Validating Inter-domain SLAs with a Programmable Traffic Control System

Elisa Boschi[1], Matthias Bossardt[2], and Thomas Dübendorfer[2]

[1] Hitachi Europe
Sophia Antipolis Lab, France
elisa.boschi@hitachi-eu.com
[2] Computer Engineering and Networks Laboratory
Swiss Federal Institute of Technology, ETH
Zürich, Switzerland
{bossardt,duebendorfer}@tik.ee.ethz.ch

Abstract. For network users and service providers it is important to validate the compliance of network services to the guarantees given in Service Level Agreements (SLAs). This is particularly challenging in inter-domain environments. In this paper, we propose a novel solution for inter-domain SLA validation, based on programmable traffic processing devices that are attached to routers and located in several autonomous systems. Using our service management infrastructure, the measurement logic is deployed on the traffic processing devices in a flexible and secure way. We safely delegate partial network management capability from network operators to network users, which are enabled to configure service logic on the traffic processing devices. At the same time, the management infrastructure guarantees against negative influence of the network user's configuration on network stability or other user's traffic. Via the flexible configuration of service logic, our system gives network users powerful means to observe quality of service parameters agreed upon in SLAs. We present a detailed scenario of the SLA validation service and its deployment across several administrative domains.

Keywords: Inter-domain measurement, programmable networks, SLA validation, network service, management delegation.

1 Introduction

The need for verifiable quality differentiation of network services is one major trigger for the deployment of measurements in IP networks. Services like VoIP, multimedia streaming, video telephony or e-gaming require a minimum guaranteed level of network performance. Internet Service Providers (ISPs) negotiate a contract with their customers called Service Level Agreement (SLA) in which they specify in measurable terms the service to be furnished. One of the main problems faced by ISPs is how to deploy SLAs that cross ISP boundaries (inter-domain SLAs) to achieve end-to-end SLA enforcement. The problem stems from the fact that although ISPs can control and monitor their own network, which allows them to validate their intra-domain SLAs, they have only minimal information about the characteristics and performance of other networks. Also customers that stipulate SLAs with one single ISPs have concerns that the agreed

D. Hutchison et al. (Eds.): IWAN 2005, LNCS 4388, pp. 1–12, 2009.

Quality of Service has been met, and are therefore interested in end-to-end, inter-domain measurements.

Classical measurement architectures determine end-to-end, or edge-to-edge performance, comparing ingress and egress reports from two measurement devices located at the end points of a flow. These architectures though, are not sufficient to determine performance of specific path portions, or to determine which segments failed to provide the expected Quality of Service (QoS) in case the end-to-end guarantees are not met. If for instance the delay is higher than agreed in the SLA, it is not possible to determine in which administrative domain the higher delay occurred (or in other words: which ISP is responsible for not meeting the requirements). Another problem with such architectures is that they require to configure two edge devices and retrieve information from them. This configuration is difficult in case the devices are not located in the same administrative domain, since ISPs have major security related concerns in delegating any management function to third parties. These concerns are based on the risk that third party configurations may negatively affect network stability or other user's network traffic.

In this paper, we present a novel solution for inter-domain SLA validation that allows deploying measurement logic on distributed devices in a flexible and secure way. The system is flexible in that it allows the deployment of almost arbitrary service logic with just a few restrictions that we specify later in this paper. These restrictions, together with the concept of *traffic ownership* [9,5] are used in our system to address the ISPs' security concerns.

The goal of our architecture is to configure on demand several measurement devices along a flow path in a multi domain environment, and to process the raw measurement data, in order to determine the QoS experienced by the flow on several nodes of its path. With these kind of measures, an end user, or a monitoring application, could determine not only the end-to-end QoS, but also the QoS provided on different path segments even if belonging to different administrative domains.

This paper is organized as follows. Section 2 discusses the state of the art in inter-domain measurements. Section 3 discusses our distributed architecture for end-to-end SLA validation, describing the underlying traffic control system and how it can be used for effective, flexible and secure inter-domain measurements. A detailed scenario is presented in Section 4. Finally, we draw our conclusions in Section 5.

2 State of the Art

2.1 Inter-domain QoS Models

The authors of [17] distinguish three different models being pursued in the attempt to provide inter-domain QoS: bilateral, cooperative and third-party.

