Nils Richerzhagen, Tao Li, Dominik Stingl, Björn Richerzhagen, Ralf Steinmetz, and Silvia Santini:
A Step Towards a Protocol-Independent Measurement Framework for Dynamic Networks. In: 40th Annual IEEE Conference on Local Computer Networks (LCN 2015), p. 671-674, October 2015. ISBN 978-1-4673-6770-7

A Step Towards a Protocol-Independent Measurement Framework for Dynamic Networks

Nils Richerzhagen*, Tao Li†, Dominik Stingl*, Björn Richerzhagen*, Ralf Steinmetz*, and Silvia Santini†
*Multimedia Communications Lab, Technische Universität Darmstadt, Darmstadt, Germany
Email: {nils.richerzhagen, dominik.stingl, bjoern.richerzhagen, ralf.steinmetz}@kom.tu-darmstadt.de
†Embedded Systems Lab, Technische Universität Dresden, Dresden, Germany
Email: {tao.li, silvia.santini}@tu-dresden.de

***Abstract*—Existing measurement frameworks typically assume that the communication protocols and mechanisms running on the devices do not change during network operation. However, recent research efforts show that by enabling devices to switch between protocols and mechanisms at runtime the overall network performance can be improved. In this paper, a novel measurement framework that enables the continuous and consistent measurement of monitoring metrics even across such adaptations in networks is presented. The framework exploits monitoring metrics locally on the devices (i) irrespectively of the used mechanisms or protocols on the devices and (ii) allows other mechanisms and applications in the network to adapt to changes by referring monitoring information from the framework. A proof-of-concept prototype of the measurement framework is used to show that the work represents a promising step towards protocol-independent, adaptive monitoring in dynamic networks.**

I. Introduction

A large number of communication mechanisms for static and dynamic networks have been presented in the literature. The characteristics of these mechanisms may vary significantly depending on their goal and the specific type of network they have been designed for. For instance, in wireless sensor networks (WSNs) communication protocols must be able to operate on sensor nodes, which have very limited computational and memory resources [1]. In mobile networks, the mobility of the nodes must be taken into account [2]. In delay-tolerant networks communication mechanisms must cope with frequent and possibly long periods of lack of connectivity [3].

This large variety of available communication mechanisms – hereafter also referred to as *protocols*, for simplicity – allows to cope with an accordingly large number of different network conditions. However, these conditions rarely remain constant during network operation. For instance, the average velocity at which nodes move or the occurring data traffic load might change over time or can be different in parts of the network. Accordingly, the ability of a protocol to provide for good performance might also change.

As an example, consider a network of 50 nodes running the Optimized Link State Routing (OLSR) [4] protocol. Figure 1 shows that the performance of OLSR – expressed in terms of overall data delivery ratio – decreases significantly when the average velocity of the nodes increases from 0 to 7 m/s. Under the same conditions, the data delivery ratio of nodes running the Ad hoc On-Demand Distance Vector Routing (AODV) protocol [5] remains nearly constant. Thus, changing at runtime the mechanisms that operate in the network can result in better network performances, as also recently shown in [6], [7], [8].

Whether or not a mechanism should be replaced at runtime depends on current network and environmental conditions as well as on application-specific requirements. To gather information about these conditions and requirements a *monitoring* mechanism is needed. Existing monitoring approaches, however, typically assume static, non-changing mechanism configurations. To address this issue, we present a measurement framework, dubbed Proton, that can operate in networks in which communication mechanisms change at runtime.

Proton is a modular measurement framework that relies on a set of generic interfaces to measure relevant data about communication mechanisms running on network nodes. This data can then be reported to other nodes using state-of-the-art data collection mechanisms, such as [8], [9], [10], which can be readily integrated in Proton. We implement a prototypical version of Proton on the OMNeT^{++} simulator and evaluate its feasibility. In particular, we show that Proton can continuously and consistently measure monitoring metrics in ad hoc networks in which routing protocols can be exchanged at runtime. The evaluation of Proton in more generic settings – e.g., other type of protocols, networks, and a multi-mechanism scenario – is left to future work.

The remainder of this paper is organized as follows. In Section II we outline the main requirements Proton must be able to comply with and describe its architecture. Section III describes Proton's preliminary proof-of-concept evaluation results. In Section IV we review representative examples of monitoring frameworks for wireless networks. Finally, Section V concludes the paper.

