An autonomic architecture for optimizing QoE in multimedia access networks

Steven Latré a, *,1, Pieter Simoens a,1, Bart De Vleeschauwer a, Wim Van de Meerssche d, Filip De Turck a, Bart Dhoedt a, Piet Demeester a, Steven Van den Berghe b, Edith Gilon de Lumley b

a Ghent University, IBBT, IBCN, Department of Information Technology, Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
b Alcatel-Lucent Bell Labs Research and Innovation, Copernicuslaan 50, B-2018 Antwerpen, Belgium

article info
Article history:
Available online 25 November 2008
Keywords:
Knowledge Plane
QoE maximization
Neural networks

abstract
The recent emergence of multimedia services, such as Broadcast TV and Video on Demand over traditional twisted pair access networks, has complicated the network management in order to guarantee a decent Quality of Experience (QoE) for each user. The huge amount of services and the wide variety of service specifics require a QoE management on a per-user and per-service basis. This complexity can be tackled through the design of an autonomic QoE management architecture. In this article, the Knowledge Plane is presented as an autonomic layer that optimizes the QoE in multimedia access networks from the service originator to the user. It autonomously detects network problems, e.g. a congested link, bit errors on a link, etc. and determines an appropriate corrective action, e.g. switching to a lower bit rate video, adding an appropriate number of FEC packets, etc. The generic Knowledge Plane architecture is discussed, incorporating the triple design goal of an autonomic, generic and scalable architecture. The viability of an implementation using neural networks is investigated, by comparing it with a reasoner based on analytical equations. Performance results are presented of both reasoners in terms of both QoS and QoE metrics.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

In today’s broadband access networks, service operators are focusing on the deployment of new added value services such as Broadcast TV, Video on Demand (VoD), online gaming and Voice over IP (VoIP). Each of these services has large service demands: they often require a considerable amount of bandwidth and only tolerate a minimum amount of packet loss, delay and jitter. In order to meet these demands, current access networks are advancing from a best-effort service delivery to a QoS aware triple-play service delivery. Of prime importance in defining the quality of these services is the Quality of Experience (QoE): the quality as experienced by the end-user. The QoE degrades due to certain anomalies in their delivery (e.g. packet loss). The type of degradation depends highly on the type of service and the current context of the network. For example, an interactive service such as VoIP is typically more vulnerable to delay than a broadcast service. In the past, service operators have successfully deployed techniques such as VoD proxies to ensure a certain quality level between the service originator and the access node. However, the same techniques often fail when applied between the access node and the end-devices due to the heterogeneity of current home network configurations. Recently, these home networks are becoming complex networks consisting of different technologies (e.g. wired and wireless) and different user devices (e.g. set top boxes and traditional pc’s), with users concurrently accessing different services.

1389-1286/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2008.11.004
The focus of this article is on the development of an autonomic architecture to maximize the QoE of all services in multimedia access networks. As illustrated in Fig. 1, this architecture spans the complete network from service originator to the end-user. Its functionality is defined through three separate layers: the Monitor Plane (MPlane), the Knowledge Plane (KPlane) and the Action Plane (APlane). The MPlane monitors the network to gather information about the current situation, the KPlane uses this information to autonomously determine the best QoE optimizing action and the APlane applies the chosen actions into the network. A possible scenario for this architecture is the following: the MPlane observes that a video service is suffering from packet loss and informs the KPlane. Based on other monitor information, such as the amount of bandwidth used inside a network, the KPlane detects that each link in the network is still underutilized in terms of bandwidth and that only the services running over a specific link suffer from a drop in quality. The KPlane concludes that bit errors on that link must occur and instructs the APlane to add a specific amount of redundancy (through Forward Error Correction (FEC) packets) to tackle the packet loss measured.

In previous work, both the MPlane [1,2] and APlane [3] architecture was studied. Also a first KPlane component was presented based on a set of analytical equations [3] and the use of fuzzy logic and neural networks to come to a more generic solution was explored [4]. Here, a generic and autonomic KPlane architecture to maximize the QoE of all services is proposed. Furthermore, a novel KPlane component is presented that is able to determine the best configuration of two QoE optimizing actions for a number of different scenarios. This novel component uses a neural network to make its decision. This component is compared to the earlier proposed analytical component. The analytical case is far from generic but performs very well and can be used as a reference to evaluate other solutions. The neural network approach is more generic and can have more advanced capabilities such as on-line learning behavior.

This article is structured as follows. First, in Section 2, some relevant work concerning autonomic computing and QoE optimization is discussed. In Section 4, the design objectives for the KPlane architecture are presented. Section 5 discusses the proposed architecture while Section 6 provides an overview of the two proposed solutions: the analytical reasoner (Section 6.2) and the neural network approach (Section 6.3). Next, in Section 7, results of both solutions obtained through simulation and a physical testbed are presented. Section 8, highlights how the architecture meets the design objectives and discusses the applicability of the two proposed solutions. Finally, Section 9 elaborates on future work.

2. Related work

The concept of a Knowledge Plane was originally presented in [5]. The authors argue that the fundamental design of the Internet, with its simple and transparent core with intelligence at the edges, leads to high management overhead. This is caused by the fact that the network simply forwards data packets, without knowing what its purpose is. While the edges can recognize that there is a problem with some service, the core cannot tell that something is wrong, because it has no idea of what should be happening. Through the Knowledge Plane, the high level goal of the network operator can be expressed by the operator or modeled by the network itself, as well as how low-level decisions, such as per router configuration of routes, relate to this. The authors argue that cognitive techniques, rather than traditional approaches, are best suited to meet the uncertainties and complexity.

We developed the Knowledge Plane concept and have first set out the architecture of an autonomous access network Knowledge Plane in [6]. We focus on the complexity of the Quality of Experience management in networks, incurred by the wide variety of new services that has emerged in recent years, such as Broadcast TV, Video on Demand and IP telephony. The complexity is due to, on one hand, the stringent Quality of Service requirements in terms of tolerable packet loss, delay, jitter and, on the other hand, the fact that these QoS requirements differ for each service. This necessitates an almost per-service and per-user management. In previous research, we have concentrated on monitoring algorithms for the Monitor Plane, to provide the Knowledge Plane with an extensive knowledge base of the network status and the QoE of the running services [1,2]. In [3], we reported on the architecture of an autonomic Knowledge Plane component on the access node and an analytical component for the reasoning. The concept of applying neural networks to implement the Knowledge Plane to enhance the QoE was introduced in [4], together with some first results and design guidelines. All previous papers focused on the autonomic component on the access node. In this article, we present a more comprehensive view on the whole Knowledge Plane, including a generic autonomic architecture. Furthermore, the concept of neural networks was studied more profoundly.

Other authors have suggested numerous QoS and QoE optimizers (e.g. protocol boosters [7], RTP retransmissions mechanism [8]). These optimizers often achieve very good results under specific circumstances but fail or are suboptimal in other situations. Through the design of the Knowledge Plane itself, we are currently concentrating on closing the loop between the monitoring of the network and the
execution of the QoE optimizing actions, so that the optimizers are activated and deactivated in the appropriate situations.

