Enabling Autonomic Access Network QoE Management through TCP Connection Monitoring

Bart De Vleeschauwer, Wim Van de Meerssche, Pieter Simoens, Filip De Turck, Bart Dhoedt and Piet Demeester
Ghent University - IBBT - IMEC, Department of Information Technology
Gaston Crommenlaan 8 bus 201, 9050 Gent, Belgium
Tel: +3293314900, Fax: +3293314899
Email: bart.devleeschauwer@intec.ugent.be

Kris Struyve, Tom Van Caenegem, Edith Gilon, Hans Dequeker, Erwin Six
Alcatel-Lucent Research & Innovation
Copernicuslaan 50, B-2018 Antwerpen, Belgium

Abstract— **The experienced quality of services such as IPTV, video on demand, online gaming and VoIP is very sensitive to packet loss, delay and jitter. Therefore it is an essential requirement that access network providers are able to monitor and control with rather high resolution the end-to-end service delivery paths. Because this task is very complicated we are developing an autonomic framework for quality of experience (QoE) management. A two layer architecture is proposed, consisting of a monitor plane and a knowledge plane. The first is responsible for providing the available monitor information, the latter uses that information to determine the location and cause of a QoE problem and determines a fitting solution. The first step in autonomicly managing service QoE is detecting when a problem occurs. In this paper we introduce ANTMA, the Access Node TCP Monitoring Algorithm. It monitors TCP connections by analyzing the packets that pass an intermediate point, such as an access network element or a residential gateway, thereby allowing fault isolation. We provide a detailed analysis of how ANTMA works and also present a thorough evaluation of its behavior. The algorithm is an essential component in enabling autonomic QoE management.**

I. Introduction

Providing QoE guarantees is becoming increasingly important in the delivery of multimedia services to residential users. The term QoE refers to the quality of a service as it is perceived by the end-user. QoE is highly sensitive to packet loss, jitter and high delay. Therefore, monitoring the network is crucial in determining the QoE, finding the cause of QoE problems and finding actions that are able to restore it.

The access network today is no longer the "last" or final mile as more and more home networks are deployed. Both play a pivotal role in the end-to-end connection. Both the xDSL loop and some home network technologies are known to sometimes introduce packet loss. It is of prime importance that the service provider can at least determine if the cause of a decrease in QoE is located in the access network or in the home network of the client. In order to manage the QoE in an efficient way, a layered architectural framework provides a good solution. This framework consists of two interacting layers, the monitor plane and the knowledge plane. The former is responsible for providing the required monitor information, the latter takes care of analyzing the monitor data, root cause analysis and finding solutions. We introduced this access network QoE management framework in greater detail in [1] and [2].

An essential challenge in QoE management is detecting when a problem arises and pinpointing its location. For this end, we are developing monitor probes that can be placed at various locations in the access network and that enable to do this. As one cannot assume full control over the end-device (set-top box, desktop PC,...), our monitor probes are located at an intermediate point such as an access network element or a residential gateway.

The TCP protocol (Transmission Control Protocol) is one of the most used protocols in providing multimedia services, such as video on demand to residential users. Therefore, TCP connection monitoring is an important building block in the monitor plane. We have developed the ANTMA algorithm, which stands for Access Network TCP Monitoring Algorithm. It observes all the packets that pass through the intermediate point and uses the information in these packets to deduce the packet loss, delay and jitter between the intermediate point and the end device that presents the service to the user. This allows us to determine whether a decrease in QoE is caused between the server and the monitor point or between the monitor point and the end device. As a result, the access network provider can determine whether he is responsible for the QoE decrease or not.

This research is part of a broader and more comprehensive vision on future service-aware access networks that is under study in the integrated research project MUSE [3]. The overall goal of the MUSE project is the research and development of a future, multi-service broadband access network. MUSE is amongst others studying the QoE requirements and architecture for various services. The work reported in this paper shows how these requirements could be implemented in the access networks.

This paper is structured as follows: In section II we give

an architectural overview of a two layered QoE management architecture. Section III describes the ANTMA algorithm and in section IV we describe work related to this paper. In section V we provide an evaluation of the ANTMA algorithm. Finally, conclusions are drawn in section VI.

II. Autonomic Access Network Architecture

The autonomic access network consists of two logical layers or "planes", spanning the whole access network from service edge router up to and including the end-device. The first layer, the Knowledge Plane (KPlane), analyzes the QoE of all running services and can restore degraded services autonomicly by determining and executing the appropriate actions. The KPlane reasons on an extensive knowledge base with monitor data on the network status. This data is provided by the layer beneath, the Monitor Plane (MPlane). It monitors the status of the network and devices, as well as the QoE of the running services. Each layer consists of a central component and distributed entities in the different parts of the access network. A logical view on the double-layered architecture is presented in Figure 1. The architecture was presented in great detail in [1], [2]. We will present the aspects relevant to the scope of this paper.

