                 Application REQuested IP over ATM (AREQUIPA)

Status of this Memo

   This memo provides information for the Internet community. This memo
   does not specify an Internet standard of any kind. Distribution of
   this memo is unlimited.

IESG Note:

   This RFC has not had the benefit of the rigorous peer review that is
   part of the process in an IETF working group. The technology it
   describes has been implemented and is now undergoing testing. It
   would be wise to analyze the results of this testing as well as to
   understand alternatives before committing to this approach for IP
   over ATM with QoS guarantees.

Abstract

   This document specifies a method for allowing ATM-attached hosts that
   have direct ATM connectivity to set up end-to-end IP over ATM
   connections within the reachable ATM cloud, on request from
   applications, and for the exclusive use by the requesting
   applications. This allows the requesting applications to benefit in a
   straightforward way from ATM's inherent ability to guarantee the
   quality of service (QoS).

   Given a mapping from service classes, as defined by INTSERV[6], to
   ATM traffic descriptors, Arequipa can be used to implement integrated
   services over ATM link layers. Usage of Arequipa to provide
   integrated services even if ATM is only available for intermediate
   links is not discussed in this document but should be straight-
   forward.

   The major advantage of using an approach like Arequipa is that it
   needs to be implemented only on the hosts using it. It requires no
   extra service (eg. NHRP or RSVP) to be deployed on the switches or
   routers of the ATM cloud.

We discuss the implementation of Arequipa for hosts running IPv4 and
IPv6. As an illustration, we also discuss how World-Wide-Web
applications can use Arequipa to deliver documents with a guaranteed
quality of service.

In particular we show how

  - Arequipa can be implemented in IPv4 by slightly modifying the
  - Arequipa can be implemented in IPv6[3] by the appropriate use of
    flow labels and the extension of the neighbour cache,
  - Arequipa can be used in the Web by adding extra information in
    the headers of HTTP requests and responses.

Finally, we address safety and security implications.

1. Introduction

   QoS guarantees are important for delivery of multi-media data and
   commercial services on the Internet. When two applications on
   machines running IP over ATM need to transfer data, all the necessary
   gears to guarantee QoS can be found in the ATM layer. We consider
   the case where it is desired to use end-to-end ATM connections
   between applications residing on ATM hosts that have end-to-end ATM
   connectivity.

   Opening direct ATM connections between two applications is possible,
   but then the already available transport protocols, like TCP, can not
   be reused.

   This is why we propose Application REQuested IP over ATM (AREQUIPA).
   Arequipa allows applications to request that two machines be
   connected by a direct ATM connection with given QoS at the link
   level. Arequipa makes sure that only data from the applications that
   requested the connection actually goes through that connection. After
   setup of the Arequipa connection, the applications can use the
   standard IP protocol suite to exchange data.

2. API semantics

   We now define a semantical API for Arequipa. Note that an actual API
   may perform additional functions (eg. mapping of a given service
   specification to ATM traffic descriptors)

   We define the three new API functions for the TCP/IP stack:

   Arequipa_preset (socket_descriptor, destination IP address,
                    destination protid/port, destination ATM Address,
                    ATM service and QoS parameters)

      Arequipa_preset establishes or prepares establishment of a new ATM
      connection to the given address with the given ATM service and QoS.
      It makes sure that any further data sent on the specified socket
      will use the new ATM connection.

      Arequipa_preset is invoked before a TCP/IP connection is
      established or before sending data(grams), respectively. (Active
      open.)

   Arequipa_expect (socket_descriptor, allow)

      Arequipa_expect prepares the system to use an expected incoming
      Arequipa connection for outgoing traffic of a given socket. If
      allow equals TRUE then, as soon as the socket receives data from an
      incoming Arequipa connection, all its return traffic is sent over
      that Arequipa connection. If allow equals FALSE the traffic from
      that socket is always sent over the standard IP route. Note that
      Arequipa_expect is only applicable to connection oriented sockets,
      eg. TCP sockets or connected UDP sockets.

      Arequipa_expect is invoked by the peer which is expecting
      data(grams) or accepting connections. (Passive open.) It is
      typically called immediately after a socket has been created. It
      may also be called when a data transfer is already going on.

   Arequipa_close (socket_descriptor)

      Closes the corresponding ATM connection. Any further traffic
      between the endpoints is routed like other traffic. Arequipa_close
      is implied when closing the socket.

