REGRESSIVE ADMISSION CONTROL ENABLED BY REAL-TIME QOS MEASUREMENTS

Mirjami Jutila[1], Jarmo Prokkola[2] and Despina Triantafyllidou[3]

[1]VTT Technical Research Centre of Finland ,Kaitoväylä 1, 90571 Oulu, Finland

ABSTRACT

We propose a novel regressive principle to Admission Control (AC) assisted by real-time passive QoS monitoring. This measurement-based AC scheme accepts flows by default, but based on the changes in the network QoS, it makes regressive decisions on the possible flow rejection, thus bringing cognition to the network path. The REgressive Admission Control (REAC) system consists of three modules performing the necessary tasks: QoS measurements, traffic identification, and the actual AC decision making and flow control. There are two major advantages with this new scheme; (i) significant optimization of the connection start-up phase, and (ii) continuous QoS knowledge of the accepted streams. In fact, the latter combined with the REAC decisions can enable guaranteed QoS without requiring any QoS support from the network. REAC was tested on a video streaming test bed and proved to have a timely and realistic match between the network's QoS and the video quality.

KEYWORDS

REAC, Passive Measurements, Network Monitoring, Admission Control, Cognitive Network

1. INTRODUCTION

The current IP networks operate mostly on a best effort (BE) basis, where the communications quality of service (QoS) is not guaranteed. These networks typically operate well, if the network load is clearly below the network capacity. However, when operating near the network capacity, or in a congested state, all the users in the network suffer. This means that, even single user's high-rate traffic flow can deteriorate the QoS for all the users in a network. On the other hand, provisioning resources for bursty traffic, leads to over-provisioning and under-utilization of the capacity.

There are various mechanisms developed for fixed and wireless networks to improve QoS including scheduling, queue management , QoS-based routing, and admission control (AC). Some of them can even guarantee QoS, but efficient resource management is a way more delicate task. Key components for the effectiveness of a QoS scheme are the end-to-end negotiations and setup, the bandwidth estimations, the resource reservation, and eventually the degree to which the network resources are utilized. So the downside is that, these methods are complicated and require extra features in the network elements, and often extra signaling. Because of these reasons, full, end-to-end QoS mechanisms are rarely used in practice. The simpler QoS mechanisms based on prioritization, like Differentiated Services (Diff Serv) in the core network side, or IEEE 802.11e in the WLAN, can provide QoS for certain types of traffic flows. However, these methods provide only *statistic QoS*, not *guaranteed QoS*, i.e., if there appears too much high-priority traffic, the performance of all the users will collapse, similarly as in the BE-based networks. Since the Internet traffic load is continuously increasing, more quality problems are expected in the future. Ironically, the applications causing high loads are the multimedia, which

DOI : 10.5121/ijcnc.2013.5602

also have clear QoS requirements. The traditional data applications (file transfer, web surfing, email, etc.) are considerably more tolerant with changing network conditions. Thus, methods enabling guaranteed QoS are likely to be valuable in the near future.

The traditional methods for QoS and AC have not become general, mostly because they are either not providing guaranteed QoS, or they are computationally and operationally costly. We propose a new regressive AC scheme, in which all the flows are a priori accepted in the network, without any negotiations. In this sense, our work presents a novel approach among AC methods, making a straight comparison between different schemes difficult. This REgressive Admission Control (REAC) method keeps track of the QoS in the network path by continuous real-time passive QoS monitoring.In the case of QoS degradation, REAC's QoS classification is either mapped to the network's QoS classes, or just brute packet dropping is performed for the last arrived low priority flows, in order to keep most of the users satisfied.

The rest of the paper is structured as follows. Section 2 describes the idea of regression for AC and establishes the principles for the REAC design. Section 3 presents existing AC work and orients REAC to the global map of AC. Section 4 describes the REAC's demonstration architecture. Section 5 describes the test bed and the measurement setup. Measurement results are presented in section 6. Section 7 concludes our work, and gives future directions.

2. REGRESSIVE ADMISSION CONTROL

Taking a step ahead from the traditional view of QoS networks, this paper follows an *Everyman's right*[1] approach to resource control. According to this traditional Finnish legal concept, every user is allowed the free right to access the environment's resources without any prior permission. Having this advantage, the user is nevertheless obliged not to cause any damage or disturbance. The application of this principle to an AC design is proposed here for the first time, and is the novelty of this work. The result, i.e., REAC, is a cognitive system combining network monitoring, traffic identification, congestion control, resource handling, QoS, and intelligent decision making.

The aim is to limit the newest traffic flows in the network path in a way that the offered traffic load stays below the bounds that the network can handle. In this way, the users already enjoying a service in the network will have guaranteed QoS, and will be satisfied. If the new traffic flows endeavoring in to the network are likely to exceed the network capacity, they are treated differently than the flows already in the network. Normally, these extra flows would ruin the performance of all the multimedia users. Thus, what happens is that REAC does not even try to perform a complex end-to-end resource allocation, but instead, the edge routers individually make the decision for the flows,handling the network path as a black box. As REAC performs no reservation of resources, there is no wasted capacity with predefined traffic classes.

A high-level view of the REAC's functionality is presented in Figure 1, where the AC policy keeps track of the flows' QoS. For this, a continuous real-time QoS monitoring is enabled to the network path between the end points. The monitoring is passive, so it gives clear measures of how the application traffic is really performing over the network path, while at the same time the control overhead is kept low. If the controlled network path has a support for prioritization (e.g. Diff Serv), the REAC's QoS classification is mapped to the network's QoS classes, but if not or it is not known, just brute dropping (or selective dropping) is performed. In case of QoS degradation, the potential following actions consider the most recent flows with the lowest QoS class. REAC uses a traffic classifier module, capable of discovering the real-time flows requiring higher priority. Thus, this is also one solution to the fundamental problem of QoS, i.e., where to

perform the traffic classification. REAC can, of course, also act according to different, or even external, prioritization policies.

REAC provides a cognitive network management system. Its cognitive cycle includes:

- Information gathering : QoS and traffic type of flows,
- Decision making: Whether or not to act and how,
- Learning: Near history QoS behaviour trends,
- Actions: Locate and drop or lower the class of a flow.

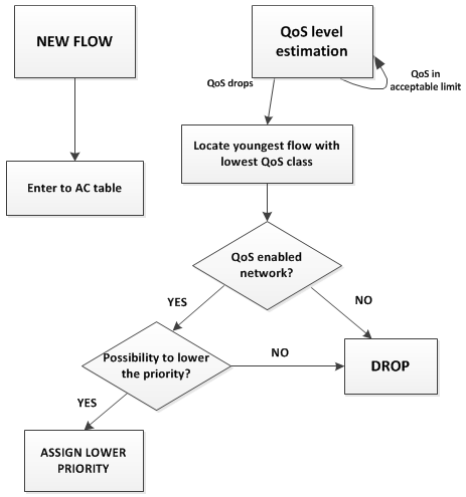


Figure 1.A high level view of theREAC process

Thus, with REAC, it is possible to maintain most of the users satisfied, i.e., to guarantee QoS for a number of flows. The users getting no service can try later again, and perhaps then, they will be let in to enjoy the guaranteed quality. This thinking is also similar to the well-known policy in POTS (Plain Old Telephone System), where users already in the system will enjoy good quality, while the new callers, in the case of congestion, are blocked. Restricting the number of flows, or even knowing the maximum number of flows, is not a trivial task in a packet switched network. This is where REAC shows its best advantage working in a regressive way: first try and then, if necessary, back off.In the worst case, this trying can cause very short-term congestion, nevertheless not to the point where the network would collapse. This is still a more gentle way than active capacity measurement, creating extra test traffic to the network, being used by many AC schemes.

