Review: Quick-Start for TCP and IP [JFAS05]

Bob Briscoe
<bob.briscoe@bt.com> BT Research and UCL,
B54/77, Adastral Park, Martlesham Heath, Ipswich, IP5 3RE, UK

November 15, 2005

Abstract

This review thoroughly analyses draft 01 of the Quick-Start proposal, focusing mostly on security issues. It is argued that the recent new QS nonce proposal gives insufficient protection against misbehaving receivers, and a new approach is suggested. But it would be perverse to strengthen protection against malicious receivers too much when the protocol only works if all senders can be trusted to comply. The review argues this is an inevitable result of choosing to have routers allocate rate to senders without keeping per-flow state. The paper also questions whether Quick-Start's under-utilisation assumption defines a distinct range of operation where fairness can be ignored. Because traffic variance will always blur the boundary, we argue that under-utilisation should be treated as the extreme of a spectrum where fairness is always an issue to some extent.

If we are to avoid per-flow state on routers, the review points to an alternative direction where endpoints allocate rate to themselves. Counter-intuitively, this allows scalable security and a spectrum of fairness to be built in from the start, but rate allocation is less deterministic.

Issues not related to security are also raised, including the possibility of a catastrophic overload if path delays are atypical. A solution to this is offered, as well as solutions to scalability issues with the range and precision of the Rate Request field. Many other more minor review comments are given.

1 Introduction

This is a review of the second IETF Transport Area working group draft (01) of Jain *et al*'s Quick-Start [JFAS05]. The review was originally of the 00 draft, but an attempt has been made to update it to take account of changes in the 01 draft. I apologise if there are still complaints about things that have already been fixed between 00 and 01—I have tried to check all the changes, but may have missed some. External references, for instance to sections of the Quick-Start I-D under review, are rendered like '§1', while internal references to other sections of this review itself are rendered like 'Section 1'. Identical body text has been incorporated into an IETF Internet Draft [Bri05].

Although the draft is long (76pp) and growing, it still refers at length to a supporting document by Sarolahti, Allman & Floyd [SAF05], which is still under submission. This paper only reviews the material in the Internet Draft, not the supporting document, on the basis that, if any details were intended for IETF consideration they would have been included in the Internet Draft. For instance, the draft says that Sarolahti *et al* presents a number of alternative router algorithms, but it is assumed that the single example given in the draft is representative, at least in terms of intent, if not implementation. The draft also claims Sarolahti *et al* presents an Extreme Quick-Start mechanism with per-flow state, but given that mechanism is not presented in the draft, and is described as 'extreme', it is assumed that it is not considered applicable for consideration by the IETF.

The final sentence of §A.6 says, "...as long as the simple mechanisms are not short-term hacks but mechanisms that lead the overall architecture in the fundamentally correct direction." The position of this review will be made clear from the start. Adding rate allocation to interior routers is not considered likely to be the fundamentally correct direction we should take. Section 3.1 justifies this position, but an overview is given here. By extension, this is also a disagreement with the similar architecture used in XCP [KHR02], but this review will focus on Quick-Start.

It is tempting to get routers to allocate rates directly because that is the final effect we are trying to achieve. But these approaches leave aside the security issue of determining whether to trust the sender until after the design choices have been made. Then solving that problem adds horrible complexity, because routers in distant networks from the sender have to have a security association with the sender. This same problem was at the root of the scalability problems behind Intserv [BBB+97]. This remote security association could potentially be aggregated so a network only needs a security associations with neighbouring networks and hosts. But doing that isn't easy. For instance, when Diffserv did that, it had to compromise by losing precision.

However, Quick-Start is a valid experimental direction to try out what happens when flows start fast, without concern about security issues. And, as it doesn't preclude any other future direction (other than using up one more IP Option codepoint and one more TCP Option codepoint, which is not a problem because they are not scarce), its move to experimental status shouldn't be opposed as long as its applicability is made absolutely clear. Then Quick-Start experiments can proceed in parallel to further research into how to securely allow senders to start quickly, which is already on the agenda of the IRTF Internet congestion control research group (ICCRG) [Han05].

So this review is divided into three main parts:

- Argumentation about basic design choices taken
- A list of suggested technical improvements setting aside disquiet about basic design choices
- Suggestions to improve clarity and simple typos.

2 Summary of Quick-Start

The Quick-Start protocol allows a Quick-Start sender to request to be allowed to directly start a flow at a high initial rate, rather than slowly probing for network capacity using TCP slow-start. The protocol is intended for controlled environments where all routers are usually under-utilised. The request uses a single hop-by-hop pass from sender to receiver requesting a rate from each router on the way using an IPv4 Option header or an IPv6 extension header. Then the receiver returns the response back to the sender in an e2e TCP Option message. When a request arrives at a router, it determines the initial rate it can support and if necessary reduces the Rate Request field in the protocol to this rate before forwarding it onward. The protocol is designed to detect if any router on the path or the destination is not Quick-Start capable and if so falls back to standard TCP slow-start. A nonce scheme is included to allow the sender to detect if the receiver has tried to feed back a higher allowed rate than it was given by the network.

3 Arguments against basic design choices taken

Quick-Start routers get the receiver to tell the source what the network can support. But they don't check whether what the source subsequently sends is what it was told it could send. And they can't as they are not required to be stateful. So we assume Quick-Start should also be confined to environments where every source is trusted. This security assumption isn't discussed in the Quick-Start I-D, but this review assumes the authors would agree it is their assumption. A large part of this review is devoted to analysing this issue of trust in the sender.

The objection may be made by the authors that other IETF protocols have not had to pass this test. TCP itself relies on trust in the sender. The difference with Quick-Start is that it raises considerable issues of fairness (the Quick-Start I-D disagrees, but Section 3.4 below justifies this statement), so it should require the same protections against sender misbehaviour that are expected of other protocols for differential QoS.

3.1 The metric: Capacity or impairment?

Approaches for resource allocation can be divided into those where endpoints ask routers for a rate, and those where routers tell endpoints their overall state so that endpoints may choose a rate. Quick-Start falls into the former category. If each router divides its available capacity among competing requests, every router must associate the 'identity' making the request with some sharing policy. In the case of Quick-Start, the sharing policy is that available capacity is shared equally among all the requests, but other policies can be chosen. However, the problem is how to recognise different 'identities'. If routers are not required to be stateful, each new request has to be defined as from a different 'identity'. Another way to say this is that the authorisation model of Quick-Start routers has to be very trusting if they are to be stateless.

That is fine for a trusted environment. But it is unlikely to "lead the overall architecture in the fundamentally correct direction" (§A.6) for when we want to move to an untrusted environment. The goal of the IETF (and those that use its standards) is to build in security from the start.

Clearly, the stateless model of Quick-Start is vulnerable to repeat requests being assumed to be multiple identities, so sources can get more, simply by asking more often (or, for that matter, by simply taking more without asking)... But that is only the first part of the problem.

