Wireless Pers Commun (2015) 84:1739–1769
DOI 10.1007/s11277-015-2519-7

Cognitive Network Management Framework and Approach for Video Streaming Optimization in Heterogeneous Networks

Tiia Ojanperä[1] · Markus Luoto[1] · Mikko Majanen[1] · Petteri Mannersalo[1] · Pekka T. Savolainen[1]

Published online: 18 March 2015

Abstract The future Internet will be highly heterogeneous in supporting a multitude of access technologies and networks with overlapping coverages. Optimization of network operations like management of resources, mobility or Quality of Service in order to ensure smooth network operation and high user satisfaction will be very challenging in the future networks. Cognitive network management can provide a solution of managing such complex systems. This paper studies cognitive network management in the context of optimizing video streaming performance in heterogeneous multi-access networks. The paper proposes a network management framework that relies on cognitive decision techniques in the joint optimization of network and video service performance. The proposed solution is also implemented and validated in part in a testbed environment. The results attest the feasibility of the solution as well as the benefits of cognitive decision techniques over non-learning or non-adaptive approaches.

Keywords Network management architecture · Decision algorithms · Video adaptation · Mobility management · QoE · Testbed

✉ Tiia Ojanperä
tiia.ojanpera@vtt.fi

Markus Luoto
markus.luoto@vtt.fi

Mikko Majanen
mikko.majanen@vtt.fi

Petteri Mannersalo
petteri.mannersalo@vtt.fi

Pekka T. Savolainen
pekka.t.savolainen@vtt.fi

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

1 Introduction

Cognitive network management has been proposed as a solution for tackling network management problems in the complex networks of the future. The future Internet will be highly heterogeneous in supporting a multitude of access technologies and networks with overlapping coverages. Optimization of network operations like resource, mobility or Quality of Service (QoS) management in order to ensure smooth network operation and high user satisfaction in such a multi-access and multi-operator environment will be very challenging. It is clear that today's network management mechanisms, relying on either human expertise or simple self* mechanisms that require an accurate view of the managed network's status, will no longer suffice. The automation of the network management operations with tolerance of uncertainties will be the key to building sustainable and manageable networks in the future. Cognitive network management can provide the required means for it.

In a general setting described by Thomas et al. [1], "A cognitive network is a network with a cognitive process that can perceive current network conditions, and then plan, decide, and act on those conditions. The network can learn from these adaptations and use them to make future decisions, all while taking into account end-to-end goals". Thus, cognitive networks comprise features, such as self-awareness, self-configuration, self-healing, self-optimization, and self-protection, which all can be achieved via knowledge representation and cognitive (learning) loops (see e.g. [2, 3]). In this paper, we study cognitive network management in the context of optimizing video streaming performance in heterogeneous multi-access networks. The problem is highly topical considering the increasing popularity of using video services wirelessly. Mobile video services have a significant role in contributing to the continuously increasing traffic loads posed to wireless networks already today, and the role will emphasize further in the future [4]. The demanding requirements of video services for network capacity and QoS highlight the importance of efficient network management mechanisms in order to ensure satisfactory Quality of Experience (QoE) for the users.

There are different ways to manage and optimize video transmission over wireless. The way the wireless networks (e.g. LTE, HSPA, WLAN) deployed today constitute heterogeneous multi-access environments has created operators the possibility to perform load balancing and off-loading for optimizing resource usage in their networks. Then again, users with multimode terminals may opt to connect to video services dynamically through different networks according to the always best connected paradigm. Moreover, it is possible to relax the QoS requirements of video streaming and match them to available network capacity by adapting the video service properties, like bitrate, accordingly. This nevertheless has an impact on the video QoE (e.g. reduced fidelity). Each of these examples encompasses its own optimization problem studied extensively in the past years. Yet, there is a lack of solutions that jointly manage such multiple optimization functions in a controlled and coordinated manner when striving to achieve the desired end-to-end goal(s) (e.g. maximizing video QoE or optimizing network resource usage).

This paper proposes a cognitive network management framework and multiple decision algorithms for optimizing video streaming services in heterogeneous multi-access networks. The solution relies on cognitive decision techniques in the network management automation and allows joint optimization of network and video service performance in terms of factors such as QoE, cost or network load. The novelty in the work lies in the usage of multiple cognitive decision algorithms in the same network management system

in a coordinated manner. Furthermore, the algorithms presented in this paper were designed and implemented specifically for the considered use case. The algorithms support multiple types of cognition, including expert systems (i.e. fuzzy logic) as well as self-learning (i.e. Q-learning) and trained algorithms (i.e. self-organizing map, SOM). The solution is also validated in part in this paper with an experimental evaluation conducted in a testbed environment. The results attest the feasibility of the solution as well as the benefits of cognitive decision techniques over non-learning or non-adaptive approaches. Further evaluation of the system has been conducted also in a simulated environment and the results will be published in a follow-up paper.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work. The architectural design of the proposed system is introduced in Sect. 3 together with details on the supported management hierarchy. The prototype implementations and the testbed integrating them into a multi-access network environment are described in Sect. 4. The experimental evaluation and main results obtained are detailed in Sect. 5. Finally, Sect. 6 concludes the paper.

2 Related Work

During the last years, cognitive mechanisms have evolved from cognitive radios [5] and self-organizing network (SON) functions [6] of the wireless access towards autonomous and intelligent management, covering the whole mobile network and its services, and including also cross-layer and end-to-end aspects [2]. As a result, the future networks will host numerous autonomous mechanisms (cognitive radios, anomaly/fault detection and recovery, resource optimization, etc.) together with solutions providing knowledge production and sharing, co-operation and coordination, and governance for those. In order to handle such complex systems, several frameworks and architectures supporting cognitive networks have been suggested (see e.g. [7–9]). Specifically, the Unified Management Framework [8] and GARSON architecture [7] are closely linked with the work presented in this paper. This is due to the fact that the Distributed Decision Engine (DDE) approach [10], considered in this paper, can be easily mapped to these both.

Cross-layer communications and network monitoring are essential in providing the required context information for cognitive network management. Cross-layer information has already proven to provide enhancements to mobility management and service adaptation in modern day heterogeneous multi-access networks [11]. This paper aims to utilize similar approaches in facilitating the required knowledge building for the proposed intelligent decision algorithms. For this, various cross-layer signalling frameworks can be used. Existing solutions include, for instance, the IEEE 802.21 Media Independent Handover (MIH) [12]. They enable the collection and distribution of extensive cross-layer information both from the user device as well as from different network nodes. Another relevant topic for the paper is that of the Complex Event Processing (CEP) paradigm [13]. CEP will have an important role in analyzing and controlling the complex series of interrelated events in the modern distributed information systems as the information gathered and distributed by cross-layer signalling solutions increases. For knowledge building and management, the paper utilizes the proprietary DDE framework. DDE shares many similarities with existing cross-layer frameworks, but includes implementation flexibility that simplifies research prototyping.

There are numerous algorithmic ways to implement intelligence into the network and service management. In addition to the popular approaches considered in this paper, that is,

SOM, reinforcement learning (RL), and fuzzy inference, there are many other possibilities. For example, the authors of [14] list more than 30 cognitive technologies and the list is far from being complete. Typically, the selection of the applied algorithms needs to be done on use case basis as there is no universally optimal approach. In the following paragraphs, some related algorithmic solutions are discussed. However, there is no point for detailed comparison since the main focus of this paper is on the overall system, not the separate algorithms and mechanisms.

Using cognitive decision techniques in controlling traffic routing and video bitstream adaptation has been studied extensively in the literature. Mobility management and access selection in heterogeneous multi-access networks based on Q-learning algorithms have been proposed, for instance, in the papers [15–17]. The work presented in the current paper includes a Q-learning algorithm for access selection as well. However, in contrast to the related works, the algorithm proposed in this paper is implemented and evaluated in a real testbed environment, and it supports video application awareness in its decision-making. The authors of [18] evaluate a context-aware vertical handover decision algorithm including application management. The authors have developed and analyzed the algorithm with a multimode mobile terminal with two WLAN interfaces and it is capable of handling multiple applications. However, the algorithm is a mobile-initiated and controlled solution and there is no operator side considered. Also, there seems to be no proper learning in the proposed approach unlike in the approach presented in this paper.

Prior works proposing more advanced adaptation algorithms for video streaming include, for instance, the paper [19] that evaluates some popular adaptive HTTP streaming players and proposes a new algorithm in order to solve the issues found in the evaluated players. The paper [20] proposes a Q-learning algorithm for video bitstream adaptation. RL is used by the authors of [21] for adaptive HTTP video streaming over multiple wireless access networks. Also, the paper [22] uses multiple links simultaneously for adaptive HTTP video streaming. The Q-learning algorithm proposed in the current paper for video bitstream adaptation is developed specifically for adaptive HTTP video streaming. It is also integrated as a part of a more extensive network management system supporting intelligent access network selection, and evaluated in a testbed environment.

Overall, the main novelty of the work presented in this paper lies in the integration of multiple cognitive management algorithms aiming at optimizing video streaming performance in heterogeneous multi-access networks into the same system. In order to achieve stability and improved performance for the overall management, the solution supports coordination of the actions triggered by the individual algorithms. The current paper builds upon and significantly extends the work presented by the authors in [23]. That is, this paper provides a more detailed design of the proposed system and presents the actual implementation and evaluation of the client-side learning algorithms as well as the more advanced coordination logic. To the best of our knowledge, this is the first paper reporting results of such an extensive integrated cognitive network management system.

3 System Architecture

The network management architecture considered in this paper is shown in Fig. 1. The architecture was first introduced in [23]. The architecture facilitates autonomous management of heterogeneous multi-access networks through novel decision algorithms realized using cognitive decision techniques. In this paper, we employ the architecture for

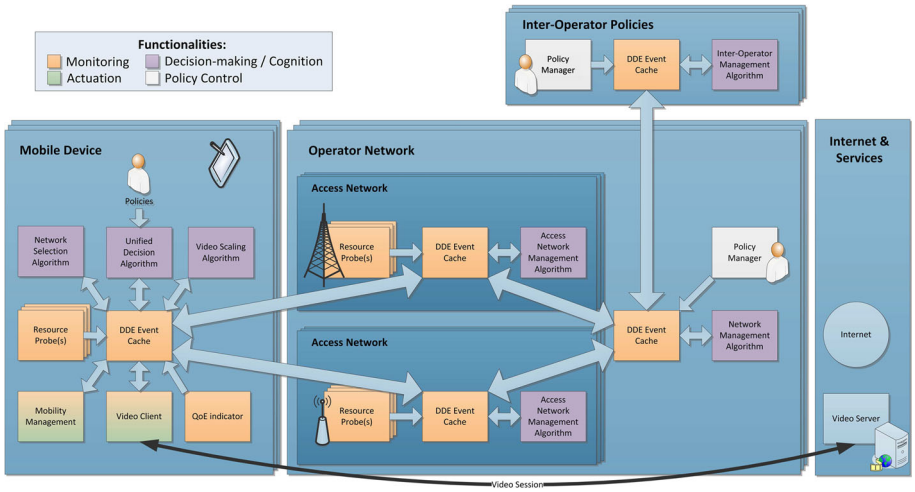

Fig. 1 The hierarchical management architecture for optimizing video streaming in heterogeneous multi-access networks

optimizing the delivery of adaptive video streaming services in multi-access networks. For this, we have also defined a set of novel management algorithms in order to optimize the video streaming performance in terms of factors like QoS, QoE, and efficient usage of network resources, automatically. This section recaps the main structure and elements of the architecture as well as adds a more detailed discussion on the design of the hierarchical management structure.