   Note that the use of Arequipa_expect or _preset only reflects the
   direction of the initial dialog in the Arequipa connection. Actual
   data can flow in both directions.

   An actual implementation may use less arguments for Arequipa_preset
   if some of the information is already passed by other networking
   operations.

3. Implementation with IPv4

   To implement Arequipa with IPv4, ATMARP must be able not only to
   handle associations of ATM addresses and IP addresses, but also
   associations of one ATM address with an IP address plus endpoint
   (socket). This allows to dedicate an ATM connection for the traffic
   between two endpoints.

For the active open, a destination ATM address must be associated
with a socket. In systems using per-socket route and ARP caching,
this can be done by presetting the caches with the appropriate
values. Establishment of the SVC is delegated to ATMARP. Care must be
taken that routing and ARP information obtained through Arequipa does
not leak to other parts of the system.

For the passive open, an incoming SVC must be associated with the
socket that terminates the corresponding connection or data flow.
Most of this functionality is already available in the existing
protocol stack. To avoid an incoming Arequipa SVC to be mistaken for
an IP-over-ATM SVC, the setup message uses a specific Broadband High
Layer Identifier (BHLI), see below. Seeing the BHLI, ATMARP knows
that the SVC is of the dedicated type. The socket to which it has to
be associated is identified as soon as a datagram is received through
the SVC. If an Arequipa_expect has been done for that socket, then
the SVC is used for all return traffic of that socket.

If application A1 on host H1 wants a direct ATM connection to
application A2 on host H2 it does the following:

Both applications first get in contact using the standard IP over ATM
to exchange the ATM address of the receiver (atm2) and the endpoints
(S1, S2) (i.e. protocol and port number; we assume that a protocol
with ports, such as TCP or UDP, is used) at both hosts between which
communication will occur. How this is performed depends on the
application protocols. In Section 5 we give an example for HTTP.

A2 invokes Arequipa_expect to indicate that it wants to make use of
an expected incoming ATM connection.

A1 invokes Arequipa_preset to open or prepare opening of an ATM
connection to H2 using ATM address atm2 and the QoS desired by A1 as
soon as data is sent through S1. The connection is associated with S1
such that no other traffic (e.g. generated by other applications)
uses the new ATM connection.

An Arequipa connection shall be signaled by using the procedures and
codings described in RFC1755 [7], with the addition that the BHLI
information element be included in the SETUP message, with the
following coding:

```
--------------------------------------------------------------------
| bb_high_layer_information                                        |
--------------------------------------------------------------------
|   high_layer_information_type    3             (vendor-specific  |
|                                                 application id.) |
|   high_layer_information         00-60-D7      (EPFL OUI)         |
|                                  01-00-00-01   (Arequipa)         |
--------------------------------------------------------------------
```

As soon as data arrives from H1:S1 at H2:S2, the ATM connection the
data has arrived on is identified as the dedicated connection for
this data flow and S2 is changed to exclusively send on that
connection.

From this point on all traffic exchanged between S1 of A1 and S2 of
A2 will use the new ATM connection with the desired QoS.

Note that it is possible for H1 and H2 to belong to the same LIS [2]
and still decide to use an Arequipa connection between applications,
in addition to one or several other, non-Arequipa ATM connections
between hosts H1 and H2. There may also exist several Arequipa
connections between two hosts.

4. Implementation with IPv6

   With IPv6, sources take advantage of the Flow Label field in the IPv6
   header [3].

   We assume as in [4] that the conceptual host model uses, among
   others, a neighbour cache and a destination cache. The destination
   cache holds entries about destinations to which traffic has been sent
   recently, while the neighbour cache holds entries about neighbours to
   which traffic has been sent recently. With the classical IP over ATM
   model [1], entries in the neighbour cache can only refer to systems
   in the same LIS; we propose to go beyond this limitation and allow
   systems beyond the LIS to appear there and be treated as neighbours,
   in the case where a direct link level connection (here, an ATM
   connection) can be established.

   The destination is keyed in [4] by the IP (destination) address. We
   replace this by the IP (destination) address and flow label.

We assume that with IPv6, a mechanism will be provided for
applications to request flow labels which, at the host, form a unique
flow-label/destination-address pair. This will prevent two different
flows which go to the same destination from accidentally using the
same flow label. Such a uniqueness requirement is also desirable for
other applications which rely on flow-label/destination-address
pairs, like for example RSVP.