The main objection is that even if we did countenance stateful routers, if rate must be allocated on each router, every router (or at least every trust domain) has to establish the identity behind each request. For simplicity, perhaps identity could be based on addressing (flow IDs), rather than cryptographic identifiers (signatures). But we all know the pitfalls of using addressing for security identification. Specifically, in the case of rate allocation we are concerned with sender spoofing and identity splitting:

Sender spoofing To allocate rates to senders, routers would need to identify the sender, but for a message to reach a receiver, the sender only needs to truthfully reveal the receiver's address to the network. If it wants the destination to correspond, it need only reveal its own address to the destination, not to the network.

Identity splitting If rates are allocated to unique addresses (more generally tuples of addresses), it is relatively easy for the source and destination to conspire to split their identity over multiple addresses between themselves (or via proxy interfaces or hosts) to get more rate.

In other words, even stateful Quick-Start routers lead us to per-flow policing at every trust border—the same security premise that Intserv started from. So it seems likely Quick-Start will eventually suffer the same fate as Intserv if it takes this path.

So what could "lead the overall architecture in the fundamentally correct direction"? A better hook for endpoint identification at the network/transport layer is to associate a sender with its ingress attachment interface. This sounds obvious, but it still seems hard to be able to identify the source of a flow with precision as it arrives at a router multiple networks downstream. However, recent research has shown how to do this scalably. The direction outlined below arguably has better potential for solving this security problem. It should allow responsibility for flows to be taken on recursively by each network as it presents its requests in bulk to the next network, without any per-flow processing in the network interior, even at borders.

This new approach requires routers to declare their 'impairment' status to endpoints. An example of impairment status is their level of congestion (e.g. ECN), which is a measure of the risk that a packet will not be served. For under-utilised scenarios like that of Quick-Start, impairment would have to be relative to a lower 'bar' than that used for congestion control. In the case of speeding up TCP slow-start, the metric might measure the risk that a router will no longer be under-utilised. Xia *et al* [XSSK05] is an example of using this mechanism to solve the same problem as Quick-Start. The virtual queue used in AntiECN [Kun03] and pre-congestion notification [BES+05] are other examples.

Current research has not reached consensus on the best pre-congestion metric to use. This is why we need to research this issue in the recently set-up IRTF Congestion Control Research Group (CCRG). But this is likely to be an easier issue to solve than fixing the basic security model of rate allocation schemes like Intserv, Quick-Start and XCP—solutions to the problem of identifying accountability for congestion don't fall out of the sky so often as new congestion control schemes.

A field for this impairment status would then be provided in every packet passing through each resource. It would need a few more bits than ECN in order to feed back the path's impairment status quickly to the source, rather than having to code a probability over hundreds of packets using binary marking. Then as packets pass through other resources they accumulate the impairment status of the path. ECN [RFB01] is an example of this model, but for pre-congestion, it is necessary to move the goal posts (the desired utilisation level) downward.

We then have impairment information intrinsically associated with rate information, so the two can be traded against each other—cost and benefit. The rate information *is* the number of bits in packets a source sends. And because the impairment information is carried in each packet, the more packets a source sends, the more impairment can be associated with that source (just as with ECN).

But we haven't solved the problem yet. Rate is measurable all along a network path, and it aggregates and de-aggregates nicely. But although impairment information aggregates and de-aggregates too, it accumulates along the path. So, in a connectionless datagram network, impairment seems to be only measurable just before the destination. With direct rate allocation, we said the problem was that every router had to have a security association with the source. But our egress still seems to need a security association with the source (because we can't assume the network can intercept feedback), which isn't much better. And even if the egress can identify the source, it's in the wrong place to allocate the rate to the source, because it can't police whether the source is complying...

But we can solve this problem by the source having to declare the impairment it believes will accumulate along the path [BJCG+05, BJS05]. Then rather than accumulating impairment state along the path, routers effectively subtract it. With sufficient bits in the pre-congestion field, after at least one round trip the source can make an accurate prediction of what it should declare in the next round. If the average of the declarations ends up negative at the end, the egress can tell there must have been cheating upstream and drop the packets.

So we have now collected all three items of required information at the ingress to the network: source identity, impairment and rate. Identifying the source has become solely the job of the ingress network, based on the attachment interface of the source. So the identification scalability problem has been solved. And the ingress network can also check that the source is doing a proper cost-benefit trade-off between impairment and rate.

The downside is that such a system does not allocate rate deterministically. It is statistical. But without per-flow state, Quick-Start suffers from that failing too, though not as seriously (Section 4.1). The question for the working group is whether improved determinism is more important than improved security, or vice versa.

The argument for the more secure approach isn't easy to grasp as it is constructed from a number of steps that have not been put together into a detailed approach for the problem of a faster start to TCP flows. All we currently have is the outline above and the references below. But it seems much more likely to lead the architecture in a future-proof direction. This architectural argument was outlined in a discussion of flow-start incentives in [BJCG+05, §3.3.3]. The same overall approach is also used in the Internet-Draft on Re-ECN [BJS05, §1], but in this case for TCP congestion control rather than pre-congestion control.

Later this review criticises the Quick-Start I-D for depending overmuch on references out to [SAF05], while this review is itself open to criticism for hypocrisy given it also refers out overmuch. The spirit of this review is to point out advances in very recent research that offer what appears to be a more fruitful architectural direction. It is not trying to prevent Quick-Start progressing; it is however wanting applicability to be clarified first.

3.2 Flow state and no flow state as distinct scenarios

§2 clearly says "No per-flow state should be required at the router. Note that while per-flow state is not required we also do not preclude a router from storing per-flow state for making Quick-Start decisions." However, there is a phrase at the end of §9.4.3 on Collusion between Misbehaving Routers saying, "the router between the ingress and egress nodes that denied the request could be monitoring connection performance, actively penalizing nodes that seem to be using Quick-Start after a Quick-Start request was denied."

It is true that flow state is NOT REQUIRED for Quick-Start to work in a trusted environment (but see Section 4.1). However, it seems flow-state is REQUIRED in an untrusted environment. If the draft confined itself to the trusted environment assumption, it wouldn't have to slip between flow-state being REQUIRED and NOT REQUIRED. Because if the draft is going to start entertaining the untrusted scenario, there are problems of state exhaustion attacks to deal with, if flow state is REQUIRED. So the authors need to either keep the untrusted scenario out of this draft, or bring it in completely.

3.3 No secure association between control and data

§9.4 or §9.5: There is no association (binding), let alone a secure association between the request/response and the subsequent data it allows. Therefore, it becomes virtually impossible for the network to somehow police the sender to ensure compliance with the response.

Even if there were a secure binding, because of the protocol model of a single pass towards the destination followed by e2e feedback, an upstream router cannot defend a downstream router that has reduced the request, given the response isn't seen again by upstream routers.

Worse, the lack of a secure binding between a request and subsequent traffic means that any other node can send a burst of traffic and claim it requested it, with no-one being able to prove it didn't. This problem would preclude a deployment of Quick-Start even if it were confined to a controlled subset of the sources, if other sources that were not part of the experiment could send unaccountable bursts of traffic claiming to be from the authorised sources.

Note that this is not an argument that the binding must be secured by cryptography. In Section 3.1 there is an alternative way to do a secure binding based on physical connectivity[1].