In the bilateral approach two providers interconnect at one or more points and agree on a set of metrics, measurement methods, service classes, settlements and issue resolution processes in a customized way. Generally the solutions agreed have a very low reusability for other peering agreements. Moreover these contracts involve just two parties, two neighboring ISPs, limiting the feasibility of end-to-end SLA validation (and QoS provision) to very "simple" cases.

Cooperative approaches extend bilateral ones by defining a set of rules that a group of cooperating ISPs has to follow to provide inter-domain QoS within that group. These rules include the definition of common metrics, common SLA monitoring and reporting methodologies, common tools. This approach requires standard measurement and reporting techniques, metrics, data formats. While the IETF [13] is partly working towards this goal, e.g. [16,14] a full standardized inter-domain QoS provision process is unlikely to come in the near future.

A more flexible approach is provided by the third-party model. There a third party composes an end-to-end service offer out of the single provider offerings and is responsible for the site-to-site measurement, and the metric definitions. Our approach falls into this category.

2.2 Inter-domain Measurement

The National Internet Measurement Infrastructure (NIMI) [20] is a software system for building network measurement infrastructures consisting of measurement servers (called NIMI *probes*) and measurement configuration and control software running on separate hosts. Each probe reports to, and is configured by, a Configuration Point of Contact (CPOC), typically one per administrative domain. The probes are accessed through access control checks, and communications between all NIMI components are encrypted via public key credentials. NIMI does not require a particular set of measurement tools, new tools can be included in the infrastructure by writing a wrapper for them and propagating both tool and wrapper to all NIMI probes.

The IP Measurement Protocol (IPMP) [15] is a protocol based on packet probes suited to measure packet delay at router level. This active measurement protocol operates as an echo protocol, allowing hosts in a domain to gather information from router units along the end-to-end path. However, only a limited set of measurements can be taken: packet loss, one-way packet length, round-trip time, and one-way delay.

In [3] inter-domain measurements are configured sending XML-based documents called Specification of Monitoring Service (SMS) to a controller, located on each administrative domain crossed by the flow to be monitored. The configuration is sent to the controller in the source domain, and then with a cascade model, each domain configures its intra-domain measurement and forwards the request pertaining to the rest of the path to the controller of the subsequent autonomous system. The Diameter protocol [11] is used for secure inter-domain communication.

All the above systems miss a flexible system to automatically deploy the requested services (i.e. a measurement technique for a particular metric) to the appropriate measurement devices in the network. Also, they do not provide adequate guarantees for network data privacy and against intended or unintended misuse of the system once the user has been authorized to configure the system. This considerably lowers the acceptance of a system especially if multiple network operators are involved.

3 Traffic Control System

To validate the conformance of a service to the guarantees given in an SLA involving several domains, it is necessary to set up an inter-domain QoS measurement. In this

section we describe the traffic control system that we use to configure, deploy, and perform the measurement. The measurement results are exported to a collector, and used as input for the QoS metrics computation done in a component called evaluator. Collector and evaluator are not part of the traffic control system and therefore are not further described in this section.

3.1 Network Model

The network model of the traffic control system distinguishes four different roles: *Internet number authority, Traffic control service provider (TCSP), Internet service provider (ISP),* and *Network user*.

The TCSP manages traffic control (TC) services. It sets up contracts with many ISPs that subsequently attach Traffic Processing Devices (TPDs), to some or all of their routers and enable their network management system to program and configure these devices (see Figure 1). The introduction of a TCSP helps to scale the management of our service. A network user needs only a single service registration with the TCSP instead of a separate one with each ISP.

A network user must first register with the TCSP before using the traffic control system. The TCSP checks the identity of the network user performing similar actions as a digital certification authority (CA), e.g. offline verification of an official identity card or online verification of a digital certificate issued by a trusted CA. To verify the claimed ownership of IP addresses the user wants to control traffic for, the TCSP checks with Internet number authorities if the IP addresses are indeed owned by the network user. Ownership of (ranges of) IP addresses is maintained in databases of organizations such as ARIN, RIPE NCC, etc. Upon successful user identification, access to the traffic control system is granted. The binding of a network user to the set of IP addresses owned and the subsequent verification when using the TC service is implemented with digital certificates signed by the TCSP.