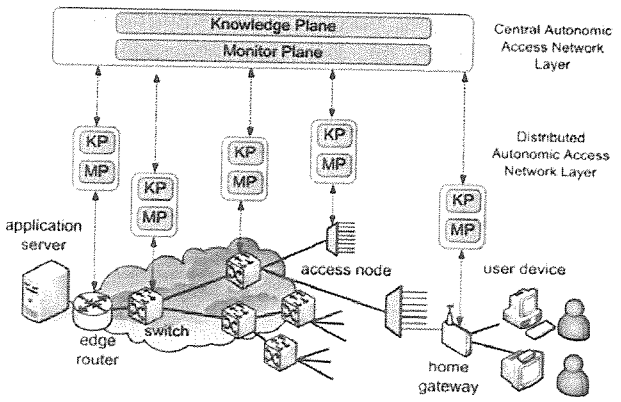


Fig. 1. An overview of the MPlane-KPlane architecture with distributed and central components.

The KPlane guarantees the QoE for all running services. It analyzes the huge amount of monitor data generated by the MPlane in order to detect QoE degradations. A wide range of techniques can be used to give a decisive answer about the root cause of the reported problem. Applying temporal and spatial correlation techniques on the monitor data can give a first indication. If several users are reporting QoE degradations for the same service, the root cause will most likely be situated in the access network. A more accurate determination can be done by reasoning on an ontology containing all the network components and their relationships. In a last step, the KPlane will undertake the appropriate QoE restoring actions, depending on the affected service and the exact nature of the problem cause. Actions can involve the application of interleaving or forward error correction on a videostream, a rerouting of the traffic or changing the priorities of the different traffic classes.

The MPlane provides a detailed view on the network status by monitoring the services, switches, routers and devices, as well as their interconnecting links. The Simple Network Monitoring Protocol (SNMP) can be used for monitoring the switches and allows communication with a central managing device. IPFIX is another technique to monitor switches and edge routers by tracking individual flows. Information on the quality of the access line between home gateway and access node can be retrieved e.g. from the gateway by using the TR-069 [4] interface via an Auto Configuration Server. This huge amount of monitor data is structured in specific data structures that allow efficient processing by the KPlane. All the information is exported via an API to the KPlane. Via this API, the KPlane activates and deactivates monitor probes and manages their configuration.

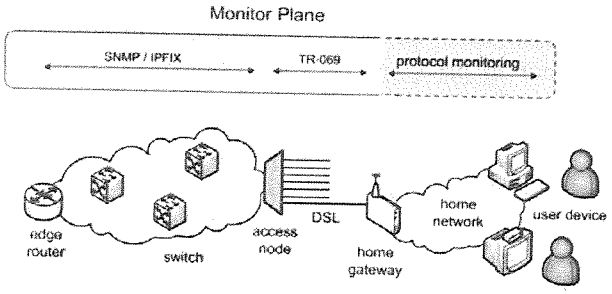


Fig. 2. Protocol monitoring on the access node or home gateway allows to monitor the home network and the end devices.

Due to firewall and NAT mechanisms and the wide range of user devices, it is difficult to install monitoring tools in the home network. Some protocols have two-way traffic, with the client sending acknowledgments (e.g. TCP) or statistical reports (e.g. RTP/RTCP) about the received data. By sniffing both the up- and downstream packets on an intermediary point like the access node or the home gateway, the QoE of the connection as perceived by the client can be derived. Also information about the home network status can be obtained. This is illustrated in Figure 2. Since the monitoring algorithms do not rely on the device specifications, they can be applied to all devices adhering to the protocol specifications. This approach requires a specific algorithm for each protocol. In practice, the number of protocols in use is limited.

III. The ANTMA algorithm

In this section, we introduce ANTMA, the "Access Network TCP Monitoring Algorithm". ANTMA passively monitors packets at an intermediate node in the access network (typically the access node (AN) or residential gateway (RGW)). ANTMA is focused on the part of the network downstream from the intermediate node (the network between the monitoring point and the receiver). ANTMA's main function is calculating upper and lower limits for downstream and upstream loss by matching data and acknowledgment packets (abbreviated DP and ACK). By matching DP and ACK packets, the algorithm is also able to estimate RTT (round-trip time, or two-way delay) and is able to derive the RTT jitter.

A. Scope of the algorithm

The ANTMA algorithm is designed specifically for access network monitoring. In this scenario we can make assertions that do not hold in an intermediate internet router, but they guarantee the correctness and accuracy of ANTMA's measurements. They are:

- All packets in both directions (i.e. all data packets and all acknowledgment packets) of the TCP connection are seen at the measurement point.
- There is no re-ordering of packets possible in the network between the measurement point and the receiver of data (the downstream side of the measurement point). This is the most important assumption made by the algorithm. Packet reordering is typically caused when packets take different routes (as with load-balancing) to get to their destination, or by internal buffers in complex high-performance routers. None of these situations normally occur in the home network. Common technologies in the home network, like switches, firewall, routers and wireless links, do not cause packet reordering. Any re-ordering that does occur in the home network, results in accuracy loss. Frequent reordering can deteriorate ANTMAs accuracy dramatically. Packet re-ordering on the upstream side of the measurement point however, does not affect ANTMA's applicability.