3.4 Fairness issues

§A.6: "...More Functionality?" says "...for a mechanism for requesting a initial sending rate in an underutilized environment, the fairness issues of a general congestion control mechanism go away,..." This is unfortunately not true.

TCP is designed to seek out the maximum capacity it can on the path. So in an under-utilised environment long-lived TCP flows will continue to rise in rate until they find congestion. But they will finish sooner. So with TCP, under-utilised means there are insufficient long-lived flows to fill capacity and shorter flows end before they have reached congestion. But it doesn't mean that instantaneous utilisation is always low. It only means average utilisation is low. With TCP, the variance of instantaneous utilisation increases greatly in an under-utilised network. Even if traffic is dominated by shorter-lived flows, there will be peaks in congestion as flow arrivals coincide. So the network continually moves rapidly from under-utilisation to congestion.

So in an under-utilised environment, fairness splits into three issues, the last two of which only appear each time we cross the boundary between instantaneous under-utilisation and instantaneous congestion:

- whether routers give differential responses (see next section 3.5)
- who has most bandwidth during brief periods of congestion, when multiple flows happen to coincide (variance of congestion over time)
- how the risk of congestion varies with path length (variance of congestion in space)

Congestion variance over time: At the boundary between a time of under-utilisation and one where congestion starts to set in, fairness depends on who asked for most bandwidth before everyone realises congestion has started. Those flows with most on entry to the period of congestion will be in the most advantageous position during the period of congestion (when we assume normal congestion avoidance takes over). With multiplicative decrease, the higher you start from, the further you fall, but you are still higher after each round trip than everyone else.

[1] Wireless connectivity can still be a problem, but local physical range constraints or link-local cryptographic authentication can solve this without a global PKI.

Congestion variance in space: The single-pass request (with rate allocation being hop by hop along the path, rather than after the request has traversed the whole path) precludes any fairness models that require the rate allocation function to know the state of the whole path. A router only knows what the upstream path can sustain and its own local condition. So rate allocation cannot weigh benefits (rate) against costs (risk of congestion). So Quick-Start is limited to benefit-only fairnesses like max-min or min-max, and cannot ever achieve cost-benefit fairnesses like proportional fairness or the root-proportional fairness of algorithms like TCP. In other words, Quick-Start can only share out benefits (rate) in various ways, without regard to costs (risk of congestion). The under-utilisation assumption essentially says that the risk of congestion is close enough to zero to be negligible (but see Section 3.6).

This approach takes no account of the increased risk of congestion the more routers a Quick-Start flow will traverse, even in an under-utilised network. What matters is the risk of congestion across a path, not the utilisation of each individual router. We show below that every QS router on a path can think it is individually below the threshold, but the flow is on a path above the threshold. So all the QS routers give the source the go-ahead when they shouldn't.

If the probability of congestion is p at every router (conveniently taken to be all the same), then the probability, P, of congestion across a path of n routers is $P = 1 - (1 - p)^n$. For low levels of path congestion ($P \ll 1$), the risk of congestion across a path is the sum of the risks of congestion on each router $P \approx \sum_n p$ (Table 1).

p \ n	1	2	5	10	20
1.000%	1.000%	1.990%	4.901%	9.562%	18.209%
0.100%	0.100%	0.200%	0.499%	0.996%	1.981%
0.010%	0.010%	0.020%	0.050%	0.100%	0.200%
0.001%	0.001%	0.002%	0.005%	0.010%	0.020%

Table 1: Probability of congestion across a path of n routers, each with probability of congestion p.

The Quick-Start protocol finds the minimum of the rates each router allows, solely considering itself in isolation. The protocol precludes a combined view across the path. A Quick-Start router denies Quick-Start requests once its local utilisation is above a threshold. It should, but cannot, take account of whether the risk of whole-path congestion is above a threshold. For instance, if the path congestion threshold for allowing Quick-Start requests were 0.020%, then Quick-Start requests should only be honoured for the cases below the staircase in Table 1.

Given Quick-Start is more useful for long RTT paths (where TCP slow-start takes longer), the above problem is more likely to occur wherever Quick-Start is most useful.

Incidentally, unfairness attacks aren't mentioned in §9.5. For instance, the router's response gapping defence against a type (1) attack, where a source increases the router's processing and state load, still reduces the number of successful responses to well-behaved nodes, which in turn gives them an incentive to fight fire with fire by increasing their rate of requests.[2] Identity splitting (Section 3.1) is another form of unfairness attack.

So fairness is an issue. And, as a consequence, I believe there will be pressures and incentives for people to have differential treatment from schemes like Quick-Start (see Section 3.5 next). Instead of saying, "...there are fewer open issues with Quick-Start..." the I-D should say, "...Quick-Start is targeted at an experimental environment where the more intractable issues can be set aside". The problem is that the chosen scenario has boundaries that the draft recognises will regularly be crossed in any practical network, even if it conforms to the assumptions most of the time.

In Section 3.8 I will return to the issue of how a device (host or router) knows whether it is on a path where the assumptions are currently valid.

[2] This requires some way to prevent each source from frequently repeating requests (the draft only discusses how a router can enforce gapping for the aggregate of signals, not how to discriminate against particularly persistent sources. See also RSVP blockade-state or call-gapping in the PSTN).

3.5 Quick-Start and QoS

§A.4 says, "The Total Rate semantics makes it easier for routers to "allocate" the same rate to all connections." One person's logic might say the obvious form of fairness is equality, so allocating the same rate to all is the desirable behaviour. Another person might say the obvious form of equality is to allocate rates in proportion to how valuable the customer is, or how much they pay. As far as I know, no other IETF protocol allocates rates equally. They either allocate the product of rate with RTT and the root of congestion equally (TCP-fairness), or they don't allocate rate (UDP). The caution in Tussle in Cyberspace [CSWB02] should be listened to at this point. If the design choices behind a protocol intrinsically only give network operators the ability to treat their customers equally, they will break its architecture to be able to sell inequality.

There is a strong likelihood that differentiated Quick-Start responses may arise as a possible business model. As it stands, Quick-Start cannot prevent users selfishly creating this differentiation themselves, because they can create multiple identities for themselves (aside from the fact that a stateless Quick-Start can't even distinguish between requests from the same identity). The approach outlined in Section 3.1 allows service differentiation to be added scalably, as a local arrangement between the sender and its ingress network. It also seems possible to make the approach recursive from one network to the next, in a similar way to that described in [BJCG+05].

But QoS is not just about differentiation within a service. We also have QoS between services. In the previous (00) draft, what is now §9.3 "Quick-Start with QoS-enabled Traffic" said "...routers should be discouraged from granting Quick-Start requests for higher- priority traffic when this is likely to result in significant packet loss for lower-priority traffic." This review originally argued that the the whole point of some higher QoS classes may be to give priority to quick-start flows to the detriment of other classes. However, as this sentence has been removed in the 01 draft, it is assumed that the authors have already realised this point.

