Closed loop control in network monitoring

Jochen Kögel
IsarNet Software Solutions GmbH
Munich, Germany
jochen.koegel@isarnet.de

***Abstract*—The increasing programmability of network and server environments allows for more flexibility at lower cost, but also leads to more dynamic traffic and behavioral patterns. This demands for more sophisticated and automated network monitoring for planning and troubleshooting capabilities in operations. However, programmability also enables more dynamic network monitoring capabilities that adapt to the changing behavior quickly. Using suitable mechanisms for closed loop control can achieve maximum insight at minimum resource utilization, as our concepts and current work shows in the following.**

I. Introduction

While concepts of programmable and softwarized networks are often considered for carrier and mobile networks, this work targets at enterprise networks. Different operational models exist, ranging from integrating network services only down to a transport service provider for the overall enterprise. Enterprise networks are business critical and performance degradation or malfunction has increasing impact on revenue as virtually every business process is network-based.

Fig. 1 shows in the middle a simplified scenario of an enterprise network monitoring scenario: interfering traffic impacts performance of a business critical application. Network monitoring goes beyond tracking component state, but analyzes traffic and performance characteristics for planning and problem solution purposes. Typical workflows consist of several steps, which are mainly performed manually today (Fig. 1 top). The AutoMon project [1] aims at automation of this workflow, as shown on the bottom.

First *Configuration* of monitoring functions and the monitoring system is required. This defines which data is acquired and how it should be processed. Today this information is kept rather simple and static. Thus, potentially far too much data is created while information for problem-specific insight is still missing. This is avoided by dynamic, automated configuration changes (bottom), which also hides complexity to the user.

In the *Problem analysis* step, today's systems provide sophisticated graphical data analysis views for interactive (but yet manual) analysis drill-down for gaining insight. Automated problem analysis, however, uses problem detection techniques such as learning-based anomaly detection. Depending on the problem state, configuration is quickly updated and allows for far more fine grained monitoring approaches. This principle of closed loop control is detailed in the next section.

This work was partly funded as part of the AutoMon project by the German Federal Ministry of Education and Research (BMBF) under contract No. 16KIS0408K. Responsibility for the information and views expressed in this publication lies entirely with the author.

Today an *Analysis report* is manually created using the insights and results of the previous step. With an automated approach this is replaced by built-in traceability of automated problem analysis functions. The same applies to *Linking other data* from incident and configuration handling systems. Instead of doing this manually, AutoMon provides data browsing functionality across different systems while giving expert feedback to the problem analysis. The last step of creating a *Management Report* requires obtaining additional business information today. In AutoMon a business intelligence component derives trends and correlations to business items.

II. Control in network monitoring

A. Controller concept

Fig. 2 illustrates our control scenario. It shows the monitored network with monitoring functions on the left and the AutoMon system for data analysis on the right.

Typical monitoring functions (mf, left) comprise of active measurement (e.g., active delay probing), flow-based traffic metering (e.g. IPFIX and Flexible NetFlow), and function state monitoring (e.g. QoS/queue utilizations). Such functions have become more sophisticated in recent years and send their data to monitoring systems (arrow M_{mf} in Fig. 2), where they are analyzed. In addition, the systems themselves are also monitored in terms of resource utilization (M_{ns}, M_{as}).

With the increasing availability of general data models (YANG) and APIs (RestConf) as well as SDN Controllers, monitoring functions can be deployed and (re)configured much faster. This does not only allow for quick initial deployment, but also for continuous adaption of the configuration to the current state of the network (Arrow C_{af}), which is the task of the AutoMon controller (top right). Further APIs for control exist to the underlying monitoring systems, e.g. for requesting additional VM resources (C_{as}), and APIs to analysis functions for configuring processing and storage granularity (C_{as}).

The AutoMon controller itself takes information on current monitoring accuracy/confidence, the problem state in the network, and system resource usage as input. Based on the latter it calculates the resource utilization of the network and monitoring system parts and combines it with information about system configuration in order to create resource models on how monitoring granularity, network load and resource utilization correlate. The controller logic then decides about configuration changes for increasing/decreasing monitoring granularity, i.e., which functions need to be deployed or reconfigured, in order to achieve the monitoring granularity

