Towards Automatic Network Fault Localization in Real Time using Probabilistic Inference

Andreas Johnsson and Catalin Meirosu
Research Area Packet Technologies
Ericsson Research
Kista, Sweden

***Abstract*—This paper describes the foundation of a novel network fault localization algorithm based on active network measurements and probabilistic inference. A fault condition could be an unacceptable large delay or packet loss rate.**

The solution is computationally efficient, autonomic in nature and provides the operator with a probability mass distribution that indicates the fault location. The probabilistic inference is based on a discrete state-space particle filter.

We present results from a first feasibility study performed in a simulated environment. The precise location of single faults is determined rapidly when measurement paths are partly overlapping.

Keywords—Autonomous network management, IP networks, Distributed network measurements, Discrete state-space particle filters

I. INTRODUCTION

Network planning, provisioning, monitoring and fault diagnosis are traditional network management tasks often carried out by highly skilled personnel in network operation centres. The tasks are complex and involve many hours of manual labour. For this reason vendors and operators in the data- and telecommunications industry focus the research and developing resources on automating tasks and workflows in order to reduce OPEX.

One particularly time consuming task, which is addressed in this paper, is to find the cause of performance issues and other faults in the network. First, the fault or performance degradation has to be observed. This is done either by a network user or by the network management administrator. Trouble tickets are filled in, registered and put in a queue. Then precise localization of the problem is carried out based on combinations of ping and traceroute and similar functionality for Ethernet and MPLS. Anecdotal evidence indicates that this can take hours or days to resolve [1].

Several measurement frameworks have been proposed in the academic literature to tackle the challenges discussed above. Section II.B reviews examples of systems related to this paper. Potential widespread deployment of such measurement and monitoring frameworks, at least when operated in pro-active modes, will generate significant amounts of data. Significant management efforts are needed for tool configuration, as well as collecting, storing and interpreting the data. Further, real-time aspects are most often not even considered. Several methods and architectures based on probabilistic modeling and inference have been proposed to overcome some but not all of these problems.

This paper describes the basic principles for a network fault localization algorithm based on active network measurements, real-time probabilistic inference and modelling of the network state. The algorithm is autonomic in nature and provides the operator with a fault location probability mass function that indicates the location of the fault. The required measurements can be performed by random scheduling thus reducing the efforts in measurement planning. The analysis itself is based on a discrete state-space particle filter, sometimes referred to as a histogram filter. The filter also performs an online computation with no need for storing results from previous measurements. The filtering analysis is lightweight from a computational perspective and does not need detailed knowledge about the network technology being measured. The paper also presents initial evaluation results obtained in a simulated environment.

The rest of this paper is organized as follows; Section II reviews recent advancements on network measurements, measurement frameworks and network fault localization; Section III provides the research problems targeted by this paper; Section IV reviews definitions; Section V describes the algorithm in detail; in Section VI the algorithm is evaluated in various scenarios; Section VII discusses the evaluation results as well as how the research problems are addressed by the solution; Conclusions are located in Section VIII.

II. RELATED WORK

A. Active network measurements

Active measurements (aka active probing) have long been an accepted method for determining performance parameters of packet-switched networks. The basic concept is to transmit probe packets from a sender towards a receiver. Each probe packet is time stamped on both sides.

The measurement endpoint (MEP) and measurement intermediate point (MIP) functionality and capabilities depends on the network technology deployed. For an IP network, the MEP functionality can be based on IETF TWAMP [2] [3] or IETF ICMP [4]. For Ethernet and MPLS networks, the functionality can be based on ITU-T Y.1731 [6]. For MPLS-

978-3-901882-50-0

TP, MIP and MEP functionality may also be based on IETF RFC 6371 [23].

These technologies are capable of measuring performance metrics such as one-way delay, round-trip time, loss, jitter and available path capacity as defined in ITU-T Y.1540 [5].

B. Measurement frameworks and probabilistic modelling

Several measurement frameworks have been proposed in the academic literature including examples such as Ripe Atlas [22], Blanton et al [10] and Boote et al [11]. Building the particular set of measurements to be triggered at a given moment in time is not presented in the above articles. Anecdotic evidence regarding the use of for example the PerfSONAR framework [11] leads us to believe that designing a particular set of measurements is a manual process performed by the network administrator. Also, interpreting the results and identifying the fault location is usually a manual process where the administrator is heavily involved.