3.6 Conceptual model of under-utilisation

Section 3.4 above on fairness raised concerns that the Quick-Start authors' conceptual model of under-utilisation perhaps did not take full account of the increased variance of instantaneous utilisation when TCP (and Quick-Start) traffic dominates an under-utilised environment. This section explores whether there is also an assumption that under-utilisation will always be due to host interface limitations.

One of the under-utilisation assumptions I had in my head while reading the paper was that any one host is generally able to over-fill available capacity, but that, given a high rate, the flow would end quickly. In this case under-utilisation meant that generally one big flow like this would finish before the next arrival would hit any of the same interfaces on the path. Surely Quick-Start should be designed to handle scenarios where a single host can saturate the network capacity? It seems a feasible scenario particularly in super-computing and data-centre type environments where Quick-Start might be useful.[3]

But then, in this 'one big flow at a time' scenario, if the network moves from under-utilisation to moderate utilisation (perhaps on a daily cycle), Quick-Start would have to handle the transition correctly. During the periods of under-utilisation (implying long gaps between these big flows) routers would be able to give each request all the remaining capacity. But during any periods when load might move to medium-utilisation, new requests might arrive more often during the time that a current request was still being served, already filling the remaining capacity.

QS could just do first-come-first-served on the full remaining capacity. Or routers could maintain an average of the arrival rate of new requests relative to the amount of capacity available as each request arrived. Then each router could cut down each new request to give predicted requests arriving during the flow a fairer share of the router's remaining capacity without having to push in using standard slow-start. But usually, during under-utilisation, this sharing algorithm would give each new request the whole of the remaining capacity.

[3]For instance, we once had a request to supply a network for moving very large amounts of astronomy observation data across multiple countries just one day in each month. Here, one host alone could saturate the network path we were considering. This is a trivial example that would be made more realistic by thinking of multiple, but infrequent, competing requests like this across a network.

The point being made is not that sharing of remaining capacity is a correct approach. It certainly may be pragmatic to say it is more important to fill the pipe with known load than leave some spare for predicted load. This review merely asks that the question be considered: whether predictive sharing can be 'better' relative to first-come first served. In turn, the absence of this question might imply an implicit assumption needs to be stated that flow rates are limited by host interface capacities before any interior network link capacities can be saturated by one flow.

The example router algorithm in §C has no notion of sharing capacity—it all goes to each request that arrives. It was only the += operator in the last line of the algorithm in Appendix (§C) that made it clear that division of remaining capacity wasn't being assumed. By reverse engineering this algorithm, it was possible to guess that there was an assumption that host capacity was smaller than the network's, so meeting a request in full would still leave a lot of spare capacity for the next request. This assumption needs to be brought out earlier; not at the end of one of the last appendices (and not by having to reverse-engineer an algorithm). This is a symptom of trying to avoid discussion of router algorithms in the draft (see Section 3.7 below).

As well as medium utilisation, we might very occasionally even get high arrival rates of these big requests. Then nearly every request would get zero capacity. Senders would be repeating requests very frequently (cf. congestion collapse on CSMACD media such as shared ethernet) to try to catch all the routers on the path just as capacity comes available before someone else gets it.[2]

Indeed, in this case of large requests relative to remaining capacity, it seems Quick-Start would (probably unwittingly) move the Internet towards flow-by-flow capacity allocation rather than packet multiplexing. Such radical thinking is not necessarily beyond consideration (for instance, Key and Massoulié make a good case for this mode of operation when transferring fixed volume objects such as data files [KM99]), but the under-utilisation assumption is clearly ambiguous as it stands.

There is probably a lot more about possible router algorithms and the under-utilisation assumptions in [SAF05]. I could not really make full sense of this I-D without having read [SAF05], which implies there is too much reference out to [SAF05] that ought to be included in the I-D.

3.7 Router algorithms purely local policy?

§C: "Possible Router Algorithm" says, "we consider the algorithm a particular router uses to be a local policy decision" Surely this approach is insufficient? Surely this I-D needs to set some constraints on possible router algorithms, to enable interworking. For instance, one network might decide that the larger the balance between the request and available capacity, the smaller the response will be (to discourage senders from asking for more than they need). While another might decide the opposite policy: the more you ask, the more you get. Unless the two co-ordinate, their two policies may fight with unpredictable results for both, or they may depend on the order that two networks deal with a request.

Alternatively, the I-D could say that experiments are needed (hence the need for experimental RFC status) in order to establish constraints required on router algorithms for interworking, robustness, fairness etc.

3.8 Applicability statement

The document doesn't ever mention that the sender is assumed to be acting in the interests of the network. Throughout the history of the Internet this assumption has been made. But that does not mean we are allowed to stop admitting that we are making this assumption.

Indeed, the text at the start of the QS nonce section §3.4 implies that the nonce is protecting the sender, rather than allowing the sender to protect the network if it chooses to. It is essential to clearly highlight such a major security assumption in an applicability section, in a new sub-section of §9.4 on misbehaving senders or in the security considerations section (preferably in all three).

Note that, not only do Quick-Start senders have to be trusted, but also other senders who could claim their data had been authorised by a Quick-Start response when it hadn't (Section 3.3). So the usual allowance for this assumption—that TCP code is embedded in the operating system so is difficult to tamper with—loses its force. One could argue that the receiver's TCP code is also embedded in the

operating system, so why the need for the QS nonce? The usual argument here is that a deployment scenario might be where only large download sites were trusted to use Quick-Start. But because Quick-Start does not require state on routers, it seems hard for QS routers to determine whether large bursts of data were authorised by a previous successful request, let alone whether they are from a trusted source or not.

§10.3: "Possible Deployment Scenarios" implies that a controlled environment is an initial scenario. It should be put more strongly: that a controlled environment is the *only* scenario that could ever be contemplated while there is a need to trust the source. The draft suggests GPRS as a possible scenario, and others have been proposed on the mailing list. It should be clearly stated that this proposal is not, and never will be applicable for general public networks without some means to ensure that the sender can be trusted.

The draft also needs to say how devices will know whether they are currently part of such a controlled environment. A router can be configured to know that it is part of a network where all requests to it are from controlled sources. But how does a trustworthy Quick-Start sender know when it has roamed to a network where the trust assumption doesn't hold so it should give up sending Quick-Start requests, as routers will always ignore them? Otherwise, (trusted) hosts on public networks will be continually sending innocent Quick-Start requests, possibly unnecessarily tying up router resources further along the path where there is a controlled environment.

Quick-Start needs a way for a router to say to a source "Don't bother with Quick-Start any more until you move to another network". Currently, a source may have the Quick-Start request option removed, but it doesn't know whether the ingress router did that on behalf of all possible future connections on this network, or some router further downstream did it, implying only that particular path doesn't support Quick-Start.

We need a way for devices to know the deployment status of what they are attached to. Just because this is a general requirement covering many new capabilities (QoS, multicast, IPv6 etc), doesn't mean it shouldn't be mentioned each time it is needed.

Finally, below are some specific points in the I-D where the trusted sender assumption is lurking implicitly in the background and needs to be brought out explicitly.

