Network Working Group<br>Request for Comments: 1046

W. Prue
J. Postel
ISI

February 1988

# A Queuing Algorithm to Provide Type-of-Service for IP Links

## Status of this Memo

This memo is intended to explore how Type-of-Service might be implemented in the Internet. The proposal describes a method of queuing which can provide the different classes of service. The technique also prohibits one class of service from consuming excessive resources or excluding other classes of service. This is an "idea paper" and discussion is strongly encouraged. Distribution of this memo is unlimited.

## Introduction

The Type-of-Service (TOS) field in IP headers allows one to chose from none to all the following service types; low delay, high throughput, and high reliability. It also has a portion allowing a priority selection from 0-7. To date, there is nothing describing what should be done with these parameters. This discussion proposes an approach to providing the different classes of service and priorities requestable in the TOS field.

## Desired Attributes

We should first consider how we want these services to perform. We must first assume that there is a demand for service that exceeds current capabilities. If not, significant queues do not form and queuing algorithms become superfluous.

The low delay class of service should have the ability to pass data through the net faster than regular data. If a request is for low delay class of service only, not high throughput or high reliability, the Internet should provide low delay for relatively less throughput, with less than high reliability. The requester is more concerned with promptness of delivery than guaranteed delivery. The Internet should provide a Maximum Guaranteed Delay (MGD) per node, or better, if the datagram successfully traverses the Internet. In the worst case, a datagram’s arrival will be MGD times the number of nodes traversed. A node is any packet switching element, including IP gateways and ARPANET IMP’s. The MGD bound will not be affected by the amount of traffic in the net. During non-busy hours, the delay provided should be better than the guarantee. If the delay a satellite link introduces is less than the MGD, that link should be considered in the route. If however, the MGD is less than the satellite link can provide, it should not be used. For this discussion it is assumed that delay for individual links are low enough that a sending node can provide the MGD service.

Low delay class of service is not the same as low Round Trip Time (RTT). Class of service is unidirectional. The datagrams responding to low delay traffic (i.e., Acking the data) might be sent with a high reliability class of service, but not low delay.

The performance of TCP might be significantly improved with an accurate estimate of the round trip time and the retransmission timeout. The TCP retransmission timeout could be set to the maximum delay for the current route (if the current route could be determined). The timeout value would have to be redetermined when the number of hops in the route changes.

High throughput class of service should get a large volume of data through the Internet. Requesters of this class are less concerned with the delay the datagrams have crossing the Internet and the reliability of their delivery. This type of traffic might be served well by a satellite link, especially if the bandwidth is high. Another attribute this class might have is consistent one way traversal time for a given burst of datagrams. This class of service will have its traversal times affected by the amount of Internet load. As the Internet load goes up, the throughput for each source will go down.

High reliability class of service should see most of its datagrams delivered if the Internet is not too heavily loaded. Source Quenches (SQ) should not be sent only when datagrams are discarded. SQs should be sent well before the queues become full, to advise the sender of the rate that can be currently supported.

Priority service should allow data that has a higher priority to be queued ahead of other lower priority data. It is important to limit the amount of priority data. The amount of preemption a lower priority datagram suffers must also be limited.

It is assumed that a queuing algorithm provides these classes of service. For one facility to be used over another, that is, making different routing decisions based upon the TOS, requires a more sophisticated routing algorithm and larger routing database. These issues are not discussed in this document.

## Applications for Class of Service

The following are examples of how classes of service might be used. They do not necessarily represent the best choices, but are presented only to illustrate how the different classes of service might be used to advantage.

Interactive timesharing access using a line-at-a-time or character-at-a-time terminal (TTY) type of access is typically low volume typing speed input with low or high volume output. Some Internet applications use echoplex or character by character echoing of user input by the destination host. PC devices also have local files that may be uploaded to remote hosts in a streaming mode. Supporting such traffic can require several types of service. User keyboard input should be forwarded with low delay. If echoplex is used, all user characters sent and echoed should be low delay to minimize the echoing delay. The computer responses should be regular or high throughput depending upon the volume of data sent and the speed of the output device. If the computer response is a single datagram of data, the user should get low delay for the response, to minimize the human/computer interaction time. If however the output takes a while to read and digest, low delay computer responses are a waste of Internet resources. When streaming input is being sent the data should be sent requesting high throughput or regular class of service.

The IBM 3270 class of terminals typically have traffic volumes greater than TTY access. Echoplex is not needed. The output devices usually handle higher speed output streams and most sites do not have the ability to stream input. Input is typically a screen at a time, but some PC implementations of 3270 use a variation of the protocol to effectively stream in volumes of data. Low delay for low volume input and output is appropriate. High throughput is appropriate for the higher volume traffic.