3. RELATED WORK

As related work to REAC, the various methods for AC can be considered, while the topic has been widely studied since the 90's. This area has continued to gain attention and development, because of the network convergence and cognition. The new era in AC methods considers and combines many aspects of cognition both from network and user side i.e. flow fairness, Quality of Experience (QoE), and service level agreements (SLAs)[2], [3].

The AC methods can coarsely be classified as proactive (parameter based) or reactive (measurement based).In parameter-based admission control (PBAC)schemes, the flow admittance/denial is based on some analytical assessment of flows given a priori flow and system characteristics [4], [5]. PBAC's main drawback is that early flow characterization cannot always

be accurate, while the traffic models assumed are usually simplified and cannot capture the flow dynamics, as in the case of compressed video. Additionally, information of the system capabilities and e.g., the probability of buffer overflow are required, so the relationship between the traffic load and the queue length distribution has to be derived. This task is generally considered non-trivial, and a number of work in this area [6], [7] resort to a buffer-less router assumption, which is not realistic. The good side of PBAC schemes is that they can be analysed and compared using formal methods, whereas MBAC can only be quantitatively compared through experiments on simulated or real networks.

In the measurement-based admission control (MBAC), the feedback from the network can be collected with either active or passive monitoring[5], [8], [9], [10].MBAC measures typically the actual traffic load, to dynamically adjust the AC parameters. This is why MBAC controllers are more robust with respect to the accuracy of the traffic model, although they are compromised by the statistical variation of the measurements. When designing MBAC, the challenge is to calculate the *acceptance region* in a way that the network can tolerate traffic bursts. The peak rates have to be known or assumed; otherwise token bucket filters are necessary. It is then usual to identify maximal target utilization for resource reservation, whose apparent trade-off is the under-utilization of the capacity. Another disadvantage is that, since MBAC needs to be tuned depending on the network settings and the traffic scenario, it may give an excellent performance under one scenario, but an inadequate performance in another. To this problem a new MBAC solution in [11], is proposed by including a *Knowledge Plane* to the measurement algorithm, to maintain a broad view of the link behaviour, and predict the expected QoS to admit the flows to the network.In the method in [12] the rejected flows may decide to wait and try again after a given time-interval.

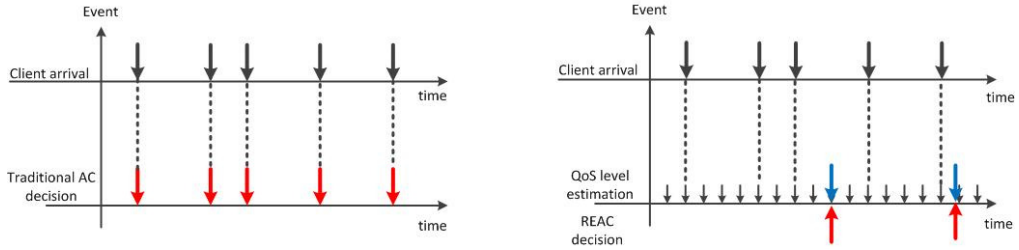
Hybrid approaches have also been developed to address the problems presented in parameter and measurement based methods. Examples include Experience Based Admission Control (EBAC) [13], and Measurement Based and a priori Traffic Descriptor Admission Control (MTAC) [14]. Both methods utilize measurements taken from the network, and knowledge of traffic descriptors to predict future bandwidth requirements. Recent hybrid approach [15] deploys Policy-Based Network Management (PBNM) techniques, resulting in an improved control and network utilization as a whole, thus eliminating the need to configure and manage each network entity separately.

Fairness is an issue typically not considered by AC, when traffic is managed into aggregates. Schemes that attempt to render a fair share of resources within the traffic aggregate e.g., [16]apply queue control functions and additional QoS router's state calculation, which increase the overhead considerably. In the latest work around fairness issue [12], an algorithm is developed that guarantees both fair occupancy and optimal usage of the resources.
Typically, many AC methods are coupled with Diff Serv[17], [18], which offers the scalability advantage of manipulating traffic aggregates, but cannot alone resolve the congestion problem, as it does not have any control on the traffic load entering the network.

3.1 Comparing REAC with Previous Work

It is interesting to position REAC with respect to the existing paradigms and compare the findings: REAC is a distributed, measurement-based system, belonging to MBAC category. The main difference between REAC and the reactive schemes is that the congestion feedback is not generated by the traffic itself via e.g. lost packets count or missing ACKs, but rendered by an external entity, i.e., the passive measurements toolfor QoS-level estimation purposes, leaving the traffic untouched.

The conceptual difference between REAC and traditional AC is shown in Figure 2. Traditional AC enforces an admission decision upon each client arrival, as depicted in Figure 2a. The traditional methods perform capacity estimation or even an end-to-end negotiation for the AC decision, adding also some extra delay[9],[10]. In Figure 2b, the REAC decision is not tied to the arrival event and all the flows are admitted to the network; it rather monitors the QoS level in frequent intervals. Whenever the measurement indicates that the quality drops below a given threshold, REAC initiates a decision-making process to locate the latest flow with lowest priority to be sacrificed in order to improve theQoS of the other flows. Thus, the AC decision is neither proactive nor reactive; it is *regressive*.



a)AC decision triggered with traditional methods b) AC decision triggered with REAC

Figure 2.AC decision triggering with traditional AC vs. REAC

4. THE REAC ARCHITECTURE

The REAC's demonstration architecture, shown in Figure 3 consists of three modules, i.e.,

- the QoS measurement tool, for passive end-to-end QoS measurements
- the traffic classification (TC) tool, for traffic identification and classification
- the REAC controller, which contains the REAC's core, i.e., the intelligence for applying the AC policy.

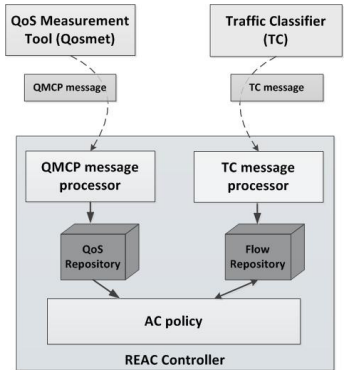


Figure 3.The REAC demonstration architecture

4.1 QoS Measurement Tool

As a QoS measurement tool, VTT's Qosmet is used. Qosmet is a lightweight and scalable passive monitoring solution and it is developed independently of REAC. Its role is to conduct end-to-end measurements of traffic statistics andQoS statistics such as average delay, jitter, packet loss, connection break duration, etc., in both communicating directions of IP networks. Qosmet is fully controllable remotely via its QMCP (QoS Measurement Control Protocol) interface, being able to convey the measurement results to third party software (SW), or even being directly controlled by a third party SW (e.g. the REAC controller).In REAC, Qosmet must perform the QoS

measurements between the controlled network path endpoints. More information of Qosmet can be found e.g., from [19] and [20].