II. Proton: Design of a Protocol-Independent Measurement Framework

The main rationale behind the design of the measurement framework is to enable a node to continuously measure basic monitoring metrics also when (protocol) adaptations occur at runtime. In this section, we first highlight the requirements Proton must be able to comply with. Subsequently, we present Proton's architecture, focusing on its key components.

978-1-4673-6770-7/15/$31.00 ©2015 IEEE

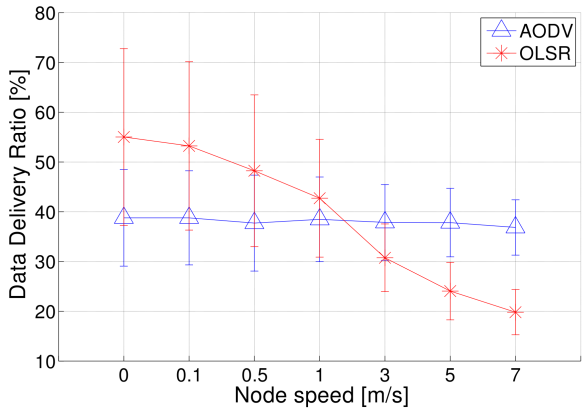

Fig. 1: Average data delivery ratio obtained for the routing protocols AODV and OLSR under different node velocities.

A. Requirements

Proton must be able to provide information about the functioning (or malfunctioning) of a network irrespectively of which specific configuration of communication protocols and mechanisms is currently operating in the network. Consequently, Proton must be *protocol-independent*. Furthermore, Proton must be *adaptive*, i.e., it must be able to operate without disruptions when protocols or mechanisms are exchanged at runtime. Proton must also be *lightweight* because it is expected to run on a large number of – possible resource-constrained – devices, Its code-size and computational and memory overhead must thus be kept as low as possible. At the same time, Proton must provide a *comprehensive* monitoring service. This implies that Proton must enable the collection of a representative set of monitoring metrics, which in turn might need to be collected according to different modalities – e.g., at regular time intervals or upon the occurrence of an event. We designed Proton to make it able to comply with the requirements listed above. In the following, we provide a brief description of Proton's architecture and prototypical implementation. Due to space constraints, a more comprehensive description of Proton is left to future work.

B. Architecture

Figure 2 shows the generic architecture we devise for Proton. We define three main abstractions: *Monitoring Components*, a *Controller*, and *Monitoring Access Points* (MAPs). Proton distinguishes between two types of metrics: *low-level* and *high-level* metrics. The former do not require complex computations or coordination among nodes to be measured. Also, low-level metrics can be collected for the majority of existing mechanisms. Examples of low-level metrics include the number of messages sent and received by routing protocols. High-level metrics are instead computed as functions of one or more low-level metrics. The number of expected transmissions [11] and the expected transmission time [12] are examples of high-level metrics. The extension and refinement of the set of metrics is part of our future work especially, when

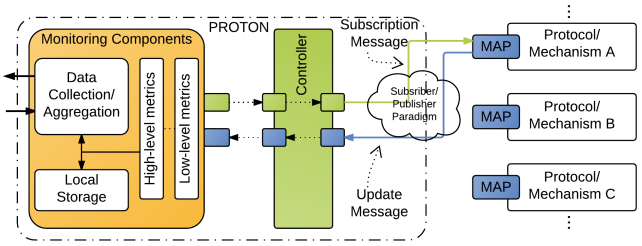

Fig. 2: Proton's main components: *Monitoring Components*, *Controller*, and *Monitoring Access Points* (MAPs)

combining Proton with adaptive overlay mechanisms, such as [6], [7].

Making Proton protocol-independent requires making it agnostic to the specific mechanisms that are currently operating on the nodes. This is achieved through the use of Monitoring Access Points (MAPs), which represent interfaces through which Proton can gather information from the mechanisms. A MAP specifies a set of metrics that can be measured along with corresponding tunable options (e.g., granularity, frequency). The monitored mechanisms must implement a MAP in their own code. We have implemented a prototypical version of Proton in OMNeT^{++} and can show that the implementation of MAPs causes a negligible increase of the overall code size of a protocol. For instance, the code snippet below shows the logic needed to update the value of a metric, which is implemented in only three lines of code.

```
if (monitoringOn) {
  forwarded_packet++;
  monitoring_access_point->
    updateMetricValue("forwarded_packet",
    forwarded_packet);
}
```

The fact that the use of Proton requires the code base of existing protocols to be modified is probably the most significant drawback of our framework. On the other side, making networks able to change protocols and mechanisms at runtime will inevitably require changes to network stack implementations.