Applications that transfer high volumes of data are typically streaming in one direction only, with acks for the data, on the return path. The data transfer should be high throughput and the acks should probably be regular class of service. Transfer initiation and termination might be served best with low delay class of service.

Requests to, and responses from a time service might use low delay class of service effectively.

These suggestions for class of service usage implies that the application sets the service based on the knowledge it has during the session. Thus, the application should have control of this setting dynamically for each send data request, not just on a per session/conversation/transaction basis. It would be possible for the transport level protocol to guess (i.e., TCP), but it would be sub-optimal.

## Algorithm

When we provide class of service queuing, one class may be more desirable than the others. We must limit the amount of resources each class consumes when there is contention, so the other classes may also operate effectively. To be fair, the algorithm provides the requested service by reducing the other service attributes. A request for multiple classes of service is an OR type of request not an AND request. For example, one can not get low delay and high throughput unless there is no contention for the available resources.

## Low Delay Queuing

To support low delay, use a limited queue so requests will not wait longer than the MGD on the queue. The low delay queue should be serviced at a lower rate than other classes of service, so low delay requests will not consume excessive resources. If the number of low delay datagrams exceeds the queue limit, discard the datagrams. The service rate should be low enough so that other data can still get through. (See discussion of service rates below.) Make the queue limit small enough so that, if the datagram is queued, it will have a guaranteed transit time (MGD). It seems unlikely that Source Quench flow control mechanisms will be an effective method of flow control because of the small size of the queue. It should not be done for this class of service. Instead, datagrams should just be discarded as required. If the bandwidth or percentage allocated to low delay is such that a large queue is possible (see formula below), SQs should be reconsidered.

The maximum delay a datagram with low delay class of service will experience (MGD), can be determined with the following information:

N = Queue size for low delay queue
P = Percentage of link resources allocated to low delay
R = Link rate (in datagrams/sec.)

$$\text{Max Delay} = \frac{N}{P * R}$$

If Max Delay is held fixed, then as P and R go up, so does N. It is probable that low delay service datagrams will prove to be, on the average, smaller than other traffic. This means that the number of datagrams that can be sent in the allocated bandwidth can be larger.

## High Reliability Queuing

To support high reliability class of service, use a queue that is longer than normal (longer queue means higher potential delay). Send SQ earlier (smaller percentage of max queue length) and don't discard datagrams until the queue is full. This queue should have a lower service rate than high throughput class of service.

Users of this class of service should specify a Time-to-Live (TTL) which is made appropriately longer so that it will survive longer queueing times for this class of service.

This queuing procedure will only be effective for Internet unreliability due to congestion. Other Internet unreliability problems such as high error rate links or reliability features such as forward error correcting modems must be dealt with by more sophisticated routing algorithms.

## High Throughput Queuing

To support high throughput class of service have a queue that is treated like current IP queuing. It should have the highest service rate. It will experience higher average through node delay than low delay because of the larger queue size.

Another thing that might be done, is to keep datagrams of the same burst together when possible. This must be done in a way that will not block other traffic. The idea is to deliver all the data to the other end in a contiguous burst. This could be an advantage by allowing piggybacking acks for the whole burst at one time. This makes some assumptions about the overlying protocol which may be inappropriate.

## Regular Service Queuing

For datagrams which request none of the three classes of service, queue the datagrams on the queue representing the least delay between the two queues, the high throughput queue or the high reliability queue. If one queue becomes full, queue on the other. If both queues are full, follow the source quench procedure for regular class of service (see RFC-1016), not the procedure for the queue the datagram failed to attain.

In the discussion of service rates described below, it is proposed that the high throughput queue get service three times for every two times for the high reliability queue. Therefore, the queue length of the high reliability queue should be increased by 50% (in this example) to compare the lengths of the two queues more accurately. A simplification to this method is to just queue new data on the queue that is the shortest. The slower service rate queue will quickly exceed the size of the faster service rate queue and new data will go on the proper queue. This however, would lead to more packet reordering than the first method.

## Service Rates

In this discussion, a higher service rate means that a queue, when non-empty, will consume a larger percentage of the available bandwidth than a lower service rate queue. It will not block a lower service rate queue even if it is always full.

For example, the service pattern could be; send low delay 17% of the time, high throughput 50% of the time, and high reliability 33% of the time. Throughput requires the most bandwidth and high reliability requires medium bandwidth. One could achieve this split using a pattern of L, R,R, T,T,T, where low delay is "L", high reliability is "R", and high throughput is "T’. We want to keep the high throughput datagrams together. We therefore send all of the high throughput data at one time, that is, not interspersed with the other classes of service. By keeping all of the high throughput data together, we may help higher level protocols, such as TCP, as described above. This would still be done in a way to not exceed the allowed service rate of the available bandwidth.

These service rates are suggestions. Some simplifications can be considered, such as having only two routing classes; low delay, and other.