Varga and Moldován [14] describe a fault management framework for service-level monitoring in Ethernet services, based on recommended performance metrics defined in e.g. ITU-T Y.1731. The framework is split in modules taking care of connectivity fault management, performance monitoring, service-level monitoring, and security. The performance monitoring is based on measurements of delays and drop. The fault localization is assumed to be performed using manual or semi-automated checks and processes.

One heavily explored approach for automatically finding the fault location is network tomography, see Castro et al [18] for an overview and [17] for a specific example. These techniques require summations of parameters of interest over all possible paths throughout a network. Such techniques have non-linear computational complexity. It is well known that, in general, network tomography techniques scale badly with the size of the network. Fast detection and localization of failures is very difficult when limited resources are available.

Lee et al. [12] present a method for choosing, in a tree topology, a set of candidate nodes and performing fault localization using this set of candidate nodes. The complexity of the heuristic algorithm that returns the minimum subset of candidate nodes for a tree is $O(|N^3|)$, N is the number of nodes.

Fraiwan and Manimaran [13] formulate the finding of network fault locations as a generalized weighted bipartite matching problem. The aim is to select a set of overlay measurements for characterizing a given underlay fault. The problem is solved with a maximum flow algorithm such as Ford-Fulkerson. Before starting the analysis, a complete set of suspect links had to be generated by a network management system.

Probabilistic management techniques [16] are solutions that are probabilistic in one or several aspects. For example, the network state can be represented by a probabilistic model; the output can be provided in terms of probability density functions; and the network observations can be based on sampling rather than deterministic information. These techniques have been applied to fault localization based on real-time measurements. For example, Steinert and Gillblad [15] describe a way of applying overlapping estimators for latency modeling based on measurements performed between neighboring nodes. The idea is to use the model for identifying latency shifts. The problem with this solution is that it only applies to a path and not a meshed network.

Several methods for network diagnosis and fault localization based on graph modeling, decision trees and Bayesian reasoning have been proposed in academic literature. Examples include the work presented in [19] [20]. In [21] a decision tree model is used for analyzing latency shifts and their root cause.

In contrast to the approaches reviewed in this section we present a solution with low computational complexity and real-time properties. The solution is designed for mesh and tree topologies.

III. Research Challenges

The overarching challenge targeted by this paper is related to difficulties in localizing faults and performance degradations in real time in packet-switched networks. The difficulty is further increased by the fact that there might be multiple packet layers (Ethernet, MPLS and IP) having overlay topologies controlled by different independent protocols. In addition, not all MIPs support performance measurements, and sometimes traffic related to fault management is filtered out or not responded by intermediary nodes. A solution must address the following aspects:

1. Automatically find fault and performance degradation locations in real time with low-complexity and scalable calculations
2. Hide complexity introduced by multi-technology and multi-layer networking
3. Reduce storage requirements for measurement data (i.e. store as little data as possible)
4. Relax requirements on intelligent measurement scheduling

This paper describes a novel algorithm pursuing all four aspects. The following sections provide the basic building blocks and an evaluation to show the algorithm feasibility.

IV. Definitions

A network is for the purpose of this paper modeled as an undirected graph G. For simplicity, the graph only allows maximum one edge between any pair of nodes. (The method described in this paper is not limited to these simple graphs.) The network can be any type of packet-switched network based on technologies such as MPLS, Ethernet and/or IP.

A path between two nodes is defined as an ordered sequence of edges. The path can be found by deploying Dijkstra's shortest path algorithm. It can also be defined by traffic engineering using network protocols such as RSVP-TE [8].

Some of the nodes in G have active measurement capabilities; such nodes are called Measurement Endpoints (MEPs). Two MEPs exchange test packets from which it is

possible to determine performance metrics such as delay, jitter and packet loss.

Each path between two edge nodes in G is associated with a service-level agreement (SLA). The SLA is a contract between the network operator and the network user that defines performance metrics and their associated threshold values. An SLA violation occurs if, for example, measurements of delay are higher than what the SLA dictates.