4.2 Traffic Classification

VTT's two-phased Traffic Classification tool (TC) performs a robust traffic classification utilizing statistical characteristics of the flows. Flows are classified in two stages, i.e., during the connection establishment and at a later point, to improve the classification accuracy. TC provides REAC with information regarding the flow type and status; active or inactive. It can also be directly controlled by REAC. More information of TC can be found e.g., from [21].

4.3 The REAC Controller

The REAC controller consists of three processing components and the two repositories, as shown Figure 3.The REAC controller exploits the information conveyed by Qosmet and TC to implement the AC logic. *The QMCP message processor* parses the measurement results and updates the QoS repository. The repository holds sets of QoS values and updates them in a rolling manner, i.e., every new set of values replaces the oldest one in the repository. *The TC message processor* parses the TC messages and updates the Flow Repository, which holds one registry per flow consisting of the flow identification and status. The Flow Repository update is done in a way that for a new flow a registry is added, the terminated flow registries are removed, and the rest gets a status and age update. *The AC policy* is the core of the AC controller, described separately in the next section.

4.4 The AC Policy

The AC policy module implements the REAC logic. The purpose of the logic is to monitor the QoS-level variation and to estimate when the QoS-level decrease affects the quality of the high priority applications. Upon such an indication, the system first traces the *suspect flow(s)*, and the neither drops packets of these flows, or decreases the flow priority using e.g. Differentiated Services Code Point (DSCP) marks. Alternative methods to treat the flows may be also included, depending on the capabilities of the controlled network path. REAC does not assume the knowledge of the controlled network path capabilities, but instead, it tracks one of the most important QoS metrics: delay. The focus is on the temporal variation of the delay, averaged over a period of the last *n* updating intervals aka "measurement window". Delay is a metric which reflects well the status of the network: steady and low delay is an indication of good network conditions, while high, and often highly variable, delay is an indication of congestion. Further, an increasing delay can be an indication of upcoming network congestion, leaving room for prediction. In several other MBAC works, the monitoring is based on a windowed time, e.g., in[8], [22], [23], where the last two also use delay as a metric for decision making.

The average packet delay got from the measurement tool (e.g., once in a second)is fed through a sliding arithmetic mean calculation (from now on referred to as *mean*), with a window of size o*qos*_history. The calculation of the sliding arithmetic mean is shown in Figure 4 .The window size expresses how much history REAC takes into account in evaluating the QoS level. We remind that this is also the size of the QoS repository; hence the QoS-level estimation takes into account the last *qos_history* values.

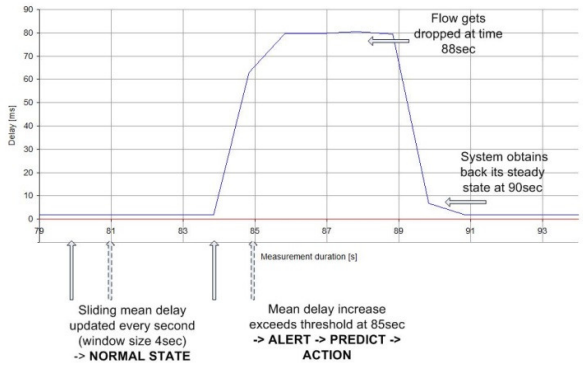


Figure 4.Sliding arithmetic mean of the average packet delay

There is a strong relation between delay characteristics and network utilization[24] when the network reaches the point of congestion. Chen et al. stated in their work [25] that human factors and technology attributes have revealed that i.e. delay can render the application requirements such as for video. Furthermore, delay variation can be accurately matched to network events such as the entrance of a new flow or a network DoS (Denial-of-Service) attack. One should notice that besides delay, other metrics could also be used. In fact, one could use a combination of metrics, and even metrics that perform pseudo-subjective analysis of the quality. An example of such metrics is PSQA (Pseudo-Subjective Quality Assessment)[26], or the more recent GQoSM (Generic QoS Measure), which is still under development by VTT. Both of the mentioned QoE related metrics can be calculated with Qosmet.

The REAC logic is implemented by a Finite State Machine(FSM), as depicted in Figure 5. The FSM has five states, namely: SETUP, NORMAL, ALERT, PREDICT, and ACTION, described next. The relation between the FSM states and mean delay is presented also in Figure 4. The FSM operation is synchronized to the arrival and processing of the QoS samples.

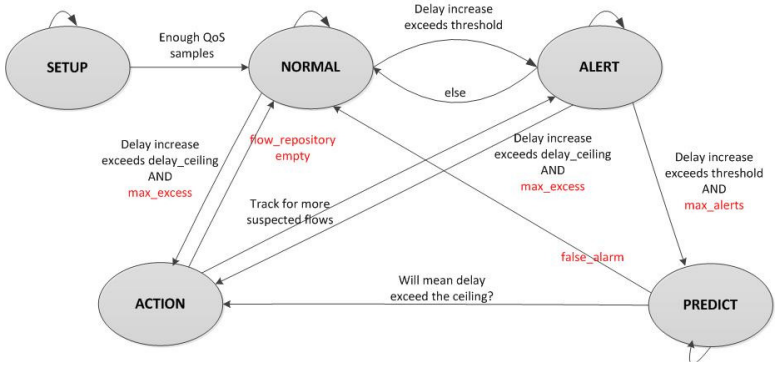


Figure 5.FSM describing the REAC logic

The SETUP state

During the SETUP state, the system fills its repositories. The aim is to collect sufficient QoS history before the actual REAC operation starts. The duration of this state is longer than the *qos_history*, in order to allow the system to obtain stable values. This justifies the use of *rounds*, e.g. 10. The *qos_history* is an input parameter to the system. The next to SETUP states are either SETUP, or NORMAL. The algorithm of the SETUP state is shown in Algorithm 1.

Algorithm 1 The SETUP state

Require: *qos_history> 0; rounds > 0*

```
samples=0;
if(+ + samples == qos_history * rounds) then
new_mean = calculate_mean ();
state = NORMAL;
end if
```

The NORMAL state

When the system is in NORMAL state, it calculates the mean delay over the *qos_history* window, from the values stored in the QoS repository. The current *mean* is compared to the previous; if their quotient exceeds a certain percentage (in REAC *relative_delay_threshold,* "threshold" in Figure 5), the system increases an *alerts* counter to indicate this excess, and enters the ALERT state. Otherwise, it remains in the NORMAL state, and decreases the *alerts* counter, if this is non-zero. The *relative_delay_threshold* is an input parameter to REAC, greater than 1. This means that we allow for an up to exponential growth of the mean before entering the ALERT state. The justification of this claim follows in the Appendix.

The, next to NORMAL, states are either NORMAL, ALERT, or ACTION. If delay just increases steadily and slowly enough, it is possible that the detection system does not notice the upcoming congestion. In order to react to this special case, there must be also a cold check for the absolute mean delay value. This is done by comparing mean delay with the delay value that represents the highest point of acceptable before e.g. the bottleneck collapses. This value, namely *delay_ceiling* in REAC, expresses the network tolerance and is an absolute limit as opposed to e.g. *relative_delay_threshold,* which is a percentage. It is network dependent and can be identified by active network measurements, before the REAC deployment. In our implementation it was identified to be roughly 40 milliseconds. The check of mean delay against the *delay_ceiling* must be positive *max_excess* times (not react instantly to a single traffic peak), before the system proceeds to ACTION.The algorithm of the NORMAL state is shown in Algorithm 2.

