Network Working Group
Request for Comments: 3366
BCP: 62
Category: Best Current Practice
August 2002

G. Fairhurst
University of Aberdeen
L. Wood
Cisco Systems Ltd

Advice to link designers on link Automatic Repeat reQuest (ARQ)

Status of this Memo

This document specifies an Internet Best Current Practices for the Internet Community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document provides advice to the designers of digital communication equipment and link-layer protocols employing link-layer Automatic Repeat reQuest (ARQ) techniques. This document presumes that the designers wish to support Internet protocols, but may be unfamiliar with the architecture of the Internet and with the implications of their design choices for the performance and efficiency of Internet traffic carried over their links.

ARQ is described in a general way that includes its use over a wide range of underlying physical media, including cellular wireless, wireless LANs, RF links, and other types of channel. This document also describes issues relevant to supporting IP traffic over physical-layer channels where performance varies, and where link ARQ is likely to be used.

Table of Contents

1. Introduction

IP, the Internet Protocol [RFC791], forms the core protocol of the global Internet and defines a simple "connectionless" packet-switched network. Over the years, Internet traffic using IP has been carried over a wide variety of links, of vastly different capacities, delays and loss characteristics. In the future, IP traffic can be expected to continue to be carried over a very wide variety of new and existing link designs, again of varied characteristics.

A companion document [DRAFTKARN02] describes the general issues associated with link design. This document should be read in conjunction with that and with other documents produced by the Performance Implications of Link Characteristics (PILC) IETF workgroup [RFC3135, RFC3155].

This document is intended for three distinct groups of readers:

a. Link designers wishing to configure (or tune) a link for the IP traffic that it will carry, using standard link-layer mechanisms such as the ISO High-level Data Link Control (HDLC) [ISO4335a] or its derivatives.

b. Link implementers who may wish to design new link mechanisms that perform well for IP traffic.

c. The community of people using or developing TCP, UDP and related protocols, who may wish to be aware of the ways in which links can operate.

The primary audiences are intended to be groups (a) and (b). Group (c) should not need to be aware of the exact details of an ARQ scheme across a single link, and should not have to consider such details for protocol implementations; much of the Internet runs across links that do not use any form of ARQ. However, the TCP/IP community does need to be aware that the IP protocol operates over a diverse range of underlying subnetworks. This document may help to raise that awareness.

Perfect reliability is not a requirement for IP networks, nor is it a requirement for links [DRAFTKARN02]. IP networks may discard packets due to a variety of reasons entirely unrelated to channel errors, including lack of queuing space, congestion management, faults, and route changes. It has long been widely understood that perfect end-to-end reliability can be ensured only at, or above, the transport layer [SALT81].

Some familiarity with TCP, the Transmission Control Protocol [RFC793, STE94], is presumed here. TCP provides a reliable byte-stream transport service, building upon the best-effort datagram delivery service provided by the Internet Protocol. TCP achieves this by dividing data into TCP segments, and transporting these segments in IP packets. TCP guarantees that a TCP session will retransmit the TCP segments contained in any data packets that are lost along the Internet path between endhosts. TCP normally performs retransmission using its Fast Retransmit procedure, but if the loss fails to be detected (or retransmission is unsuccessful), TCP falls back to a Retransmission Time Out (RTO) retransmission using a timer [RFC2581, RFC2988]. (Link protocols also implement timers to verify integrity of the link, and to assist link ARQ.) TCP also copes with any duplication or reordering introduced by the IP network. There are a number of variants of TCP, with differing levels of sophistication in their procedures for handling loss recovery and congestion avoidance. Far from being static, the TCP protocol is itself subject to ongoing gradual refinement and evolution, e.g., [RFC2488, RFC2760].

Internet networks may reasonably be expected to carry traffic from a wide and evolving range of applications. Not all applications require or benefit from using the reliable service provided by TCP. In the Internet, these applications are carried by alternate transport protocols, such as the User Datagram Protocol (UDP) [RFC768].

