Network Working Group   C. Partridge
Request for Comments: 1363   BBN
                                                      September 1992

                     A Proposed Flow Specification

Abstract

   A flow specification (or "flow spec") is a data structure used by
   internetwork hosts to request special services of the internetwork,
   often guarantees about how the internetwork will handle some of the
   hosts' traffic. In the future, hosts are expected to have to request
   such services on behalf of distributed applications such as
   multimedia conferencing.

   The flow specification defined in this memo is intended for
   information and possible experimentation (i.e., experimental use by
   consenting routers and applications only). This RFC is a product of
   the Internet Research Task Force (IRTF).

Introduction

   The Internet research community is currently studying the problems of
   supporting a new suite of distributed applications over
   internetworks. These applications, which include multimedia
   conferencing, data fusion, visualization, and virtual reality, have
   the property that they require the distributed system (the collection
   of hosts that support the applications along with the internetwork to
   which they are attached) be able to provide guarantees about the
   quality of communication between applications. For example, a video
   conference may require a certain minimum bandwidth to be sure that
   the video images are delivered in a timely way to all recipients.

   One way for the distributed system to provide guarantees is for hosts
   to negotiate with the internetwork for rights to use a certain part
   of the internetwork's resources. (An alternative is to have the
   internetwork infer the hosts' needs from information embedded in the
   data traffic each host injects into the network. Currently, it is
   not clear how to make this scheme work except for a rather limited
   set of traffic classes.)

Partridge   [Page 1]

There are a number of ways to effect a negotiation. For example a
negotiation can be done in-band or out-of-band. It can also be done
in advance of sending data (possibly days in advance), as the first
part of a connection setup, or concurrently with sending (i.e., a
host starts sending data and starts a negotiation to try to ensure
that it will allowed to continue sending). Insofar as is possible,
this memo is agnostic with regard to the variety of negotiation that
is to be done.

The purpose of this memo is to define a data structure, called a flow
specification or flow spec, that can be used as part of the
negotiation to describe the type of service that the hosts need from
the internetwork. This memo defines the format of the fields of the
data structure and their interpretation. It also briefly describes
what purpose the different fields fill, and discusses why this set of
fields is thought to be both necessary and sufficient.

It is important to note that the goal of this flow spec is to able to
describe *any* flow requirement, both for guaranteed flows and for
applications that simply want to give hints to the internetwork about
their requirements.

Format of the Flow Spec

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Version           |    Maximum Transmission Unit  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       Token Bucket Rate       |        Token Bucket Size      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Maximum Transmission Rate   |      Minimum Delay Noticed    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Maximum Delay Variation   |        Loss Sensitivity       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Burst Loss Sensitivity    |         Loss Interval         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Quality of Guarantee      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Discussion of the Flow Spec

The flow spec indicates service requirements for a single direction.
Multidirectional flows will need to request services in both
directions (using two flow specs).

To characterize a unidirectional flow, the flow spec needs to do four
things.

First, it needs to characterize how the flow's traffic will be
injected into the internetwork. If the internetwork doesn't know
what to expect (is it a gigabit-per-second flow or a three kilobit-
per-second flow?) then it is difficult for the internetwork to make
guarantees. (Note the word "difficult" rather than "impossible." It
may be possible to statistically manage traffic or over-engineer the
network so well that the network can accept almost all flows, without
setup. But this problem looks far harder than asking the sender to
approximate its behavior so the network can plan.) In this flow
spec, injected traffic is characterized as having a sustainable rate
(the token bucket rate) a peak rate (the maximum transmission rate),
and an approximate burst size (the token bucket size). A more
precise definition of each of these fields is given below. The
characterization is based, in part, on the work done in [1].

Second, the flow spec needs to characterize sensitivity to delay.
Some applications are more sensitive than others. At the same time,
the internetwork will likely have a choice of routes with various
delays available from the source to destination. For example, both
routes using satellites (which have very long delays) and routes
using terrestrial lines (which will have shorter delays) may be
available. So the sending host needs to indicate the flow's
sensitivity to delay. However, this field is only advisory. It only
tells the network when to stop trying to reduce the delay - it does
not specify a maximum acceptable delay.

There are two problems with allowing applications to specify the
maximum acceptable delay.

