Integrating IP Traffic Flow Measurement: Overview

Marcelo Pias* Juergen Quittek† Marcus Brunner† Jon Crowcroft*

*University College London (UCL)
Department of Computer Science
Gower Street, London WC1E 6BT, UK

†NEC Europe Ltd.
C&C Research Laboratories
Adenauerplatz 6, 69115 Heidelberg, Germany

Abstract

In this short paper we present an overview of a work being discussed within the IETF in the area of flow measurements. We aim to discuss it and get feedback from the research/industry community in Brazil. At the same time, we want to identify where our work might be useful within the RNP2 (Internet2) groups. Those currently working within areas such as QoS, traffic measurements (IPv4 and IPv6) and policy-based network management may find this work relevant.

Keywords: Flow measurement, IETF RTFM architecture, Accounting, Charging, QoS, Policy-based network management

1 Introduction

The IETF working group on Realtime Traffic Flow Measurements (RTFM) has developed an architecture [8] defining components of traffic flow measurements systems and their interactions. Such components include a meter collecting traffic flow data, a reader obtaining flow data from the meter for further processing, and a manager which configures meters and readers.

Concerning the specification of flows to be measured, the architecture is very powerful. Each meter contains a programmable packet matching engine (PME), deciding what to do with observed packets. Traffic flows are specified by providing code for the engine. Available instructions, called rules, include matching conditions for single fields of the packets, conditional jumps, etc. A set of rules specify one or more flows to be metered. A meter may contain several rule sets with each of them being executed for each observed packet. While being a powerful and flexible method for specifying traffic flows, creating rule sets is also non-trivial and potentially complex.

When someone wants to perform precise and elaborated measurements of a network, it is appropriate and acceptable to develop a rule set matching his requirements manually in a few hours or a few days. However, when traffic flow measurement is only one component of a large management system and should be configured dynamically, then the procedural specification by rule sets is not adequate anymore.

At UCL and NEC we have observed this shortcoming of the RTFM architecture independently in several projects, where traffic flow measurements were to be integrated into management applications. In each of those projects, we solved the problem by developing a high-level abstraction layer offering to the particular management application exactly the required functionality and hiding the rest of the complexity. After doing this several times at both places we decided to jointly use our experiences for developing a more general abstraction of the RTFM architecture.

2 Applications Requiring Traffic Measurement

In a general view, we see two categories of applications using the RTFM architecture:

1. Plain (standalone) traffic measurement

   This kind of application is well supported by the architecture. The rule sets are created manually or with SRL [6] to be uploaded into the meter. The metered data is collected by the readers and stored (e.g. in a file) for further processing.

2. Applications with integrated traffic measurement

   Applications that require the automatic generation of rule sets and measurement data collection for immediate processing. The support by the RTFM architecture for those applications is not sufficient. Therefore, we understand that using a library is the suitable approach for the integration of applications and the RTFM architecture. The library should provide a functional and easy to use interface.

Two scenarios belonging to the second category are described below. The scenarios motivate the suggested interface for integrating traffic measurement into applications. The first scenario deals with accounting and charging for QoS. The subject of the second one is policy-based network management.