1.1 Link ARQ

At the link layer, ARQ operates on blocks of data, known as frames, and attempts to deliver frames from the link sender to the link receiver over a channel. The channel provides the physical-layer connection over which the link protocol operates. In its simplest form, a channel may be a direct physical-layer connection between the two link nodes (e.g., across a length of cable or over a wireless medium). ARQ may also be used edge-to-edge across a subnetwork, where the path includes more than one physical-layer medium. Frames often have a small fixed or maximum size for convenience of processing by Medium-Access Control (MAC) and link protocols. This contrasts with the variable lengths of IP datagrams, or ’packets’. A link-layer frame may contain all, or part of, one or more IP packets. A link ARQ mechanism relies on an integrity check for each frame (e.g., strong link-layer CRC [DRAFTKARN02]) to detect channel errors, and uses a retransmission process to retransmit lost (i.e., missing or corrupted) frames.

Links may be full-duplex (allowing two-way communication over separate forward and reverse channels), half-duplex (where two-way communication uses a shared forward and reverse channel, e.g., IrDA, IEEE 802.11) or simplex (where a single channel permits communication in only one direction).

ARQ requires both a forward and return path, and therefore link ARQ may be used over links that employ full- or half-duplex links. When a channel is shared between two or more link nodes, a link MAC protocol is required to ensure all nodes requiring transmission can gain access to the shared channel. Such schemes may add to the delay (jitter) associated with transmission of packet data and ARQ control frames.

When using ARQ over a link where the probability of frame loss is related to the frame size, there is an optimal frame size for any specific target channel error rate. To allow for efficient use of the channel, this maximum link frame size may be considerably lower than the maximum IP datagram size specified by the IP Maximum Transmission Unit (MTU). Each frame will then contain only a fraction of an IP packet, and transparent implicit fragmentation of the IP datagram is used [DRAFTKARN02]. A smaller frame size introduces more frame header overhead per payload byte transported.

Explicit network-layer IP fragmentation is undesirable for a variety of reasons, and should be avoided [KEN87, DRAFTKARN02]. Its use can be minimized with use of Path MTU discovery [RFC1191, RFC1435, RFC1981].

Another way to reduce the frame loss rate (or reduce transmit signal power for the same rate of frame loss) is to use coding, e.g., Forward Error Correction (FEC) [LIN93].

FEC is commonly included in the physical-layer design of wireless links and may be used simultaneously with link ARQ. FEC schemes which combine modulation and coding also exist, and may also be adaptive. Hybrid ARQ [LIN93] combines adaptive FEC with link ARQ procedures to reduce the probability of loss of retransmitted frames. Interleaving may also be used to reduce the probability of frame loss by dispersing the occurrence of errors more widely in the channel to improve error recovery; this adds further delay to the channel’s existing propagation delay.

The document does not consider the use of link ARQ to support a broadcast or multicast service within a subnetwork, where a link may send a single packet to more than one recipient using a single link transmit operation. Although such schemes are supported in some subnetworks, they raise a number of additional issues not examined here.

Links supporting stateful reservation-based quality of service (QoS) according to the Integrated Services (intserv) model are also not explicitly discussed.

1.2 Causes of Packet Loss on a Link

Not all packets sent to a link are necessarily received successfully by the receiver at the other end of the link. There are a number of possible causes of packet loss. These may occur as frames travel across a link, and include:

a. Loss due to channel noise, often characterised by random frame loss. Channel noise may also result from other traffic degrading channel conditions.

b. Frame loss due to channel interference. This interference can be random, structured, and in some cases even periodic.

c. A link outage, a period during which the link loses all or virtually all frames, until the link is restored. This is a common characteristic of some types of link, e.g., mobile cellular radio.