- §9.2: The moderate added complexity at routers is only valid if senders can be trusted.
- §9.4.3: Scenarios of collusion between networks are fairly unlikely within the already limited set of scenarios where the sender is trusted. Collusion between ingress and egress of a network (intranet example in the draft) involves allowing a request that would have been rejected by an interior node. If the sender is trusted to keep to its promises by its access network, why should its access network not be trusted? So it is surely most reasonable to think about collusion between ingress and egress as extensions of the sender and receiver. Given the scheme is vulnerable to senders cheating it will be vulnerable to ingress networks cheating.
- §9.4.3: The rather optimistic argument that QS router collusion is not as bad as ECN router collusion is false. Somehow avoiding congestion drops by a flow being falsely ECN capable is seen as better than starting at a higher rate than you should, thus causing drops to others as well as yourself. The loss to yourself is minimal compared to the loss to others as a whole. This goes back to the lack of secure association between request and data. However, again, the whole idea of worrying about colluding routers when the source can do what it likes seems moot.

Discussion of the under-utilisation assumption could also be part of this applicability statement. It is conceded that the Quick-Start draft recognises the difficulties caused by the interaction between routers allocating capacity to Quick-Start traffic and source transports allocating capacity to non-Quick-Start traffic under end to end control. However, the rationale given for why it is good to mix these two forms of capacity allocation is essentially that Quick-Start is surrounded by a set of assumptions (trust and under-utilisation) that rule the question out of scope.

So, in summary, ruling questions like fairness out (see above) is fine for a scoped experiment, but in real life we can't set such clear bounds on applicability. The bounds of the experiment merge imperceptibly with the scenarios where the experimental assumptions break down.

4 Suggested improvements; taking the Quick-Start design choices as given

4.1 No control-data association

Because of the requirement of no router flow state, there is no association between control messages (Quick-Start requests/responses) and subsequent data. So, the router cannot know when or whether data has started to arrive as a result of an earlier request. The scheme has been worked out to avoid this being a problem, by:

- the use of conservative timers—each router conservatively sets aside capacity for responses given in the last few time-slots irrespective of whether downstream routers reduced the response later,
- conservative assumptions about all requests being new total requests, even if they are actually for additional rate (but see Section 5.3).

However, if responses are delayed (perfectly normal for a best-efforts service) beyond the conservative router timeout, large flows of data could arrive at routers after they had timed out their memory of having allocated the capacity requested for the data. After timing out this memory, any router may well have allocated more capacity to other requests. Being stateless, it assumes its current load includes the data that is actually still approaching in the cloud of dust over the horizon, so to speak. The result could be catastrophic overload.

One solution to this problem is for the timeout used by routers to be standardised. And for senders to use the same timeout. So if a request isn't answered within the timeout, the sender re-sends the request and ignores the response if it does arrive.

There remains a problem where the data is delayed more than the request/response was delayed. That is after the data leaves the sender but before arriving at a router N that has timed out the request. This could happen because the data itself builds up queues in buffers upstream of N, so its arrival at N gets spread over a longer period, delaying some of it past the timeout of N. Or there could be external causes of increased data delay upstream of router N.

Without completely changing the Quick-Start protocol, an improved, but still not completely safe, approach would be for the above proposed source timeout to be considerably less than the router timeout.

The likelihood of these race conditions is perhaps the size of a pimple on a pimple, given Quick-Start is intended for an under-utilised environment in the first place. It may be an acceptable risk to occasionally allow unexpected late arrivals of Quick-Start data to overflow a router without any more mechanism than these conservative timers. Particularly given re-routes can already do similar damage. However, we should not really introduce new protocols that can do damage using the excuse that it is no worse than the existing situation. Otherwise, if we solve the re-route problem, we will still have the Quick-Start problem.

4.2 Alternative rate reduced nonce

§3.4: The scenario where receivers may be evil, fickle beasts, whereas senders are *always* trusted to act totally and utterly in the interests of the commonweal seems very contrived. Certainly some senders might be trusted. But Quick-Start requires absolutely *all* senders that might use a Quick-Start network to be trusted.

That being said, we promised to set aside such concerns in this section, so let us assume that some rationale for this state of affairs has been written in to the draft (Section 3.8). Then a QS nonce can be used to check the receiver is honestly reporting the rate response that has managed to emerge from the network.

The newly proposed scheme in draft 01 means that a dishonest receiver has a 25% chance of correctly guessing how to undo a reduction of one step in the rate response.

The ECN nonce of RFC3540 [SWE03] can get away with allowing a receiver to guess how to lie correctly 50% of the time, because the guess must be repeated every CE-marked packet in a data stream. The chances of repeatedly making a 50 : 50 guess correctly, P, reduce exponentially with both the number of packets streamed n and the packet marking rate p. That is $P = 2^{-np}$. So, the more havoc the receiver tries to wreak, the less likely it can remain undetected.

The QS nonce has a different requirement, because it protects a control packet that will authorise a rate for a large number of future packets. With the ECN nonce, as soon as a sender detected a receiver lying it could stop the transfer, and the longer the transfer continued the more challenges were issued to the receiver. But with the QS nonce, the challenge is only issued once at the start. If the receiver guesses correctly just once, its gain is assured for a large number of future packets.

If the receiver happens to guess right first time (1 : 4 chance), it will get 2× initial download speed. It becomes more difficult to guess how to undo two or more rate reductions: the chance to get $2^r\times$ the rate is $1 : 2^{2r}$. But there is insufficient incentive to prevent receivers having a go sometimes, particularly if their identity is hidden (e.g. behind a large NAT), so the sender cannot record the receiver's reputation against its address, in case they encounter each other again later.

A possible alternative QS nonce would work as follows. A w-bit field is set aside for the QS nonce. The bigger the width w is, the (exponentially) more hard it is to brute force the undoing of a rate reduction. The sender generates a random nonce, stores it and puts it in this field. A router that reduces the Rate Request field by r (that is, reducing the rate requested by 2^r) should hash the QS nonce r times, using a one way hash function, such as MD5 [Riv92] or the secure hash 1 (SHA1) [NIS95]. Which hash function to use and its initialisation vector would have to be standardised for use in Quick-Start. Then the receiver simply returns the resulting QS nonce to the sender with the rate response. The sender knows what it originally requested and what the rate response is, so it can calculate the difference r. The sender knows what it originally set the QS nonce to, which it hashes r times and tests whether the result is the same as the QS nonce in the response.

Vulnerability to processor exhaustion attacks can be avoided by limiting the queue of responses to be hashed on routers to bound the amount of processing. This does of course still mean that attackers create unfairness in the shares of requests that get processed (Section 3.4).

If it were decided to use a floating point representation for the rate request (Section 4.4), rather than just the exponent as currently defined, this nonce scheme would only be practical if it were applied just to the exponent field, ignoring the mantissa. This would at least prevent a dishonest receiver from undoing a rate reduction to more than 2× the network's desired response.

But, to be honest, we need to see a rationale for why we should always trust senders, before we expend too much effort protecting against dishonest receivers. Especially given that the alternative scheme outlined in Section 3.1 moves in the right direction to potentially protect against misbehaving senders *and* receivers, without using any cryptography.