3.1 Overview

As illustrated in Fig. 1, the proposed architecture encompasses various elements, including multi-mode *Mobile Devices* receiving a video streaming *Service* from the *Internet*; one or more *Operator Networks* each containing several *Access Networks* (*AN*) of different technologies and with one or more network points of attachment (PoA); and an *Inter-Operator Management* level handling inter-operator roaming. The architecture essentially allows the Mobile Devices to utilize the available ANs efficiently by having the video traffic dynamically routed via and scaled to the most suitable network at all times. In the context of this paper, the process of *rerouting* video traffic is always associated with *handovers* from one access network to another. Thus, the terms may be used interchangeably in the text.

The architecture implements a cognitive network management system for autonomous mobility and video adaptation decisions. For this, the architecture adopts the DDE framework [10] as means for integrating the software components that implement the required functionalities, that is, *monitoring*, *decision-making/cognition*, *actuation* as well as *policy control*, into the management system. The differing roles of the components are emphasized with colour-coding in Fig. 1.

In the architecture, DDE facilitates event-based knowledge building and dissemination between and within (i.e. cross-layer) the network nodes. The DDE framework comprises *Event Producers*, which feed DDE with information; *Event Consumers*, which receive information in the form of events they have subscribed to; and *Event Caches* that

orchestrate this information exchange and can be interconnected in cascade fashion. When mapping DDE into the architecture, monitoring entities corresponds to the Event Producers and actuation entities to the Event Consumers. The decision-making entities have a dual role and connect to DDE as both producers and consumers. Finally, the Event Caches include information caching and processing/filtering functions for controlling the publish-subscribe based event dissemination.

In the use case considered in this paper, the actuating entities or actors of the system are located in the Mobile Device. These are the client-side application of the adaptive video streaming service (i.e. *Video Client*) and the *Mobility Management* shown in Fig. 1. The role of the Video Client is to explicitly request the video representation (i.e. fidelity and bitrate) it wishes to receive from the server and is able to change the representation dynamically throughout the streaming session. The server is assumed to be located in the Internet, that is, the service is delivered over-the-top in operator networks. In principle, the different versions of the video may be made available by storing multiple representations of the same video on the server, using scalable video coding or modifying the encoding parameters dynamically during live streaming. The Mobility Management is responsible for enforcing decisions regarding mobility of Mobile Devices and maintaining session continuity within heterogeneous multi-access and multi-operator IP networks. In the architecture, the actors are controlled by external decision algorithms through DDE.

The decision algorithms depicted in Fig. 1 include three algorithms in the client-side, namely Network Selection Algorithm (NSA), Video Scaling Algorithm (VSA), and Unified Decision Algorithm (UDA), and multiple levels of management in the network. For the network, we have defined three algorithms: a PoA Status Classification Algorithm, Load-Balancing Algorithm (LBA), and Bandwidth Control Algorithm (BCA) that may be used for controlling the mobility and resource allocation of Mobile Devices based on an extensive view on the overall network's status. The algorithms have been designed using cognitive decision techniques (esp. SOM, fuzzy logic, Q-learning) in order to achieve the required automation and reliability. The detailed design of all the proposed algorithms is reported by the authors in [24]. Yet, we would like to point out that, so far, we have only a simulator implementation of LBA and none of BCA. Therefore, they are discussed only conceptually in this paper.

In order to facilitate intelligent video bitrate adaptation and handover decisions, the architecture includes various sources of system status information and distributed knowledge building through DDE. For instance, the Video Client can provide feedback on the video streaming performance and the requested bitrate in order to enable application-aware decision-making through DDE. Additionally, the architecture includes various Resource Probes both in Mobile Devices and ANs. The Resource Probes provide, for example, information regarding the status and characteristics of the access links and networks. The system could potentially integrate also other information sources, such as QoS/QoE monitoring tools, if required by the decision algorithms.

Finally, although the system is to operate autonomously, human control can be included through *Policy Managers*. These provide a control interface for network operators and users allowing them to manage access to and dissemination of DDE events as well as the enforcement of the algorithms' decisions.

3.2 Management Hierarchy

The proposed architecture enables flexible and gradual deployment of cognitive decision algorithms by supporting multiple levels of management for deploying the algorithms into

a hierarchy. The hierarchy and the main inputs of the algorithms are shown in Figs. 2 and 3, and will be discussed in more detail in this section. The red color is used in the figures to illustrate the parts which are currently implemented in the testbed presented in this paper. The other parts of the system have so far been evaluated by the means of simulation and will be documented separately.

Overall, the decisions are made on bandwidth limitations, application traffic routing (i.e. handovers), and adaptation (i.e. lowering/increasing the video bitrate and fidelity). The main difference between the client- and network-side decisions is that the client-side is interested only in its own experienced quality whereas the network-side (operator) decisions must take into account the limited available resources and try to share them somewhat equally among network customers. Of course, operator initiated policies can be used to control the client-side decision making, for example, by directly affecting the decision algorithms or limiting the available information about alternative accesses.

The client-side mechanisms located in the Mobile Device are composed of three parts: the network selection algorithms (e.g. NSA) are seeking the optimal wireless link(s) for the active applications; the service adaptation algorithms (e.g. VSA) manage the fine tuning of active applications; and the unified decision algorithm (i.e. UDA) coordinates the (possible contradicting) outputs these two mechanisms as well as instructions from the network.

In the network, the access network management algorithms (e.g. PoA Status Classification or BCA) take decisions in the access network scope, and control for instance how much bandwidth is allowed for each client on different subscription-levels. The network management algorithms (e.g. LBA) optimise the operator level performance, and the inter-operator management algorithms (e.g. LBA) manage the usage of multiple operator networks. They may respond to changes in dynamic load levels by moving

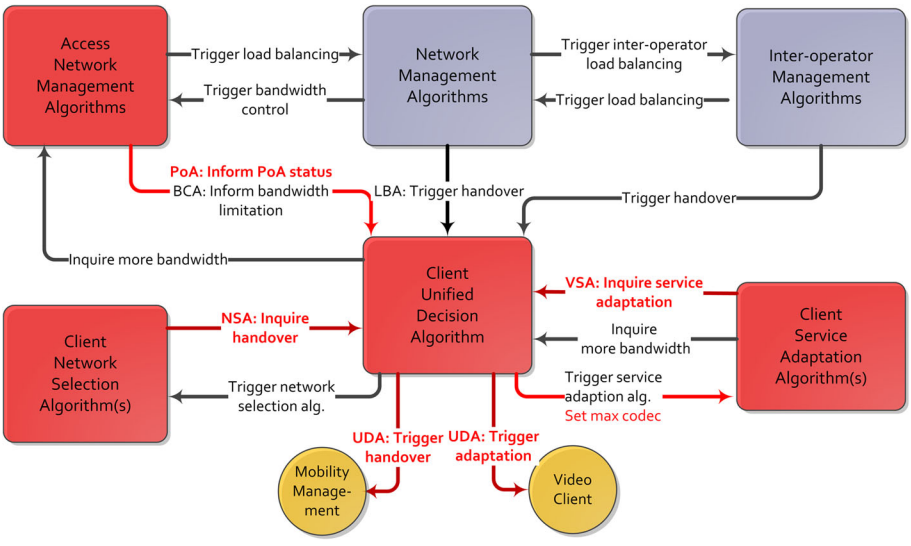

Fig. 2 Algorithm hierarchy and commands between the algorithms

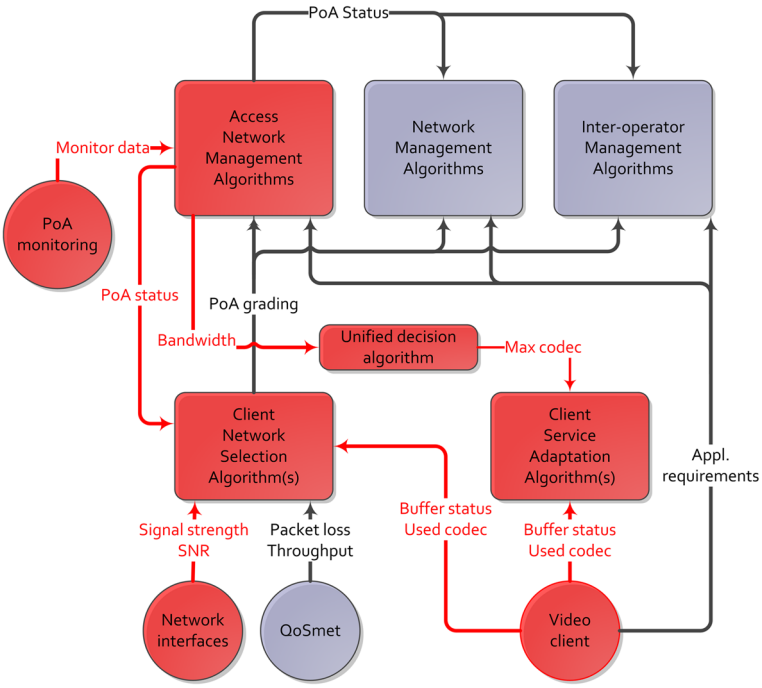

Fig. 3 Main inputs for the algorithms

clients (or a part of the clients' traffic) from an overloaded cell/network to other available cells/networks

Figure 3 showcases the information sharing process in the system. The PoA monitoring modules produce raw data on the status of the accesses. This information is processed further by the access network management algorithms and mapped to PoA statuses and bandwidth constraints. The client network selection algorithm provides grading of the available accesses by combining the local measurements of the network and application status with the generic PoA status information. The PoA gradings are used both by the client itself and by the network side mechanisms to optimize the access selection process. At the same time, the client service adaptation algorithm optimizes the parameters of the active applications by utilizing input from the application and the unified decision algorithm.