Algorithm 2 The NORMAL state

```
old_mean  =new_mean;
new_mean = calculate_mean ();

if(new_mean / old_mean>relative_delay_threshold) then
        + + alerts;
        state = ALERT;
else if (alerts > 0) then
        -- alerts;
end if
if (new_mean>delay_ceiling)then
        if(+ + excess == max_excess) then
                excess = 0;
                state = ACTION;
        end if
else
        excess = 0;
end if
```

The ALERT state

The system maintains its ALERT state whilst the mean delay increase remains over the *relative_delay_threshold*. Every such increase produces an alert. When a number of *max_alerts*is reached, the system resets the counter and enters a new state, namely PREDICT. The *max_alerts* is an input parameter to the REAC system, in order to get some knowledge about the system operation before going to PREDICT state. If the increase rate stays below the *relative_delay_threshold*, the algorithm may revert to the NORMAL state, leaving the *alerts* counter unmodified. The counter will be gradually reduced while the system continues in the NORMAL state, which means that REAC can forget the previous shock gradually. Whether or not the next state will be the NORMAL depends as well on the mean delay. In addition, in ALERT state, there is a similar check with the absolute mean delay with the *delay_ceiling* as in the NORMAL state, in order to cause an immediate action to ensure that (i) high congestion has not gradually built up without triggering any alarms, or (ii) sufficient action was taken during the ACTION state. The, next to ALERT, states are either NORMAL, ALERT, PREDICT, or ACTION. The algorithm of the ALERT state is shown in Algorithm 3.

Algorithm 3 The ALERT state

```
old_mean   = new_mean;
new_mean = calculate_mean();

if (new_mean / old_mean<relative_delay_threshold) then
      if (new_mean>delay_ceiling) then
           if(+ + excess == max_excess) then
                      excess = 0;
                      state = ACTION;
           end if
      else
           excess = 0;
           state = NORMAL;
       end if
else if(+ + alerts == max_alerts)then
      alerts = 0;
      state = PREDICT;
end if
```

The PREDICT state

The system transits to the PREDICT state because the mean delay has kept increasing for at least a period of *max_alerts* intervals. After sequential *max_alerts*, REAC evaluates the mean delay with respect to the *delay_ceiling*. Instead of comparing the two values, REAC makes a projection into the future of its current behaviour. The prediction assumes that, since the mean delay has been increasing in a faster-than-exponential way during the last updating intervals, it will keep the same tendency. The purpose for such a prediction is to give the system some self-knowledge, and the possibility to diagnose whether its current state is pathological, or not. If the mean delay stops increasing exponentially, the system increases a *false_alarms* counter. It allows for *max_predictions* to happen, before considering that the shock period is finished, and revert back to NORMAL state. On the other hand, if the prediction shows that in the next interval the system will have surpassed the ceiling, it immediately enters the ACTION state. The, next to PREDICT, states are NORMAL, PREDICT, or ACTION. The algorithm of the PREDICT state is shown in Algorithm 4.

Algorithm 4 The PREDICT state

```
old_mean = new_mean;
new_mean = predict_mean();

if(new_mean>delay_ceiling) then
        state = ACTION;

else if (++false_alarms == max_predictions) then
        false_alarms = 0;
        state = NORMAL;

end if
```

The ACTION state

In the ACTION state, the system invokes a process for tracking down the suspect flow, i.e., latest flow with lowest priority. In order for this to be feasible, the Flow Repository has to be non-empty; otherwise no action can be taken, and the system reverts back to the NORMAL state. REAC makes the following assumptions regarding the suspected flow: (i) it has not entered later than the PREDICT state, (ii) it is not treated during previous ACTION state or (iii) it does not belong to the protected flows e.g. QMCP and NTP (Network Time Protocol for time synchronization).

Taking an action over the suspect practically means that the client's traffic (of that single application, responsible of the suspect flow) will either be dropped (BE network) or marked (prioritization-enabled network). The dropping or marking rules are composed and given to the edge node's configuration in real time. After the appropriate action over the suspect has been taken, the system enters the ALERT state, and expects to see that the mean delay has stopped increasing with an exponential rate. If the system is still congested after dropping or marking the suspected flow, *delay_ceiling* will be used as limit for instantly re-entering the ACTION state. In some cases it is not sufficient to terminate just one flow, hence also other flows need to be dropped or put to lower priority in order to alleviate the congestion. Next to ACTION, state is ALERT and NORMAL. The algorithm of the ACTION state is shown in Algorithm 5

Algorithm 5 The ACTION state

```
if(Flow repository.empty()) then
        state = NORMAL;
else
        find suspect();
        if(target_action == DSCP)then
                mark_suspect_flow();

        else if(target_action == DROP)then
                discard_suspect_flow();

        end if
        state = ALERT;

end if
   false_alarms = 0
```

---

5. TEST BED AND MEASUREMENT SETUP

REAC was implemented and validated in a test bed setup in the VTT's Converging Networks Laboratory [27].Figure 6illustrates the development and testing platform with three network entities, i.e., a core network, a router network with REAC and an access network. All other connections are 100 Mbps full-duplex links, except the bottleneck between the Cisco3600 and the edge router, which is restricted to 10 Mbps (full-duplex), in order to more easily congest the network for the testing purposes.

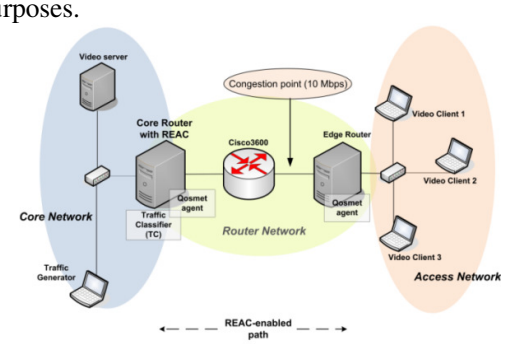


Figure 6.The REAC test bed setup

The core network comprises of the video server using Windows XP and a traffic generator (TG) running on Linux (Ubuntu). The router network consists of core (CR) and edge (ER) routers running Linux (RedHat), and a Cisco3600 router. CR carries the intelligence with the REAC module that gathers information from TC and Qosmet measurement agents. The access network consists of a number of clients, i.e. in our case 3 Windows XP clients, each consuming one or more UDP video streams. The Edge and Core routers are synchronized with NTP in order to be able to calculate one way delay. The NTP's accuracy is on the order of 1–10 milliseconds, but it is enough for detecting congestion. The test bed measurements were conducted according to two scenarios described next.

5.1 Measurement Setup for the Proof of Concept Scenario

First scenario is a proof of concept (POC) for REAC. We measured a number of high-priority video clients along with artificially generated low-priority traffic (by D-ITG [28]) to congest the network. POC scenario imitates a case, where a single high-rate flow dominates the load of the network path, being similar also to a network DoS attack situation. Compressed low quality (500 kbps-1 Mbps) and high quality (2 Mbps-4 Mbps) videos were entering the network periodically in turns, according to the timing diagram depicted inFigure 7.The first video streaming started after a 30 seconds idle period. Every new video was streamed for 60 seconds before entering the high-rate burst. The bursts had duration of 30 seconds. After the burst was ended, 60 seconds was waited before entering a new video. Altogether five videos were entered to the network, of which 3 were low quality and 2 high quality.