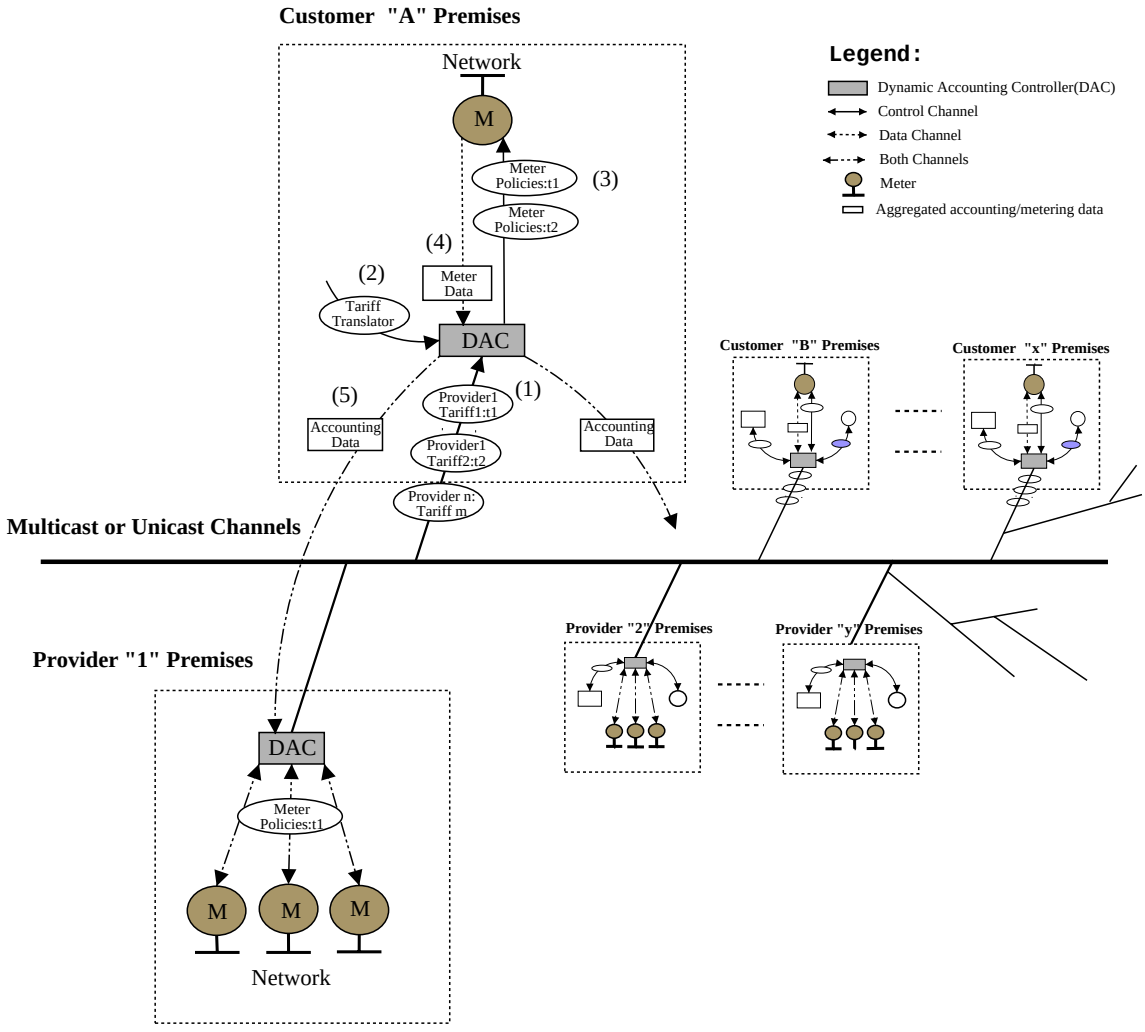

Figure 1: QoS Charging Scenario

2.1 Scenario 1: Accounting and Charging for QoS

In the scenario illustrated by Fig. 1, a service provider multicasts tariff objects to all accounting modules for his customers (1). Once a customer accounting module has received a tariff, it loads dynamically a module called Tariff Translator (TT) in order to perform a translation to appropriate Meter Policies (MP) (2,3). Thus, the MP is a set of ECA (Event-Condition-Action) generic policies derived from the tariff. It specifies explicitly/implicitly accounting/metering requirements. After that, the MP is transformed to meter specific control and uploaded into a meter. For the RTFM architecture, the meter control is a rule set as described in [8].

The provider/customer collect the accounting data from meters either using one of the following modes (4,5):

1. Pull Mode or Proactive: a reader polls the meter periodically in order to gather metered data, usually through a request/reply mechanism.
2. Push Mode or Reactive: a meter pushes out data to a set of readers/collectors registered as listeners when a pre-defined event occurs. We have identified so far some categories of events as follows:
   - packet-related (e.g. set of packets arrival),
   - temporal (e.g. clock timeout),
   - protocol-related (specific TCP stream)...

Finally, the bill is generated using the logic within the tariff. The translation performed in steps (2,3) has derived the accounting requirements. Therefore, the collected data from the meters is one form of input to calculate the usage charge. Other inputs might be derived from the marketing and business strategies of the service provider.

Each customer accounting module makes use of an API in order to cope with configuration and data collection. All the logic of those processes are encapsulated by the implementation of two partial interfaces which composes this API. One for control and another for data.

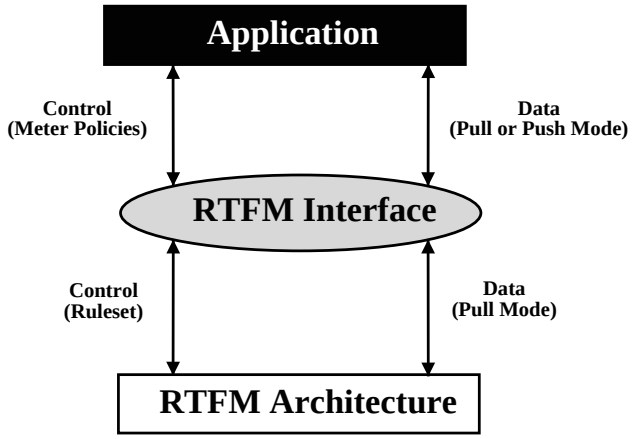

Figure 2: RTFM Abstraction Layer: Interface

The interface defines an abstract programming model for the RTFM architecture, which hides several details not easy understandable by application programmers. An implementation of the interface establishes an abstraction layer between the application and the RTFM architecture (Fig. 2).