In this paper, we consider only one application for simplicity, that is, adaptive HTTP video streaming, but the approach can be extended to handle a more general setting with many simultaneous applications. For instance, LBA has already been experimented with file transfer type of traffic [16]. Such an extension would nevertheless require the inclusion of a broader set of application-specific performance metrics in the network selection part. So far, we have left the definition of metrics for other than adaptive HTTP video streaming for future work. Also, bitrate adaptation applies only to those applications and services supporting it, which would need to be taken into account in the design of a more generic UDA. The architecture would also support managing multiple simultaneous connections per Mobile Device provided that multihoming is enabled. This is however not considered in this paper.

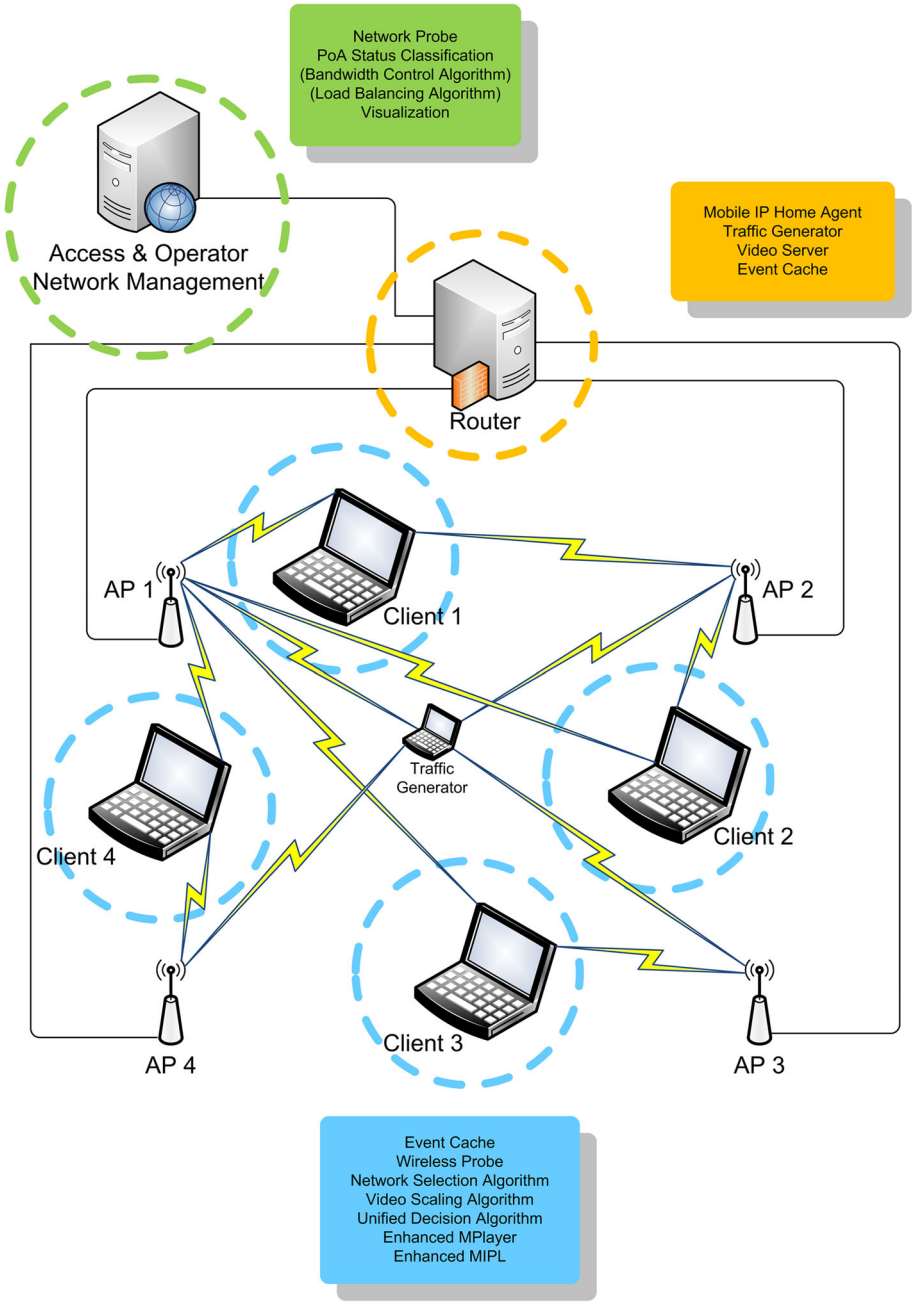

Fig. 4 Mobile video streaming management testbed

4 Prototype Implementation

For facilitating the testing and validation of the proposed system, we have built a multi-access network testbed. The testbed includes the main parts of the architecture, realized with WLANs, an adaptive HTTP video streaming platform, and Mobile IPv6 (MIPv6) based mobility management. These are then enhanced with the proposed cognitive management features. Moreover, the testbed includes prototype implementations of the proposed client-side decision algorithms. For the moment, the validation of the network-side decision algorithms has been conducted with network simulations in a more extensive scope. Thus, they are not considered in this paper. This section begins with a brief overview of the testbed followed by more detailed discussion on the implementation of the included components and algorithms.

4.1 Testbed

The testbed is illustrated in Fig. 4. The testbed includes four WLAN access points (*AP*), each forming an IP subnet, and four *Video Clients*, connecting to two of the WLANs each. The fifth client laptop runs the *Traffic Generator* software and is connected to all the WLANs. The clients, which are normal laptop computers, run Ubuntu 12.04 LTS. For access technology, AP1 supports IEEE 802.11g at 2.4 GHz and the other APs support IEEE 802.11a at 5 GHz. In the wired network, the *Router* is interconnecting the WLANs as well as connecting the *Access and Operator Network Management* computer to the rest of the network. On the network side the *Router* runs Ubuntu 12.04 LTS as well and the *Access and Operator Network Management* computer runs Windows 7.

All the Video Clients include the following components: the video streaming client application (i.e. *MPlayer*), *DDE Event Cache*, *Wireless Probe*, *NSA*, *VSA*, *UDA*, and the MIPv6 client (i.e. *MIPL*), but may be configured differently for testing purposes. In the network side, the Router machine runs the *Video Server*, *MIPv6 Home Agent*, network-side *Traffic Generator* as well as *DDE Event Cache*. Finally, the Access and Operator Network Management machine runs the network-side management components. In the testbed, we have the *Network Probe* and *PoA Status Classification Algorithm* currently deployed. The two decision algorithms given in parenthesis in Fig. 4 will be deployed into the same computer, once available for the testbed. Furthermore, the network management machine runs the Visualization Tool demonstrating the system's operation in runtime.

4.2 Actors

The testbed uses adaptive HTTP video streaming for the video service delivery. Specifically, the service is based on Apple's HTTP Live Streaming (HLS) technique, implemented into the MPlayer, and MPEG-2 TS encapsulation. This type of solution is widely employed, for instance, in IPTV services. During a streaming session, MPlayer requests pre-encoded video streams from the server (e.g. Apache) using the HTTP GET method according to a playlist/index file it receives at the beginning of the session. After receiving a certain amount of segments, MPlayer starts to decode and display the video frames. The server contains multiple bitrate segments for the same video in order to allow dynamic video bitrate adaptation at segment boundaries. In the testbed, MPlayer is one of the *actors* controlled by the cognitive network management system. That is, the management system instructs MPlayer to adapt the video stream bitrate based on the

Table 1 Message formats in the DDE framework

Type of message				Message fields				
Registration	Message type	Producer ID	Producer name	Event ID	Event name			
Subscription	Message type	Consumer ID	Consumer name	Consumer address	Event ID	Type filter	Producer filter	Digital signature
Event	Message type	Producer ID	Event ID	Event type	Time-to-live	Payload Length	Payload	Digital signature

algorithms' decisions via DDE, and MPlayer executes the action by starting to request different bitrate segments accordingly from the server.

MIPv6 was selected as a baseline interworking solution for the different IP-based AN technologies in the architecture. For a MIPv6 implementation, we employ the open source UMIP software (http://www.umip.org) in the testbed. In consequence of using MIPv6 for mobility management, the enforcement of the handover decision (i.e. action) is located in the Mobile Device, more specifically, in the MIPv6 client application (i.e. MIPL client).

Table 2 Events used in the system

Event ID	Event name	Description
Events from wireless probe		
1000	RSSI/noise	RSSI/noise
1002	Link speed	Negotiated link speed in Mbps
1010	TX/RX Bps	Bytes send/received (Bps)
Events from MIPL		
1100	Current PoA	Currently used PoA
Events from MPlayer		
2000	Buffer status	Receiver buffer status in MPlayer in percentage and used video representation
2001	BW requirement	Minimum BW requirement
Events from Video Scaling Algorithm		
2500	Adaptation recommendation	Video representation to use (1–5)
Events from Network Selection Algorithm		
3000	PoA list	Grading of available PoAs (the best first)
3001	Connection state	State of the current connection
3002	Handover recommendation	Client's name, current PoA ID, and destination PoA ID
Events from Unified Decision Algorithm		
4000	BW request	Request to increase amount of allowed BW
4001	Handover command	Interface name
4002	Adaptation command	Video representation to use (1–5)
Events from Network Probe		
5000	Short name	User defined event containing a KPI. The name of the event defines the content
5303	Capacity	Maximum capacity, currently used BW, and currently available BW in kBps
5310	Packet loss	Packet loss in percent
5504	Radio IF quality	Radio interface quality 1 and 2 (0–5)
5800	Nb of clients	Number of active wireless clients
Events from PoA Status Classification Algorithm		
6001	Network type	Network type (e.g. WLAN, LTE, 2G, 3G)
6502	Congestion	PoA congestion status (0/1)
6512	PoA status	PoA state (0–5)

Thus, the MIPL client is the other actor in the testbed, and it is interfaced with DDE for handover commands from the cognitive decision algorithms.

4.3 Knowledge Building and Dissemination

The testbed uses the DDE framework for knowledge building and dissemination. The main components involved in the process are the DDE Event Cache and a set of event sources that generate information regarding the system's status. The implementation of these components is detailed in this section. Also, the section lists the different types of events implemented into the testbed.

4.3.1 DDE Event Cache Implementation

The DDE Event Cache is designed to act as a common interface for different information producers as well as for decision algorithms and actors that use the information. It provides a distributed publish-subscribe message delivery system. The DDE Event Cache also includes information caching functionality for keeping the latest information available and up-to-date as well as information processing functionality for minimizing the data sent between network nodes. For communication, DDE uses socket interfaces and all the messages are encoded using XDR [25]. Information fields used in the different types of messages are shown in Table 1.