B. Overview

ANTMA monitors the packets of a TCP connection at an intermediate point. The most important TCP header fields used by the algorithm are the sequence number, acknowledgment number and data length. These are abbreviated by "seqNR", "ackNR" and "LEN" respectively. ANTMA monitors only downstream data-packets, ignoring their ackNR, and upstream ACKs, ignoring their seqNR. By keeping track of these values, the algorithm is able to estimate the packet loss between the monitor point and the end device. The core idea behind ANTMA is that it tries to "match" data and acknowledgment packets. This means that it tries to uniquely identify the data packet that caused the client to send out an acknowledgment packet. While doing this, lost packets are detected. A simple example of this is shown in Figure 3.

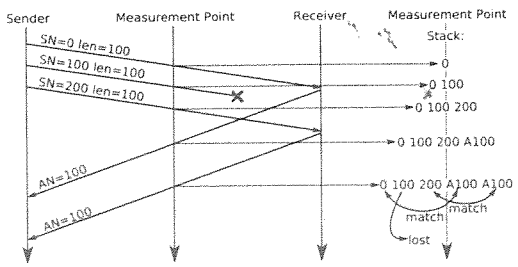


Fig. 3. Packet matching & loss detection example

The ANTMA algorithm is easily modeled as a state machine. Packets seen at the monitoring point form the input, and loss measurements are the output. Each sniffed packet changes the state, and subsequently a number of specified rules transform the new state to a reduced state. The state is a stack of entries corresponding to packet header information and annotations. For each packet that arrives, an entry is pushed on the stack. The rules then apply, and transform the annotations, as well as remove packet entries from the stack. During this removal, loss and RTT estimations are generated. Each rule can only be applied when certain conditions are met. The rules keep being tried and applied, until no rules can be applied anymore.

1) Packet annotations: For every packet entry on the stack some information is maintained. Below we will present the notation that will be used for the more detailed description of the rules in the remainder of the paper.

Annotations common for all packets are the TCP fields, such as ackNR, seqNR and LEN, respectively denoted by: P.ACKNR, P.SEQNR and P.LEN.

The order and time of arrival of the packets at the monitor probe are important, so the packet arrival time is annotated. Packet arrival time is denoted by P.TIME. This time is used for RTT calculation, while the order in which packets arrive is very important for almost every rule.

The annotations specific for a DP are:

- DP.MAXAN: the maximum ackNR of the ACK that can be triggered by this DP. This annotation is set by the "NewDP" rule.
- DP.CR: (Certainly Received), this annotation is set by the CR1 & CR2 rules and is set when it is certain that this packet was received by the client.
- DP.RETRANS: (Retransmit), this annotation is set for DPs, when a packet with the same seqNR was already seen. Rule "NewDP" sets this annotation.

ACK-specific annotations are:

- ACK.PM: a list of possible matches. This list is the most important annotation used in ANTMA. It contains the data packets that could have triggered the client to send this ACK. The "NewACK" rules initializes it, while the M1-M7 and "Match Update" rules reduce this list. It is used in matching a DP and an ACK.
- ACK.CAUSED: This annotation is set by the "NewACK" rule. It reflects whether the ACK was sent as a reaction by the receiver because it just received a data packet or whether it is not known why the ACK was sent. In this case, possible reasons next to DP receipt are a TCP window update or a receiver sending data. Note that when ACK-caused is false, it means "Possibly uncaused", and not caused.

2) ANTMA rule overview: There are two rules that add packets to the stack:

- The "NewDP" rule is called when a new data packet arrives. It initializes the annotation for this packet and pushes it on the stack.
- The "NewACK" rule is called when a new acknowledgment packet is seen at the monitor point. It initializes the annotation and pushes it on the stack.

There are four rules which remove packets from the stack.

- The "Match"-rule removes a pair of a uniquely matched data and acknowledgment packet. This means that the ACK packet was sent when the DP arrived at the client.
- The "Match Cleanup"-rule runs after a match and removes data and acknowledgment packets that will no longer undergo annotation changes.
- The "Disjunct detection"-rule removes packets for which no further annotation changes will occur.
- The "Certainly Lost"-rule detects when a data packet has been lost between the monitor point and the client and remove these packets from the stack.

Rules which manipulate annotations are:

- The CR1 & CR2 rules detect whether a packet is certainly received by the client or not.
- The M1-M7 rules remove data packets from the "possible matches"-annotation of ACKs
- The "Match update" rule removes data packets from the "possible matches"-annotation of ACKs, after the match rule ran.

The "Count Algorithm" runs when rules "Match Cleanup" or "Disjunct detection" remove packets. It examines the annotations, and estimates the minimum and maximum loss that occurred. Note that the "Certainly Lost" rule also adds to the estimation by reporting exact loss.

Figure 4 gives an overview of the algorithm, more details can be found in the following sections.