Other forms of packet loss are not related to channel conditions, but include:

i. Loss of a frame transmitted in a shared channel where a contention-aware MAC protocol is used (e.g., due to collision). Here, many protocols require that retransmission is deferred to promote stability of the shared channel (i.e., prevent excessive channel contention). This is discussed further in section 1.5.

ii. Packet discards due to congestion. Queues will eventually overflow as the arrival rate of new packets to send continues to exceed the outgoing packet transmission rate over the link.

iii. Loss due to implementation errors, including hardware faults and software errors. This is recognised as a common cause of packet corruption detected in the endhosts [STONE00].

The rate of loss and patterns of loss experienced are functions of the design of the physical and link layers. These vary significantly across different link configurations. The performance of a specific implementation may also vary considerably across the same link configuration when operated over different types of channel.

1.3 Why Use ARQ?

Reasons that encourage considering the use of ARQ include:

a. ARQ across a single link has a faster control loop than TCP’s acknowledgement control loop, which takes place over the longer end-to-end path over which TCP must operate. It is therefore possible for ARQ to provide more rapid retransmission of TCP segments lost on the link, at least for a reasonable number of retries [RFC3155, SALT81].

b. Link ARQ can operate on individual frames, using implicit transparent link fragmentation [DRAFTKARN02]. Frames may be much smaller than IP packets, and repetition of smaller frames containing lost or errored parts of an IP packet may improve the efficiency of the ARQ process and the efficiency of the link.

A link ARQ procedure may be able to use local knowledge that is not available to endhosts, to optimise delivery performance for the current link conditions. This information can include information about the state of the link and channel, e.g., knowledge of the current available transmission rate, the prevailing error environment, or available transmit power in wireless links.

1.4 Commonly-used ARQ Techniques

A link ARQ protocol uses a link protocol mechanism to allow the sender to detect lost or corrupted frames and to schedule retransmission. Detection of frame loss may be via a link protocol timer, by detecting missing positive link acknowledgement frames, by receiving explicit negative acknowledgement frames and/or by polling the link receiver status.

Whatever mechanisms are chosen, there are two easily-described categories of ARQ retransmission process that are widely used:

1.4.1 Stop-And-Wait ARQ

A sender using stop-and-wait ARQ (sometimes known as 'Idle ARQ' [LIN93]) transmits a single frame and then waits for an acknowledgement from the receiver for that frame. The sender then either continues transmission with the next frame, or repeats transmission of the same frame if the acknowledgement indicates that the original frame was lost or corrupted.

Stop-and-wait ARQ is simple, if inefficient, for protocol designers to implement, and therefore popular, e.g., tftp [RFC1350] at the transport layer. However, when stop-and-wait ARQ is used in the link layer, it is well-suited only to links with low bandwidth-delay products. This technique is not discussed further in this document.

1.4.2 Sliding-Window ARQ

A protocol using sliding-window link ARQ [LIN93] numbers every frame with a unique sequence number, according to a modulus. The modulus defines the numbering base for frame sequence numbers, and the size of the sequence space. The largest sequence number value is viewed by the link protocol as contiguous with the first (0), since the numbering space wraps around.

TCP is itself a sliding-window protocol at the transport layer [STE94], so similarities between a link-interface-to-link-interface protocol and end-to-end TCP may be recognisable. A sliding-window link protocol is much more complex in implementation than the simpler stop-and-wait protocol described in the previous section, particularly if per-flow ordering is preserved.

At any time the link sender may have a number of frames outstanding and awaiting acknowledgement, up to the space available in its transmission window. A sufficiently-large link sender window (equivalent to or greater than the number of frames sent, or larger than the bandwidth*delay product capacity of the link) permits continuous transmission of new frames. A smaller link sender window causes the sender to pause transmission of new frames until a timeout or a control frame, such as an acknowledgement, is received. When frames are lost, a larger window, i.e., more than the link's bandwidth*delay product, is needed to allow continuous operation while frame retransmission takes place.

