DISPATCH Working Group   J.J. Garcia Aranda
                                                      J. Perez Lajo
Internet Draft   L.M. Diaz Vizcaino
                                                G. Munoz Fernandez
Intended status: Standards Track   Alcatel-Lucent
Expires: September 2015   C. Barcenilla
                                                          M. Cortes
                                                      J. Salvachua
                                                        J. Quemada
                                        Univ. Politecnica de Madrid

                                                    March 27, 2015

                   The Quality for Service Protocol
                   draft-aranda-dispatch-q4s-03.txt

Status of this Memo

   This Internet-Draft is submitted to IETF in full conformance with
   the provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF), its areas, and its working groups. Note that
   other groups may also distribute working documents as Internet-
   Drafts.

   Internet-Drafts are draft documents valid for a maximum of six
   months and may be updated, replaced, or obsoleted by other documents
   at any time. It is inappropriate to use Internet-Drafts as
   reference material or to cite them other than as "work in progress."

   The list of current Internet-Drafts can be accessed at
   http://www.ietf.org/ietf/1id-abstracts.txt

   The list of Internet-Draft Shadow Directories can be accessed at
   http://www.ietf.org/shadow.html

   This Internet-Draft will expire on September 27, 2015.

Copyright Notice

Abstract

   This memo describes an application level protocol for the standard
   communication of e2e QoS compliance information using a protocol
   based on Hypertext Transfer Protocol (HTTP), which forms the basis
   for the World Wide Web, and Session Description Protocol (SDP).
   Quality for Service Protocol (Q4S) provides a mechanism for latency,
   jitter, bandwidth and packet loss negotiation and monitoring,
   alerting whenever one of the negotiated conditions is violated.

   Implementation details on the actions to be triggered upon
   reception/detection of QoS alerts exchanged by the protocol are out
   of scope of this draft, it is application dependant (e.g. increase
   quality, reduce bit-rate) or even network dependant (e.g. change
   connection's quality profile).

Table of Contents

1. Introduction

   The World Wide Web (WWW) is a distributed hypermedia system which
   has gained widespread acceptance among Internet users. Although WWW
   browsers support other, preexisting Internet application protocols,
   the native and primary protocol used between WWW clients and servers
   is the HyperText Transfer Protocol (HTTP) (RFC 2616 [1]). The ease

of use of the Web has prompted its widespread employment as a
client/server architecture for many applications. Many of such
applications require the client and the server to be able to
communicate each other and exchange information with certain quality
constraints.

Quality in communications at application level consists of four
measurable parameters:

   o Latency: The time a message takes to travel from source to
      destination. It may be approximated to RTT/2 (Round trip time),
      assuming the networks are symmetrical. In this context we will
      consider the statistical median formula.

   o Jitter: latency variation. There are some formulas to calculate
      Jitter, and in this context we will consider the arithmetic
      mean formula.

   o Bandwidth: bit rate of communication. To assure quality, a
      protocol MUST assure the availability of the bandwidth needed
      by the application.

   o Packet loss: The percentage of packet loss is closely related
      to bandwidth and jitter. Affects bandwidth because a high
      packet loss implies sometimes retransmissions that also
      consumes extra bandwidth, other times the retransmissions are
      not achieved (for example in video streaming over UDP) and the
      information received is less than the required bandwidth. In
      terms of jitter, a packet loss sometimes is seen by the
      destination like a larger time between arrivals, causing a
      jitter growth.

Q4S provides a mechanism for quality monitoring based on an HTTP
syntax and SDP in order to be easily integrated in WWW, but it may
be used by any type of application, not only those based on HTTP.
Quality requirements may be needed by any type of application that
communicates using any kind of protocol, especially those with real-
time constraints. Depending on the nature of each application the
constraints may be different leading to different parameter
thresholds that need to be met.

Q4S is an application level Client/Server protocol that continuously
measures session quality for a given flow (or set of flows), end-to-
end and in real-time; raising alerts if quality parameters are below
a given pre-negotiated threshold. Q4S describes when these alerts
need to be sent and the entity receiving them. The actions

undertaken by the receiver of the alert are out of scope of the protocol.

Q4S is session-independent from the application flows, in order to minimize the impact on them. To perform the measurements, two control flows are created on both communication paths (forward and reverse directions).

1.1. Motivation

Monitoring quality of service (QoS) in computer networks is useful for several reasons:

   o Enable real-time services and applications to verify whether
      network resources achieve a certain QoS level.

   o Monitoring helps real-time services and applications to run
      through the Internet, allowing the existence of Application
      Content providers (ACPs) which offer guaranteed real-time
      services to the final users.

   o Monitoring may also be required by Peer to Peer (P2P) real-time
      applications for which Q4S can be used

   o Enable ISPs to offer QoS to any ACP or final user application
      in an accountable way

   o Enable e2e negotiation of QoS parameters, independently of the
      Internet service providers of both endpoints.

  A protocol to monitor QoS must address the following issues:

   o Must be ready to be used in conjunction with current standard
      protocols and applications, without forcing a change on them.

   o Must have a formal and compact way to specify quality
      constraints of the desired application to run.

   o Must have measurement mechanisms avoiding application
      disruption, minimizing network resources consumption.

   o Must have specific messages to alert about the violation of
      quality constraints in different directions (forward and
      reverse), because network routing may not be symmetrical, and
      of course, quality constraints may not be symmetrical.

o   Must protect the data (constrains, measurements, QoS levels
    demanded from the network) in order to avoid the injection of
    malicious data in the measurements.

1.2. Summary of Features

   Quality for Service (Q4S) is a message-oriented communication
protocol that can be used in conjunction with any other application-
level protocol. Q4S protocol is a measurement protocol. Any taken
action derived from its measurements are out of scope of the
protocol.

The benefits in quality measurements provided by Q4S can be used by
any type of application that uses any type of protocol for data
transport. It provides a quality monitoring scheme for any
communication that takes place between the client and the server,
not only the Q4S communication itself.

Q4S does not establish multimedia sessions and it does not transport
application data. It monitors the fulfillment of the quality
requirements of the communication between the client and the server,
and therefore does not impose any restrictions on the type of
application, protocol or the type of usage of the monitored quality
connection.

Some applications may vary their quality requirements dynamically
for any given quality parameter. Q4S is able to adapt to the
changing application needs modifying the parameter thresholds to the
new values and monitoring the network quality according to the new
constraints. It will raise alerts if the new constraints are
violated.

Q4S session lifetime is composed of four phases with different
purposes: Handshake, Negotiation, Continuity and Termination.
Negotiation and Continuity phases perform network parameter
measurements as per a negotiated measurement procedure. Different
measurement procedures COULD be used inside Q4S, although one
default measurement mechanism is needed for compatibility reasons
and is the one defined in this draft. Basically, Q4S defines how to
transport application quality requirements and measurement results
between client and server and providing monitoring and alerting too.

Q4S MUST be executed just before starting a client-server
application which needs a quality connection in terms of latency,
jitter, bandwidth and/or packet loss. Once client and server have

succeeded in establishing communication under quality constraints, the application can start, and Q4S continues measuring and alerting if necessary.

The quality parameters can be suggested by the client in the first message of the handshake phase, but it's the server that accepts these parameter values or forces others. The server is in charge of deciding the final values of quality connection.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [3].

3. Overview of Operation

   This section introduces the basic operation of Q4S using simple examples. This section is of tutorial nature and does not contain any normative statements.

The first example shows the basic functions of a Q4S: communication establishment between a client and a server, quality requirement negotiations for the requested application, application start and continuous quality parameter measurements, and finally communication termination.

The client triggers the establishment of the communication requesting a specific service or application from the server. This first message must have a special URI (RFC 3986), which may force the use of the Q4S protocol if it is implemented in a standard web browser. This message consists of a Q4S BEGIN method, which can optionally include initial communication quality requirements in an SDP body.

This request is answered by the server with a Q4S 200 OK response letting the client know that it accepts the request. This response message MUST contain an SDP body with:

   o The assigned Q4S session id

   o The quality constraints required by the requested application

   o The measurement procedure to use.

o The alerting mode: there are two different scenarios for sending alerts that trigger actions on the network when measurements identify violated quality constraints. In both cases, Q4S alerts are triggered by the server.

   a) q4S-aware-network: the network is Q4S aware, and reacts by itself to these alerts. Q4S alerts are sent by the server to the client.

   b) policy-server: the network is not Q4S aware and a specific node (policy server) is in charge of triggering network tuning mechanisms. Q4S alerts are sent by the server to the policy server.

o network-alert-pause: the amount of time the server waits between consecutive Q4S alerts. Measurements are not stopped in Negotiation or Continuity Phases during this period of time, but no alerts are fired even with violated network quality constraints allowing network reconfigurations.

o app-alert-pause: the amount of time Q4S stack waits between consecutive notifications to the application even with violated application quality constraints. Notice that the Q4S client stack is in charge of notifying the client application and the Q4S server stack the server application.

Once the communication has been established (handshake phase is finished), the protocol will verify that the communication path between the client and the server meets the quality constraints on both directions, from and to the server (negotiation phase). This negotiation phase requires taking measurements of the quality parameters: latencies, jitter, bandwidth and packet loss. This phase is initiated with a client message containing a Q4S READY method, which will be answered by the server with a Q4S 200 OK response.

Negotiation measurements are achieved in two sequential stages:

o Stage 0: latency and jitter measurements

o Stage 1: bandwidth and packet loss measurements

Stage 0 measurements are being taken through Q4S PING messages sent both from both the client and the server. All Q4S PING requests will be answered by Q4S 200 OK messages to allow for bidirectional measurements.

After a pre-agreed number of measurements have been performed, determined by the measurement procedure sent by the server, two scenarios may be possible:

a) Measurements do not meet the requirements: in this case the stage 0 is repeated after sending an alert from the server to the client or from the server to the network policy server, depending on the alerting mode defined in the Handshake phase. Notice that measurements continue to be taken but no Q4S alert is fired during the network-alert-pause time.

b) Measurements do meet the requirements: in this case client moves to stage 1 sending a new READY message.

Stage 1 measurements are achieved through Q4S BWIDTH messages sent both from the client and the server. Unlike PING messages, Q4S BWIDTH requests will not be answered.

If Stage 0 and 1 meet the application quality constraints, the application may start. Q4S will enter the continuity phase measuring the network quality parameters and raising alerts in case of violation. This will be done through the Q4S PING message exchange on both connection paths.

Once the client wants to terminate the communication it sends a Q4S CANCEL message, which will be acknowledged by the server with another Q4S CANCEL message.

This figure depicts the message exchange in a successful scenario.

```
              +--------------------------------------------+
              |                                            |
              | Client                              Server |
              |                                            |
Handshake     |      --------- Q4S BEGIN ------------>     |
              |      <-------- Q4S 200 OK -----------      |
              |                                            |
Negotiation   |                                            |
(Stage 0)     |      --------- Q4S READY 0---------->      |
              |      <-------- Q4S 200 OK -----------      |
              |                                            |
              |      --------- Q4S PING ------------>      |
              |      <-------- Q4S 200 OK -----------      |
              |      <-------- Q4S PING -------------      |
              |       -------- Q4S 200 OK ---------->      |
              |      --------- Q4S PING ------------>      |
              |      <-------- Q4S PING -------------      |
              |      --------- Q4S 200 OK ---------->      |
              |      <-------- Q4S 200 OK -----------      |
              |                     ...                    |
Negotiation   |                                            |
(Stage 1)     |      --------- Q4S READY 1---------->      |
              |      <-------- Q4S 200 OK -----------      |
              |                                            |
              |      --------- Q4S BWITDH ---------->      |
              |      <-------- Q4S BWIDTH-----------       |
              |      --------- Q4S BWITDH ---------->      |
              |      <-------- Q4S BWIDTH-----------       |
              |                     ...                    |
Continuity    |      --------- Q4S READY 2 --------->      |
              |      <-------- Q4S 200 OK -----------      |          app start
              |                                            |
              |      --------- Q4S PING ------------>      |
              |      <-------- Q4S 200 OK -----------      |
              |      <-------- Q4S PING -------------      |
              |       -------- Q4S 200 OK ---------->      |
              |                                            |
Termination   |      --------- Q4S CANCEL ---------->      |          app end
              |      <-------- Q4S CANCEL -----------      |
              |                                            |
              +--------------------------------------------+
```
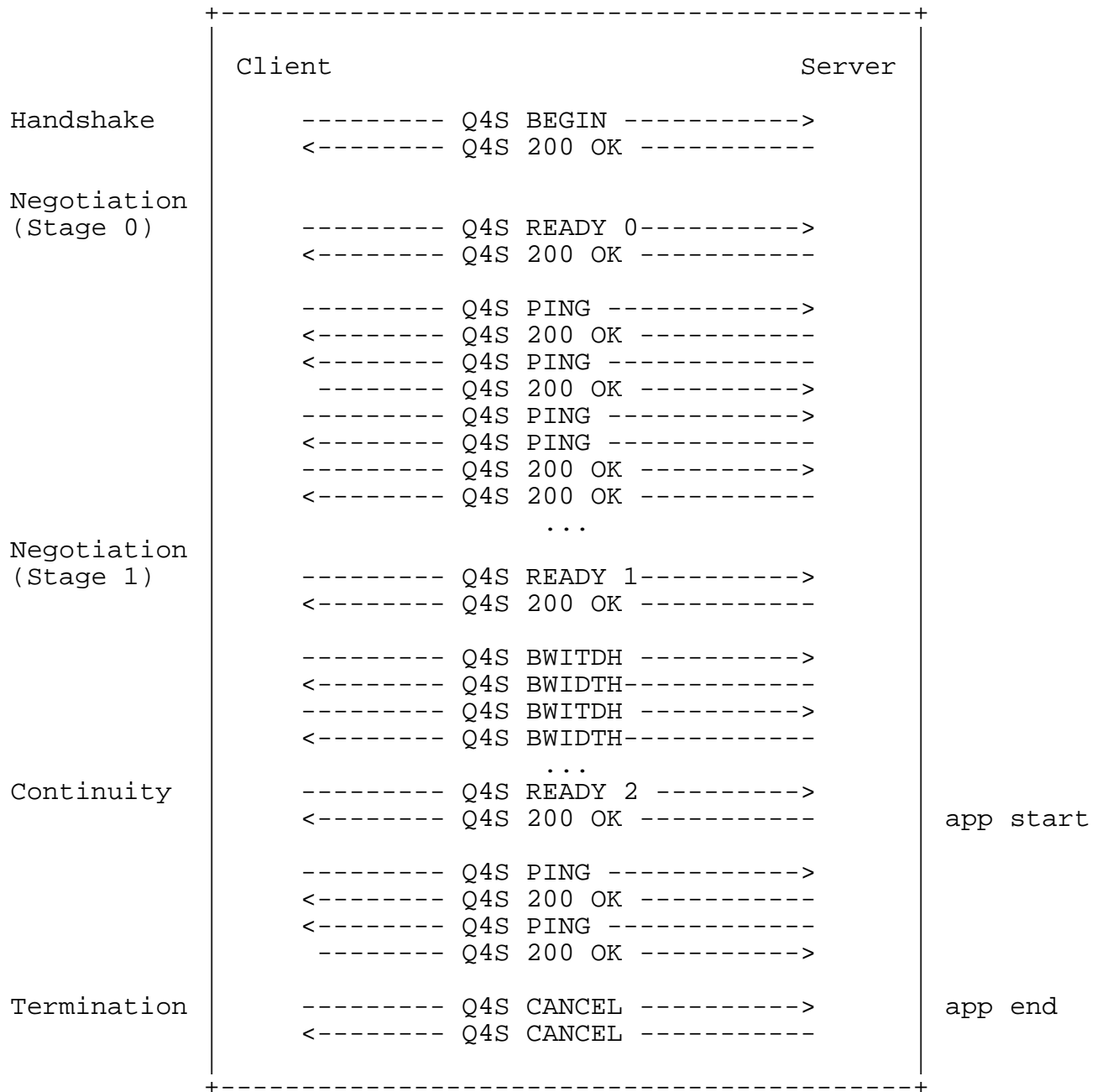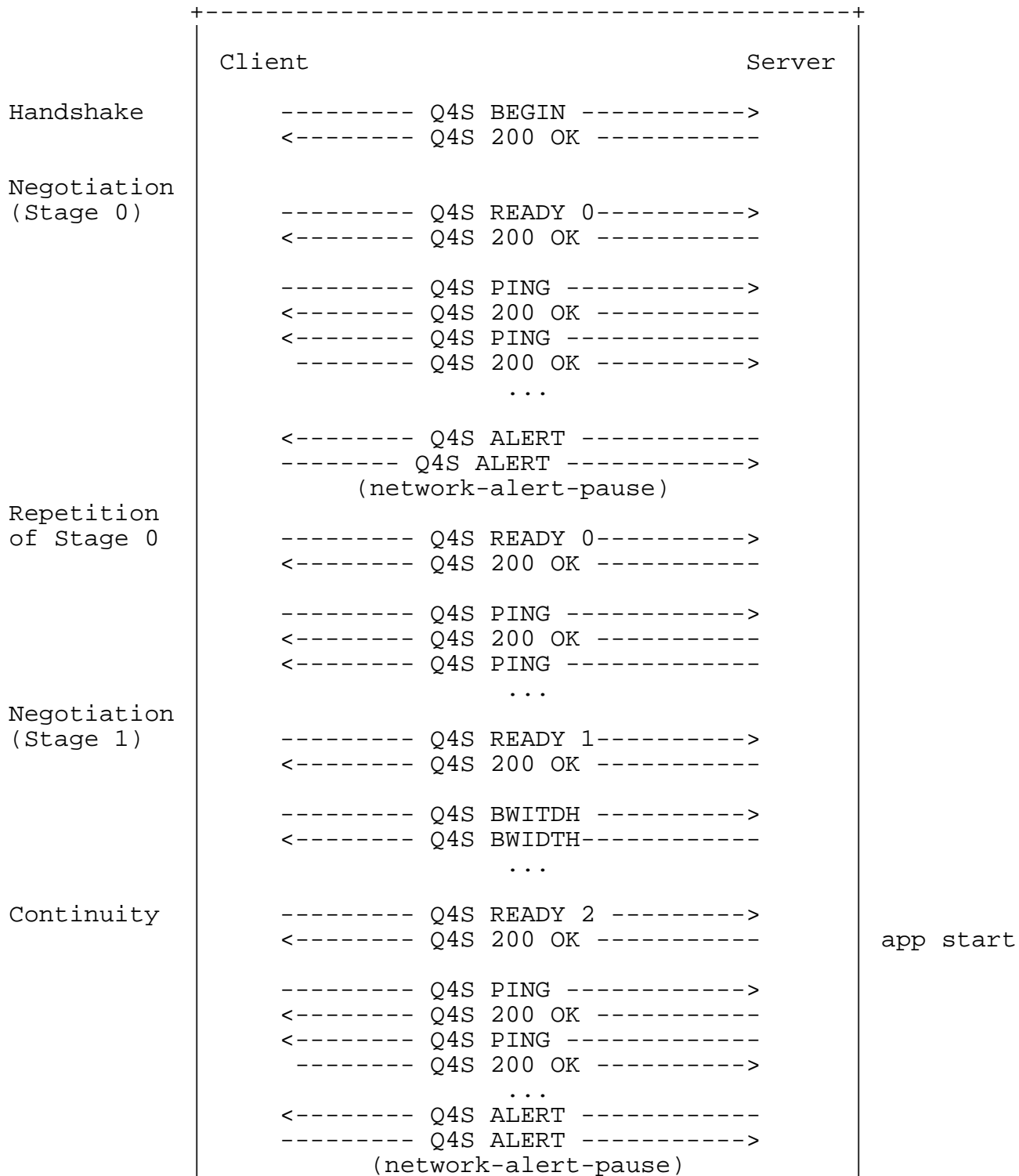
Figure 1 Successful Q4S message exchange.

