CLIENT-SERVER ARCHITECTURE AND ALGORITHMS FOR UBIQUITOUS SERVICE

M.Sc. Dissertation for Research Project

Submitted by Orit Yudilevich

Under Supervision of
Dr. Ronit Nossenson (Jerusalem College of Technology)
And Prof. Tami Tamir (IDC)

December, 2011
ABSTRACT
This work introduces novel client-server system architecture and algorithms for ubiquitous live video and VOD service support. The work included definition of technical requirements, development of system architecture and the relevant algorithms up to the level of high level system design.

For the first time, whole system architecture is presented, concerning variety of issues like QoS, personalization, privacy, simultaneous usage of a device by a number of users etc. Due to targeting to a full solution (and not focusing on one issue), many mutual influences between requirements were recognized and interworked in the model. As a result, user mobility, which is focal issue in ubiquitous service, is resolved "by the way" in the proposed architecture.

The main features of the system are: efficient usage of network resources, emphasis on user personalization and ease of implementation. In order to achieve efficient usage of network resources, the architecture is based on the "best k" algorithm. "Best k" algorithm provides high quality live-video transmission by using a small number of agents and contains methods for minimizing the usage of network resources.

An additional important advantage of the system is that it is network independent, and thus can use any RAT technology. Obviously, it can coexist with other ubiquitous service architectures.
# CONTENTS

Abstract ............................................................................................................................ 2

List of Figures .................................................................................................................... 4

1. Introduction .................................................................................................................. 5

2. System Requirements .................................................................................................. 6
   Definitions ...................................................................................................................... 6
   General Requirements ................................................................................................. 7
   Usability Requirements ............................................................................................... 8
   QoS requirements ........................................................................................................ 8
   Multiple User Requirements ....................................................................................... 9
   Privacy Requirement ................................................................................................... 9

3. System Specifications .................................................................................................. 9
   Service State Machine ................................................................................................. 10
   Agent State Machine .................................................................................................. 11
   QoS Specifications ...................................................................................................... 12
   User Interface Specifications ..................................................................................... 18
   Multiple User Specifications ...................................................................................... 19
   User Mobility ............................................................................................................. 21

4. Further Work and Conclusion .................................................................................... 21

5. References .................................................................................................................. 23
LIST OF FIGURES
Fig. 1. High level description of a client-server system for ubiquitous service.......................... 6
Fig. 2. User-Device-Client-Agent-Relationship........................................................................... 7
Fig. 3. The service state machine.............................................................................................. 10
Fig. 4. Agent State Machine...................................................................................................... 12
Fig. 5. Service State Machine and QoS management state machine for active service. .......... 13
Fig. 6. Swap agent by QoS procedure flow chart ........................................................................ 15
Fig. 7. Swap agent by overhead procedure Flow Chart ............................................................. 16
Fig. 8. The process of removing a user from a service.............................................................. 20
1. INTRODUCTION

A new generation of distributed services ranging from entertainment services such as live video streaming, video on demand and on-line gaming, to life-saving applications such as medical services and monitoring are being deployed in heterogeneous and ubiquitous environments. To be accepted by both users and network operators these ubiquitous services must be delivered continuously and provide adaptive and satisfactory Quality-of-Service with a minimum overhead of network resources. Providing ubiquitous services entails a number of complex issues, such as supporting the required QoS during a session, seamless handovers between different radio access technologies (RATs), supporting user mobility, etc.

In this work we introduce simple client-server architecture and algorithms for live video and Video On Demand (VOD) ubiquitous services. To achieve satisfactory continuous service with minimum overhead, collaboration and coordination between small number of agents use several communication methods including wireless or cellular connections.

The main features of the architecture are as follows:

1) Efficient usage of network resources complying with the required/preferred QoS.
2) User-driven architecture which enables easy personalization.
3) Ease of implementation.

Previous work on ubiquitous services has focused on middleware solutions (see, for example, [1, 2, 3]), which are much more complicated in terms of implementation and require the addition of new nodes and protocols to the network. These studies tend to address only one issue at a time, such as inter-RAT handover or QoS provisioning and do not supply full solutions. Furthermore, the requirements for ubiquitous systems have mutual influences that cannot be handled in a single dimension solution. However, some of the issues arising in middleware systems are easily solved in the client-server architecture described in this work.

As illustrated in Fig. 1, our novel architecture relies on both the user's communication devices (for example, smart phone, PDA or laptop) and the continuous service provider system; no changes or extensions are needed in the operator network. The system can completely handle a range of continuous service requirements: QoS provision, user mobility between RATs and between different communication devices, simple user interface and personalization, usage of the same device by a number of users (while protecting privacy) and so on. The description focuses on ubiquitous live video and VOD services, but the system can be easily extended to other services as well.