First, observe that an application would probably be happy with a
maximum delay of 100 ms between the US and Japan but very unhappy
with a delay of 100 ms within the same city. This observation
suggests that the maximum delay is actually variable, and is a
function of the delay that is considered achievable. But the
achievable delay is largely determined by the geographic distance
between the two peers, and this sort of geographical information is
usually not available from a network. Worse yet, the advent of
mobile hosts makes such information increasingly hard to provide. So
there is reason to believe that applications may have difficulty
choosing a rational maximum delay.

The second problem with maximum delays is that they are an attempt to
quantify what performance is acceptable to users, and an application
usually does not know what performance will be acceptable its user.
For example, a common justification for specifying a maximum
acceptable delay is that human users find it difficult to talk to
each other over a link with more than about 100 ms of delay.
Certainly such delays can make the conversation less pleasant, but it

is still possible to converse when delays are several seconds long,
and given a choice between no connection and a long delay, many users
will pick the delay. (The phone call may involve an important matter
that must be resolved.)

As part of specifying a flow's delay sensitivity, the flow spec must
also characterize how sensitive the flow is to the distortion of its
data stream.

Packets injected into a network according to some pattern will not
normally come out of the network still conforming to the pattern.
Instead, the pattern will have been distorted by queueing effects in
the network. Since there is reason to believe that it may make
network design easier to continue to allow the networks slightly
distort traffic patterns, it is expected that those applications
which are sensitive to distortion will require their hosts to use
some amount of buffering to reshape the flow back into its original
form. It seems reasonable to assume that buffer space is not
infinite and that a receiving system will wish to limit the amount of
buffering that a single flow can use.

The amount of buffer space required for removing distortion at the
receiving system is determined by the variation in end-to-end
transmission delays for data sent over the flow. If the transmission
delay is a mean delay, D, plus or minus a variance, V, the receiving
system needs buffer space equivalent to $2 * V *$ the transmission
rate. To see why this is so, consider two packets, A and B, sent T
time units apart which must be delivered to the receiving application
T time units apart. In the worst case, A arrives after a delay of
D-V time units (the minimum delay) and B arrives after a delay of D+V
time units (the maximum delay). The receiver cannot deliver B until
it arrives, which is $T + 2 * V$ time units after A. To ensure that A
is delivered T time units before B, A must be buffered for $2 * V$ time
units. The delay variance field is the value of $2 * V$, and allows
the receiver to indicate how much buffering it is willing to provide.

A third function of the flow spec is to signal sensitivity to loss of
data. Some applications are more sensitive to the loss of their data
than other applications. Some real-time applications are both
sensitive to loss and unable to wait for retransmissions of data.
For these particularly sensitive applications, hosts may implement
forward error correction on a flow to try to absolutely minimize
loss. The loss fields allow hosts to request loss properties
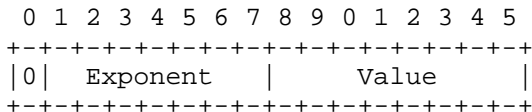appropriate for the application's requirements.

Finally, it is expected that the internetwork may be able to provide
a range of service guarantees. At the best, the internetwork may be
asked to guarantee (with tight probability bounds) the quality of

service it will provide. Or the internetwork may simply be asked to
ensure that packets sent over the flow take a terrestrial path. The
quality of guarantee field indicates what type of service guarantee
the application desires.
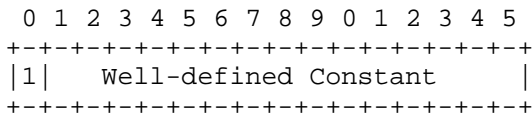
Definition of Individual Fields

General Format of Fields

   With a few exceptions, fields of the flow spec are expressed using a
   common 16-bit format. This format has two forms. The first form is
   shown below.

```
            0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |0|  Exponent    |     Value     |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   In this format, the first bit is 0, followed by 7 bits of an exponent
   (E), and an 8-bit value (V). This format encodes a number, of the
   form $V * (2^E)$. This representation was chosen to allow easy
   representation of a wide range of values, while avoiding over-precise
   representations.

   In some case, systems will not wish to request a precise value but
   rather simply indicate some sensitivity. For example, a virtual
   terminal application like Telnet will likely want to indicate that it
   is sensitive to delay, but it may not be worth expressing particular
   delay values for the network to try to achieve. For these cases,
   instead of a number, the field in the flow spec will take the
   following form:

```
            0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
           |1|   Well-defined Constant     |
           +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

   The first bit of the field is one, and is followed by a 15-bit
   constant. The values of the constants for given fields are defined
   below. Any additional values can be requested from the Internet
   Assigned Numbers Authority (IANA).

