BPEL Implementation of QoS-based Management in Multi-modal Mobile Communications

[1]Florin SANDU, [2]Dan Nicolae ROBU, [3]Cosmin COSTACHE
[1]"Transilvania" University of Brasov Bd Eroilor nr. 29A RO-500036 Brasov, [2,3]SIEMENS Program & System Engineering Bd M.Kogalniceanu nr 21, bl.C6 RO-500090 Brasov,
[1]sandu@unitbv.ro, [2]dan.robu@siemens.com, [3]cosmin.costache@siemens.com

***Abstract* – The authors consider the concept of "partial handover" of connections in multi-modal mobile packet communications. This is based on the idea of a partial transfer - not a total one, e.g. 10%-90% - of the connection. Maintaining at a minimum level the channel with reduced QoS (Quality of Service), with drastically reduced energy consumption, has also the advantage of keeping a signaling path - practically "on common channel". It is also useful to simply restore the preponderance of a channel if its QoS increases significantly, without the complications of restarting the radio link, even if this could be internally triggered, at the mobile terminal level, through the other path which is still active. The proof-of-concept is oriented on a 3G/WLAN multi-modal communication, with a binary decision, based on the QoS calculus - as double weighted sum of performance parameters - on a path and its comparison with a threshold value or with the "cost function" of the QoS on the other path.**

***Index Terms* — Communication Systems, Communication Standards, Communication System Signaling, Computer networks, Protocols**

I. INTRODUCTION

According to the standards of ITU (International Telecommunications Union") for TMN („Telecom Management Networks") [1] the mobile communication systems include central servers for PDC – „Performance Data Collection". PDC functionality can provide permanent measured values (and some statistics) on performance „benchmarks" that are specific to QoS. In order to collect the traffic values, PDC can be associated with the mechanism of "charging" – real time recording of all useful information – counters (for time, for volume), jitter, RTT (Round-Trip-Time), bit rate etc, included in „charging tickets".

In mobile data communications, „PDP (Packet Data Protocol) Context" can be taken, through the SS7 Signaling System, directly from the mobile data terminal, in the Context Activation phase. This set of parameters can also be used for the conditioning of QoS calculus [2].

II. THE DOUBLE-WEIGHTED SUM OF QOS

There were considered two types of criteria sets, general enough to be common for different wireless networks (e.g. 3G, Wi-Fi etc) which can operate in an optimized multi-modal communication.

*Subscriber Profile***,** which may contain a primary set of conventional weights - for example, on a scale from "student" to "manager": at the "student" level the profile will be oriented on low cost while at the "manager" level the resource allocation is more important (guaranteed bandwidth, low delays etc.).

In future developments, some of these weights could be configured by the individual subscriber himself/herself, who can transmit to the server a series of their profile modifications.

Weighted terms associated with the "Application" level (from the OSI - „Open System Interconnect" - stack). For example, they can be taken into consideration, as reference, the four *traffic classes* defined by the 3G standards [3]:

CONVCL – "Conversational Class: Real time applications. The most well known use of the Conversational Class is telephony speech. But with internet and multimedia, a number of new applications will require this scheme, e.g. voice-over-IP, video- conferencing tools etc."

STREAMCL – "Streaming Class: Services like real-time video and/or audio performance which are one-way transports and very sensitive to time variation (JITTER) between information entities" (packages).

INTACTCL – "Interactive Class: Non-real time applications. The Interactive Class is used by applications expecting messages" (responses) "within a certain time."

BACKGRCL – "Background Class: Applications in the background, in which the receiver is not expecting the data within a certain time."

Apparently all inputs (in the calculus procedure) are at the same importance level but, throughout the computation, certain conditions should be imposed by the application. For example:

At the "conversational" level, for both uplink and

downlink, the jitter will be taken in consideration with a weighting factor different than zero, while, in the rest of the classes, it can be considered zero.

For the "background" or „conversational" classes, the most important weighted factor is the cost, not the bit rate. The aim is to obtain a lower, but cheaper transfer rate (usually Wi-Fi is free compared to 3G).

The QoS calculus will consider different costs for the "uplink" and "downlink" effective traffic and special costs for resource allocation (bandwidth etc) and assure a minimum delay, jitter and RTT for the connection.

The "delay" factor is very important and it will be considered separately even if it could be used directly in the calculus of the quality factor for both „uplink" and „downlink".