V. Network Fault Localization Algorithm

This section describes the network fault localization algorithm. We begin with an overview, while the following subsections discuss detailed aspects of the algorithm, including the measurement model, the filter and classification mechanisms for interpreting filter outcome.

A. Algorithm overview

The process of localizing a fault is depicted in Figure 1. Active edge-to-edge measurements of for example jitter, delay or loss are performed between MEPs. This is illustrated in the left part of the figure. The measurements are then fed into the network management system (NMS) where the network fault localization algorithm operates. The algorithm infers the location of a fault as illustrated in the right part of the figure.

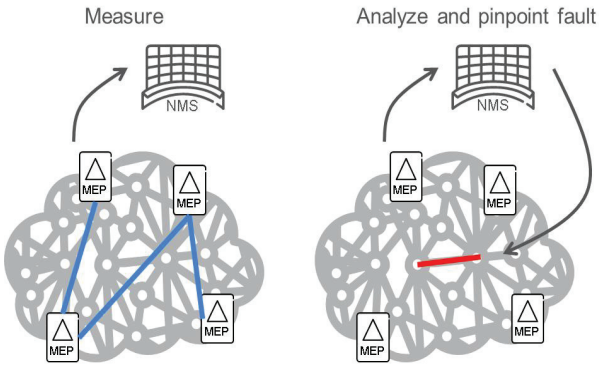

Figure 1: Overview of the network fault localization algorithm. Blue lines correspond to edge-to-edge measurements and the red line is the pinpointed faulty segment after filter analysis.

The network fault localization algorithm itself operates on a probability mass distribution which is implemented as a vector mapping network segment identifiers to probability mass distribution values (referred herein as weights). The weights are either increased or decreased, depending on the outcome of the measurements.

Each measurement result is compared to an agreed SLA for the path between the two MEPs. The algorithm increases all weights corresponding to the edges of a path where an SLA violation is detected. All other edge weights are always decreased. After each modification of the weights the vector is normalized. The weights in the vector must sum up to one, in order to make the weights correspond to probabilities.

All edge probabilities will be fairly equal – and proportional to the number of edges in G if there is no SLA violation or fault in the network. If, on the other hand, there is an edge in the network not meeting the SLA requirements, the probability associated with that edge will increase over time. That edge can then be identified by classification of the edge vector.

B. Network measurement model

The network fault localization problem is modeled using a discrete state-space particle filter. The basic concept behind filter-based estimation in general is to track a system state by repeated sampling [9].

The model for particle filters assumes that the system state can be modeled as a first order Markov process such that

$$a_k = g(a_{k-1}) + \omega_k \quad (1)$$

where a_k is system state at time k, ω_k is noise with some probability density function and $g(.)$ is an arbitrary function.

It is also assumed that consecutive measurements z_k of the system state a_k are independent of each other. Further, measurement of z_k shall only depend on a_k such that

$$z_k = h(a_k) + v_k \quad (2)$$

where a_k is the system state at time k and v_k is noise with some probability density function.

In this paper the system state space is discrete and corresponds to the set of edge identifiers. Hence the noise term in (1) will be zero. The fault location (i.e. an edge in G) is modeled as the system state a_k. If there are no faults in the network the state a_k points to a virtual null state. Observe that there are no general constraints on the functions $g(.)$ and $h(.)$.

Further, a measurement z is defined by a vector $<m_i, m_e, P, b>$ where m_i is the ingress MEP, m_e the egress MEP, P is the path between m_i and m_e and b is a Boolean which is either true or false such that

$$b = \begin{cases} true & SLA\ not\ OK \\ false & SLA\ OK \end{cases} \quad (3)$$

The interpretation of (3) is the following; if there is an SLA violation for the given metric then $b = true$, otherwise $b = false$. The elements of z are denoted z_{mi}, z_{me}, z_P and z_b. Observe that z_P is a sequence of edges between z_{mi} and z_{me}.

The measurements z_k corresponds to active measurements of metrics such as delay, jitter or loss between two MEPs over a path P. The measurements are independent, as required by the model, under the assumption that two measurement packets does not co-exist on the same edge or node at the same time. In a practical scenario this is the case due to the low measurement overhead.