After successfully registering to the basic TC service, a network user can initiate the deployment of a specific service (e.g. QoS traffic monitoring), which is implemented on top of the TC service.

3.2 Node Architecture

The node architecture is based on a legacy Internet router with basic filtering and redirection mechanisms. The router is extended with a programmable Traffic Processing Device (TPD), as shown in Figure 1.

Network user traffic can be redirected permanently based on source or destination IP address in the transported IP packet to the traffic processing device, processed according to the service requested by the network user, and further sent along its path. Services are composed of components that are arranged as directed graphs [18,6], each of which performs some well defined packet processing. When the TPD processes a network packet, it first executes traffic control, such as e.g. monitoring, on behalf of the owner of the source IP address (first processing stage) and subsequently on behalf of the owner of the IP destination address (second processing stage). The functionality of service components is restricted as specified in Section 3.3. For instance, service components

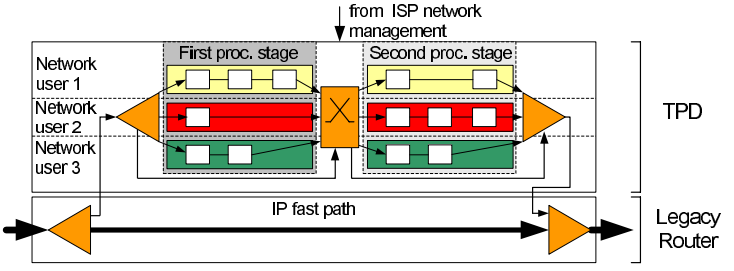

Fig. 1. Node architecture

that match traffic by header fields, payload (or payload hashes), or timing characteristics can be installed, configured, and activated instantly.

3.3 Security Considerations

For the proposed distributed traffic control service to be accepted by ISPs, it is vital, that traffic processing devices keep the network manageable by the network operators and that it cannot be misused for an attack itself. This is addressed by the core concept of the traffic control system, which is *traffic ownership*. We restrict the traffic control for each network address owner to his/her own traffic, i.e. packets to/from owned IP addresses. This allows our service to assure that traffic owned by other parties is not affected. Hence, collateral damage caused by misconfigurations or malicious behavior of users having access to such devices can be prevented. In addition, ISPs do not loose control over their network.

As any misuse of such a novel service must be prevented from the very beginning for gaining acceptance by network operators, we restrict it even further. We do not allow the adaptive device to modify the source and the destination IP address of a packet. Such rerouting could wreak havoc easily (causing routing loops, interference with other routing mechanisms, transparent source spoofing, or "forwarding" of attack traffic). Also the TTL (time to live) field of IP packets is a field we cannot allow to be modified as it aims to set an upper bound of network resources a packet is able to use. Furthermore, we need to prevent that the service can cause amplifying network-like effects. The traffic control must not allow the *packet rate* to increase. In addition, the amount of the network traffic leaving the traffic processing device must be equal or less[1] compared to the amount of traffic entering it. I.e. packet size may only stay the same or become smaller. New service components for the traffic processing devices must be checked for security compliance before deployment.

The security concerns of ISPs with respect to the danger of delegating partial control of the network from the network operator to the customers are adequately addressed as countermeasures against effects of misconfigurations and misuse were taken into consideration when designing the traffic control system.

[1] For e.g. logging, statistics or trigger event services, we will allow a reasonable amount of additional traffic.

3.4 Deployment Process

The deployment process is subdivided into *TCSP, ISP, TCU* and *Device* layer. A Traffic Control Unit (TCU) is defined as the combination of a router interface and all the TPDs that traffic from the interface can be redirected to. The TPDs can be physically separate, even located at different sites, or integrated into future routers. The complete deployment process is carried out at the management stations of TCSP and ISP. For each service a layer offers, a *service descriptor* specifies the following:

- The mapping of the service to sub-services offered by the layer below.
- The set of mandatory and optional parameters, their default values and their mapping to parameters for sub-services.
- Restrictions that direct the placement of service logic.