The use of adaptive architectures, such as neural networks, in QoE reasoning is a viable approach, since it allows modeling new usage patterns and QoE requirements. Neural networks have already been successfully applied to adaptive routing [9]. The use of neural networks has also been investigated in the field of QoS and QoE optimization, e.g. to optimize VoIP [10] or achieving QoS in 802.11 networks [11]. Furthermore, neural networks have been employed to predict the QoE of both video [12] and audio [13] services. In [14], a video quality monitoring suite is presented, which uses video and network related information to measure the QoE real-time. In this paper, we want to take the next step and include the execution of restoring actions. Our research focuses on developing a generic and autonomic loop that monitors the network, reasons about the obtained information but also executes QoE maximizing actions.

3. Relation between QoS and QoE of multimedia services

The QoE of multimedia services is mainly affected by the original quality of the multimedia service and the quality of its delivery. The first aspect is straightforward: audio and video services are encoded at a certain bitrate and these bitrate settings have a direct impact on the QoE of the services. A higher bitrate will result in a higher QoE and the service operator can decide at which QoE level he offers the service (e.g. a HD video as opposed to a SD video).

In our work, we focus on the transmission of these services. The quality of this delivery depends on QoS parameters such as packet loss, delay and jitter. Video services are often compressed to be transmitted over a bit rate-limited channel such as packet based networks. These compression techniques use inter-frame prediction to produce the compressed bitstream (e.g. by encoding the video in I, P and B frames). The transmission of video services can suffer from packet loss, which leads to loss of information and in turn to a drop of QoE. While there is an obvious relationship between packet loss and QoE, it is not easy to find a clear mapping. The resulting QoE value depends heavily on the type of information that was lost (e.g. an I-frame vs. a B-frame). In previous work, techniques have been presented that try to estimate the QoE based on the type of loss [15,16]. The impact of delay and jitter on the QoE has also been studied [17]. The authors argue that there is a relation between these parameters and QoE but no clear mapping can be made due to the complexity of the compression and delivery of the services.

Since no QoS parameter can accurately define the QoE of multimedia services, specific objective video quality metrics have been proposed in the past. These objective video quality metrics are mostly full-reference metrics, meaning that they compare the original video with the transmitted one and calculate a corresponding QoE value. This value should have a good correlation with the values obtained through subjective testing. In our evaluation, we calculated the QoE using two different objective video quality metrics: the Peak Signal to Noise Ratio (PSNR) and the Structural Similarity Index (SSIM). The PSNR is an objective video quality metric which is most commonly used because of its ease of calculation. In the literature [18], a PSNR above 30 dB is considered as good, while a PSNR between 25 and 30 dB is considered as having some errors but not too many. A PSNR below 25 dB is considered as poor and a PSNR below 20 dB as bad. Tests [19] show that the PSNR has a moderate correlation (i.e. 0.71) with subjective video tests. The SSIM is an objective full-reference quality metric based upon the assumption that the human visual system is more specialized in the extraction of structural information from scenes. The SSIM model takes the original and the distorted signal as input and produces a score between 0 and 1, where 1 represents perfect quality. Although the SSIM is originally an image quality metric, it is increasingly used for video quality evaluation [20]. The SSIM values should be interpreted as follows: a SSIM score below 0.7 is barely watchable, a SSIM score between 0.7 and 0.8 is considered as poor, a SSIM score between 0.8 and 0.85 has some visible distortion but is still acceptable for most people, and a SSIM score of 0.9 and higher is almost indistinguishable from the original. The SSIM is generally regarded as a better video quality metric [19] as it has a higher correlation with subjective tests (i.e. 0.82).

As the QoE is the quality as perceived by the end-user, it can only be exactly measured at the client side. However, the home network is often out of reach for a service provider or network operator as the end-user has the freedom to construct his home network as he wants. While protocols such as TR-069 [21] and RTCP Extended Reports [22] allow monitoring or managing the home network, they are not yet available on many home devices. As detailed in [16], there is a direct relation between QoS parameters and QoE. The MPLane detailed in Section 1, calculates the QoS parameters, which are used by the KPlane to optimize QoE.

4. Design objectives

We have applied the Knowledge Plane [5] paradigm to optimize the QoE of multimedia services and extended it to a three plane approach consisting of a Knowledge Plane, a Monitor Plane and an Action Plane. The three planes form a fully autonomic loop that maps to the Monitor-Analyze-Plan-Execute (MAPE) control loop as originally presented by IBM [23]. The MPLane monitors service related information such as packet loss and router queue sizes through a set of monitor algorithms. By continuously collecting this information it builds up knowledge about the network configuration. The Knowledge Plane uses this knowledge to automatically detect QoE drops for a given service and to recover from these drops by applying QoE optimizing actions. The last layer in the autonomic loop, the Action Plane, is responsible for planning and executing these actions in the network itself.

The focus is on the KPlane component in the architecture to close the loop between the MPLane and the APlane. As a fully autonomic loop is targeted, different design goals are required for the KPlane architecture. First, the
architecture must be autonomic to tackle the complexity of today’s broadband access networks. This means that the KPlane should run continuously and no interference by humans is necessary to achieve a normal KPlane operation. Additionally, the KPlane should be self-adaptive and therefore have learning capabilities. We foresee two functions of this learning behavior. One function is the discovery of new services and their interaction in the network. When a new service is deployed in the network the KPlane should detect the presence of this service, even if it has no a priori knowledge about this service. If an anomaly in the network occurs, the KPlane will have to try different actions to grasp the characteristics of this new service and how its QoE is influenced by the different QoE optimizing actions. The other learning function is correcting wrong decisions. The main idea is to enable the KPlane to solve the most common problems and grasp the operator’s policy. After deployment, it will continue to improve its decision making process, in order to adapt itself to the home network specifics by continuously evaluating its decision. When the learning component inside the KPlane detects that it is making the wrong decisions (e.g. when deploying an action that is intended to tackle packet loss only increases the packet loss) it should alter the reasoning component inside the KPlane to change its behavior. A last characteristic of this autonomic behavior is that it should be both proactive and reactive. This means that the KPlane should take both precautionary measures to avoid QoE drops as well as measures to react to a sudden QoE drop.

A second design goal is that the architecture must be generic. Current broadband networks are continuously evolving as new services and technologies emerge. Consequently, new service enablers in the form of QoE optimizing actions are introduced as well. This requires an architecture that is easily extensible. The KPlane should provide a platform that allows the addition of new QoE optimizing actions and new monitor inputs. However, as adding this knowledge requires an operators policy to change the behavior of the KPlane, changing the inputs and outputs of the KPlane does not have to be part of the normal KPlane operation. Instead, such modifications can be done apart from the already running KPlane (e.g. by training and testing the KPlane in an off-line simulation environment). When the new KPlane is ready to be deployed in an on-line setting, it can replace the old one, e.g. through a firmware update.

Scalability is the third design goal. At any given time, a varying number of users can employ a varying number of services. The KPlane should cope with all these services and should therefore support a multiservice QoE optimization. Furthermore, we are targeting a global QoE maximization of all given services. This means that the KPlane should not maximize one service if this means discriminating other services. Finally, as the KPlane should span the complete network from service originator up to the user, it should be distributed.

5. Architecture functional description

Fig. 2 illustrates the architecture of the three plane approach consisting of the earlier proposed Monitor Plane and Action Plane. In this article, the main focus is on the other two components in the architecture: a knowledge base comprising all relevant information and the Knowledge Plane that closes the autonomic loop through reasoning.