## Priority

There is the ability to select 8 levels of priority 0-7, in addition to the class of service selected. To provide this without blocking the least priority requests, we must give preempted datagrams frustration points every time a higher priority request cuts in line in front of it. Thus if a datagram with low priority waits, it will always get through even when competing against the highest priority requests. This assumes the TTL (Time-to-Live) field does not expire.

When a datagram with priority arrives at a node, the node will queue the datagram on the appropriate queue ahead of all datagrams with lower priority. Each datagram that was preempted gets its priority raised (locally). The priority data will not bump a lower priority datagram off its queue, discarding the data. If the queue is full, the newest data (priority or not) will be discarded. The priority preemption will preempt only within the class of service queue to which the priority data is targeted. A request specifying regular class of service, will contend on the queue where it is placed, high throughput or high reliability.

An implementation strategy is to multiply the requested priority by 2 or 4, then store the value in a buffer overhead area. Each time the datagram is preempted, increment the value by one. Looking at an example, assume we use a multiplier of 2. A priority 6 buffer will have an initial local value of 12. A new priority 7 datagram would have a local value of 14. If 2 priority 7 datagrams arrive, preempting the priority 6 datagram, its local value is incremented to 14. It can no longer be preempted. After that, it has the same local value as a priority 7 datagram and will no longer be preempted within this node. In our example, this means that a priority 0 datagram can be preempted by no more than 14 higher priority datagrams. The priority is raised only locally in the node. The datagram could again be preempted in the next node on the route.

Priority queuing changes the effects we were obtaining with the low delay queuing described above. Once a buffer was queued, the delay that a datagram would see could be determined. When we accepted low delay data, we could guarantee a certain maximum delay. With this addition, if the datagram requesting low delay does not also request high priority, the guaranteed delay can vary a lot more. It could be 1 up to 28 times as much as without priority queuing.

## Discussion and Details

If a low delay queue is for a satellite link (or any high delay link), the max queue size should be reduced by the number of datagrams that can be forwarded from the queue during the one way delay for the link. That is, if the service rate for the low delay queue is L datagrams per second, the delay added by the high delay link is D seconds and M is the max delay per node allowed (MGD) in seconds, then the maximum queue size should be:

```
      Max Queue Size = L ( M - D),  M > D
                     = 0         ,  M <= D
```

If the result is negative (M is less than the delay introduced by the link), then the maximum queue size should be zero because the link could never provide a delay less than the guaranteed M value. If the bandwidth is high (as in T1 links), the delay introduced by a terrestrial link and the terminating equipment could be significant and greater than the average service time for a single datagram on the low delay queue. If so, this formula should be used to reduce the queue size as well. Note that this is reducing the queue size and is not the same as the allocated bandwidth. Even though the queue size is reduced, the chit scheme described below will give low delay requesters a chance to use the allocated bandwidth.

If a datagram requests multiple classes of service, only one class can be provided. For example, when both low delay and high reliability classes are requested, and if the low delay queue is full, queue the data on the high reliability queue instead. If we are able to queue the data on the low delay queue, then the datagram gets part of the high reliability service it also requested, because, once data is queued, data will not be discarded. However, the datagram will be routed as a low delay request. The same scheme is used for any other combinations of service requested. The order of selection for classes of service when more than one is requested would be low delay, high throughput, then high reliability. If a block of datagrams request multiple classes of service, it is quite possible that datagram reordering will occur. If one queue is full causing the other queue to be used for some of the data, data will be forwarded at different service rates. Requesting multiple classes of service gives the data a better chance of making it through the net because they have multiple chances of getting on a service queue. However, the datagrams pay the penalty of possible reordering and more variability in the one way transmission times.

Besides total buffer consumption, individual class of service queue sizes should be used to SQ those asking for service except as noted above.

A request for regular class of service is handled by queuing to the high reliability or high throughput queues evenly (proportional to the service rates of queue). The low delay queue should only receive data with the low delay service type. Its queue is too small to accept other traffic.

Because of the small queue size for low delay suggested above, it is difficult for low delay service requests to consume the bandwidth allocated. To do so, low delay users must keep the small queue continuously non-empty. This is hard to do with a small queue. Traffic flow has been shown to be bursty in nature. In order for the low delay queue to be able to consume the allocated bandwidth, a count of the various types being forwarded should be kept. The service rate should increase if the actual percentage falls too low for the low delay queue. The measure of service rates would have to be smoothed over time.