The architecture is based on the "best k” algorithm to ensure efficient use of network resources [9]. This algorithm provides high quality live-video transmission by using a small number of agents and proposes ways to minimize the usage of network resources. Experimental results show that by using the best-k algorithm, high quality video can be achieved with an overhead factor of 1.07 [9].
2. System Requirements

Providing ubiquitous service raises a number of issues that must be addressed, for example switching from one network to another with real-time synchronization and minimal user intervention. In this section we specify the technical requirements for a ubiquitous service system. First we define the terms used in this work. Next, we list the requirements according to their functionality: general requirements, usability requirements, connectivity requirements, multiple users’ requirements and privacy requirements.

Definitions

*Content provider* functions as the server side. Its responsibilities include user management, QoS management and content provision. *Service* is a video service such as live video or VOD (Video on Demand). *User* is a person who subscribes to the service. In order to receive the service the user has to register at least one device.

*Client* is the software that provides a special service on a user device. A user has one client on each device, per service. For example, a user who subscribes to 3 services via 4 devices has 12 clients. The client is responsible for communication between the user and the system, measuring and reporting QoS, agent management and combining the received data when necessary (see below).

*Agent* is the software that is responsible for receiving and transmitting data for the service via a specific communication interface/technology. For example, a device with cellular, WiFi and BT connections has one client and up to three agents per service and user (see Fig. 2).
Service has three states (per user): (a) "Not Active" (not initiated). In this state neither the system nor the clients keep any data regarding the service; (b) "Paused", where the user initiated the service and paused it, in this case no data transition takes place, but the system and the clients keep service session related information; (c) "Active". The content is provided, both the system and the client manage the data. The transition between the states and further information is described below.

Agent has three states: (a) In the "Not Active" state data is not transmitted between the system and the agent; (b) In the "Available" state the agent has established a connection to the server and they support a keep-alive signal exchange. The agent is ready to transmit data; (c) In the "Active" state the service data is transmitted via the agent. There may be more than one active agent per service. User states are similar to states of service.

Two or more users can use the same service via one device simultaneously (watching a movie together, for example). It this case the first user who activated the service is the primary user and the other users are referred to as secondary users. The primary user and secondary users together are referred to as the service users’ group on a device. The primary user manages the service users’ group (adding and removing users) that is using the service. This issue is explained further in this work.

**GENERAL REQUIREMENTS**

G1. The system supports ubiquitous service for live video transmission and video on demand (VOD) applications.

G2. The system attempts to provide QoS as close as possible to the preferred quality (see requirement Q1 below), with minimal user intervention and with minimal overhead for network resources.

G3. Scalability requirements:
- The parameters A, B, C, D and E, to be used below, can be extended by simple hardware extension.
- The system supports up to $A$ potential agents per user.
- The system supports up to $B$ activations of service per minute.
- The system supports up to $C$ simultaneous active services.
- A user may have up to $D$ active services and up to $E$ paused services simultaneously.

**USABILITY REQUIREMENTS**

U1. Service is supplied via one client at a specific time.
U2. The system provides a convenient user interface.
U3. The user needs to configure a set of agents for each service for each client (device).
U4. For a service, at least one device (client) and one agent must be registered.
U5. The user can alter its set of agents at any time using a convenient interface.
U6. For each service specified in requirement G1 the system supports the following operations:
   a. Start
   b. Stop
   c. Pause
   d. Resume

U7. The "Start" operation is used for service activation the first time as well as for service re-activation after a "Stop" action. It can be used for inactive service only.
U8. The operation "Stop" is used for termination of the active or paused service.
U9. The operation "Pause" is used for temporary halts of an active service, up to a predefined timeout. The timeout can be interrupted by user operation ("Resume"). Otherwise the service is terminated.
U10. The "Resume" operation is used to continue the service after the "Pause" operation, depending on the type of service, assuming that the timeout has not expired.
U11. The outcome of the "Resume" operation for VOD application is continuation of the video transmission from the point where it was paused. For the Live Video application it corresponds to resumption of the on-line video at the current time.
U12. For an active service, at least one agent is active.
U13. Changes of agents, without a change in client (switching, adding and subtract agents) are done transparently without the user's intervention.
U14. A paused service can be resumed from any subscribed client (device continuously (see requirement U9).
U15. There can be only one active service per device.
U16. A user can have several active services, on several devices.