   Version Field

      This field is a 16-bit integer in Internet byte order. It is the
      version number of the flow specification. The version number of
      the flow specification defined in this document is 1. The IANA is
      responsible for assigning future version numbers for any proposed

revisions of this flow specification.

This field does not use the general field format.

   Maximum Transmission Unit (MTU)

A 16-bit integer in Internet byte order which is the maximum
number of bytes in the largest possible packet to be transmitted
over this flow.

This field does not use the general field format.

The field serves two purposes.

It is a convenient unit for expressing loss properties. Using the
default MTU of the internetwork is inappropriate since the
internetwork have very large MTU, such the 64Kbytes of IP, but
applications and hosts may be sensitive to losses of far less than
an MTU's amount of data -- for example, a voice application would
be sensitive to a loss of several consecutive small packets.

The MTU also bounds the amount of time that a flow can transmit,
uninterrupted, on a shared media.

Similarly, the loss rates of links that suffer bit errors will
vary dramatically based on the MTU size.

   Token Bucket Rate

The token bucket rate is one of three fields used to define how
traffic will be injected into the internetwork by the sending
application. (The other two fields are the token bucket size and
the maximum transmission rate.)

The token rate is the rate at which tokens (credits) are placed
into an imaginary token bucket. For each flow, a separate bucket
is maintained. To send a packet over the flow, a host must remove
a number of credits equal to the size of the packet from the token
bucket. If there are not enough credits, the host must wait until
enough credits accumulate in the bucket.

Note that the fact that the rate is expressed in terms of a token
bucket rate does not mean that hosts must implement token buckets.
Any traffic management scheme that yields equivalent behavior is
permitted.

The field is in the general field format and counts the number of
byte credits (i.e., right to send a byte) per second which are

deposited into the token bucket. The value must be a number (not
a well-known constant).

The value zero is slightly special. It is used to indicate that
the application is not making a request for bandwidth guarantees.
If this field is zero, then the Token Bucket Size must also be
zero, and the type of guarantee requested may be no higher than
predicted service.

Token Bucket Size

The token bucket size controls the maximum amount of data that the
flow can send at the peak rate. More formally, if the token
bucket size is B, and the token bucket rate is R, over any
arbitrarily chosen interval T in the life of the flow, the amount
of data that the flow sends cannot have exceeded B + (R * T)
bytes.

The token bucket is filled at the token bucket rate. The bucket
size limits how many credits the flow may store. When the bucket
is full, new credits are discarded.

The field is in the general field format and indicates the size of
the bucket in bytes. The value must be a number.

Note that the bucket size must be greater than or equal to the MTU
size.

Zero is a legal value for the field and indicates that no credits
are saved.

Maximum Transmission Rate

The maximum transmission rate limits how fast packets may be sent
back to back from the host. Consider that if the token bucket is
full, it is possible for the flow to send a series of back-to-back
packets equal to the size of the token bucket. If the token
bucket size is large, this back-to-back run may be long enough to
significantly inhibit multiplexing.

To limit this effect, the maximum transmission rate bounds how
fast successive packets may be placed on the network.

One can think of the maximum transmission rate control as being a
form of a leaky bucket. When a packet is sent, a number of
credits equal to the size of the packet is placed into an empty
bucket, which drains credits at the maximum transmission rate. No
more packets may be sent until the bucket has emptied again.

The maximum transmission rate is the rate at which the bucket is
emptied. The field is in the general field format and indicates
the size of the bucket in bytes. The value must be a number and
must be greater than or equal to the token bucket rate.

Note that the MTU size can be used in conjunction with the maximum
transmission rate to bound how long an individual packet blocks
other transmissions. The MTU specifies the maximum time an
individual packet may take. The Maximum Transmission Rate, limits
the frequency with which packets may be placed on the network.

Minimum Delay Noticed

The minimum delay noticed field tells the internetwork that the
host and application are effectively insensitive to improvements
in end-to-end delay below this value. The network is encouraged
to drive the delay down to this value but need not try to improve
the delay further.

The field is in the general field format.