The values from the network can be treated separately than the ones from the mobile terminal (mobile computation node, „smart phone", „communicator" etc).

The implemented formula (QoS calculus as double weighted sum):

$$QoS = \sum_{k=1}^{N} w_{profile,\ k} \cdot w_{class,\ k} \cdot p_{normalized,\ k}$$

Normalizing the parameter p_k is done at the possible *technical* optimum (if we were to consider the optimum from the CONVERSATIONAL *class*, for instance, or the optimum from the PREMIUM *profile*, an unwanted redundancy should occur). It could be also considered the *statistic* part of PDC in order to normalize to common, "realistic" values.

Indeed, if TDELAY – "Total Delay", is considered referred to the "maximum" of 4000 ms, more rarely found (only on low quality transmissions), using this too big numerator, the normalizing fraction of TDELAY becomes overwhelming and decreases the significance, "visibility", relevance and representation of the other normalized parameters.

This *optimum* will be either at the nominator or denominator depending on how the un-normalized value should ideally be maximum or minimum.

For example, with N=4 measured parameters - GBR (Guaranteed Bit Rate), TDELAY (Total Delay), JITTER and ER (Error Rate) - and 4 profiles (Premium, Voice, Normal, and Basic), here it is, a possible set of weights and normalizations:

$w_{Premium,\ 1} = 1,\ w_{Premium,\ 2} = 1,\ w_{Premium,\ 3} = 1,\ w_{Premium,\ 4} = 1$

$w_{Voice,\ 1} = 0.75,\ w_{Voice,\ 2} = 1,\ w_{Voice,\ 3} = 1,\ w_{Voice,\ 4} = 0.5$

$w_{Normal,\ 1} = 1, w_{Normal,\ 2} = 0.5, w_{Normal,\ 3} = 0.5, w_{Normal,\ 4} = 0.75$

$w_{Basic,\ 1} = 1, w_{Basic,\ 2} = 0.25, w_{Basic,\ 3} = 0.25, w_{Basic,\ 4} = 0.5$

k=1 $p_{normalized,\ 1} = GBR$ / 8640 kbps

$w_{STREAMCL,\ 1} = 1,\ w_{CONVCL,\ 1} = 1$,

$w_{INTACTCL,\ 1} = 0.5$, $w_{BACKGRCL,\ 1} = 0.25$,

k=2 $p_{normalized,\ 2}$ = 4000 ms / *TDELAY*

$w_{CONVCL,\ 2} = 1$, $w_{STREAMCL,\ 2} = 0.75$,

$w_{INTACTCL,\ 2} = 0.5$, $w_{BACKGRCL,\ 2} = 0.25$,

the DELAY is unidirectional (only for UPLINK or only for DOWNLINK) ; RTT (Round-Trip-Time) covers both directions

k=3 $p_{normalized,\ 3}$ = 100 ms / *JITTER*

$w_{STREAMCL,\ 3} = 1$, $w_{CONVCL,\ 3} = 1$,

$w_{INTACTCL,\ 3} = 0.25$, $w_{BACKGRCL,\ 3} = 0.25$,

k=4 $p_{normalized,\ 4}$ = 25% / *ER* (Error Rate)

$w_{STREAMCL,\ 4} = 0.5$, $w_{CONVCL,\ 4} = 0.5$,

$w_{INTACTCL,\ 4} = 1$, $w_{BACKGRCL,\ 4} = 0.5$,

QoS_{3G} and $QoS_{Wi\text{-}Fi}$ are calculated, as well as their ratio, and the partial handover from Wi-Fi to 3G is decided if QoS_{3G} / $QoS_{Wi\text{-}Fi}$ is over 1.1 ($QoS_{Wi\text{-}Fi}$ / QoS_{3G} drops below 0.9).

The 10% conventional value is chosen for the handover tolerance (a "hysteresis" interval) – with the purpose to grant an oscillation-free handover (a higher frequency handover can consume too many resources).

III. SOFTWARE TECHNOLOGIES EMPLOYED

Portability of the developed services was a main goal - the developed solution has to be portable across different operating systems and platforms. *SOAP* (Simple Object Access Protocol) - a protocol based on *XML* (Extended Markup-Language) [4] - was used to manage the change of information between computing nodes.