New service activation on a device that already has another service active on it, is subject to the approval of the primary user of the active service and is equivalent to pausing/termination of the previous service.

**QoS REQUIREMENTS**

Q1. The user can define *Preferred QoS parameters* and *Required QoS parameters* per service. In addition, the system has default values for these parameters per service.
Q2. The system aims to provide the user with QoS as close as possible to the preferred QoS, with minimal user intervention. If this is impossible, the system aims to provide QoS above the required QoS. If the QoS falls below the required level, the user is informed and the service is terminated.
Q3. The system provides continuous service for the user as long as there is at least one active client and active agent capable of providing QoS above the required level.

Q4. The QoS parameters are defined for each service separately, including the following characteristics as a minimum: bandwidth, delay and jitter (the deviation from the network mean latency).

Q5. When the QoS parameters are higher than the required level, but below the preferred thresholds, the system attempts to improve the service in the following ways: (1) if the preferred QoS can be achieved using the current client (by changing agent/s), the change is performed transparently; (2) if the preferred QoS can be achieved only by another client (device), the transition can be performed, subject to the user’s approval; (3) if the preferred QoS cannot be achieved using any other client (device), the system provides the best QoS possible via the current client (device).

Q6. If the QoS is below the required threshold, the system tries to improve the QoS via the current client. If this is impossible, the user is advised to move to another device. If the required QoS cannot be met the service is terminated.

Q7. For each user and active service, the system maintains a set of potential agents as a function of QoS parameters, environmental changes, user preferences, and agent availability.

**Multiple User Requirements**

M1. An active service has one primary user.

M2. There can be several secondary users for an active service.

M3. A user can join a service on a specific device which is managed by a different primary user, subject to both users’ approval.

M4. A secondary user can disjoin a service; this action is equivalent to pausing or stopping the service to the specific user.

A primary user can be replaced by another user, subject to both users’ approval. The previous primary user becomes a secondary user in this case.

**Privacy Requirement**

P1. A user cannot access information on services that another user is subscribed to, even if they co-exist on the same device (see fig. 2 for example), unless the user is a primary user who is aware of the secondary users of the same service.

**3. System Specifications**

In this section we elaborate on the technical problems and provide specifications and algorithms for the system. The listed specifications provide a feasible solution for the pre-defined requirements. This section is divided into the following sub-sections: service state machine, agent state machine, QoS specification, user interface specification, multiple user specification and user mobility.
According to the requirements, each service can be in one of three states per user: Not Active, Active or Paused. Fig. 3 depicts the transitions between the states, showing all the valid transitions, their triggers and actions; the initial state is "not active service".

1) **From "Not Active Service" to "Active Service"(1):** This transition occurs when the "Start" command is initiated by the user. The actions that take place in this case are: (i) the system establishes a connection with the agent that sent the "Start" command before it starts to transmit the content; (ii) A handshake procedure is performed with each user's agents, and a list of available agents is generated. The handshake procedure and the management of the available agent list are described in the agent state machine below.

2) **From "Active Service" to "Not Active Service"(2):** This transition can occur in two cases: upon a user "Stop" command or if the QoS falls below the required level (see requirements Q2, Q6). In these cases the data flow to/from the client is terminated, a termination message is sent to all available agents (through the clients), so they can move to the "not active" state, all service-session related data are dropped on both the server and client sides.

3) **From "Paused Service" to Not Active Service"(3):** A service can go from "Paused" to "Not Active" in two cases: by a user "Stop" command or by a pause timeout expiration, see requirement U9. In these cases, the system sends a termination message to all clients to turn the available agents to "not active". Service-session related information is dropped from both the server and client sides.

4) **From "Active Service" to "Paused Service"(4):** This transition is triggered solely by user command. The user can implement this transition in several ways: by sending pause command, activation of another service on the same device, approval of client swapping proposed by the system and switching from primary user to secondary user in the multiple user service mode (see requirements U9, U17, M3). In these cases all the service session data are saved in both the server and the client. Service paused messages are sent to the active agents, so they can go into the "available" state. Data transmission is stopped. On the server side, additional data are stored, such as last active client, pointer to the last

**Figure 3. The service state machine. Valid transitions of service states are depicted.**
transmitted I frame, last packet sequence number, file offset (the location in the
movie for VOD service), and last decoding format in use. In addition, the timer
for maximum paused time is activated.