As shown in the table, there are three types of messages used in DDE, namely registration, subscription, and event messages. The message type field in each message is an integer that denotes the type of the message to the Event Cache. Before a producer is able to send events to the Event Cache, it has to send a registration message. Respectively, when leaving the system, the producer has to announce this to the Event Cache by sending another registration message. The Producer ID field in the registration message is the public key of the producer and the Event ID field is an integer identifying the registered event while producer name and event name fields are human readable information about the producer and event.

A consumer can order events with a subscription message from the Event Cache. In addition to fields similar to the registration message, the subscription message also contains the address in which the consumer listens to the events, two filter fields in order to restrict the type and producer of the event, and a digital signature for verifying the message. Event messages carry the actual information distributed within DDE. The event message includes an Event Type field, which specifies a subtype of a certain Event ID, a Time-to-Live field, which contains the validity time of the event in seconds, and naturally the Payload Length and Payload fields.

Our current testbed implementation includes 21 different events for transferring information throughout the system via the Event Caches. The events are summarized in Table 2. The events with IDs in the range of 1000–2001 carry information about the situation of the client and are used by the algorithms. The events with IDs in the range of 2500–4002 are the outputs of the client-side algorithms, and the events with IDs 5000–6512 carry network information originating from both the producers and algorithms.

4.3.2 Event Sources for the Algorithms

There are four information producers used in our testbed feeding the algorithms with information through DDE as listed in Table 2. Three of them: the Wireless Probe, MIPL, and MPlayer, are run at the client and the fourth, the Network Probe, is run on the Network Management Server. The operation and implementation of these components are described in the following.

The Wireless Probe is a wireless information producer. It runs a simple loop every second which reads information from /proc/net/wireless and the outputs of the ifconfig and iwconfig commands. From these, it parses the RSSI, noise, negotiated link speed, received bytes per second, and sent bytes per second values. It sends those out as events with IDs 1000, 1002, and 1010, respectively, when there is sufficient change in the values to warrant an update. RSSI and noise values are provided as percentage with a RSSI value 0 or noise value 100 meaning an unusable link. The negotiated link speed is provided in Mbps.

The MIPL software has been modified to produce information on the currently used PoA. It creates and sends an event with Event ID 1100 at startup and whenever the used PoA changes. The corresponding event contains the MAC address of the currently used PoA. The event's Time-to-Live value is set very high, so that the DDE Event Cache always has the information of the current PoA, even if there are no handovers.

The MPlayer software has been modified to produce information on the current buffer status and requested video bitrate to DDE. Three times a second, it sends an event with Event ID 2000 that tells which percentage of the buffer is filled and which representation of the video is being requested from the server. In the testbed, we use a pre-encoded H.264 AVC video with 5 different representations (i.e. bitrates and fidelities), hence the numbering 1–5. MPlayer sends also another event (ID = 2001) that tells the minimum bandwidth in kbps required by the current video representation. This event is sent every time the requested video representation changes, that is, at the beginning and whenever the video bitrate is increased or decreased.

Finally, the Network Probe monitors the PoAs and produces events based on the traffic in each interface and the number of clients associated with the PoA. The Network Probe implementation is access technology specific. The WLAN Network Probe used in the testbed employs Simple Network Management Protocol (SNMP) to access the data stored in the WLAN APs' Management Information Bases (MIB). The probe uses a three step event generation process. First it generates SNMP queries for each device. After that, it reads the counter values from the MIBs and generates key performance indicators (KPI) from the raw counters. And finally, it checks which events are defined to be send and sends them if the time limits are reached. In our system, the Network Probe is configured to produce events with IDs 5000, 5303, 5310, 5504, and 5800. Event ID 5000 makes an exception in the list because it represents a generic KPI event—the name of the event is used to define the content. The other events are specified for one purpose only.

4.4 Management Algorithms

This section details the algorithms supported in the current testbed implementation. Two of the client-side algorithms, namely NSA and VSA, support RL. Therefore, this section begins with an introduction to the used approach. Furthermore, the client-side decision-making is assisted by the network by two mechanisms, also briefly presented in this

section. Finally, we introduce the implementations of the algorithms, specifically designed and developed for the use case considered in this paper, that is, NSA, VSA, and UDA.

4.4.1 General Control Approach

The general control approaches for access selection and video adaptation are both based on RL [26]. In our case, the controllers are functioning in discrete time steps and each control loop contains the following steps, as illustrated also in Fig. 5:

1. Analyze and classify the current status of the system;
2. Calculate the rewards and update the control policies;
3. Select the control action based on the current state and knowledge gained earlier;
4. Measure the system changes.

The complexity of the classification process can vary a lot. In the case of the video client management, the state is defined simply by the current playout buffer length and the current video bitrate. On the other hand, grading the available wireless connections comprises several sub-processes which utilize SOM, clustering, principal component analysis (PCA), and fuzzy interference.

The instantaneous rewards R_t, calculated for each time step t, are defined to be application aware so that the clients' QoE would be maximized. The overall objective of the controller is to maximize the expected discounted reward

$$\mathrm{E}\left(\sum_{t=0}^{\infty}\gamma^t R_{t+1}\right),$$

where the discount parameter $\gamma \in [0, 1)$ determines how much weight the short and long term revenues have.

In the testbed, we have utilized Q-learning based controllers [27]. These utilize so-called Q tables to learn the expected discounted rewards for all possible actions $a \in \mathcal{A}$ given the current system state. In the testbed, the applied control policies follow the rule: Given the state $X_t = x$ and an action a, the higher the value $Q_t(x, a)$, the more likely the action a will be selected at time t. The ϵ-greedy and Boltzmann exploration are typical examples. The former selects the best action (based on the current knowledge) with probability $1 - \epsilon$ and otherwise a random action. In Boltzmann exploration, the probability of performing the action a, when the system is in state X_t at time t, is given by

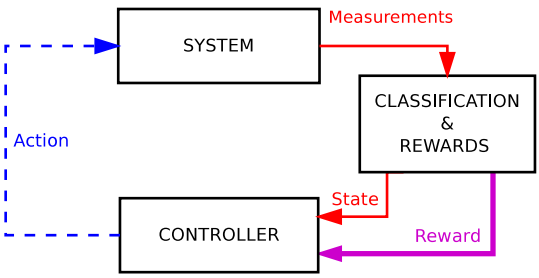

Fig. 5 General control approach

$$\pi(X_t, a) = \frac{e^{\beta Q_t(X_t,a)}}{\sum_{a' \in A} e^{\beta Q(X_t,a')}}. \quad (1)$$

Once an action A_t is applied, then the knowledge update rule for transition $(X_t, A_t) \longrightarrow (X_{t+1}, R_{t+1})$ is given by

$$Q_{t+1}(X_t, A_t) = (1 - \alpha_t) Q_t(X_t, A_t) + \alpha_t \left(R_{t+1} + \gamma \max_{a \in \mathcal{A}} Q_t(X_{t+1}, a) \right), \quad (2)$$

where α_t is the learning rate parameter at time t.

4.4.2 Network-Side Mechanisms

In the testbed, the client-side algorithms are assisted by two network-side mechanisms, namely the WLAN Network Probe and PoA Status Classification Algorithm. These components were developed in an earlier project and introduced in [28]. For the purposes of this paper, the components were integrated into the DDE-based management framework and adapted to the considered use case and testbed. In this section, we recap their implementation details.

4.4.2.1 Network Probe for WLAN The WLAN Network Probe monitors traffic of different WLAN components, for example, APs and routers. As explained in Sect. 4.3.2, it uses SNMP to access the data stored in the devices' MIB tables. The Probe configuration dictates how the KPIs are created from the raw counters. Formulas may be applied for computing the KPIs over defined time windows and adding parameters for sending the events to the DDE Event Cache, such as event IDs and threshold values.

4.4.2.2 PoA Status Classification Algorithm The PoA status classification algorithm is a SOM-based solution depicted in Fig. 6. The resource probe(s) (e.g. WLAN Network Probe) pre-processes the data and generates network dependent and independent KPIs. The monitored data is fetched to the algorithm via DDE for autonomous or semi-autonomous

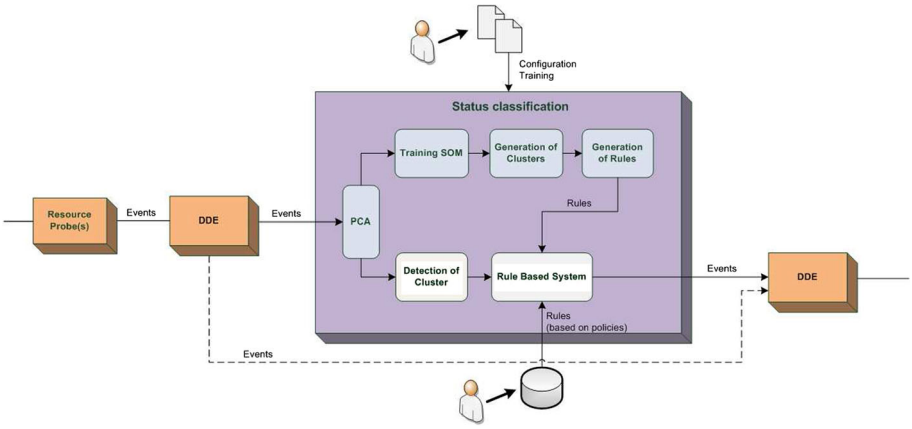

Fig. 6 SOM based access classification

decision-making. The algorithm requires a training before it is assigned to decision-making. An artificial training data is generated (or specific training data, collected from resource probes, is read from a file) and forwarded to PCA component to reduce high dimensional data sets to lower dimensions. After PCA, SOM training is done to reduce the data dimensions further yet maintaining the topological relationships of the data. Next, SOM clustering is performed with K-means, and the features of the detected clusters are discovered. After the clustering, the interpretation of each cluster is resolved by comparing special labelling data with the cluster data profiles. Finally, the characteristics of detected clusters with the associated rules are automatically saved for a later use. The training using PCA, SOM, and K-means algorithms are described more detailed in [28].

After training, the algorithm is set into action. Each time the algorithm receives events from probes, it constructs data samples from them. The correctness of the samples is validated and erroneous ones are discarded. For each sample, PCA is performed and the outcome is compared against the discovered clusters. When the algorithm detects a data sample belonging to a specific cluster, it fires the associated rules, and if needed, generates events for DDE as specified in Table 2.

4.4.3 Mobile Video Client Management

In the testbed, the client-side management is realized with three algorithms: NSA, VSA, and UDA, which were specifically developed for the use case considered in this paper. This section details their implementation. We have developed also non-learning versions of NSA and VSA for evaluation purposes, and they are presented in this section as well.

