Using Fuzzy Policies to Improve Context Interpretation in Adaptive Systems

Lucas Provensi, Frank Eliassen and Roman Vitenberg
Department of Informatics
University of Oslo, Norway
{provensi, frank, romanvi}@ifi.uio.no

Romain Rouvoy
Inria Lille-Nord Europe
University of Lille, France
romain.rouvoy@lifl.fr

ABSTRACT

Adaptation is an increasingly important requirement for software systems executing in large-scale, heterogeneous, and dynamic environments. A central aspect of the adaptation the methodology is management of contextual information needed to support the adaptation process. A major design challenge of managing contextual data lies in the fact that the information is partial, uncertain, and inherently suitable for diverging interpretations. While existing adaptation solutions focus on techniques, methods, and tools, the challenge of managing and interpreting ambiguous contextual information remains largely unresolved. In this article, we present a new approach to knowledge management in adaptation feedback control loops that aims to overcome these issues by applying fuzzy set theory and approximate reasoning. Our new knowledge management scheme interprets imprecise information and effectively integrates this information into the adaptation feedback control loop. To test and evaluate our solution, we implemented it in an adaptation engine to perform rate control for media streaming applications. The evaluation results show the benefits of our approach in terms of flexibility and performance when compared to more traditional methods, such as TCP-friendly rate control.[1]

Categories and Subject Descriptors

C.2 [**Computer-Communication networks**]: Distributed Systems

Keywords

adaptation; fuzzy rules; rate control;

1. INTRODUCTION

With the ubiquitous proliferation of large-scale software systems operating in highly dynamic environments, autonomous system execution in presence of everchanging operational settings and personalized user requirements become increasingly important. Design of modern software systems consequently emphasizes self-adaptive capabilities whose goal is to provide one or more of the self-* properties (self-configuration, self-healing, self-optimization, and self-protection).

In order to support the control tasks that constitute the adaptation process, there is a need to collect, store, and interpret relevant contextual information. A major design challenge of managing contextual data lies in the fact that the information is partial, uncertain, and inherently suitable for diverging interpretations. For example, media streaming applications may need to adapt the streaming protocol and its configuration parameters when the packet loss fraction becomes “high” or the amount of bandwidth available on a shared link becomes “low”. Yet, the threshold for being high or low depends on a great deal of factors and settings that vary at runtime whereas the assessment of runtime parameters (such as a packet loss fraction or available bandwidth) is essentially partial and imprecise. This challenge is further aggravated by the need to maintain the adaptation process over time and evolve its individual elements independently of the rest of the system: When new versions of the streaming protocol become available, it may be beneficial to start using them under the same conditions of packet loss and bandwidth. Monitoring of runtime parameters may need to be dynamically adjusted. New rules that trigger the adaptation can be added on the fly. If multiple rules can be applied to the same situation, then which rules to be applied need to be effectively resolved. In fuzzy logic this issue can be addressed by assigning relative weights to the rules and by dynamically adjusting the them. If the network link is upgraded or if the media streaming application updates the requirements (e.g., because of switching to a different encoder or a different video client), this will affect the interpretation of “low” bandwidth for the same rule.

In this article, we propose a complete yet highly modular approach for knowledge management in adaptive systems that facilitates interpretation of imprecise or vague contextual information. The approach is based on applying *fuzzy set theory* [23], which allows the separation of knowledge from its interpretation, so that the interpretation can be independently formulated and updated. Additionally, the knowledge management approach comprises a variation of the well-known MAPE-K control loop [8] adapted for dealing with fuzzy knowledge. The MAPE-K model aims at separating the concerns of the adaptation process from the

[1]Copyright is held by the authors. This work is based on an earlier work: SAC ’13 Proceedings of the 2013 ACM Symposium on Applied Computing, Copyright 2013 ACM 978-1-4503-1656-9/13/03. http://doi.acm.org/10.1145/2480362.2480447

application logic, improving the reusability of the middleware framework. The modularity of our approach promotes individual evolution of different aspects of knowledge management: formalization of a shared vocabulary, imprecise interpretation of concepts, definition of strategies/policies, and specification of the adaptation mechanism, as we further discuss and illustrate in Section 5.

While there exist a few works that employ principles of fuzzy set theory in connection with adaptation, each of these works only focuses on a particular adaptation aspect without considering how fuzzy set theory can be weaved into the entire adaptation control loop in a modular way. In [5, 22, 10] the authors implement heuristic controllers based on imprecise information. However, these controllers are based on fixed ad-hoc knowledge management, which might compromise the solution reusability and its ability to evolve. Fuzzy sets and fuzzy logic have also been applied to develop specification languages for presenting adaptation requirements and goals [20, 1]. While these languages provide a systematic way of describing uncertain knowledge, it may be non-trivial to implement them and support these goals at runtime.

In order to validate the proposed approach, we have integrated it into a adaptive video streaming application scenario and evaluated its impact on the performance of the application. We demonstrate how the approach allows the construction of an adaptation engine that effectively controls the streaming rate according to bandwidth availability. To concretely show the efficiency of this engine, we evaluate its performance through a set of simulations (using the ns-2 network simulator), and compare it with an alternative implementation that uses the standard network level *TCP Friendly Rate Control* (TFRC) protocol [7]. The simulation results confirm that our adaptation method is capable of producing smoother bitrate changes while keeping the loss rate acceptable. We also evaluate the benefits of the achieved bitrate in terms of the quality of the corresponding video stream according to a set of standard metrics, such as *Peak Signal-to-Noise Ratio* (PSNR) and *Structural Similarity Index* (SSIM). Considering the same encoding reconfiguration method for both protocols, we show that the video streams produced using the proposed fuzzy logic control generally achieve a 10% to 30% better quality than the video streams produced using the TFRC protocol.

The rest of the article is organized as follows. Section 2 presents two application scenarios that illustrate the potential benefits of the proposed approach. Section 3 discusses knowledge management in our approach, showing how a shared vocabulary is specified and used to build adaptation policies. Section 4 describes how a fuzzy inference engine can be integrated into a MAPE-K adaptation loop, while in Section 5 the main advantages of the proposed modular approach are discussed. Section 6 presents the evaluation of our approach for the video streaming application scenario. Section 7 discusses related work, and finally Section 8 provides some concluding remarks and outlook to future work.

2. APPLICATION SCENARIOS

In this section we present two application scenarios where the proposed approach can be applied to improve context interpretation is adaptive software systems. The first one is adaptive video streaming, which is the main scenario discussed throughout the article and used in our extensive evaluation. The second scenario is planning-based adaptation, in which we discuss how our approach can be used as an alternative to utility equations.