The upper level of the software solution is the *Web Server Layer*, responsible for the HTTP communication with the client terminals (e.g. consoles for subscribers' data base administration). The web service SOAP requests and responses are sent using the HTTP protocol.

The Web Service Layer is responsible for processing the SOAP requests received from the client, executing the required operation and sending back the response to the client. All the business logic for the web service operation is implemented at this level.

We decided to build the main software modules in Java.

Any SOAP message contains the following parts:

Envelope (compulsory): the root element of a SOAP message; this element defines the XML document as a SOAP message. The namespace defines the Envelope as a SOAP Envelope. If a different namespace is used, the application generates an error and discards the message

Header (optional): contains application-specific information (like authentication, payment, etc) about the SOAP message. If the Header element is present, it must be the first child element of the Envelope.

Body (compulsory): contains the actual SOAP message.

The portable *service logic* – the partial-handover QoS-based *scenario* for bi-modal mobile communications (3G/WLAN) was programmed in *BPEL* ("Business Process Execution Language") [5] as a particular implementation of

business processes. BPEL is based on the XML scheme, on the SOAP protocol as well as on the Web Services Description Language (WSDL) [6]. BPEL offers a standard for *orchestrating* and *executing* the business process. Using BPEL, a business process that integrates a series of discrete services is designed in an end-to-end process flow.

BPEL allows to define the way through which XML messages are changed with remotely executed services, to manipulate data structures, administrate events and exceptions and design parallel flows in processes' execution.

A BPEL process "specifies the exact order of the participant web services to be invoked, sequentially or parallel".

An ordinary situation can be when a BPEL business process gets a request. To fulfill the request, the process invokes the implied web services and then answers to the original caller.

Because the BPEL process communicates with other web services, it is mostly based on the WSDL description of the web services invoked by the composite web service.

A BPEL process contains multiple steps; each of them is called an "activity". BPEL supports primitive activities as well as structural ones.

The BPEL process is described in an XML document with .bpel extension.

The document has the structure of Fig.1:

```
<process>
   <!- Definition and roles of process participants -->
   <partnerLinks> ... </partnerLinks>
   <!- Data/state used within the process -->
   <variables> ... </variables>
   <!- Properties that enable conversations -->
   <correlationSets> ... </correlationSets>
   <!- Exception handling -->
   <faultHandlers> ... </faultHandlers>
   <!- Error recovery - undoing actions  -->
   <compensationHandlers> ... </compensationHandlers>
   <!- Concurrent events with process itself  -->
   <eventHandlers> ... </eventHandlers>
   <!- Business process flow  -->
   (activities)*
</process>
```

Fig.1 XML description of a BPEL process

Partner links allow definition of external services that BPEL process can interact with. The partner link type characterizes the conversational relation between the services, defining the roles played by each of them and specifying the granted port type supplied by each service, in order to receive messages in the conversational context.

In order to build the scenario in the BPEL experiment, the ORACLE SOA Suite [7] was used as server platform, with Oracle JDeveloper for BPEL process designing.

Oracle SOA Suite contains the "Oracle BPEL Process Manager" which represents the container in which BPEL processes will be installed and executed.

The components of the Oracle BPEL Process Manager are:

The developing medium (JDeveloper BPEL Designer or Eclipse BPEL Designer) that allows designing and installing of BPEL processes.

Upon design completion, the process is deployed from the development medium into the Oracle BPEL server.

If the implementation is successful, the BPEL process can be managed from the Oracle BPEL console.

IV. Implementation of the Multi-modal QoS Evaluation Service

The implemented service basic scheme is presented in Fig.2.