4.4.3.1 Network Selection Algorithm (NSA) NSA's main responsibility is to select a PoA, which offers the best user experience for the video streaming service. It strives to achieve this by giving handover recommendations when necessary. If NSA is content with the current quality of the service or if a handover would not benefit the user experience, NSA recommends to stay in the current PoA. On the other hand, if NSA decides that a handover is beneficial, a handover recommendation is sent to DDE. In the testbed, the number of each client's wireless connections is limited to two WLAN APs, but NSA could also be used for selecting the best PoA from more than two choices. It would be able to support different access technologies (e.g. WLAN, 3G, LTE) as well, provided that necessary input data is available.

Input data for the algorithm is divided into three categories: PoA related inputs, client related inputs, and application related inputs. The PoA related inputs contain data about the status of the PoA from the PoA's point of view, that is, how does the PoA see its own condition. The client related inputs contain data about the status of the PoA from the client's point of view. The application related inputs contain data about the quality and performance of the video stream at MPlayer. The input events for NSA originate from four producers: MIPL, Wireless Probe, MPlayer, and Network Probe. The input events are RSSI and noise (ID 1000), link speed (ID 1002), MPlayer buffer status and video representation (ID 2000), and PoA status (ID 6512). NSA also needs the client's current PoA (ID 1100) from MIPL, which is updated every time the client makes a handover. In addition, initialization data is read from a configuration file (e.g. Q-learning initialization parameters).

NSA monitors both the quality of the user's service on the current PoA and the alternative quality on another PoA, simultaneously. Alternative quality means the quality the client would experience on another PoA if a handover was made. The quality of the current PoA and the alternative PoA is quantified by a grade, which is an integer from 0 to 5, with 0 meaning a

bad or non-existent connection and 5 meaning a nearly perfect connection. NSA produces three output events; *PoA list* (ID 3000), *connection state* (ID 3001), and *handover recommendation* (ID 3002). *PoA list* contains the grades and ID's (i.e. WLAN SSID) of all discovered PoAs. *Connection state* is a grade given for the quality of the user's video service in the current PoA. Whenever the client receives new data, *connection state* and *PoA list* are updated. The *PoA list* expands, if the client moves to an area where there are new previously undiscovered PoAs, or shrinks, if any PoA goes out of the client's range. *Connection state* and *PoA list* events are triggered when new events arrive at the client and periodically based on the time-to-live (ttl) parameter. This means that a new *PoA list* or a *connection state* event is produced and sent when new data arrives. If no new data arrives and ttl expires, events 3000 and 3001 are sent again with the same payload as previously. The Q-learning controller works in 4 s intervals so that a new *handover recommendation* event is calculated every 4 s even if no new data has been received. *Handover recommendation* event includes the client's name, current PoA ID, and the destination PoA ID.

Figure 7 gives an overview on the implementation of the algorithm. *Fuzzy classifier 1* calculates the current PoA grade (G_1) from *Input data set 1*. Grade G_1 is the grade sent in the *connection state* event. *Fuzzy classifier 2* calculates all alternative PoA grades (G_2) from *Input data set 2*. These are collected into the *PoA list* event. We need to use two separate fuzzy classifiers because all data is not available for alternative PoAs (e.g. link speed) since the client has no active connection to them. Contents of the input data sets can be seen in Table 3. In general, Q-learning algorithms cannot use arbitrarily large state spaces, so to keep the size of the state space under control, grades are quantized into an integer between 0 and 5. Grade G_1 and the best alternative PoA grade G_2 are given to the *State classification* block, which determines the current state of the algorithm in the following way. Every state of the algorithm can be described with current PoA grade $G_1 \epsilon \{0, 1, 2, 3, 4, 5\}$ and alternative PoA grade $G_2 \epsilon \{0, 1, 2, 3, 4, 5\}$. With grades G_1 and G_2 together, we can denote the state of the client as G_1:G_2. Six G_1 grades and six G_2 grades give a total of 36 permutations for G_1:G_2. That is, the 1st state is 0:0, the 2nd state is 0:1, ..., the 35th state is 5:4, and the 36th state is 5:5. The state is then given to the *Q-learning controller*, which decides the next control action. There are two actions in each state: *action 0* and *action 1*. When *action 0* is chosen, the client stays in the current PoA and when *action 1* is chosen, NSA sends a *handover recommendation*. With the

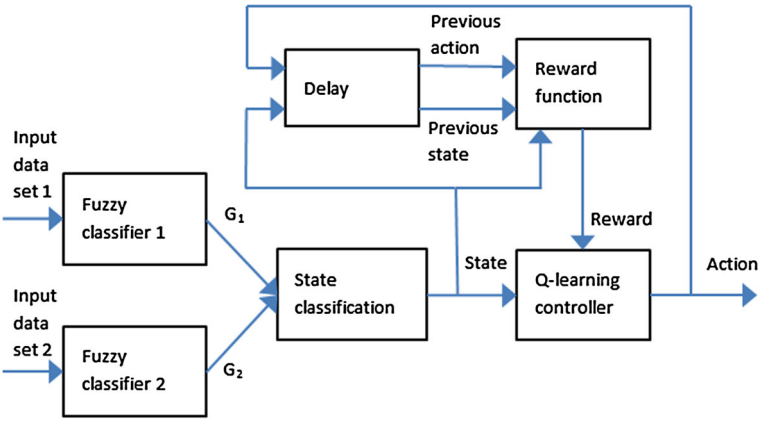

Fig. 7 Network selection algorithm

Table 3 Event data used by fuzzy classifiers

Event ID	Event name
Input data set 1:	
1000	RSSI/noise
1002	Link speed
2000	Buffer status
6512	PoA status
Input data set 2:	
1000	RSSI/noise
6512	PoA status

two actions and 36 states, there are a total of 72 values in the Q-table. NSA usesϵ-greedy exploration with $\epsilon = 0.01$.

Learning of the algorithm is based on the *Reward function*. The state-action pair is stored into NSA's memory, which is depicted as *Delay* in Fig. 7. Reward gained by the *Q-learning controller* depends on the previous state-action pair and the current state. *Reward function* is presented for each state-action pair in Table 4. The controller can make smart decisions or unwise decisions and the amount of gained reward reflects this. An example of a smart decision is if the controller selected action 1 (handover) when the PoA grade was 0 and the current PoA grade is 4. This gives the controller a 150 point reward. An example of an unwise decision is if action 1 is selected while PoA grade is 5 and the resulting PoA grade is 0. This decision is punished with a reward of −50. The PoA grade may also get better or worse without NSA's input. For example, when action 0 is selected while PoA grade is 1 and the resulting PoA grade is 3. This 'lucky break' gives a 25 point reward. Rewards in Table 4 have been found experimentally. How to find the optimal rewards is beyond the scope of the current paper and is left for future work.

4.4.3.2 Non-learning Network Selection Algorithm In order to assess the performance and usefulness of the NSA with Q-learning support, a NSA without any learning capabilities (i.e. non-learning NSA) was developed. Both the algorithms use the same input data. The regular NSA exploits fuzzy classification for calculating the current state of the client, and then Q-learning for determining whether to generate a handover recommendation. The non-learning NSA exploits the same fuzzy classifiers as the regular NSA in order to calculate the current

Table 4 Reward function

Previous PoA grade	Current PoA grade	Reward with action 0	Reward with action 1
0 or 1	0 or 1	0	−1
0 or 1	2 or 3	25	50
0 or 1	4 or 5	25	150
2 or 3	0 or 1	0	−25
2 or 3	2 or 3	50	0
2 or 3	4 or 5	25	100
4 or 5	0 or 1	0	−50
4 or 5	2 or 3	0	−25
4 or 5	4 or 5	100	0

state but the Q-learning component has been substituted for a more straightforward solution. As described earlier, the state of the algorithm depends on the current PoA grade G_1 and the alternative PoA grade G_2. The non-learning NSA then decides:

- If $G_1 < G_2 \Rightarrow$ send handover recommendation or
- if $G_1 \geq G_2 \Rightarrow$ stay in current PoA.

The non-learning NSA also uses the same reward function as the regular NSA. This provides a way to compare the rewards gained when using the two algorithms. The non-learning algorithm is susceptible to ping-pong effect because even minor traffic variations may cause the current PoA grade G_1 to drop below the alternative PoA grade G_2.

4.4.3.3 Video Scaling Algorithm (VSA) VSA sends video bitrate adaptation recommendation events (ID 2500) based on the current playout buffer status and the requested video bitrate. The algorithm is based on RL, and especially tabular Q-learning described earlier in Sect. 4.4.1. The algorithm is periodically triggered by the reception of the buffer status event (ID 2000) produced by the MPlayer. The event message contains the current playout buffer status in percentage, and the video representation used for the streaming. The playout buffer status (0–100 %) is divided into 3 zones: *green* ($\geq$90 %), *yellow* (60–90 %), and *red* (<60 %). Red means that the buffer is empty or almost empty, green that it is adequately filled, while yellow presents something between them. Based on the tests, it was found that the red zone should be quite large in order to prevent buffer underruns. Also, the green zone should be quite near to the maximum, since there is no point to try to increase the video bitrate, if there are already problems to keep the buffer filled with the lower bitrate. As mentioned earlier in this paper, the video used in the testbed has 5 different representations of differing bitrates.

The allowed actions A are to increase or decrease the video bitrate by one step, or keep it the same. For each action option, a probability is calculated by using the action-value Q function values. Q function keeps track on the reward for each (system state X_t, action A_t) pair. The bigger the reward, the bigger the probability to choose that action in that system state also in the future.

The action is selected by using Boltzmann exploration policy Eq. (1) with $\beta = 0.08$. The reward function R is defined as

$$R = \#\text{quality} * \log(1 + \text{Buffer\_status}) - \text{Penalty},$$

where #quality is the representation of the video (1–5) used for the streaming (i.e. the higher bitrate, the higher fidelity and quality), Buffer_status is the playout buffer length (0–100), and Penalty = 30 is given if the implemented action turns out to be a bad action. That is, when the buffer zone drops as a consequence of increasing or keeping the video bitrate. Q function values are initialized in order to prevent some illegal states (e.g. 0 or too high bitrate) and speed up the learning process by preventing some bad choices (e.g. decrease when in green zone, increase when in red or yellow zone), and by forcing some good choices (e.g. decrease when in red). When updating the Q function values according to Eq. (2), we used learning rate parameter $\alpha_t = 0.5$ and discount parameter $\gamma = 0.7$. The reward was not calculated right from the next arriving buffer status event message, but instead some time was given to the action to take some effect to the buffer status. When the action was to increase the bitrate, the reward was calculated from the 10th event message if the buffer zone remained in the green zone. If it was in the yellow or red zones, the reward (i.e. penalty in this case) was calculated based on that event. If the action was to decrease

the bitrate, the reward was calculated from the 5th event message. MPlayer sends the buffer status events 3 times a second.