5) From “Paused Service” to “Active Service” (5): This can occur only upon an explicit
"Resume" user command. In case the service is resumed on the same client it
was paused on, all the data exist on both the client and the server sides and the
service is simply resumed. Otherwise, if the service is resumed on a different
client (device) a format adaptation is performed if needed. The last I frame is
sent to the client together with the following P frames and a specific notification
for the video player film offset. This enables VOD service to resume from the
same point it was paused on the previous device. For live video service, the
service is resumed according to the current time.

6) From “Active Service” to “Active Service”(6): Occurs on “resume” command for
active service. If the "resume" command is initiated on the active client, the
command is discarded, otherwise the command is equivalent to pausing the
service on a current client and resuming it on a new one.

Other state transitions are invalid and should not be made available to the user.

AGENT STATE MACHINE

Agent management is a key issue in this architecture. In this sub-section we
introduce the agent state machine.

As mentioned earlier, for active service, one or more agents can be used for service
supply (they are the active agents), while the other agents for this service are in the
"available" state. Upon service activation the system sends activation messages to all
user clients (for the current service). The clients instruct their agents to move to the
"available state". The clients and agents must respond to the activation message. This
process is referred to as the "handshake procedure". The outcome of this "handshake
procedure" is a list of all available agents for the current service (the use of the list is
described in QoS Specification sub-section above). In order to keep the available agents
list updated, all agents must send (by client) keep-alive messages. When a client
recognizes that agent/s becomes available after unavailability (for example device turn
on) a notification is sent to the system. If the service is active, an activation command to
the agent/s is sent. Similarly, when agents become unavailable for active or paused
service, they should inform the system if possible. Once an agent does not send a keep-
alive message for a specific period of time it is removed from the available agents list.
Agent that has completed the handshake procedure is regarded as in available or active
state until service termination.

As described in the user interface sub-section, the system has to know if there is
another active service (for this or another user). This information is piggy-backed on
keep-alive messages. Below are the detailed agent state transitions (see Fig. 4 below).
The initial state is "agent not active".
1) From "Not Active" to "Active". This transition occurs only upon service activation on the specific device.

2) From "Active" to "Not Active". This transition takes place when the agent was one of the agents that provided the service and the service is terminated due to a user's command or due poor QoS (see sub-section QoS specification above).

3) From "Active" to "Available", this transition occurs in the case of agents/client switching for QoS reasons (see QoS specification sub-section above) or a service state change from active to paused (see service state machine sub-section above).

4) From "Available" to "Active", this can occur if the agent is selected by the system for service transmission for QoS reasons or when the service state moves from "Paused" to "Active" by command from this agent.

5) From "Available" to "Not Active", this transition takes place for available agents of a service when the service state is changed to "Not Active" (from "Active" or "Paused" states).

6) From "Not Active" to "Available", this transition takes place for all agents upon service activation command.

In "Not Active" state the agents have no connection to the system. In the "Available" state the agents are connected to the system and keep-alive messages are exchanged as described above. An agent is "Active" if it is one of the agents that supply the service.

**QoS Specifications**

This sub-section provides details on the "Active Service" state presented in the previous sub-section by discussing the QoS provision. It presents the high-level architecture and algorithms designed to address the QoS requirements. Specifically, it considers requirements G2, U12, U13, Q2, Q3, Q5, Q6 and Q7 above.

As mentioned above, the only components that are involved in the ubiquitous service are the ubiquitous content provider and the user devices (handsets, computers and etc.).

The essential server functions are:
- Ensuring an acceptable QoS level via agent management.
- Providing maximum transparency to the user.
- Ensuring efficient usage and minimum overhead of the network resources.

The essential client functions are:
- Providing the user interface to the system.
• Monitoring the QoS parameters for active services and informing the server if needed (for example, when a QoS parameter crosses threshold, when QoS parameter falls below the required level).
• Combining and synchronizing the data if received by more than one agent, using the algorithm presented in [9] or a similar algorithm.

The following information is stored on the server per active user of a specific service:
• The current QoS parameters.
• The current active client and its set of active agents.
• A list of all available clients with their available agents.
• Session-associated information, for example compression information.
• List of clients and agents used in the last G2 minutes. (see user-interface subsection below for details).
• List of forbidden client transitions. For example, the user can prevent switching from its home computer to the one at work.

During Active service, the server runs the state machine as presented in Fig. 5 (the initial state is "service not active", similarly to Fig. 3). In state A the user receives its preferred QoS by one agent; thus the system does not have to improve its QoS or to reduce network resource overhead associated with it.

State B is characterized by the QoS between the preferred and required levels, thus the service can be continued along with system efforts to improve the QoS according to requirements G2 and Q5.