2.1 Adaptive video streaming

The application scenario that is used throughout this article consists of a video streaming application, in which client nodes can connect to media servers to receive video streams in real time. Figure 1 shows the main components executing on the server and client nodes. This illustration is representative of many existing media streaming applications (e.g., VLC Media Player and Darwin Streaming Server) and the components are based on commonly used technologies and protocols. The server is composed of an H.264 [21] component, responsible for encoding raw video data into *Network Abstraction Layer* (NAL) units, suitable for streaming, and a *Real Time Streaming Protocol* (RTSP) [17] component, that packages and transmits the NAL units to client nodes. During a streaming session, RTSP components also exchange *Real-Time Transport Protocol* (RTP) control reports, which contain useful information for controlling the session quality, such as packet loss fraction and inter-arrival jitter.

Since clients and server may be connected through a network with variable bandwidth availability, the application needs to continually adjust the streaming rate to match the network characteristics. This article evaluates adaptation scenarios that focus on the bitrate produced by the encoder component as the primary quality dimension. Nevertheless, the proposed methods for knowledge management presented in Section 3 and control loop construction in Section 4 are generic, and can be used to implement new adaptation policies (or extend existing ones) that consider other video quality dimensions, such as resolution and frame rate.

There are different approaches for implementing rate adaptation in this type of application. One approach is to rely on network level protocols for congestion control (such as increase/decrease algorithms) to obtain a target bitrate, and adapt the encoder accordingly [14]. Protocols such as TFRC, use loss rate, round-trip time and packet size as parameters of throughput equations, designed to obtain a more accurate estimate of the available bandwidth. TFRC is arguably the best option for media streaming, however its throughput equation cannot be easily modified to reflect specific features of the application model (such as encoder tolerance to packet loss). Another approach is to completely implement the adaptation at the application level, with algorithms to scale up or down the video quality based on different network conditions, such as the one presented in [18].

A fundamental problem when adapting this application (at the network or application layer) is the imprecision of the information used in the reasoning and decision making processes. First, it is not possible for the application to precisely calculate the currently available bandwidth, and this information needs to be inferred from context data fed back from clients to servers (e.g., through the mentioned RTCP reports), which in turn, may not be accurate. Second, the de-

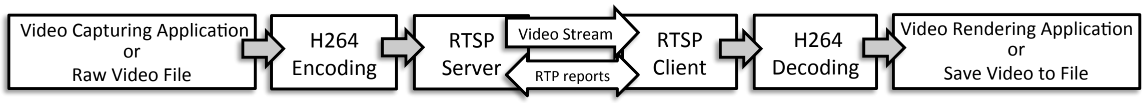


Figure 1: Media Streaming Application

sirable state (adaptation goal) needs to be expressed in terms of imprecise context situations. For instance, does a packet loss measurement necessarily mean congestion? What is the acceptable loss fraction considering measurements inaccuracy? Therefore, the adaptation result may not be optimal (e.g., bitrate overshoots and oscillations).

2.2 Planning-based adaptation

The second scenario is planning-based adaptation, where a software configuration (components and their interactions) is evaluated and adapted at runtime to provide the best possible utility to end-users [16]. Different configurations of a given application can satisfy different constraints and provide different *Quality of Service* (QoS) depending on the operating conditions and user preferences. Changes in the operating environment or user requirements will trigger the planning mechanism, which consists of computing the utility of all possible configurations and selecting the one that is most suitable for the current situation (highest utility).

One example of such an application is the Travel Assistant, which helps travellers with route planning, ticket vending, detecting delays, etc. This application was originally developed using the MUSIC middleware, and a more complete description can be found in [15]. The Travel Assistant runs in a mobile device and consists of a set of interacting components implementing different aspects of the application, such as route planner and geographical location. Variations of the same component type can provide the same service with different quality levels, and the middleware is responsible to identify relevant context changes and find the most useful configuration according to the new context. The utility of a given configuration is influenced by the device context (battery level, GPS signal, etc.) and user preferences regarding cost, accuracy, and resources consumption.

The problem of imprecision can also be found in this scenario: First, QoS prediction models are used to determine the new values for the configurations properties when a context change occur (e.g., how much battery will a given configuration consume under the new operating conditions?). Second, subjective user preferences are used to determine which quality properties are more important (e.g., the user prefers high accuracy over low battery consumption). Predictions and subjective preferences are imprecise and uncertain by nature, but nevertheless, utility functions are described as mathematical equations that try to precisely quantify these properties and preferences. The result is the selection of optimal configurations given the assumption of a precise quality model. In reality, it is difficult to verify if the optimal configuration is really better than a sub-optimal one, since the difference might be too small to be perceived by users. If the difference is not perceivable, than the cost of reconfiguring the application might exceed the benefits of the new configuration. The imprecision problem is unavoidable in both scenarios, but it should at least be considered in the development process, so that the imprecision is made explicit in the system design and correctly reflected in the adaptation behavior. In Section 3.2 we show how we can make the imprecision explicit in both scenarios by using fuzzy adaptation policies.

3. KNOWLEDGE MANAGEMENT

In this section, we present our approach for systematically organizing the adaptation knowledge. As discussed in Section 1, we have augmented the knowledge management of a MAPE-K loop to support imprecision by applying *fuzzy set theory*. Traditional set theory assumes a precise model with no ambiguities and well known parameters, e.g., an element either belongs to a set or does not. In cases where these assumptions do not hold, which is often the case for adaptive systems with partial knowledge, we suggest the use of fuzzy set theory to quantify the model's imprecision. In the fuzzy set theory, an element belongs to a set to some degree, defined by a *membership function*. Consider, for instance, the statement: *"The packet loss fraction is high"*. Using traditional set theory, a given fraction belongs either to the *low* set or to the *high* set (boolean membership relation). Using fuzzy set theory, since *high* is a subjective concept and cannot be precisely determined, a given fraction belongs to the *high* set to some degree, but at the same time, it can also belong to the *low* set to some degree (fuzzy membership relation).

3.1 Structuring Application Knowledge

Most adaptive systems employ ad hoc solutions for knowledge management, where the knowledge is scattered throughout different models, making it difficult for independent entities (such as adaptation managers and individual control tasks) to share and reuse common concepts. In our approach, the knowledge is described using domain specific ontologies, which help in solving issues such as ambiguity, data organization and semantic interoperability. The result is a shared vocabulary, describing all the important concepts in the application domain. The vocabulary can be enhanced with imprecision and vagueness, by associating the interpretation of specific concepts with fuzzy sets.