The Controller shown in Figure 2 allows Proton to communicate with the monitored mechanisms. The communication between MAPs and the Controller adopts a subscriber-publisher paradigm to enable flexible selection and harvesting of monitored metrics. In Proton, the Controller subscribes to metrics. MAPs located in each protocol are the corresponding publishers of these metrics. The Controller creates a *subscription message* that contains the required metrics, including the specification of their updating strategy (e.g., periodical or threshold-based). MAPs in turn decode the subscription packet and adapt their own configuration accordingly. Once a metric is subscribed to, its values are published using *update messages*. The reasons behind our choice of a subscriber-publisher-model are (i) the heterogeneity of the mechanism (with respect to available metrics) and (ii) the heterogeneous requests for different dimensions in granularity of the monitored data.

Description	Symbol	Value	Unit
Total simulation time	T_{sim}	300	s
Network deployment area	$L_x \times L_y$	2500×1500	m
Number of nodes	$\#_{\text{nodes}}$	50	
Inter Packet Interval (IPI)	T_{IPI}	0.5	s
Randomization of IPI	T_{rand}	$[-0.1 \ldots 0.1]$	s
Protocol switch times	t_{tr}	80, 160, 240	s
Randomized node start time	startTime	$[1 \ldots 60]$	s

TABLE I: Descriptions, symbols, and values of the most relevant simulation parameters.

The role of the Controller is that of dispatching messages from and to Monitoring Components and MAPs. Thereby, the Controller can include additional logic to pre-process the data in a protocol-dependent manner whenever needed or appropriate. Thus, it is assumed that the Controller is aware of the deployed protocols on the node. This allows for additional flexibility and better decoupling between Monitoring Components and MAPs. The presence of the Controller is also meant to support situations in which more than one protocol is active at the same time. We assume this situation to be likely to appear for short time periods when an adaptation is performed.

Key features of the Monitoring Components abstraction are the *local storage* and component for *data collection*. The storage locally records metrics measured from protocols. Proton allows using different data structures or connecting with a database as the storage back-end. In the prototype of Proton, we use a cyclic queue to manage measured metrics on one single node. The data collection component can inherit complex collection and aggregation mechanisms such as [8], [9], [10]. In doing so, Proton can focus on local data measurement whereas other mechanisms that are specifically developed for data collection or aggregation can be plugged into the framework to enable intercommunication between monitoring devices.

III. Evaluation

The prototype of Proton allows us to show its ability to provide for monitoring capabilities in adaptive networks. In particular, we show that Proton is able to measure monitoring information across protocol adaptations irrespectively of the specific protocol running on the node. To this end, we consider two representative routing protocols – AODV and OLSR [5], [4]. We program Proton to monitor an exemplary low-level metric, the number of control packets sent by the routing protocol, indicated with CP_{tx}. For the simulation we rely on a scenario as specified by the parameters listed in Table I. We simulate a network of 50 nodes that are uniformly deployed at random over a rectangular area of $2500m \times 1500m$. Nodes move through the modeled area using the default *MassMobility* model of OMNeT^{++}. We let each node generate a data packet at a pre-defined Inter Packet Interval (IPI), which is to 0.5 seconds. To avoid packet collisions, we also add a random delay of ± 0.1 seconds to the IPI. Also, nodes start actively sending both control and data packets in the network only after a *startTime* delay has passed. The value of this delay is determined individually by each node and is chosen uniformly at random from the interval $[1s, 60s]$. We let the simulation run over 300 seconds so that at least a few hundreds packets can be transmitted by each node. In the beginning the AODV protocol is deployed on the nodes. At a pre-defined time, an adaptation is triggered and the OLSR protocol is dynamically loaded onto the nodes to replace AODV. At a later point in time, OLSR is replaced by AODV and then takes over again. We specify the timestamps for these adaptations to be 80s, 160s, and 240s. As a reminder, the pre-defined protocol adaptations are used in this proof-of-concept evaluation as a replacement for future adaptations triggered by other mechanisms like [6], [7], [8].