In state C the QoS is equal or higher than the preferred QoS, but is supplied by more than one agent. In this case the system should try to reduce network resource overhead, by attempting to supply the preferred QoS using one agent if possible (requirement G2).

![Figure 5. Service State Machine and QoS management state machine for active service.](image-url)
In state D the QoS is below the required threshold. In this case the system attempts to improve the QoS; if this attempt fails the service is terminated (see requirement Q6.)

The client is responsible for assessing and monitoring the QoS. The server is informed when a meaningful change in QoS (transition from/to preferred QoS and required QoS) occurs.

A detailed explanation of the states and the operations of these states are provided below.

1) State A (QoS ≥ Preferred QoS): As mentioned above this state does not require any system action.

2) State B (Required QoS<QoS< Preferred QoS): In this state the QoS is below the preferred QoS and above required QoS. Thus, the system should attempt to improve the QoS. The requirements define the following priorities: QoS, minimum user intervention and network resources (see requirements G2, U13, Q5, Q6); hence, in state B, the following procedure is performed periodically:

StateBProc(PreferredQoS, CurrentClient, AllAvailableClients):
Begin
  BoolIsPreferredQoSPossible = false;
  BoolIsPreferredQoSReached = false;
  IsPreferredQoSPossible = BestQoS(PreferredQoS, CurrentDevice, AllAvailableClients);
  If (IsPreferredQoSPossible) Then
    BoolIsPreferredQoSReached = SwapAgentbyQoS (PreferredQoS, QoSList);
  End
  If (Not IsPreferredQoSReached) Then
    ImproveCurrentClientQoS (CurrentQoS, CurrentClient);
  End
End

The function \textit{BestQoS} is responsible for generating the list of respective agents and possible QoS by staging a competition between the agents (for details regarding agent competition see [9]). The QoS list includes: (i) Best QoS that can be reached by one agent of the current client (device) (\textit{BestQoSSingleAgentCurrentClient}) and corresponding agent; (ii) Best QoS that can be reached by two or more agents on the current client (device) (\textit{BestQoSMultiAgentCurrentClient}) and corresponding agents; (iii) Similar data about other available clients and agents (iv) Best possible QoS for each client achieved by one or more agents. The inputs to the function are: requested QoS threshold, current device, full list of devices and agents.

The pseudo code of the function is as follows:

\begin{verbatim}
BestQoS(QoSThreshold, CurrentClient, AllAvailableClients):
Begin
  Perform Competition on Current Client’s Agents;
  Set BestQoSSingleAgentCurrentClient, Agent;
  Set BestQoSMultiAgentCurrentClient, Agents;
  If ((BestQoSSingleAgentCurrentClient ≥QoSThreshold ) Or
    (BestQoSMultiAgentCurrentClient ≥QoSThreshold))
    Then return true;
\end{verbatim}
/* NeededQoS can’t be reached by current client, rest of the clients are checked */
Perform Competition on Available Clients;
SetBestQoS SingleAgentOtherClient, AgentList;
Set BestQoS MultiAgentOtherClient, AgentMatrix;
maxQoS = max(BestQoS SingleAgentOtherClient, BestQoS MultiAgentOtherClient);
return (maxQoS ≥ QoSThreshold)
End

If the current active agent set cannot meet the QoS threshold, a competition is staged between the active client's agents. If the current client cannot meet the QoS threshold, then, a competition is staged between the clients. The competitions take place in this order (and not competition over all clients in the first place) to save network resources (meeting requirement G2). The resulting list of QoS is used by other procedures as well.

In state B, when agent/s that supply the required QoS is/are found, the system attempts to swap to this/these agent/s. This is performed by the SwapAgentByQoS() procedure. This procedure flow chart is presented in Fig. 6. This procedure scans the possible agents according to a pre-defined order (requirements G2, U13, Q5, Q6). The first choice is one agent on the current client, then, multiple agents on the current clients, finally single and multiple agents of other clients.

SwapAgents() replaces an agent or agent group by another agent or agent group on the same client without loss seamlessly to the user.

The procedure ProposeSwapDevice() displays a list of possible clients to the user to achieve acceptable QoS. It is described in the user interface specifications section below.

The procedure SwapClient() should be performed sparingly to meet requirements G2 and Q2 to minimize user intervention. Thus this operation should be performed only if there is no other way to achieve the preferred QoS, and with a sufficient period between swaps.

The system should introduce the user to the full list of other clients that can provide the preferred QoS. Client switching is always subject to user approval.