For each layer a database contains *context information* about the infrastructure relevant to that layer. These logical databases may be merged into two physical databases located at the TCSP and ISP management stations. Information at the TCSP layer includes the identities of contracted ISPs and properties of their networks, e.g. whether they transport transit traffic or provide a stub network, or BGP information. At the ISP layer relevant information includes the location of the TCUs, e.g. whether it is located at the border of a network or in the core network. At the TCU layer details about the pairing of TPDs and routers are kept as context information. Finally, at the Device layer information about the make and version of TPDs and routers and their configuration interfaces must be kept. Additionally, context information can contain dynamic state information about managed objects and deployed services.

Deployment logic on each layer *maps* the service request from the layer above to services provided by the layer below based on information provided by the service descriptors. Taking into account restrictions specified in the service descriptor and context information from the databases, sub-services are placed on the managed objects of the corresponding lower layer (ISPs, TCUs, TPDs and routers, respectively). The deployment process ends with the configuration of the devices that were selected to run part of the service logic.

4 Delay Variation Measurement Scenario

This section describes how the traffic control architecture can be used to perform end-to-end QoS measurements for SLA validation. We describe here the deployment of the delay variation measurement service. Other QoS measurements (e.g. one way delay, one way loss, round trip time) could be similarly performed.

4.1 Scenario Description

Let's suppose that a corporate Internet user, e.g. a video streaming company (from now on identified simply as "user"), wants to verify that the performance parameters agreed in an SLA stipulated with ISP_1 have been met. The SLA specifies guarantees on the jitter of a video the user sends from point A in the network of ISP_1 to point Z in

the administrative domain of ISP_n (cf. Figure 2). The user employs our traffic control system to measure the jitter of the video flow, and verify the SLA conformance of the service he's providing.

IP delay variation, or jitter, is defined as the difference of one-way delay values for selected packets [8]. This metric can be calculated by performing passive measurement of one-way delay [1] for subsequent packets (e.g. of a flow) and then calculating the differences. Jitter is particularly robust with respect to differences and skews of the clocks of the measurement points. This allows performing jitter measurements even if the TPDs are not synchronized. As described in [8], indications of reciprocal skew of the clocks can be derived from the measurement and corrections are possible.

The measurement requires the collection of data at two measurement points at least (in our case the TPDs) situated at the end points of the flow. If more measurement points are involved in the measurement, detailed information on jitter values on the different path segments (or autonomous systems) can be obtained.

The data returned by the TPDs need to be collected and post-processed to calculate the jitter. These actions are performed at a *collector* where data collected at the TPDs are exported to, and an *evaluator*, where the delay variation is computed.

In an inter-domain environment it is crucial to have standard formats and protocols to export measurement results. The IP Flow Information eXport (IPFIX) [7] protocol is about to become a standard for exporting flow information from routers and probes, while standardized methods for packet selection and the export of per packet information will be provided by the IETF group on packet sampling (PSAMP) [21].

4.2 Measurement Service Deployment at the TCSP Layer

The network user requests the TCSP to deploy the jitter monitoring service in the network, selecting the service from among those that the TCSP has made available (see Figure 2). The service, with the parameters it requires are described in the *service description* (see Figure 3a), while the user provides the necessary parameter values in the *service request* shown in Figure 2.

In the service request, the user identifies itself as the owner of the source address of the flow and specifies parameters necessary to the measurement service. Parameters listed are the source and destination addresses of the flow, the addresses of the uplink interface of source A to ISP_1 and the downlink interface from ISP_n to the flow destination Z, and the address of the data collector the measurement results have to be sent to. A set of fields that allow to further specify the flow to be measured are optional. In this example, we provide also source and destination ports.

The user can specify start and end time of the measurement. These parameters are defined as optional in the TCSP layer service description (see Figure 3a). That is, if no value is provided default values are used: the service is started immediately, as specified with the *default* element and/or ended when the given flow ends using a flow termination criterium that is hard coded into the service component.