If expressed as a number it is the number of microseconds of delay
below which the host and application do not care about
improvements. Human users only care about delays in the
millisecond range but some applications will be computer to
computer and computers now have clock times measured in a handful
of nanoseconds. For such computers, microseconds are an
appreciable time. For this reason, this field measures in
microseconds, even though that may seem small.

If expressed as a well-known constant (first bit set), two field
values are accepted:

    0 - the application is not sensitive to delay

    1 - the application is moderately delay sensitive
        e.g., avoid satellite links where possible).

Maximum Delay Variation

If a receiving application requires data to be delivered in the
same pattern that the data was transmitted, it may be necessary
for the receiving host to briefly buffer data as it is received so
that the receiver can restore the old transmission pattern. (An
easy example of this is a case where an application wishes to send
and transmit data such as voice samples, which are generated and
played at regular intervals. The regular intervals may be
distorted by queueing effects in the network and the receiver may

have to restore the regular spacing.)

The amount of buffer space that the receiving host is willing to
provide determines the amount of variation in delay permitted for
individual packets within a given flow. The maximum delay
variation field makes it possible to tell the network how much
variation is permitted. (Implementors should note that the
restrictions on the maximum transmission rate may cause data
traffic patterns to be distorted before they are placed on the
network, and that this distortion must be accounted for in
determining the receiver buffer size.)

The field is in the general field format and must be a number. It
is the difference, in microseconds, between the maximum and
minimum possible delay that a packet will experience. (There is
some question about whether microsecond units are too large. At a
terabit per second, one microsecond is a megabit. Presumably if a
host is willing to receive data at terabit speeds it is willing to
provide megabits of buffer space.)

The value of 0, meaning the receiving host will not buffer out
delays, is acceptable but the receiving host must still have
enough buffer space to receive a maximum transmission unit sized
packet from the sending host. Note that it is expected that a
value of 0 will make it unlikely that a flow can be established.

Loss Sensitivity

This field indicates how sensitive the flow's traffic is to
losses. Loss sensitivity can be expressed in one of two ways:
either as a number of losses of MTU-sized packets in an interval,
or simply as a value indicating a level of sensitivity.

The field is in the general field format.

If the value is a number, then the value is the number of MTU-
sized packets that may be lost out of the number of MTU-sized
packets listed in the Loss Interval field.

If the value is a well-known constant, then one of two values is
permitted:

    0 - the flow is insensitive to loss

    1 - the flow is sensitive to loss (where possible
        choose the path with the lowest loss rate).

Burst Loss Sensitivity

   This field states how sensitive the flow is to losses of
   consecutive packets. The field enumerates the maximum number of
   consecutive MTU-sized packets that may be lost.

   The field is in the general field format.

   If the value is a number, then the value is the number of
   consecutive MTU-sized packets that may be lost.

   If the value is a well-known constant, then the value 0 indicates
   that the flow is insensitive to burst loss.

   Note that it is permissible to set the loss sensitivity field to
   simply indicate sensitivity to loss, and set a numerical limit on
   the number of consecutive packets that can be lost.

Loss Interval

   This field determines the period over which the maximum number of
   losses per interval are measured. In other words, given any
   arbitrarily chosen interval of this length, the number of losses
   may not exceed the number in the Loss Sensitivity field.

   The field is in the general field format.

   If the Loss Sensitivity field is a number, then this field must
   also be a number and must indicate the number of MTU-sized packets
   which constitutes a loss interval.

   If the Loss Sensitivity field is not a number (i.e., is a well-
   known constant) then this field must use the well-known constant
   of 0 (i.e., first bit set, all other bits 0) indicating that no
   loss interval is defined.

Quality of Guarantee

   It is expected that the internetwork will likely have to offer
   more than one type of guarantee.

   There are two unrelated issues related to guarantees.

   First, it may not be possible for the internetwork to make a firm
   guarantee. Consider a path through an internetwork in which the
   last hop is an Ethernet. Experience has shown (e.g., some of the
   IETF conferencing experiments) that an Ethernet can often give
   acceptable performance, but clearly the internetwork cannot

guarantee that the Ethernet will not saturate at some time during
a flow's lifetime. Thus it must be possible to distinguish
between flows which cannot tolerate the small possibility of a
failure (and thus must guaranteed at every hop in the path) and
those that can tolerate islands of uncertainty.