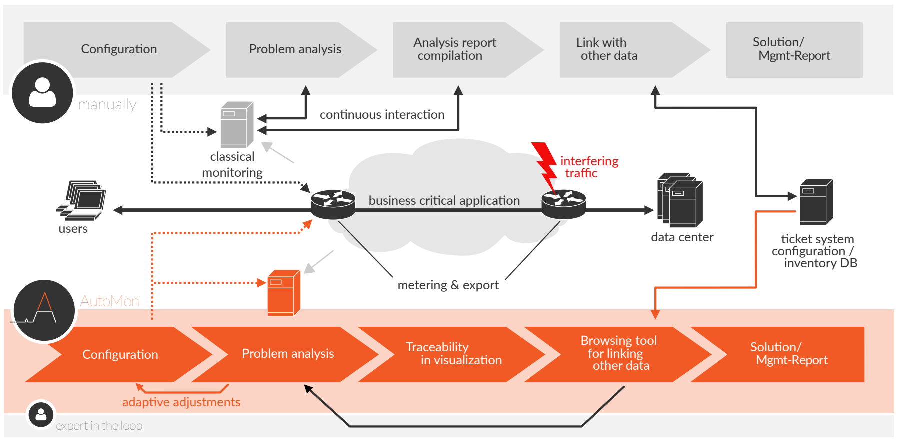

Fig. 1. AutoMon overview [1]

required for gaining the best insight. By taking the problem state with current confidence F_a, current resource utilization F_r, and resource models F_m into account, the controller tries to maximize the insight from available overall resources. This closes the control loop shown in Fig. 2 consisting of configuration changes as controller output, monitoring data as measured system behavior and feedback from processing/analysis as controller input.

B. *Example: Network-wide delay measurement*

Our current work deals with large scale passive delay measurement [2]. This provides an example on how closed loop control can help in finding optimal monitoring results.

While delay issues have heavy impact on business applications, active delay measurements have limitations: First, suitable end points need to be detected/configured. Second, measurement traffic may take different paths or experience ACL, firewall or protocol-specific effects. Consequently, the typically deployed active and passive measurements provide a limited view only leaving certain paths unobserved.

The work presented in [2] performs passive delay measurements based on TCP timestamps with packet sampling. This does not provide a complete accurate insight, but can provide enough samples for a first indication of delay problems on unobserved paths, which can trigger more fine grained measurements via the controller. In addition to this, the controller also adapts TCP-based measurement in terms of adapting the network functions parameters regarding packet sampling and the processing functions for timestamp analysis: While obtaining more and larger packet samples leads to a higher accuracy and confidence, this also requires more resources on the monitoring device, in the network and on the monitoring system. As this trade-off depends on the number of TCP timestamps found, it is traffic and end system dependent. This means there is no general rule of thumb for reasonable parameter settings, but closed loop control is mandatory.

Acknowledgment

The author thanks Sebastian Meier for reviewing the paper. Furthermore, we thank the AutoMon application partners DB Systel GmbH and IBM AIS for valuable input and fruitful discussions. Thanks to Sandstorm Media for creating the AutoMon overview figure.

References

[1] AutoMon project website: https://automon-projekt.de
[2] S. Meier, J. Kögel: Indirect passive measurement of network characteristics in the AutoMon project. NMRG Workshop on Measurement-Based Network Management at IETF 99, Prague, 2017.

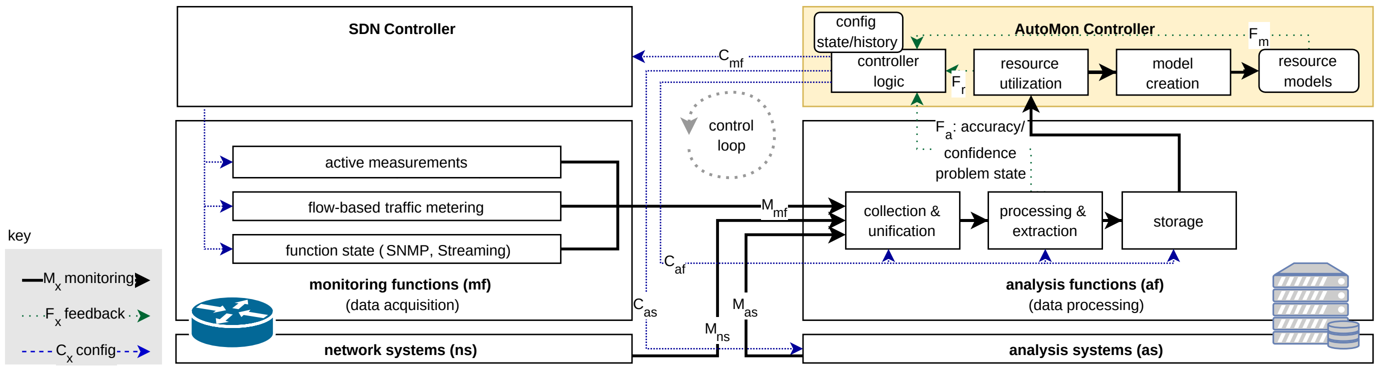

Fig. 2. Closed loop control with the AutoMon controller