The TCSP maps the request to the sub-service component *jitterOnEgressRouters* and selects appropriate ISPs according to the restriction as defined in the *restrictionDefinition*. In our case, this restriction limits the ISPs to those on the BGP path between A and

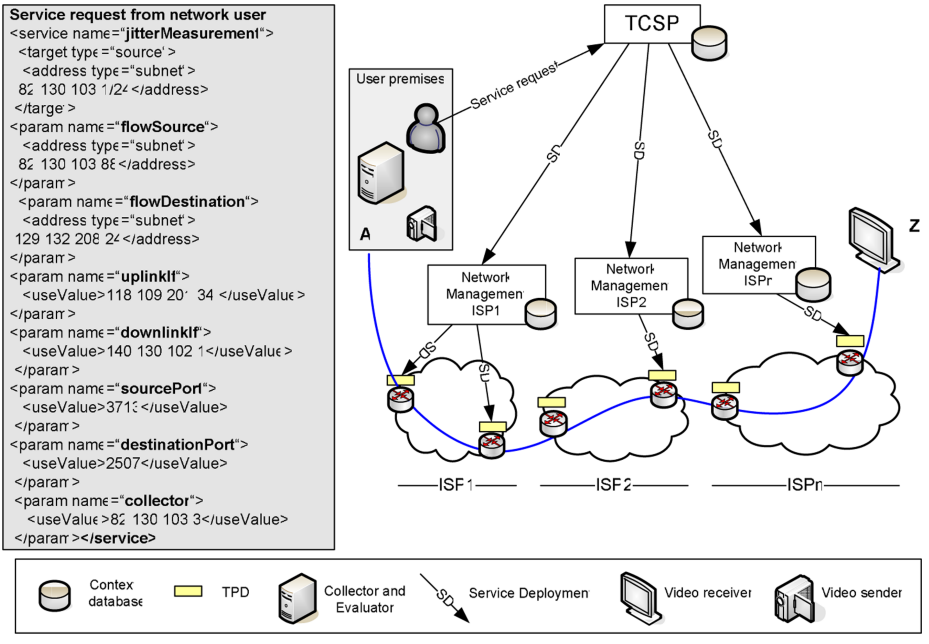

Fig. 2. Service request and deployment

Z[2]. The restriction yields true if the $ASnumber$ taken from the context database is one that can be found on the BGP path. The TCSP obtains the BGP path using the function *getBGPPath* specified in the service description for the TCSP layer (see Figure 3a).

Required and optional parameters needed by the *jitterOnEgressRouters* sub-service complete the description. The parameters can be defined as fixed values, taken from the network user's service request or calculated using a function (e.g. *getNextAS*).

4.3 Measurement Service Deployment at the ISP Layer

ISPs in turn select appropriate TCUs and TPDs to deploy and configure the jitter service components according to the service descriptor in Figure 3b. The restrictions are to deploy the service "only on egress routers on the path from previous autonomous system $prevAS$ to next autonomous system $nextAS$ (refID 1) and at the uplink (refID 2) and downlink interfaces (refID 3)".

The sub-services, or service components, to be deployed on all selected TPDs are specified in the service descriptor: *pktSelection*, *timestamp*, *IDGeneration*, *jitterRecordGeneration*, *ipfixExport*. The deployed components are shown in Figure 4. Packets belonging to the flow to be measured are first selected and then timestamped. Timestamping should be done as early as possible in order to get the best possible accuracy for the arrival time and to reduce further variable delay effects like variations in packet processing time. The selection function could either select all subsequent packets in a