Client and server measurements are included into PING and BWIDTH messages, allowing both sides of the communication to be are aware of all measurements in both directions.

The following two examples show the behavior of the Q4S protocol when quality constraints are violated. The first example shows the q4S-aware-network alerting mode scenario:

```
                 +---------------------------------------------+
                 |                                             |
                 | Client                               Server |
                 |                                             |
Handshake        |      --------- Q4S BEGIN ------------>       |
                 |      <-------- Q4S 200 OK -----------        |
                 |                                             |
Negotiation      |                                             |
(Stage 0)        |      --------- Q4S READY 0---------->        |
                 |      <-------- Q4S 200 OK -----------        |
                 |                                             |
                 |      --------- Q4S PING ------------>        |
                 |      <-------- Q4S 200 OK -----------        |
                 |      <-------- Q4S PING -------------        |
                 |       -------- Q4S 200 OK ---------->        |
                 |                    ...                      |
                 |                                             |
                 |      <-------- Q4S ALERT ------------        |
                 |      -------- Q4S ALERT ------------>        |
                 |           (network-alert-pause)             |
Repetition       |                                             |
of Stage 0       |      --------- Q4S READY 0---------->        |
                 |      <-------- Q4S 200 OK -----------        |
                 |                                             |
                 |      --------- Q4S PING ------------>        |
                 |      <-------- Q4S 200 OK -----------        |
                 |      <-------- Q4S PING -------------        |
                 |                    ...                      |
Negotiation      |                                             |
(Stage 1)        |      --------- Q4S READY 1---------->        |
                 |      <-------- Q4S 200 OK -----------        |
                 |                                             |
                 |      --------- Q4S BWITDH ---------->        |
                 |      <-------- Q4S BWIDTH-----------         |
                 |                    ...                      |
                 |                                             |
Continuity       |      --------- Q4S READY 2 --------->        |
                 |      <-------- Q4S 200 OK -----------        | app start
                 |                                             |
                 |      --------- Q4S PING ------------>        |
                 |      <-------- Q4S 200 OK -----------        |
                 |      <-------- Q4S PING -------------        |
                 |       -------- Q4S 200 OK ---------->        |
                 |                    ...                      |
                 |      <-------- Q4S ALERT ------------        |
                 |      --------- Q4S ALERT ---------->         |
                 |           (network-alert-pause)             |
                 |                                             |
```

```
                      --------- Q4S PING ------------>
                      <-------- Q4S 200 OK -----------
                      <-------- Q4S PING ------------
                      --------- Q4S 200 OK ---------->
                                   ...
                   (pause expires & violated constraints)
                      <-------- Q4S ALERT ------------
                      --------- Q4S ALERT ---------->
                            (network-alert-pause)
                      --------- Q4S PING ------------>
                      <-------- Q4S 200 OK -----------
                      <-------- Q4S PING ------------
                       -------- Q4S 200 OK ---------->
                                   ...
                      --------- Q4S READY 2 --------->
                      <-------- Q4S 200 OK -----------

                      --------- Q4S PING ------------>
                      <-------- Q4S 200 OK -----------
                      <-------- Q4S PING ------------
                       -------- Q4S 200 OK ---------->
                                   ...

   Termination        --------- Q4S CANCEL ---------->        app end
                      <-------- Q4S CANCEL -----------

        +---------------------------------------------+
              Figure 2 q4S-aware-network alerting mode.
```
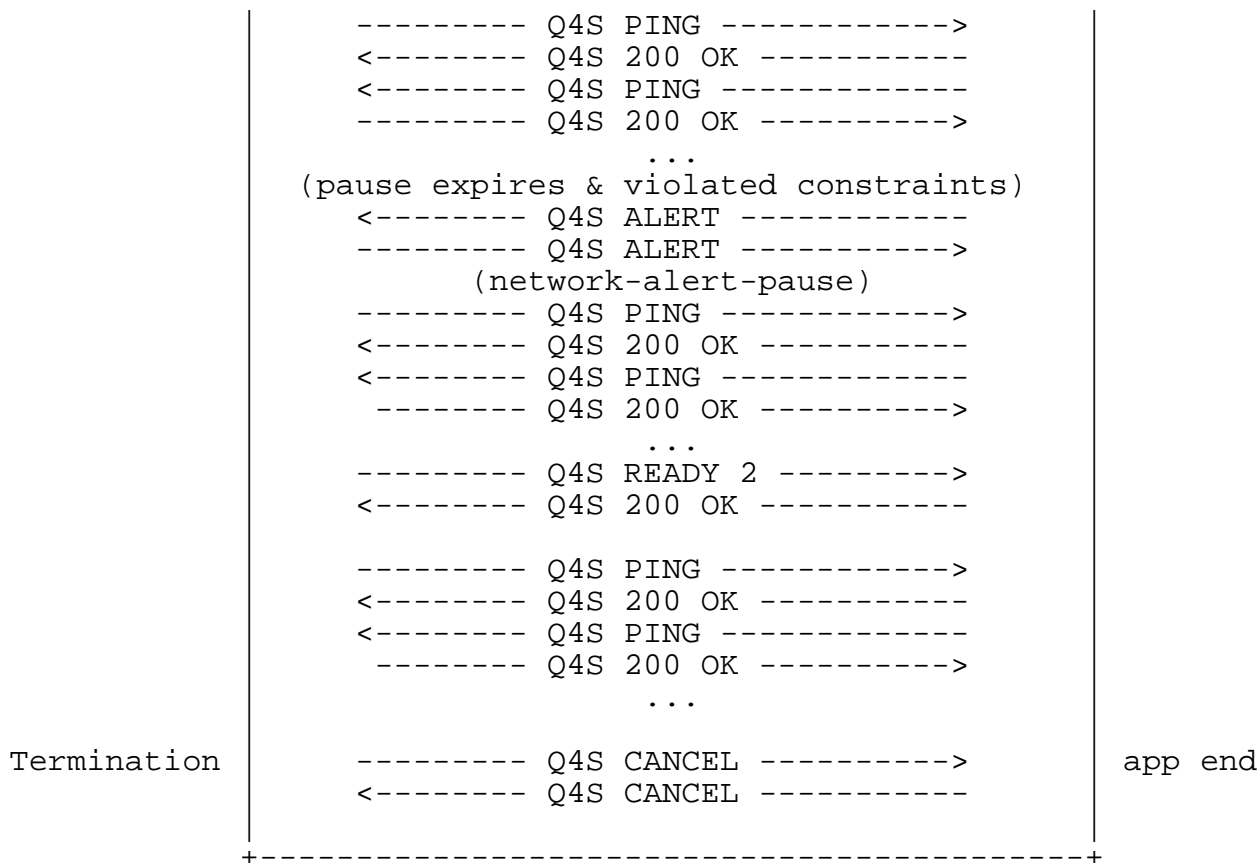
   In this q4s-aware-network alerting mode scenario, the server may
   send Q4S alerts to the client at any time on detection of violated
   quality constraints. This alerting exchange must not interrupt the
   continuity quality parameter measurements between client and server.

   The second example depicted in the following figure represents the
   policy-server alerting mode scenario, in which alerts are sent from
   the Q4S server to a network policy server. The policy server is an
   entity that can act on the network. It has a pre-defined set of
   different quality levels pre-agreed upon between the Application
   Content Provider and the ISP. The policy-server alerting mode is the
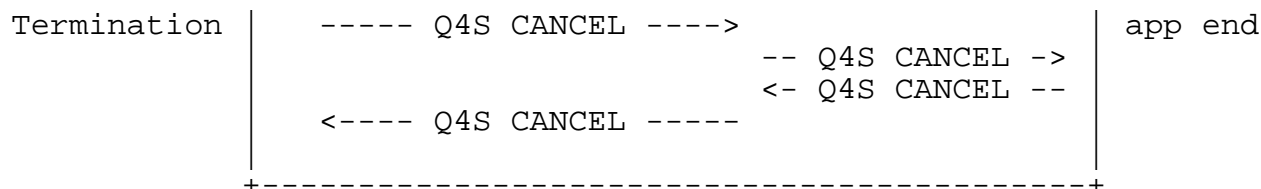   default mode.

```
                  +------------------------------------------+
                  |                                          |
                  | Client              Server       Policy  |
                  |                                  Server   |
    Handshake     |    ----- Q4S BEGIN ----->                |
                  |    <---- Q4S 200 OK -----                 |
                  |                                          |
    Negotiation   |                                          |
    (Stage 0)     |    ----- Q4S READY 0---->                |
                  |    <---- Q4S 200 OK -----                 |
                  |                                          |
                  |    ----- Q4S PING ------>                |
                  |    <---- Q4S 200 OK -----                 |
                  |    <---- Q4S PING -------                 |
                  |     ---- Q4S 200 OK ---->                 |
                  |              ...                          |
                  |    (network-alert-pause)  -- Q4S ALERT -->|
                  |                           <- Q4S ALERT ---|
    Repetition    |                                          |
    of Stage 0    |    ----- Q4S READY 0---->                |
                  |    <---- Q4S 200 OK -----                 |
                  |                                          |
                  |    ----- Q4S PING ------>                |
                  |    <---- Q4S 200 OK -----                 |
                  |    <---- Q4S PING -------                 |
                  |              ...                          |
    Negotiation   |                                          |
    (Stage 1)     |    ----- Q4S READY 1---->                |
                  |    <---- Q4S 200 OK -----                 |
                  |                                          |
                  |    ----- Q4S BWITDH ---->                |
                  |    <---- Q4S BWIDTH------                 |
                  |              ...                          |
    Continuity    |    ----- Q4S READY 2 --->                |
                  |    <---- Q4S 200 OK -----          app start
                  |                                          |
                  |    ----- Q4S PING ------>                |
                  |    <---- Q4S 200 OK -----                 |
                  |    <---- Q4S PING -------                 |
                  |    ----- Q4S PING ------>                |
                  |    <---- Q4S 200 OK ----- -- Q4S ALERT -->|
                  |    <---- Q4S PING ------- <- Q4S ALERT ---|
                  |    ----- Q4S 200 OK ---->                 |
                  |    ----- Q4S PING ------>                |
                  |              ...                          |
                  |                                          |
                  +------------------------------------------+
```

```
Termination │      ----- Q4S CANCEL ---->                    │ app end
            │                           -- Q4S CANCEL ->     │
            │                           <- Q4S CANCEL --     │
            │      <---- Q4S CANCEL -----                    │
            │                                                │
            +------------------------------------------------+
                     Figure 3 Policy-server alerting mode.
```

At the end of any Negotiation phase stage, the server sends a Q4S
alert to the policy-server in order to tune the network if quality
constraints are violated. During the period of time defined by the
network-alert-pause parameter, no further Q4S alerts are send, but
measurements are not interrupted. This way, both the client and the
server will detect network improvements as soon as possible. In a
similar way, during the continuity phase, the server may send Q4S
alerts at any time to the network policy server on detection of
violated quality constraints. This alerting exchange must not
interrupt the continuity measurements between client and server.

Q4S CANCEL messages must be forwarded from the server to the network
policy server in order to release possible assigned resources for
the session.

4. Q4S messages

Q4S is a text-based protocol and uses the UTF-8 charset (RFC 3629
[11]). A Q4S message is either a request or a response.

Both Request and Response messages use the basic format of Internet
Message Format (RFC 5322 [12]). Both types of messages consist of a
start-line, one or more header fields, an empty line indicating the
end of the header fields, and an optional message-body.

        generic-message = start-line
                            *message-header
                            CRLF
                            [message-body]
        start-line   = Request-Line / Status-Line

The start-line, each message-header line, and the empty line MUST be
terminated by a carriage-return line-feed sequence (CRLF). Note
that the empty line MUST be present even if the message-body is not.

Much of Q4S's messages and header field syntax are identical to HTTP/1.1. However, Q4S is not an extension of HTTP.

4.1. Requests

Q4S requests are distinguished by having a Request-Line for a start-line. A Request-Line contains a method name, a Request-URI, and the protocol version separated by a single space (SP) character.

The Request-Line ends with CRLF. No CR or LF are allowed except in the end-of-line CRLF sequence. No linear whitespace (LWS) is allowed in any of the elements.

        Request-Line = Method SP Request-URI SP Q4S-Version CRLF

  Method: This specification defines six methods: BEGIN for starting
        and negotiate quality sessions, READY for synchronization of
        measurements, PING and BWIDTH for quality measurements
        purpose, CANCEL for terminating sessions, and Q4S-ALERT for
        quality violations reporting.

  Request-URI: The Request-URI is a Q4S URI (RFC 2396) as described in
        7.4. The Request-URI MUST NOT contain unescaped spaces or
        control characters and MUST NOT be enclosed in "<>".

  Q4S-Version: Both request and response messages include the version
        of Q4S in use. To be compliant with this specification,
        applications sending Q4S messages MUST include a Q4S-Version
        of "Q4S/1.0". The Q4S-Version string is case-insensitive,
        but implementations MUST send upper-case. Unlike HTTP/1.1,
        Q4S treats the version number as a literal string. In
        practice, this should make no difference.

4.2. Responses

Q4S responses are distinguished from requests by having a Status-Line as their start-line. A Status-Line consists of the protocol version followed by a numeric Status-Code and its associated textual phrase, with each element separated by a single SP character. No CR or LF is allowed except in the final CRLF sequence.

     Status-Line = Q4S-Version SP Status-Code SP Reason-Phrase CRLF

The Status-Code is a 3-digit integer result code that indicates the outcome of an attempt to understand and satisfy a request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata, whereas the Reason-Phrase is intended for the human user. A client is not required to examine or display the Reason-Phrase.

While this specification suggests specific wording for the reason phrase, implementations MAY choose other text, for example, in the language indicated in the Accept-Language header field of the request.