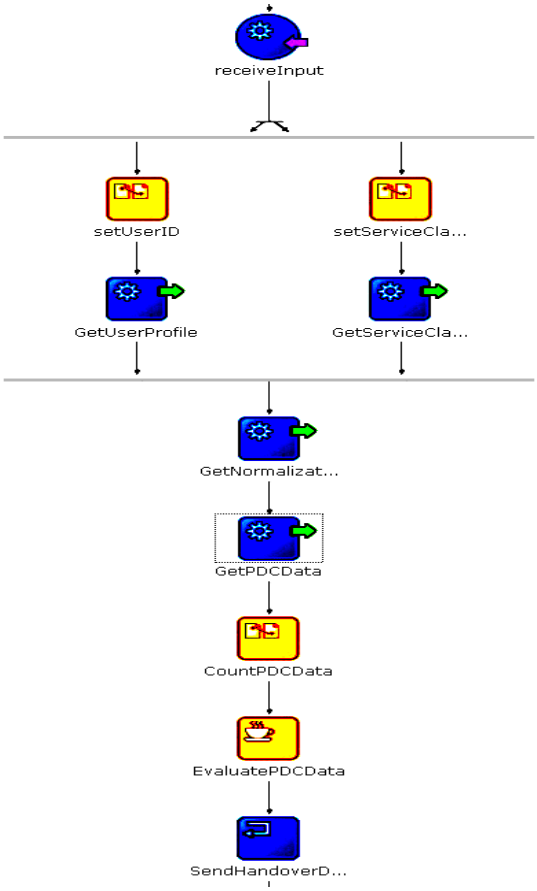

Fig.2 The implemented service logic

The *interface* developed to *emulate* the functionality of HLR (Home Location Register - or a similar database for the WLAN) that delivers the ServiceClassConfig(uration) is presented in Fig.3, both the HTML Form and the XML Source.

Via this interface, the weights of the application classes can be introduced (see the example of Fig.4) and of the subscriber profiles (user profiles - see the example of Fig.5).