5.1. Monitor Plane

The main objective of the MPlane is to provide the autonomic loop with a complete and detailed view of the network. Such a view is obtained by installing several monitor probes at demarcation points (e.g. access nodes,
video servers) to monitor parameters such as packet loss and router queue sizes. If needed, more service specific information can be monitored such as the frame rate of the video. The MPlane consists of a set of different monitor algorithms, monitor probes and a way to summarize the data generated by the monitor algorithms on both a spatial and temporal basis. Today, a broad range of monitoring frameworks exist [24, 25] for each part of the network. In the past, we also proposed algorithms that can monitor the packet loss experienced at the user for both TCP-related [1] and UDP-related [2] traffic.

5.2. Action Pane

The Action Plane is responsible for executing QoE optimizing actions in the network. To do this the Action Plane expects a complete configuration of the actions to take from the Knowledge Plane. Based on this configuration, the APlane instructs specific nodes to alter their network configuration. For example, a possible action can be to add Forward Error Correction (FEC) packets to an existing stream [7]. These FEC packets can be considered as a form of redundancy: when packet loss occurs, these redundant FEC packets can be used to reconstruct the original data packets. In this case, the KPlane determines the right amount of FEC packets to add. If the Knowledge Plane instructs the Action Plane to add a specific amount of FEC for only one specific link, the APlane can instruct a component just before the link to generate additional FEC packets and instruct a component just after the link to reconstruct the stream.

Note that such a scheme is only beneficial when both a FEC encoder and FEC decoder exist in the network. While the KPlane can instruct the FEC encoder to generate a certain level of redundancy, the FEC decoder needs to recognize these FEC packets and reconstruct the original stream. When such a FEC decoder is not present in the end-to-end path, this mechanism is useless. The FEC decoder does not need to be informed about the level of redundancy generated by the FEC encoder. Today, packet level FEC has been employed in several commercial products (e.g. in DVB technologies where Pro-MPEG FEC COP #3 [26] is implemented). In a Broadcast TV scenario, where videos are most commonly transported using multicast techniques, the concept of FEC activation can aid in adding redundancy on a per subscriber basis. A common example is to incorporate the FEC encoder in the access node, and the FEC decoder in the residential gateway.

To ensure such behavior, the APlane needs to know the capabilities of each node. For example, if only the service originator can generate FEC packets (due to computation limitations) the APlane should instruct each node in the path from service originator to the specific link to let these FEC packets through. In Section 6.1, we define an access network model that enables the APlane to execute two types of QoE optimizing actions: adding FEC and switching to a different video bit rate.

5.3. Knowledge base

During each phase of the autonomic loop (i.e. monitoring, reasoning and executing actions) more relevant information about the network is obtained. All this information is stored in the knowledge base. The current monitor input, as measured by the MPlane, is of prime importance here as it reflects the current network status. However, other data should also be taken into account to provide the KPlane with a complete view of the network situation. Typical examples are more historical data such as older monitor inputs and QoE optimizing actions taken in the past, but also characteristics of the network and its services. To come to a generic architecture, the myriad of knowledge is stored in a generic representation that can express the complete network and the QoE optimizing actions, which results in more scalability, another design goal. Using this representation, different optimization scenarios such as cross-service QoE maximization can be expressed in a flexible manner. A possible approach is to employ ontologies [27] to grasp the complete network configuration, as they provide a formal way of describing objects and their relationships, but also less formal ways such as an XML scheme can be a viable solution. More details about an ontology approach can be found in [28].

5.4. Knowledge Plane

The KPlane closes the loop between the MPlane and APlane. Here, the autonomic design goal is met as the KPlane determines the right QoE optimizing actions to take without any human interference. Furthermore, a learning controller enables the KPlane to self adapt its behavior if needed.

5.4.1. Determining the QoE optimizing actions

Determining which QoE optimizing actions to take based on MPlane information and other relevant data such as historical information and APlane configuration can be considered as the core functionality of the KPlane, as these actions will have a direct impact on the quality as perceived by the user. From our experience, pinpointing the problem and finding the best QoE optimizing actions is a complicated process. For a simple scenario, comprising a few monitor inputs and a few actions, it should be possible to construct a component that behaves well. However, when the complexity increases (e.g. by adding more scenarios and services) it becomes extremely difficult to make the right decisions in one global component. As will be explained in Section 6, we designed two reasoners that are able to maximize the QoE of all services for a simple scenario with only two QoE optimizing actions. For more complex scenarios we propose to split the KPlane functionality into different parts that can function independently and tackle different problems.

As illustrated in Fig. 2, we defined a three-step process where each step analyzes the knowledge currently available and forwards its decision to the next component in the chain. The first step in the decision process is the problem detection step: it detects suboptimal situations in the controlled network that lead to QoE drops. If a QoE drop is detected, the problem detection step forwards all the relevant knowledge to the next step, the problem tracking. Different problem tracking components exist, each one optimized for a particular problem domain. The problem
detection step needs to select the appropriate component to forward the knowledge data to. For example, one of the functions of the KPlane, is that it needs to take proactive actions to tackle a recurring problem. To support this requirement, the problem detection step can detect such a repeating problem based on historical data and trigger a specific problem tracking component optimized for detecting recurring QoE drops. In some cases, the reasoning component might decide that the knowledge that is currently available is not sufficient to determine the appropriate actions to take. In such a case, it can decide to perform additional tests through active monitoring. For example, when the reasoner suspects that a machine is not responding anymore, it can send out ping requests to verify this hypothesis.

In the problem tracking step, the location and the type of the problem is determined by investigating the knowledge received from the previous step. For example, by investigating the knowledge available for each link, the problem tracking might detect a high amount of packet loss together with a cumulative throughput of all services that is close to the link speed capacity. Based on this information, the problem tracking detects congestion on the xDSL line and should forward the relevant knowledge to the final step, the problem solving. As the reasoning process and the number of possible actions to take can differ depending on the type and location of the anomaly, different problem solving components exist and the problem tracking step needs to trigger the appropriate problem solving component. Different network anomalies can occur on different links at the same time. The problem tracking triggers a problem solving component for each anomaly.

The final step in the decision process is the problem solving step. Each problem solving component needs to determine the best actions for a specific anomaly. As the overall MPlane, KPlane and APlane architecture is distributed, both local and global problem solving components are needed. Local components only affect a specific node, while global components can alter the network characteristics by deploying actions everywhere in the network, hence allowing performing cross-service optimizations. In some cases, the implementation of such a problem solving component can be very straightforward. For example, when the only solution to a low priority service causing congestion is stopping the service, a problem solving component that tackles this problem only needs to trigger an action that prevents this service from reaching its destination. In other cases, for example when there is an unclear mapping between the input parameters and the right action to take, more advanced techniques will be necessary. However, as learning capabilities are a vital part of the KPlane functionalities, the problem solving component should always be able to change the details of its behavior. In Section 6 two problem solving components are presented that are both capable of solving the same problem: tackling congestion and random loss in the home network or xDSL line by applying FEC or switching to a different video bit rate. However, only the neural network approach has built-in capabilities to perform online learning.

5.4.2. Learning controller

The second part of the reasoning component incorporates another design goal: the learning capabilities of the KPlane. As discussed in Section 4, the learning behavior is twofold. First, it should detect new services. For example, it can occur that the KPlane is trained to optimize only one specific service (e.g. a Broadcast TV service when only Broadcast TV services run over the network). However, other services, such as on-line gaming applications, may need another policy to maximize their QoE. If such a new service is introduced, the KPlane should detect such a new service and detect the right actions to take through trial and error.