C. Detailed description

In this section we will focus on the composing elements of ANTMA. It contains a detailed description of the rules and algorithms that together constitute the algorithm.

1) "NewDP"-rule: The "NewDP"-rule pushes a new data packet on the stack and initializes its "MaxAN" and "Retransmit" annotations. The maximum ackNR will be the seqNR + LENof the last packet up to which we have seen all TCP data bytes. The "Retransmit" annotation is set when a packet with the same sequence number has been seen before.

2) "NewACK"-rule: The "NewACK"-rule initializes both ACK annotations. The "Caused" annotation means that the ACK is certainly sent as reaction to receiving a DP, and is set if one of the following conditions is met:

- The ackNR of the ACK is different than the ackNR of the previous ACK seen.
- The ACK contains no data, and the window size of the ACK is not greater than that of the previous ACK seen.

Otherwise, the "Possibly uncaused" annotation is set. "Possibly uncaused" ACKs are either upstream data packets or TCP window updates.

The possible matches list is initially filled with all data packets which are on the stack when the ACK arrives.

3) "Match"-rule: When a "Caused" ACK is on the stack and has a possible matches list with exactly 1 element the match rule is activated. In this case, the ACK is sent as a reaction to this data packet.

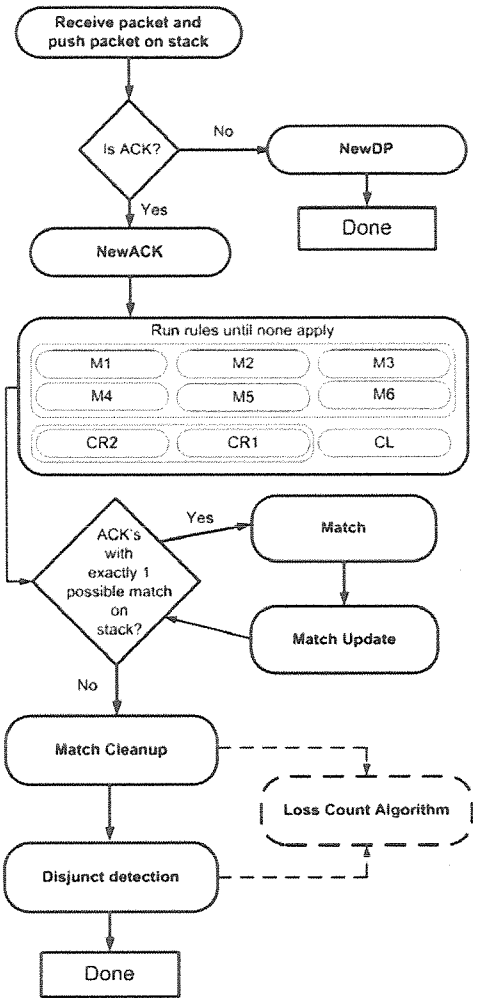


Fig. 4. Algorithm Overview

For a "Possibly uncaused" ACK to match, there are additional conditions. The "Possibly uncaused" ACK's previous ACK must also be on the stack, and it must not include the match of the "Possibly uncaused" ACK in its matches list. For matching packets, the rule removes both the ACK and the DP from the stack. It then triggers the "Match Update" rule. The "Match cleanup"-rule is triggered in a special way as well (for details, see the "Match cleanup"-rule's description) Finally, it calculates the RTT, of the matched pair ($DP.\text{TIME} - ACK.\text{TIME}$), and reports it.

4) "Match Update"-rule: After each "Match"-rule execution, we can update all the previous ACKs possible matches lists, by removing all data packets later than the matched data packet from their list of possibilities. All the next ACKs possible matches are also updated, by removing all data packets earlier than the matched data packet from their list. These decisions can be made because there is no reordering between the end device and the monitor point.

Algorithm III.1: MATCHUPDATE(A_m, D_m)

for each $A \in stack$
do for each $D \in A$.PM
do **if** A.TIME $< A_m$.TIME **and** D.TIME $> D_m$.TIME **or** A.TIME $> A_m$.TIME **and** D.TIME $< D_m$.TIME **then** A.PM $\leftarrow A$.PM $\setminus \{D\}$

5) "Match cleanup"-rule: The match cleanup rule is triggered in a special way, it only runs after the last "Match". This means that all rules are allowed to run, and when no rule changes the stack anymore, the "Match cleanup"-rule will run if the "Match"-rule has at least run once. This rule will remove all data packets earlier than the last matched data packet from the stack, and remove all ACKs earlier than the last matched ACK from the stack. All packets removed in this way will be sent to the "loss-count" algorithm that will determine the packet loss that has occurred.

Algorithm III.2: MATCHCLEANUP(A_m, D_m)

for each $A \in stack$
do **if** A.TIME $< A_m$.TIME **then** REMOVE(A)
for each $D \in stack$
do **if** D.TIME $< D_m$.TIME **then** REMOVE(D)

6) "Disjunct detection"-rule: This rule is triggered when the first possible match of an ACK is later than the previous ACK. In that case, it is known that there will never be any more updates of the possible matches lists of all packets received before this first possible match. Those packets are removed and sent to the "loss-count" algorithm.