Figure 2 shows a concept from the vocabulary used for the streaming application. The *event* concept corresponds to any relevant information observed by the adaptation engine. The *loss event* concept describes information about packet losses experienced by the application, which is used by the adaptation controller to reason about the network condition. The *loss fraction* property indicates the fraction of packets that were not received by the client since the last report. This vocabulary is then extended with fuzzy predicates, so

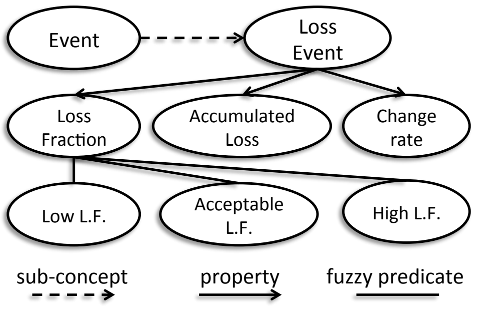


Figure 2: Loss event concept

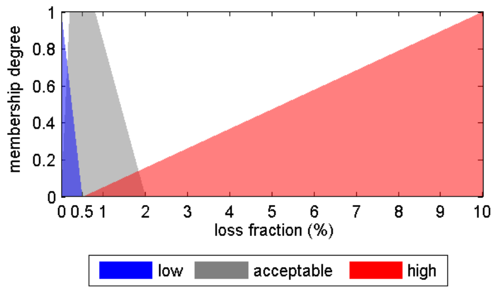


Figure 3: Loss fraction membership functions

that the loss fraction property can be classified as *high*, *acceptable* and *low* (with different confidence levels obtained by applying the corresponding membership functions).

Figure 3 shows the membership functions used to interpret the loss fraction concept, corresponding to the predicates in Figure 2. In this case, we have used piecewise linear functions for simplicity, although other types of functions can be used (Gaussian, sigmoidal, etc.). Piecewise linear functions are easier to specify and computationally efficient (the imprecision regions between no membership and full membership are captured by simple linear functions), and are frequently used in fuzzy systems.

It is important to notice that membership functions represent the developer's knowledge about the application domain. The developer of the adaptive streaming application, for instance, knows that 10% is definitely a *high* value for loss fraction and 0% is a *low* value. He/she also knows that, in some cases, values between 0 and 2% can be tolerated by the video decoder (although the exact values are uncertain), and uses a trapedoizal shape for the *acceptable* function, as shown in Figure 3. In later development phases, these functions can be fine tuned to reflect new knowledge and better understanding of the application behavior.

3.2 Fuzzy Adaptation Policies

The shared vocabulary constructed with fuzzy concepts allows the introduction of fuzzy logic into the adaptation reasoning process. Instead of using conventional boolean statements (e.g., loss fraction $\geq 2.0\%$), it is now possible to apply fuzzy logic statements containing linguistic predicates (e.g., loss fraction is high). Fuzzy statements are employed in our approach to create rules, which can be grouped into high-level policies to drive the adaptation process. For example, by using the loss fraction concept described earlier, a simple increase/decrease rate control policy could be built with the following set of rules:

```
RULE 1: IF loss_fraction IS low
        THEN adjustment IS positive;
RULE 2: IF loss_fraction IS high
        THEN adjustment IS negative;
RULE 3: IF loss_fraction IS acceptable
        THEN adjustment IS null;
```

In the above rules, *adjustment* is the rate adaptation parameter (related to a reconfiguration method), and *positive*, *negative* and *null* indicate respectively an increase, decrease or no change in the rate. The evaluation of this set of rules will produce membership degrees for all three rate adaptation possibilities, that will finally have to be converted to a single crisp value through a process called *defuzzification*, discussed in Section 4.

We have extended the above simple policy to implement a fuzzy *proportional-integral-derivative* (PID) controller to adapt the video streaming application. The proportional term is already captured by the increase/decrease rules, since the rate adjustment is proportional to the loss fraction. The integral term represents the accumulation of the losses over time, and can help in stabilizing the adaptation process. The derivative term represents the rate of change, and can make the adaptation more responsive to considerable loss increases. The integral and derivative terms are also described as properties of the loss event concept shown in Figure 2, and are associated to fuzzy predicates describing the membership to the appropriate fuzzy sets: *positive* and *negative* for the integral term (*accumulated loss*) and *increasing* and *decreasing* for the derivative term (*change rate*).

The complete set of rules is presented below. The rules are weighted (using the keyword *WITH*) to give different importance to the PID terms, and *round-trip time* (RTT) information is included to prevent possible latency increase caused by buffering.

```
RULE 1: IF loss_fraction IS low AND
        RTT IS NOT high THEN
        adjustment IS positive WITH 0.4;
RULE 2: IF accumulated_loss IS negative
        AND RTT IS NOT high THEN
        adjustment IS positive WITH 0.2;
RULE 3: IF change_rate IS decreasing
        AND RTT IS NOT high THEN
        adjustment IS positive WITH 0.4;
RULE 4: IF loss_fraction IS high THEN
        adjustment IS negative WITH 0.4;
RULE 5: IF accumulated_loss IS positive THEN
        adjustment IS negative WITH 0.2;
RULE 6: IF change_rate IS increasing THEN
        adjustment IS negative WITH 0.4;
RULE 7: IF loss_fraction IS acceptable
        THEN adjustment IS null;
```

Note that these rules use the vocabulary defined in the domain ontology (Figure 2), and offer a description of the adaptation process that is linguistically close to how a human would describe them. Also, since the only numerical values used are the relative weights, the set of rules is likely to remain the same as the application evolves, while the membership functions would be changing to reflect new knowledge with more precise values.

One recurrent problem with traditional adaptation policies is the possibility of conflicts, which is aggravated by the boolean nature of the decision making process. This problem can be alleviated with fuzzy logic, where the evaluation of each rule results in a *degree of truth* instead of *true* and *false* values, facilitating the choice and prioritization of the actions. For instance, when evaluating the set of rules described above, *adjustment* can be *positive* and *negative* at the same time, with different degrees of truth for different rules. By applying simple methods for accumulation and defuzzification, the results can be combined and converted to a single *adjustment* value, as will be discussed in Section 4. If there are multiple policies resulting in multiple conflicting actions, then a strategy to prioritize actions should be applied, such as executing only the action with highest degree of truth. In the video streaming application scenario, since the adaptation consists of adjusting only a single parameter, policies implemented as fuzzy rules are sufficient and conflicts are not likely to happen.