A typical scenario is:

Application A1 on host H1 and application A2 on host H2 first get in
contact using the standard IP over ATM to exchange their ATM address
(atm1, atm2) and to define a protocol, port number pair (S1, S2) and
flow labels (L1, L2) for the communication over the ATM connection.
(We assume that a protocol with ports, such as TCP or UDP, is used).
How this is performed depends on the application protocols. In
Section 5 we give an example for HTTP.

A2 tells its networking entity that it wants to send its outgoing
packets with flow label L2 over an expected incoming ATM connection.
A1 tells its data link entity to open an ATM connection to H2 using
ATM address atm2, with the QoS desired by A1. The connection is
associated with L1 and L2 as explained below so that no other traffic
generated by other applications uses the new ATM connection.

From this point on all traffic exchanged between applications A1 on
H1 and application A2 on H2 will use this ATM connection.

An example of destination and neighbour cache entries at H1 is given
below.

Destination Cache

IPAddr	flowLabel	neighbourCache	pathMTU
H2	L1	ptr1	(1)
H2	*	ptr2	(2)

Neighbour Cache

IPAddr	linkLayerAddr	isRouter	reachabilityState	invalidationTimer
H2	v2	no	(3)	t2
R3	v3	yes	REACHABLE	t3

In the example, the route to destination H2 for all traffic other
than the one using the ATM connection requested between application
A1 and A2 uses the default route (perhaps set up by the classical IP
model), with router R3 as the next hop; v2 is a pointer to an ATM
interface and a VPCI/VCI that identifies the Arequipa connection.
Similarly, v3 points to the ATM connection to router R3. ptr1 points

to the first line in Neighbour Cache, and ptr2 to the second one.
Path MTUs (1) and (2) are obtained by ATM signaling; they may be
different. Reachability state (3) is determined as usual by the
reachability protocol [4].

Host H1 must restrict the use of this ATM connection to datagrams
with flow label L1. Other traffic from H1 to H2 must use the generic
entry in the destination table (flow label = "*"). Host H1 must
restrict the use of flow label L1 for destination H2 to traffic
generated by application A1 on port S1. (The same holds by analogy
for host H2).

On the receiving side, host H2 may use label L1 for routing
internally the IP packets to the appropriate entity.

5. Example: Arequipa for the Web

   This is a brief explanation of how Web [5] servers and browsers can
   use Arequipa to transmit documents with a guaranteed QoS.

   What we describe below does not violate the standards of HTML and
   HTTP but makes use of their built-in extensibility. The server and
   client we describe can thus interact seamlessly with non-modified
   servers or clients. A similar extension could be used if Web
   documents were to be exchanged using RSVP.

   Browsers add one extra field in all their requests or responses to
   indicate their ATM address. Web documents are extended with meta
   information to describe the ATM service and corresponding QoS needed
   to transmit them. Note that this information could be in form of an
   intserv flowspec and mapped to ATM traffic descriptors.

   If a browser always wants documents with QoS meta-information to be
   delivered using Arequipa, it adds an additional field in its request
   to indicate the port on which it is expecting the data.

   If a browser wants to decide whether Arequipa should be used or not,
   it does not give the port on which the server should send the data.

   When a server gets a request with an ATM address, it checks whether
   the requested document has QoS meta-information. If this is not the
   case, it delivers the document like a standard server. If the
   document has QoS meta-information, the server looks for a port
   information in the request. If it finds a port, it opens an Arequipa
   socket (Arequipa_preset) to the ATM address of the client with the
   QoS given in the document. It sends the reply through this new
   connection. If the server finds no port information, it sends only
   the header of the reply (which includes meta-information) over the

standard HTTP connection, as if the client had issued a HEAD or GET-
IF-MODIFIED request.

When a client receives the header of a document it can decide whether
it wants the document to be transmitted using Arequipa or not. A
client without a priori knowledge about the document, may therefore
always want to retrieve the header before requesting the full
document.