Algorithm III.3: DISJUNCTCLEANUP(A_1, A_2)

if $A_2 =$ NEXTACK(A_1)
then $D_2^{first} \leftarrow$ FIRST(A_2.PM)
if A_1.TIME $< D_2^{first}$.TIME
then **for each** $P \in stack$
do **if** P.TIME $< D_2^{first}$.TIME **then** REMOVE(P)

This rule typically only needs to run once after all rules have stopped running, and only for the last ACK that has arrived.

7) Annotation rules: This set of rules use the annotations and influence the possible matches list for the ACKs on the stack. There are 6 rules that work in this way:

- Rule M1: A data packet cannot cause an ACK with an ackNR equal to its seqNR.

  Algorithm III.4: CHECKRULEM1(A)

  for each $D \in A$.PM
  do **if** D.SEQNR $= A$.ACKNR **then** A.PM $\leftarrow A$.PM $\setminus \{D\}$

- Rule M2: Consider an ACK, and an earlier "caused ACK". Only data packets coming after the earliest possible match of the earliest ACK can have caused the latest ACK.

  Algorithm III.5: CHECKRULEM2(A_1, A_2)

  if A_1.TIME $< A_2$.TIME **and** A_1.CAUSED
  then $D_1 \leftarrow$ FIRST(A_1.PM)
  for each $D_2 \in A_2$.PM
  do **if** D_1.TIME $\geq D_2$.TIME **then** A_2.PM $\leftarrow A_2$.PM $\setminus \{D_2\}$

- Rule M3: A data packet with a "maximum ackNR" annotation lower than an ACKs ackNR cannot have caused it.

  Algorithm III.6: CHECKRULEM3(A)

  for each $D \in A$.PM
  do **if** D.MAXAN $< A$.ACKNR **then** A.PM $\leftarrow A$.PM $\setminus \{D\}$

- Rule M4: Consider a "Caused" ACK, and an earlier ACK. The earlier ACK cannot match the last possible match of the "Caused" ACK, or any DP later than this last possible match.

  Algorithm III.7: CHECKRULEM4(A_1, A_2)

  if A_1.TIME $< A_2$.TIME **and** A_2.CAUSED
  then $D_2 \leftarrow$ LAST(A_2.PM)
  for each $D_1 \in A_1$.PM
  do **if** D_1.TIME $\geq D_2$.TIME **then** A_1.PM $\leftarrow A_1$.PM $\setminus \{D_1\}$

- Rule M5: Consider each pair of an ACK, and the next (so later) ACK that is received. The first ACK can never have a data packet in its possible matches list, if the seqNR of that packet is equal to the ackNR of the last ACK.

  Algorithm III.8: CHECKRULEM5(A_1, A_2)

  if A_1.TIME $< A_2$.TIME
  then **for each** $D_1 \in A_1$.PM
  do **if** D_1.SEQNR $= A_2$.ACKNR **then** A_1.PM $\leftarrow A_1$.PM $\setminus \{D_1\}$

- Rule M6: Consider a DP that is "Certainly Received", and an ACK that is later. If the ackNR of the ACK is equal to the seqNR of the DP, the ACK cannot match any packet that is later than the DP

  Algorithm III.9: CHECKRULEM6(D, A)

  if D.TIME $< A$.TIME **and** A.ACKNR $= D$.SEQNR
  then **for each** $D_A \in A$.PM
  do **if** D_A.TIME $> D$.TIME **then** A.PM $\leftarrow A$.PM $\setminus \{D_A\}$

8) "Certainly Received"-rules: Rule CR1: If a "Caused" ACK is received, then all data packets of which there is no retransmit on the stack earlier than the last possible match of the ACK are certainly received if their seqNR is lower than the ACKs ackNR.

Algorithm III.10: CHECKRULECR1(A)

if A.CAUSED
then
- $D_{last} \leftarrow$ LAST(A.PM)
- **for each** $D \in$ BEFORE(A)
- **do**
  - **if** D.SEQNR $<$ A.ACKNR **and**
    D.TIME $<$ D_{last}.TIME **and**
    NOTRETRANSMITTED(D)
  - **then** D.CR $\leftarrow$ *true*

Rule CR2: If a "Caused" ACK's next "Caused ACK" has a bigger ackNR, examine the range between the first possible match of the first ACK, and the last possible match of the second ACK (including the first and last data packets). We know there is at least one data packet received, in the examined range, with a seqNR equal to the first ACK's ackNR. So if there is only 1 such data packet between the 2 points, it is certainly received.