The modulus numbering space determines the size of the frame header sequence number field. This sequence space needs to be larger than the link window size and, if using selective repeat ARQ, larger than twice the link window size. For continuous operation, the sequence space should be larger than the product of the link capacity and the link ARQ persistence (discussed in section 2), so that in-flight frames can be identified uniquely.

As with TCP, which provides sliding-window delivery across an entire end-to-end path rather than across a single link, there are a large number of variations on the basic sliding-window implementation, with increased complexity and sophistication to make them suitable for various conditions. Selective Repeat (SR), also known as Selective Reject (SREJ), and Go-Back-N, also known as Reject (REJ), are examples of ARQ techniques using protocols implementing sliding window ARQ.

1.5 Causes of Delay Across a Link

Links and link protocols contribute to the total path delay experienced between communicating applications on endhosts. Delay has a number of causes, including:

a. Input packet queuing and frame buffering at the link head before transmission over the channel.

b. Retransmission back-off, an additional delay introduced for retransmissions by some MAC schemes when operating over a shared channel to prevent excessive contention. A high level of contention may otherwise arise, if, for example, a set of link receivers all retransmitted immediately after a collision on a busy shared channel. Link ARQ protocols designed for shared channels may select a backoff delay, which increases with the number of attempts taken to retransmit a frame; analogies can be drawn with end-to-end TCP congestion avoidance at the transport layer [RFC2581]. In contrast, a link over a dedicated channel (which has capacity pre-allocated to the link) may send a retransmission at the earliest possible time.

c. Waiting for access to the allocated channel when the channel is shared. There may be processing or protocol-induced delay before transmission takes place [FER99, PAR00].

d. Frame serialisation and transmission processing. These are functions of frame size and the transmission speed of the link.

e. Physical-layer propagation time, limited by the speed of transmission of the signal in the physical medium of the channel.

f. Per-frame processing, including the cost of QoS scheduling, encryption, FEC and interleaving. FEC and interleaving also add substantial delay and, in some cases, additional jitter. Hybrid link ARQ schemes [LIN93], in particular, may incur significant receiver processing delay.

g. Packet processing, including buffering frame contents at the link receiver for packet reassembly, before onward transmission of the packet.

When link ARQ is used, steps (b), (c), (d), (e), and (f) may be repeated a number of times, every time that retransmission of a frame occurs, increasing overall delay for the packet carried in part by the frame. Adaptive ARQ schemes (e.g., hybrid ARQ using adaptive FEC codes) may also incur extra per-frame processing for retransmitted frames.

It is important to understand that applications and transport protocols at the endhosts are unaware of the individual delays contributed by each link in the path, and only see the overall path delay. Application performance is therefore determined by the cumulative delay of the entire end-to-end Internet path. This path may include an arbitrary or even a widely-fluctuating number of links, where any link may or may not use ARQ. As a result, it is not possible to state fixed limits on the acceptable delay that a link can add to a path; other links in the path will add an unknown delay.

2. ARQ Persistence

ARQ protocols may be characterised by their persistency. Persistence is the willingness of the protocol to retransmit lost frames to ensure reliable delivery of traffic across the link.

A link’s retransmission persistency defines how long the link is allowed to delay a packet, in an attempt to transmit all the frames carrying the packet and its content over the link, before giving up and discarding the packet. This persistency can normally be measured in milliseconds, but may, if the link propagation delay is specified, be expressed in terms of the maximum number of link retransmission attempts permitted. The latter does not always map onto an exact time limit, for the reasons discussed in section 1.5.

An example of a reliable link protocol that is perfectly persistent is the ISO HDLC protocol in the Asynchronous Balanced Mode (ABM) [ISO4335a].

A protocol that only retransmits a number of times before giving up is less persistent, e.g., Ethernet [FER99], IEEE 802.11, or GSM RLP [RFC2757]. Here, lower persistence also ensures stability and fair sharing of a shared channel, even when many senders are attempting retransmissions.