The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. For this reason, any response with a status code between 100 and 199 is referred to as a "1xx response", any response with a status code between 200 and 299 as a "2xx response", and so on. Q4S/1.0 allows following values for the first digit:

1xx: Provisional -- request received, continuing to process the request;

2xx: Success -- the action was successfully received, understood, and accepted;

3xx: Redirection -- further action needs to be taken in order to complete the request;

4xx: Request Failure -- the request contains bad syntax or cannot be fulfilled at this server;

5xx: Server Error -- the server failed to fulfill an apparently valid request;

6xx: Global Failure -- the request cannot be fulfilled at any server.

The status codes are the same described in HTTP (RFC 2616 [1]). In the same way as HTTP, Q4S applications are not required to understand the meaning of all registered status codes, though such understanding is obviously desirable. However, applications MUST understand the class of any status code, as indicated by the first digit, and treat any unrecognized response as being equivalent to the x00 status code of that class.

The Q4S-ALERT and CANCEL requests do not have to be responded.
However, after receiving a Q4S-ALERT or CANCEL request, the server
SHOULD sends a Q4S-ALERT or CANCEL request to the client

4.3. Header Fields

Q4S header fields are identical to HTTP header fields in both syntax
and semantics.

Some header fields only make sense in requests or responses. These
are called request header fields and response header fields,
respectively. If a header field appears in a message not matching
its category (such as a request header field in a response), it MUST
be ignored.

o

   4.3.1. Common Q4S Header Fields

These fields may appear in Request and Response messages.

   o   Session-Id: the value for this header is the same session id
       used in SDP (embedded in "o" SDP parameter) and is assigned by
       the server. The messages without SDP MUST include this header.
       If a message has and SDP body, this header is optional. The
       method of <session id> allocation is up to the creating tool,
       but it is suggested that a UTC timestamp be used to ensure
       uniqueness.

   o   Sequence-Number: sequential and cyclic positive integer number
       assigned to PING and BWIDTH messages, and acknowledged in 200
       OK responses.

   o   Timestamp: this optional header contains the system time (with
       the best possible accuracy). It indicates the time in which the
       PING request was sent. If this header is present in PING
       messages, then the 200 OK response messages include this value.

   o   Stage: this is used in client's READY requests and server's
       200 OK responses during the Negotiation and Continuity phases
       in order to synchronize the initiation of the measurements.
       Example: Stage: 0

4.3.2. Specific Q4S Request Header Fields

In addition to HTTP header fields, these are the specific Q4S request header fields

o User-Agent: this header contains information about the implementation of the user agent. This is for statistical purposes, the tracing of protocol violations, and the automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents SHOULD include this field with requests. The field can contain multiple product tokens and comments identifying the agent and any sub-products which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

o Signature: this header contains a digital signature that can be used by the network or policy server to validate the SDP, preventing security attacks. The signature is an optional header generated by the server according to the pre-agreed security policies between the Application Content Provider and the ISP. For example, a hash algorithm and encryption method such as MD5 (RFC 1321 [6]) and RSA (RFC 2437 [7]) based on the server certificate could be used. This certificate is supposed to be delivered by a Certification Authority (CA) or policy owner to the server. The signature is applied to the SDP body.

   Signature= RSA (MD5 (<sdp>), <certificate>)

   If the signature is not present, other validation mechanism may be implemented in order to provide assured quality with security and control.

o Measurements: this header carries the measurements of the quality parameters in PING and BWIDTH requests. The format is:

Measurements: "l=" " "|[0..9999] ", j=" " "|[0..9999] ", pl=" " "|[0.00 .. 100.00] ", bw=" " "|[0..9999]

Where "l" stands for latency followed by the measured value or an empty space, "j" stands for jitter followed by the measured value or an empty space, "pl" stands for packetloss followed by the measured value in % or an empty space and "bw" stands for bandwidth followed by the measured value or an empty space.

4.3.3. Specific Q4S Response Header Fields

o Expires: its purpose is to provide a sanity check and allow the
   server to close inactive sessions. If the client does not send
   a new request before the expiration time, the server MAY close
   the session. The value MUST be an integer and the measurement
   units are milliseconds.

   In order to keep the session open the server MUST send a Q4S
   alert before the session expiration (Expires header), with the
   same quality levels and an alert cause of "keep-alive". The
   purpose of this alert is to avoid TCP sockets (which were
   opened with READY message) from being closed, specially in NAT
   scenarios.

4.4. Bodies

   Requests, including new requests defined in extensions to this
specification, MAY contain message bodies unless otherwise noted.
The interpretation of the body depends on the request method.

For response messages, the request method and the response status
code determine the type and interpretation of any message body. All
responses MAY include a body.

The Internet media type of the message body MUST be given by the
Content-Type header field.

   4.4.1. Encoding

   The body MUST NOT be compressed. This mechanism is valid for
other protocols such as HTTP and SIP (RFC 3261 [13]), but
a compression/coding scheme will limit certain logical
implementations of the way the request is parsed, thus, making the
protocol concept more implementation dependent. In addition,
bandwidth calculation may not be valid if compression is used.
Therefore, the HTTP request header "Accept-Encoding" cannot be used
in Q4S with different values than "identity" and if it is present in
a request, the server MUST ignore it. In addition, the response
header "Content-Encoding" is optional, but if present, the unique
permitted value is "identity".

The body length in bytes is provided by the Content-Length header
field. The "chunked" transfer encoding of HTTP/1.1 MUST NOT be used

for Q4S (Note: The chunked encoding modifies the body of a message
in order to transfer it as a series of chunks, each one with its own
size indicator.)

5. Q4S method definitions

The Method token indicates the method to be performed on the
resource identified by the Request-URI. The method is case-
sensitive.

        Method = "BEGIN" | "READY" | "PING" | "BWIDTH" |
                  "Q4S-ALERT" | "CANCEL" | extension-method

        extension-method = token

The list of methods allowed by a resource can be specified in an
"Allow" header field (RFC 2616 [1] section 14.7). The return code of
the response always notifies the client when a method is currently
allowed on a resource, since the set of allowed methods can change
dynamically. Any server application SHOULD return the status code
405 (Method Not Allowed) if the method is known, but not allowed for
the requested resource, and 501 (Not Implemented) if the method is
unrecognized or not implemented by the server.

5.1. BEGIN

The BEGIN method requests information from a resource identified by
a Q4S URI. The semantics of this method is the starting of a quality
session.

This method is only used during the handshake phase to retrieve the
SDP containing session id and all quality and operation parameters
for the desired application to run.

When a BEGIN message is received by the server, any current quality
session is cancelled and a new session should be created.

The response to a Q4S BEGIN request is not cacheable.

5.2. READY

   The READY method is used to synchronize the starting time for
   sending of PING and BWIDTH messages over UDP between clients and
   servers. The stage header included in this method is mandatory.

   This message is only used in negotiation and continuity phases, and
   only just before making a measurement. Otherwise (out of this
   context), the server MUST ignore this method.

5.3. PING

   This message is used during the negotiation and continuity phases to
   measure the RTT and jitter of a session. The message MUST be sent
   only over UDP ports.

   The fundamental difference between the PING and BWIDTH requests is
   reflected in the different measurements achieved with them. PING is
   a short message, and MUST be answered in order to measure RTT and
   jitter, whereas BWIDTH is a long message (1 Kbyte or more) and MUST
   NOT be answered.

   PING is a request method that can be originated by client but also
   by server. Client MUST also answer the server PING messages,
   assuming a "server role" for these messages during measurement
   process.

   The Measurements header included in this method is mandatory, and
   provides updated measurements values for latency, jitter and packet
   loss to the counterpart.

5.4. BWIDTH

   This message is used only during the Negotiation phase to measure
   the bandwidth and packet loss of a session. The message MUST be sent
   only over UDP ports.

   BWIDTH is a request method that can be originated by the client but
   also by server. Both (client and server) MUST NOT answer BWIDTH
   messages.

   The Measurements header included in this method is mandatory, and
   provides updated measurements values for bandwidth and packet loss
   to the counterpart.

5.5. Q4S-ALERT

This is the request message that Q4S generates when the measurements
indicate that quality constraints are being violated. It is used
during the negotiation and continuity phases.

This informative message indicates that the user experience is being
degraded and includes the details of the problem (bandwidth, jitter,
packet loss measurements). The Q4S-ALERT message does not contain
any detail on the actions to be taken, which depends on the
agreements between all involved parties.

Q4S-ALERT request does not have to be answered with a response
message unless there is an error condition. However, after receiving
a Q4S-ALERT request, the counterpart answers with a Q4S-ALERT
request. The response to a Q4S-ALERT request is not cacheable.

This method is always initiated by the server in both alerting
modes. In q4s-aware-network alerting mode, the Q4S-ALERT messages
are fired by the server and sent to the client, advising the network
to react by itself. In policy-server alerting mode the Q4S-ALERT
messages are triggered by the server and sent to network policy
server.

5.6. CANCEL

The semantics of CANCEL message is the release of the Q4S session id
and the possible resources assigned to the session. This message
could be triggered by Q4S stack or by the application using the
stack (through an implementation dependant API).

In the same way as Q4S-ALERT, CANCEL must not be answered with a
response message. However, if the server receives a CANCEL message,
it must answer with a CANCEL request message towards the client,
acknowledging the reception.

In the policy-server alerting mode, the server MUST forward the Q4S
CANCEL messages received from the client to network policy server in
order to release possible assigned resources for the session. The
policy server must answer the CANCEL message with a CANCEL request
message towards the server, acknowledging the reception.

6. Response codes

   Q4S response codes are used for TCP and UDP. However, in UDP only
   the response code 200 is used.

   6.1. 100 Trying

   This response indicates that the request has been received by the
   next-hop server (the policy server) and that some unspecified action
   is being taken on behalf of this request (for example, a database is
   being consulted). This response, like all other provisional
   responses, stops retransmissions of a Q4S-ALERT during the network-
   alert-pause time.

   6.2. Success 2xx

   2xx responses give information about success of a request.

      6.2.1. 200 OK

   The request has succeeded.

   6.3. Redirection 3xx

   3xx responses give information about the user's new location, or
   about alternative services that might be able to satisfy the
   request.

   The requesting client SHOULD retry the request at the new
   address(es) given by the Location header field.

   6.4. Request Failure 4xx

   4xx responses are definite failure responses from a particular
   server. The client SHOULD NOT retry the same request without
   modification (for example, adding appropriate headers or SDP
   values). However, the same request to a different server might be
   successful.

      6.4.1. 400 Bad Request

   The request could not be understood due to malformed syntax. The
   Reason-Phrase SHOULD identify the syntax problem in more detail, for
   example, "Missing Sequence-Number header field".

6.4.2. 404 Not Found

The server has definitive information that the user does not exist at the domain specified in the Request-URI. This status is also returned if the domain in the Request-URI does not match any of the domains handled by the recipient of the request.

6.4.3. 405 Method Not Allowed

The method specified in the Request-Line is understood, but not allowed for the address identified by the Request-URI.

The response MUST include an Allow header field containing a list of valid methods for the indicated address.

6.4.4. 406 Not Acceptable

The resource identified by the request is only able of generating response entities that have content characteristics not acceptable according to the Accept header field sent in the request.

6.4.5. 408 Request Timeout

The server could not produce a response within a suitable amount of time, and the client MAY repeat the request without modifications at any later time

6.4.6. 413 Request Entity Too Large

The server is refusing to process a request because the request entity-body is larger than the one that the server is willing or able to process. The server MAY close the connection to prevent the client from continuing the request.

6.4.7. 414 Request-URI Too Long

The server is refusing to process the request because the Request-URI is longer than the one that the server accepts.

6.4.8. 415 Unsupported Media Type

The server is refusing to process the request because the message body of the request is in a format not supported by the server for the requested method. The server MUST return a list of acceptable formats using the Accept, Accept-Encoding, or Accept-Language header field, depending on the specific problem with the content.

6.4.9. 416 Unsupported URI Scheme

The server cannot process the request because the scheme of the URI
in the Request-URI is unknown to the server.

6.5. Server Failure 5xx

5xx responses are failure responses given when a server itself is
having trouble.

6.5.1. 500 Server Internal Error

The server encountered an unexpected condition that prevented it
from fulfilling the request. The client MAY display the specific
error condition and MAY retry the request after several seconds.

6.5.2. 501 Not Implemented

The server does not support the functionality required to fulfill
the request. This is the appropriate response when a Server does not
recognize the request method and it is not capable of supporting it
for any user.

Note that a 405 (Method Not Allowed) is sent when the server
recognizes the request method, but that method is not allowed or
supported.

6.5.3. 503 Service Unavailable

The server is temporarily unable to process the request due to a
temporary overloading or maintenance of the server. The server MAY
indicate when the client should retry the request in a Retry-After
header field. If no Retry-After is given, the client MUST act as if
it had received a 500 (Server Internal Error) response.

A client receiving a 503 (Service Unavailable) SHOULD attempt to
forward the request to an alternate server. It SHOULD NOT forward
any other requests to that server for the duration specified in the
Retry-After header field, if present.

Servers MAY refuse the connection or drop the request instead of
responding with 503 (Service Unavailable).

6.5.4. 504 Server Time-out

The server did not receive a timely response from an external server
it accessed in attempting to process the request.

6.5.5. 505 Version Not Supported

The server does not support, or refuses to support, the Q4S protocol version that was used in the request. The server is indicating that it is unable or unwilling to complete the request using the same major version as the client, other than with this error message.

6.5.6. 513 Message Too Large

The server was unable to process the request since the message length exceeded its capabilities.

6.6. Global Failures 6xx

6xx responses indicate that a server has definitive information about a particular policy not satisfied for processing the request.

6.6.1. 600 session does not exist

The Session-Id is not valid

6.6.2. 601 quality level not allowed

The QOS level requested is not allowed for the pair client/server

6.6.3. 603 Session not allowed

The session is not allowed due to some policy (number of sessions allowed for the server is exceeded, or the time band of the Q4S-ALERT is not allowed for the pair client/server, etc)

6.6.4. 604 authorization not allowed

The policy server does not authorize the Q4S-ALERT quality session improvement operation due to an internal or external reason.

7. Protocol

This section describes the measurement procedures, the SDP structure of the Q4S messages, the different Q4S protocol phases and the messages exchanged in them.

7.1. Protocol Phases

   All elements of the IP network contribute to the quality in terms
of latency, jitter, bandwidth and packet loss. All these elements
have their own quality policies in terms of priorities, traffic
mode, etc. and each element has its own way to manage the quality.
The purpose of a quality connection is to establish an end-to-end
communication with enough quality for the application to function
flawlessly.

To monitor quality constraints of the application, four phases are
defined and can be seen in the following figure:

```
+-----------------------------------------------------------------+
|                                                                 |
|                                                                 |
|  Handshake ---> Negotiation -+--> Continuity ---> Termination   |
|                 A            |       (app start)     (app end)   |
|                 |            |                          A        |
|                 |        violated                       |        |
|                 |        constraints                    |        |
|                 |          |      |                      |        |
|                 +------+   |      +--------------------+ |        |
|                                                                 |
+-----------------------------------------------------------------+
```
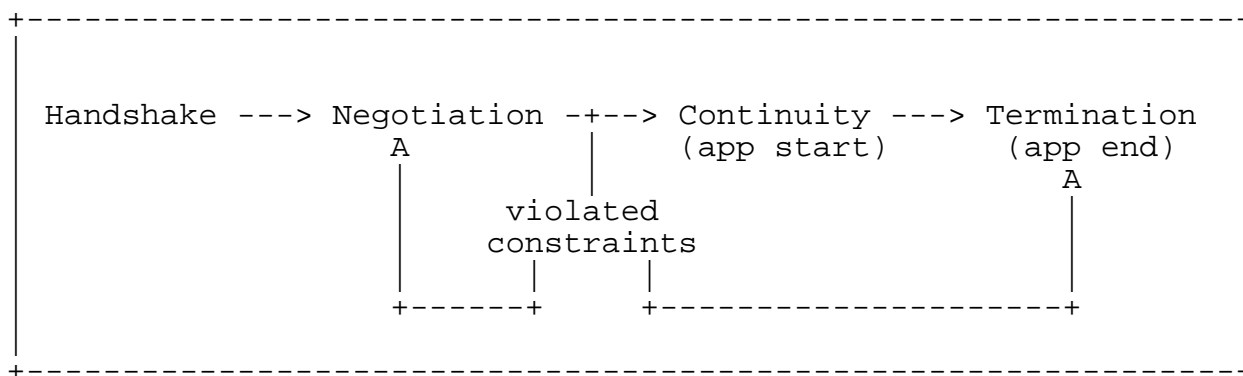
                  Figure 4 Session lifetime phases.

   o Handshake phase: in which the server is contacted by the client
      and in the answer message the quality constraints for the
      application is communicated embedded in an SDP.

   o Negotiation phase: in which the quality of the connection is
      measured in both directions (latency, jitter, bandwidth and
      packet loss), and Q4S messages may be sent in order to alert if
      the measured quality does not meet the constraints. This phase
      is iterative until quality constraints are reached or the
      session is cancelled after a number of measurement cycles with
      consistent violation of the quality constraints. Just after
      reaching the quality requirements, Q4S provides a simple
      optional mechanism using HTTP to start the application.

   o Continuity phase: in which quality is continuously measured. In
      this phase the measurements MUST avoid disturbing the
      application by consuming network resources. If application
      quality constraints are not met, the Q4S stack will notify the
      application. If later the quality improves again, the Q4S stack
      will notify the application too. If quality falls below the
      network parameter constraints, a Q4S network alert shall be
      issued. If after some alerts the quality constraints are
      unreachable, the protocol SHOULD move to Termination phase.

   o Termination phase: in which the Q4S session is terminated. The
      application may be closed too or may not start.