C. Particle filter for network fault localization

The particle filter in the solution uses a discrete state space and evolves in discrete time [9]. Such filters are sometimes referred to as histogram filters. The algorithm operates on a probability mass distribution where each discrete state-space point corresponds to exactly one particle. This paper does not provide a thorough description of the mathematical foundation

of filters; instead the focus is on applicability to network fault localization.

The basic principle of the particle filter is to construct a discrete sample based representation of the probability function for the tracked system state. In a traditional particle filter multiple copies, particles, of the system state are used. Each particle is associated with a weight that corresponds to the probability of that specific particle. The system state estimate is obtained by calculating the weighted average of all particles. A new estimate is obtained for each sample.

The solution in this paper contrasts from the traditional use of particle filters by discretizing the system state space by assigning each particle to exactly one such discrete point in the state space (i.e. one edge from the set of edges). The weight of each particle is updated according to the measurement model; observe that the particle position is not changed. This is a key differentiator and is necessary for the purpose of this paper. The system state a_k is determined by classifying the probability mass function. This step is discussed in Section V.D.

Particle filters are iterative in their nature and operates in two phases; the prediction and the update phase. In the prediction phase each particle is updated according to a model known to govern the system state, if such a model exists. Further, the update phase recalculates the particle weight based on measurements, or samples, of the system.

In this paper, a particle x is represented by a vector $<e, w>$ where e is the edge identifier and w is the particle weight which is a normalized probability. The elements of x are denoted x_e and x_w. Each particle x belongs to the particle set S and the number of particles in that set is denoted $|G|$ – that is the number of edges in the graph G.

The prediction phase is modeled as $a_k = a_{k-1}$ for simplicity. That is, the method does not try to predict where a fault might occur in the future. Hence there are no requirements on specifying *g(.)* in (1). The update phase is based on the measurement samples from active measurements. The pseudo code for the algorithm is shown below.

1. Construct initial set S of |G| particles with equal weight 1/|G|
2. S′ = {0}, i.e. S′ is the empty set
3. For i = 1, …, |G|
   a) Set $x'_i = x_i$
   b) Calculate the new weight $w'_i = p(z, x'_i)$ for particle x'_i from the set S given sample z, add Gaussian noise to w'_i
   c) Update the new particle set S′, S′ = union(S′, $\{x'_i\}$)
4. Normalize w'_i for i = 1, …, |G| (i.e. $\Sigma(w'_i)$ must be 1 since w'_i corresponds to probabilities)
5. If the probability w'_i is higher than a threshold value T, the edge corresponding to x'_i is considered the fault edge
6. S = S′

The algorithm iterates (step 2 – 6) for each new measurement z. Gaussian noise is added to the weight component of a particle x' in order to increase the efficiency of the algorithm [9]. The mean and standard deviation of the noise are configurable parameters.

The key to finding the edge causing SLA violation using particle filters is to define the sample z (as done above) and to define a weight update function *p(z, x)*. In this paper the weight is updated according to

$$w_k^i = w_{k-1}^i p(z,x) \tag{4}$$

where

$$p(z,x) = \begin{cases} \delta & (z_b = true) \wedge (x_e \in z_P) \\ \gamma & otherwise \end{cases} \tag{5}$$

where $\delta > \gamma$. The function is used, in combination with the normalization step, to either increase or decrease the current weight of a particle. This means that if edge x_e is on the path P where an SLA violation was detected (i.e. $z_b = true$) then the weight x_w increases. Otherwise the particle weight decreases.

D. Filter classification mechanism

There is a need to define a strategy for determining the system state a_k based on the particle filter. That is, determining that a specific edge (or several edges) is the root of an SLA violation. Traditionally, a_k at time k is estimated by a weighted average calculation considering all particles in the filter. This is not practical when the state space is discrete. Further, there is no linear relation between two edges. Also, consider the fact that particle filters by definition are multimodal meaning that there can be several peaks in the probability function.

This paper utilizes a simple classification strategy based on selecting a hard threshold value for the particle weights. If the weight of a specific particle crosses this threshold, and stays there for a defined period of time, that particle is selected as the fault location and hence a_k is assigned that edge identifier value.