2.2 Scenario 2: Policy-based Network Management

The subject of the second scenario is policy-based network management. Core components of a policy-based network management system are policy decision points (PDPs) and policy enforcement points (PEPs) [10].

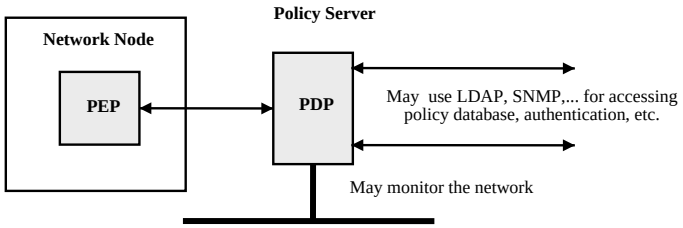

Figure 3: Policy Server Scenario

The PEP is located at a network node and enforces policy decision made by the PDP. Decisions of the PDP are based on policy rules stored at policy databases and on the current state of the managed network. Policy decisions may be triggered by requests of a PEP, by user interaction, or by other events in the monitored network (Fig. 3)

This includes two cases where traffic measurement functionality is required:

1. A policy may depend on current traffic information, e.g. the decision on whether a user is allowed to open a new connection may depend on the traffic volume he already produced. In this case, the policy server must be able to retrieve accounting information for that particular user.

   The requirement of the PDP application is similar to the pull mode in the first scenario. Loading a policy depending on transferred data volume from the policy database corresponds to the Tariff multicast. After parsing the policy, the PDP must initiate traffic measurements for all users affected by the policy, in order to have measured data available when needed. Therefore, it configures one or more traffic meters. When a decision has to be made, the PDP collects traffic data in pull mode.

2. The second case corresponds to case 2 (push mode) of the first scenario. A policy decision of the PDP, e.g. to reconfigure resource reservations, is triggered by traffic events, e.g. if the bandwidth or volume of a particular user exceeds a given limit.

   In this case, the PDP receives traffic information in push mode. An active entity monitoring traffic must notify the PDP.

Also for this scenario an interface establishing an abstract layer between the application (the PDP) and the RTFM architecture appears to be desirable. As in the first scenario, the interface hides details of traffic measurements, and offers functions for control and push/pull data transfer.

2.3 Requirements for Integrating Traffic Flow Measurement

The following requirements were identified and are addressed in this document:

1. Applications should be able to control traffic measurements at a high level of abstraction from the RTFM architecture. The level of abstraction should be as high as possible, but at the same time it should be as low as necessary for providing sufficient functionality.

2. Applications should be able to gather measured traffic data in pull and push mode at a high level of abstraction. Again, the level of abstraction should be as high as possible and as low as necessary.

3. The functionality provided at the interface should be sufficiently generic to support different kinds of traffic measurement applications including Accounting and Charging, QoS monitoring as well as plain traffic measurement applications.

3 Traffic Flow Measurement Interface

This section describes our proposal for an application-oriented high-level interface to the RTFM architecture. We first discuss a declarative flow model comparing it to the procedural model of the RTFM architecture. Then, we explain the notation used in the following semi-formal description of the interface. At the end of this section, we briefly summarize our implementation experiences.

3.1 Declarative Flow Model

One issue arising from requirement 1 of Subsection 2.3 is the specification of flows to be measured.

A specification by rule sets as defined by the RTFM architecture is rather complicated and error-prone, because it is a procedural specification. Rule sets are sequences of instructions to be executed by the pattern matching engine within the meter. A flow is specified by the results of the execution of this procedure for all possible packet headers passing through the meter.

This is not well suited for an application that needs to generate rule sets automatically. Thus, a declarative specification of flows has been chosen. A flow specification consists of a list of attribute constraints which may be exact, wild-carded or generic. The latter one may generate more than one flow table entry out of a single flow description. Note that we term flow specification and flow description in an interchangeable way. Thus, they refer to the same concept, although sometimes called differently.

Implementations of the RTFM interface have to provide the functionality of translating declarative flow specifications into procedural rule sets.

3.2 Interface Specification (Overview)

The interface specification is open to be implemented as API or as network protocol. The interface consists of five groups: FlowDescription, FlowAttribute, FlowRecord, Manager, Reader, and Meter. Each group contains a data structure and a set of functions on those data structures. In the case of FlowDescription and FlowAttribute the data structures are the more relevant part. On the other hand, the offered functions are of more importance for for Manager, Reader, and Meter.