[2] In case of BGP route changes the service must be deployed again using the same descriptors.

```
<service name="jitterMeasurement" xsi:type="tcspService">
 <target type="source" use="required"/>
 <param name="uplinkIf" use="required" />
 <param name="downlinkIf" use="required" />
<param name="sourceAddress" use="required" />
 <param name="destinationAddress" use="required" />
 <param name="sourcePort" use="optional" />
 <param name="destinationPort" use="optional" />
 <param name="protocol" use="optional" />
  <param name="collector" type="IPAddress"   use="required" />
 <param name="startTime" use="optional">
     <default><useFunction>getCurrentTime</useFunction></default>
 </param>
 <param name="endTime" use="optional"/>

<functionDefinition>
<function refID=1
 <useFunction name="getBGPPath">
     <arg name="destinationASNumber">
      <useFunction name="getASNumberFromIPAddress">
      <arg="IPAddress"><useParam>destinationAddress</useParam></arg>
      </useFunction>
     </arg>
     <arg name="sourceASNumber">
      <useFunction name="getASNumberFromIPAddress">
      <arg="IPAddress"><useParam>sourceAddress</useParam></arg>
      </useFunction>
     </arg>
 </useFunction>
</function>
</functionDefinition>

<restrictionDefinition>
  <restriction refID="1" xsi:type="selection">
     <scope>ISP</scope>
     <expressionL><useContext>ASNumber</useContext></expressionL>
     <relation>isIn</relation>
     <expressionR>
          <useFunctionRef refID="1" />
     </expressionR>
  </restriction>
</restricitionDefinition>

<subservice name="jitterOnEgressRouters">

  <restrictionRef refID=1 />

  <param name="prevAS">
     <useFunction name="getPrevAS">
      <arg="BGPPath"><useFunctionRef refID="1" />
      <arg="thisAS"><useContext>this.AS<useContext /></arg>
     </useFunction>
  </param>

  <param name="nextAS">
     <useFunction name="getNextAS">
      <arg="BGPPath"><useFunctionRef refID="1" />
      <arg="thisAS"><useContext>this.AS<useContext /></arg>
     </useFunction>
  </param>

  <param name="collector"><useParam>collector</useParam></param>
  <param name="sourceAddress"><useParam>sourceAddress</useParam>
  </param>
  <param name="destinationAddress">
     <useParam>destinationAddress</useParam></param>
  <param name="sourcePort"><useParam>sourcePort</useParam></param>
  <param name="destinationPort"><useParam>destinationPort</useParam>
  </param>
  <param name="protocol"><useParam>protocol</useParam></param>
  <param name="uplinkIf"><useParam>uplinkIf</useParam></param>
  <param name="downlinkIf"><useParam>downlinkIf</useParam></param>

</subservice>

</service>
```

(a)

```
<service="jitterOnEgressRouters" xsi:type="ispService">
     <target type="source" use="required" />
     <param name="prevAS" use="required" />
     <param name="nextAS" use="required" />
     <param name="collector" use="required" />
     <param name="uplinkIf" use="required" />
     <param name="sourceAddress" use="required" />
     <param name="destinationAddress" use="required" />
     <param name="sourcePort" use="optional" />
     <param name="destinationPort" use="optional" />
     <param name="protocol" use="optional" />

<restrictionDefinition>
   <restriction refID="1" xsi:type="selection">
     <scope>TCU</scope>
     <!-- all AS egress routers -->
     <expressionL><useContext>IPaddress</useContext></expressionL>
     <relation>isIn</relation>
     <expressionR>
     <useFunction name="getBGPNextHop">
          <arg name="nextAS"><useParam>nextAS</useParam>
          <arg name="prevAS"><useParam>prevAS</useParam>
     </useFunction>
     </expressionR>
   </restriction>
   <restriction refID="2" xsi:type="selection">
     <scope>TCU</scope>
     <!-- Uplink Interface -->
     <expressionL><useContext>IPAddress<useContext></expressionL>
     <relation>is</relation>
     <expressionR><useParam>uplinkIf</useParam></expressionR>
   </restriction>
   <restriction refID="3" xsi:type="selection">
     <scope>TCU</scope>
     <!-- last hop -->
     <expressionL><useContext>IPAddress<useContext></expressionL>
     <relation>is</relation>
     <expressionR><useParam>downlinkIf</useParam></expressionR>
   </restriction>
</restrictionDefinition>

  <subservice name="pktSelection">
     <restrictionRef refID="1"/>
     <!-- omitted references to parameters -->
   </subservice>
<subservice name="pktSelection">
     <restrictionRef refID="2"/>
     <!-- omitted references to parameters -->
   </subservice>
<subservice name="pktSelection">
     <restrictionRef refID="3"/>
     <!-- omitted references to parameters -->
   </subservice

  <!-- the same for the components timestamp, IDGeneration,
     jitterRecordGeneration, ipfixExport  -->

</service>
```

(b)

Fig. 3. Jitter measurement service description at the TCSP layer (a) and at the ISP layer (b)