Algorithm III.11: CHECKRULECR2(A_1, A_2)

if A_1.CAUSED **and**
A_2.CAUSED **and**
A_1.TIME $<$ A_2.TIME
then
- $D_1^{first} \leftarrow$ FIRST(A_1.PM)
- $D_2^{last} \leftarrow$ LAST(A_2.PM)
- $count \leftarrow 0$
- **for each** $D \in stack$
- **do**
  - **if** D.TIME $\geq$ D_1^{first}.TIME **and**
    D.TIME $\leq$ D_2^{last}.TIME **and**
    D.ACKNR $=$ A_1.ACKNR
  - **then**
    - $count \leftarrow count + 1$
    - $D_u \leftarrow D$
- **if** $help = 1$
  - **then** D_u.CR $\leftarrow$ *true*

This rule is also valid when the first ACK has been removed from the stack.

9) "Certainly Lost"-rule: Rule CL: If a "Caused ACK" is received, all data packets with a seqNR equal to its ackNR, and earlier than its first possible match, are certainly not received.

Algorithm III.12: CHECKRULECL(A)

if A.CAUSED
then
- $D_{first} \leftarrow$ FIRST(A.PM)
- **for each** $D \in$ BEFORE(A)
- **do**
  - **if** D.SEQNR $=$ A.ACKNR **and**
    D.TIME $<$ D_{first}.TIME
  - **then** CLEANCL(D)

Packets that are certainly lost are removed from the stack and from all possible matches lists. Certain data packet loss, are added to the total loss counters. They increase both the minimum and maximum data packet loss.

10) "Loss Count"-algorithm: The "Match Cleanup" and "Disjunct detection" rules hand a sequence of packets over to the loss detection algorithm. This algorithm analyzes the packets and determines boundaries for the packet loss for both data and acknowledgment packets. Together with the results from the certainly lost rule these values add up to the total result of the algorithm. When it receives the packets it looks at the annotations and determines the following values:

- The number of ACKs: A
- The number of data packets: D
- The number of data-packets that are annotated "Certainly Received": C

Base on these values, the algorithm will determine the lower and upper bounds for the following values:

- The amount of the data-packets that were lost before they reached the client, but passed on the measurement point (DL):

$$DL \in [0, D - max(A, C)] \quad (1)$$

- The amount of ACKs that are lost on their way to the measurement point. (AL)

$$AL \in [max(0, C - A), D - A] \quad (2)$$

This algorithm works well for data packets. However, when delayed acknowledgments occur, they are detected as loss of acknowledgments. This doesn't affect the data packet loss estimation, but renders the ACK loss estimation useless. This problem could possibly partially be avoided by extending the algorithm to take the TCP timestamp option into account (which can help in detecting delayed ACKs). However, this option is not always present in TCP.

D. RTT

The previous section discussed how the ANTMA algorithm is able to estimate the packet loss that has occurred between the monitor device and the client. However, because it matches packets and their replies, ANTMA can also measure RTT between middle-point and receiver (this is a measurement, not an estimation). For such RTT measurements, ANTMA is not required, as simpler algorithms than ANTMA can more easily uniquely match data packets with their ACKs. These simplified matching algorithms will be sufficient for a basic RTT estimation. When jitter and loss occur, ANTMA will be able to match more packets than simpler algorithms. This results in more measurements at these critical moments, which should make the RTT estimates more precise and increase the usefulness for detecting jitter.

IV. RELATED WORK

Several methods already exist for monitoring loss, RTT, jitter and various TCP related metrics. Each method has its own monitoring position, either on receiver or sender, or at some intermediate point, where it requires data from one or

both directions of the TCP connection. Some of the methods that monitor from an intermediate point make an end-to-end estimation, while others make estimations for each part of the network (before and after the monitoring point).

Monitoring RTT between client and server in an intermediate point is a non-trivial problem. The authors of [5] outline a simple method that measures RTT in the 3-way handshake and slow-start phase, at the beginning of every TCP connection. In [6] this method is expanded, and the "frequency algorithm" is introduced. This algorithm measures RTT continuously, by using only the packets going from sender to client. It is based on the fact that inter-packet times are approximately periodic, with a period roughly equal to the RTT. In [7] 2 methods for measuring end-to-end RTT from an intermediary node are examined. The first is based on matching data packets and the acks that caused them by using the TCP timestamp method, while the second is based on detecting patterns in the packet timing.

The authors of [8] propose a receiver-side statistical estimation method, which tries to match DPs with the ACK that caused the sender to send them. This is the opposite of ANTMA, which matches ACKs with the DP that caused them. The algorithm uses TCP congestion algorithm information. It estimates several TCP specific metrics, such as *cwnd*, *ssthresh* and the type of TCP implementation.

End-to-end loss detection is easy at the sender or receiver, but loss detection in an intermediate point is more challenging. In [9] an algorithm is propose, which discards suspicious retransmissions to estimate a loss rate for both parts of the network.

V. ANTMA Evaluation

A. Setup

To evaluate ANTMA, we used a testbed consisting of 5 active devices, over which a video file was streamed using TCP. There are two PCs running click modular router software [10] to introduce network anomalies like packet loss and jitter. Figure 5 shows an overview of the test setup. It consists of:

- A server, running an apache HTTP server.
- A click router which emulates loss, delay and jitter in the network upstream from the monitoring probe.
- A router, where the ANTMA monitoring algorithm runs.
- Another click router for emulating loss, delay and jitter in the network downstream from the monitoring probe.
- A client, running VideoLAN software.