Second, there is some preliminary work (see [2]) that suggests
that some applications will be able to adapt to modest variations
in internetwork performance and that network designers can exploit
this flexibility to allow better network utilization. In this
model, the internetwork would be allowed to deviate slightly from
the promised flow parameters during periods of load. This class
of service is called predicted service (to distinguish it from
guaranteed service).

The difference between predicted service and service which cannot
be perfectly guaranteed (e.g., the Ethernet example mentioned
above) is that the imperfect guarantee makes no statistical
promises about how it might mis-behave. In the worst case, the
imperfect guarantee will not work at all, whereas predicted
service will give slightly degraded service. Note too that
predicted service assumes that the routers and links in a path all
cooperate (to some degree) whereas an imperfect guarantee states
that some routers or links will not cooperate.

The field is a 16-bit field in Internet byte order. There are six
legal values:

    0 - no guarantee is required (the host is simply expressing
        desired performance for the flow)

    100 (hex) - an imperfect guarantee is requested.

    200 (hex) - predicted service is requested and if unavailable,
                then no flow should be established.

    201 (hex) - predicted service is requested but an imperfect
                guarantee is acceptable.

    300 (hex) - guaranteed service is requested and if a firm
                guarantee cannot be given, then no flow should be
                established.

    301 (hex) - guaranteed service is request and but an imperfect
                guarantee is acceptable.

It is expected that asking for predicted service or permitting an
imperfect guarantee will substantially increase the chance that a

flow request will be accepted.

Possible Limitations in the Proposed Flow Spec

   There are at least three places where the flow spec is arguably
   imperfect, based on what we currently know about flow reservation.
   In addition, since this is a first attempt at a flow spec, readers
   should expect modifications as we learn more.

   First, the loss model is not perfect. Simply stating that an
   application is sensitive to loss and to burst loss is a rather crude
   indication of sensitivity. However, explicitly enumerating loss
   requirements within a cycle is also an imperfect mechanism. The key
   problem with the explicit values is that not all packets sent over a
   flow will be a full MTU in size. Expressed another way, the current
   flow spec expects that an MTU-sized packet will be the unit of error
   recovery. If flows send packets in a range of sizes, then the loss
   bounds may not be very useful. However, the thought of allowing a
   flow to request a set of loss models (one per packet size) is
   sufficiently painful that I've limited the flow to one loss profile.
   Further study of loss models is clearly needed.

   Second, the minimum delay sensitivity field limits a flow to stating
   that there is one point on a performance sensitivity curve below
   which the flow is no longer interested in improved performance. It
   may be that a single point is insufficient to fully express a flow's
   sensitivity. For example, consider a flow for supporting part of a
   two-way voice conversation. Human users will notice improvements in
   delay down to a few 10s of milliseconds. However, the key point of
   sensitivity is the delay at which normal conversation begins to
   become awkward (about 100 milliseconds). By allowing only one
   sensitivity point, the flow spec forces the flow designer to either
   ask for the best possible delay (e.g, a few 10's of ms) to try to get
   maximum performance from the network, or state a sensitivity of about
   95 ms, and accept the possibility that the internetwork will not try
   to improve delay below that value, even if it could (and even though
   the user would notice the improvement). My expectation is that a
   simple point is likely to be easier to deal with than attempting to
   enumerate two (or three or four) points in the sensitivity curve.

   Third, the models for service guarantees is still evolving and it is
   by no means clear that the service choices provided are the correct
   set.

How an Internetwork is Expected to Handle a Flow Spec

   There are at least two parts to the issue of how an internetwork is
   expected to handle a flow spec. The first part deals with how the
   flow spec is interpreted so that the internetwork can find a route
   which will allow the internetwork to match the flow's requirements.
   The second part deals with how the network replies to the host's
   request.

   The precise mechanism for setting up a flow, given a flow spec, is a
   large topic and beyond the scope of this memo. The purpose of the
   next few paragraphs is simply to sketch an argument that this flow
   spec is sufficient to the requirements of the setup mechanisms known
   to the author.

   The key problem in setting up a flow is determining if there exist
   one or more routes from the source to the destination(s) which might
   be able to support the quality of service requested. Once one has a
   route (or set of candidate routes) one can take whatever actions may
   be appropriate to confirm that the route is actually viable and to
   cause the flow's data to follow that route.