The presented knowledge model also supports the use of fuzzy rules as an alternative way of expressing goal policies and utility functions, producing approximate solutions instead of optimal ones, as we are going to show next using the Travel Assistant application discussed in Section 2.2. The utility of a given configuration of the Travel Assistant application is defined as a function of the accuracy that the configuration can provide (in terms of map detail, route reliability and location precision) and how much battery the configuration might consume: $utility = w_{acc} * accuracy + w_{bat} * (1 - battery)$. In this function, the user preferences are captured in form of relative weights (w_{acc} and w_{bat}). The main assumption used to formulate the utility function is that *"the user always prefers high accuracy and low battery consumption"* [15]. However, the imprecision of the *low* and *high* concepts in this statement is not preserved in the utility function. The same assumption can be easily converted into a set of fuzzy rules that, unlike the utility equation, preserves the imprecision of the statement:

```
RULE 1: IF accuracy IS high AND battery IS low
        THEN utility IS high;
RULE 2: IF accuracy IS low AND battery IS high
        THEN utility IS low;
RULE 3: IF accuracy IS high AND battery IS high
        THEN utility IS medium;
RULE 4: IF accuracy IS low AND battery IS low
        THEN utility IS medium;
```

These rules are independent of the interpretation of the *low* and *high* terms for the *accuracy* and *battery* properties, which can be formulated as membership functions. The membership functions can also reflect the user profile, since the *high* and *low* predicates for each property may be adjusted according to the user preferences. Nevertheless, relative weights can also be used to devise rules that give preference to either battery consumption or accuracy individually and build fuzzy policies that can produce results closer to weighted sum equations:

```
RULE 1: IF accuracy IS high
       THEN utility IS high WITH w_acc;
RULE 2: IF battery IS low
       THEN utility IS high WITH w_bat;
RULE 3: IF accuracy IS low
       THEN utility IS low WITH w_acc;
RULE 4: IF battery IS high
       THEN utility IS low WITH w_bat;
```

By replacing utility equations with fuzzy policies, we make the imprecision of relevant properties explicit, and separate the interpretation of these imprecise properties (ontology concepts) form the adaptation policies. This approach presents several benefits for the evolution of the application, as is further discussed in Section 5. Another benefit is the stabilization of the adaptation process. In a dynamic environment, the application context is constantly changing and therefore adaptation mechanism will be activated frequently. When using utility equations, the optimal configuration is always applied even when the utility difference between the current configuration and the new one is very small. By using fuzzy sets instead of numeric utility values, we can reduce the number of reconfigurations, since different configurations that are part of the same utility fuzzy set can be considered equally useful to users.

4. FUZZY ENGINE AS A MAPE-K LOOP

In this section, we propose a new effective way of introducing approximate reasoning into a MAPE-K adaptation loop, by exploiting the imprecise nature of fuzzy policies. We will use only the video streaming application as example in this section, but the general adaptation loop can also be applied in combination with a planning-based adaptation middleware. To help us to describe and construct a control loop for the streaming application, we have developed a prototype of a flexible framework for adaptation, which is based on a conceptual model defined in a previous work [13]. The framework provides support for the description and implementation of MAPE-K control tasks as semantic enabled services (using OWL-S notation [11]), allowing us to make design decisions at the level of individual control tasks. The adaptation loop is therefore described and executed as a composite service, which in this case implements a *fuzzy inference engine* capable of interpreting imprecise concepts and evaluating fuzzy policies. Figure 4 illustrates the adaptation loop where the steps of the fuzzy inference process are mapped to the appropriate control tasks of a MAPE-K loop.

The adaptation process assumes a shared knowledge base, as shown in Figure 4. We have implemented the knowledge base as a simple repository service, that feeds adaptation rules to the control tasks forming the adaptation loop. The other elements of the shared knowledge, such as the domain specific ontology discussed in Section 3.1, are made available as web resources. The tasks are configured to use those resources as part of their own vocabulary, so they can correlate the linguistic terms used in the rules with the corresponding concepts in the ontology.

The feedback control loop starts with the *monitoring task*, which aggregates context data received from sensors deployed in the application. For the streaming application, the sensors are simple plug-ins that intercept RTCP reports

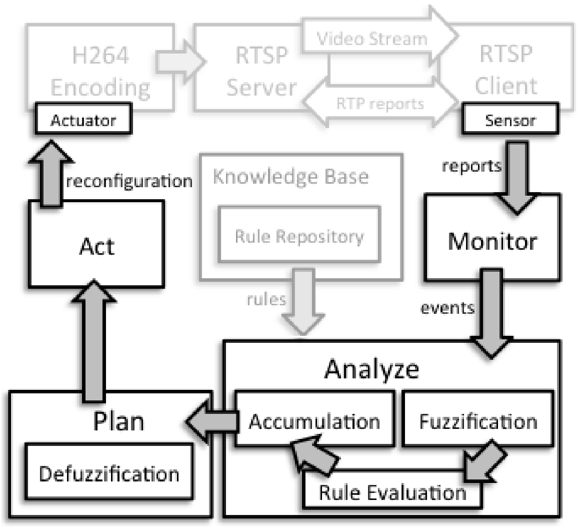


Figure 4: Control loop for the application scenario

and forward them to the monitoring task. This task extracts the context information from the reports (loss fraction and RTT), derives other required inputs (accumulated loss and change rate) and produces the events consumed by the analyzing task. At this point, the loss event contains only *crisp* (i.e., raw) numerical values for its properties.

The *analyzing task* is responsible for the inference process, converting the crisp context data into membership degree values (fuzzification) and evaluating the rules. Events are interpreted according to the shared vocabulary, and the fuzzy predicates (with the appropriate membership function) are applied to obtain the membership degrees. After evaluating all the rules, the results should be combined in order to obtain a single membership value for each possible action (i.e., the results of rules 1,2 and 3 need to be combined to obtain the membership degree for the *positive* set). The combination is done by using the rule weights (if present) in a sum function that aggregates the membership degrees for each action. This aggregation method is closer to the PID method, but can also be configured statically or dynamically, as a parameter to fine tune the adaptation loop. Yet another configurable parameter is the method used to implement the fuzzy operators. We have applied the traditional *min* and *max* implementation of the *AND* and *OR* operators, respectively [23].

The *planning task* determines what action should be taken, considering the membership values of all possible actions provided by the analyzing task. The planner implements the defuzzification method, which is also a configuration parameter of the control task. There are many defuzzification methods that can be applied, such as center of mass, center of gravity and diverse maxima methods [23]. There are also different criteria for selecting the appropriate method, such as continuity and computational efficiency [9]. The developer can select the appropriate method based on the characteristics of system, for instance, if the adaptation loop is running on a resource constrained device (e.g., the Travel Assistant application), a method that satisfies the computational efficiency criteria would be preferable. For the video streaming application scenario, we have selected the *center of gravity* (COG) method as it satisfies the continuity criteria. This criteria is important for control systems, and states that small variations in the inputs should not cause big changes in the output. With COG, the output of the planning task is affected by the membership values of all possible actions, and not only by the one presenting the maximum degree.