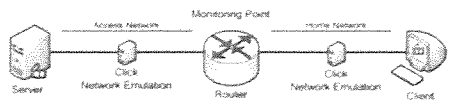


Fig. 5. Test Network Setup

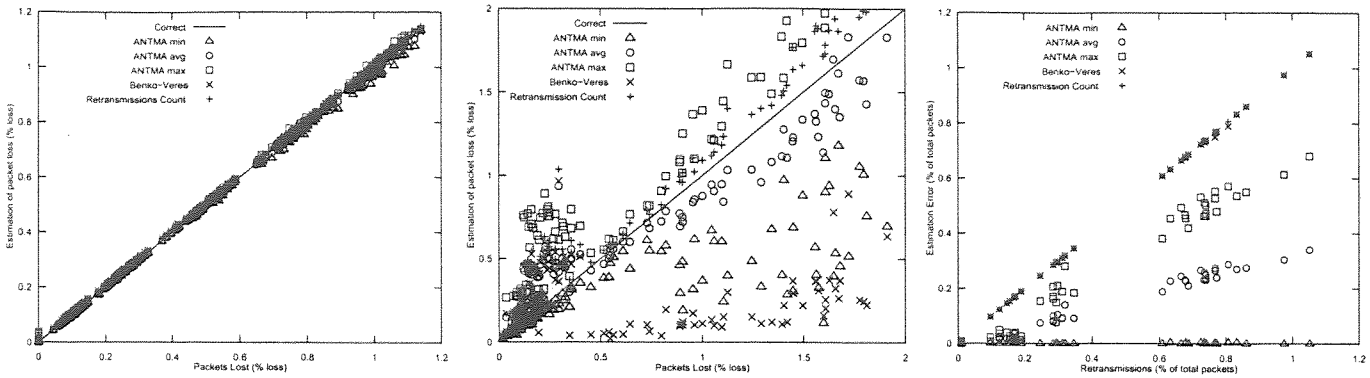
This testbed maps to a real IPTV deployment with e.g. a VoD server and a client node. The monitoring algorithm can run on an intermediary point like the access node or the residential gateway. The downstream impairment node emulates the loss introduced on the xDSL link and the home network.

Different scenarios with respect to packet loss, extra delay and jitter at both sides of the monitoring probe were emulated using the impairment nodes. For each network impairment configuration, 10 tests were performed, to eliminate random factors introduced by random loss combined with TCP's behavior. Our test movie was 128 seconds long and had a size of 33 MB (±2MBit/sec). During the tests we captured all packets at the middle-point and at the client. These traces were used in order to determine the exact loss. The traces captured on the middle node also served as input for the ANTMA monitoring algorithm and for two alternative monitoring algorithms. For ANTMA, the minimum and maximum loss estimations are shown. We also indicate the average of these two values on the graph. The first alternative algorithm simply uses the amount of TCP retransmissions as an estimation for loss. The second alternative is the algorithm described in [9] and is referred to as Benko-Veres in the remainder of this paper. This algorithm is based on counting retransmissions, but includes logic to eliminate retransmissions due to jitter when assessing the experienced packet loss on the downstream link. The obtained results are presented in fig. 6 and figs. 7. In the graphs of fig. 6, each point represents the result of applying one algorithm for one specific test. Fig. 7 contains the relative difference of the packet loss that the algorithms detected and the actual packet loss that occurred, averaged over all the tests for one scenario.

B. Loss

In the first scenario random packet loss was applied either downstream or upstream from the probe, or on both parts of the network. Each packet has the same chance of being dropped, independent of the previous or next packets. The packet loss ratios were 0.1%, 0.25%, 0.5%, 0.75% and 1%, while a delay of 10ms, 25ms, 50ms and 100ms was used. Figure 6(a) shows that all algorithms are able to estimate accurately the packet loss between the monitor point and the client node. In addition to this, loss in the upstream network segment has no influence on their accuracy. In fig. 7(a), one can see that the three ANTMA estimates outperform the two alternative algorithms. The standard deviation in this graph was 1.83, 1.56, 1.68, 3.35 and 3.29, for ANTMAmin, ANTMAavg, ANTMAmax, counting retransmissions and the Benko-Veres algorithm. This test proves that for random packet loss, the ANTMA algorithm outperforms alternative algorithms and is able to give an accurate estimate of the actual packet loss ratio that occurs.