The FlowDescription data structure specifies a single flow or a set of different flows to be metered. The FlowAttribute data structure describes a single attribute of a flow by an identifier and value. Within the interface definition, mostly sets of FlowAttribute data structures are used. The Manager, Reader, and Meter model well known components of the RTFM architecture.

Group Name	Description
FlowDescription	Specifies a single flow or a set of different flows.
FlowAttribute	Simple pair of an attribute ID and value.
Meter	Used for meter identification
Manager	Provides functionality of a RTFM Manager
Reader	Offers means for collecting data (pull or push modes)
FlowRecord	Data record generated by the FlowDescription installed in a Meter

Table 1: Interface Groups

Further information on the interface specification may be found in [3] [4].

3.3 Implementation

We have implemented the interface independently at NEC and UCL. The NEC implementation is a contribution to European research projects including the IST MobiVAS projects sponsored by the European Union.

Both implementations were made in Java. At NEC the NeTraMet software of Nevil Brownlee and the jmgmt package of Sven Doerr were used. NeTraMet and jmgmt are free available.

The UCL implementation sponsored by British Telecommunications (BT Labs,UK) has also used NeTraMet. It was necessary to port NeTraMet to Windows (98,NT,2000) platforms in order to cope with the metering

at customers premises (Fig. 1). Also, it was added to NeTraMet support for ECN (explicit congestion notification) [11] [5] for congestion-based tariffs. A preliminary version of this implementation was used within internal projects at BT and as a support measurement platform in the project P906-Quasimodo from the European Telecom companies consortium (Eurescom).

Practical experiences in the mentioned projects showed that the interface matches the requirements listed in Section 2.3. The abstraction is sufficiently high to get a very easy integration of traffic flow measurement into an application. Pull and push modes of data collection is supported and the functionality was sufficient for several applications.

4 Future Work

The future usage for the UCL implementation includes: (a) Market Managed Multiservice Internet (M3I) [9] when traffic metering is necessary in order to build specific scenarios; (b) UCL project funded by the Internet2 consortium [2] which aims to do trials for the M3I project.

5 Conclusion

We have described an application-oriented high-level interface to the IETF RTFM architecture. It simplifies the integration of traffic flow measurement into network and service management applications.

We motivated our work by describing application scenarios of management applications requiring integration of traffic flow measurement. From those scenarios we derived requirements for an application-oriented interface.

The presented interface matches those requirements. It replaces the complex procedural flow model of the RTFM architecture by a much simpler declarative one. Furthermore, it hides several details of the RTFM architecture.

Applications using our interface are able to control traffic measurements easily at a high level of abstraction. Still, they have sufficient functionality available. They can collect measured flow data either in pull or push modes.

The feasibility of the interface has been proven by the independent implementations. The use of those implementations in different projects also proved, that the interface is sufficiently generic for a wide range of applications.

□

References

[1] Bob Briscoe, Mike Rizzo, Jerome Tassel, Kostas damianakis, and Nicolai Guba. Lightweight Policing and Charging for Packet Networks. In *Third IEEE Conference on Open Architectures and Network Programming - OpenArch 2000*, March 2000.

[2] Internet2. Internet2 Consortium. URL: http://www.internet2.org/.

[3] Juergen Quittek and Marcelo Pias. A high-level application-oriented interface to the traffic flow measurement architecture (RTFM). *IETF I-D, work in progress*, October 2000.

[4] Juergen Quittek, Marcelo Pias, and Marcus Brunner. Integrating IP Traffic Flow Measurement. In *PAM2001: Passive and Active Measurements Workshop*. Amsterdam, March 2001.

[5] K. Ramakrishnan and S. Floyd. A Proposal to Add Explicit Congestion Notification (ECN) to IP. *IETF, RFC2481*, January 1999.

[6] N. Brownlee. SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups. *IETF, RFC2723*, October 1999.

[7] N. Brownlee. Traffic Flow Measurement: MIB. *IETF,RFC2720*, October 1999.

[8] N. Brownlee, C. Mills, and G. Ruth. Traffic Flow Measurement: Architecture. *IETF, RFC2722*, October 1999.

[9] M3I Project. M3I - Market Managed Multiservice Internet. URL: http://www.m3i.org/.

[10] R. Yavatkar, D. Pendarakis, and R. Guerin. A Framework for Policy-based Admission Control". *IETF, RFC2753*, January 2000.

[11] S. Floyd. TCP and Explicit Congestion Notification. In *ACM Computer Communication Review*, volume 24, October 1994.

[12] S. Handelman, S. Stibler, and N. Brownlee. RTFM: New Attributes for Traffic Flow Measurement. *IETF,RFC2724*, October 1999.