7.2. SDP Structure

The original goal of SDP was to announce necessary information for
the participants and multicast MBONE (Multicast Backbone)
applications. Right now, its use has been extended to the
announcement and the negotiation of multimedia sessions. The purpose
of Q4S is not to establish media stream sessions, but to monitor a
quality connection. This connection may be later used to establish
any type of session including media sessions; Q4S does not impose
any conditions on the type of communication requiring quality
parameters.

SDP will be used by Q4S to exchange quality constraints and will
therefore always have all the media attributes ("m") set to zero.

The SDP embedded in the messages is the container of the quality
parameters. As these may vary depending on the direction of the
communication (to and from the client) all quality parameters need
to specify the uplink and downlink values: <uplink> / <downlink>.
When one or both of these values are empty, it MUST be understood as
needing no constraint on that parameter and/or that direction.

The uplink direction MUST be considered as being the communication
from the client to the server. The downlink direction MUST be
considered as being the communication from the server to the client.

The SDP information can comprise all or some of the following
parameters shown in the example below. This is an example of an SDP
message used by Q4S included in the 200 OK response to a Q4S BEGIN
request.

```
v=0
o=q4s-UA 53655765 2353687637 IN IP4 192.0.2.33
s=Q4S
i=Q4S parameters
t=0 0
a=qos-level:0/0
a=alerting-mode:policy-server
a=network-alert-pause:5000
a=app-alert-pause:25000
a=public-address:client IP4 80.0.2.33
a=public-address:server IP4 198.51.100.58
a=network:latency:40
a=network:jitter:10/10
a=network:bandwidth:20/6000
a=network:packetloss:0.50/0.50
a=application:latency:35
a=applicacion:jitter:8/8
a=applicacion:bandwidth:25/7000
a=applicacion:packetloss:0.30/0.30
a=flow:app clientListeningPort TCP/10000-20000
a=flow:app clientListeningPort UDP/15000-18000
a=flow:app serverListeningPort TCP/56000
a=flow:app serverListeningPort UDP/56000
a=flow:q4s clientListeningPort UDP/55000
a=flow:q4s clientListeningPort TCP/55001
a=flow:q4s serverListeningPort UDP/56000
a=flow:q4s serverListeningPort TCP/56001
```

Notice that in this SDP example there are no measurement attributes, these will only be included in the SDP body in Q4S-ALERT messages.

7.2.1. "qos-level" attribute

The "qos-level" attribute contains the QoS level for uplink and downlink. Default values are 0 for both directions. The meaning of each level is out of scope of Q4S, but a higher level SHOULD correspond to a better service quality.

The "qos-level" attribute may be changed during the protocol lifetime raising or lowering the value as necessary following the network measurements and the application needs.

7.2.2. "alerting-mode" attribute

The "alerting-mode" attribute specifies the player in charge of triggering Q4S alerts in case of constraint violation. There are two possible values:

a) q4s-aware-network: Q4S alerts are triggered by the server to the client. In this case the network is supposed to be Q4S aware, and reacts by itself to these alerts.

b) policy-server: Q4S alerts are sent by the server to the network policy server. In this case the network is not Q4S aware and a specific node (policy server) is in charge of triggering network tuning mechanisms.

The "alerting-mode" attribute is optional and if not present "policy-server" mode is assumed.

7.2.3. "network-alert-pause" attribute

The "network-alert-pause" attribute specifies the amount of time (in milliseconds) the server waits between consecutive Q4S alerts. Measurements are not stopped in Negotiation or Continuity Phases during this period of time, but no alerts are fired even with violated network quality constraints allowing network reconfigurations.

7.2.4. "app-alert-pause" attribute

The "app -alert-pause" attribute specifies the amount of time (in milliseconds) that Q4S stack waits between consecutive notifications to the application even with violated application quality constraints. Notice that the Q4S client stack is in charge of notifying the client application and the Q4S server stack is in charge of notifying the server application. If this attribute is not present, the Q4S stack will never notify the application.

7.2.5. "public-address" attributes

This attribute contains the public IP address of the client and the server. The server fills these attributes with his own public IP address and the public IP address of the first message received from the client in the handshake phase.

The purpose of these attributes is to make available the addressing information to network policy server or other external entities in charge of processing Q4S-ALERT messages.

   7.2.6. "network:latency" attribute

The maximum latency (considered equal for uplink and downlink)
tolerance are specified in the "network:latency" attribute,
expressed in milliseconds. If the latency constraints are not met, a
Q4S-ALERT method will be raised. If the "network:latency" attribute
is not present or has a 0 value, no latency constraints need to be
met and no measurements MAY be taken.

   7.2.7. "network:jitter" attribute

The maximum uplink and downlink jitter tolerance are specified in
the "network:jitter" attribute, expressed in milliseconds. If the
jitter constraints are not met, a Q4S-ALERT method will be raised.
If "network:jitter" attribute is not present or has a 0 value, no
jitter constraints need to be met and no measurements MAY be taken.

   7.2.8. "network:bandwidth" attribute

The minimum uplink and downlink bandwidth are specified in the
"network:bandwidth" attribute, expressed in kbps. If the bandwidth
constraints are not met, a Q4S-ALERT method will be raised. If
"network:bandwidth" attribute is not present or has a 0 value, no
bandwidth constraints need to be met and no measurements MAY be
taken.

   7.2.9. "network:packetloss" attribute

The maximum uplink and downlink packet loss tolerance are specified
in the "network:packetloss" attribute expressed in percentage (two
decimal accuracy). If the packetloss constraints are not met, a Q4S-
ALERT method will be raised. If "network:packetloss" attribute is
not present or has a 0 value, no packetloss constraints need to be
met and no measurements MAY be taken.

   7.2.10. Application attributes

One or more of these attributes may not be present. They MUST not be
used if the application requiring quality service does not accept
alert messages. They specify the thresholds for latency, jitter,
bandwidth and packetloss in order for the Q4S stack to notify the
application. The Q4S client stack will notify the client application
and the Q4S server stack will notify the server application. The
mechanisms of this notification is implementation dependent and out
of scope of this draft.

There are four application attributes:

```
a=application:latency:35
a=applicacion:jitter:8/8
a=applicacion:bandwidth:25/7000
a=applicacion:packetloss:0.30/0.30
```

The uplink/downlink values of the application attributes SHOULD be
more restrictive than the network quality constraints. This will
give the application the possibility to react and adapt to network
issues before the network quality degrades more, for example, by
starting the application in lower video quality requiring less
bandwidth or reducing interaction speed at multiplayer games when
latency grows.

Application parameters MAY be present without network quality
parameter requirements. If this is the case, no Q4S-ALERTS will be
triggered but the application will be notified of any quality
parameter that falls below the application quality constraints.
Therefore, in this case, "app-alert-pause" attribute is mandatory.

If the "app-alert-pause" attribute is present but no application
parameters have been specified, the Q4S stack SHOULD notify the
application all measured parameters (as specified by the network
quality requirements) every app-alert-pause milliseconds. This
scenario is useful when application needs feedback about
measurements periodically and not only in specific degraded
conditions.

   7.2.11. "flow" attributes

These attributes specify the flows (protocol, destination IP/ports)
of data over TCP and UDP ports to be used in uplink and downlink
communications.

Several "flow" attributes can be defined. These flows identify the
listening port (client or server), the protocol (TCP or UDP) (RFC
761 [8] and RFC 768 [9]) with the range of ports that are going to
be used by the application and, of course, by the Q4S protocol (for
quality measurements). All defined flows (app and q4s) will be
considered within the same quality profile, which is determined by
the qos-level attribute in each direction. This allows to assume
that measurements on q4s flows are the same experimented by the
application which is using app flows.

During negotiation and continuity phases the specified Q4S ports in
the "flow:q4s" attributes of SDP will be used for Q4S messages.

The Q4S flows comprise two UDP flows and two TCP flows (one uplink and one downlink for each one) whereas application traffic MAY consist of many flows, depending on its nature. The handshake phase takes place through the Q4S Contact URI, using TCP port 80, for example. However the negotiation and continuity phases will take place on the specified Q4S ports (UDP and TCP) specified in the SDP.

The "clientListeningPort" is a port in which the client listens for server requests and MUST be used as origin port of client responses. The "serverListeningPort" is a port in which server is listening for incoming messages from the client. The origin port of server responses may be different than "serverListeningPort" value.

If "clientListeningPort" is zero (a=flow:q4s clientListeningPort TCP/0), the client MAY choose one randomly as per OS standard rules. Client ports inside the SDP must always be matched against actual received port values on the server side in order to deal with NAT/NATP devices. If zero value or incorrect value is present, server must set the value to the received origin port in the next message with SDP (200 OK, ALERT and CANCEL messages).

7.2.12. Measurement attributes

These attributes contain the measurement procedure and the results of the quality measurements.

Measurement parameters are included using the session attribute "measurement". The first measurement parameter is the procedure. Q4S provides a "default" procedure for measurements, but others like RTP/RTCP might be used and defined later. This draft will only define and explain the "default" procedure.

In the initial client request a set of measurement procedures can be sent to the server for negotiation. One measurement procedure line MUST be included in the SDP message for each proposed method. The server MUST answer with only one line with the chosen procedure.

For each procedure, a set of values of parameters separated by "," can be included in the same attribute line. The amount and type of parameters depends on the procedure type.

In the following example the "default" procedure type is chosen:

a=measurement:procedure default(50/50,75/75,5000,40/80,100/256)

In the "default" procedure, the meaning of these parameters is:

o The first parameter is the interval of time (in milliseconds)
   between PING requests during the negotiation phase. Uplink and
   downlink values from the client's point of view are separated
   by "/". This allows having different responsiveness values
   depending on the control resources used in each direction.

o The second parameter is the time interval (in milliseconds)
   between PING requests during the continuity phase. Uplink and
   downlink values are separated by "/". This allows having two
   different responsiveness values depending on the control
   resources used in each direction.

o The third parameter is the time interval to be used to measure
   bandwidth during the negotiation phase..

o The fourth parameter indicates the window size for jitter and
   latency calculations. Uplink and downlink values are separated
   by "/".

o The fifth parameter indicates the window size for packet loss
   calculations. Uplink and downlink values are separated by "/".

There are four more measurement attributes:

a=measurement:latency 45
a=measurement:jitter 3/12
a=measurement:bandwidth 200/9800
a=measurement:packetloss 0.00/1.00

The latency, jitter, bandwidth and packetloss measurement attributes
contain the values measured for each of these quality parameters in
uplink and downlink directions. Notice that latency is considered
equal for uplink and downlink directions. Quality parameter values
in these measurement attributes provide a snapshot of the quality
reached and MUST only be included in Q4S-ALERT messages in the SDP
body such that they can be protected from malicious attacks as these
alerts include a signature of the SDP body in the header. The rest
of messages will include the measured values in the Measurements
header.

7.3. Measurements

This section describes the way quality parameters are measured as defined by the "default" procedure. Measurements MUST be taken for any quality parameter with constraints, that is, specified in the SDP attributes with non-zero values. For non-present attributes measurements MAY be omitted.

7.3.1. Latency

Latency measurements will be performed if the latency attribute and/or the application latency attribute are present and with non-zero values.

Q4S defines a PING method in order to exchange packets between the client and the server. Based on this PING exchange the client and the server are able to calculate the round trip time (RTT). The RTT is the sum of downlink latency (normally named "reverse latency") and uplink latency (normally named "forward latency").

At least 255 samples of RTT MUST be taken by the client and server. As the forward and reverse latencies are impossible to measure, client and server will assume that both latencies are identical (symmetric network assumption). The latency will therefore be calculated as the statistical median value of all the RTT samples divided by 2.

7.3.2. Jitter

Jitter measurements will be performed if the jitter attribute and/or the application jitter attribute are present and with non-zero values.

The jitter can be calculated independently by the client and by the server. The downlink jitter is calculated by the client taking into account the time interval between PING requests as defined by the measurement procedure attribute in the first or second parameter depending on the Q4S protocol phase. The client and the server MUST send these PING requests at the specified intervals. The client measures the downlink jitter whereas the server measures the uplink jitter. Note that PING responses are not taken into account when calculating jitter values.

Every time a PING request message is received by an endpoint (either server or client), the corresponding jitter value is updated using

the Statistical Jitter value calculated on the first 255 packets received using the arithmetic mean of the absolute values of elapsed times.

Each endpoint sends a PING periodically with a fixed interval, each value of "elapsed time" (ET) should be very close to this interval. If a PING message is lost, the elapsed time value is doubled. Identifying lost PING messages, however, is not an issue because all PING messages are labeled with a Sequence-Number header. Therefore the receiver can discard this elapsed time value.

In order to have the first jitter sample, the receiver MUST wait until it receives 3 PING requests, because each ET is the time between two PINGs and a Jitter needs at least two ET.

The client measures the values of RTT and downlink jitter and the server measures RTT and uplink jitter, but all measurements are shared with the counterpart by means of "Measurements" header of PING message.

   7.3.3. Bandwidth

Bandwidth measurements will be performed if the bandwidth attribute and/or the application bandwidth attribute is present and with non-zero values.

In order to measure the available bandwidth, both the client and the server MUST start sending BWIDTH messages simultaneously using the UDP control ports exchanged during the handshake phase in the SDP message, at the needed rate to verify the availability of the bandwidth constraint in each direction using messages of 1 Kbyte or more in length. The messages are sent during the period of time defined in the third parameter of the SDP measurement default procedure attribute in millisecond units.

a=measurement:procedure default(50/50,75/75,5000,256/256,256/256)

```
+-----------------------------------------------------+
|             Rate                                    |
|              A                                      |
|              |                                      |
|downlink rate-|------------------+ <-- traffic       |
|              |                  |     sent by       |
|              |                  |     server        |
|              |                  |                   |
|              |                  |                   |
|              |                  |                   |
|              |                  |                   |
|              |                  |                   |
|              |                  |                   |
|              |                  |                   |
|              |                  |                   |
|   uplink rate-|------------------+ <-- traffic      |
|              |                  |     sent by       |
|              |                  |     client        |
|              |                  |                   |
|              |---|---|---|---|---|----> time        |
|              0   1   2   3   4   5     (sec.)        |
|                                                     |
+-----------------------------------------------------+
```
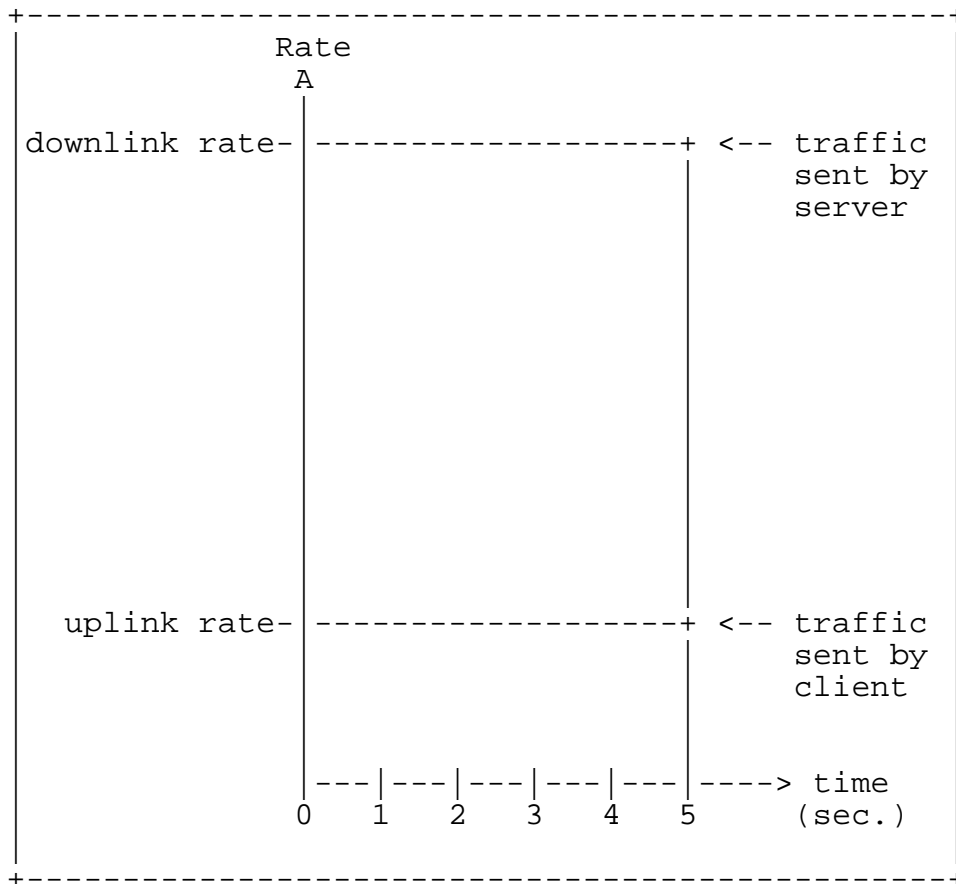
Figure 5 Bandwidth and packet loss measurements.