Second, it should detect and alter wrong decisions. Such a situation can occur when the configuration of the network is altered (e.g. when the user adds a wireless IEEE 802.11b device that slows the current IEEE 802.11g home network down). In this case, it can occur that the old reasoning process leads to wrong decisions. The KPlane should detect this and try to correct these decisions, again by trying different configurations, to ensure that the right decision is immediately taken the next time the same problem occurs. To this extent, the learning controller defines a cost function that expresses the penalty obtained by executing the last action. Basically, this cost function can be considered as some kind of metric that expresses the decrease in quality of each service. The goal of the learning controller is to minimize the cost function at all time by altering parameters in the three-step decision process. For example, when a sudden increase in this cost function occurs, this is a sign to alter the decision making process as the current reasoning steps no longer give satisfying results. The learning controller can then try randomly chosen configurations or predefined backup configurations until better results are obtained. Although the learning controller should not alter the configuration of services that do not suffer from a drop in QoE, it is reasonable that the KPlane attempts a few configurations on services that do, as long as in the end a better quality is obtained. This justifies the use of algorithms that apply some random behavior to come to a viable solution. A possible approach to achieve this learning capability, is by applying the paradigms of Q-learning [29] or other reinforcement learning techniques.

Currently, we have developed two problem solving components that both handle the case of tackling congestion on the xDSL line and random loss in the home network or on the xDSL line. In this case, we have applied the KPlane on the access node. This is useful as, in an xDSL network, it is the point nearest to the user that is under control of the network operator. Expanding this architecture to a fully distributed architecture is part of future work. Both problem solving components and the access network model in which they can be applied are discussed in the next section.

6. KPlane reasoning implementation

In this section, we will describe how we have implemented the reasoning behavior for a selected access net-
work model. We have designed an analytical component that determines the correct actions to take by solving a number of mathematical equations. A second component makes use of neural networks to determine which actions to undertake.

6.1. Access network model

In this section, we describe an access network model where the QoE of video services can be optimized through different actions. In this model, a number of Broadcast TV (BCTV) channels are being streamed in the access network. However, in contrast to a typical BCTV service, the streaming server sends out a number of streams for each channel that allow enhancing the QoE by taking action on the access node. These streams can be divided into two classes:

- For each channel the server sends out several versions that have a different bit rate. The advantage of having several channel versions, each with a different bit rate, is that sending a lower bit rate to the client allows the solving of the congestion problem on the access line.
- The server also sends out a stream of Forward Error Correction (FEC) packets, for each bit rate version of each channel. The advantage of using FEC is that this technique enables the client to restore the data packet stream in case a number of data packets are lost at the cost of additional traffic. We use optimal erasure codes, where the reception of $N$ packets out of $N$ data packets and $K$ FEC packets allows the receiver to reconstruct the original $N$ data packets.

The concept of using multiple channels at different bit rates can be seen as using the simulcast paradigm on the video level [30]. Using such a scheme in a Broadcast TV scenario enables optimization of the QoE on a per subscriber basis. A simulcast technique results in an overhead between the video streamer and the access node. The SVC standard, as specified in Annex G of H.264/AVC [31], allows reducing this overhead by streaming a video where quality levels can be dropped easily. While such a technique is certainly better suited, we choose to use the more traditional simulcast technique for two reasons. First, as we focus on the last mile, the behavior between the video streamer and access node are of lesser importance. The choice between a simulcast based technique or SVC does not affect the decision making process itself: it is merely a different implementation of the same kind of QoE optimizing action. Second, at the time of writing, we believe that the SVC standard is not mature enough. To our knowledge, no suitable combination of streamer, player and stream modifier exist today to allow the SVC behavior in real-time.

The MPlane gathers monitor information on the packet loss between the access node and the end-device. It also receives information on the bandwidth that is being used on the access line. The APlane in this model has the ability to perform two types of actions: selecting which bit rate version is streamed to the client and determining how many FEC packets are sent to the client device. By performing these actions, the QoE management component is able to solve problems that occur due to congestion or due to random packet loss on the xDSL link. In this model, the KPlane component is located on the access node and needs to determine which bandwidth is used for each BCTV channel that is watched by users in the home network and the correct amount of FEC packets that should be sent to the user for each individual channel. There is a trade-off in determining the values of these parameters. The KPlane component should try to find the solution that minimizes the bandwidth that is used on the access line but maximizes the global QoE of the services. For instance, it should not send more FEC packets than required to the end-device.

Fig. 3 shows an example of the actions that can be taken in the access node. The KPlane component decides that the client should receive a high bandwidth video and a number of FEC packets. This can correspond to a scenario where there is packet loss on a link in the home network or on the xDSL line and where enough bandwidth is available on the access line to support both the high bandwidth version of the BCTV channel and the FEC packets. Only some of the generated FEC packets are streamed to the client, as no more are needed to solve the packet loss.

6.2. Analytical reasoner

We have developed an analytical reasoner that uses a number of equations to determine the amount of FEC packets that is required and the video bit rate that is selected for each channel. The reasoner finds the correct values in two phases. First, the reasoner uses the monitor input to determine whether the access line is congested or not. If it is congested, the reasoner sets the number of FEC packets to 0 for the active services. If there is no congestion, the reasoner determines the amount of FEC that is required to solve the packet loss occurring between the access node and the client for each connection. Once the amount of FEC packets has been determined, the analytical reasoner determines which bit rate should be selected for the channels.

![Fig. 3](image-url) The access network model. In the access node, it is decided that the client should receive the high bandwidth channel and a number of FEC packets.
Eq. (1) is used for determining the probability $Z_k$ that up to $K_i$ packets are lost out of a total of $N_i + K_i$ packets for service $i$ when $Loss_i$ packet loss was detected. The probability that $K_i$ FEC packets do not suffice for service $i$ is thus $(1 - Z_k)$. The analytical reasoner determines the value of $K_i$ for each channel $i$ such that the value for $(1 - Z_k)$ is lower than a predefined threshold:

$$Z_k = \frac{(N_i + K_i)!}{J!(N_i + K_i - J)!} \cdot (1 - Loss_i)^{N_i + K_i - J}.$$  

(1)

Once the FEC rate is calculated, we need to determine which bit rate is selected for each flow. To determine the bit rate, we start by first assigning every service the lowest bit rate. Then we go through the connection set and try to select one higher bit rate for each flow. This is repeated until either all the flows are assigned the highest possible bandwidth or there is no stream left for which Eq. (2) holds:

$$\frac{N_i \times AvailBW_i}{K_i + N_i} \geq Bitrate_i^{m - 1}.$$  

(2)

In this equation, $N_i$ and $K_i$ denote the values of the number of data and parity packets for the FEC stream and $AvailBW_i$ contains the available bandwidth that may be consumed for channel $i$, taking into account the bandwidth that was already assigned to all the other services. The parameter $Bitrate_i^m$ denotes the bit rate value $m$ for service $i$. This is the last bit rate that was selected for service $i$ and will be chosen if Eq. (2) is no longer valid. More specifically, if it is no longer possible to increase the bandwidth for service $i$ without violating the available bandwidth for the service.