For multimodal solutions several edges can be determined to be the cause of a fault according to the above classification. In such case a_k will be a vector of edge identifiers. This is briefly discussed in the next section on evaluation.

VI. Evaluation

This paper presents a proof-of-concept evaluation using a simple discrete event simulator implemented in Java and MATLAB. It is based on simulations and only targets the scenario with no more than one simultaneous fault.

A. Evaluation setup

The network topology, which is illustrated in Figure 2, is represented by an adjacency matrix of Boolean values. The network contains 8 MEPs, 25 nodes with switching/routing capabilities and 44 edges. The mesh network topology is similar to parts of a mobile backhaul aggregation network. Dijkstra's algorithm is used to calculate the path between two MEPs. Further, the measurements are simulated by random choice of two MEPs (M0 – M7), then that MEP pair is associated with a delay measurement that is either below or

above a threshold predefined in an SLA. The simulation repeats the measurements 1000 times.

Two scenarios are studied in this evaluation. For scenario 1 there is no performance degradation in the network during time intervals (0, 199), (400, 599) and (800, 1000). Performance degradation, in terms of increased delay, occurs in the complementary time intervals (200, 399) and (600, 799) on edges E19 and E7 in Figure 2, respectively.

For scenario 2 the network is fault free except in the time interval (400, 600). In this case the performance degradation is located on edge E12.

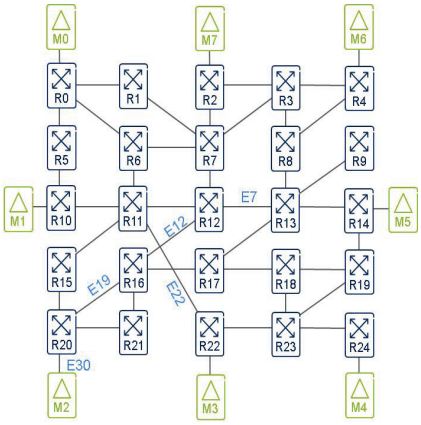

Figure 2: Simulated network topology. R0 – R24 are nodes with routing/switch capabilities while M0 – M7 have measurement capabilities.

The particle filter parameters δ and γ are set to 5.0 and 0.975 respectively. These values are chosen based on experience from experimenting with the algorithm. However, future work will provide a thorough tuning study. A hard threshold for classification of the filter is set to 0.25. Automation of threshold-based detection is also subject to further study.

B. *Evaluation results*

Figure 3 shows 6 snapshots of the probability mass distribution as it varies over time for scenario 1. In this case the performance degradation is located at edge E19 between time 200 and 400. At time 200 the probability mass function is, as expected, approximately uniformly distributed. A spike corresponding to Edge E19 is clearly visible already at time 250. The intensity increases over time and stays above 0.5. The performance degradation disappears at time 400 and the spike is almost gone after 50 additional measurement cycles. Based on the detection threshold the fault is localized at time 250.

Performance degradation also occurs between times 600 and 800 for scenario 1 as described in Section VI.A. Figure 4 shows 6 similar snapshots of the probability mass distribution function and how it varies over this new time interval. In this case the degradation is located at edge E7 instead of E19. There is a clearly identifiable spike at time 650 and hence the fault is localized.

Figure 5 shows six snapshots corresponding to scenario 2. In this case the fault is located on edge E12 in Figure 2. Observe that three peaks are visible in the probability mass distribution after adding the fault. That is, the solution is multimodal in this case. The peaks correspond to edges E30, E19 and E12. We need to determine whether such distribution reflects real problems on those segments or is an artifact of the fault localization method itself.

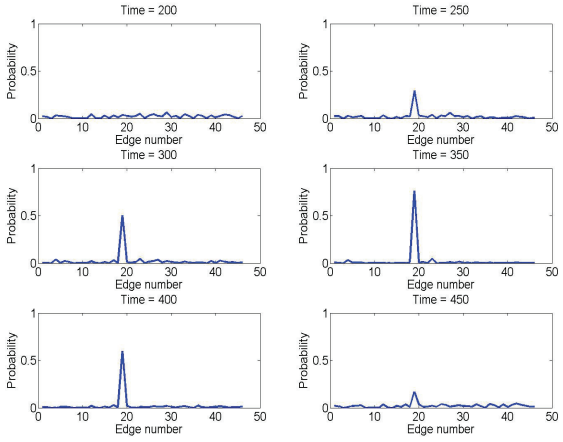

Figure 3: The probability mass distribution snapshots for scenario 1 at times spanning from 200 – 450.