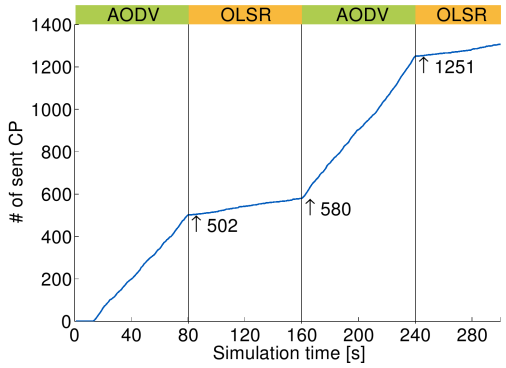

Fig. 3: Number of transmitted control packets over time for the node with ID 24.

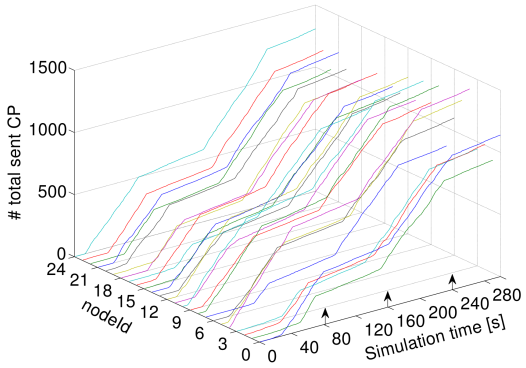

Fig. 4: Number of transmitted control packets over time for 25 nodes (identifiers 0 to 24).

Figure 3 shows the progress of the local value of CP_{tx} over the complete simulation time for a randomly picked node. The picture also outlines the different protocol adaptations and the exact values of CP_{tx} measured right after an adaptation occurs. During the first approximately 15s the value of CP_{tx} is zero due to the random start-up delay. Figure 4 shows the same plot as Figure 3 but for all nodes with identifiers between 0 and 24. Both figures show that Proton continuously captures monitoring metrics even when adaptations occur.

IV. Related Work

For consistency with the mentioned example and the evaluation that both consider mobile (ad hoc) networks, the related

work focuses more on that network area. Many monitoring solutions focus on the collection of monitoring data over the network, assuming that the monitoring data is already present locally. It is not further investigated how to measure and access data on nodes in the network. To tackle this problem the local data measurement is of main interest. Some approaches collect monitoring data through dedicated devices that are co-located with the network to monitor. They build a separate network for *passive* monitoring. MMAN [13] relies on dedicated nodes that passively collect data by overhearing messages exchanged by the monitored nodes. Similar to MMAN, the framework presented by Badonnel et al. [14] relies on separate nodes to capture and process the data over a two-tiered architecture. Badonnel et al. present an extensive network information model that can be extended to collect additional metrics specific to the underlying routing protocol. However, both passive approaches lack the possibility to retrieve more detailed information that cannot be overheard. Furthermore, none of the approaches is able to handle a protocol adaptation at runtime. The concept of Proton instead is not bound to a specific protocol or mechanism, as the concept of generic monitoring access points entails the flexible measurement of metrics during adaptations.

Active monitoring approaches directly measure data on the respective nodes. DAMON [15] is an example of such a two-tiered monitoring-architecture. It uses network nodes to periodically capture relevant monitoring data and then lets the nodes subsequently push this data to dedicated sinks. In addition to the monitoring of network metrics, DAMON is able to capture relevant statistics of AODV. Approaches like DAMON heavily rely on the availability of a specific routing protocol to work properly. Thus, they are unable to adapt protocol transitions that can occur at runtime. Mesh-Mon [16] and the Grid Base Hierarchy [17] operate on different routing protocols or can be extended to capture relevant metrics of a new routing protocol [18]. However, they are not able to adapt to the transition of a routing protocol at runtime and to capture protocol-independent metrics. Other very sophisticated data collection and dissemination approaches such as [8], [9], [10] assume that data is present locally on the nodes. Proton can be used for such approaches as component to measure that data.

V. Conclusions and Outlook

In this paper we have described the preliminary design and implementation of Proton, a protocol-independent measurement framework for dynamic networks. In this context, we derived the requirements that must be met by a measurement framework. Although still limited in terms of functionality, Proton represents a first concrete step towards enabling reliable local measurement in adaptive networks.