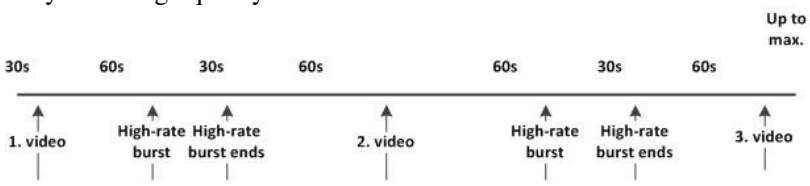


Figure 7. Proof of concept scenario timing diagram

Measurements were carried out for the different network path setups: (i) pure BE, (ii) DiffServ with DSCP marks, (iii) REAC over BE (packet dropping), and (iv) REAC over DiffServ (DSCP marks). Each measurement was run four times to gain statistical certainty for the results. This was enough, since the test scenario is sufficiently static.

5.2 Measurement Setup for High Priority Dilemma Scenario

In the second scenario, the video clients enter the network one-by-one at random times with exponentially distributed inter-arrival times. The distribution mean is 10 seconds with a maximum of 100 seconds. Eventually, the system is pushed at its extreme with more high-priority clients that the system can tolerate. The videos, obviously, would be of high priority, while there are too many of them for the network. Hence, we call this scenario as the High Priority Dilemma scenario. Videos entering the system are similar to what is used in the POC scenario; the low and high quality videos are streamed in turns. For the bottleneck link that we were using (10 Mbps), 8 videos consisting of 4 low and 4 high quality videos, were enough to cause a serious congestion. Measurements were performed for three different path setups: (i) pure BE, (ii) REAC over BE (packet dropping), and (iii) REAC over DiffServ (DSCP marks). Measurements for each path were performed ten times to gain statistical certainty. This scenario needed more repetitions because of the random arrival of clients. This is an interesting scenario where DiffServ fails, since all the flows are of high priority. As a results, since DiffServ performs equally to the BE path, the results of DiffServ path are not reported.

6. MEASUREMENT RESULTS

6.1 On the Performance Metrics

To evaluate the performance of REAC, we used the application level QoS information, along with QoE subjective analysis, because the end-user experience is what eventually matters. Subjective tests were carried out by means of real users observing the video quality. Even a limited number of test users was enough to draw conclusions, especially, since the results seen later have significant differences between the different AC methods. QoE can be measured in numerous ways, but in this work Mean Opinion Score (MOS)was used, that indicates the average opinion score of a group of users. The scale of MOS is often between 1 and 5 where the numbers present a verbal counterpart of the perceived quality [29]. In this work we used absolute category rating (ACR) where 5 stands for “excellent”, and 1 for “unusable” quality. Value 3 represents a midpoint value where the quality is, on the average, fair, but impairments are already slightly annoying, being not suitable for long time use. From the real-time runs, we recorded the average quality, but also – what is perhaps more important – the percentage of time that the quality was tolerable for longer use (> 3), and when it was not (< 3).

One has to notice that there are a number of videos injected to the network. The user observes the quality of the first video flow that provides a good view of the overall performance, since, when e.g., in a BE network congestion occurs, all the flows will suffer. In addition to the subjective quality, delay is measured, as is its standard deviation (one form of *jitter*). Also the percentage of the total time the delay ceiling (40ms in the test bed, explained in section 4.4) is exceeded is measured.

Utilization, average throughput and control overhead are also important metrics. When considering the utilization of different cases, we calculated first the normalized average throughput S_{avg}, i.e., the amount of routed data traffic per time unit, over the whole measurement. Normalization is done to the nominal maximum data rate of the network path, D_{r_max}, which now

is 10 Mb/s. *Maximum utilization*(U_{max})in Equation 1, and delay under the maximum utilization periods, i.e., *congestion delay*d_c in Equation 2, were calculated over the time when the total offered load (G_{tot}) exceeded the bottleneck link capacity. Control overhead C_n, which is, in practice, Qosmet's load, is simply the Qosmet's calculated load normalized to the total load. Utilization is calculated as

$$U_{max} = \frac{\sum_{O_n > D_{r\_max}} (T_n - C_n)}{\sum_{O_n > D_{r\_max}} (N)} \quad (1)$$

, where T_n and O_n represents measured throughput and offered (total) load samples n, respectively. N expresses the number of samples n. In BE and DiffServ paths, , always. Congestion delay is calculated as

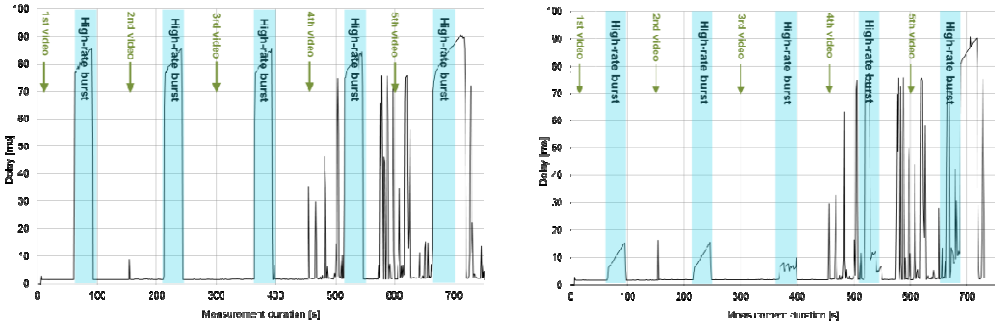
$$d_c = \frac{\sum_{O_n > D_{r\_max}} (d_n)}{\sum_{O_n > D_{r\_max}} (N)} \quad (2)$$

, where $d_n$represents measured delay samples n.

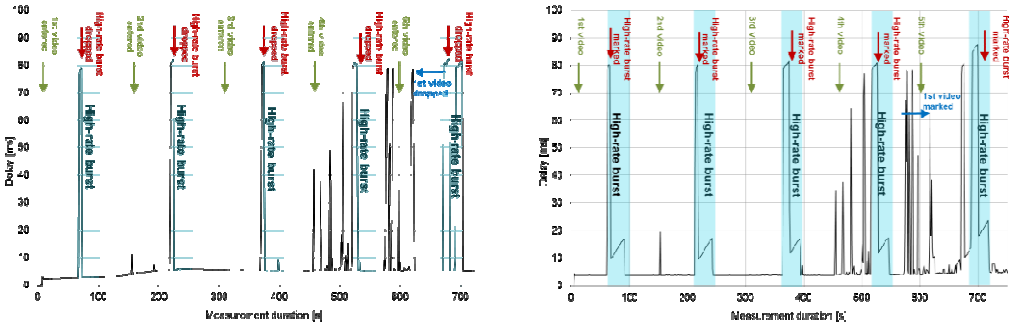
6.2Measurement Results of the POC Scenario

Exemplary snapshots of single measurement runs, showing the video delay behavior and comparisons for the POC scenario are presented in Figure 8.Table I shows average results collected over all the measurement runs of POC scenario. Note that since the measurements are carried on the fly, the averages in the table are averages over time and also over different runs. Thus, for instance, delay standard deviation (Delay St.Dev.) presented in Table I is not the deviation between different runs, but deviation of delay samples during the measurement run averaged over all the runs.