The analytical reasoner is straightforward but has some limitations. The use of this type of reasoner is limited to this specific case. While it is possible to extend the equations as more actions and monitor inputs are available, the complexity would increase rapidly. Another complication is the learning aspect: as the equations cannot be altered, the analytical reasoner is not able to learn new actions or modify existing ones to adapt its behavior to the network it is deployed in. All these aspects make the reasoner a far from scalable and extensible solution. While an analytical reasoner can be constructed that behaves well for a specific scenario with well defined actions, more complex techniques are needed to provide a generic solution, that includes the ability to add new actions on-the-fly and that incorporates a learning mechanism.

### 6.3. Neural network based reasoner

We designed a second reasoner, based on neural networks, that is also applicable to the access network model described in Section 6.1. A neural network can be considered as a black box able to solve complex problems, which makes it a perfect tool for designing a component that makes complex decisions that are difficult to cast in rules or equations. In supervised learning, the employed neural learning technique for this reasoner, the network can be trained by presenting example input–output pairs. For more information about neural networks we refer to [32]. We constructed a feed-forward neural network consisting of one hidden layer, with five hidden neurons, and used the Levenberg–Marquardt [33] algorithm to train the network. The activation function of the input layer was a linear function, while the activation function of both the hidden and output layer was a sigmoidal function. Furthermore, we applied the early stopping algorithm to avoid overfitting. These parameters were experimentally determined [4].

As illustrated in Fig. 4, the neural network tries to map a set of monitor values to the configuration of the two possible QoE optimizing actions to take. The network consists of six input neurons and two output neurons. The six input neurons represent the measured packet loss before and after FEC restoration, the bandwidth used by the service, the residual bandwidth and the current configuration of the APlane. The two output neurons represent the configuration of both actions: the bit rate at which the video is streamed and the amount of FEC that is used to protect this stream.

To train the neural network, we constructed a training set consisting of desired input and output pairs, using the simulation environment described in Section 7.1.1. To construct this training set, we evaluated the effect of each APlane configuration in different scenarios (e.g. random packet loss in the wireless network, congestion on the xDSL line). The video quality was measured using the Peak Signal to Noise Ratio (PSNR) for each possible combination. The use of the PSNR as a video quality metric is described in Section 3. For each scenario, we selected the APlane configuration that resulted in the highest PSNR value and added this configuration to the training set. Building such a training set through simulation is an effective solution when it is hard to cast the mapping between the monitor values and the QoE optimizing actions. However, if such a mapping is easy a training set can also be constructed by simply giving some exemplary input–output patterns.

#### 6.3.1. Extending with on-line learning capabilities

In this section, we discuss how the neural network reasoner can be extended to incorporate on-line learning behavior. The techniques presented here are based on existing search techniques such as Q-learning [29], genetic algorithms [34] and simulated annealing [35]. All these algorithms incorporate some random behavior when trying to look for the optimal solution. Here, we try to describe how this random behavior can be applied for our neural network reasoner.

![Fig. 4. The neural network reasoner maps the monitor values, measured through the MPlane, to an APlane configuration. This mapping is done using six input neurons, representing the monitor values, and two output neurons, representing the actions.](image-url)
In Section 4, we discussed that the learning controller should be able to adapt itself if it detects that it is making the wrong decisions. Therefore, we described the concept of a cost function to evaluate the decision making process in Section 5.4.2. Each time the APlane executes a QoE optimizing action, the MPlane provides monitor data that expresses the network response to this action. Based on this monitor data a cost function can be defined per-service, that expresses the immediate cost of executing this specific action. For example, for a video service a basic cost function can be to express the packet loss ratio, while for an on-line gaming application a combination of latency and packet loss ratio can be used. If this cost function increases, the service must be suffering from a drop in quality. When such a drop occurs, the learning controller should interfere and change the decision making process.

This cost function defines the goal of the learning controller. The goal is to minimize the total cost the component receives in the long run. Therefore, a second function is defined, the value function, which can be seen as the total cost in the past and the future. If \( C_t(a,n) \) is the cost of executing a set of QoE optimizing actions \( a \) into the network \( n \) at time \( t \), the value function \( V \) can be seen as

\[
V(n) = \sum_{t=\infty} V_t(n) \tag{3}
\]

The goal is to find the correct set of consecutive QoE optimizing actions that minimize \( V \). As more QoE optimizing actions are taken, more knowledge becomes available and the value of this value function can be approximated. Different algorithms exist to approximate and minimize this value function by trying random actions [29].

We propose to implement the on-line learning behavior as follows. If the learning controller in the architecture detects a sudden increase in the cost function it initiates a search for a better decision process. In such a search, the learning controller should try different QoE optimizing actions on the service until a better quality is obtained (measured through the cost function). The learning controller only tries different actions on the affected service to avoid a quality decrease of other services. The tried QoE optimizing actions can have a positive or negative effect. However, we believe that we can tolerate another drop in quality for a service that is already suffering from a considerable loss in quality, as long as we find a solution in the end. In this search procedure, the actions that have a positive effect on the service are presented to the neural network reasoner together with their original monitor data as a way of supervised learning.

Trying these random actions comes with a cost. First, it can take some time before a QoE optimizing action can be found that optimizes the services QoE. Second, as each possible QoE optimizing action is applied in the network, the user can see its effect which can sometimes be troublesome. Therefore, some extensions to the search procedure are possible. First, instead of trying random QoE optimizing actions, the operator can define some predefined configurations that have a good chance of optimizing the QoE of the service. If these predefined actions fail, the search procedure can continue by randomly trying different configurations. Second, instead of trying these different configurations in a real network where the user can see the effect of each action, they can first be executed in a simulation environment. Based on these results, a subset of possible configurations can be derived that have a positive effect on the service. Then, this subset can be used to choose the final QoE optimizing action to take by validating the results in the actual network. Even more advanced techniques, where different configurations of the simulation environment itself are employed and the best is selected using a genetic algorithm, can be a viable approach.

The neural network approach has two main advantages with respect to the analytical approach. First, as a neural network is a black box, less effort is needed to come to a working solution. It is not necessary to design a set of equations by thinking about the interactions between the different QoE optimizing actions. It suffices to construct a training set through trial and error (e.g. using a simulation). Second, as a neural network has built-in learning capabilities, it is much easier to alter its behavior if needed. It suffices to present the neural network with a new training set. In the analytical case, the equations need to be altered which can be a complex task.

Furthermore, as will be explained in the next section, the performance of the neural network approach matches the performance of the analytical reasoner. We tested both approaches in different scenarios by measuring the packet loss ratio and video quality at the user. In each scenario, the obtained values were close to identical.

7. Results description

7.1. Test setting

To test the performance of both reasoners we implemented the access network model as described in Section 6.1. The employed topology is illustrated in Fig. 5. In this topology two receivers can access different video services that are streamed from the video server over the access network. As explained in Section 6.1, the video server streams both a high quality (with an average bit rate of 2 Mbps) and a low quality (with an average bit rate of 500 kbps) video to the clients. We investigated three different scenarios. In the first scenario, only one receiver accesses a video service but experiences some random packet loss on the wireless link. This random loss was varied from 0% to 10% in steps of 0.5%. In the second scenario, the wireless link still experiences a certain level of random packet loss, which is again varied as in the first scenario, but now a second receiver also accesses a video service.