4.4.3.4 Threshold-Based Video Scaling Algorithm For comparison, a threshold-based VSA was developed. Just like the Q-learning based VSA, also this one takes the buffer status and current video representation values as inputs. If the buffer status is on the green zone, it increases the video bitrate. If the buffer status is on the red zone, it decreases the video bitrate. On the yellow zone, it does nothing. Thus, on the red zone, it behaves like the Q-learning algorithm (decreases the video bitrate), but on the yellow and green zones the behaviour may be different: the Q-learning algorithm may also drop the video bitrate on the yellow zone, and on the green zone it does not necessarily always increase the video bitrate. The threshold-based VSA may cause more adjustments to the chosen video representation. This is because even during high congestion, the buffer status may sometimes reach the green zone. The Q-learning algorithm may learn that it is not wise to increase the video bitrate every time when the buffer status reaches the green zone.

4.4.3.5 Unified Decision Algorithm (UDA) The purpose of UDA is to coordinate the actions triggered by the decisions of the other algorithms. UDA runs on the client and coordinates the actions suggested by the different decision algorithms. It receives information on the decisions taken by all the algorithms included in the system, that is, both network- and client-side, via DDE, and makes the ultimate decision on the optimization action to be enforced in a given situation. The current UDA design takes into account also the network-side LBA in addition to NSA and VSA, although its implementation is currently not available for the testbed. Nevertheless, LBA is considered as an important future extension and thus cannot be excluded.

In the proposed system, UDA has three main functions: (1) to prioritize the decisions of LBA over those of the NSA, (2) to prevent erratic behaviour by implementing a withdrawal period after a handover, and (3) in case of a handover, adapt the bitrate of the video stream to fit the target network.

The UDA implementation is designed to prioritize the decisions of the LBA over the decisions of the NSA. This is to allow the operator to have control over terminals connected to its networks. The prioritization is done by applying a configurable withdrawal period for UDA after a handover decided by the LBA, during which UDA rejects handover suggestions from NSA. This way, the operator can distribute the network load with minimal interference from the clients and clients can then further optimize their connections after the withdrawal period.

Furthermore, in order to prevent erratic behaviour during and after handovers, UDA applies a short configurable withdrawal period after each handover, during which time all other handover and video adaptation decisions are rejected by the UDA. This prevents both oscillating handover behaviour (i.e. ping-pong effect) as well unnecessary video adaptation due to short connection breaks caused by handovers. Additionally, when accepting a handover decision, UDA evaluates the need for video adaptation based on currently used video bitrate and currently available bandwidth in the target network. If the currently used video bitrate is more than the currently available bandwidth at the target network, UDA issues a suitable video adaptation command to lower the bitrate before issuing the handover command. This assures that the handover to the target network will not cause congestion.

Aside from the special situations mentioned above, UDA always respects the decisions made by LBA, NSA, and VSA. When UDA receives a decision from an algorithm, it

promptly sends a command to the respective actor. In order to maintain up-to-date information for the evaluation of video adaptation demand in case of handovers, UDA also subscribes the events for the currently used video bitrate and the available bandwidth in the networks in addition to the decisions made by the algorithms.

4.5 Testing and Demonstration Tools

The testbed includes tools for testing and demonstration purposes. Traffic generator software (Iperf) is deployed in the Router and an additional multi-access client terminal simultaneously connected to all the WLANs. The Traffic Generators allow introducing additional traffic to the networks and thus testing the algorithms' performance under varying load conditions. We have also developed a Visualization Tool, which can be used for showing the event dissemination within the system as well as the decisions taken by UDA (i.e. selected access and video bitrate) in real-time. The Visualization Tool shown in Fig. 8 is implemented in Python using the Cocos2d visualization library (http://cocos2d.org), and it gets the event information through the visualization interface of the DDE Event Cache.

5 Experimental Evaluation

This section describes the experimental evaluation conducted on the current testbed implementation. The purpose of the testing was to evaluate the whole system's performance when all the decision algorithms are simultaneously active. This section briefly introduces the test setup and scenario, as well as provides a summary and discussion on the main results obtained.

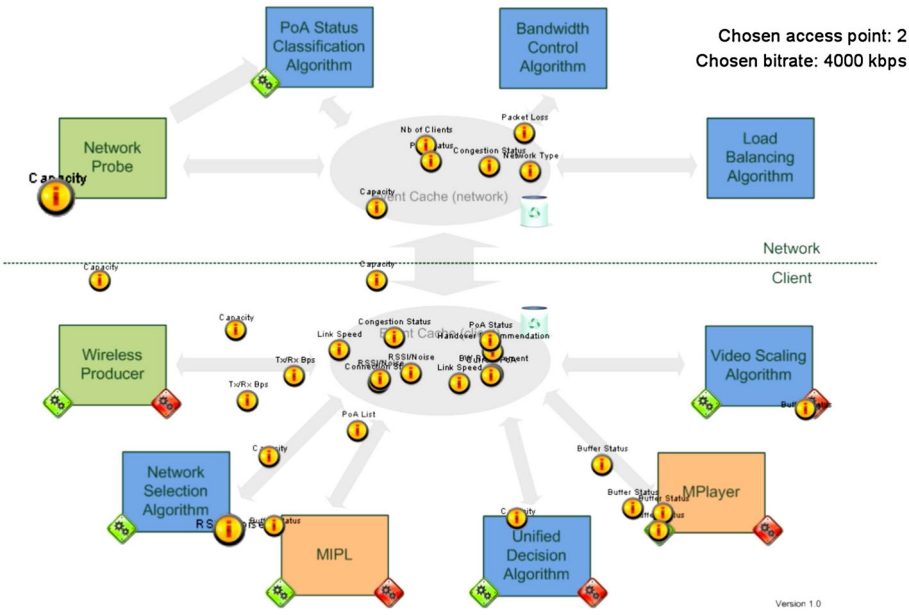

Fig. 8 The visualization tool

5.1 Scenario

The experimental evaluation of the proposed system was performed using a testbed configuration shown in Fig. 9. The purpose of the testing was to evaluate the performance and joint operation of the client-side decision algorithms under varying network load. For this, the testbed was configured to include three Video Clients of different types:

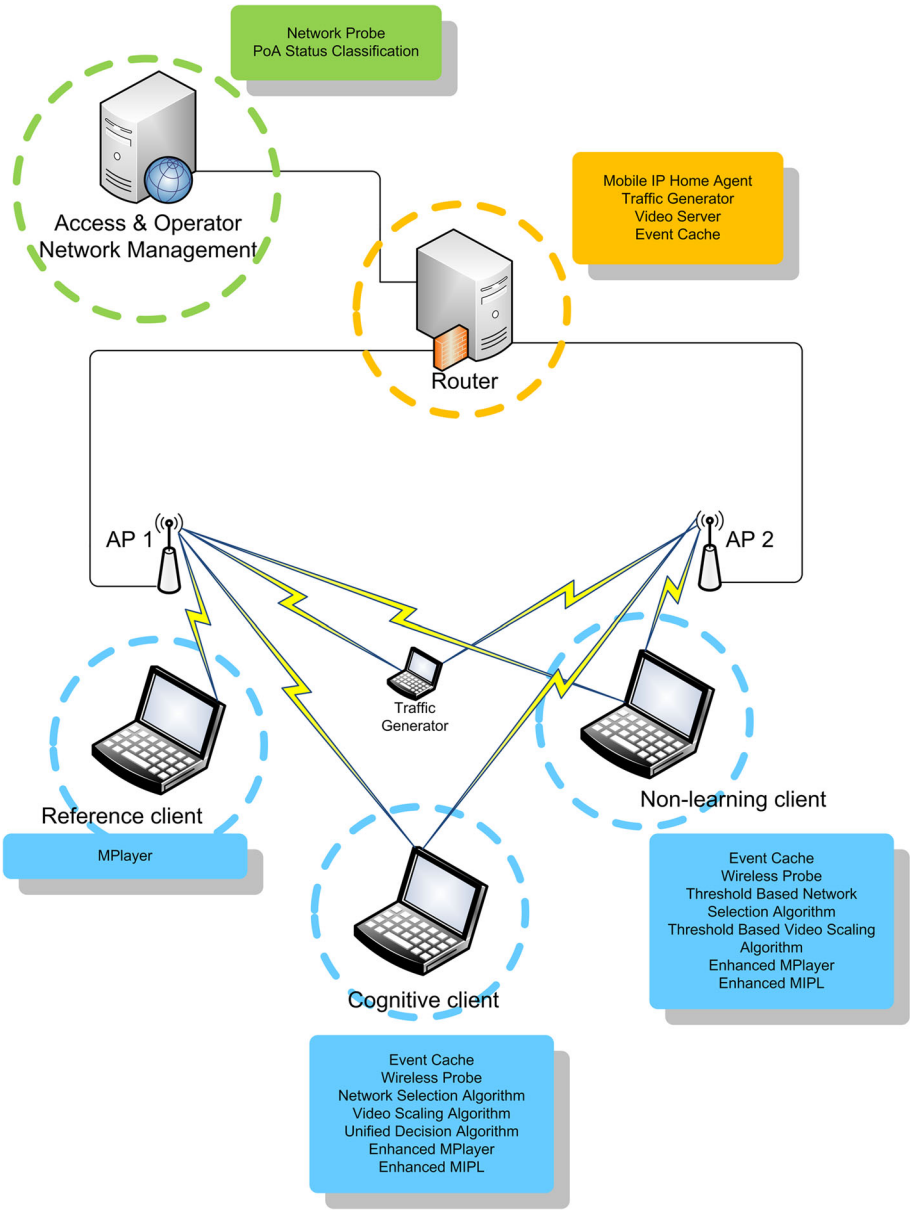

Fig. 9 The testbed configuration used in the evaluation

- *A cognitive client*, supporting the proposed management system and cognitive decision algorithms.
- *A non-learning client*, having a limited configuration with the basic threshold-based handover and video adaptation algorithms.
- *A reference client*, supporting no handovers and video streaming only in the full quality (i.e. bitrate).

In the setup, two WLAN APs, namely AP1 and AP2, are simultaneously available to the cognitive client as well as to the non-learning client. The reference client is constantly connected to AP1. AP1 uses 802.11g in the 2.4 GHz band and AP2 802.11a in the 5.0 GHz band.

The test scenario was as follows. A Video Client is receiving the video streaming service while additional UDP-based background traffic is introduced into the APs symmetrically to both uplink and downlink directions. The different types of Video Clients were tested separately, that is, only one Video Client was activated at a time during the tests in order to evaluate the system-level performance without the impact of competing video streams. More complex test scenarios are left for future work. For instance, the performance of LBA has already been experimented with multiple clients in a simulator environment, but the results will be published in a follow-up paper.