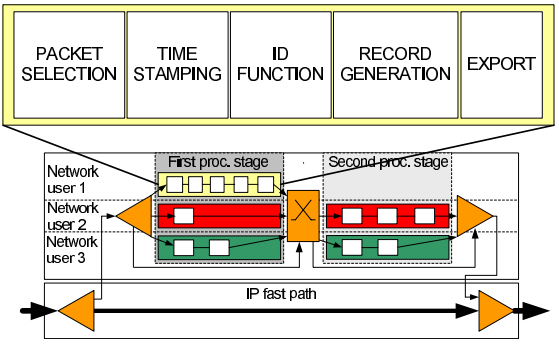

Fig. 4. Jitter measurement service components

given time interval, or be a sampling function. In our scenario, the *pktSelection* function selects all packets matching the parameters specified in the request: source and destination address, and source and destination port.

It is necessary to recognize the same packets captured at different measurement points to correlate packet arrival events. To recognize a packet parts of its header and eventually payload need to be captured. To reduce the amount of measurement data a unique packet ID can be calculated from the header and part of the content e.g. by using a CRC or hash function [12, 10, 23]. This identifier must be unique during a relatively long period of the flow measurement in order to avoid duplicate packet identification.

At least, timestamp and packet ID need to be exported. Packet size should be reported as well since it influences the measurement: the delay measurement starts with the first bit of the packet sent from the source and ends with the last bit received at destination. These data are exported with IPFIX using the solution proposed in [4], that optimizes the export of per-packet information and is therefore particularly suited in case of jitter measurements.

4.4 Scalability Considerations

The scaling factors that our distributed traffic control service depends on are 1) the number of service subscribers (i.e. network users), 2) the total number of ISPs deploying our service, 3) the number of service components installed per network user, and 4) the bandwidth of network links. These scaling factors influence several parameters:

Service logic and state per TPD. Following the estimation made in [5] on the number of users and number of services run on a TPD per user, the memory needed is a rather modest requirement.

Signalling effort. Even if we assume that each AS corresponds to one ISP and that all ISPs offer our distributed traffic control service, signalling overhead due to the secure distribution of the small service deployment messages by the TCSP to a few thousand ISPs is not a bottleneck.

Traffic processing capacity. A hardware based solution for our traffic processing devices is favorable. Research prototypes of FPGA based devices exist that can

concurrently filter 8 mill. flows [22] on a 2.5 Gbps (OC-48) link. According to [2], faster FPGAs allow achieving advanced packet filtering at 10 Gbps (OC-192).

A more detailed scalability analysis can be found in [5].

5 Conclusions

In this paper, we have shown how guarantees given in an SLA spanning multiple autonomous systems can be validated by setting up inter-domain QoS measurements. We described a scenario where an end-to-end jitter measurement service is deployed and automatically configured with our management system using a service description language.

The jitter measurement service is executed on a programmable traffic control system, able to safely delegate partial control over traffic processing devices to network users. Our system allows network users to deploy almost arbitrary measurement logic on distributed traffic processing devices attached to routers located in different autonomous systems. The service deployment has just a few restrictions, that guarantee against negative influence on network stability or other user's traffic. The measurement service is highly modular, i.e. composed of functional components. This modularity has the advantages of reusability of some components in other services, and it simplifies restriction compliance tests. Thus, we provide a network user with the means to locate his measurement logic on the end-to-end path to be monitored, and ISPs with guarantees against collateral damage due to users' intended or unintended misbehavior.

Jitter measurement is by no means the only service that can be provided by our TCS: one way delay, one way loss, RTT, flow volume are other metrics that can be measured (just to cite a few). Inter-domain measurement is not even the only application for our system. Mitigation of DDoS attacks, as well as other emerging applications based on the presented TCS have already been investigated in [9,5] and showed promising results.

Leveraging acceptance by ISPs is vital. We think that our traffic control system offers many incentives for ISPs and at the same time a high level of security against misuse, which was a major concern with other approaches in the field of active and programmable networks and is still one of the major concerns in inter-domain data exchange or control delegation.

Currently, we are implementing the measurement service components for Click router– [18] and Field Programmable Gate Array (FPGA)–based [19] traffic processing devices.