Once the planner finishes the defuzzification process, it notifies the *acting task* of the selected action and the final crisp value (percentage to increase or decrease). The acting task executes the adaptation through a set of actuators plugged into the application. If the resulting output belongs to the *null* set, then no action is taken. If the output belongs to the *positive* or *negative* sets, the actuator is invoked to apply the rate adjustment using the reconfiguration method offered by the H.264 encoder. The described control loop is reusable and does not require any particular programming model, making its integration into legacy systems easier. However, extra development effort is required to plug the controller into the managed system, as sensors and actuators are application-specific.

5. ADVANTAGES OF THE PROPOSED MODULAR APPROACH

The modular nature of our approach enables the individual evolution of different aspects of knowledge management in adaptive systems. The first aspect is the shared vocabulary, which is specified as domain specific ontologies. Those ontologies can be formulated and evolved independently of the adaptation controller, and can be reused by various control tasks and in different adaptation policies. The loss event presented in Figure 2, for instance, could have been initially defined as containing only the *Loss Fraction* property, and later evolved (as a consequence of new requirements and adaptation knowledge) to include the *Accumulated Loss* and *Change Rate* properties, needed by the rules implementing the PID controller. Without using a modular approach, the adaptation knowledge would have been scattered among different policies and at the risk of developing incompatible vocabularies.

The second aspect is the interpretation of concepts. In our approach the fuzzy sets can evolve independently of the adaptation rules using them. One example is the *acceptable* fuzzy set shown in Figure 3: a particular loss fraction measurement can be interpreted as *acceptable* if it presents a high degree of membership to this set. By adjusting the membership function that defines the *acceptable* fuzzy set, the adaptation behavior can be changed to be less tolerant to packet loss, without the need to modify any of the adaptation rules. Likewise, in the Travel Assistant scenario, the membership functions for the battery consumption and accuracy properties can be adjusted to be more strict (closer to precise optimal values) or more tolerant (imprecise, with more overlapping areas). Using conventional adaptation rules (boolean logic) or utility equations, the interpretation is a fixed part of the rules/equation, making it difficult to reason about and evolve adaptation knowledge that is inherently imprecise.

The third aspect is the specification of adaptation policies.

By separating the interpretation of concepts from the policy definition, we make it possible to specify complete and semantically correct rules with only partial knowledge about the adaptation domain. The evolution of adaptation policies is then done at an higher abstraction level (without having to deal with numerical precision): Adding and removing rules, augmenting existing rules with new fuzzy statements and changing relative weights. As an example, the relative weights that have been assigned to the rules implementing the PID controller, can be modified in later development iterations to give different importance to the PID terms. This modification does not affect the interpretation of the concepts used by the rule, but just the final accumulated result of the rule set. Another possible modification is the use of fuzzy adverbs, such as using "`utility IS very high`" , instead of using the numerical weights in "`utility IS high WITH w_acc`". The w_{acc} weight is a numerical value, and replacing it by a linguistic modifier can further improve the separation of the concepts used in the rules and their interpretation.

The fourth aspect is the specification of the adaptation engine. The modularity of our approach enables the definition of configurable MAPE-K loops with loosely coupled control tasks, that can be easily modified according to different application requirements. The adaptation loop implementing the fuzzy inference engine, for instance, can be reconfigured with different implementations for the accumulation and defuzzification methods. As an example, a more lightweight implementation of the adaptation loop could have been implemented by selecting only less complex and computationally efficient methods.

6. EVALUATION

In this section we present the evaluation of the proposed knowledge management approach for the video streaming scenario. We have selected this scenario because it is possible to quantify the performance gain and compare it to existing approaches that use precise models. While the performance gain cannot be directed linked to the advantages presented in Section 5, the proposed modular approach makes it easier to tune the adaptive behavior toward the target performance. In the video streaming scenario, for instance, given that the high level policy described in Section 3.2 is correct, tuning the adaptive behavior consists of adjusting only the membership functions. In our experience, this tuning would otherwise involve a more complex redesign of the adaptive behavior. Therefore, the objective of the evaluation is to show the benefits that our approach brings to the video streaming application, in terms of bandwidth utilization and media quality. We compared the performance of the application built using our approach with an application that uses a conventional implementation based on TFRC protocol. The evaluation is divided into two parts: Section 6.1 evaluates the adaptation loop performance at the network level and Section 6.2 evaluates the quality gain of the produced video streams. For the first part, we simulate different network conditions (using the ns-2[2] discrete event simulator), and compare the loss pattern and bitrates achieved by using the fuzzy method and the ones achieved by the TRFC protocol. For the second part, we have modified the video streaming application to reproduce the same network conditions by replaying the recorded simulation traces, resulting in adapted video sequences. The adapted video sequences are stored by the receiver, and we evaluate their quality using a set of standard video quality metrics.

6.1 Network behavior

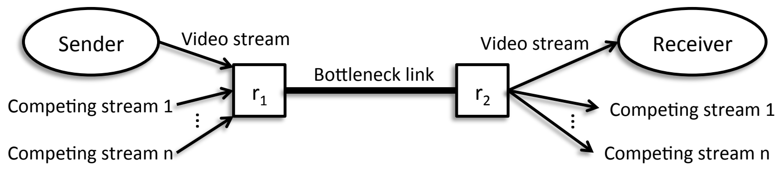
For the first part of the evaluation, we extended ns-2 with the RTP and TFRC implementation proposed in [2], and with the adaptation mechanism presented in Section 4. We have used *dumbbell* topologies as illustraded in Figure 5a to simulate bottleneck links between two Internet routers (r_1 and r_2) connecting a video stream sender and receiver. Some simulations also include n other data streams with different characteristics competing for the same bottleneck link between r_1 and r_2. We have also used *parking lot* topologies as illustrated in Figure 5b, which introduce n intermediate routers (r_1 to r_n in the figure) into the link. In this case, there are producers ($P_1, P_2, ...$) and consumers ($C_1, C_2, ...$) of data traffic connected to different routers across the link, resulting in multiple bottlenecks with cross traffic traversing all the routers in the path. These topologies are commonly used to simulate bandwidth-limited bottleneck links. We have tested a broad range of network conditions, with varying link capacities (0.5 to 10 Mbits) and number of flows competing for the link (up to 20 flows).

Figures 6(a) and 6(b) show the results obtained with the first (and simplest) ns-2 simulation scenario using the dumbbell topology, where the link capacity is fixed and there are no competing streams. The streaming session lasts about 10 minutes and, since there is no initial assumption about the link capacity, the session starts at low bitrate (100 kbps). From the figures, it is possible to see that the fuzzy rate adaptation utilizes the whole link capacity at the cost of a constant acceptable loss fraction (less than 1% average), while TFRC produces large oscillations, with considerable bitrate reductions when a loss event is detected.