The goal of these measurements is not to identify the available bandwidth of the communication path but to determine if the required bandwidth is available, meeting the application's constraints. Therefore, the requested bandwidth MUST be measured sending only the highest bit rate required by the network:bandwidth and application:bandwidth attributes.

When measuring bandwidth, all BWIDTH requests sent MUST be 1 kilobyte in length (UDP payload length), and MUST include a Sequence-Number header with a sequential number starting at 0. The Sequence-Number MUST be incremented by 1 with each BWIDTH packet sent. If any measurement stage needs to be repeated, the sequence number MUST start at zero again. BWIDTH requests MUST NOT be answered. Examples:

Client message:
===========================
        BWIDTH q4s://www.example.com Q4S/1.0
        User-Agent: q4s-ua-experimental-1.0
        Session-Id: 53655765
        Sequence-Number: 0
        Content-Type: text
        Content-Length: XXXX
        Measurements: l=22, j=10, pl=0.00, bw=3000

        aaaaaaaaaaaaaa (to complete 1024 bytes UDP payload length)
===========================

The client MUST send BWIDTH packets to the server to allow the
server to measure the uplink bandwidth. The server MUST send BWIDTH
packets to the client to allow the client to measure the downlink
bandwidth.

Server message:
===========================
        BWIDTH q4s://www.example.com Q4S/1.0
        Session-Id: 53655765
        Sequence-Number: 0
        Content-Type: text
        Content-Length: XXXX
        Measurements: l=22, j=7, pl=0.00, bw=200

        aaaaaaaaaaaaaa (to complete 1024 bytes UDP payload length)
===========================

   7.3.4. Packet loss

Packet loss and bandwidth are measured simultaneously using the
BWIDTH packets sent by both the client and the server. Because the
BWIDTH packets contain a Sequence-Number header incremented
sequentially with each sent packet, lost packets can be easily
identified. The lost packets have to be counted during the
measurement time.

7.4. Handshake Phase

The first phase consists of a Q4S BEGIN method issued from the
client to the server.

The first Q4S message MUST have a special URI (RFC 3986 [4]), which
forces the use of the Q4S protocol if it is implemented in a
standard web browser.

This URI, named "Contact URI", is used to request the start of a
session. Its scheme MUST be:

    "q4s:" "//" host [":" port] [path["?" query]

Optionally, the client can send the desired quality parameters
enclosed in the body of the message as an SDP document. The server
MAY take them into account when building the answer message with the
final values in the SDP body, following a request / response schema
(RFC 3464 [5]).

If the request is accepted, the server MUST answer it with a Q4S 200
OK message, which MUST contain an SDP body (RFC 4566 [2]) with the
assigned session id (embedded in the "o" SDP parameter), the IP
addresses to be used, the flow ports to be used, the measurement
procedure to be followed and information about the required quality
constraints. Additionally, the alerting-mode and alert-pause time
parameters may be included. Q4S responses should use the protocol
designator "Q4S/1.0".

After these two messages are exchanged, the first phase is
completed. The quality parameter thresholds have been sent to the
client. The next step is to measure the actual quality of the
communication path between the client and the server and alert if
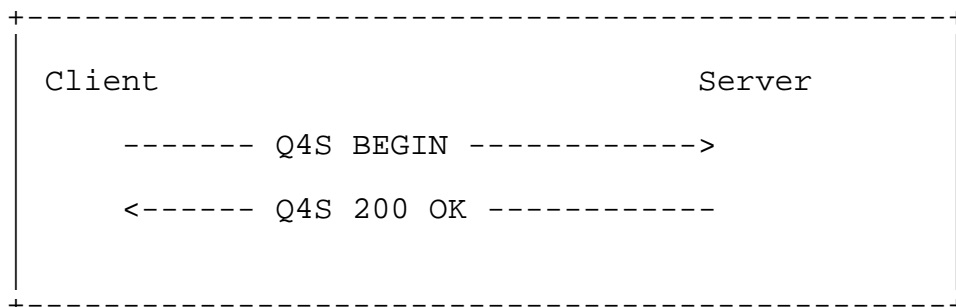the SLA is being violated.

```
+-------------------------------------------------+
|                                                 |
|  Client                             Server      |
|                                                 |
|     ------- Q4S BEGIN ------------>             |
|                                                 |
|     <------ Q4S 200 OK ------------             |
|                                                 |
|                                                 |
+-------------------------------------------------+
```

                     Figure 6 Handshake phase.

Example of Client Request and Server Answer:

Client Request:
```
========================
BEGIN q4s://www.example.com Q4S/1.0
Content-Type: application/sdp
User-Agent: q4s-ua-experimental-1.0
Content-Length: 142

(SDP not shown)
========================
```

Server Answer:
```
========================
Q4S/1.0 200 OK
Date: Mon, 10 Jun 2010 10:00:01 GMT
Content-Type: application/sdp
Expires: 3000
Signature: 6ec1ba40e2adf2d783de530ae254acd4f3477ac4
Content-Length: 131

(SDP not shown)
========================
```

The headers used are explained in section 4.3.

7.5. Negotiation phase

The negotiation phase is in charge of measuring the quality
parameters and verifying that the communication paths meet the
required quality constraints on both directions as specified in the
SDP body.

The measured parameters will be compared with the application and
network constraints specified in the SDP body. If the quality
session is compliant with all the quality constraints the
application can start. If network quality constraints are met, but
application quality constraints are not, the application will be
notified of those parameters such that it can take action by, for
example, starting with lower quality or reducing quality during
execution.

If the network constraints are not met, a higher quality service
level will be demanded through a Q4S-ALERT method triggered by the
server. After receiving the same Q4S-ALERT from the counterpart, no

other alert will be triggered during the "network-alert-pause" in
order to allow the network to react, but measurements will continue
to be taken to achieve early detection of improved network
conditions and a fast application start. If after several
measurement cycles, the network constraints cannot be met the
quality session is terminated.

The steps to be taken in this phase depend on the measurement
procedure exchanged during the handshake phase. This document only
describes the "default" procedure, but others can be used, like
RTP/RTCP (RFC 3550 [10]).

Measurements of latency and jitter are done calculating the
differences in arrival times of packets and can be achieved with
little bandwidth consumption. The bandwidth measurement, on the
other hand, involves higher bandwidth consumption in both directions
(uplink and downlink).

To avoid wasting unnecessary network resources these two types of
measurements will be performed in two separate stages. If the
required latencies and jitters cannot be reached, it makes no sense
to waste network resources measuring bandwidth. In addition, if
achieving the required latency and jitter thresholds implies
upgrading the quality session level, the chance of obtaining
compliant bandwidth measurements without retries is higher, saving
network traffic again. Therefore, the default procedure, determines
that the measurements are taken in two stages:

   o Stage 0: Measurement of latencies, jitters and packet loss

   o Stage 1: Measurement of bandwidths and packet loss

Notice that packet loss can be measured in both stages, as all
messages exchanged include a sequence-number header that allows for
easy packet loss detection.

The client starts the negotiation phase sending a READY request
using the TCP Q4S ports defined in the SDP. This READY request
includes a "Stage" header that indicates the measurement stage.

If either jitter, latency or both are specified, the negotiation
phase begins with the measurement of latencies and jitters (stage
0). If none of those attributes are specified, stage 0 is skipped.

   7.5.1. Stage 0: Measurement of latencies and jitters

   The Stage 0 MUST start with a synchronization message exchange
   initiated with the client's READY message.

   Client request, READY message:
   =========================
         READY q4s://www.example.com Q4S/1.0
         Stage: 0
         Session-Id: 53655765
         User-Agent: q4s-ua-experimental-1.0
         Content-Length: 0
   =========================

   Server Response:
   =========================
     Q4S/1.0 200 OK
         Session-Id: 53655765
         Stage:0
         Content-Length: 0
   =========================

   This triggers the exchange of a sequence of PING requests and
   responses that will lead to the calculation of RTT (latency), jitter
   and packet loss.

   After receiving 200 OK, the client must send the first PING message
   and the server will wait to send PINGs until the reception of this
   first client PING.

   Client and server MUST send PING requests to each other. The
   Sequence-Number header of the first PING MUST be set to 0. Client
   and server will manage their own sequence numbers.

```
+---------------------------------------------------+
|                                                   |
|   Client                                Server    |
|                                                   |
|         --------- Q4S READY 0 --------->          |
|         <------- Q4S 200 OK -----------           |
|                                                   |
|         --------- Q4S PING ------------>          |
|         <------- Q4S 200 OK -----------           |
|         <------- Q4S PING -------------           |
|          ------- Q4S 200 OK ----------->          |
|         --------- Q4S PING ------------>          |
|         <------- Q4S PING -------------           |
|         --------- Q4S 200 OK ----------->         |
|         <------- Q4S 200 OK -----------           |
|                        ...                        |
|                                                   |
+---------------------------------------------------+
```
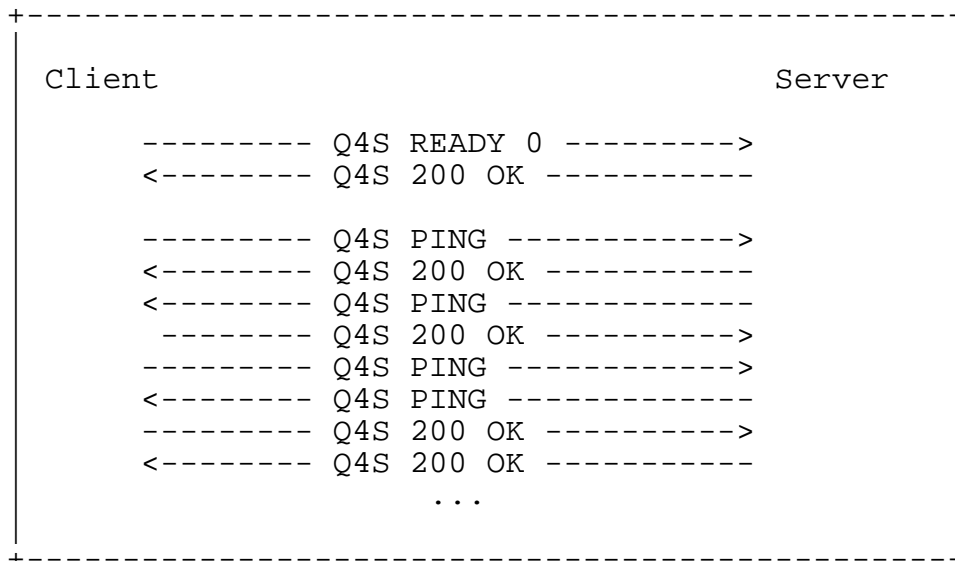
        Figure 7 Simultaneous exchange of PING request and responses.

    This is an example of the PING request sent from the client and the
    server's response:

    Client Request:
    ========================
            PING q4s://www.example.com Q4S/1.0
            Session-Id: 53655765
            Sequence-Number: 0
            User-Agent: q4s-ua-experimental-1.0
            Measurements: l=22, j=12, pl=0.20, bw=
            Content-Length: 0
    ========================

    Server Response:
    ========================
      Q4S/1.0 200 OK
            Session-Id: 53655765
            Sequence-Number: 0
            Content-Length: 0
    ========================

    The function of the PING method is similar to the ICMP echo request
    message. The server MUST answer as soon as it receives the message.

Both endpoints MUST send Q4S PING messages with the periodicity
specified in the first parameter of SDP measurement procedure
attribute, using always the same UDP ports and incrementing the
Sequence-Number with each message.

In the following example, the SDP measurement procedure attribute,
this value is 50 milliseconds (from the client to the server) and
60ms (from the server to the client).

a=measurement:procedure default(50/60,50/50,5000,256/256,256/256)

They MUST NOT wait for a response to send the next PING request. The
"Sequence-Number" header value is incremented sequentially and MUST
start at zero. If this stage is repeated, the initial Sequence-
Number MUST start at zero again.

All PING requests MUST contain a "Measurements" header, with the
values of the latency, jitter and packet loss measured by each
entity up to that moment. The client will send its measurements to
the server and the server his measurements to the client. Example:

   Measurements: l=22, j=13, pl=0.10, bw=

Where l stands for latency, j for jitter, pl for packetloss and bw
for bandwidth. The bandwidth value is omitted, as it is not measured
at this stage.

Optionally the PING request can include a "Timestamp" header, with
the time in which the message has been sent. In case the header is
present, the server MUST include the header in the response without
changing the value.

A minimum number of PING messages MUST be exchanged in order to be
able to measure latency, jitter and packet-loss with certain
accuracy (at least 256 samples are recommended to get a accurate
packet loss measurement). Both the client and the server calculate
the respective measured parameter values. The mechanisms to
calculate the different parameters are described in section 7.3.

At the end of this stage 0, there are three possibilities:

   o The network and application latency, jitter and packet loss
      constraints are reached in both directions

   o The network latency, jitter and packet loss constraints are
      reached in both directions but one or more parameters fall
      below the application constraints in one or both directions.

   o   The network latency, jitter and packet loss constraints are not
       reached in one or both directions

   In the first and second cases, Stage 0 is finished. Client and
   server are ready for Stage 1: bandwidth and packet loss measurement.
   The client moves to stage 1 by sending a READY message including the
   header "Stage: 1".

   If the network and application bandwidth constraints are empty or
   with value zero, the negotiation phase MUST terminate and both
   client and server may initiate the Continuity Phase. In this case
   client moves to Continuity phase by sending a READY message
   including the header "Stage: 2". But notice that if only application
   constraints were violated (second case), the answer to the READY
   message for Continuity phase MAY include optionally the URI for
   triggering the application using a simple HTTP based mechanism based
   on an optional "Trigger-URI" header. This URI may include specific
   information for the application to start under degraded network
   conditions.

   The third case, in which one or more network constraints have not
   been met, is detailed in section 7.5.4. Network constraints not
   reached.

   7.5.2. Stage 1: Measurement of bandwidth and packet loss

   This stage begins in a similar way to stage 0, sending a READY
   request over TCP. This READY message "Stage" header value is 1. The
   server answers with a Q4S 200 OK message to synchronize the
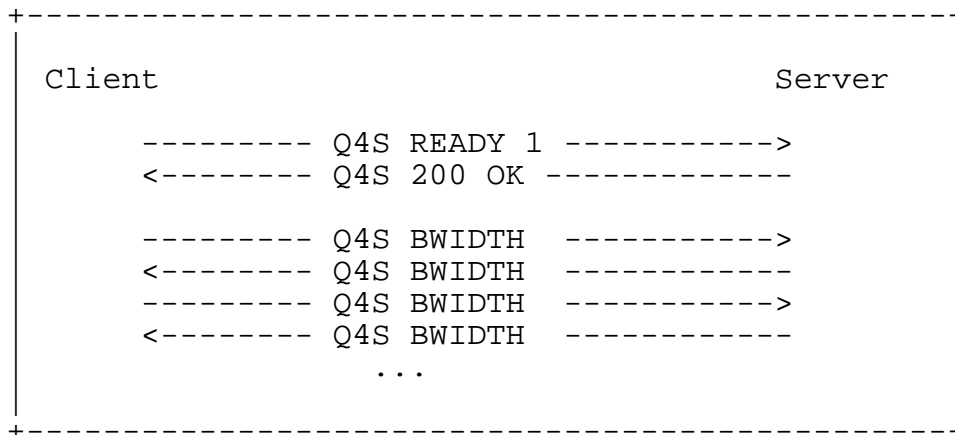   initiation of the measurements.

```
+----------------------------------------------------+
|                                                    |
|  Client                                  Server    |
|                                                    |
|      --------- Q4S READY 1 ----------->            |
|      <-------- Q4S 200 OK -------------             |
|                                                    |
|      --------- Q4S BWIDTH  ----------->            |
|      <-------- Q4S BWIDTH  ------------             |
|      --------- Q4S BWIDTH  ----------->            |
|      <-------- Q4S BWIDTH  ------------             |
|                    ...                             |
|                                                    |
+----------------------------------------------------+
```
         Figure 8 Starting bandwidth and packet loss measurement

Client Request:
========================
        READY q4s://www.example.com Q4S/1.0
        User-Agent: q4s-ua-experimental-1.0
        Stage: 1
        Session-Id: 53655765
        Content-Length: 0

========================

Server Response:
========================
  Q4S/1.0 200 OK
        Session-Id: 53655765
        Stage: 1
        Content-Length: 0

========================

Just after receiving the 200 OK, both the client and the server MUST
start sending BWIDTH messages simultaneously using the UDP q4s
ports. Section 7.3.3 describes the bandwidth measurement in detail.