The measured delay for the pure best effort path with sequential video streams and artificial high-rate BE traffic bursts is shown in Figure 8a. The BE path suffers from long delay periods, when the big burst of artificial traffic enters. When having all the 5 multimedia streams running, the overall bit rate of the videos start to exceed the network path capacity and there is a lot of fluctuation observed in delay behavior. During the subjective evaluations, it was noticed (not shown) that when high delay was measured, the quality of all videos gets unsatisfactory. The overall average delay for BE path is 23 milliseconds, which is twice the delay compared to REAC cases as noticed in Table I. Even though the overall subjective score(MOS) is 4.3, being acceptable, the subjective quality is poor ~ 16% of the time, which is a very high value. This tells that the congestion, observed as a high delay values, reflects to the subjective quality as well.



a) Pure BE path b) DiffServ DSCP marking path



c) REAC over BE (pkt drop) d) REAC over DiffServ (pkt mark)

Figure 8. Measurement results of POC scenario

The DiffServ configuration gives a better performance than the pure BE since, by its nature, it is able to cope with the traffic bursts of the BE class as the videos belong to the high-priority class. However, it fails when there is too much high-priority traffic, as perceived from Figure 8b: The delay behavior becomes very similar with that of the BE case (Figure 8a). As the network path is congested by high-priority traffic, prioritization does not help any longer, and all the flows start to suffer from bad quality. This demonstrates the problem of statistical QoS as discussed earlier. The subjective quality is poor ~ 8% of the time (Table I).

The REAC performance with measured delay values are shown in Figure 8c and Figure 8d, with drop and marking policies, respectively. The graphs show that during the high-rate burst periods, there is a short delay spike in the beginning, which represents the time it takes from REAC to make decisions considering the newly entered flow. During this period, user's experienced quality drops, but only just for couple of seconds. REAC is able to cope with the BE traffic bursts, using the novel AC policy, even though the network path itself would not support prioritization (REAC over BE, Figure 8c). The drawback against pure DiffServ case (Figure 8b) is that DiffServ allows direct prioritization, and there is no delay spike when the BE traffic enters the network path. However, while DiffServ fails in the presence of too many high-priority flows, REAC does not: good quality is guaranteed to the majority of the flows. The limit, for how many flows the quality can be guaranteed, comes directly from the relation of the controlled network path capacity and load of the traffic flows. Roughly, the QoS can be guaranteed for the flows which are of high-priority class, and as an aggregate, do not exceed the network path capacity. In our 10 Mbit/s test scenarios, REAC was able to guarantee good quality for approximately two low-quality and two high-quality videos. The REAC's better performance can be seen already in the behaviour of the delays, but it is evident, when observing the results in Table I. The subjective quality is poor only about 2-3% of the time, depending on the REAC method, being clearly better than with DiffServ, and considerably better when compared to the pure BE path. The difference between REAC dropping and marking performances is not very notable in the POC scenario, as observed from Table I. REAC is capable of enabling guaranteed QoS for a subset of the aggregate of traffic regardless of the network path's support for prioritization.

Table I. Averaged subjective quality and delay values in POCscenario.

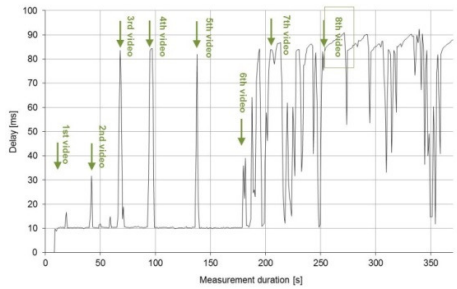
	pureBE	DiffServ	REAC-drop	REAC-mark
Subjective score (MOS)	4.3	4.6	4.8	4.8
% of time goodquality	73.4	87.3	91.6	91.5
% of time intolerable quality	15.9	8.0	3.1	1.9
Delay [ms]	23.2	11.1	10.7	12.2
Delay St. Dev. [ms]	34.1	21.0	20.4	20.1
% of time above threshold	28.0	10.7	10.2	11.4

6.3 Measurement Results of High Priority Dilemma Scenario

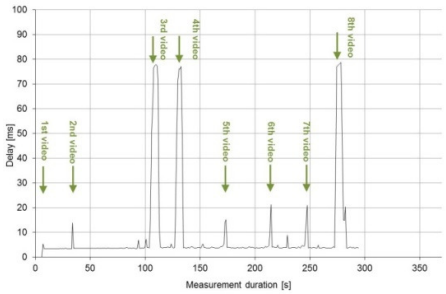
In the second scenario, we are pushing the network to its extreme by introducing the network with more high-priority video flows than it can handle .In this kind of scenario, where all the users are of high-priority type, the performance of DiffServ path equals the BE path. Thus, we leave the DiffServ path out and put the focus only to the BE path as a point of comparison.

An exemplary snapshot of a single case over the pure BE path is shown in Figure 9a. The BE path is able to handle, with a reasonable quality, an aggregate of 5 videos. After the 6th video enters, the delay starts to fluctuate heavily. At this point during the tests it was also visually observed that all the videos start to suffer from bad quality. This means that these extra flows ruin the performance of all the multimedia users.

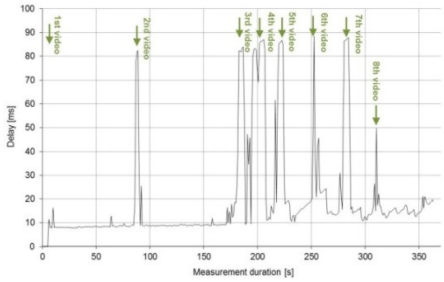
In the REAC case, our AC module ensures that the most of the videos will keep on enjoying the good quality, even though the total offered traffic load exceeds the capacity. When exceeding the capacity, the Qosmet monitoring solution quickly notices the QoS degradation, and the AC module can react within a couple of seconds, making no notable damage to the existing users in the network. This can be seen in the snapshots of the delay behavior in Figure 9b and Figure 9c, being very different from that ofthe BE path case. For a reader, please notice that as the runs are independent random measurement runs, the videos arrive at different points at different runs, hindering the direct visual comparison.



a) Pure BE path



b) REAC with packet dropping



c) REAC with packet marking

Figure 9.Measurement results of High Priority Dilemma scenario

As seen in Table II, throughout the averaged measurement sets, the subjective quality is poor only < 2% of the time in both REAC dropping and marking methods, while the BE suffers bad quality ~ 16% of time. Also, REAC cases allow the quality to be at good level > 92% of time, while only about 50% is achieved with the BE case. While it was not a surprise that REAC outperforms the BE case, the results clearly show that REAC can be used to give guarantees to quality. Further, it must be recalled that the got results for REAC are got with a single set of many available parameters. For example, if one wishes to cut the percentage of time that the user is experiencing bad quality, one could, e.g., reduce the delay_ceiling, and/or max_excess values, that are explained in section 4.4. As a consequence, this could also cut the number of videos for whom the good quality can be guaranteed. Therefore, setting these parameters is balancing between, how much disturbance is allowed, against for, how many users the quality is to be guaranteed. After entering maximum amount of videos (8) into the network, REAC cases were able to guarantee good quality from 3 to 4 users, compared to zero guaranteed users in pure BE path, as seen inTable II.