![Fig. 5. The topology used to test the performance of the analytical and neural network reasoner. By altering the number of users and the inflicted packet loss, three different scenarios were introduced.](Image)
Due to the demand of this client, the xDSL line becomes congested. We assume that the second receiver accesses a higher priority video service than the first receiver, which limits the available bandwidth for the first receiver to approximately 1.5 Mbps. Therefore, the reasoners should decide to lower the video bit rate of the first video if congestion occurs. In a third scenario, we assume the user has sufficient bandwidth available to receive the highest video bitrate but the link between access node and base station suffers from bursty packet loss. We varied the burst size as follows: the length of a bad period was fixed to 12.5 ms, while the length of a good period was varied from 100 to 500 ms with steps of 25 ms. Such a loss scenario matches the SHINE model for impulse noise on a DSL line [36].

We used a threshold value of 0.01 for the analytical reasoner. Each test was repeated 55 times, here we present average values. The reasoners were verified in a simulation environment and in a practical testbed. In the practical testbed we measured the packet loss ratio. In the simulation environment we measured the packet loss ratio and the video quality through the PSNR and SSIM. To prove that the performance of the reasoners has a significant difference or not we performed one-way ANOVA tests on all the results. The two test environments and the results of the tests are described in the next sections.

7.1.1. Simulation environment

To simulate the different scenarios, we modified the NS-2 [37] simulator to enable the delivery emulation of real video sequences over a simulated NS-2 network. In this environment, the packets that are being transmitted over the simulated network do not contain real data but are dummy packets based on trace files of the video. These trace files can be generated using a dump of the generated traffic of the video streamer. On the receiving end, the video can be reconstructed based on additional trace files generated by the simulation environment. The original and the received video can then be used to calculate the video quality using an objective video metric or subjective tests. The used simulation environment works with virtually any video streamer, player and codec because there is a loose coupling between video streamer and player on the one hand and the simulated network on the other hand. The main advantage of this environment is the speed by which different scenarios can be investigated. For example, simulating 105 h of video playback can be simulated in approximately 4 h using a 1800 MHz AMD Athlon machine with 512 MB of memory.

7.1.2. Practical testbed

Thanks to the simulator environment, a lot of tests can be run within a short period. However, to obtain experimental results, we constructed a practical testbed to emulate an access network. In this testbed, the videos are streamed over the network and a Reed-Solomon FEC code is used. This testbed is depicted in Fig. 6. The testbed emulates a xDSL-environment, comprising a video streamer, an access node and two clients in a single home network. To introduce packet loss on this home network we employed an impairment node based on the Click modular router [38]. The two clients may compete for the limited bandwidth on the xDSL link, which can also be defined through the impairment node.

7.2. Packet loss ratio

Fig. 7 illustrates the measured packet loss in the simulation environment for the random and burst loss scenarios. We measured the packet loss ratio after FEC correction is applied for the case without reasoning, with the analytical reasoner and with the neural network reasoner. In the random scenario (Fig. 7a) both the neural network reasoner and the analytical reasoner are able to reduce the packet loss to almost zero by adding FEC packets to the stream. Here, the x-axis denotes the amount of packet loss introduced random packet loss on the link (%)

![Fig. 6. The practical testbed that was used to obtain experimental results. The impairment node can introduce losses or a limited amount of bandwidth.](image)

![Fig. 7. Measurements of packet loss experienced by the user in the simulation environment for different injected network packet loss: random packet loss (top) and bursty packet loss (bottom).](image)
that was introduced into the network (this is the packet loss that can be measured before FEC correction), while the y-axis is the measured packet loss after FEC correction. In this case, the performance of both reasoners can be seen as almost identical. ANOVA tests showed that there is no significant difference between these two reasoners. This is obvious as both reasoners have been optimized to tackle random packet loss. The analytical reasoner assumes that the packet loss has a random distribution in its probability calculation (Eq. 1 in Section 6.2), while the neural network has explicitly been trained with training data gathered after applying random packet loss.

In the bursty scenario, there is a difference in the performance of both reasoners as illustrated in Fig. 7b. In this figure, the x-axis denotes the length of the good period. As this length increases along the x-axis, fewer bursts will occur and the actual measured packet loss ratio will decrease. The analytical reasoner prioritizes the minimization of packet loss by calculating the FEC level first and then filling up the pipe with the highest bitrate possible. The neural network based reasoner tries to do both at once. In fact, the analytical reasoner pays more attention to the optimization of QoS parameters (in this case: packet loss), while the neural network based reasoner tries to optimize the QoE itself, because it has been trained with objective video quality measurements. This can be seen in the results regarding packet loss in Fig. 7b: the analytical reasoner generates a higher FEC level than the neural network reasoner and is therefore able to obtain a lower packet loss ratio than the neural network reasoner. Thus, solely based on QoS parameters, the analytical reasoner has a better performance in the bursty scenario. ANOVA tests show that in all measurement points, there was a significant difference between the two reasoners with a confidence level of 99.9%.

We tested the random and congestion scenarios in the practical testbed. As illustrated in Table 1, the obtained results are very similar to the ones obtained through simulation. In both scenarios, applying the analytical or neural network results in a measured packet loss ratio that is close to zero. In the random scenario, this is achieved by adding FEC, while in the second scenario the reasoner switches to a lower bitrate and adds FEC as the amount of random link increases. When the packet loss on the wireless link increases, it can sometimes occur that a packet is lost but the measured packet loss ratio values are always lower than 0.02%. Again, the differences in performance between the analytical reasoner and the neural network reasoner is negligible. Similar to the simulation based tests, applying ANOVA tests showed that there is no significant difference between the neural network and analytical reasoner in both scenarios.

### 7.3. Video quality

Packet loss is a parameter that reflects the objective Quality of Service (QoS) of the network. However, the user is more concerned with the subjective Quality of Experience (QoE). As discussed in Section 3, there is a relationship between these QoS parameters but it is not possible to find a clear mapping. To assess the video quality, we calculated the QoE of the received videos using both the PSNR and SSIM. We streamed four different types of content over the simulated network: a high motion football game, a fast scene switching music video clip, a static news broadcast and a nature documentary, containing many complex pictures. Each movie was encoded as an H.264 video. The videos that are streamed by the video streamer are different videos than those used to train the neural network KPlane. This is to show that the neural network based approach is able to generalize from the given training set, and is not trained for one specific video. For each content type, the trends were similar. Here, we present average values: the standard deviation between the different content types was always lower than 1.5 dB for the PSNR and 0.05 for the SSIM.

#### 7.3.1. PSNR

We have measured the PSNR in all three scenarios: random, congestion and bursty. The results are presented in Fig. 8. In the random scenario (Fig. 8a), we see how the minimization of the packet loss, as illustrated in the previous section, also leads to a stabilization of the QoE for an increasing amount of introduced packet loss. Without any QoE optimizing actions the PSNR value of the video quickly decreases below 30 dB at around 1% packet loss and even drops below the 20 dB value at around 4%. Hence, without any corrective actions, only a packet loss up to 1% can really be tolerated. The activation of a reasoning component clearly has a beneficial effect. Even for a packet loss around 10%, both the neural network and the analytical reasoner are able to keep the PSNR at almost the same level as when there was no loss. ANOVA tests pointed out that there is no significant difference between both reasoners. Their performance can therefore be seen as identical.