In addition, the system should maintain a list of clients whose transitions are forbidden (see user interface specification for details), to avoid undesirable proposals to the user (requirement G2, Q5).
The *SwapAgentbyQoS()* procedure updates the QoS state machine, as required. If the preferred QoS cannot be reached the system should try to improve the QoS on the current client (requirement Q5), to the best possible QoS. This is performed by the *ImproveCurrentClientQoS()* procedure. It uses the current QoS and current client as inputs. This procedure does not affect the QoS state machine, since after its termination the QoS level is still between the preferred and the required thresholds.

```plaintext
ImproveCurrentClientQoS (CurrentQoS, CurrentClient)
Begin
    MaxQoS = max(BestQoSSingleAgentCurrentClient, BestQoSMultiAgentCurrentClient);
    If ((MaxQoS>CurrentQoS) And (MaxQoS==BestQoSSingleAgentCurrentClient))
        Then SwapAgents(CurrentAgents, NewAgent);
    Else if (MaxQoS>CurrentQoS)
        Then SwapAgents(CurrentAgents, NewAgents[]);
End
```

Figure 6. Swap agent by QoS procedure flow chart
3) State C (QoS ≥ Preferred QoS, achieved by two or more clients). In this state the preferred QoS is accepted, but it is achieved by several agents. The system should try to reduce the number of active agents. This is done by the `SwapAgentbyOverhead()` procedure that runs periodically. Fig. 7 presents the flow chart for this procedure. The procedure updates the QoS state machine accordingly.

4) State D (QoS < Required QoS): In this state the required QoS cannot be supplied and according to requirement Q2 the service should be terminated. The following procedure runs once; if it fails the service is stopped (Moving to state “Not Active Service” in the state machine presented in Fig. 3 above).

```
STATEDPROC():
Begin
BoolIsPreferredQoS, IsRequiredQoS = false;
IsPreferredQoS = SwapAgentbyQoS (PreferredQos);
If (PreferredQos) Then
    Move to State A;
    Return;
IsRequiredQoS = SwapAgentbyQoS(RequiredQos);
If (!IsRequiredQoS) Then
    Notify user of service termination;
    Exit Active Service State Machine;
    Service state = Not Active;
    Terminate service;
Else
    ImproveQoSCurrentClient;
    Move to State B;
End
```

**Figure 7. Swap agent by overhead procedure Flow Chart**
The system attempts to achieve the preferred QoS. If the preferred QoS is reached, the QoS state machine is updated and the procedure terminates. Otherwise, the system attempts to supply the required QoS. Once the required QoS is achieved, an attempt to improve the QoS is performed (preferred QoS cannot be achieved), to meet requirement Q6. If QoS is below the required level, the service is terminated.

**User Interface Specifications**

User interface has to meet requirements G3, U3, U4, U5, U6, Q1 and P1.

In order to meet privacy requirements (P1) the interface to a service should be a remote web page and it should be password protected, for example.

The actions that can be performed by the user interface are:

- Join the service.
- Client and Agent Management.
- Activation, stopping, pausing and resumption of a service.
- Multiple User Service Management.
- Disjoin the Service.

These actions are described below.

1) **Join the Service**: First, the user has to register and to accept a username and password (requirement P1) to access the system. The next step is setting the preferred and required QoS (requirement Q1). The user can choose the default values. The use of these values is described in the QoS specification sub-section above. Then, the user should register the devices; this process is described below.

2) **Client and Agent Management**: Client and agent management is divided into three sub-processes: client/agent addition, client/agent subtraction and client transition management. As a guideline the user does not have to address agents. The user chooses to alter the client setting; the system requests the data (list of connection interfaces) from the client and presents the options to the user. These procedures can be implemented at any time according to requirement U5.

   a. **Client/Agent Addition**: When subscribing a device or agent to a service, a user can choose to connect from the current device ("Add this device") or to identify the subject device by a unique identification, an IP address for instance. After the device is chosen the system queries the device about possible agents (connection interfaces). All possible agents are presented to the user and the user chooses which agents to install. In case of agent addition the user should choose "modify client settings" and the system will return all the options. After the user confirms the choice the profile at the client side is updated. If the number of potential agents exceeds the A parameter from requirement G3, the system displays the message to the user and client/agent addition is aborted.