TCP, STCP [RFC2960] and a number of applications using UDP (e.g., tftp) implement their own end-to-end reliable delivery mechanisms. Many TCP and UDP applications, e.g., streaming multimedia, benefit from timely delivery from lower layers with sufficient reliability, rather than perfect reliability with increased link delays.

2.1 Perfectly-Persistent (Reliable) ARQ Protocols

A perfectly-persistent ARQ protocol is one that attempts to provide a reliable service, i.e., in-order delivery of packets to the other end of the link, with no missing packets and no duplicate packets. The perfectly-persistent ARQ protocol will repeat a lost or corrupted frame an indefinite (and potentially infinite) number of times until the frame is successfully received.

If traffic is going no further than across one link, and losses do not occur within the endhosts, perfect persistence ensures reliability between the two link ends without requiring any higher-layer protocols. This reliability can become counterproductive for traffic traversing multiple links, as it duplicates and interacts with functionality in protocol mechanisms at higher layers [SALT81].

Arguments against the use of perfect persistence for IP traffic include:

a. Variable link delay; the impact of ARQ introduces a degree of jitter, a function of the physical-layer delay and frame serialisation and transmission times (discussed in section 1.5), to all flows sharing a link performing frame retransmission.

b. Perfect persistence does not provide a clear upper bound on the maximum retransmission delay for the link. Significant changes in path delay caused by excessive link retransmissions may lead to timeouts of TCP retransmission timers, although a high variance in link delay and the resulting overall path delay may also cause a large TCP RTO value to be selected [LUD99b, PAR00]. This will alter TCP throughput, decreasing overall performance, but, in mitigation, it can also decrease the occurrence of timeouts due to continued packet loss.

c. Applications needing perfectly-reliable delivery can implement a form of perfectly-persistent ARQ themselves, or use a reliable transport protocol within the endhosts. Implementing perfect persistence at each link along the path between the endhosts is redundant, but cannot ensure the same reliability as end-to-end transport [SALT81].

d. Link ARQ should not adversely delay the flow of end-to-end control information. As an example, the ARQ retransmission of data for one or more flows should not excessively extend the protocol control loops. Excessive delay of duplicate TCP acknowledgements (dupacks [STE94, BAL97]), SACK, or Explicit Congestion Notification (ECN) indicators will reduce the responsiveness of TCP flows to congestion events. Similar issues exist for TCP-Friendly Rate Control (TFRC), where equation-based congestion control is used with UDP [DRAFTHAN01].

Perfectly-persistent link protocols that perform unlimited ARQ, i.e., that continue to retransmit frames indefinitely until the frames are successfully received, are seldom found in real implementations.

Most practical link protocols give up retransmission at some point, but do not necessarily do so with the intention of bounding the ARQ retransmission persistence. A protocol may, for instance, terminate retransmission after a link connection failure, e.g., after no frames have been successfully received within a pre-configured timer period. The number of times a protocol retransmits a specific frame (or the maximum number of retransmissions) therefore becomes a function of many different parameters (ARQ procedure, link timer values, and procedure for link monitoring), rather than being pre-configured.

Another common feature of this type of behaviour is that some protocol implementers presume that, after a link failure, packets queued to be sent over the link are no longer significant and can be discarded when giving up ARQ retransmission.

Examples of ARQ protocols that are perfectly persistent include ISO/ITU-T LAP-B [ISO7776] and ISO HDLC in the Asynchronously Balanced Mode (ABM) [ISO4335a], e.g., using Multiple Selective Reject (MSREJ [ISO4335b]). These protocols will retransmit a frame an unlimited number of times until receipt of the frame is acknowledged.

2.2 High-Persistence (Highly-Reliable) ARQ Protocols

High-persistence ARQ protocols limit the number of times (or number of attempts) that ARQ may retransmit a particular frame before the sender gives up on retransmission of the missing frame and moves on to forwarding subsequent buffered in-sequence frames. Ceasing retransmission of a frame does not imply a lack of link connectivity and does not cause a link protocol state change.