The results are similar in the congestion case (Fig. 8b). In this figure, the amount of introduced random loss is increased on the x-axis, and the corresponding PSNR value is measured and plotted on the y-axis. Here, the PSNR drops below 15 dB if no reasoning is applied which can be considered extremely bad. Of course, this is expected behavior as the corresponding packet loss ratio is more than 20%. When the neural network or analytical reasoner are applied, the obtained PSNR values are still very reasonable. In this case, the PSNR value is around 31 dB which is somewhat lower than the 35 dB in the first scenario. However, this is due to the fact that the reasoners have switched to a lower bit rate. Even if the amount of random loss on

| Table 1 |
| Measurements of average packet loss experienced by the user in the practical testbed for different injected network packet loss: pure random packet loss and a combination of random packet loss and congestion. |
| Packet loss on the wireless link | No reasoning (%) | Analytical reasoner (%) | Neural network reasoner (%) |
| Random scenario | | | |
| 0 | 0.0 | 0.0 | 0.0 |
| 5 | 5.56 | 0.011 | 0.009 |
| 10 | 10.05 | 0.018 | 0.015 |
| Congestion scenario | | | |
| 0 | 22.53 | 0.0 | 0.0 |
| 5 | 27.43 | 0.011 | 0.012 |
| 10 | 31.95 | 0.013 | 0.013 |
the wireless link is increased, the PSNR values stay stable as more FEC packets are added. As no considerable errors in the transmission occur thanks to the generated redundancy, the received QoE is almost identical to the QoE of the original lower quality video: 31.23 dB. Similar to the random case, we see that both reasoners are able to stabilize the QoE value but their performance is different. Although the analytical reasoner is able to obtain a lower packet loss by adding more FEC packets than the neural network based reasoner (as described in the previous section) it does not achieve a better PSNR value than the neural network reasoner. The high level of redundancy generated by the analytical reasoner that resulted in a lower packet loss eliminates the use of the highest bitrate. On the other side, the neural network based reasoner suffers from a higher packet loss but still chooses to transmit a higher video bitrate. Fig. 8c shows that the latter is more beneficiary to the PSNR value of the video. The neural network reasoner is able to achieve an average PSNR value of 3–5 dB higher than the analytical reasoner. ANOVA tests indicated that there is a significant difference between the values obtained from the neural network reasoner and analytical reasoner with a confidence level of 99.95%. The neural network reasoner can therefore be seen as a QoE optimizer, while the analytical reasoner is more a QoS optimizer.

7.3.2. SSIM

We calculated the SSIM score for the random and bursty scenario, the results are illustrated in Fig. 9. The SSIM scores confirm the same trends as described in the previous section. In the random scenario (Fig. 9a), the SSIM

![Fig. 8. Measurements of video quality, in terms of PSNR, for different injected network packet loss: pure random packet loss (a), a combination of random packet loss and congestion (b) and bursty packet loss (c).](image1)

![Fig. 9. Measurements of video quality, in terms of SSIM, for different injected network packet loss: pure random packet loss (top) and bursty packet loss (bottom).](image2)
score quickly drops to an unacceptable level (below 0.75 at 1% loss and beyond) if no reasoner is applied. The use of the analytical reasoner or neural network reasoner leads to a stabilization of the SSIM score. In this case the analytical reasoner and neural network reasoner both have a SSIM score of approximately 0.94 which corresponds to almost perfect quality. The performed ANOVA tests indicate that, similar to the previous results in the random scenario, there is no significant difference between both reasoners.

For the bursty scenario (Fig. 9b), there is a difference in behavior between the two reasoners. Again, the SSIM tests confirm the results obtained in the previous section. Applying one of both reasoners has a positive effect on the SSIM score, as the SSIM increases on average with 0.2, but the neural network reasoner is able to obtain a better SSIM score. While the SSIM score of the neural network reasoner is not a lot higher, ANOVA tests point out that there is a significant difference between both reasoners on all measurement points with a confidence level of at least 99.95%.

These differences point out an interesting aspect of QoE. As discussed in Section 3, there is no clear mapping between QoS and QoE. This is confirmed through the results in Fig. 9b. When only looking at the measured packet loss, one could assume that the QoE of the analytical reasoner would be the highest. However, the opposite is true as we also take into account the bitrate of the video. In the analytical reasoner, the received video will have a lower bitrate, and hence an overall lower quality to start with but will only have limited visual artefacts in the video due to packet loss. When applying the neural network reasoner, the received video is a high bitrate video but with more visual errors due to packet loss. While neither approaches are optimal in the bursty scenario, the question is which video an end-user would rank the highest: a low bitrate video with an occasional error, or a high bitrate video with a few more errors. This is a highly subjective question, but both the SSIM and PSNR metrics indicate that the latter video, obtained through the neural network reasoner, has a higher QoE.

8. Discussion

As described in Section 5, an architecture was defined to optimize the QoE in multimedia access networks. The architecture forms a fully autonomic loop. No human interference is necessary to obtain a normal behavior as the architecture continuously monitors the network, reasons about which actions to take and executes these actions. Furthermore, as discussed in the previous section, a learning controller makes the architecture self-adaptive and introduces learning behavior. The third autonomic aspect, both proactive and reactive behavior, is reached as different components can exist in the three-step decision process. For example, for the proactive case, the problem detection step can also investigate historical data to detect a recurring anomaly and then call a problem tracking component that specifically deals with repeating problems over time.

The overall KPlane architecture has a modular structure to ensure a generic design. To simplify the decision making process, a three-step approach is used. Each step can be divided in many modules that all tackle a simple and well-defined problem, and hence pave the way to one global approach. When a new problem should be tackled or modified (e.g. because a new QoE optimizing action is available that does a better job in tackling the problem), it should suffice to add or alter one or more modules instead of the complete system. Adding new monitor inputs and possible QoE optimizing actions to be taken should be straightforward through the use of one global and generic representation.

Extendibility is achieved through the component-based design and the generic representation that defines the knowledge. The component-based design of the architecture allows components to be plugged in and plugged out when needed. The use of a generic knowledge representation allows grasping the inner workings of the system. By distributing the KPlane among different nodes in the architecture, more scalability can be achieved when compared to a purely centralized approach. For example, in the access network model described in Section 6.1, the KPlane can be deployed on several nodes in the tree-like access network topology. Each KPlane can decide which FEC level is needed for the nodes downstream. In this case, the QoE can be optimized on a per-service basis as every node will be responsible for a limited set of subscribers. As we target a fully distributed system that spans the complete network from service originator to user, we foresee having both local and global problem solving components. Local components only alter the specifics of one node, while global components can alter configurations in the complete network. To ensure that one component will not disable the actions taken by another component, the behavior of the different components will be orchestrated.

The proposed reasoners can be seen as a proof of concept for a specific access network model, where the QoE of video services can be optimized by adding FEC or switching to a different bit rate. This model is just an example of how an operator can choose to optimize the services running on its network from the broad set of QoE optimizing actions available. Also other access network models are possible such as managing a Call Admission Control mechanism to prevent congestion in the network of VoIP services or finding the optimal quality for each user in a multicasted Scalable Video Coding stream.