At the end of this stage 1, there are three possibilities:

o The network and application bandwidth and packet loss
   constraints are reached in both directions

o The network bandwidth and packet loss constraints are reached
   in both directions but one or more parameters fall below the
   application constraints in one or both directions.

o The network bandwidth and packet loss constraints are not
   reached in one both directions.

In the first and second cases, Stage 1 is finished. Client and
server are ready for Continuity phase. The client moves to this
phase by sending a READY message including the header "Stage: 2".
The server answer MUST be 200 OK, and MAY include optionally the URI
for triggering the application using a simple HTTP based mechanism
based on an optional "Trigger-URI" header. This URI may include
specific information for the application to start under degraded
network conditions if application constraints are not met.

```
+--------------------------------------------------+
|                                                  |
|  Client                                 Server   |
|                                                  |
|     ---------   Q4S READY 2 -------------->      |
|     <---- Q4S 200 OK with trigger URI-----       |
|                                                  |
|     ---------    HTTP GET --------------->       |
|                                                  |
|           (Application starts)                   |
|                                                  |
+--------------------------------------------------+
```
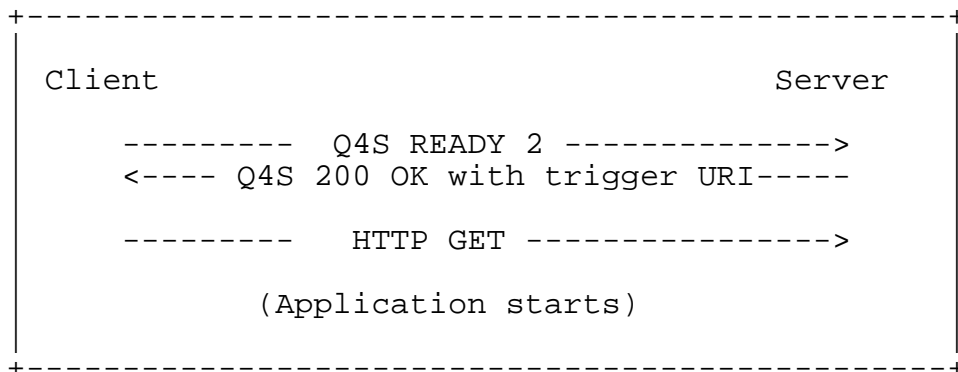
Figure 9 Trigger the application using HTTP URI

In the second case of stage 0 and/or stage 1 in which network
constrains have been reached but application constraints have not
the behavior to follow is described in section 7.5.3 Application
constraints not reached.

```
Client Request:
=========================
READY q4s://www.example.com Q4S/1.0
User-Agent: q4s-ua-experimental-1.0
Stage: 2
Session-Id: 53655765
Content-Length: 0

=========================


Server Answer:
=========================
Q4S/1.0 200 OK
Date: Mon, 10 Jun 2010 10:00:01 GMT
Session-Id: 53655765
Trigger-URI: http://www.example.com/app_start
Expires: 3000
Content-Type: application/sdp
Signature: 6ec1ba40e2adf2d783de530ae254acd4f3477ac4
Content-Length: 131

(SDP not shown)
=========================
```

If the "Trigger-URI" header is present, the client SHOULD send an
HTTP request to this URI.

The third case, with violated network constraints is explained in
7.5.4 Network constraints not reached.

7.5.3. Application constraints not reached

After finishing Stage 1 of the Negotiation phase, the client and the
server have each others measured parameter values as these have been
exchanged in the "Measurements" headers of the PING and BWIDTH
messages. If there is one or more parameters that do not comply with
the uplink or downlink application constraints required both the
server and the client are aware of it.

As the quality parameter values meet the network constraints the
server MUST answer the client's READY request to enter the
Continuity phase with a 200 OK message that may include a "Trigger-
URI" header to initiate the application. This URI may include a
special path to indicate the application that one or more quality
parameters are below the application thresholds. The application
SHOULD take this into account and start with specific configuration
options that allow it to work under the circumstances. The
particular measures taken by the application are beyond the scope of
this draft.

```
Server's 200 Answer:
=========================
Q4S/1.0 200 OK
Date: Mon, 10 Jun 2010 10:00:01 GMT
Session-Id: 53655765
Content-Type: application/sdp
Expires: 3000
Trigger-URI: http://www.example.com/app_start?bandwidth=x
Signature: 6ec1ba40e2adf2d783de530ae254acd4f3477ac4
Content-Length: 131

(SDP not shown)
=========================
```

Measurements of jitter, latency and packetloss continue during the
Continuity phase described in section 7.6 exchanging PING requests
that contain the "Measurements" header. If during these exchanges
the measured values drop below the application thresholds, the
client Q4S stack SHOULD notify the client application and the server
Q4S stack SHOULD notify the server application. The notification
consists of a message to the application and is implementation
dependent and out of scope of this draft. Q4S stacks SHOULD also
notify the application if measured values recover over time above
the application thresholds.

7.5.4. Network constraints not reached

If there is any parameter that does not comply with the network
uplink or downlink quality constraints specified in the SDP message,
two scenarios are possible depending on the specified alerting-mode
(if not present, default value is "policy-server" alerting mode):

a) q4s-aware-network alerting mode: the server MUST send a Q4S-ALERT
   message to the client including the digital signature header, and
   the client MUST answer with the same Q4S-ALERT message. The
   Signature header contains the signed hash value of the SDP body
   in order to protect all the SDP the data and therefore it MUST
   contain the measurement parameters in the body.

```
Server request
==========================
Q4S-ALERT q4s://www.example.com Q4S/1.0
Host: www.example.com
User-Agent: q4s-ua-experimental-1.0
Session-Id: 53655765
Content-Type: application/sdp
Content-Length: 142

v=0
o=q4s-UA 53655765 2353687637 IN IP4 192.0.2.33
s=Q4S
i=Q4S parameters
t=0 0
a=qos-level:1/2
a=alerting-mode: q4s-aware-network
a=network-alert-pause:5000
a=app-alert-pause:0
a=public-address:client IP4 80.0.2.33
a=public-address:server IP4 198.51.100.58
a=network:latency:40
a=network:jitter:10/10
a=network:bandwidth:20/6000
a=network:packetloss:0.50/0.50
a=application:latency:35
a=applicacion:jitter:8/8
a=applicacion:bandwidth:25/7000
a=applicacion:packetloss:0.30/0.30
a=flow:app downlink TCP/10000-20000
a=flow:app uplink TCP/56000
a=flow:q4s downlink UDP/55000
a=flow:q4s downlink TCP/55001
a=flow:q4s uplink UDP/56000
a=flow:q4s uplink TCP/56001
a=measurement:procedure default(50/50,50/50,5000,256/256,256/256)
a=measurement:latency 30
a=measurement:jitter 6/4
a=measurement:bandwidth 200/4000
a=measurement:packetloss 0.20/0.33
==========================
```

At this point, both client and server keep on measuring but without
sending new Q4S alerts during the "network-alert-pause"
milliseconds.

   b) Policy-server alerting mode: the server MUST send a Q4S-ALERT
      message to the network policy server, and the policy server MUST
      answer with the same Q4S-ALERT request message.

```
Server request
==========================
Q4S-ALERT q4s://www.example.com Q4S/1.0
Host: www.example.com
User-Agent: q4s-ua-experimental-1.0
Session-Id: 53655765
Signature: 6ec1ba40e2adf2d783de530ae254acd4f3477ac4
Content-Type: application/sdp
Content-Length: 142

v=0
o=q4s-UA 53655765 2353687637 IN IP4 192.0.2.33
s=Q4S
i=Q4S parameters
t=0 0
a=qos-level:1/2
a=alerting-mode: policy-server
a=network-alert-pause:5000
a=app-alert-pause:25000
a=public-address:client IP4 192.0.2.33
a=public-address:server IP4 198.51.100.58
a=network:latency:40
a=network:jitter:10/10
a=network:bandwidth:20/6000
a=network:packetloss:0.50/0.50
a=application:latency:35
a=applicacion:jitter:8/8
a=applicacion:bandwidth:25/7000
a=applicacion:packetloss:0.30/0.30
a=flow:app downlink TCP/10000-20000
a=flow:app uplink TCP/56000
a=flow:q4s downlink UDP/55000
a=flow:q4s downlink TCP/55001
a=flow:q4s uplink UDP/56000
a=flow:q4s uplink TCP/56001
a=measurement:procedure default(50/50,50/50,5000,256/256,256/256)
a=measurement:latency 30
a=measurement:jitter 6/4
a=measurement:bandwidth 200/4000
a=measurement:packetloss 0.20/0.30
==========================
```

At this point, during Negotiation phase, both client and server keep
on measuring without sending new Q4S alerts during the "network-
alert-pause" milliseconds specified in the SDP. This way, both
client and server will detect any improvement in network conditions

as soon as the network reacts. The application can start as soon as the number of measurements indicated in the measurement procedure attribute indicates that the quality parameters are met.

Same applies to Continuity phase: the measurement dialog between client and server must not be interrupted by any possible ALERT message.

7.5.4.1. Policy server role

A network policy server in charge of Q4S ALERT processing may implement all or some of these features (but not exclusive to):

   o Server validation in terms of quality constraints.

   o Authentication (Signature validation) and security (block malicious clients)

   o Policy rules (following rules are only examples):

      - Maximum quality level allowed for the ACP

      - Time bands allowed for providing quality sessions

      - Number of simultaneous quality sessions allowed

      - Maximum time used by allowed quality sessions

      - Etc.

If any of the policy rules fail, a Q4S-ALERT message must be answered by a 6XX error, indicating the cause.

7.5.5. QoS Level changes

If any constraint was violated, client or server (depending on alerting mode) MAY trigger a Q4S-ALERT asking for higher qos-level attribute. The maximum qos-level allowed is 9, both uplink and downlink.

If the qos-level has reached the maximum value for downlink or uplink without matching the constraints, then a CANCEL request MUST be sent by the client using the TCP port determined in the handshake

phase in order to release the session. In reaction to the reception
of the CANCEL request, the server MUST send a CANCEL request too. If
no CANCEL request is received, the expiration time cancels the
session at server side.

    Client Request:
    =========================
    CANCEL q4s://www.example.com Q4S/1.0
    User-Agent: q4s-ua-experimental-1.0
    Session-Id: 53655765
    Content-Type: application/sdp
    Content-Length: 142

    (SDP not shown)
    =========================

    Server Request in reaction to Client Request:
    =========================
    CANCEL q4s://www.example.com Q4S/1.0
    Session-Id: 53655765
    Expires: 0
    Content-Type: application/sdp
    Signature: 6ec1ba40e2adf2d783de530ae254acd4f3477ac4
    Content-Length: 131

    (SDP not shown)
    =========================

  7.6. Continuity phase

  During the negotiation phase, latency, jitter, bandwidth and packet
  loss have been measured. During the continuity phase bandwidth will
  not be measured again because bandwidth measurements may disturb
  application performance.

  This phase is supposed to be executed at the same time as the real
  time application is being used.

  This draft only covers the default procedure. The continuity
  operation with default procedure is based on a sliding window of
  samples. The number of samples involved in the sliding window may be

different for jitter and latency than for packet-loss calculations
according to the fifth and sixth parameters of the measurement
procedure attribute. In this example, the jitter and latency sliding
window comprises 40 samples whereas the size of the packet-loss
sliding window is 100 samples:

a=measurement:procedure default(50/50,75/75,5000,40/40,100/100)

In addition, the sizes of these windows are configurable per
direction: uplink and downlink values may differ.

PING requests are sent continuously (in both directions) and when
the Sequence-Number header reaches the maximum value, the client
continues sending PING messages with the Sequence-Number header
starting again at zero. When the server PING Sequence-Number header
reaches the maximum value, it does the same, starting again from
zero.

On the client side, the measured values of downlink jitter, downlink
packet loss and latency are calculated using the last samples,
discarding older ones, in a sliding window schema.

```
+---------------------------------------------------+
|                                                   |
|  55 56 57 . . . 253 254 255 0 1 2 . . . 55 56     |
|          A                                 A      |
|          |                                 |      |
|          +---------------------------------+      |
|                                                   |
+---------------------------------------------------+
```
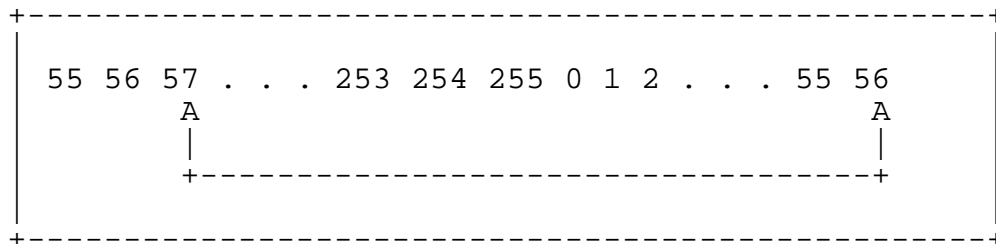
Figure 10   Sliding samples window

Only if the server detects that the measured values (downlink or
uplink jitter, packet loss or latency) are not reaching the network
constraints, a Q4S ALERT is triggered and sent to the client.

In q4s-aware-network alerting mode, if the client receives a Q4S
ALERT message, it MUST answer sending the Q4S ALERT request message
back to the server including the SDP (with its corresponding digital
signature).

Both client and server will keep performing measurements but no
other Q4S ALERT message MUST be sent during "network-alert-pause"
milliseconds. The operations needed to act on the network and the
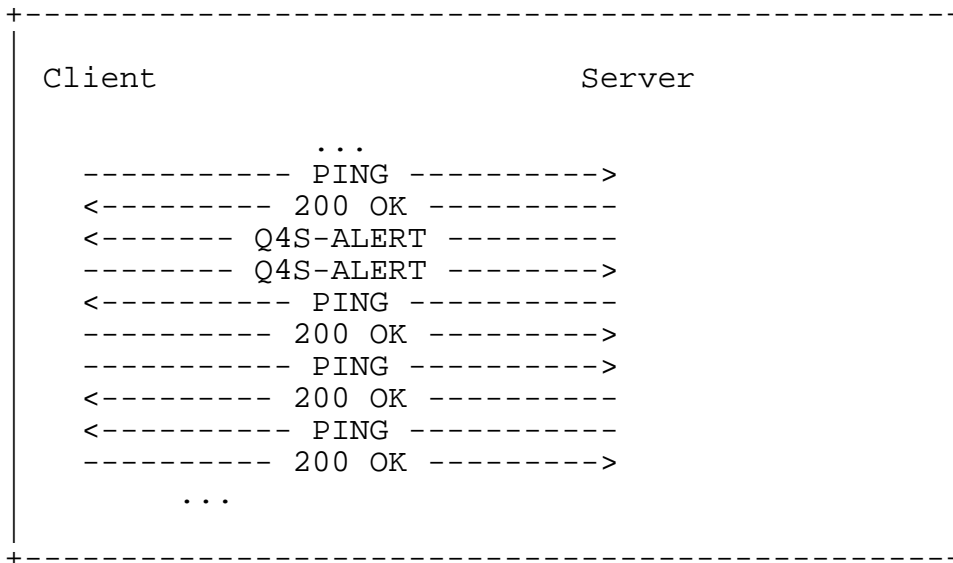agents in charge of them are out of scope of this draft.

```
+-------------------------------------------------+
|                                                 |
|   Client                         Server         |
|                                                 |
|                 ...                             |
|     ----------- PING ---------->                |
|     <--------- 200 OK ----------                |
|     <------- Q4S-ALERT ---------                |
|     -------- Q4S-ALERT -------->                |
|     <--------- PING -----------                 |
|     --------- 200 OK --------->                 |
|     ---------- PING ---------->                 |
|     <--------- 200 OK ----------                |
|     <--------- PING -----------                 |
|     --------- 200 OK --------->                 |
|          ...                                    |
|                                                 |
+-------------------------------------------------+
```

        Figure 11   Continuity in q4s-aware-network alerting mode

   In policy-server alerting mode, if the server detects that the
   measured values (downlink or uplink jitter, packet loss or latency)
   are not reaching the network constraints, a Q4S ALERT is triggered
   and sent to the policy server.