In a second scenario, we emulated home network packet loss according to typical xDSL loss characteristics, where the link downstream of the probe represents the xDSL link. In this model, packets are dropped periodically for certain amounts of time, generating a bursty loss pattern. This corresponds to the typical packet loss caused by xDSL noise. The results are shown in fig. 6(b) and fig.7(b). The graphs show that the ANTMA algorithm significantly outperforms the Benko-Veres algorithm which underestimates the packet loss with

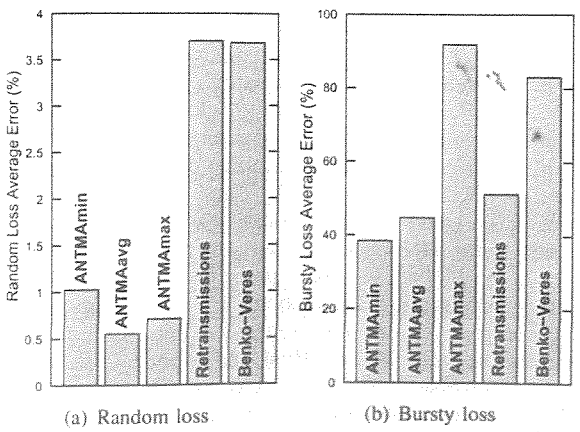


(a) Random loss (b) Bursty loss (c) Jitter and reordering

Fig. 6. Detected packet loss ratio for the algorithms for different types of network impairment. The x-axis shows the actual packet loss for graphs 6(a) and 6(b) and the number of TCP retransmissions for graph 6(c), since no packet loss was present in this case.

its minimal and average packet loss estimates. The ANTMA-max estimate however, has the worst performance. Counting retransmissions also performs well in this scenario. As can be seen in fig. 6(b), the standard deviation in this test was rather high, with values of 29.34, 51.27, 110.44, 62.66 and 44.04 for ANTMAmin, ANTMAavg, ANTMAmax, counting retransmissions and the Benko-Veres algorithm.

In a third scenario we applied a specific type of jitter to the packets flowing downstream from the probe, but no packet loss. This jitter is created by delaying all packets 10 milliseconds, selecting 0.5% of the packets, and delaying those for an additional time, ranging from 1 to 25 milliseconds. This causes reordering, but so infrequent that TCP's RTO timer will not adapt sufficiently, resulting in needless retransmissions. In Figure 6(c) the estimated packet loss for the 3 algorithms is shown versus the number of retransmissions which took place as a result of the imposed jitter pattern. We see that both the Benko-Veres and Retransmission-count algorithms fail, as they both interpret the retransmissions as loss. ANTMA performs better. Its minimum estimations are matching the real loss values (being zero), while its maximum and average estimations are more accurate compared to the alternative algorithms.



(a) Random loss (b) Bursty loss

Fig. 7. The average accuracy for the algorithms in a scenario with random loss and bursty xDSL like packet loss.

VI. Conclusion

In this paper we have presented an algorithm for monitoring TCP connections from an intermediate point and derive information on the segment between the monitor point and the end-device that represents the service to the user. The algorithm is able to determine exact boundaries for both the minimum and maximum packet loss that is experienced by a TCP connection. The results indicate that the average of these values and the minimal loss value of the algorithm give a good estimate of the actual packet loss that has occurred for random packet loss and that ANTMA outperforms other algorithms in more complicated loss scenarios. In the end, the ANTMA algorithm will provide an essential element in a broader architecture for autonomic access network QoE management, since it can be used as a trigger for performing QoE restorative action.

VII. Acknowledgements

The research is funded by a PhD Grant and a post-doctoral grant of the Fund for Scientific Research (FWO-V) for Pieter Simoens and Filip De Turck respectively.

References

[1] P. Simoens, B. De Vleeschauwer, W. Van de Meerssche, F. De Turck, B. Dhoedt, P. Demeester, E. Gilon, K. Struyve, and T. Van Caenegem. Towards Autonomic Access Networks for Service QoE Optimization. In *Modelling Autonomic Communications Environments, First IEEE International Workshop, MACE2006*, pages 223–233, 2006.

[2] B. De Vleeschauwer, W. Van de Meerssche, P. Simoens, F. De Turck, B. Dhoedt, P. Demeester, E. Gilon, K. Struyve, and T. Van Caenegem. On the Enhancement of QoE for IPTV Services through Knowledge Plane Deployment. In *Broadband Europe*, 2006.

[3] Multi service access everywhere. [online] http://www.ist-muse.org.

[4] DSLForum. Technical Report-069: CPE Management Protocol, 2004.

[5] H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times, 2002.

[6] Ryan Lance and Ian Frommer. Round-trip time inference via passive monitoring. *SIGMETRICS Perform. Eval. Rev.*, 33(3):32–38, 2005.

[7] Bryan Veal, Kang Li, and David K. Lowenthal. New methods for passive estimation of tcp round-trip times. In *PAM*, pages 121–134, 2005.

[8] Guohan Lu and Xing Li. On the correspondency between tcp acknowledgment packet and data packet. In *IMC '03: Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 259–272, New York, NY, USA, 2003. ACM Press.

[9] P. Benko and A. Veres. A passive method for estimating end-to-end tcp packet loss, 2002.

[10] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Trans. Comput. Syst.*, 18(3):263–297, 2000.