4.3 Maximum rate request

§3.1: The max rate request of 1.3Gbps is inadequate for future-proofing. I know of projects already considering designs for burst-mode allocation of the capacity of optical access networks to single applications at higher rates than this.

The Quick-Start protocol seems to go to great lengths to minimise the size of the IP Option field required, to the extent that the rate request granularity has to be coarse (making a guess by a dishonest receiver more likely to be correct—last para §9.4.2). Given the smallest rate Quick-Start can request is 80kbps, worrying about keeping to a 32bit header seems a little obsessive and unnecessarily limiting [since writing this, draft 01 has changed to a 64-bit header, but the Rate Request field is still 4 bits].

The argument (§A.2) that TCP window scaling tops out at 1.07Gbps is not relevant, as the whole point of Quick-Start is to start to solve the scaling problems of TCP. If we solve the TCP window scaling problem, we don't want to be left with the Quick-Start scaling problem.

4.4 Alternate rate encoding

The powers of two coding chosen is too coarse at the top end. Given there is no particular need to keep this protocol within a 32bit option field in total, an obvious alternative is to use a mantissa and exponent representation. The IEEE single-precision representation for floating-point numbers seems fairly appropriate as it has an 8 bit exponent, meaning it can raise the mantissa to $2^{255} \approx 6.10^{76}$. Nonetheless, note that however large the maximum number that can be represented, I would prefer to only specify normalised numbers in protocol headers, to avoid Y2K-style problems. This is another reason for my preference for the approach outlined in Section 3.1.

The IEEE float's exponent is preceded by one sign bit and followed by a 23 bit mantissa, requiring 32 bits in all. We could either use it directly, or better (I believe) modify it for our purposes.

The mantissa is interpreted by treating it as if there is a binary point before the first bit of the mantissa which therefore represents a binary fraction $f, 0 \leq f < 1$, with the first bit representing 1/2, the second bit 1/4 and so on. Then 1 is always added to the resulting fraction. So, with binary numbers e and f in the exponent and mantissa fields, the number represented is $(1 + f) \times 2^{(e-b)}$, where b is a bias explained below (we recommend $b = 0$).

Because $1 \leq (1 + f) < 2$ for all f, if two values have different exponents, they can always be compared solely by comparing their exponents. Only if the exponents are the same do the mantissas need comparing. This is an important property of the number representation, given Quick-Start must be able to compare two numbers fast (some other floating point representations contain redundancy so they can represent the same number with different combinations of mantissa and exponent).

The IEEE single precision representation has some irrelevant features. For instance, we wouldn't need the sign bit (set to 1 means a negative mantissa). It could be used to signal certain error conditions, but it would be more correct to have specific flags in Quick-Start if we need them. Also, the IEEE exponent is a biased value with a bias of $b = 127$, meaning 127 is subtracted from the value before being used as the exponent of 2 (rather than using twos-complement representation of negative numbers). This allows fractions to be represented using negative exponents, which we do not need. It would be sensible to use a bias of $b = 0$ for Quick-Start.

To reduce Quick-Start's coarseness at the high end of the exponent range, we could choose between 8, 16 or 24 bit precision of the mantissa rather than the IEEE 23 bit. Even if we used 24 bits, the rate field as a whole would take up 32 bits. If we allowed the whole IP option to take up 64 bits, we could still also fit in a more robust nonce (see Section 4.2).

4.5 Request refusal behaviour

This is a picky point about over-constraining implementation choices.

§3.3, 2nd para: When a router wishes to deny a Quick-Start Request the QS I-D allows it to zero the Rate Request, QS TTL and QS nonce, rather than removing the Option altogether, which may be less efficient. Instead, it would be more liberal to say the router should zero the Rate Request, and should set both the QS TTL and QS nonce to random values, which may be implemented by clearing the fields to zero for efficiency. This is because the value zero for the QS TTL or the QS nonce is not a magic value that the sender tests for, so there is no need to use it.

Alternatively, error codes could be placed in these secondary fields giving the reason for denying the request. Perhaps these codes could be used to indicate to the sender whether it is attached to a network that doesn't support QS at all (Section 3.8) as opposed to a temporary refusal.

4.6 Sender's DoS response

§9.4.1: If a sender gets multiple responses to a single request, it should stop processing them.

5 Clarity and nits

5.1 For clarity

5.2 Qualify "incrementally deployable"

§13: "Conclusions" and §A.5 "Alternate Responses to the Loss of a Quick-Start Packet" claim that Quick-Start is incrementally deployable. Although this is strictly not an incorrect statement, it greatly overstates the true position. A network continues to work while Quick-Start is incrementally deployed. But Quick-Start doesn't work at all until all routers on a path and both the source and destination have been upgraded. So, if the probability of a host being upgraded is P_h and that of a router being upgraded is P_r and the average network diameter is d routers, the probability of a Quick-Start request succeeding is $P_h^2.P_r^d$. So for example, even for a small network with $d = 3$, when 10% of all routers and hosts have been upgraded the probability of Quick-Start succeeding is $10\%^5 = 0.001\%$. Even if half of all routers and hosts have been upgraded on such a small network, the chance of being able to start quickly is only 3%. To remain able to start quickly while growing to a slightly larger network (with a diameter of say 6), still with half of all devices upgraded, the chance of success drops again to 0.4%.

So there is a very late network effect, with the benefits only appearing once nearly everyone has upgraded. This gives no-one any incentive to start the upgrade process. Thus the only realistic deployment scenario is where a central administration upgrades nearly every router on the network at once (assuming each legacy router is capable of upgrade). Thus the term incrementally deployable rather overplays the reality unless it is heavily qualified. Perhaps 'backward compatible' would be a better description.

This analysis tends to answer the question of §A.6 "Why Not Include More Functionality?". If it will take this long to get any benefit, it seems sensible to make sure we add other benefits at the same time.

5.3 Additional rate semantics unclear

It is not clear what the semantics are intended to be for a request for additional rate. Reverse engineering the text about gaming the system and such like in §A.4, this is how it seems to work: the sender requests the total rate, Z, it wants irrespective of how much rate it is already sending, A. The router doesn't know or care how much rate is already being sent and treats the request as it would treat any completely new request. So it responds giving the source $X \leq Z$. Then, if $X > A$ the source increases its rate by $X - A$. Otherwise the source reverts to standard congestion control. Whether this is what is meant or not, it needs clarifying (the para in §3.1 doesn't give the whole semantics and there is nothing in §3.3 or §A.4 about the semantics on a router).

If the router is stateless (as is proposed), it cannot know how much rate is already being used for a flow. So it cannot decide how much extra capacity is required for a request for additional capacity when only the total rate is requested. So it cannot decide whether to allow the request, *unless* it takes the conservative approach of assuming the current rate for every request is zero. This might be what the draft intends, but it is not clear in either §3.1 or §A.4.