   There are a number of ways to find a route. One might try to build a
   route on the fly by establishing the flow hop-by-hop (as ST-II does)
   or one might consult a route server which provides a set of candidate
   source routes derived from a routing database. However, whatever
   system is used, some basic information about the flow needs to be
   provided to the routing system. This information is:

      * How much bandwidth the flow may require. There's no point
        in routing a flow that expects to send at over 10 megabits per
        second via a T1 (1.5 megabit per second) link.

      * How delay sensitive the application is. One does not wish
        to route a delay-sensitive application over a satellite link,
        unless the satellite link is the only possible route from here
        to there.

      * How much error can be tolerated. Can we send this flow over
        our microwave channel on a rainy day or is a more reliable link
        required?

      * How firm the guarantees need to be. Can we put an Ethernet
        in as one of the hops?

      * How much delay variation is tolerated. Again, can an Ethernet
        be included in the path? Does the routing system need to worry
        if the addition of this flow will cause a few routers to run

at close to capacity? (A side note: we assume that the routers
are running with priority queueing systems, so running the router
close to capacity doesn't mean that all flows get long and
variable delays. Rather, running close to capacity means that
high priority flows will be unaffected, and low priority flows
will get hit with a lot of delay and variation.)

The flow spec provides all of this information. So it seems
plausible to assume it provides enough information to make routing
decisions at setup time.

The flow spec was designed with the expectation that the network
would give a yes or no reply to a request for a guaranteed flow.

Some researchers have suggested that the negotiation to set up a flow
might be an extended negotiation, in which the requesting host
initially requests the best possible flow it could desire and then
haggles with the network until they agree on a flow with properties
that the network can actually provide and the application still finds
useful. This notion bothers me for at least two reasons. First, it
means setting up a flow is a potentially long process. Second, the
general problem of finding all possible routes with a given set of
properties is a version of the traveling salesman problem, and I
don't want to embed traveling salesman algorithms into a network's
routing system.

The model used in designing this flow spec was that a system would
ask for the minimum level of service that was deemed acceptable and
the network would try to find a route that met that level of service.
If the network is unable to achieve the desired level of service, it
refuses the flow, otherwise it accepts the flow.

The Flow Spec as a Return Value

This memo does not specify the data structures that the network uses
to accept or reject a flow. However, the flow spec has been designed
so that it can be used to return the type of service being
guaranteed.

If the request is being accepted, the minimum delay field could be
set to the guaranteed or predicted delay, and the quality of
guarantee field could be set to no guarantee (0), imperfect guarantee
(100 hex), predicted service (200 hex), or guaranteed service (300
hex).

If the request is being rejected, the flow spec could be modified to
indicate what type of flow the network believes it could accept e.g.,
the traffic shape or delay characteristics could be adjusted or the

type of guarantee lowered). Note that this returned flow spec would
likely be a hint, not a promised offer of service.

Why Type of Service is not Good Enough

The flow spec proposed in this memo takes the form of a set of
parameters describing the properties and requirements of the flow.
An alternative approach which is sometimes mentioned (and which is
currently incorporated into IP) is to use a Type of Service (TOS)
value.

The TOS value is an integer (or bit pattern) whose values have been
predefined to represent requested quality of services. Thus, a TOS
of 47 might request service for a flow using up to 1 gigabit per
second of bandwidth with a minimum delay sensitivity of 100
milliseconds.

TOS schemes work well if the different quality of services that may
be requested are both enumerable and reasonably small.
Unfortunately, these conditions do not appear to apply to future
internetworks. The range of possible bandwidth requests alone is
huge. Combine this range with several gradations of delay
requirements, and widely different sensitivities to errors and the
set of TOS values required becomes extremely large. (At least one
person has suggested to the author that perhaps a TOS field combined
with a bandwidth parameter might be appropriate. In other words, a
two parameter model. That's a tempting idea but my gut feeling is
that it is not quite sufficient so I'm proposing a more complete
parametric model.)

Another reason to prefer parametric service is optimization issues.
A key issue in flow setup is trying to design the the routing system
to optimize its management of flows. One can optimize on a number of
criteria. A good example of an optimization problem is the following
question (expressed by Isidro Castineyra of BBN):

  "Given a request to establish a flow, how can the internetwork
  accept that request in such a way as to maximize the chance that
  the internetwork will also be able to accept the next flow
  request?"

The optimization goal here is call-completion - maximizing the chance
that requests to establish flows will succeed. One might
alternatively try to maximize revenue (if one is charging for flows).