   The policy server MUST answer sending the Q4S ALERT request message
   back to the server including the SDP (with its corresponding digital
   signature), or with a 6xx error response message.

   The measurement dialog between the client and the server must not be
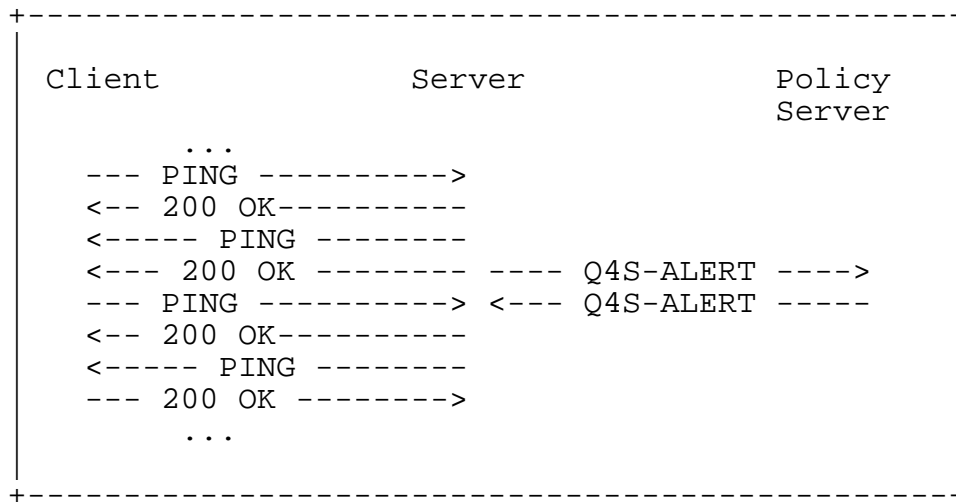   interrupted by any possible ALERT message.

```
+-----------------------------------------------+
|                                               |
|  Client             Server          Policy    |
|                                     Server     |
|                                               |
|        ...                                    |
|    --- PING ---------->                       |
|    <-- 200 OK----------                        |
|    <----- PING --------                        |
|    <--- 200 OK -------- ---- Q4S-ALERT ---->   |
|    --- PING ----------> <--- Q4S-ALERT -----   |
|    <-- 200 OK----------                        |
|    <----- PING --------                        |
|    --- 200 OK -------->                        |
|         ...                                   |
|                                               |
+-----------------------------------------------+
```

Figure 12   Continuity in policy-server alerting mode

7.7. Termination Phase

The Termination phase is not a phase itself but an end point for the established Q4S session. This phase is reached in the following cases:

* A CANCEL message has been received. The client sends a CANCEL message due to the impossibility of the network to meet the required quality constraints. The client and server application will be notified by the respective Q4S stack.

* Session expires: if after the Expires time no client or server activity is detected, that end cancels the session.

* A BEGIN message has been received by the server. The pre-existing Q4S quality session is cancelled and a new session will be initiated.

The meaning of Termination phase in terms of release of resources or accounting is application dependent and out of scope of the Q4S protocol.

In policy-server alerting mode, Q4S CANCEL messages must be forwarded from the server to the network policy server in order to release possible assigned resources for the session.

7.8. Dynamic constraints and flows

Depending on the nature of the application, the constraints to be reached may evolve, changing some or all constraint values in any direction.

The client MUST be able to deal with this possibility. When the server sends an SDP document attached to a reply (200 OK, or Q4S-ALERT, etc), the client MUST take all the new received values, overriding any previous value in use.

The dynamic changes on the constraints can be as a result of two possibilities:

  o The application communicates to the Q4S server a change in the constraints. In this case the application requirements can evolve and the Q4S server will be aware of them.

  o The application uses TCP flows. In that case, in order to guarantee a constant throughput, the nature of TCP behavior forces the use of a composite constraint function, which depends on RTT, packet loss and window control mechanism implemented in each TCP stack.

TCP throughput can be less than actual bandwidth if the Bandwidth-Delay Product (BDP) is large or if the network suffers from a high packet loss rate. In both cases, TCP congestion control algorithms may result in a suboptimal performance.

Different TCP congestion control implementations like Reno [14], High Speed TCP (RFC 3649 [15]), CUBIC [16], Compound TCP (CTCP [17]), etc. reach different throughputs under the same network conditions of RTT and packet loss. In all cases, depending on the RTT measured value, the Q4S server could change dynamically the packetloss constraints (defined in SDP) in order to make possible to reach a required throughput or vice versa (use packetloss measurement to change dynamically latency constraints).

A general guideline to calculate the packetloss constraint and RTT constraint consists in approximating the throughput using a simplified formula, which should take into account the TCP stack implementation of the receiver, in addition to RTT and packet loss:

        Th= Function(RTT, packet loss, ...)

Then, depending on RTT measured values, set dynamically the packetloss constraint.

It is possible to easily calculate a worst-case boundary for the
Reno algorithm, which should ensure for all algorithms that the
target throughput is actually achieved. Except that, high-speed
algorithms will then have even a larger throughput, if more
bandwidth is available.

For the Reno algorithm, the Mathis' formula may be used [15] for the
upper bound on the throughput:

$$Th <= (MSS/RTT)*(1 / sqrt\{p\})$$

In absence of packet loss, a practical limit for the TCP throughput
is the receiver_window_size divided by the round-trip time. However,
if the TCP implementation uses a window scale option, this limit can
reach the available bandwidth value.

7.9. Qos-level downgrade operation

During the continuity phase it might be desirable to downgrade the
current qos-level SDP parameter.

The strategy to carry out downgrades must include the possibility to
exclude specific app flows from SDP dynamically. The Q4S flows would
be downgraded to allow for measurements on a lower quality level
without interference of the application flows. A Q4S client MUST
allow this kind of SDP modifications by the server.

Periodically (every several minutes, depending on the
implementation) a Q4S-ALERT could be triggered, in which the level
is downgraded for Q4S flows, excluding application flows from the
embedded SDP of that request.

This mechanism allows to measure at lower levels of quality while
application flows continue using a higher qos level value.

   o If the measurements in the lower level meet the constraints,
      then a new Q4S-ALERT to this lower qos-level may be triggered,
      in which the SDP includes the application flows in addition to
      Q4S flows.

   o If the measurements in the lower level do not meet the
      constraints, then a new Q4S-ALERT to the previous qos-level
      MUST be triggered, in which the SDP includes only the Q4S
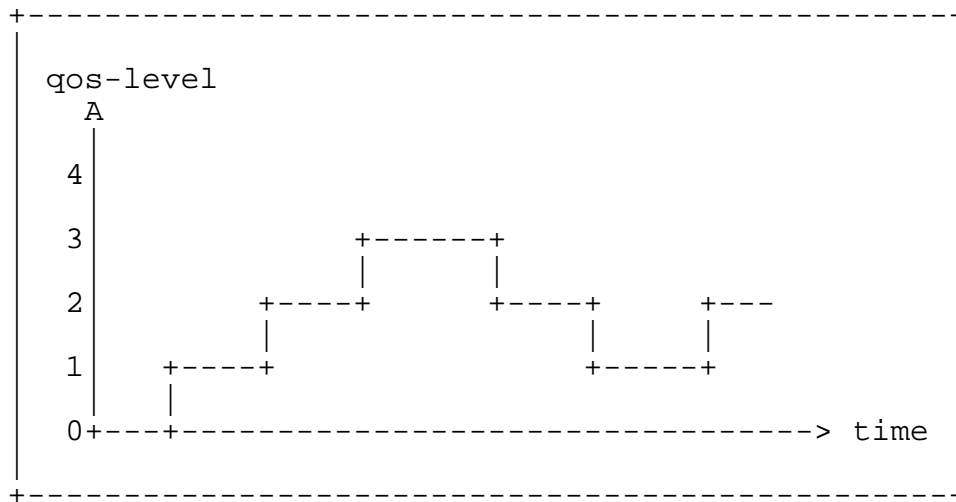   flows.

```
      +------------------------------------------------+
      |                                                |
      |   qos-level                                    |
      |      A                                         |
      |      |                                         |
      |   4  |                                         |
      |      |            +------+                     |
      |   3  |            |      |                     |
      |      |      +----+|      |+----+     +---       |
      |   2  |      |    ||      ||    |     |          |
      |      | +----+    |+------+|    |     |          |
      |   1  | |    |             |    +-----+          |
      |      | |    |             |                     |
      |   0+-+----+-------------------------------> time|
      |      |                                          |
      +------------------------------------------------+
```

Figure 13   Possible evolution of qos-level

This mechanism avoids the risk of disturbing the application, while
the measurements are being run in lower levels. However, this
optional optimization of resources MUST be used carefully.

The chosen period to measure a lower qos level is implementation
dependent. Therefore it is not included as a measurement procedure
parameter. It is recommended to use a large value, such as 20
minutes.

7.10. Sanity check of Quality sessions

A session may finish due to several reasons (client shutdown, client
CANCEL request, constraints not reached, etc), and any session
finished MUST release the assigned resources.

In order to release the assigned server resources for the session,
the "Expires" header indicates the maximum interval of time without
exchanging any Q4S message.

8. General User Agent behavior

   8.1. Roles in peer to peer scenarios

In order to allow peer to peer applications, a Q4S User Agent (UA)
MUST be able to assume both client and server role. The role assumed
depends on who sends the first message.

In a communication between two UAs, the UA that sends the Q4S BEGIN
request in the first place, for starting the handshake phase, shall
assume the client role.

If both UASs send the BEGIN request at the same time, they will wait
for a random time to restart again.

Otherwise, an UA may be configured to act only as server (e.g.,
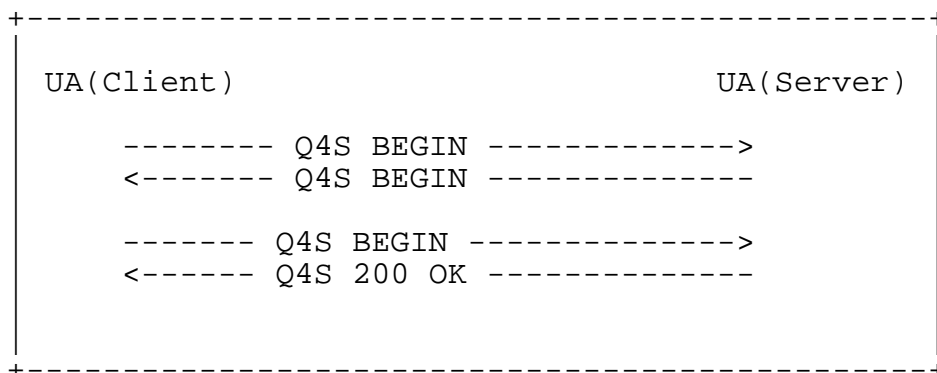content provider's side).

```
+------------------------------------------------+
|                                                |
|  UA(Client)                      UA(Server)    |
|                                                |
|     -------- Q4S BEGIN ------------->          |
|     <------- Q4S BEGIN --------------          |
|                                                |
|     ------- Q4S BEGIN -------------->          |
|     <------ Q4S 200 OK --------------          |
|                                                |
|                                                |
+------------------------------------------------+
```

                   Figure 14   P2P roles.

8.2. Multiple Quality sessions in parallel

A Q4S session is intended to be used for an application. It means
that for using the application, the client MUST establish only one
Q4S session against the server. Indeed, the relation between
session-id and application is 1 to 1.

If a user wants to participate in several independent Q4S sessions
simultaneously against different servers (or against the same
server) can execute different Q4S clients to establish separately
different Q4S sessions but it is not recommended, because:

        The establishment of a new Q4S session may affect other
running applications over other Q4S sessions during bandwidth
measurement.

        If the negotiation phase is executed separately before running
any application, the summation of bandwidth requirements could not
be met when the applications are running in parallel.

8.3. General client behavior

A Q4S Client has different behaviors. We will use letters X,Y,Z to designate each different behavior (follow the letter bullets in the figure below).

   X) When it sends messages over TCP (methods BEGIN, READY, Q4S-ALERT and CANCEL) it behaves strictly like a state machine that sends requests and waits for responses. Depending on the response type it enters in a new state.

When it sends UDP messages (methods PING and BWIDTH), a Q4S client is not strictly a state machine that sends messages and waits for responses because:

   Y) At latency, jitter and packet loss measurement, the PING requests are sent periodically, not after receiving the response to the previous request. In addition, the client MUST answer the PING requests coming from the server, therefore assumes the role of a server.

   Z) At bandwidth and packet loss measurement stage, the client does not expect to receive responses when sending BWIDTH requests to the server. In addition, it MUST receive and process all server messages in order to achieve the downlink measurement.

The Q4S-ALERT and CANCEL may have a conventional answer if an error is produced, otherwise the corresponding answer is formatted as a request message.
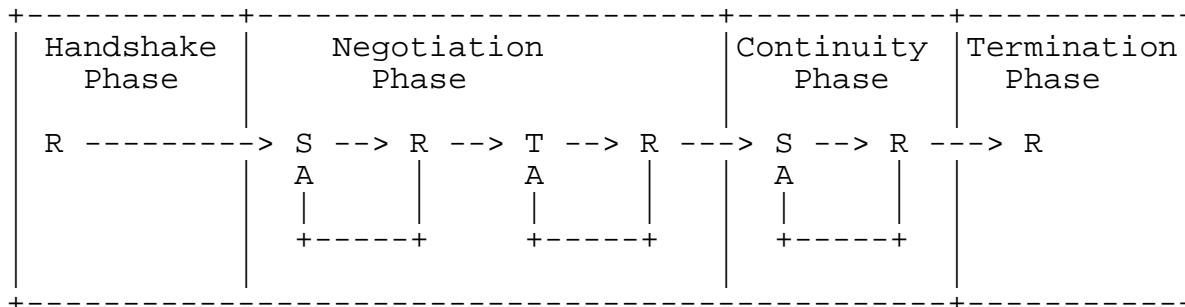
```
+-----------+------------------------+----------+-----------+
| Handshake |      Negotiation       |Continuity|Termination|
|   Phase   |        Phase           |  Phase   |  Phase    |
|           |                        |          |           |
| X --------> Y --> X --> Z --> X ---> Y --> X ---> X        |
|           | A    |     A     |    | A     |   |           |
|           | |    |     |     |    | |     |   |           |
|           | +----+     +----+     | +----+     |           |
|           |                       |            |           |
+--------------------------------------------+-----------+
```
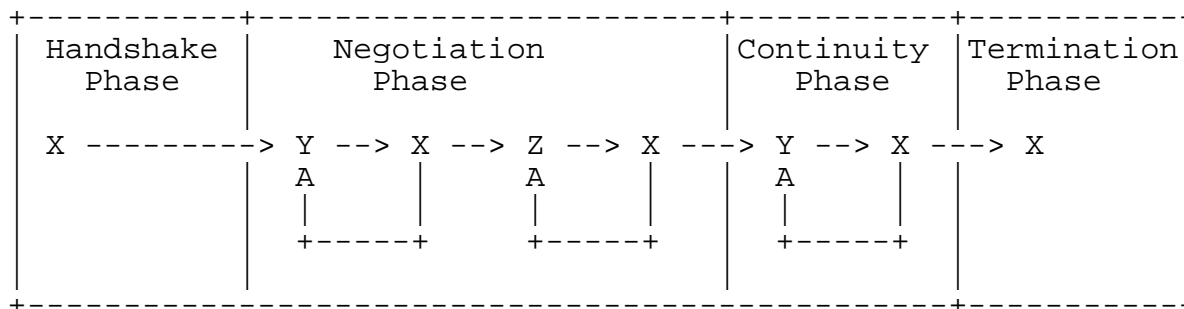
Figure 15   Phases & client behaviors.

8.3.1. Generating requests

A valid Q4S request formulated by a Client MUST, at a minimum,
contain the following header fields:

If no SDP is included: the header Session-Id and Sequence-Number are
mandatory.

If SDP is included: Session-Id is embedded into SDP, therefore the
inclusion of Session-Id header is optional but if present must have
the same value. Measurements are embedded into the SDP only for Q4S-
ALERT messages in order to be signed.

At any time, if the server sends a new SDP with updated values,
client MUST take it into account.

8.4. General server behavior

If a server does not understand a header field in a request (that
is, the header field is not defined in this specification or in any
supported extension), the server MUST ignore that header field and
continue processing the message.