The characteristics of the video sequence used are listed in Table 5. The background traffic profile is illustrated in Fig. 10. The traffic profile was chosen experimentally in order to evaluate the response of our proposed system under different load conditions in the available networks. It includes transitions from low load to highly congested network conditions for AP1 and a constant high load level for AP2. The traffic profile also includes two 60-s-long periods in an especially interesting network condition, during which both of the APs are loaded with the same amount of background traffic.

The evaluation included the three client-side algorithms, namely NSA, VSA, and UDA, as well as the PoA Status Classification algorithm in the network. The cognitive client supported them all. The non-learning client run the non-learning or threshold based versions of NSA and VSA, and no UDA. The reference case did not support any decision algorithms, adaptations or handovers. The locations of the algorithms are shown in Fig. 9.

5.2 Results

The test scenario was run ten times consecutively on our testbed, separately for each type of client. In this section, we summarize the main results obtained in the evaluation. We

Table 5 Test video sequence characteristics

Video sequence	
Compression type	H.264 AVC
Encoder	FFMpeg
Sequence name	Sintel
Number of videos with different bitrate	5
Bitrate (kbit/s)	200–500–1000–2000–4000
Frame rate (fps)	25
Segmentation length (s)	1

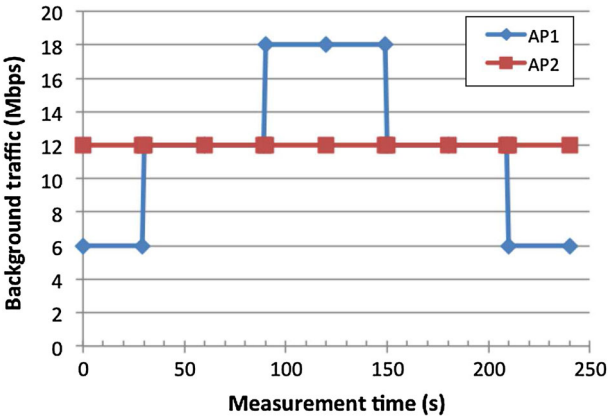

Fig. 10 Background traffic used in the evaluation

measured the status of the client's video buffer while monitoring changes in the used AP and requested video bitrate.

Figure 11 shows the fill factor of the video buffer (%) for each type of client during one representative test run. As can be seen from the figure, the reference client suffers the most from the background traffic, while the non-learning client performs much better. Of the three clients, the cognitive client experiences the smallest drops in the video buffer fullness which also do not lead to gaps in the video playout. Overall, the mean video buffer fill factor (i.e. the mean value of how full the video buffer is across all the measurement runs) is 93.4 % for the cognitive client, 87.1 % for the non-learning client, and 77.0 % for the reference client.

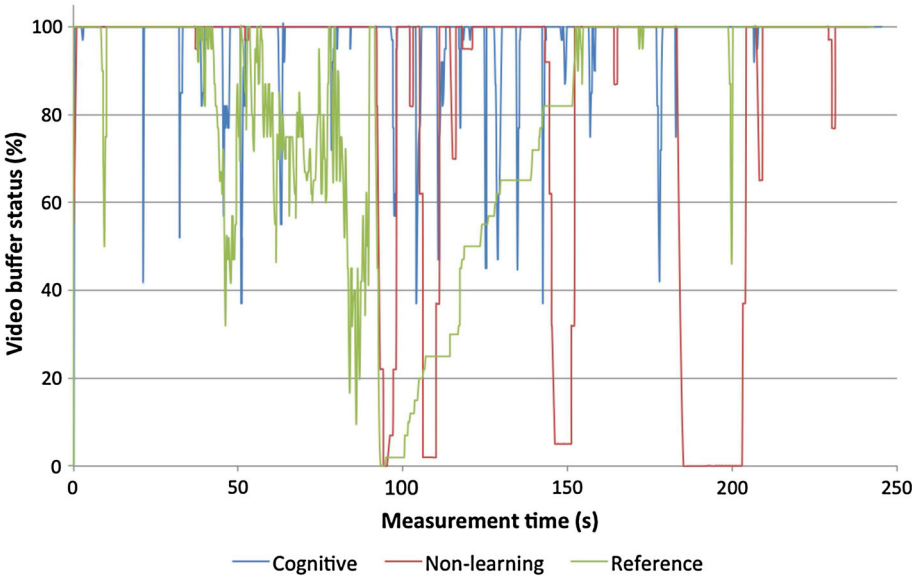

Fig. 11 Video buffer status

Assuming that the video playout stalls whenever the buffer fill factor drops below 20 % (i.e. the player nolonger has enough data to compose a full frame from the buffer) and continues after the fill factor has risen back to 80 % (i.e. the player buffers until this level in order to minimize future gaps in the playout), then on average, the video playout on the cognitive client would have stalled 3.4 times per test run and the video would have stayed stopped altogether for 11.9 s. The corresponding values for the non-learning client are 3.6 times and 25.7 s, and for the reference client 7.2 times and 69.6 s. In our representative test run depicted in Fig. 11, the exact values are one 0.33 s stop right at the beginning for the cognitive client, four stops totalling 34.04 s for the non-learning client, and two stops totalling 55.5 s for the reference client. Based on these values, we can state that the clients with support from the proposed management system offer significantly better video viewing experience. Furthermore, the cognitive client performs much better than its non-learning counterpart.

Figure 12 plots the chosen bitrate for the video streaming during the same representative test run as in Fig. 10. As can be seen from Fig. 12, the reference client with no means to adapt tries to stream constantly with the full bitrate of 4 Mbps. The non-learning client adapts the bitrate far more seldom than the cognitive client, which adapts the bitrate very frequently. This behaviour is also reflected on the mean bitrates requested by the clients. The reference client's mean chosen bitrate is naturally 4 Mbps. The mean bitrate for the non-learning client is 3739 kbps, and for the cognitive client 2925 kbps. On the other hand, the cognitive client makes on average only 7.5 handovers per test run, while the non-learning client makes 14.9 handovers. The reference client with no means to switch networks does not make any. Figure 13 plots the chosen AP for the different clients during the same representative test run. As can be seen from the figure, the reference client always stays connected to AP1 and the non-learning client makes the most handovers (16) during the test run. The cognitive client makes 6 handovers during this example test run.

Overall, the best QoE was subjectively observed on the cognitive client. It made less handovers than the non-learning client, and UDA adjusted the video bitrate according to

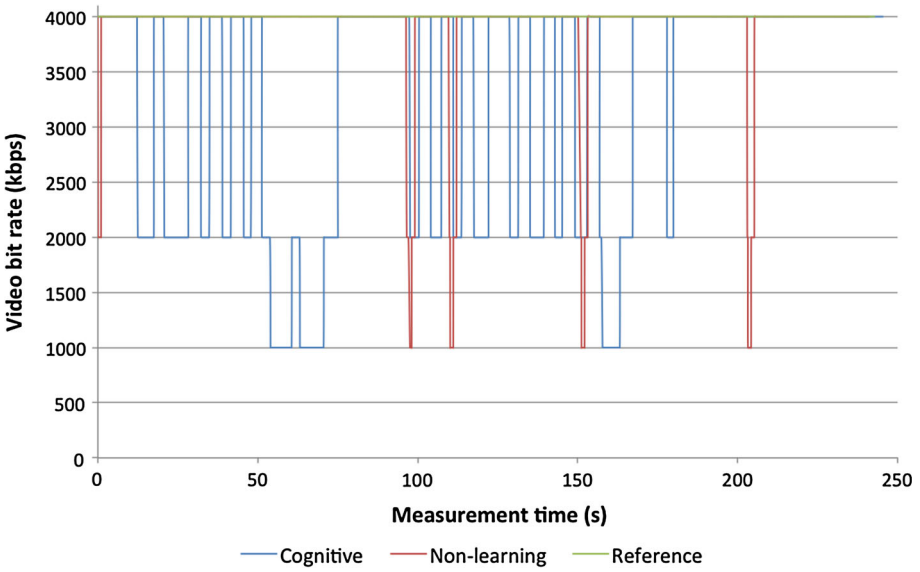

Fig. 12 Chosen video bitrate

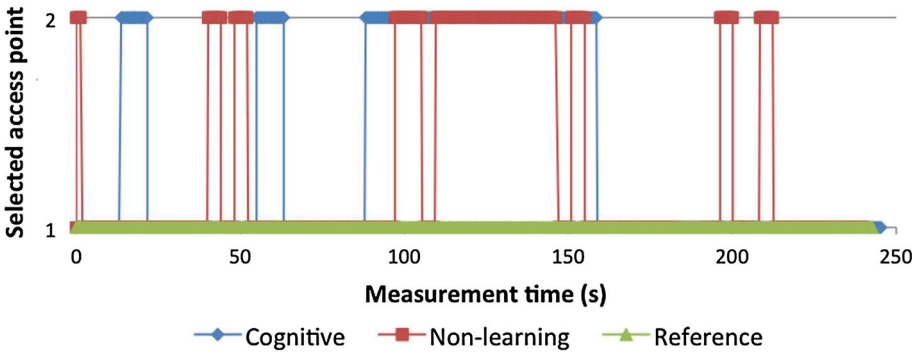

Fig. 13 Selected access point

the available bandwidth in the target network. Partly due to that, it suffered from stalling less frequently than the non-learning client. Also the video adaptation algorithm played a role in stalling frequency. As discussed earlier in Sect. 4.4.3, the non-learning, threshold-based video adaptation algorithm turned out to be too greedy to increase the video bitrate (increases always in the green zone), and too slow to decrease it (does nothing on the yellow zone), which contributed to the higher stalling frequency. On the other hand, the non-learning client used a higher bitrate on average than the cognitive client, and made adaptations less frequently than the cognitive client, which are both good for QoE. However, stalling is the worst degradation and it has to be avoided at the costs of the initial delay or quality adaptation [29]. Also, the most dominant impact of adaptation on QoE is the adaptation amplitude [29], and not the frequency (even though it is also important). The adaptation amplitude was minimized in our video scaling algorithms by allowing bitrate changes only one level up or down at a time. In order to avoid flickering or ping-pong effects, that is, too high adaptation frequency, the VSA and UDA implementations included some stabilizing time for the adaptations to take effect. Thus, the most negative effects of the adaptations on QoE were minimized and the observed difference in QoE was mainly due to the frequency and length of the stalling periods.