   b. **Client/Agent subtraction**: The procedure can be performed from any device. In agent subtraction, as mentioned above, user chooses "modify client settings" and the system shows all agents. In the case of client subtraction the system presents the user list of all of its clients (devices) and the user chooses the client/s to subtract. The system does not allow subtraction of the last client/agent (requirement U4). After the user confirms the choice the profile at both server and client sides is updated.
3) **CLIENT TRANSITIONS MANAGEMENT:** In the QoS specifications sub-section, it was noted that in case of insufficient QoS or inefficient use of network resources the system can suggest switching clients to the user. As mentioned earlier the system should try to minimize the use of this option, to meet the requirement for minimal user intervention (G2). For this reason there should be a minimum period of G₁ minutes between two successive proposals to switch devices. Additionally, the system should store recently (for G₃ minutes) used agents and clients, in order to avoid "ping-pong" transitions between clients. For the user’s convenience the system should avoid proposing invalid transitions. For this reason the system stores a list of forbidden client transitions, per user. The user should manage this list. This list can be updated in two ways: by the user interface or when the system displays the list of optional clients for transition, the user can assign "Do not propose this transition again" to one or more clients. The system also should not propose a switch to a device in active service (of any user) to comply with requirement U15. In summary, client transition can be proposed to a user if: this is the first proposal for G₁ minutes and there exists a potential device for transition that has not been used in the last G₂ by this user and the service, does not appear on the forbidden transitions list, does not run an ubiquitous service to any user and meets the QoS threshold. Upon the user’s approval of the transition client, a swapping procedure is performed. Client swapping is similar to pausing and resuming the service on a new client; the only difference is that the “pausing” is triggered by the user’s approval of the client switching and not by the pause command. Pausing and resuming service is described in the service state machine sub-section above. Further work could be done to elaborate the list of proposed clients for transition, by studying users' preferred transitions, for example.

4) **ACTIVATION, STOPPING, PAUSING AND RESUMPTION OF A SERVICE.** These actions are described in the service state machine and the QoS sub-sections. For privacy (requirement P1) the activation and resumption of a service should be password protected operations.

5) **MULTIPLE USER SERVICE MANAGEMENT:** The interface is described in the multiple user service management sub-section below.

6) **DISJOIN THE SERVICE:** The user should choose to unsubscribe to the service. After the user confirms the choice, all associated session information on both the client and server sides are removed.

**MULTIPLE USER SPECIFICATIONS**

One aspect of ubiquitous service is that several users may start to consume a service together (watching a VOD movie for example) and then want to continue to consume it separately at a different time and place. The requirements for multiple users (M1 – M5) specify this case. The appropriate specifications are introduced below.

In order to support the multiple user mode the system and the client need to support the following actions:

- Add user to service user group.
- Remove user from service user group.
- Primary user substitution.

All of the above only apply to active service.

1) **ADD USER TO SERVICE USER GROUP:** To add a user to a service group, the primary user should choose the "Add user to the current service" option and specify the user. The
system asks for the new user’s confirmation, according to requirement M2, by requesting a password. Once the subject user is approved to join the service, the system performs the following steps: the user is added to the service’s user group, a connection is established with all the users’ clients to move agents to the "available" state and to generate a list of available agents. The service state for the new user is "paused". Note that the user must be subscribed to the service, but the device can only be subscribed by the primary user. For instance, user Bob subscribes devices A, B and C to the service. User Alice subscribes devices B, D and E. Alice starts to watch live video using device D. Bob joins her (although he did not subscribe device D to the service). After a while Bob chooses to continue watching the video using device A, so he disjoins Alice and resumes the service on device A.

2) **REMOVE USER FROM SERVICE GROUP:** Fig. 8. below depicts the steps for removing a user from the service user group. Confirmation by the subject user is performed as above. If the primary user wants to leave the service a new primary user should be chosen according to requirement M5. The primary user substitution process is described in the next paragraph. Then the departing user should chose to stop or to pause the service (requirement M3). Stopping the service is done in a similar way as moving from the "paused" state to "not active" state (see service state transitions for details). If the user wants to pause the service, all available agents stay available. The service state of the user is paused with the last client field empty.

3) **PRIMARY USER SUBSTITUTION:** Whenever the service's primary user needs to be substituted (one case was introduced in the previous paragraph) the following steps should be performed: (i) upon user request to change primary user, the system should generate a list of potential primary users. The new primary user can be a user who is in the service's user group and subscribed the current device to the service. If there is no such user, the primary user change cannot be done. (ii) The list of potential primary users is displayed to the current primary user and he/she should choose the new primary user. (iii) The last step is to stage the competition (details in [9]) between agents of the new primary user on the current client, and to choose active agent/s for the service. The previous primary user becomes the secondary user and stays in the service's user group.
User mobility is a focal issue in ubiquitous service; see [1] for example. In this subsection we show that the mobility issue is resolved in our system.