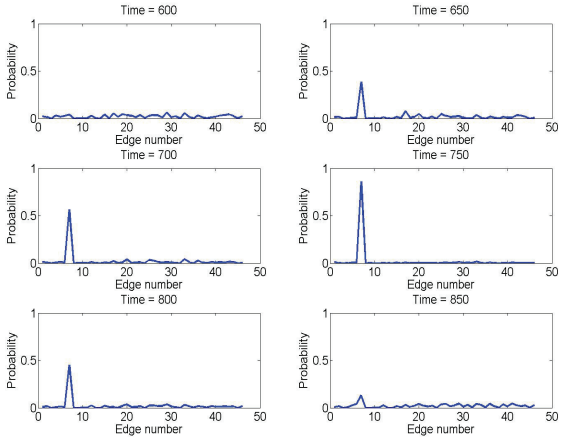

Figure 4: The probability mass distribution snapshots for scenario 1 at times spanning from 600 – 800.

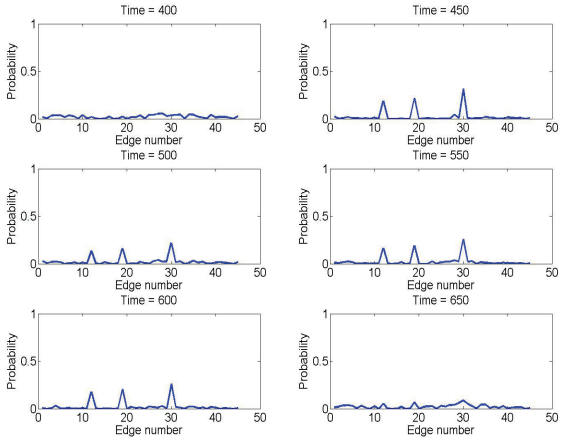

Figure 5: The probability mass distribution snapshots for scenario 2 at times spanning from 400 – 650.

The explanation for the multimodal solution in this case is that the subpath E30:E19:E12 does not overlap in a desirable way with other path between the MEPs in the topology. In other words there are no measurement samples that can decrease the weights corresponding to E30 and E19 (knowing that the fault is located on E12). As such, the multi-modal distribution is due to the combination of scheduling strategy and the way of building the particle filter. Observe that the algorithm still indicates the part of the network where the fault happened.

VII. Discussion

This section briefly walks through the advantages of the network fault localization algorithm in the light of desired properties briefly introduced in Section III and the evaluation results in Section VI.

The algorithm does find the location of faults and performance degradations in real time as shown in the previous section. That is, property 1 in Section III is covered.

The algorithm is technology agnostic which makes it suitable for multi-layer fault localization. The edges in the filter may represent adjacencies at different network layers using different technologies. Some edges in the filter could represent Ethernet links while others could correspond to IP/MPLS links. This relates to property 2 in Section III and we plan to further investigate this in the future.

The algorithm only requires storage for the elements of the filter. That is, the model parameters are stored instead of all measurement data collected over a period of time. Memory requirements are thus linearly dependent on the number of edges in the network graph. This targets property 3 in Section III.

The current scheduling of measurements between MEPs is trivial; it is randomized thus targeting property 4. However, the time to find the location of the fault depends on various parameters such as MEP placement and the outcome of the random measurement scheduling process.

Further, the algorithm is simple to implement, effective and also computationally efficient. The computational complexity of the algorithm actually increases linearly with the number of edges in the network. That is, the number of floating point operations per filter update is $O(|G|)$. This complexity is lower than what traditional network tomography solutions propose.

VIII. Conclusions and Future Work

This paper presented the foundations for a novel automatic network fault localization algorithm based on probabilistic inference using particle filtering and distributed active measurements. Key aspects of the algorithm include real time analysis of data, technology-agnostic operations and computational scalability. This paper also presented a first small-scale evaluation of the network fault localization algorithm. The results show promise.