As described in Section 7, both reasoners succeed in optimizing the QoE of the video services for the random and congestion scenario. In the bursty scenario, applying the neural network reasoner instead of the analytical reasoner leads to a higher QoE. Because the neural network reasoner is a black box concept, less effort is needed to come to a working solution. Finding a good training set to train the neural network can take more time than defining a good set of equations but the former is a process that can be automated, while the latter is not. Furthermore, neural networks have learning mechanisms built-in. For the neural network reasoner, on-line learning behavior can be achieved using a reinforcement learning technique as discussed in Section 6.3. This is harder for the analytical reasoner as there is no real learning mechanism built-in and changing the behavior means changing the equations. Therefore, we see the neural network reasoner as the most promising solution.
9. Future work

Our work has mainly focused on a single Knowledge Plane instance on the access node, close to the user. This is a natural choice in a hierarchically, tree-like structured access network, where most problems can be solved locally. However, this limits the action range of the Knowledge Plane and can lead to suboptimal solutions. For example, when several clients are reporting packet loss for the same service, the cause is probably situated closer to the service originator and should be solved by a higher level coordination mechanism. Our future work will aim at the extension of the Knowledge Plane toward an autonomic platform encompassing all parts of the network and enabling more monitor inputs and output actions.

First, we will work toward a more distributed Knowledge Plane with communicating Knowledge Plane instances in the different parts of the network: the access node, aggregation switches, edge routers, etc. This not only involves the exchange of knowledge, e.g. which services are currently underperforming, but also of the actions taken and initiating a mutual learning process between KPlane components.

Second, the inclusion of user QoE patterns will be incorporated. The Knowledge Plane will grasp the user patterns, such as bandwidth usage, typical services and the home network configuration. For example, the KPlane could detect that every evening, the user requests via VoD the evening news. The KPlane could take this into account and optimize its decision process to limit the bandwidth of the download that is occasionally started some minutes before the news.

10. Conclusion

We defined an autonomic management architecture to optimize the QoE in multimedia access networks using a three plane approach. This three plane approach consists of a Monitor Plane that monitors the network and builds up knowledge about this network, a Knowledge Plane that analyzes the knowledge and determines the ideal QoE actions, and an Action Plane that enforces these actions into the network. We focused on the Knowledge Plane inside the architecture and defined how this architecture can achieve three main design goals: autonomic behavior, a generic design and scalability.

As a proof of concept, we designed two reasoners that are part of the KPlane architecture: an analytical reasoner based on a set of equations and a neural network based reasoner. These reasoners can optimize the QoE of video services in multimedia access networks for an access network model with two optimizing actions: applying FEC to reduce the packet loss caused by errors on a link and switching to a different video bit rate to avoid congestion or to obtain a better video quality.

We tested the performance of both reasoners in terms of packet loss ratio and PSNR using a modified version of the NS-2 simulator and a practical testbed. The simulation environment increases the number of tests that can be done in a limited amount of time, while the practical testbed can be used to obtain experimental results. We investigated three different scenarios: in the first two scenarios the last mile is only suffering from packet loss due to bit errors (using a random or bursty distribution) while in the third scenario the last mile is suffering from both congestion and random packet loss. The results show that both reasoners are capable of increasing the video quality and lowering the packet loss ratio when packets are lost due to bit errors or when congestion occurs. In fact, the performance of both reasoners can be seen as identical in the random and congestion scenario. However, in the bursty scenario the analytical reasoner cannot achieve the same QoE score as the neural network reasoner because it only optimizes QoS (through the minimization of packet loss) and not QoE. Because of these reasons and as the neural network reasoner has more capabilities to support on-line learning behavior, the neural network reasoner is the best candidate to achieve real autonomic behavior.

Acknowledgements

The research was performed partially within the framework of the MUSE-project, funded by the European Commission as part of the IST 6th Framework Program. The authors would like to thank all MUSE partners for their valuable contributions and feedback.

References


[38] The Click Modular Router project (online).<http://www.read.cs.ucla.edu/click/>.

Steven Latré obtained a Masters degree in Computer Science from Ghent University, Belgium, in June 2006. Since August 2006, he is affiliated as a Ph.D. student with the Department of Information Technology at Ghent University. He is funded by grant of the Fund for Scientific Research of Flanders, Belgium (F.W.O.-V.). His main research interest is the optimization of multimedia services through autonomic communications. He was involved in the IST FP6 project MUSE and is currently involved in the EUREKA CELTIC project RUBENS.

Pieter Simoen received his M.Sc. degree in Electronic Engineering from Ghent University, Belgium, in July 2005. Since October 2005, he is affiliated as research assistant with the Department of Information Technology of Ghent University. He is funded through the Fund for Scientific Research of Flanders, Belgium (F.W.O.-V.). His main research interest is the development of intelligent and adaptive network architectures and protocols designed for thin client computing and autonomous environments. He cooperated in the IST FP6 project MUSE and is now work package leader for the protocol design in the IST FP7 project MobiThin.

Bart De Vleeschauwer obtained a Masters degree in Computer Science from the Ghent University, Belgium, in June 2003. Since August 2003, he is affiliated with the Department of Information Technology of the Ghent University. He is funded through autonomic communications. He was involved in the optimization of multimedia services through autonomic environments. He cooperated in the IST FP6 project MUSE and is currently involved in the EUREKA CELTIC project RUBENS.

Wim Van De Meerssche received his M.Sc. degree in software development in 2004 from Ghent University, Belgium. In August 2004, he started working on software technologies for access networks in the Department of Information Technology (INTEC), at the same university. His work has been published in several scientific publications in international conferences. He was involved in the IST FP6 project MUSE and is currently involved in the EUREKA CELTIC project RUBENS.
Filip De Turck received his M.Sc. degree in Electronic Engineering from Ghent University, Belgium, in June 1997. In May 2002, he obtained the Ph.D. degree in Electronic Engineering from the same university. At the moment, he is a full-time professor affiliated with the Department of Information Technology of Ghent University. His main research interests include scalable software architectures for telecommunication network and service management, performance evaluation and design of new telecommunication services.

Bart Dhoedt received a degree in Engineering from Ghent University in 1990. In September 1990, he joined the Department of Information Technology, Ghent University. His research, addressing the use of micro-optics to realize parallel free space optical interconnects, resulted in a Ph.D. degree in 1995. After a 2-year post-doc in opto-electronics, he became Professor at the Department of Information Technology. He is responsible for courses on algorithms, programming and software development. His research interests are software engineering and mobile & wireless communications. His current research addresses software technologies for communication networks, peer-to-peer networks, mobile networks and active networks.

Piet Demeester received the Masters degree in Electro-technical Engineering and the Ph.D degree from Ghent University, in 1984 and 1988, respectively. In 1992, he started a new research activity on broadband communication networks resulting in the IBCN-group (INTEC Broadband communications network research group). Since 1993, he became Professor at Ghent University where he is responsible for the research and education on communication networks. The research activities cover various communication networks, including network planning, network and service management, telecom software, internetworking, network protocols for QoS support, etc.

Steven Van den Bergh graduated as Master in Computer Science at Ghent University in 1999. In the same year, he joined the Broadband Communication Network Group at Ghent University, where he performed research in the area of quality of service, traffic engineering and monitoring in IP networks, for which he was granted an IWT scholarship. This research resulted in the award of his Ph.D. in 2005. He joined the Alcatel-Lucent Bell Labs Wired Access team in 2006. His main topics of interest include design, control and management of multimedia and broadband service delivery networks.

Edith Gilon de Lumley graduated in Electrical Engineering in 1998 at the UCL, Belgium. After her graduation she joined Alcatel-Lucent Bell and worked on Passive Optical Networks (PON) systems, especially the Physical Medium Dependent (PMD) layer from 1998, and she led the IST GIANT project from 2003 to 2005. She was coordinator of the IST MUSE sub-project B focusing on multimedia broadband.