As §A.4 says, "For either of these alternatives, there would not be room to report the current sending rate in the Quick-Start Option using the current minimal format for the Quick-Start Request." The I-D says this as a justification for only requesting a Total Rate (because an Additional Rate could be gamed without knowing the Current Rate as well). So there seems to be an implication that somehow the router can work out the current rate in order to know whether to admit a flow, but it can't work out the current rate in order to check whether it is being gamed. The only way I could resolve this apparently conflicting logic was to assume the router was being conservative (by always assuming the current rate is zero).

If we are trusting the sender, why don't we provide two fields for the sender to report its current rate and its requested rate, given we don't need to be stingy with header size in a high capacity network?

5.4 Other improvements for clarity

- (Picky) The first bullet of §3.3 says a router approving a Quick-Start Request must decrement the QS TTL by one. It would be safer to say it should decrement the QS TTL by the same value that it decrements the IP TTL, which may not always be 1 even though RFC1812 currently says it should be (e.g. the now deprecated TTL scoping used on the MBone had a different TTL decrement at different types of border gateway—this is not to say that Quick-Start should work with multicast, just that there may be other operational practices that decrement the TTL by more than 1).

- §3.4 "The QS Nonce", first sentence:
  "The QS Nonce gives the Quick-Start sender some protection against receivers lying about the value of the received Rate Request...."
  →
  "The QS Nonce allows the Quick-Start sender to give the network some protection against receivers lying about the value of the received Rate Request..."

- Fig 5: It would make more sense to call the Rate Request field in the TCP Option the Rate Response field.

- §9.1: A more complete calculation of the benefit of Quick-Start would help here (also necessary for the API in §10.1). That is, in order to weigh the extra complexity against the benefit, we need a formula for the benefit relative to TCP flows that would last longer than slow-start as well as those that would finish within slow-start.

- §9.4.4: "Misbehaving Middleboxes and the IP TTL" should surely not be within §9.4 "Protection against Misbehaving Nodes" as it is a feature interaction (albeit due to irritating attempts by middleboxes to 'improve' packets), not a malicious attack.

- Why is §9.5 "Attacks on Quick-Start" not a sub-section of §9.4 "Protection against Misbehaving Nodes"? My suspicion is that this structure is due to an assumption that Quick-Start is somehow secure as long as Quick-Start senders are trusted, even if other senders on the same network aren't trusted (see Section 3.3).

- Other related work includes the class of approaches where an initial request is sent into a 'scavenger' class (Singh *et al* [SGF05] give a useful set of references). Also Adams *et al* [ARI05].

- §A.7: "The Earlier QuickStart Nonce". This appendix might consider other related work that could have provided an alternative nonce mechanism, in order to give rationale for why they haven't been chosen. The closest example I can think of is Yang *et al*'s capability validation approach [YWA05] and its references (Perrig etc.).

- (Picky) §10.1: "Implementation issues...": As well as an additional timer, Quick-Start requires the source to hold the additional state of the TTL-Diff and QS nonce.

- §A1.1: "ICMP" The ICMP message would need the source and destination port numbers to know where to demultiplex to at each host.

- Both §A1.1 "ICMP" and §A1.2 "RSVP" talk of a corresponding transport level to be used for the response. But these requests at the network layer don't imply any particular transport protocol — unless it is encapsulated inside the ICMP or RSVP header.

- §A.6. "...requires less input to routers than XCP..." [explain what this means?]

5.5 Nits
(mostly found in draft 00, so some may have been fixed)

Throughout:
"an Rate Request"
→
"a Rate Request"
(vestige from an earlier change from "Initial Rate" to "Rate Request").

Contents and §4.7:
'...Middle of Connection..."
→
"...Middle of a Connection..."

§1 "Introduction", last sentence:
"In contrast, routers would not use Quick-Start to get congestion information,..."
→
"In contrast, routers would not use Quick-Start to give congestion information,..."

§2 "General Principles", Last bullet:
"A second practical consideration is that packets could be dropped..."
→
"A second practical consideration is that request packets could be dropped..."

§6(?) "Quick-Start in IP tunnels", last para of way #(1):
"...then the egress node should remove..."
→
"...then a Quick-Start aware egress node should remove..."

§6.2 last sentence says "Section 6.2 discusses..." [It must mean some other section].

§9.4.1 "Receivers Lying...", second para:
the the
→
then the

§10.2 "Implementation issues..." Last sentence:
send
→
sent

§12. Security considerations:
"Sections 9.4 and 9.5 discuss..."
→
"Sections 3.4, 9.4 and 9.5 discuss..."

§A.5 "Alternate Responses to the Loss of a Quick-Start Packet", final sentence:
"However,..."
→
"In other words,..."

§A.6. "...More Functionality?", Last sentence of first para:
"...that the current congestion..."
→
"...than the current congestion..."

§A.6. penultimate para:
"...a initial sending rate..."
→
"...an initial sending rate..."

§A.6. last para:
"...a positive step of meeting..."
→
"...a positive step towards meeting..."

§B. "Quick-Start with DCCP", last bullet numbered (1)
"...to send more that twice as fast as the receiver has reported received..."
→
"...to send more tha*n* twice as fast as the rate that the receiver has reported received..."

6 Conclusions

The Quick-Start proposal requires that every sender on a network must be trusted to comply with the Quick-Start protocol. We argue this is an inevitable consequence of choosing to have senders ask routers to allocate their rate. Any protocol that expects routers to do rate allocation must also require every trust domain along the path to hold per-flow-state in order to police each sender. If instead routers merely tell the endpoints their utilisation, we believe it is possible for endpoints to allocate their own rate without having to trust them—using the policing framework of re-feedback [BJCG+05, §3.3.3].

With endpoint-based rate allocation, the metric used would have to be a network impairment (i.e. pre-congestion) rather than rate. Although congestion is negligible in an under-utilised environment, it is possible to define 'the risk of not being under-utilised' ('pre-congestion') as a form of congestion.

Using pre-congestion as the metric makes endpoint-based rate allocation less deterministic. But we believe security considerations should be paramount—precise rate allocation is useless if senders can do what they want anyway. In other words, Quick-Start should take note of the classic dictum: 'Build in security from the start.'

Whether or not the authors agree about direction, the current draft certainly must state very clearly that it assumes all senders are completely trusted, especially given all the attention to malicious receivers, malicious networks and even malicious senders launching DoS attacks (presumably these senders are different ones from those trusted to comply with Quick-Start).

It would make more sense for the Quick-Start specification to completely ignore security issues and assume a trusted environment, rather than shore up three walls of the castle while leaving the fourth unbuilt. This would at least allow us to move forward rapidly, to see what happens when flows start quickly in controlled experiments with trusted devices and under-utilised capacity. However, if that is the chosen way forward, it should be very clearly stated that it is a tactical step, not an architectural direction.

This review also questions whether Quick-Start's under-utilisation assumption allows a distinct range of operation to be defined where issues like fairness can be ignored, or whether under-utilisation is just an extreme of a spectrum, making fairness an issue that must still be handled sometimes and in some places, because traffic variance will always blur the boundary of the under-utilisation assumption. The alternative pre-congestion-based approaches use the fact that pre-congestion sits on a spectrum where fairness is not an issue at one end, but becomes an issue as pre-congestion increasingly turns into congestion.