The internetwork is presumably in a better position to do
optimizations if it has more information about the flow's expected
behavior. For example, if a TOS system says only that a flow is

delay sensitive, the routing system must seek out the most direct
route for the flow. But if the routing system is told that the flow
is sensitive only to delays over 100 milliseconds, there may be a
number of routes other than the most direct route which can satisfy
this delay, thus leaving the most direct route available for a later
flow which needs a far lower delay.

In fairness, it should be noted that a danger of a parametric model
is that it is very easy to have too many parameters. The yearn to
optimize can be overdone. The goal of this flow spec is to enumerate
just enough parameters that it appears that essential needs can be
expressed, and the internetwork has some information it can use to
try to manage the flows. Features that would simply be nice or
useful to have (but not essential) are left out to keep the parameter
space small.

An Implication of the Flow Spec

It is important to observe that the there are fields in the flow spec
that are based on information from the sender (such as rate
information) and fields in the flow spec that are based on
information from the receiver (such as delay variation). There are
also fields that may sender and receiver to negotiate in advance.
For example, the acceptable loss rate may depend on whether the
sender and receiver both support the same type of forward error
correction. The delay sensitivity for a voice connection may depend,
in part, on whether both sender and receiver support echo cancelling.

The implication is that the internetwork must permit the sender and
receiver to communicate in advance of setting up a flow, because a
flow spec can only be defined once both sender and receiver have had
their say. In other words, a reserved flow should not be the only
form of communication.   There must be some mechanism to perform a
short exchange of messages in preparation for setting up a flow.

(Another aside: it has been suggested that perhaps the solution to
this problem is to have the sender establish a flow with an
incomplete flow spec, and when the receiver gets the flow spec, have
the receiver send the completed flow spec back along the flow, so the
internetwork can "revise" the flow spec according to the receiver's
desires. I have two problems with this approach. First, it is
entirely possible that the receiver's information may lead the
internetwork to conclude that the flow established by the sender is
no good. For example, the receiver may indicate it has a smaller
tolerance for delay variation than expected and force the flow to be
rerouted over a completely different path. Second, if we try to
avoid having the receiver's information cause the flow to fail, then
we have to over-allocate the flow's during the preliminary setup.

But over allocating the resources requested may lead us to choose
better quality paths than we need for this flow. In other words, our
attempts to optimize use of the network will fail.)

Advance Reservations and Flow Duration

The primary purpose of a flow specification is to provide information
to the internetwork so the internetwork can properly manage the
proposed flow's traffic in the context of other traffic in the
internetwork. One question is whether the flow should give the
network information about when the flow is expected to start and how
long the flow is expected to last.

Announcing when a flow will start is generally of interest for
advance reservations. (If the flow is not be reserved substantially
in advance, the presentation of the flow spec to the internetwork can
be taken as an implicit request for a flow, now.) It is my view that
advance reservation is a distinct problem from the describing the
properties of a flow. Advanced reservations will require some
mechanism to maintain information in the network about flows which
are not currently active but are expected to be activated at some
time in the future. I anticipate this will require some sort of
distributed database to ensure that information about advanced
reservations is not accidentally lost if parts of the internetwork
crash. In other words, advance reservations will require
considerable additional supporting baggage that it would probably be
better to keep out of the average flow spec.

Deciding whether a flow spec should contain information about how
long the flow is expected to run is a harder decision to make.
Clearly if we anticipate that the internetwork will support advance
reservations, it will be necessary for elements of the internetwork
to predict their traffic load, so they can ensure that advance
reservations are not compromised by new flow requests. However,
there is a school of thought that believes that estimating future
load from current behavior of existing flows is more accurate than
anything the flows may have declared in their flow specs. For this
reason, I've left a duration field out of the flow spec.

Examples

To illustrate how the flow spec values might be used, this section
presents three example flow specs.

Telnet

For the first example, consider using the flow spec to request
service for an existing application: Telnet. Telnet is a virtual

terminal protocol, and one can think of it as stringing a virtual
wire across the network between the user's terminal and a remote
host.