Illustration:

A client requests some documents but wants to decide if QoS sensitive
documents should be sent using Arequipa or not. Thus it adds to its
requests its ATM address but not the socket information.

```
GET batman.mpeg
UserAgent: MyAgent/1.0
ATM-address: my_public_address.my_private_address
```

The server checks batman.mpeg for QoS meta info. It finds the meta
info and sees an ATM address, but no socket pragma in the request. It
only returns the header of the document, which includes the meta-
information:

```
                              HTTP/1.0 200 OK
                              Server: MyAgent/1.0
                              ATM-Service: CBR
                              ATM-QoS-PCR: 2000
                              Content-type: video/mpeg
```

The client sees the QoS info and decides that it wants to download
the document using Arequipa. It opens a TCP socket for listening,
makes the Arequipa_expect call and sends the following request:

```
GET batman.mpeg
UserAgent: MyAgent/1.0
ATM-address: my_public_address.my_private_address
Pragma: socket=TCP.8090
```

Again the server checks batman.mpeg for QoS meta info. It finds the
meta info and sees the ATM address and the socket pragma in the
request. It creates a TCP socket, makes the Arequipa_preset call,
connects its TCP socket to the one of the client and sends the
response over the new TCP connection:

```
                              HTTP/1.0 200 OK
                              Server: MyAgent/1.0 ATM.address
                              ATM-Service: CBR
                              ATM-QoS-PCR: 2000
                              Content-type: video/mpeg

                              <mpeg data>
```

When the server sends the data over the new TCP connection it also
sends a copy of the response header over the TCP connection on which
the request was made. For example, this allows a browser to spawn a
viewer before requesting the data, to give the Arequipa connection to
the viewer and to still get the status of the request over the normal
TCP connection.

6. Safety considerations (loops)

   A major concern about ATM shortcuts in IP networks are routing loops.
   Arequipa is not prone to such dangers since it establishes
   connections between applications and not between hosts. All datagrams
   traveling through an Arequipa connection are destined for a given
   socket on the machine at the end of the connection and don't need to
   be forwarded by the IP layer. Therefore, neither hosts nor routers
   implementing Arequipa as described in this document must ever forward
   IP packets received over Arequipa connections.

7. Security considerations

   The main security problem we see with Arequipa is that it could be
   used to bypass IP firewalls.

   IP firewalls are used to protect private networks connected to
   untrusted IP networks. The network is configured such that all
   traffic going into or coming from the protected network has to go
   through the machine(s) acting as a firewall.

   If hosts in a network protected by a firewall are able to establish
   direct ATM connections to hosts outside the protected network, then
   Arequipa could be used to bypass the firewall. To avoid this, hosts
   inside a protected network should not be given direct connectivity to
   the outside of the network.

Arequipa can be used in a safe way by machines inside and outside a
protected network, if an application proxy is installed on the
firewall. In the Web, this is a typical scenario. Proxy HTTP servers
are often found on firewalls, not only for security reasons, but also
for caching. If an application proxy is used, each host can establish
an Arequipa connection to the proxy which can then relay and monitor
the traffic across the firewall.

Note that hosts can easily identify (and refuse) unsolicited Arequipa
connections by the BHLI identifier that is passed at connection
setup.

8. References

    [1] Laubach, M., Classical IP and ARP over ATM, RFC1577,
        January 1994.

    [2] Cole, R. G., D. H. Shur, C. Villamizar, IP over ATM: A Framework
        Document, RFC1932, April 1996.

    [3] Hinden, R. and S. Deering, Internet Protocol Version (IPv6)
        Addressing Architecture, RFC1884, December 1995.

    [4] Narten, T., E. Nordmark and W.A. Simpson, Neighbour Discovery for
        IPv6 (IPv6), RFC1970, August 1996.

    [5] Berners-Lee, T., R. Fielding, H. Nielsen, Hypertext Transfer
        Protocol -- HTTP/1.0, RFC1945, May 1996.

    [6] Shenker, S./J. Wroclawski, Network Element Service Specification
        Template, Work in Progess, November, 1995.

    [7] Perez, M., F.-C. Liaw, A. Mankin, E. Hoffman, D. Grossman, A.
        Malis, ATM Signaling Support for IP over ATM, RFC1755, February
        1995.

9. Authors' Address

    Werner Almesberger,
    Jean-Yves Le Boudec,
    Philippe Oechslin (contact author)

    Laboratoire de Reseaux de Communication
    Swiss Federal Institute of Technology (EPFL)
    1015 Lausanne
    Switzerland

    email: {almesber, leboudec, oechslin}@di.epfl.ch