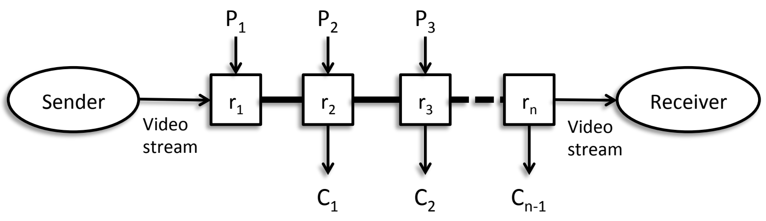
Figures 7(a), 7(b) and 7(c) show the results of a simulation scenario involving the competition for the link capacity between a FTP flow (TCP), a constant bitrate flow (UDP) and the video stream (RTP). For this simulation, we have used the Linux implementation of the TCP protocol (TCP-Linux), with the *new reno*[3] congestion control algorithm. The upperbound for the TCP congestion window was set to a value large enough to allow the full utilization of the link. This scenario shows that the fuzzy method produces higher bitrates with less oscillations than TFRC, while keeping the loss fraction (0.49% avarage) close to the *acceptable* fuzzy set. However, it should be noted that when the fuzzy method is applied the bandwidth sharing is not *fair*, and at some point the RTP flow will be consuming more bandwidth than the TCP flow. If fairness is one of the application requirements, the adaptation rules and/or membership functions should be adjusted to achieve a fair behavior, otherwise

[2]http://www.isi.edu/nsnam/ns/

[3]http://www.ietf.org/rfc/rfc2582.txt



(a) Dumbbell topology



(b) Parking lot topology

Figure 5: Network topologies used in the simulations

the fuzzy method is not recommended.

Figures 8(a) and 8(b) show the average goodput (application level throughput) and loss fraction, achieved in simulations with fixed link capacity and increasing number of competing TCP flows. From the graphs it may be noticed that, with less competition and as long as the loss fraction is acceptable, the fuzzy method displays a more aggressive behavior than TFRC, achieving considerably higher goodput. With a high competition, the achieved goodput is lower than the one produced by TFRC, but the loss fraction is also lower, to fit the *acceptable* set.

A number of other simulation scenarios were evaluated, and the results were consistent with the ones presented in this article. Figures 9(a) and 9(b) show the simulation results when we applied the parking lot topology. In this case, the traffic between adjacent routers is generated in bursts, with sending and idle times taken from a Pareto distribution. The bursts are intermittent and the traffic could be already interrupted when a rate adaptation occurs, which might result in inaccurate adaptive behavior. In addition, the context interpretation becomes more challenging as we add more intermediate routers, which increases the total latency experienced by the application and the chances of having a congested link in the path between sender and receiver. With increasing number of intermediate routers, the simulation results show that, even with intermittent traffic and increasing latency, the fuzzy method is able to keep the loss rate within the *acceptable* set and achieve a considerably higher goodput than TFRC in general.

6.2 Media Quality evaluation

In the second part of the evaluation, we generated adapted video sequences using the bitrate variations and loss patterns obtained with the simulations. We used uncompressed video sequences[4] as input for the application. The encoding process is done at runtime to generate NAL units according to the target bitrate. When the bitrate is adapted, the encoder is reconfigured to reduce the encoding quality for the video frames while maintaining the same resolution and frame rate. The loss pattern is applied to the streaming session, so the client stores the adapted video without the NAL units lost during transmission.

We applied a set of metrics to evaluate the video quality obtained with the fuzzy method in comparison with the quality obtained with TFRC. The chosen metrics were *Peak Signal-to-Noise Ratio* (PSNR), *Structural Similarity Index* (SSIM), *Spatial-Temporal SSIM* (stSSIM), which considers temporal distortions, and the *DCT based Video Quality Metric* (VQM). PSNR is the most commonly used metric for the evaluation of the quality difference among video frames, but it is known to have poor correlation with subjective metrics (the ones based on human perception) [19]. To have a more reliable indication of the video quality, we have also applied SSIM, stSSIM and VQM. Furthermore, we have shown that the intensity of the adaptation and the frequency of change per report interval are reduced when using the fuzzy method (as can be seen in Figure 7). The amplitude and frequency of adaptations are know to have significant impact on the subjective quality [12], thus we believe that our approach can improve subjective user satisfaction as well.

Table 1 summarizes the quality evaluation of video sequences obtained from some of the simulation scenarios discussed above. The values shown in the table are the average scores of the adapted video sequences compared with the original uncompressed sequences. Different portions of the sequence may present different scores, depending on the characteristics of the frames and the target bitrate. In general, the fuzzy method produces better video quality than TFRC according to all metrics tested. However, it should be noted that this is only possible with the appropriate decoding support (the H.264 decoder can tolerate NAL losses).

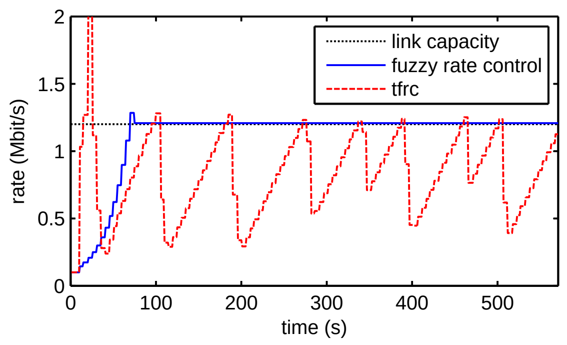
7. RELATED WORK

In this section, we present related works that apply fuzzy logic concepts to deal with uncertainty when specifying adaptive behavior and discuss how these works differ from our

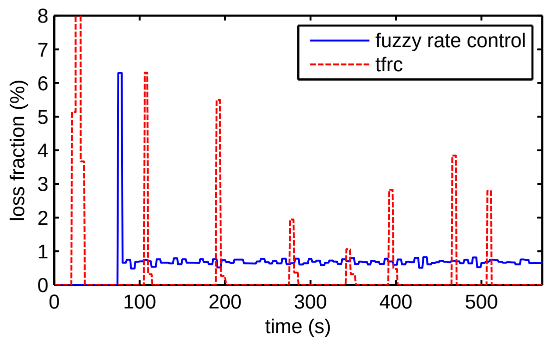
[4]Obtained from http://media.xiph.org/video/derf/

Table 1: Video Quality Evaluation

Metric	Scenario 1 (No competing flows)		Scenario 2 (Figure 5)		Scenario 3 (10 TCP flows)	
	Fuzzy Method	TFRC	Fuzzy Method	TFRC	Fuzzy Method	TFRC
PSNR (higher is better)	**33.07**	29.77	**35.02**	32.77	**26.17**	25.24
SSIM (higher is better)	**0.91**	0.82	**0.97**	0.89	0.76	**0.78**
stSSIM (higher is better)	**0.53**	0.32	**0.61**	0.49	**0.34**	0.26
VQM (lower is better)	**1.69**	2.20	**1.52**	1.58	**2.31**	2.53

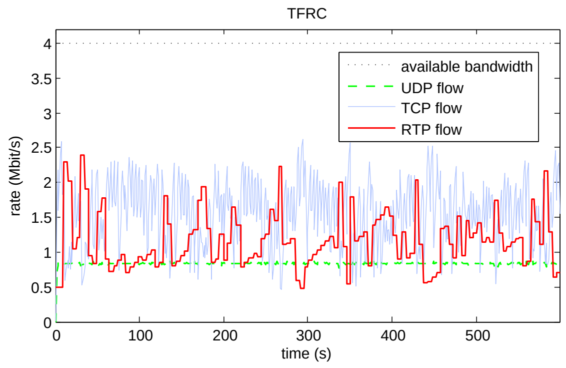


(a) Achieved throughput.