Our future work includes the extension of the metric set supported by Proton as well as the consideration of more advanced Monitoring Components. In particular, we until now focused on the local behavior of Proton on a single node. We plan to extend Proton's implementation with adequate collection and aggregation mechanisms such as [8], [9], [10]. Incorporating such approaches is highly relevant for the monitoring, as more accurate regional and global knowledge can be established and analysis steps are distributed in the network. Furthermore, we plan to use the framework in other scenarios than the one proof-of-concept scenario of this paper. One example for that may be a more user-centric application such as live video-streaming [6] or context based publish/subscribe [7]. As such applications/systems also need network information to derive the overall system load, a monitoring system has to continuously observe the system state even when e.g. dissemination or scheduling schemes are changed.

Acknowledgment

This work has been funded by the German Research Foundation (DFG) as part of projects B01/C02 within the Collaborative Research Centre (CRC) 1053 – MAKI.

References

[1] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325 – 349, 2005.

[2] S. Taneja and A. Kush, "A Survey of Routing Protocols in Mobile Ad Hoc Networks," *Innovation, Management and Technology*, vol. 1, no. 3, pp. 279–285, 2010.

[3] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," in *ACM SIGCOMM*, 2004, pp. 145–158.

[4] T. H. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," *RFC 3626*, 2003.

[5] S. R. Das, E. M. Belding-Royer, and C. E. Perkins, "Ad hoc On-Demand Distance Vector (AODV) Routing," *RFC 3561*, 2003.

[6] M. Wichtlhuber, B. Richerzhagen, J. Rückert, and D. Hausheer, "TRANSIT: Supporting Transitions in Peer-to-Peer Live Video Streaming," in *IEEE/IFIP Networking*, 2014.

[7] B. Richerzhagen, D. Stingl, R. Hans, C. Gross, and R. Steinmetz, "Bypassing the Cloud: Peer-assisted Event Dissemination for Augmented Reality Games," in *IEEE P2P*, 2014.

[8] N. Richerzhagen, D. Stingl, B. Richerzhagen, A. Mauthe, and R. Steinmetz, "Adaptive Monitoring for Mobile Networks in Challenging Environments (accepted for Publication)," in *IEEE ICCCN*, 2015.

[9] D. Stingl, C. Gross, L. Nobach, R. Steinmetz, and D. Hausheer, "BlockTree: Location-aware Decentralized Monitoring in Mobile Ad Hoc Networks," in *IEEE LCN*, 2013.

[10] D. Stingl, R. Retz, B. Richerzhagen, C. Gross, and R. Steinmetz, "Mobi-G: Gossip-based Monitoring in MANETs," in *IEEE/IFIP NOMS*, 2014.

[11] D. S. J. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing," *Wireless Networks*, vol. 11, no. 4, pp. 419–434, 2005.

[12] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks," in *ACM MobiCom*, 2004.

[13] H. Kazemi, G. Hadjichristofi, and L. A. DaSilva, "MMAN - A Monitor for Mobile Ad hoc Networks: Design, Implementation, and Experimental Evaluation," in *ACM WiNTECH*, 2008, pp. 57–64.

[14] R. Badonnel, R. State, and O. Festor, "Management of Mobile Ad Hoc Networks: Information Model and Probe-based Architecture," *Network Management*, vol. 15, no. 5, pp. 335–347, 2005.

[15] K. N. Ramachandran, E. M. Belding-Royer, and K. C. Almeroth, "DAMON: A Distributed Architecture for Monitoring Multi-Hop Mobile Networks," in *IEEE SECON*, 2004.

[16] S. Nanda and D. Kotz, "Mesh-Mon: A Multi-Radio Mesh Monitoring and Management System," *Special Issue: Modeling, Testbeds, and Applications in Wireless Mesh Networks in Computer Communications*, vol. 31, no. 8, pp. 1588–1601, 2008.

[17] I. Gupta, R. van Renesse, and K. P. Birman, "Scalable Fault-Tolerant Aggregation in Large Process Groups," in *IEEE DSN*, 2001.

[18] R. Riggio, M. Gerola, D. Miorandi, A. Zanardi, and F. Jan, "A Distributed Network Monitoring Framework for Wireless Networks," in *IFIP/IEEE IM*, 2011.