While this does sound complicated, a reasonably efficient way can be described. Every Q seconds, where Q is less than or equal to the MGD, each class gets N M P chits proportional to their allowed percentage. Send data for the low delay queue up to the number of chits it receives decrementing the chits as datagrams are sent. Next send from the high reliability queue as many as it has chits for. Finally, send from the high throughput queue. At this point, each queue gets N M P chits again. If the low delay queue does not consume all of its chits, when a low delay datagram arrives, before chit replenishment, send from the low delay queue immediately. This provides some smoothing of the actual bandwidth made available for low delay traffic. If operational experience shows that low delay requests are experiencing excessive congestion loss but still not consuming the classes allocated bandwidth, adjustments should be made. The service rates should be made larger and the queue sizes adjusted accordingly. This is more important on lower speed links where the above formula makes the queue small.

What we should see during the Q seconds is that low delay data will be sent as soon as possible (as long as the volume is below the allowed percentage). Also, the tendency will be to send all the high throughput datagrams contiguously. This will give a more regular measured round trip time for bursts of datagrams. Classes of service will tend to be grouped together at each intermediate node in the route. If all of the queues with datagrams have consumed all of their allocated chits, but one or more classes with empty queues have unused chits then a percentage of these left over chits should be carried over. Divide the remaining chit counts by two (with round down), then add in the refresh chit counts. This allows a 50% carry over for the next interval. The carry over is self limiting to less than or equal to the refresh chit count. This prevents excessive build up. It provides some smoothing of the percentage allocation over time but will not allow an unused queue to build up chits indefinitely. No timer is required.

If only a simple subset of the described algorithm is to be implemented, then low delay queuing would be the best choice. One should use a small queue. Service the queue with a high service rate but restrict the bandwidth to a small reasonable percentage of the available bandwidth. Currently, wide area networks with high traffic volumes do not provide low delay service unless low delay requests are able to preempt other traffic.

## Applicability

When the output speed and volume match the input speed and volume, queues don’t get large. If the queues never grow large enough to exceed the guaranteed low delay performance, no queuing algorithm other than first in, first out, should be used.

The algorithm could be turned on when the main queue size exceeds a certain threshold. The routing node can periodically check for queue build up. This queuing algorithm can be turned on when the maximum delays will exceed the allowed nodal delay for low delay class of service. It can also be turned off when queue sizes are no longer a problem.

## Issues

Several issues need to be addressed before type of service queuing as described should be implemented. What percentage of the bandwidth should each class of service consume assuming an infinite supply of each class of service datagrams? What maximum delay (MGD) should be guaranteed per node for low delay datagrams?

It is possible to provide a more optimal route if the queue sizes for each class of service are considered in the routing decision. This, however, adds additional overhead and complexity to each routing node. This may be an unacceptable additional complexity.

How are we going to limit the use of more desirable classes of service and higher priorities? The algorithm limits use of the various classes by restricting queue sizes especially the low delay queue size. This helps but it seems likely we will want to instrument the number of datagrams requesting each Type-of-Service and priority. When a datagram requests multiple classes of service, increment the instrumentation count once based upon the queue actually used, selecting, low delay, high throughput, high reliability, then regular. If instrumentation reveals an excessive imbalance, Internet operations can give this to administrators to handle. This instrumentation will show the distribution for types of service requested by the Internet users. This information can be used to tune the Internet to service the user demands.

Will the routing algorithms in use today have problems when routing data with this algorithm? Simulation tests need to be done to model how the Internet will react. If, for example, an application requests multiple classes of service, round trip times may fluctuate significantly. Would TCP have to be more sophisticated in its round trip time estimator?

An objection to this type of queuing algorithm is that it is making the routing and queuing more complicated. There is current interest in high speed packet switches which have very little protocol overhead when handling/routing packets. This algorithm complicates not simplifies the protocol. The bandwidth being made available is increasing. More T1 (1.5 Mbps) and higher speed links are being used all the time. However, in the history of communications, it seems that the demand for bandwidth has always exceeded the supply. When there is wide spread use of optical fiber we may temporarily experience a glut of capacity. As soon as 1 gigabit optical fiber link becomes reasonably priced, new applications will be created to consume it all. A single full motion high resolution color image system can consume, as an upper limit, nearly a gigabit per second channel (30 fps X 24 b/pixel X 1024 X 1024 pixels).

In the study of one gateway, Dave Clark discovered that the per datagram processing of the IP header constituted about 20% of the processing time. Much of the time per datagram was spent on restarting input, starting output and queuing datagrams. He thought that a small additional amount of processing to support Type-of-Service would be reasonable. He suggests that even if the code does slow the gateway down, we need to see if TOS is good for anything, so this experiment is valuable. To support the new high speed communications of the near future, Dave wants to see switches which will run one to two orders of magnitude faster. This can not be done by trimming a few instructions here or there.

From a practical perspective, the problem this algorithm is trying to solve is the lack of low delay service through the Internet today. Implementing only the low delay queuing portion of this algorithm would allow the Internet to provide a class of service it otherwise could not provide. Requesters of this class of service would not get it for free. Low delay class of datagram streams get low delay at the cost of reliability and throughput.