Table II.Averaged subjective quality and delay values in High Priority Dilemma scenario.

	pureBE	**REAC-drop**	**REAC-mark**
Subjective score (MOS)	3.6	4.8	4.8
% of time goodquality	52.6	92.0	92.1
% of time intolerable quality	15.9	1.8	1.6
Delay [ms]	49.9	16.5	21.3
Delay St. Dev. [ms]	35.6	18.9	20.6
% of time above threshold	60.1	13.4	25.7
No. ofQoS guaranteed videos (max.8)	0	3	4

As it was seen also in POCscenario (Table I), the measured average delay (Table II) in the dropping case is slightly smaller than in the DiffServ marking case. This relates to the implementation of DiffServ mechanism in the routers of our test bed. In the dropping case, the flows are really dropped before they enter to the network path, i.e., they never enter the congestion point. In the mark case, however, all the traffic is allowed to enter the network until to the point of congestion, which is the Cisco3600 router. The router will execute the queue policies for the packets based on their DSCP value and its DiffServ setup. This requires some effort from the router, and the overall delay level is measurably increased.

6.4 Considering REAC's Overhead and Network Utilization

AC mechanisms always bring some overhead with the methods they are using, e.g., for measurements and control policies, and this is one of their drawbacks. In the REAC scheme, however, the information about network conditions is rendered by an external entity, i.e., the measurement tool, and there is no other control overhead needed. Another drawback of some AC schemes is the decision delay, which however, in REAC case, because of the regressive operation, is zero for the flows who will be allowed to continue as high-priority flows.

A snapshot of Qosmet's control load characteristics in REAC usage compared to total load in POC scenario, REAC dropping case, is presented in
Figure 10. We see that Qosmet's load seems to somewhat follow the total load, which is caused by the internal functionality of Qosmet. Still, overall Qosmet's load is only a fraction of the total load (Table III).

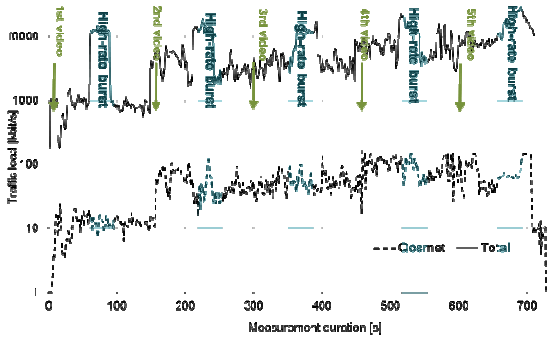


Figure 10.Qosmet load vs. total load [kb/s] in POC scenario

Table III.POC scenario utilization.

	pureBE	DiffServ	REAC-drop	REAC-mark
Control overhead	-	-	0.014	0.010
Avg. throughput	0.558	0.564	0.375	0.558
Max. utilization	0.973	0.970	0.451	0.972
Congestion delay [ms]	67.3	21.8	22.0	23.6

Table IV.High priority dilemma scenario utilization.

	pureBE / DiffServ	REAC-drop	REAC-mark
Control overhead	-	0.014	0.011
Avg. throughput	0.767	0.501	0.733
Max. utililzation	0.950	0.671	0.967
Congestion delay [ms]	75.9	19.6	27.2

The control overhead is between 1.0 – 1.4% over all the REAC cases as seen in Table III and Table IV, being reasonable. When considering average throughput, there are no significant differences between pure BE, DiffServ and REAC marking cases. However, REAC-drop gives considerably lower throughput than the other cases, which is, naturally, caused by the fact that the excess flows are completely dropped before entering the network path. This, unfortunately, leads also to low maximum utilization. We must, however, recall that this test case is pessimistic for REAC-drop path, since the traffic load of a single flow is high when compared to the D_{r_max}, leading easily to a situation, where high utilization rates cannot be achieved. Then again, REAC marking shows very good performance when observing maximum utilization: the got values, ≈ 0.97 are very high values taking into account that many AC schemes suffer from low utilization. The biggest differences are shown in the congestion delay, where REAC paths achieve clearly the lowest values, meaning that REAC is able to fight against congestions.

7. CONCLUSIONS

We presented a new principle for AC, i.e., *regression*, according to which the network unconditionally accepts new flows, but evaluates their impact upon existing clients, in order to perform the AC decisions. Regression allows instant access of new flows without a need of waiting a decision. REAC is also able to bring QoS for a limited number of flows, even in a network path which does not support any QoS methods. The basic policy favors old flows over the new ones (in the case they have the same priority class), being similar to well proven and accepted policy in the POTS. From the implementation point of view an advantage of the presented REAC is that we measure only the aggregate of flows, and thus generate much less calculations, data to process, and overhead than for having a separate measurement for each flow. REAC handles the network path as a black box with extra intelligence added only to the network path edges. A drawback, naturally, is the implementation of the logic on the edges, but as modifications are needed only there, the overhead is comparable to the many alternative AC schemes, some of which might even assume participation from every router in the network.
We introduced a technology demonstration implementation of REAC, and performed test bed experiments that proved REAC to be effective and robust. The effectiveness of REAC is greatly dependent on the accuracy of the measurements, and of the applied admission policy and logic. In the tests, we only used the delay as the metric to do the decisions. It worked very well, but there might be some cases where it does not, considering e.g., that one-way delay needs good clock synchronization, which cannot be always provided. Thus, besides delay, other metrics can be identified and used to determine the quality of the path.

8. FUTURE WORK

This paper presents the first results of the REAC system. In the future work, links of higher speed need to be inserted into the test bed, and the system needs to be tested in a larger scale, and by measuring the quality of all the high-priority flows e.g., with pseudo-subjective metrics. Also, different kind of traffic types are needed to better approximate real active networks. Also, it would be interesting to compare REAC directly with other AC methods.

The measurement overhead of REAC cannot be either neglected, but so far the pros seem to be much stronger than the cons. As a side product, REAC can be used to guarantee performance for certain important applications and/or users. In this we mean something beyond classification of traffic flows to high-priority multimedia vs. background data transmission, i.e., REAC could be used to enable guarantee of delivery of some vital information, e.g., emergency calls, or other similar data in public authority networks or even in military communications networks.

9. APPENDIX: FORMAL JUSTIFICATION

Our construction is based on a delay sequence $\{d_1, \dots, d_n\}$ being defined on the equidistant time instances $\{t_1, \dots, t_n\}$, $t_{i+1} - t_i = \Delta t$, and measuring the sliding arithmetic mean of the delay. In the NORMAL state we consider that

$$d_{n+1} \leq h_n d_n$$

where$h_n > 1$ denotes the relative_delay_threshold; in the general caseh_n can be time dependent, therefore the notation. It is straightforward to compute

$$d_{n+1} - d_n \leq (h_n - 1) d_n$$

which, due to Taylor series expansion, about the point t^n with $\Delta t = 1$, constitutes a first order discrete approximation of the following differential inequality

$$d'(t) \leq (h(t) - 1)d(t)$$

at $t = t^n$, where $d(\cdot)$ is a continuously differentiable interpolation of the delay sequence $\{d_1, \dots, d_n\}$and $h(\cdot)$ a continuous interpolate of the relative delay threshold sequence$\{h_1, \dots, h_n\}$.

We assume that $d'(t) \leq (h(t) - 1)d(t)$ for every $t \in (t_n, t_{n+1})$. We note that $h(t) \geq 1$ so that the solution of the differential inequality can be given by the Gronwall's theorem

$$d(t) \leq d(t_n)e^{\int_{t_n}^{t}(h(s)-1)ds} = d_n e^{\int_{t_n}^{t} h(s)ds-(t-t_n)}.$$

Remark:
More adaptable to our case is a constant value $h(t) = h$which yields an exponential growth of the form

$$d(t) \leq d_n e^{(h-1)(t-t_n)}.$$

Experimentally we have noticed that the constant value function $h(t) = h = 1.3$is sufficient.