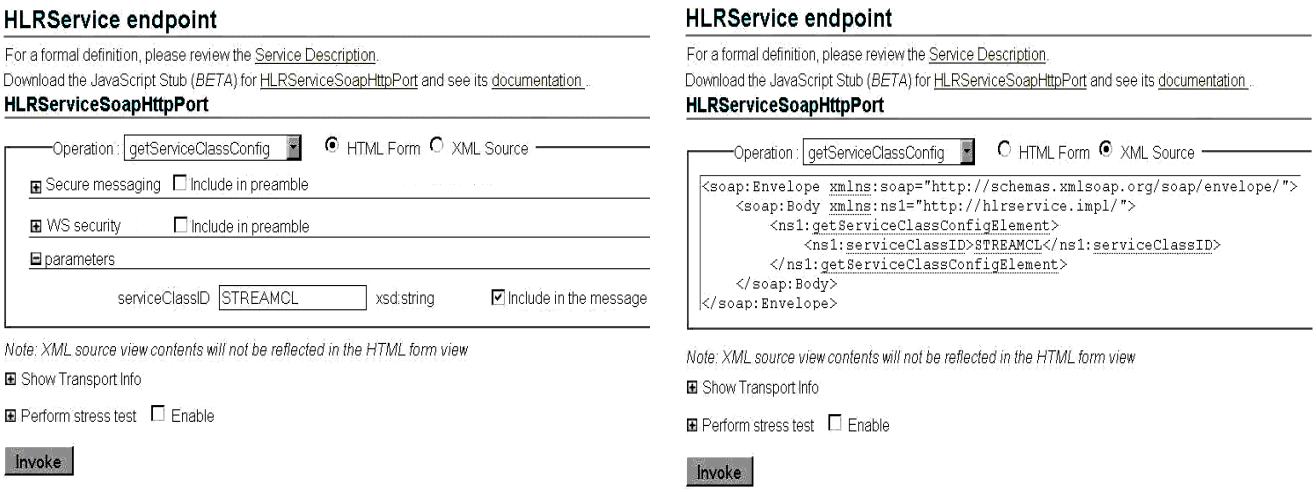

Fig.3 Interface for emulated HLR-*like* service

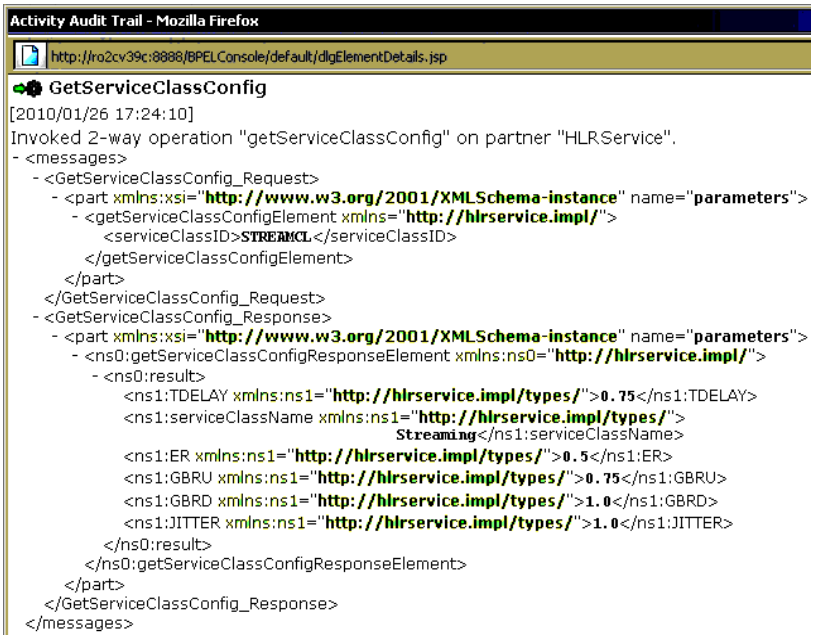

Fig.4 XML sample of Service Class weights

Fig.5 XML sample of UserProfile weights

The same input can be used also for normalization data (nevertheless, these nominators or denominators can be introduced directly in the calculation of the double-weighted QoS) – Fig.6.

```
GetNormalizationParams
...
<ns1:TDELAY xmlns:ns1="http://hlrservice.impl/types/">4000.0</ns1:TDELAY>
<ns1:ER xmlns:ns1="http://hlrservice.impl/types/">25.0</ns1:ER>
<ns1:GBRU xmlns:ns1="http://hlrservice.impl/types/">2048.0</ns1:GBRU>
<ns1:GBRD xmlns:ns1="http://hlrservice.impl/types/">8640.0</ns1:GBRD>
<ns1:JITTER xmlns:ns1="http://hlrservice.impl/types/">100.0</ns1:JITTER>
...
```

Fig.6 XML sample of Normalization Parameters

The Performance Data Collection is emulated via the interface presented in Fig.7.

PDCService endpoint

For a formal definition, please review the Service Description.
Download the JavaScript Stub (*BETA*) for PDCServiceSoapHttpPort and see its documentation.

PDCServiceSoapHttpPort

Operation: getPDCData HTML Form XML Source

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body xmlns:ns1="http://pdcservice.impl/">
        <ns1:getPDCDataElement/>
    </soap:Body>
</soap:Envelope>
```

Fig.7 Interface for emulated PDC

In fig.8, it can be seen an example of measured values.

```
GetPDCData
...
<ns0:getPDCDataResponseElement xmlns:ns0="http://pdcservice.impl/">
 - <ns0:result>
    <ns1:availableBitrateDownload xmlns:ns1="http://pdcservice.impl/types/">
                                  2465.0</ns1:availableBitrateDownload>
    <ns1:networkID xmlns:ns1="http://pdcservice.impl/types/">wifi</ns1:networkID>
    <ns1:jitter xmlns:ns1="http://pdcservice.impl/types/">11.0</ns1:jitter>
    <ns1:delay xmlns:ns1="http://pdcservice.impl/types/">1513.0</ns1:delay>
    <ns1:errorRate xmlns:ns1="http://pdcservice.impl/types/">7.0</ns1:errorRate>
    <ns1:availableBitrateUpload xmlns:ns1="http://pdcservice.impl/types/">
                                  735.0</ns1:availableBitrateUpload>
  </ns0:result>
 - <ns0:result>
    <ns1:availableBitrateDownload xmlns:ns1="http://pdcservice.impl/types/">
                                  6911.0</ns1:availableBitrateDownload>
    <ns1:networkID xmlns:ns1="http://pdcservice.impl/types/">gprs</ns1:networkID>
    <ns1:jitter xmlns:ns1="http://pdcservice.impl/types/">55.0</ns1:jitter>
    <ns1:delay xmlns:ns1="http://pdcservice.impl/types/">748.0</ns1:delay>
    <ns1:errorRate xmlns:ns1="http://pdcservice.impl/types/">22.0</ns1:errorRate>
    <ns1:availableBitrateUpload xmlns:ns1="http://pdcservice.impl/types/">
                                  172.0</ns1:availableBitrateUpload>
...
```

Fig.8 XML sample of collected performance data

Having this data, the QoS level offered by the transmission networks can be calculated and then decided the network which will route most of the traffic. Here are some details of the corresponding Java code.

In the evaluation of *available* QoS, each relevant value *„measured"* by the Performance Data Collector (avbu – available bit rate for upload; avbd – available bit rate for download etc) will be first multiplied by a weight representing the profile of the subscriber (w_pr_), then with a weight representing the class of the service (w_cl_), then normalized (multiplied or divided accordingly) with the reference normalization parameters (np_):