User mobility can have two negative effects: the user needs to switch devices and/or the QoS degrades due to coverage or load issues. If the user needs to switch devices this should be done by pausing the service on the old client and resuming it on the new one (details in service states machine and user interface sub-sections). The case of QoS degradation is discussed in the QoS specification sub-section above.

Thus our specifications resolve the user mobility issue without having to take any location associated actions, by enabling a high level of customization and addressing QoS as a general issue that is not related solely to mobility.

4. Further Work and Conclusion

This work describes a novel approach to ubiquitous service - client/server architecture. The system incorporates detailed requirements, specifications and algorithms. We address all known issues related to ubiquitous service: QoS management, efficient usage of network resources, limited overhead, simultaneously...
usage of a device by a number of users, user mobility and user interface. Previous works propose middleware solutions with wide influence on network architecture and functionality. These studies tend to address only one issue at a time and do not supply full solutions. Furthermore, previous works did not handle the issue of simultaneously usage of a device by a number of users. An additional advantage of our system is that it is network independent, and thus can use any RAT technology. Obviously, it can coexist with other ubiquitous service architectures.

Although it only discusses video on demand and live video services, the system can be extended to other services as well. The system is general, so the extension to other services is simple. Further customization can also be made, for example, prioritization of user clients and agents and/or statistically studying favorite client transitions as mentioned in user interface specifications.
5. REFERENCES


ארتفاعل chóng מגדיל ורגולאורהים לשירות ע"י פלסימ פתרונות

עלומר שירート הספלידים יздравו ו밀וי לשירות לאפליקציות חזותיותếmות בשירות
בובלות תכונות הזרימה ותחליפיות של המשקפיות במגזר ושירותיו ישראליות במכון פלוספרנוט
שנתה התוכנה זו מתאימה מתאימים ומגזרות שונים ידועים עם שירותים שונים
וכל благодар, כולם מעורבים במגע עם משפחת קול לטרו. על VOD
בכלouver תכל הצורה של איל חרדי ל넷ות בسرعة
במעורר מדורי חברה בחברות המשקפיות וניהון בשירות הנכונה לשירות הבכירה.

הHarmonics המרחפת ח artisans הממעטו [4] best k למקים שירוט "ע"י פלסימ סוניה. האלאורטרם
Harmonics המרחפת מתחברת אל- FOURIM / WiFi BT סולאריא / WiFi (ק领导者)
מסקן שירוט להתקנת שירותים מיוחדים (QoS) "י"ה בחירה ההכנה של אפקטי התוביר (כנון)
ובד), תחתי ומכבישי בד-מקסיקו (ק领导者) השיאני.
לבובות קורמת כי קיימת מספק הצעה האלאורטרם הבכירה לשירות רציע. כולם העבדות
ה"ג"ל בכנות חרות לע בקっていくי מתמנים חותם בنظم קורים ואתבוצא, כמו היכל מספק מכבייה המופיעה שלמד. בפשח, והעבדות הקורמות מיתיות
וליאן מספקים כנף המוחות עב/Middleware או והמקסיקו handoff לשירות QR ו/א והמגזר QR ו/א بالمגזר QR descripvers ייצר אל מיצות פתרון שלמד. בקורה זה לפי שיטת
ישל, לא הם שילוב עד לשיגור התמונות המוחות המוחות בשירה. הככוב עד זה היראותון מיתיות
וליאן על מספקים שלמתוגים ב-גרנד מכבייה מוסר.

לובודד מג在玩家中 והתראות הנבויותזכים לשירות שירוט רציע שלאמר בהמדאת פטיס
מכסה את כל המאשיות הנמשכים למגזר שירוט רציע: QoS, מספקים מתאימים, שיתופי, פתרונים ומגזר שירוט
למסקן קורמות פאוז ולכלי פלט. לשירותית המוחות מпущים את מבצעים הטרופי בנושה של שירת שירוט
פורנוליזציה.
המכון הבינתחומי, הרצליה
בית ספר אפי ארצי למדעי המחשב

ארכיטקטורות מערכות وأنלגוריתמים לשרות רציף
על"י סוכנים מחובמים

M.Sc. עבדה גמר להרא

מגישה: אורית יודילביץ'

מנחوت:
ד"ר רונית נסנטן (בית הספר הנביה לטכנולוגיה ב一如既יו)
פורפ' תמימי טמרי (המכון הבינתחומי הרצליה)

דצמבר 2011