For future work, we are considering in-depth assessments with additional scenarios and topologies, including multi-layer. Further, it is also crucial to develop an enhanced method for filter classification and automatic handling of multimodal solutions. This includes algorithms for determining whether the multimodal solution originates from one fault (i.e. non-overlapping path problem) or several faults.

References

[1] Ari Banerjee. “Assurance of Real-Time Cloud Services Requires Insights From Correlated Content, Sessions & IP Topology Planes”. White paper from Heavy Reading, August 2012.

[2] K. Hedayat et al. “A Two-Way Active Measurement Protocol (TWAMP)”. IETF RFC 5357, October 2008.

[3] S. Baillargeon et al. “Ericsson TWAMP Value added octets”. IETF RFC 6802, November 2012.

[4] J. Postel. “Internet Control Message Protocol”. IETF RFC 792, September 2001.

[5] International Telecommunication Union (ITU-T) Recommendation Y.1540, March 2011.

[6] International Telecommunication Union (ITU-T) Recommendation Y.1731, July 2011.

[7] R. Braden et al. “Resource Reservation Protocol (RSVP)”, IETF RFC 2205, September 1997.

[8] D. Awduche et al. “RSVP-TE: Extensions to RSVP for LSP tunnels”. IETF RFC 3209, December 2001.

[9] A. Doucet, A. M. Johansen. ”A Tutorial on Particle Filtering and smoothing: fifteen years later”. Technical report, Department of Statistics, University of British Columbia, December 2008.

[10] E. Blanton, S. Fahmy, S. Banerjee. “A Framework for an On-Demand Measurement Service”. Technical report at University of Purdue University, USA.

[11] B. Tierney, J. Boote, E. Boyd, A. Brown, M. Grigoriev, J. Metzger, M. Swany, M. Zekauskas, Y. Li, and J. Zurawski. “Instantiating a Global Network Measurement Framework”. LBNL Technical Report LBNL-1452E, January 2009.

[12] P. Lee, V. Misra, D. Rubenstein. ”Toward Optimal Network Fault Correction via End-to-End Inference”. In proceeding of INFOCOM, USA, May 2007.

[13] M. Fraiwan, G. Manimaran. "Localization of IP Links Faults Using Overlay Measurements," IEEE International Conference on Communications, May 2008.

[14] P. Varga and I. Moldován. “Integration of Service-Level Monitoring with Fault Management for End-to-End Multi-Provider Ethernet Services”. In IEEE Transactions on Network and Service Management, 2007.

[15] R. Steinert, D. Gillblad. “Long-term adaptation and distributed detection of local network changes”. In Proceedings of IEEE GLOBECOM, USA. 2010.

[16] A. G. Prieto, D. Gillblad, R. Steinert, A. Miron. ”Towards Decentralized Probabilistic Management”. In IEEE Communications Magazine, July 2011.

[17] T. Rizzo et al. “High quality queuing information from accelerated active network tomography”. In Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities (TridentCom '08), 2008.

[18] R. Castro, M. Coates, G. Liang, R. Nowak and B. Yu, “Network Tomography: Recent developments". In Statistical Science, 2004.

[19] G. J. Lee, “CAPRI: A Common Architecture for Distributed Probabilistic Internet Fault Diagnosis”, Ph. D. dissertation, CSAIL-MIT, Cambridge, MA, USA, 2007.

[20] F. J. Garcia-Algarra, P. Arozarena-Llopis, S. Garcia-Gomez, and A. Carrera-Barroso, "A Lightweight Approach to Distributed Network Diagnosis under Uncertainty", In Proceedings of Intelligent Networking and Collaborative Systems, Nov 2009.

[21] Y. Zhu, B. Helsley, J. Rexford, A. Siganporia, and S. Srinivasan, “LatLong: Diagnosing Wide-Area Latency Changes for CDNs”. In IEEE Transactions on Network and Service Management, September 2012.

[22] RIPE Atlas, URL (October 20, 2012) https://atlas.ripe.net/

[23] I. Busi et al. “Operations, Administration, and Maintenance Framework for MPLS-Based Transport Networks”. IETF RFC 6371, Sep 2011.