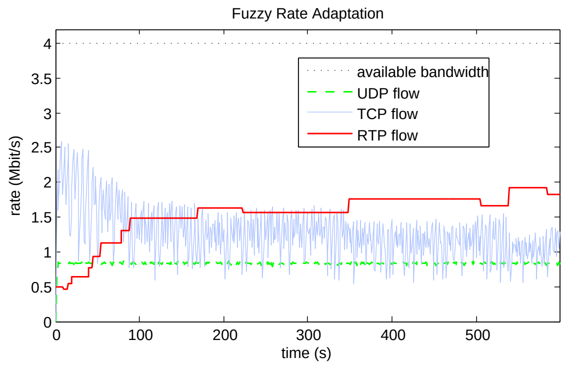


(b) Observed packet loss.

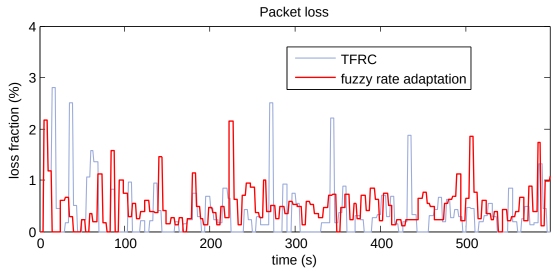
Figure 6: Simulation results of the fuzzy rate adaptation vs. TFRC for the first scenario.



(a) TFRC



(b) Fuzzy Rate Adaptation



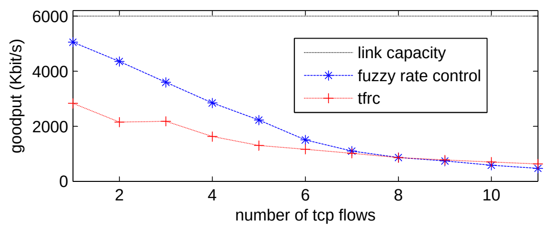
(c) Packet Losses

Figure 7: Simulation results of TFRC (a) and fuzzy rate adaptation (b) for the scenario with competing TCP and UDP flows, and the observed packet loss values (c)

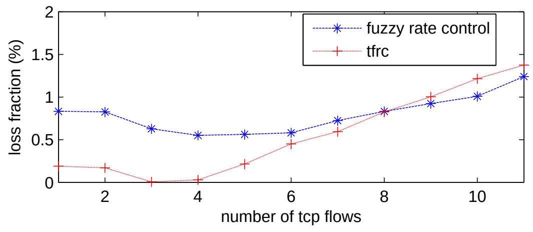
approach. In our previous short paper [13], we outlined the idea of a flexible framework for self-adaptation. However, while [13] discusses flexibility and evolvability at a conceptual level, this work is focused on a concrete approach for knowledge management using fuzzy set theory.

The RELAX language by Whittle et al. [20], deals with uncertainty when defining adaptation behavior. The focus is on requirements specification using structured natural language, that can be relaxed at runtime based on environment changes. RELAX is in principle a language independent of specific adaptation mechanism, therefore a runtime system needs to be able to interpret and maintain the requirements. The work in [6] shows how RELAX can be integrated into the KAOS goal modeling framework. Nevertheless, the goal model still requires runtime adaptation support.

In [1], the authors show how an augmented KAOS model together with the RELAX notation could be integrated with a BPEL engine, to transform the system specification into running service compositions. These works deal with adaptation behavior specification at a different abstraction level (requirement elicitation) than ours, and we see them as complementary approaches, as the development process advocated by our framework can be modified to integrate structured languages used to specify requirements and goals.

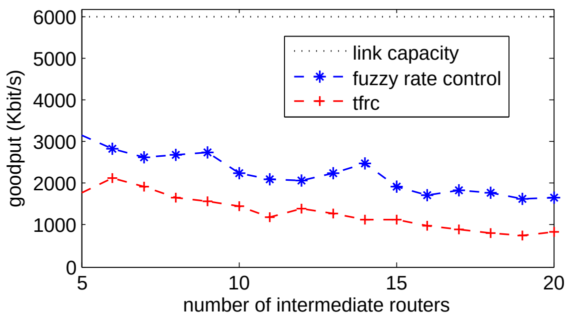


(a) Fuzzy Rate Adaptation vs. TFRC.

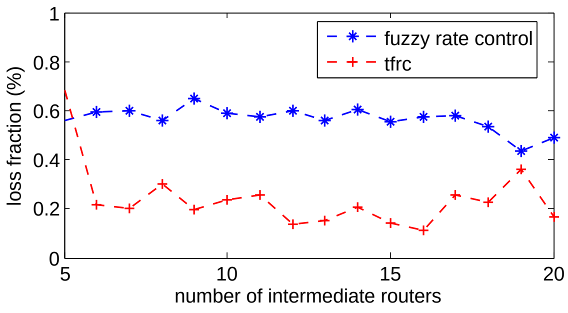


(b) Observed packet loss.

Figure 8: Average goodput (a), and the observed packet loss (b) for simulations with increasing number of competing TCP flows.



(a) Fuzzy Rate Adaptation vs. TFRC.



(b) Observed packet loss.

Figure 9: Average goodput (a), and the observed packet loss (b) for simulations with increasing number of intermediate routers in a parking lot topology.

The work by Chauvel et al. [5] presents an approach for qualitative description of adaptation policies, specified at design time and interpreted at runtime. The policies can be composed using two different types of fuzzy rule sets: local reconfiguration rules, for adjustment of local system properties, and architectural reconfiguration rules, which describe the utility of a specific architecture reconfiguration. Other control-based approaches that make use of fuzzy set theory to deal with ambiguity and imprecision include [10, 22]. These works are similar to ours in the use of fuzzy rules for describing adaptation behavior, whereas no special attention is given to knowledge management. The adaptation loop is also implemented as a rigid controller, resulting in reusability issues and reducing the system's capacity to evolve.