It has been recommended that a single IP packet should never be delayed by the network for more than the Maximum Segment Lifetime (MSL) of 120 seconds defined for TCP [RFC1122]. It is, however, difficult in practice to bound the maximum path delay of an Internet path. One case where segment (packet) lifetime may be significant is where alternate paths of different delays exist between endhosts and route flapping or flow-unaware traffic engineering is used. Some TCP packets may follow a short path, while others follow a much longer path, e.g., using persistent ARQ over a link outage.

Failure to limit the maximum packet lifetime can result in TCP sequence numbers wrapping at high transmission rates, where old data segments may be confused with newer segments if the sequence number space has been exhausted and reused in the interim. Some TCP implementations use the Round Trip Timestamp Measurement (RTTM) option in TCP packets to remove this ambiguity, using the Protection Against Wrapped Sequence number (PAWS) algorithm [RFC1323].

In practice, the MSL is usually very large compared to the typical TCP RTO. The calculation of TCP RTO is based on estimated round-trip path delay [RFC2988]. If the number of link retransmissions causes a path delay larger than the value of RTO, the TCP retransmission timer can expire, leading to a timeout and retransmission of a segment (packet) by the TCP sender.

Although high persistency may benefit bulk flows, the additional delay (and variations in delay) that it introduces may be highly undesirable for other types of flows. Being able to treat flows separately, with different classes of link service, is useful, and is discussed in section 3.

Examples of high-persistence ARQ protocols include [BHA97, ECK98, LUD99a, MEY99].

2.3 Low-Persistence (Partially-Reliable) ARQ Protocols

The characteristics of a link using a low-persistence ARQ protocol may be summarised as:

a. The link is not perfectly reliable and does not provide an absolute guarantee of delivery, i.e., the transmitter will discard some frames as it ’gives up’ before receiving an acknowledgement of successful transmission across the link.

b. There is a lowered limit on the maximum added delay that IP packets will experience when travelling across the link (typically lower than the TCP path RTO). This reduces interaction with TCP timers or with UDP-based error-control schemes.

c. The link offers a low bound for the time that retransmission for any one frame can block completed transmission and assembly of other correctly and completely-received IP packets whose transmission was begun before the missing frame was sent. Limiting delay avoids aggravating contention or interaction between different packet flows (see also section 3.2).

Examples of low-persistence ARQ protocols include [SAM96, WARD95, CHE00].

2.4 Choosing Your Persistency

The TCP Maximum RTO is an upper limit on the maximum time that TCP will wait until it performs a retransmission. Most TCP implementations will generally have a TCP RTO of at least several times the path delay.

Setting a lower link persistency (e.g., of the order 2-5 retransmission attempts) reduces potential interaction with the TCP RTO timer, and may therefore reduce the probability of duplicate copies of the same packet being present in the link transmit buffer under some patterns of loss.

A link using a physical layer with a low propagation delay may allow tens of retransmission attempts to deliver a single frame, and still satisfy a bound for (b) in section 2.3. In this case, a low delay is defined as being where the total packet transmission time across the link is much less than 100 ms (a common value for the granularity of the internal TCP system timer).

A packet may traverse a number of successive links on its total end-to-end path. This is therefore an argument for much lower persistency on any individual link, as delay due to persistency is accumulated along the path taken by each packet.

Some implementers have chosen a lower persistence, falling back on the judgement of TCP or of a UDP application to retransmit any packets that are not recovered by the link ARQ protocol.

2.5 Impact of Link Outages

Links experiencing persistent loss, where many consecutive frames are corrupted over an extended time, may also need to be considered. Examples of channel behaviour leading to link outages include fading, roaming, and some forms of interference. During the loss event, there is an increased probability that a retransmission request may be corrupted, and/or an increased probability that a retransmitted frame will also be lost. This type of loss event is often known as a "transient outage".