Telnet has proved a very successful application without a need to
reserve bandwidth: the amount of data sent over any Telnet
connection tends to be quite small. However, Telnet users are
often quite sensitive to delay, because delay can affect the time
it takes to echo characters. This suggests that a Telnet
connection might benefit from asking the internetwork to avoid
long delay paths. It could so so using the following flow spec
(for both directions):

```
Version=1
MTU=80 [40 bytes of overhead + 40 bytes user data]
Token Bucket Rate=0/0/0 [don't want a guarantee]
Token Bucket Size=0/0/0
Maximum Transmission Rate=0/0/0
Maximum Delay Noticed=1/1 [constant = delay sensitive]
Maximum Delay Variation=0/0/0 [not a concern]
Loss Sensitivity=1/0 [don't worry about loss]
Burst Loss Sensitivity=1/0
Loss Interval=1/0
Quality of Guarantee=1/0 [just asking]
```

It is worth noting that Telnet's flow spec is likely to be the
same for all instantiations of a Telnet connection. As a result,
there may be some optimizations possible (such as just tagging
Telnet packets as being subject to the well-known Telnet flow
spec).

A Voice Flow

Now consider transmitting voice over the Internet. Currently,
good quality voice can be delivered at rates of 32Kbit/s or
16Kbit/s. Assuming the rate is 32Kbit/s and voice samples are 16
bit samples packaged into UDP datagrams (for a data rate of about
60 Kbyte/s), a flow spec might be:

```
Version=1
MTU=30 [2 byte sample in UDP datagram]
Token Bucket Rate=0/10/59 [60.4 Kbytes/s]
Token Bucket Size=0/0/30 [save enough to send immediately
                         after pauses]
Maximum Transmission Rate=0/10/59 [peak same as mean]
Maximum Delay Noticed=0/10/100 [100 ms]
Maximum Delay Variation=0/10/10 [keep variation low]
Loss Sensitivity=1/1 [loss sensitive]
```

```
      Burst Loss Sensitivity=0/0/5 [keep bursts small]
      Loss Interval=1/0
      Quality of Guarantee=1/201 [predicted service and I'll accept
                                 worse]
```

A Variable Bit-Rate Video Flow

Variable bit-rate video transmissions vary the rate at which they
send data according to the amount of the video image that has
changed between frames. In this example, we consider a one-way
broadcast of a picture. If we assume 30 frames a second and that
a full frame is about 1 megabit of data, and that on average about
10% of the frame changes, but in the worst case the entire frame
changes, the flow spec might be:

```
      Version=1
      MTU=4096 [big so we can put lots of bits in each packet]
      Token Bucket Rate=0/20/1 [8 Mbits/s]
      Token Bucket Size=0/17/2 [2 Mbits/s]
      Maximum Transmission Rate=0/20/30 [30 Mbits/s]
      Maximum Delay Noticed=1/1 [somewhat delay sensitive]
      Maximum Delay Variation=0/10/1 [no more than one second of
                                     buffering]
      Loss Sensitivity=0/0/1 [worst case, one loss per frame]
      Burst Loss Sensitivity=0/0/1 [no burst errors please]
      Loss Interval=0/0/33 [one frame in MTU sized packets]
      Quality of Guarantee=1/300 [guaranteed service only]
```

The token bucket is sized to be two frames of data, and the bucket
rate will fill the bucket every 250 ms. The expectation is that
full scene changes will be rare and that a fast rate with a large
bucket size should accommodate even a series of scene changes.

Disclaimer

In all cases, these examples are simply to sketch the use of the
flow spec. The author makes no claims that the actual values used
are the correct ones for a particular application.

Security Considerations

Security considerations definitely exist. For example, one might
assume that users are charged for guaranteed flows. In that case,
some mechanism must exist to ensure that a flow request (including
flow spec) is authenticated. However I believe that such issues have
to be dealt with as part of designing a negotiation protocol, and are
not part of designing the flow spec data structure.

References

   1. Parekh, A., "A Generalized Processor Sharing Approach
      to Flow Control in Integrated Services Networks",
      MIT Laboratory for Information and Decision Systems,
      Report No. LIDS-TH-2089.

   2. Clark, D., Shenker, S., and L. Zhang, "Supporting Real-Time
      Applications in an Integrated Services Packet Network:
      Architecture and Mechanism", Proceedings of ACM SIGCOMM '92,
      August 1992.

Author's Address

   Craig Partridge
   BBN
   824 Kipling St
   Palo Alto, CA 94301

   Phone: 415-325-4541

   EMail: craig@aland.bbn.com