based packet selection, and a centralized engine for distributing responsibilities across routers. The authors claim that CSAMP can provide greater monitoring coverage and an improved use of router resources. However, as CSAMP relies on a centralized party there are concerns about reliability as well as the cost and delay associated with the dissemination of routing manifests. Furthermore, the approach requires modifications in the measurement mechanisms.

Pietro et al [3] proposed DECON, a decentralized coordination system aimed at assigning monitoring probes. Authors

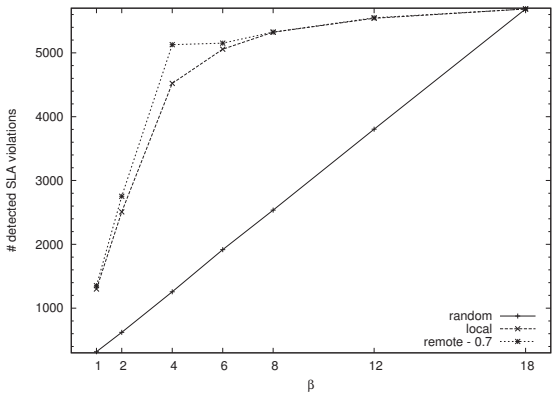


Fig. 3. Number of detected SLA Violations

claim that DECON scales up to large numbers of flows without requiring network topology information, traffic matrices and packet marking. However, as DECON operates using a detached P2P overlay, it is necessary to add up the ownership cost of additional hardware due to the detached overlay.

Barford et al. [1] proposed a framework for detecting and localizing performance anomalies based on using an active probe-enabled measurement infrastructure deployed on the periphery of a network. The authors aim at full coverage through decomposing end-to-end paths and selecting the paths over which probes will be sent. However, the measurements of metrics that need to be end-to-end measured are less accurate as the framework is focused at network decomposition. Besides that, the framework assumes a centralized controller for path selection.

VI. Conclusions and Future Work

Critical networked services established between service provider and customer are expected to operate respecting Service Level Agreements (SLAs). An interesting possibility to monitor these SLAs is using active measurement mechanisms. However, these mechanisms are expensive in terms of network devices resource consumption and also increase the network load because of the injected traffic. In addition, monitoring all paths in a large and complex network infrastructure is usually too costly. The common approach, enabling the observation of just a subset of network paths (defined by a human administrator), does not scale well and is ineffective on dynamic network conditions. This can lead to SLA violations being missed, which invariably affects the performance of several applications.

In this paper we propose a solution to increase the number of SLA violations detected by an active measurement mechanism in a network infrastructure. Our proposal aims at the utilization of past service level measurement results and resource constraints in a per network device basis. Our solution is based on Peer-to-Peer (P2P) principles, such as a high degree of decentralization and the use of local data and logic. In this context, network devices perform probe placement decision algorithms and exchange information to improve these decisions. In order to enable this exchange, network devices build an "in-network" management overlay using the concept of correlated peers, i.e., devices that have correlated measurement results. We have presented an evaluation of our solution through simulation experiments. The results show that the proposed probe placement algorithms perform significantly better than the random probe placement.

Although the proposed solution shows good results in simulation experiments performed until the present moment, it is necessary to evaluate more network topologies (e.g., in number of network devices) and conditions (e.g., using data from network traces). We also intend to investigate probe placement decision algorithms for cooperation among network devices in order to further increase the number of detected SLA violations. Furthermore, our ongoing work is aimed at developing a prototype to run on commercial network equipment. This is feasible since current network equipment vendors provide an increasing level of processing power and programmability in their devices.

References

[1] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, april 2009.

[2] M. S. Chiba, A. Clemm, S. Medley, J. Salowey, S. Thombare, and Y. E., "Cisco service level assurance protocol," Work in progress as an Internet-Draft, Internet Engineering Task Force, November 2011.

[3] A. di Pietro, F. Huici, D. Costantini, and S. Niccolini, "Decon: Decentralized coordination for large-scale flow monitoring," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM) Workshops*, march 2010.

[4] P. A. P. R. Duarte, J. C. Nobre, L. Z. Granville, and L. M. R. Tarouco, "A P2P-Based Self-Healing Service for Network Maintenance," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011)*, may 2011.

[5] L. Z. Granville, D. M. da Rosa, A. Panisson, C. Melchiors, M. J. B. Almeida, and L. M. R. Tarouco, "Managing computer networks using peer-to-peer technologies," *Communications Magazine, IEEE*, vol. 43, no. 10, pp. 62–68, 2005.

[6] K. Hedayat, R. Krzanowski, A. Morton, K. Yum, and J. Babiarz, "A two-way active measurement protocol (twamp)," RFC 5357 (Proposed Standard), Internet Engineering Task Force, October 2008.

[7] P. Mahadevan, C. Hubble, D. Krioukov, B. Huffaker, and A. Vahdat, "Orbis: rescaling degree correlations to generate annotated internet topologies," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, pp. 325–336, 2007.

[8] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Proceedings of the 9th International Conference on Peer-to-Peer (P2P)*, 2009.

[9] V. Sekar, M. Reiter, W. Willinger, H. Zhang, R. Kompella, and D. G. Andersen, "Csamp: a system for network-wide flow monitoring," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, april 2008.

[10] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, and M. Zekauskas, "A one-way active measurement protocol (owamp)," RFC 4656 (Proposed Standard), Internet Engineering Task Force, September 2006.

[11] N. Spring, R. Mahajan, and D. Wetherall, "Measuring isp topologies with rocketfuel," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, pp. 133–145, 2002.

[12] F. Wuhib, M. Dam, and R. Stadler, "A gossiping protocol for detecting global threshold crossings," *Network and Service Management, IEEE Transactions on*, vol. 7, no. 1, pp. 42–57, 2010.