If the transient outage extends for longer than the TCP RTO, the TCP sender will also perform transport-layer retransmission. At the same time, the TCP sender will reduce its congestion window (cwnd) to 1 segment (packet), recalculate its RTO, and wait for an ACK packet. If no acknowledgement is received, TCP will retransmit again, up to a retry limit. TCP only determines that the outage is over (i.e., that path capacity is restored) by receipt of an ACK. If link ARQ protocol persistency causes a link in the path to discard the ACK, the TCP sender must wait for the next RTO retransmission and its ACK to learn that the link is restored. This can be many seconds after the end of the transient outage.

When a link layer is able to differentiate a set of link service classes (see section 3.3), a link ARQ persistency longer than the largest link loss event may benefit a TCP session. This would allow TCP to rapidly restore transmission without the need to wait for a retransmission time out, generally improving TCP performance in the face of transient outages. Implementation of such schemes remains a research issue.

When an outage occurs for a sender sharing a common channel with other nodes, uncontrolled high persistence can continue to consume transmission resources for the duration of the outage. This may be undesirable, since it reduces the capacity available for other nodes sharing the channel, which do not necessarily experience the same outage. These nodes could otherwise use the channel for more productive transfers. The persistence is often limited by another controlling mechanism in such cases. To counter such contention effects, ARQ protocols may delay retransmission requests, or defer the retransmission of requested frames until the outage ends for the sender.

An alternate suggested approach for a link layer that is able to identify separate flows is to use low link persistency (section 2.3) along with a higher-layer mechanism, for example one that attempts to deliver one packet (or whole TCP segment) per TCP flow after a loss event [DRAFTKARN02]. This is intended to ensure that TCP transmission is restored rapidly. Algorithms to implement this remain an area of research.

3. Treatment of Packets and Flows

3.1 Packet Ordering

A common debate is whether a link should be allowed to forward packets in an order different from that in which they were originally received at its transmit interface.

IP networks are not required to deliver all IP packets in order, although in most cases networks do deliver IP packets in their original transmission order. Routers supporting class-based queuing do reorder received packets, by reordering packets in different flows, but these usually retain per-flow ordering.

Policy-based queuing, allowing fairer access to the link, may also reorder packets. There is still much debate on optimal algorithms, and on optimal queue sizes for particular link speeds. This, however, is not related to the use of link ARQ and applies to any (potential) bottleneck router.

Although small amounts of reordering are common in IP networks [BEN00], significant reordering within a flow is undesirable as it can have a number of effects:

a. Reordering will increase packet jitter for real-time applications. This may lead to application data loss if a small play-out buffer is used by the receiving application.

b. Reordering will interleave arrival of TCP segments, leading to generation of duplicate ACKs (dupacks), leading to assumptions of loss. Reception of an ACK followed by a sequence of three identical dupacks causes the TCP sender to trigger fast retransmission and recovery, a form of congestion avoidance, since TCP always presumes that packet loss is due to congestion [RFC2581, STE94]. This reduces TCP throughput efficiency as far as the application is concerned, although it should not impact data integrity.

In addition, reordering may negatively impact processing by some existing poorly-implemented TCP/IP stacks, by leading to unwanted side-effects in poorly-implemented IP fragment reassembly code, poorly-implemented IP demultiplexing (filter) code, or in poorly-implemented UDP applications.

Ordering effects must also be considered when breaking the end-to-end paradigm and evaluating transport-layer relays such as split-TCP implementations or Protocol Enhancing Proxies [RFC3135].

As with total path delay, TCP and UDP flows are impacted by the cumulative effect of reordering along the entire path. Link protocol designers must not assume that their link is the only link undertaking packet reordering, as some level of reordering may be introduced by other links along the same path, or by router processing within the network [BEN00]. Ideally, the link protocol should not contribute to reordering within a flow, or at least ensure that it does not significantly increase the level of reordering within the flow. To achieve this, buffering is required at the link receiver. The total amount of buffering required is a function of the link’s bandwidth*delay product and the level of ARQ persistency, and is bounded by the link window size.