5.3 Discussion

Based on the evaluation results, we can state that the proposed cognitive network management solution can be used for enhancing video streaming performance in multi-access networks. The performance criterion considered in the evaluation was video streaming QoE, which was estimated based on the video player's buffer status as well as the requested video bitrate. The buffer status indicated the smoothness of the video playout whereas the bitrate mapped to the visual fidelity of the video. These criteria were considered as sufficient for the purposes of the present work, taking into account the fact that no suitable objective QoE estimation tools were available for the considered setup, and performing a subjective analysis was out of the scope of the work. In addition, being simple parameters used by the player, they could be easily made available to the decision-making algorithms via DDE. Also, cognition helped to reduce the number of handovers conducted during the tests. This is desirable as handovers always cause additional signaling overhead to the network as well as may have negative impact on QoE depending on the handover delay and incurred packet loss. In our setup, however, the last two were

minimized due to the fact that the L2 links of the video clients were constantly established during the evaluation.

Moreover, the proposed algorithms improve the network management system's stability and tolerance of uncertainty. Especially, the dynamic learning capability included in NSA and VSA can reduce the ping-pong effect in the decision-making. Overall, the reliance on multiple information sources helps the algorithms to cope with corrupted or incomplete information, which can easily happen in real networks.

A direct extension to the current work is the evaluation of the network-side algorithms that are capable of optimizing the operation of multiple clients based on a more extensive view of the status of the overall network environment. In the first phase, this work is conducted in a simulator in order to have a more extensive network environment for validation. The results are planned to be published in a follow-up paper. Later on, the network-side algorithms will be integrated into the testbed. This continuation work will also consider the scalability of the different management approaches (i.e. client- or network-side) in terms of the required signalling overhead. Additionally, the network-side algorithms will allow including more dimensions to the management, for instance, by classifying the mobile devices into 'gold', 'silver', and 'bronze' users. In the client-based strategy evaluated in this paper, the mobile devices just optimize their own operation based on the status of available networks without knowledge of the actions of the other users.

6 Conclusions

Cognitive network management can provide a solution for managing the complex networks of the future. This paper proposed a cognitive network management solution for optimizing video streaming performance in heterogeneous multi-access networks. The solution addresses a topical problem of finding means to manage the constantly increasing video traffic loads over wireless and providing acceptable QoE to the end users. The paper presents a cognitive network management framework for optimizing video stream bitrate adaptation and routing based on novel decision algorithms. The algorithms are arranged hierarchically for local optimization and scalability. The work presented in the paper includes also prototype implementations of the framework and algorithms as well as their integration into a multi-access testbed. Additionally, the paper describes an experimental evaluation of the client-side management components that are assisted by network-side knowledge. The results obtained in the evaluation attest the feasibility of the solution as well as the benefits of cognitive decision techniques over non-learning or non-adaptive approaches. The future work will include a more extensive evaluation of the whole system, including the network-side decision algorithms specified for the framework. In the first phase, the evaluation will be conducted in a simulator environment for more extensive testing, but eventually the network-side decision algorithms will be integrated into the testbed as well. Additionally, the inclusion of other access technologies besides WLAN (e.g. LTE) into the testbed as well as the technologies' impact on the knowledge building methods and decision algorithms will be studied as a part of the future work.

Acknowledgments The work reported in this paper was partly supported by the Finnish Funding Agency for Technology and Innovation (Tekes) in the framework of the EUREKA/Celtic Cognitive Network Management under Uncertainty (COMMUNE) Project. The authors would like to thank their colleagues who have contributed to the project and especially those who have participated in the implementation of the prototype.

References

1. Thomas, R., Friend, D., DaSilva, L., & MacKenzie, A. (2006). Cognitive networks: Adaptation and learning to achieve end-to-end performance objectives. *IEEE Communications Magazine*, *44*(12), 51–57.
2. Fortuna, C., & Mohoric, M. (2009). Trends in the development of communication networks: Cognitive networks. *Computer networks*, *53*(9), 1354–1376.
3. Qi, J., Zhang, S., Sun, Y., & Sun, Y. (2010). Challenges for cognitive network. In *Proceedings of WiCom 2010*.
4. Cisco: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017.
5. Sherman, M., Mody, A., Martinez, R., Rodriguez, C., & Reddy, R. (2008). IEEE standards supporting cognitive radio and networks, dynamic spectrum access, and coexistence. *IEEE Communications Magazine*, *46*(7), 72–79.
6. 3GPP. (2009). Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Self-configuring and self-optimizing network (SON) use cases and solutions (release 9). TR 36.902, 3rd Generation Partnership Project (3GPP).
7. Kukliński, S., Skrocki, M., Rajewski, L., Llopis, J. M., & Wereszczyński, Z. (2012). GARSON: Management performance aware approach to autonomic and cognitive networks. In *IEEE MENS 2012. Anaheim, CA, USA*.
8. Univerself project. (2013). D2.4—UMF design release 3 http://www.univerself-project.eu/
9. Tsagkaris, K., Nguengang, G., Galani, A., Grida Ben Yahia, I., Ghader, M., Kaloxylos, A., et al. (2013). A survey of autonomic networking architectures: towards a unified management framework. *International Journal of Network Management*, *23*(6), 402–423.
10. Luoto, M., Rautio, T., Ojanperä, T., & Mäkelä, J. (2015). Distributed decision engine—An information management architecture for autonomous wireless networking. In *IM 2015, to appear*.
11. Mäkelä, J., Luoto, M., Sutinen, T., & Pentikousis, K. (2011). Distributed information service architecture for overlapping multiaccess networks. *Multimedia Tools and Applications*, *55*(2), 289–306.
12. IEEE. (2009). Media independent handover services. IEEE-Std 802.21.
13. Luckham, D. (2002). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Boston: Addison-Wesley.
14. Russel, S., & Norvig, P. (2003). *Artificial intelligence: A modern approach* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.
15. Feng, C. W., Huang, L. F., Ye, P. Z., Tang, Y, & Chao, H. C. (2014). A Q-learning-based heterogeneous wireless network selection algorithm. *Journal of Computers*, *24*, 80–88.
16. Mämmelä, O., & Mannersalo, P. (2014). Cognitive wireless access selection at client side: Performance study of a Q-learning approach. In *NOMS 2014*.
17. Zhao, Y. Q., Zhou, W. F., & Zhu, Q. (2012). Q-learning based heterogeneous network selection algorithm. *Recent Advances in Computer Science and Information Engineering*, *127*, 471–477.
18. Ahmed, T., Kyamakya, K., & Ludwig, M. (2006). A context-aware vertical handover decision algorithm for multimode mobile terminals and its performance. In *Proceedings of the IEEE/ACM Euro American conference on telematics and information systems (EATIS 2006)* (pp. 19–28). Santa Marta, Colombia. ISBN 958-8166-36-5.
19. Akshabi, S., Narayanaswamy, S., Begen, A.-C., & Dovrolis, C. (2012). An experimental evaluation of rate-adaptive video players over HTTP. *Signal Processing: Image Communication*, *27*(4), 271–287.
20. Istepanian, R. S. H., Philip, N., & Martini, M. (2009). Medical QoS provision based on reinforcement learning in ultrasound streaming over 3.5G wireless systems. *Selected Areas in Communications, IEEE Journal on*, *27*(4), 566–574.
21. Xing, M., Xiang, S., & Cai, L. (2014). A real-time adaptive algorithm for video streaming over multiple wireless access networks. *IEEE Journal on Selected Areas in Communications*, *32*(4), 795–805.
22. Evensen, K., et al. (2011). Using bandwidth aggregation to improve the performance of quality-adaptive streaming. *Signal Processing-Image Communication*. doi:10.1016/j.image.2011.10.007
23. Ojanperä, T., Luoto, M., Uitto, M., & Kokkoniemi-Tarkkanen, H. (2014). Hierarchical management architecture and testbed for mobile video service optimization. In *ICNC 2014* (pp. 999–1005).
24. COMMUNE project. (2012). D4.1—Specification of Knowledge-based Reasoning Algorithms http://projects.celtic-initiative.org/commune/
25. Eisler, M. (2006). XDR: External data representation standard. IETF Request for Comments: 4506.
26. Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.
27. Watkins, C. (1989). *Learning from delayed rewards*. Ph.D. thesis. England: Cambridge University.

28. Horsmanheimo, S., Eskelinen, J., & Kokkoniemi-Tarkkanen, H. (2010). NES—Network Expert System for heterogeneous networks. In *ICT2010* (pp. 680–685). Doha Qatar.
29. Seufert, M., Egger, S., Slanina, M., Zinner, T., Hoßfeld, T., & Tran-Gia, P. (2014). A survey on quality of experience of HTTP adaptive streaming. *IEEE Communications Surveys & Tutorials*, *17*(1), 469–492. doi:10.1109/COMST.2014.2360940.

Tiia Ojanperä is a senior scientist at VTT Technical Research Centre of Finland. She received her M.Sc. (Tech.) degree in Information Engineering from the University of Oulu, Finland, in 2004 and Ph.D. (Tech.) degree in Computer Science and Engineering from the same University in 2013. She has gained experience of several years in national and international research projects. The main focus in her research work is cognitive network management focusing in particular in the joint management of adaptive video streaming and terminal mobility in heterogeneous multi-access networks. Under these topics, she has published her work in international scientific forums.

Markus Luoto received his M.Sc. (Tech.) degree in Computer Science from the University of Oulu, Department of Electrical and Information Engineering, Finland, in 2008. He has been working at VTT Technical Research Centre of Finland since 2006 and his current research at VTT has focused on topics surrounding IP mobility including mobility triggering and management, multi-access, mobility protocols and wireless networks. His work also heavily includes prototyping and thus he has been actively involved in maintaining and developing network infrastructure of VTT's Converging Networks Laboratory.

Mikko Majanen received his M.Sc. degree in theoretical physics from the University of Oulu, Finland, in 2003. He is continuing his studies towards M.Sc. (Tech.) in Information Engineering in the same University. He has been working at VTT Technical Research Centre of Finland since 2000. He is focusing on projects related to network simulations. He has been involved in many national and international projects and he has led some customer projects as a project manager. He has published his work in international scientific forums and he has received two best paper awards (ICWMC 2007 and ENERGY 2012).

Petteri Mannersalo is a principal scientist at VTT Technical Research Centre of Finland. He received his M.Sc. (1993) in Mathematics and Ph.D. (2003) in teletraffic theory, both from Helsinki University of Technology. He has been contributing to many different topics related to analysis of communication networks. These include, for example, queueing theory, spatial models, traffic modelling and simulation techniques. Currently, his main interests focus on developing mechanisms and algorithms which enable autonomous management of networks and services.

Pekka T. Savolainen received his M.Sc. (Tech.) degree in Telecommunications from the Helsinki University of Technology in 2007. He has been working as a research scientist at VTT Technical Research Centre of Finland since 2007. His research work has included digital signal processing, simulations of communication systems and transmission methods, and hardware and software implementations of algorithms and decision-making components. His current interests are in machine learning and especially reinforcement learning in various applications including telecommunication networks and smart grids.