```
// evaluate pdc data
double qos =
    w_pr_gbru * w_cl_gbru * avbu / np_gbru +
    w_pr_gbrd * w_cl_gbrd * avbd / np_gbrd +
    w_pr_tdelay * w_cl_tdelay * np_tdelay / delay +
    w_pr_jitter * w_cl_jitter * np_jitter / jitter +
    w_pr_er * w_cl_er * np_er / errorRate;
```

V. CONCLUSIONS AND FURTHER DEVELOPMENT

It could be also considered, in a similar way, the QoS required by the application, with a partial handover to the best calculated actual QoS *only* if this value is decreasing *under* the required QoS (this means no handover effort for cheaper applications that aren't so much resource-demanding).

Similar Java code could compute this rq_qos, with rq_gbu instead of avgbu, rq_gbd instead of avgbd etc. in the previous calculation.

Here it is, from RFC2326 [8], an example of a streaming application that can determine the required parameters from a DESCRIBE command sent to the streaming media server:

```
"
C->W: GET /concert.sdp HTTP/1.1
   Host: www.example.com
W->C: HTTP/1.1 200 OK
   Content-Type: application/x-rtsl
    <session>
     <track
src="rtsp://live.example.com/concert/audio">
    </session>
C->M: DESCRIBE
rtsp://live.example.com/concert/audio RTSP/1.0
   CSeq: 1
M->C: RTSP/1.0 200 OK
   CSeq: 1
   Content-Type: application/sdp
   Content-Length: 44
   v=0
   o=- 2890844526 2890842807 IN IP4 192.16.24.202
   s=RTSP Session
   m=audio 3456 RTP/AVP 0
   a=control:rtsp://live.example.com/concert/audio
   c=IN IP4 224.2.0.1/16
C->M: SETUP rtsp://live.example.com/concert/audio
RTSP/1.0
   CSeq: 2
"
```

In a similar way, VoIP applications can determine their requirements for bitrates, delays/jitter and error rate from the codecs used to encode the voice data (on the user-plane usually the same protocol – RTP – is used in both streaming and voice/video conference).

In case when not all the parameters can be found for a specific application (like web browsing where the delay is not that important and the jitter isn't at all relevant), some default, acceptable values can be used.

Ideally, if we have applications that run on a multimodal mobile node, they should have at least a description of the QoS profile that can be used for this weighted sum calculation.

An alternative to standard BPEL in our implementation could have been proprietary suites like the popular „BizAgi" [9].

Although such suites seem more easy to use and

convenient (as they are integrated), BizAgi has some disadvantages:

BizAgi is dependent on Microsoft SQL Server ;

BizAgi can be executed only on the Microsoft Windows platform, being necessary to install a „Runtime Engine" for both C++ and J# (while Microsoft will not be able to provide support for J# programming language, the development being on hold for a while);

While modeling the process with the BizAgi editor, the process structure and all its configurations are saved in the data base, which makes porting to other platforms more difficult;

There are only a few controls and elements which can be used to model processes - very important elements are missing, such as parallel processing, error handling and the possibility to invoke partner web services.

It was also taken into consideration to test the portability of the implemented service on other BPEL platforms – „the service logic", expressed in a standard way in XPDL format („XML for Process Description Languages"), could be evaluated in BizAgi or Intalio [10] context.

I. REFERENCES

[1] ITU-T M.3000: Overview of TMN Recommendations
[2] 3GPP TS 23.060 General Packet Radio Service (GPRS); Service description; Stage 2, Release 6, September 2005.
[3] 3GPP TS 23.107 Quality of Service (QoS) concept and architecture, Release 6, June 2005
[4] W3Schools.com – SOAP Specification
http://www.w3schools.com/soap/default.asp
[5] Oracle BPEL Process Manager Developer's Guide
[6] W3Schools.com – WSDL Specification
http://www.w3schools.com/wsdl/default.asp
[7] IBM SOA -
http://www.ibm.com/developerworks/webservices/library/ws-soa-term1/
[8] RFC 2326 - Real Time Streaming Protocol (RTSP)
[9] BizAgi – Bussines Agility http://www.bizagi.com
[10] Intalio | Cloud- http://www.intalio.com/