Our work is also related to context awareness and more specifically to *Quality of Context* (QoC) [4], which defines precision as a property of the context information. The problem of imprecision in context acquisition is aggravated when this information is aggregated from various sources, and the precision and accuracy of each sensor component need to be taken into consideration. In [24], fuzzy logic is used to obtain a reliability measure of aggregate context information. In our approach, the focus is on the knowledge imprecision as a consequence of human interpretation, and the membership functions represent the developer's understanding of the context situation (which includes his/her knowledge about sensor precision and accuracy). This way, we avoid the complexity of defining precision boundaries for individual sensors and managing reliability of aggregated context, while providing support to behavior evolution based of human knowledge.

Another relevant work, reported by Bridges et al. [3], is a fuzzy-based framework to control a video streaming application. While the focus of this work is on strategy composition and coordination, less attention is given to knowledge management and interpretation of contextual information. Furthermore, the authors only provide indications of the effectiveness of adaptation composition and coordination, but do not show any improvements regarding the video rate adaptation, since the application displays similar behavior as the increase/decrease protocol.

8. CONCLUSION

In this article we have presented our approach for adaptation in presence of inherently imprecise/vague information. We have proposed a method for organizing knowledge and its interpretation that preserves the imprecise nature of contextual information, and demonstrated how to effectively integrate this method into a MAPE-K adaptation loop. We have applied fuzzy set theory and domain specific ontologies to separate the high level adaptation knowledge (adaptation rules and policies) form its interpretation (fuzzy predicates). We discussed how this approach can be used in two distinct application scenarios and its benefits for the evolution of adaptive applications.

We have evaluated the knowledge management approach for the case of video streaming applications, and we showed that it generally brings benefits in terms of achieved throughput and resulting media quality when compared to a conventional implementation using TFRC. By using the proposed approach, we were able to identify and make contextual imprecision explicit in the adaptation knowledge for the evaluated application. In our experience, this way of describing and evolving adaptation knowledge considerably helps the process of tuning of the adaptive application, resulting in the performance gain presented in Section 6.

As future work, we intend to explore other application scenarios and also investigate knowledge evolution at runtime. We believe that by dynamically changing the membership values, we can achieve more accurate situation dependent adaptations. For instance, the membership function defining what is *high* and *low* battery consumption may change at runtime depending on the current battery level. When a device is running out of battery, it is reasonable to adjust the *high* membership set to involve a bigger area of the battery consumption space, and the *low* set a smaller area, because now low consumption is critical.

9. ACKNOWLEDGEMENTS

We would like to thank Amirhosein Taherkordi for useful comments about the presented work.

10. REFERENCES

[1] L. Baresi and L. Pasquale. Adaptation goals for adaptive service-oriented architectures. In *Relating Software Requirements and Architectures*. Springer Berlin Heidelberg, 2011.

[2] C. Bouras, A. Gkamas, and G. Kioumourtzis. Extending the functionality of rtp/rtcp implementation in network simulator (ns-2) to support tcp friendly congestion control. In *1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems*. ICST, 2008.

[3] P. Bridges, M. Hiltunen, and R. Schlichting. Cholla: A framework for composing and coordinating adaptations in networked systems. *Computers, IEEE Transactions on*, 58(11), 2009.

[4] T. Buchholz, A. Küpper, and M. Schiffers. Quality of context: What it is and why we need it. In *Proceedings of the workshop of the HP OpenView University Association*, 2003.

[5] F. Chauvel, O. Barais, I. Borne, and J. Jezequel. Composition of qualitative adaptation policies. In *23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2008.

[6] B. Cheng, P. Sawyer, N. Bencomo, and J. Whittle. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. *Model Driven Engineering Languages and Systems*, pages 468–483, 2009.

[7] M. Handley, S. Floyd, J. Padhye, and J. Widmer. Tcp friendly rate control (tfrc): Protocol specification. *RFC 3448*, 2003.

[8] J. Kephart and D. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41 – 50, Jan. 2003.

[9] W. Leekwijck and E. Kerre. Defuzzification: criteria and classification. *Fuzzy sets and systems*, 108(2):159–178, 1999.

[10] B. Li and K. Nahrstedt. A control-based middleware framework for quality-of-service adaptations. *Selected Areas in Communications, IEEE Journal on*, 17(9):1632–1650, 1999.

[11] D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D. Mcguinness, E. Sirin, and N. Srinivasan. Bringing semantics to web services with owl-s. *World Wide Web*, 10(3):243–277, 2007.

[12] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen. Flicker effects in adaptive video streaming to handheld devices. In *19th ACM international conference on Multimedia*. ACM, 2011.

[13] L. Provensi and F. Eliassen. Towards a flexible and evolvable framework for self-adaptation. *Electronic Communications of the EASST*, 43(0), 2011.

[14] R. Rejaie, M. Handley, and D. Estrin. Layered quality adaptation for internet video streaming. *IEEE Journal on Selected Areas in Communications*, 2000.

[15] R. Rouvoy, P. Barone, Y. Ding, F. Eliassen, S. Hallsteinsen, J. Lorenzo, A. Mamelli, and U. Scholz. Music: Middleware support for self-adaptation in ubiquitous and service-oriented environments. In *Software engineering for self-adaptive systems*, pages 164–182. Springer, 2009.

[16] R. Rouvoy, F. Eliassen, J. Floch, S. Hallsteinsen, and E. Stav. Composing components and services using a planning-based adaptation middleware. In *Software Composition*, pages 52–67. Springer, 2008.

[17] H. Schulzrinne, A. Rao, and R. Lanphier. Real time streaming protocol (rtsp). *Internet Engineering Task Force, RFC 2326*, 1998.

[18] G. Toma, L. Schumacher, and C. De Vleeschouwer. Offering streaming rate adaptation to common media players. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–7. IEEE, 2011.

[19] Y. Wang. Survey of objective video quality measurements. *EMC Corporation Hopkinton, MA*, 1748, 2006.

[20] J. Whittle, P. Sawyer, N. Bencomo, B. Cheng, and J. Bruel. Relax: Incorporating uncertainty into the specification of self-adaptive systems. In *17th IEEE International Requirements Engineering Conference*. IEEE, 2009.

[21] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[22] Z. Yu, N. Lin, Y. Nakamura, S. Kajita, and K. Mase. Fuzzy recommendation towards qos-aware pervasive learning. In *International Conference on Advanced Information Networking and Applications*, pages 604–610. IEEE, 2007.

[23] L. Zadeh. Fuzzy sets*. *Information and control*, 8(3):338–353, 1965.

[24] T. Zimmer et al. Qoc: Quality of context-improving the performance of context-aware applications. *Advances in Pervasive Computing*, 2006.