When someone asks for directions, a favourite response in Ireland is, "If you wanted to get there, I wouldn't have started from here." The bulk of this review of Quick-Start [JFAS05] is in that spirit. However, rather than being so unhelpful, pains have been taken to explain why it would have been better to start from somewhere else. But also many suggestions for improving the protocol within its own terms of reference have been made, by temporarily setting aside disquiet with the underlying assumptions.

This review argues that the recent nonce proposal gives insufficient protection against misbehaving receivers, and a new approach is suggested. Issues not related to security are also raised, including the possibility of a catastrophic overload if path delays are atypical. A solution to this is offered, as well as an improved encoding of the Rate Request field giving better scaling of range and precision. Many other more minor review comments are given.

Acknowledgements

Thanks to Alessandro Salvatori and Louise Burness (BT) for numerous useful review comments mainly improving clarity and to Martin Koyabe (BT) for pointing out the standard IEEE float encoding. Also thanks go to Mark Handley (UCL) for pointing out that solutions to lack of trust in the sender come along less often than congestion control solutions, which therefore need to be built around the trust solutions we have. Also Sally Floyd and Pasi Sarolahti have helped by reviewing this review and clarifying the intentions of Quick-Start.

References

[ARI05] John Adams, Lawrence G. Roberts, and Avril IJsselmuiden. Changing the Internet to support real-time content supply from a large fraction of broadband residential users. *BTTJ*, 23(2), April 2005.

[BBB+97] F. Baker, B. Braden, S. Bradner, A. Mankin, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang. Resource ReSerVation protocol (RSVP) — version 1 applicability statement; Some guidelines on deployment. Request for comments 2208, Internet Engineering Task Force, URL: rfc2208.txt, January 1997.

[BES+05] Bob Briscoe, Philip Eardley, David Songhurst, Francois Le Faucheur, Anna Charny, Jozef Babiarz, and Kwok-Ho Chan. A framework for admission control over DiffServ using pre-congestion notification. Internet Draft draft-briscoe-tsvwg-cl-architecture-01.txt, Internet Engineering Task Force, URL: draft-briscoe-tsvwg-cl-architecture-01.txt, October 2005. (Work in progress).

[BJCG+05] Bob Briscoe, Arnaud Jacquet, Carla Di Cairano-Gilfedder, Alessandro Salvatori, Andrea Soppera, and Martin Koyabe. Policing congestion response in an internetwork using re-feedback. *Proc. ACM SIGCOMM'05, Computer Communication Review*, 35(4):277–288, August 2005.

[BJS05] Bob Briscoe, Arnaud Jacquet, and Alessandro Salvatori. Re-ECN: Adding accountability for causing congestion to TCP/IP. Internet Draft draft-briscoe-tsvwg-re-ecn-tcp-00.txt, Internet Engineering Task Force, URL: http://www.cs.ucl.ac.uk/staff/B.Briscoe/pubs.html#retcp, October 2005. (Work in progress).

[Bri05] Bob Briscoe. Review: Quick-Start for TCP and IP. Internet Draft draft-ietf-tsvwg-quickstart–rvw-00.txt, Internet Engineering Task Force, URL: http://www.cs.ucl.ac.uk/staff/B.Briscoe/pubs.html#qsrvw, November 2005. (Work in progress).

[CSWB02] David Clark, Karen Sollins, John Wroclawski, and Robert Braden. Tussle in cyberspace: Defining tomorrow's Internet. *Proc. ACM SIGCOMM'02, Computer Communication Review*, 32(4):347–356, October 2002.

[Han05] Mark Handley. Internet congestion control research group. Working group charter, Internet Research Task Force, URL: http://nrg.cs.ucl.ac.uk/mjh/iccrg/, July 2005. (Proposal) (Continuously updated).

[JFAS05] Amit Jain, Sally Floyd, Mark Allman, and Pasi Sarolahti. Quick-Start for TCP and IP. Internet Draft draft-ietf-tsvwg-quickstart-01.txt, Internet Engineering Task Force, URL: http://www.icir.org/floyd/quickstart.html, October 2005. (Work in progress).

[KHR02] Dina Katabi, Mark Handley, and Charlie Rohrs. Congestion control for high bandwidth-delay product networks. *Proc. ACM SIGCOMM'02, Computer Communication Review*, 32(4):89–102, October 2002.

[KM99] Peter Key and Laurent Massoulié. User policies in a network implementing congestion pricing. In *Proc. Workshop on Internet Service Quality and Economics*, URL: http://www.marengoresearch.com/isqe/index.htm or URL: http://research.microsoft.com/research/network/publications/ISQElm.ps, December 1999. MIT.

[Kun03] Srisankar S. Kunniyur. AntiECN marking: A marking scheme for high bandwidth delay connections. In *Proc. ICC'03*, URL: http://www.seas.upenn.edu/~kunniyur/papers/aecn.html, May 2003. IEEE.

[NIS95] Secure hash standard. FIPS publication 180-1, NIST, U.S. Department of Commerce, Washington, D.C., April 1995.

[RFB01] K. K. Ramakrishnan, Sally Floyd, and David Black. The addition of explicit congestion notification (ECN) to IP. Request for comments 3168, Internet Engineering Task Force, URL: rfc3168.txt, September 2001.

[Riv92] Ronald L. Rivest. The MD5 message-digest algorithm. Request for comments 1321, Internet Engineering Task Force, URL: rfc1321.txt, 1992.

[SAF05] Pasi Sarolahti, Mark Allman, and Sally Floyd. Evaluating Quick-Start for TCP. URL: http://www.icir.org/floyd/quickstart.html, February 2005. (under submission).

[SGF05] M. Singh, Saikat Guha, and Paul Francis. Utilizing spare network bandwidth to improve TCP performance. In ACM SIGCOMM 2005 Work in Progress session, URL: https://www.guha.cc/saikat/pub/sigcomm05-lowtcp.pdf, August 2005.

[SWE03] Neil Spring, David Wetherall, and David Ely. Robust explicit congestion notification (ECN) signaling with nonces. Request for comments RFC3540, Internet Engineering Task Force, URL: http://www.ietf.org/rfcs/rfc3540.txt, June 2003. (Status: informational).

[XSSK05] Yong Xia, Lakshminarayanan Subramanian, Ion Stoica, and Shivkumar Kalyanaraman. One more bit is enough. *Proc. ACM SIGCOMM'05, Computer Communication Review*, 35(4):37–48, 2005.

[YWA05] Xiaowei Yang, David Wetherall, and Tom Anderson. A DoS-limiting network architecture. *Proc. ACM SIGCOMM'05, Computer Communication Review*, 35(4):241–252, August 2005.

Document history

Version	Date	Author	Details of change
00A	18 Sep 2005	Bob Briscoe	First Draft (bullet points)
00	27 Oct 2005	Bob Briscoe	First completed version
01	04 Nov 2005	Bob Briscoe	Improved clarity following review
02	15 Nov 2005	Bob Briscoe	Improved clarity of abstract and conclusions