References

1. Almes, G., Kalidindi, S., Zekauskas, M.: RFC 2679, A One-way Delay Metric for IPPM (September 1999), ftp://ftp.rfc-editor.org/in-notes/rfc2679.txt
2. Attig, M., Lockwood, J.W.: A Framework for Rule Processing in Reconfigurable Network Systems. In: Proceedings of IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, USA (April 2005)
3. Boschi, E., Denazis, S., Zseby, T.: A Measuremnet Infrastructure for Inter-domain SLA Validation. Elsevier Journal of Computer Communications: Special Issue on End-to-end QoS Provision Advances (to appear)

4. Boschi, E., Mark, L.: Use of IPFIX for Export of Per-Packet Information, Internet-draft, work in progress (2005)
5. Bossardt, M., Dübendorfer, T., Plattner, B.: Enhanced Internet Security by a Distributed Traffic Control Service Based on Traffic Ownership. Elsevier Journal of Network and Computer Applications: Special Issue on DDoS and Intrusion Detection (to appear, 2005)
6. Bossardt, M., Hoog Antink, R., Moser, A., Plattner, B.: Chameleon: Realizing Automatic Service Composition for Extensible Active Routers. In: Wakamiya, N., Solarski, M., Sterbenz, J.P.G. (eds.) IWAN 2003. LNCS, vol. 2982. Springer, Heidelberg (2004)
7. Claise, B., Bryant, S., Sadasivan, G., Leinen, S., Dietz, T.: IPFIX Protocol Specification, Interrnet-draft, work in progress (2005)
8. Demichelis, C., Chimento, P.: RFC 3393, IP Packet Delay Variation (November 2002), ftp://ftp.rfc-editor.org/in-notes/rfc3393.txt
9. Dübendorfer, T., Bossardt, M., Plattner, B.: Adaptive Distributed Traffic Control Service for DDoS Attack Mitigation. In: IEEE Proceedings of IPDPS, International Workshop on Security in Systems and Networks SSN (2005)
10. Duffield, N., Grossglauser, M.: Trajectory Sampling for Direct Traffic Observation. In: ACM SIGCOMM 2000 (2000)
11. Calhoun, P., et al.: RFC 3588, Diameter Base Protocol (September 2003), ftp://ftp.rfc-editor.org/in-notes/rfc3588.txt
12. Graham, I.D., Donnelly, S.F., Martin, S., Martens, J., Cleary, J.G.: Nonintrusive and accurate measurement of unidirectional delay and delay variation on the internet. In: INET 1998 Proceedings (1998)
13. Internet Engineering Task Force, http://www.ietf.org/
14. IP Performance Metrics (IPPM), http://www.ietf.org/html.charters/ippm-charter.html
15. IPMP homepage, http://watt.nlanr.net/AMP/IPMP/
16. IP Flow Information Export (IPFIX), http://www.ietf.org/html.charters/ipfix-charter.html
17. Jacobs, P., Davie, B.: Technical Challenges in the Delivery of Interprovider QoS. IEEE Communications Magazine, 112–118 (June 2005)
18. Kohler, E., Morris, R., Chen, B., Jannotti, J., Kaashoek, M.F.: The Click Modular Router. ACM Transactions on Computer Systems 18(3), 263–297 (2000)
19. Lockwood, J., Naufel, N., Turner, J., Taylor, D.: Reprogrammable network packet processing on the field programmable port extender (FPX). In: Proceedings of the ACM International Symposium on Field Programmable Gate Arrays (FPGA 2001) (February 2001)
20. NIMI National Internet Measurement Infrastructure, http://www.ncne.nlanr.net/nimi/
21. Packet SAMPling (PSAMP), http://www.ietf.org/html.charters/psamp-charter.html
22. Schuehler, D.V., Lockwood, J.W.: A Modular System for FPGA-based TCP Flow Processing in High-Speed Networks. In: Becker, J., Platzner, M., Vernalde, S. (eds.) FPL 2004. LNCS, vol. 3203, pp. 301–310. Springer, Heidelberg (2004)
23. Zseby, T., Zander, S., Carle, G.: Evaluation of Building Blocks for Passive One-way-delay Measurements. In: Proceedings of Passive and Active Measurement Workshop (PAM) (2001)