ACKNOWLEDGEMENTS

We would like to acknowledge the Finnish Funding Agency for Technology and Innovation (Tekes) for their financial support during this work.

REFERENCES

[1] Finnish Ministry of the Environment, Everyman's Right, available at: http://www.gigablast.com/get?q=&c=dmoz3&d=229807102451&cnsp=0
[2] Seppänen, J. & Varela, M. (2013), "QoE-driven Network Management For Real-Time Over-the-top Multimedia Services", IEEE WCNC.
[3] HichemAyedHarhira, H. A. & Pierre S. (2009), "Dynamic Admission Control and Path Allocation for SLAs in DiffServ Networks", In IEEE CCECE.
[4] Nabhen, R. et al. (2007) "DiffServ PBAC Design with Optimization Method", Book chapter in IP Operations and Management.
[5] Kim, J.&Jamalipour A., (2001), "Traffic management and QoS provisioning in Future Wireless IP Networks", IEEE Personal Communications.
[6] Xu, Y., & Guerin, R. (2005) "Individual QoS versus aggregate QoS: a loss performance study", In IEEE/ACM Transactions on Networking 13.
[7] Knightly, E.W., &Shroff, N.B. (1999). "Admission control for Statistical QoS: Theory and Practice", In IEEE Network 13.
[8] Jamin, S., Shenker, S. & Danzig, P. (1997)."Comparison of Measurement-based Admission Control Algorithms for Controlled-Load Service," Proc. of INFOCOM '97.
[9] Dabrowski, M., &Strohmeier, F. (2003). "Measurement-based Admission Control in the Aquila Network and Improvements by Passive Measurements", In proc. of Architectures for Quality of Service in the Internet, Art-QoS'03, pages 189-202, Berlin, Germany.
[10] ajszczvk, A. et al. (2013), "Enhanced measurement-based admission control for flow-aware networks", In International Conference on Computing, Networking and Communications (ICNC), pp. 922-926.
[11] Ammar D. et al., (2012), "KBAC: Knowledge-Based Admission Conrol", In IEEE Conf. In Local Computer Networks.
[12] Casagrande, D. et al, (2013), "Fair and optimal dynamic admission control of elastic flows", In Journal of Computer Networks, Elsevier Publications.

[13] Milbrandt. J. et al., (2006), "Experience-Based Admission Control with Type-Specific Overbooking", IPOM'06, pp. 72-83.
[14] Georgoulas, S., Trimintzios, P., Pavlou, G., (2004), "Joint Measurement- and Traffic Descriptor-based Admission Control at Real-Time Traffic Aggregation Points", IEEE International Conference on Communications, ICC 2004, vol. 4, pp., 1841 – 1845.
[15] Yerima, S.Y., (2011), "Implementation and Evaluation of Measurement-Based Admission Control Schemes within a Converged Networks QoS Management Framework", In IJCNC, vol.3, no. 4.
[16] Ming, L. & Hoang, D.B. (2005)."FIAC: a resource discovery-based two-level admission control for differentiated service network", Journal of Computer Communications in Elsevier Publications.
[17] Más, I., &Karlsson, G. (2007) "Probe-based admission control for a differentiated-services internet", Elevier Journal of Computer Networks.
[18] Alipour, E. &Mohammadi, K. (2008)."Adaptive admission control for quality of service guarantee in differentiated services networks", IJCSNS.IJC-SNS, 8(6).
[19] Prokkola, J. et al., (2007). "Measuring WCDMA and HSDPA Delay Characteristics with Qosmet," In proc. of ICC 2007, Glasgow, UK, 24-28.
[20] Qosmet – Enabling passive QoSmeasurements, URL: http://www.cnl.fi/qosmet.html, checked 10/2013.
[21] Hirvonen M., & Laulajainen, J-P. (2009). "Two-phased network traffic classification method for quality of service management", IEEE 13th International Symposium on Consumer Electronics (ISCE).
[22] Jamin, S., Danzing, P. B., Shenker S. J. & L. Zhang, (1997) "A Measurement-Based Admission Control Algorithm for Integrated Services Packet Networks,", ACM/IEEE Trans.Netw., vol. 5, no 1, pp. 56-70.
[23] C. Casseti, J. Kurose, and D. Towsley, (2000), "An AdaptiveAlgorithm for Measurement-Based Admission Control in Integrated Services Packet Networks" J. Computer Communication, vol 23, no. 14-15, pp.1363-1376.
[24] Jacobsen, V., & Karels, M.J. (1998). "Congestion Avoidance and Control", ACM SiggComm.
[25] Yan, C., Farley, T. & Ye, N., (2004), "QoS Requirements of Network Applications on the Internet", Journal on Information, Knowledge and Systems Management.
[26] Varela, M. (2005). "Pseudo-Subjective Quality Assessment of Multimedia Streams and its Applications in Control," Ph.D. thesis, University of Rennes 1, France.
[27] "Converging Networks Laboratory," VTT, URL: http://www.cnl.fi,checked 10/2013.
[28] Distributed Internet Traffic Generator. URL (10/2013): http://www.grid.unina.it/software/ITG/
[29] Methods for subjective determination of transmission quality, ITU-T, Recommendation P.800, Aug. 1996.

Authors

Ph.D student MirjamiJutila received her M.Sc. degree from Department of Electrical Engineering, University of Oulu, Finland in 2005. She joined VTT Technical Research Centre of Finland in 2004 and before that she worked in industry. In VTT she has been participating in several international and nat ional research projects in the field of telecommunications and currently working in Eureka/ITEA2 Disaster Control Management (DiCoMa) project. Her research interests are widely on traffic management issues including decision making mechanisms, congestion control and network QoS.

Dr. Jarmo Prokkola received his M.Sc. degree in 2001 and Dr. Tech. degree in 2008, both from Department of Electrical Engineering, University of Oulu, Finland. He has about15 years of working experience in the field of telecommunications. In 2004, he joined VTT Technical Research Centre of Finland, where he is currently working as a senior scientist. Earlier he worked in Centre for Wireless Communications, University of Oulu. Overall, Jarmo's career has involved various different topics from multiple layers of telecommunications, which has given an ability of cross-layer orientated thinking, and has raised his interest towards cognitive networking

Dr.DespinaTriantafyllidou received her M.Sc. in Networks and Telecommunications from the University of Crete in 2005 and her Ph.D. from the University of Paris-XI in joint research with the University of C rete, in 2009. She conducted her PhD thesis at FORTH in Heraklion, INRIA Rocquencourt and LRI in Paris. She completed her post-doc as an ERCIM fellow in VTT Technical Research Centre of Finland, in 2010. Her research has focused on IP-based intelligent QoS services exploiting network traffic measurements, cross-layer design of TCP over 802.11 networks with proactive routing in mobile ad hoc networks, delay analysis and load-based marking for congestion control in 802.11 ad hoc multihop networks, and wireless MAC protocols. She has extensive experience in NS2 and OPNET models developing an optimized scalable simulation model for TCP. She is currently working as a Solution Architect in Dimetis GmbH.