The role of the server is changed at negotiation and continuity
phases, in which server MUST send packets to measure jitter, latency
and bandwidth. Therefore, the different behaviors of server are
(follow the letter bullets in the figure below):

   R) When the client sends messages over TCP (methods BEGIN, READY
   Q4S-ALERT and CANCEL) it behaves strictly like a state machine
   that receives messages and sends responses.

When the client begins to send UDP messages (methods PING and
BWIDTH), a Q4S server is not strictly a state machine that receives
messages and sends responses because:

   S) At latency, jitter and packet loss measurement, the PING
   requests are sent periodically by the client but also by the
   server. In this case the server behaves as a server answering
   client requests but also behaves as a client, sending PING
   requests toward the client and receiving responses.

   T) At bandwidth and packet loss measurement, the server sends
   BWIDTH requests to the client. In addition, it MUST receive and
   process client messages in order to achieve the uplink
   measurement.

The Q4S-ALERT and CANCEL may have a conventional answer if an error
is produced, otherwise the corresponding answer is formatted as a
request message.

   .


```
+-----------+-----------------------+-----------+-----------+
|           |                       |           |           |
| Handshake |     Negotiation       |Continuity |Termination|
|   Phase   |        Phase          |   Phase   |   Phase   |
|           |                       |           |           |
| R --------> S --> R --> T --> R ---> S --> R ---> R        |
|           | A     |     A     |     A     |   |           |
|           | |     |     |     |     |     |   |           |
|           | +-----+     +-----+     | +-----+   |         |
|           |                       |           |           |
+-------------------------------------------------+---------+
```

              Figure 16   Phases & server behaviours.

9. Implementation Recommendations

  9.1. Default client constraints

  To provide a default configuration, it would be good that the client
  had a configurable set of Quality headers in the implementation
  settings menu. Otherwise these quality headers will not be present
  in the first message.

Different business models (out of scope of this proposal) may be achieved: depending on who pays for the quality session, the server can accept certain Client parameters sent in the first message, or force billing parameters on the server side.

9.2. Latency and Jitter measurements

In execution systems, where the timers are not accurate, a recommended approach consists of including the optional header "Timestamp" in the PING request with the time in which the message has been sent. This allows an accurate measurement of the jitter even with no identical intervals of time between PINGs.

9.3. Bandwidth measurements

In programming languages or Operating Systems with timers or limited clock resolution, it is recommended to use an approach based on several intervals to send messages of 1KB, in order to reach the required bandwidth consumption using a rate as close as possible to a constant rate.

For example, if the resolution is 1 millisecond, and the bandwidth to reach is 11Mbps, a good approach consists of sending:

   1 message of 1KB every 1 millisecond +

   1 message of 1KB every 3 milliseconds +

   1 message of 1KB every 23 milliseconds

The number of intervals depends on required bandwidth and accuracy that the programmer wants to achieve.

In execution systems where the timers are not accurate, a recommended approach consists of checking at each interval, the number of packets that should have been sent at this timestamp since origin and send the needed number of packets in order to reach the required bandwidth.

The shorter packets are used, the more constant is the rate of bandwidth measurement. However, this may stress the execution system in charge of receiving and processing packets. As a consequence, some packets may be lost because of stack overflows. To deal with this potential issue, a larger packet is recommended (2KB or more) taking into account the overhead produced by the chunks headers.

9.4. Packet loss measurement resolution

Depending on application nature and network conditions, a packet
loss resolution less than 1% may be needed. In such case, there is
no limit to the number of samples used for this calculation. A
tradeoff between time and resolution should be reached in each case.
For example, in order to have a resolution of 1/10000, the last
10000 samples should be considered in the packetloss measured value.

The problem of this approach is the reliability of old samples. If
the interval used between PING messages is 50ms, then to have a
resolution of 1/1000 it takes 50 seconds and a resolution of 1/10000
takes 500 seconds (more than 8 minutes). The reliability of a packet
loss calculation based on a sliding window of 8 minutes depends on
how fast network conditions evolve.

9.5. Measurements and reactions

Q4S can be used as a mechanism for measure and trigger network
tuning and application level actions (i.e. lowering video bit-rate,
reduce multiplayer interaction speed, etc) in real-time in order to
reach the application constraints, addressing measured possible
network degradation.

9.6. Instability treatments

There are two scenarios in which Q4S can be affected by network
problems: loss of Q4S packets and outlier samples

9.5.1. Loss of control packets

Lost UDP packets (PING or BWIDTH messages) don't cause any problems
for the Q4S state machine, but if TCP packets are lost, some
undesirable consequences could arise.

Q4S does have protection mechanisms to overcome these situations.
Examples:

    o If a BEGIN packet is lost or its corresponding answer, after a
       certain timeout, the client SHOULD resend another BEGIN packet,
       resetting the session

o If a READY packet is lost, after a certain timeout, the client
   SHOULD resend another READY packet.

o If a QOS ALERT request is lost or its corresponding answer,
   after a certain timeout, the originator SHOULD resend another
   Q4S-ALERT packet.

o If CANCEL request is lost or its corresponding answer, after a
   certain timeout, the originator SHOULD resend another CANCEL
   packet.

9.5.2. Outlier samples

Outlier samples are those jitter or latency values far from the
general/average values of most samples.

Hence Q4S default measurement method uses the statistical median
formula for latency calculation, the outlier samples are
neutralized. This is a very common filtering for noise or errors on
signal and image processing.

9.7. Scenarios

Q4S could be used in two scenarios:

o client to ACP (Application content provider)

o client to client (peer to peer scenario)

9.7.1. Client to ACP

One server:

It is the common scenario in which client contact server to
establish a Q4S session.

N servers:

In Content Delivery Networks and in general applications where
delivery of contents can be achieved by different delivery nodes,
two working mechanisms can be defined

    o   Starting mode: End-user may run Q4S against several delivery
       nodes and after some seconds choose the best one to start the
       multimedia session

    o   Prevention mode:During streaming session, user keeps several
       Q4S dialogs against different alternative delivery nodes. In
       case of congestion, end-user may change to the best alternative
       delivery node

9.7.2. Client to client

In order to solve the client to client scenario, a Q4S register
function MUST be implemented. This allows clients contact each other
for sending the BEGIN message. In this scenario, the Register server
would be used by peers to publish their Q4S-Resource-Server header
and their public IP address to make possible the assumption of
server role.

The register function is out of scope of this protocol version,
because different HTTP mechanisms can be used and Q4S MUST NOT force
any.

10. Security Considerations

Different types of attacks can be avoided:

      o Spoofing of server IP address can be avoided using the
        digital signature mechanism. The network can easily
        validate this digital signature using the public key of the
        server certificate.

This protocol could be supported over IPSec to increase privacy,
although it is out of scope of this proposal.

11. IANA Considerations

    A specific port for Q4S TCP control flow mechanism could be
    assigned. It could simplify the network implementation. Other
    possibility is to use any other port (like 80, HTTP). In this case
    the network could use the protocol designator "Q4S" as the mark for
    distinguish and treat the packets.

    Q4S uses SDP as a container for session information, in which
    quality attributes have been added as extended "session-level"
    attributes. These set of new attributes should be registered (in
    order to avoid the prefix "X-"). In this document, this set of
    attributes has been presented as registered attributes.

    This is the list of attribute field names to register:

Attribute name: qos-level
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: defines the current QoS profile in uplink
and downlink for the communication between client and server. The
exact meaning of each level is implementation dependant but in
general, a higher qos-level value corresponds to a better quality
network profile.
Appropriate attribute values: [0..9] "/" [0..9]

Attribute name: alerting-mode
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: defines the receiver of the Q4S alerts sent
by the server. In q4s-aware-network alerting mode, Q4S alerts are
sent to the client. In this case the network is supposed to be Q4S
aware, and reacts by itself to these alerts. In policy-server
alerting mode, Q4S alerts sent to the network policy server. In this
case the network is not Q4S aware and a specific node (policy
server) is supposed to be in charge of achieving network tuning
mechanisms. The "alerting-mode" attribute is optional, and its
default value, when it is not present, is "policy-server".
Appropriate attribute values: <"q4s-aware-network"|"policy-server">

Attribute name: network-alert-pause
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: interval of time in milliseconds that the
server must wait between Q4S-ALERT messages in order to allow
network tuning operations. Measurements are not affected by this
pause.
Appropriate attribute values: [0..60000]

Attribute name: app-alert-pause
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: interval of time in milliseconds between
application notifications of violated quality constraints as
specified in the application parameters.
Appropriate attribute values: [0..300000]

Attribute name: public-address
Type of attribute: session level
Subject to the charset attribute: NO

Explanation of purpose: contains the public IP address of the client
or the server.
Appropriate attribute values:<"client"|"server"><"IP4"|"IP6"> <value
of IP address>

Attribute name: latency
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: defines the latency constraints in
milliseconds for the communication between client and server.
Appropriate attribute values: [0..9999]

Attribute name: jitter
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: defines the jitter constraints in
milliseconds in uplink and downlink for the communication between
client and server.
Appropriate attribute values: [0..9999] "/" [0..9999]

Attribute name: bandwidth
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: define the bandwidth constraints in kbps in
uplink and downlink for the communication between client and server.
Appropriate attribute values: [0..99999] "/" [0..99999]

Attribute name: packetloss
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: define the packet loss tolerance constraints
in 100% in uplink and downlink for the communication between client
and server.
Appropriate attribute values: [0.00 ..100.00] "/"[0.00 ..100.00]

Attribute name: application
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: define the quality parameter tolerance
constraints with uplink and downlink values for the communication
between the client and the server for four different parameters:
latency, jitter, bandwidth and packetloss.
Attribute values:
   <"latency:"> [0..9999]
   <"jitter:"|"bandwidth:"> [0..99999] "/" [0..99999]]
   <"packetloss:" [0.00 ..100.00] "/"[0.00 ..100.00]]

Attribute name: flow
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: define a flow between a client and a server.
The flow involves purpose (application or q4s –control-),
destination port (server or client) protocol (UDP or TCP) and port
or range or ports
The "clientListeningPort" is a port in which the client listens for
server requests and MUST be used as origin port of client responses.
The "serverListeningPort" is a port in which server is listening for
incoming messages from the client. The origin port of server
responses may be different than "serverListeningPort" value.
Attribute values:
    <"q4s"|"app"> <"serverListeningPort"|"clientListeningPort">
<"UDP"|"TCP"> <0..65535>["-" [0..65535]]

Attribute name: measurement
Type of attribute: session level
Subject to the charset attribute: NO
Explanation of purpose: define the procedure to measure the quality
and the different values for each measurement
Attribute values: "procedure/" <procedure> |
                    "latency "[0..9999] "/" [0..9999] |
                    "jitter "[0..9999] "/" [0..9999] |
                    "bandwidth "[0..99999] "/" [0..99999] |
                    "packetloss "[0.00..100.00] "/" [0.00..100.00]

If the attribute value is "procedure", the rest of the line MUST
contain the name of the procedure and optional parameters, separated
by ",".

In the case of procedure "default", the valid values are:

a=measurement:procedure default,[0..999]"/" [0..999] "," [0..999]
"/" [0..999] "," [0..9999] "," [0..999]/[0..999] ","
[0..999]/[0..999]

  where:

  o The first parameter is the interval of time (in milliseconds)
     between PING requests during the negotiation phase. Uplink and
     downlink values from the client's point of view are separated
     by "/". This allows having different responsiveness values
     depending on the control resources used in each direction.

   o The second parameter is the time interval (in milliseconds)
      between PING requests during the continuity phase. Uplink and
      downlink values are separated by "/". This allows having two
      different responsiveness values depending on the control
      resources used in each direction.

   o The third parameter is the time interval to be used to measure
      bandwidth during the negotiation phase.

   o The fourth parameter indicates the window size for jitter and
      latency calculations. Forward and reverse values are separated
      by "/".

   o The fifth parameter indicates the window size for packet loss
      calculations. Forward and reverse values are separated by "/".

   Other procedure names are allowed, but at least "default" procedure
   implementation is mandatory in client and servers.

12. Conclusions

   Q4S defines four phases with different purposes, and inside these
   phases the negotiated measurement procedure is used. Basically, Q4S
   only defines how to transport SLA information and measurement
   results as well as providing some mechanisms for alerting network
   and/or application.

   Q4S does not ask for resources neither define actions to do in case
   quality is degraded. Q4S only alerts if one (or some) of SLA quality
   parameters are being violated. Depends on server (Application
   content provider) to do something with this information and return
   it back to a SLA-compliant state.

13. References

13.1. Normative References

[1]   Fielding, R., Gettys, J., Mogul, J., Frystyk, H.,Masinter, L.,
        Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol --
        HTTP/1.1" RFC 2616, June 1999.

[2]   Handley, M. and V. Jacobson, "SDP: Session Description
        Protocol", RFC 4566, July 2006.

[3]   Bradner, S., "Key words for use in RFCs to Indicate
        RequirementLevels", BCP 14, RFC 2119, March 1997.

[4]   Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform
        Resource Identifiers (URI): Generic Syntax", RFC 3986, January
        2005.

[5]   Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with
        SDP", RFC 3264, June 2002.

[6]   Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321,
        April 1992.

[7]   Johnsson, J., B. Kaliski, "Public-Key Cryptography Standards
        (PCS) #1: RSA Cryptography Specifications version 2.1", RFC
        3447, February 2003.

[8]   Postel, J., "DoD Standard Transmission Control Protocol", RFC
        761, January 1980.

[9]   Postel, J., "User Datagram Protocol", STD 6, RFC 768, August
        1980.

[10]   Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V. "RTP:
        A Transport Protocol for Real-Time Applications", RFC 3550,
        july 2003.

[11]   Yergeau, F., "UTF-8, a transformation format of ISO 10646",
        RFC   3629, November 2003.

[12]   Resnick, P., "Internet Message Format", RFC 5322, October 2008

13.2. Informative References

[13] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A.
      Peterson, J., Sparks, R., Handley, M. and Schooler, E. , "SIP:
      Session Initiation Protocol", RFC 3261, June 2002.

[14] Mathis, M., Semke, J., Mahdavi, J., Ott, T., "The Macroscopic
      Behavior of the TCP Congestion Avoidance Algorithm", Computer
      Communications Review, 27(3), July 1997.

[15] Floyd, S., "HighSpeed TCP for a Large Congestion Windows", RFC
      3649, December 2003.

[16] Rhee, I., Xu, L., Ha, S., "CUBIC for Fast Long-Distance
      Networks", Internet-draft draft-rhee-tcpm-cubic-02, February
      2009.

[17] Sridharan, M., Tan, K., Bansal, D., Thaler, D., "Compound TCP:
      A New TCP Congestion Control for High-Speed and Long Distance
      Networks", Internet-draft draft-sridharan-tcpm-ctcp-02,
      November, 2008.

14. Acknowledgments

Many people have made comments and suggestions contributing to this document. In particular, we would like to thank:

Sonia Herranz Pablo, Clara Cubillo Pastor, Francisco Duran Pina, Michael Scharf, Jesus Soto Viso and Federico Guillen.

15. Authors' Addresses

    Jose Javier Garcia Aranda
    Alcatel-Lucent
    C/Maria Tubau 9
    28050 Madrid
    Spain
    Phone: +34 91 330 4348
    Email: Jose_Javier.Garcia_Aranda@alcatel-lucent.com

    Jacobo Perez Lajo
    Alcatel-Lucent
    C/Maria Tubau 9
    28050 Madrid
    Spain
    Phone: +34 91 330 4165
    Email: jacobo.perez@alcatel-lucent.com

    Luis Miguel Diaz Vizcaino
    Alcatel-Lucent
    C/Maria Tubau 9
    28050 Madrid
    Spain
    Phone: +34 91 330 4871
    Email: Luismi.Diaz@alcatel-lucent.com

    Gonzalo Munoz Fernandez
    Alcatel-Lucent
    C/Maria Tubau 9
    28050 Madrid
    Spain
    Phone: +34 91 330 4000
    Email: gonzalo.munoz@alcatel-lucent.com

    Carlos Barcenilla
    Universidad Politecnica de Madrid
    Avenida Complutense 30
    28040 Madrid
    Spain
    Phone: +34 91 549 5700 - 3032
    Email: barcenilla@dit.upm.es

    Monica Cortes
    Universidad Politecnica de Madrid
    Avenida Complutense 30

28040 Madrid
Spain
Phone: +34 91 336 5700 - 3044
Email: cortesm@dit.upm.es

Joaquin Salvachua
Universidad Politecnica de Madrid
Avenida Complutense 30
28040 Madrid
Spain
Phone: +34 91 549 5700 - 3056
Email: jsr@dit.upm.es

Juan Quemada
Universidad Politecnica de Madrid
Avenida Complutense 30
28040 Madrid
Spain
Phone: +34 91 336 7331
Email: jquemada@dit.upm.es