A number of experimental ARQ protocols have allowed out-of-order delivery [BAL95, SAM96, WARD95].

3.2 Using Link ARQ to Support Multiple Flows

Most links can be expected to carry more than one IP flow at a time. Some high-capacity links are expected to carry a very large number of simultaneous flows, often from and to a large number of different endhosts. With use of multiple applications at an endhost, multiple flows can be considered the norm rather than the exception, even for last-hop links.

When packets from several flows are simultaneously in transit within a link ARQ protocol, ARQ may cause a number of additional effects:

a. ARQ introduces variable delay (jitter) to a TCP flow sharing a link with another flow experiencing loss. This additional delay, introduced by the need for a link to provide in-sequence delivery of packets, may adversely impact other applications sharing the link, and can increase the duration of the initial slow-start period for TCP flows for these applications. This is significant for short-lived TCP flows (e.g., those used by HTTP/1.0 and earlier), which spend most of their lives in slow-start.

b. ARQ introduces jitter to UDP flows that share a link with another flow experiencing loss. An end-to-end protocol may not require reliable delivery for its flows, particularly if it is supporting a delay-sensitive application.

c. High-persistence ARQ may delay packets long enough to cause the premature timeout of another TCP flow’s RTO timer, although modern TCP implementations should not experience this since their computed RTO values should leave a sufficient margin over path RTTs to cope with reasonable amounts of jitter.

Reordering of packets belonging to different flows may be desirable [LUD99b, CHE00] to achieve fair sharing of the link between established bulk-data transfer sessions and sessions that transmit less data, but would benefit from lower link transit delay. Preserving ordering within each individual flow, to avoid the effects of reordering described earlier in section 3.1, is worthwhile.

3.3 Differentiation of Link Service Classes

High ARQ persistency is generally considered unsuitable for many applications using UDP, where reliable delivery is not always required and where it may introduce unacceptable jitter, but may benefit bulk data transfers under certain link conditions. A scheme that differentiates packet flows into two or more classes, to provide a different service to each class, is therefore desirable.

Observation of flow behaviour can tell you which flows are controlled and congestion-sensitive, or uncontrolled and not, so that you can treat them differently and ensure fairness. However, this cannot tell you whether a flow is intended as reliable or unreliable by its application, or what the application requires for best operation.

Supporting different link services for different classes of flows therefore requires that the link is able to distinguish the different flows from each other. This generally needs an explicit indication of the class associated with each flow.

Some potential schemes for indicating the class of a packet include:

a. Using the Type of Service (ToS) bits in the IP header [RFC791]. The IETF has replaced these globally-defined bits, which were not widely used, with the differentiated services model (diffserv [RFC2475, RFC3260]). In diffserv, each packet carries a Differentiated Service Code Point (DSCP), which indicates which one of a set of diffserv classes the flow belongs to. Each router maps the DSCP value of a received IP packet to one of a set of Per Hop Behaviours (PHBs) as the packet is processed within the network. Diffserv uses include policy-based routing, class-based queuing, and support for other QoS metrics, including IP packet priority, delay, reliability, and cost.

b. Inspecting the network packet header and viewing the IP protocol type [RFC791] to gain an idea of the transport protocol used and thus guess its needs. This is not possible when carrying an encrypted payload, e.g., using the IP security extensions (IPSec) with Encapsulation Security Payload (ESP) [RFC2406] payload encryption.

c. By inspecting the transport packet header information to view the TCP or UDP headers and port numbers (e.g., [PAR00, BAL95]). This is not possible when using payload encryption, e.g., IPSec with ESP payload encryption [RFC2406], and incurs processing overhead for each packet sent over the link.

There are, however, some drawbacks to these schemes:

i. The ToS/Differentiated Services Code Point (DSCP) values [RFC2475] may not be set reliably, and may be overwritten by intermediate routers along the packet’s path. These values may be set by an ISP, and do not necessarily indicate the level of reliability required by the end application. The link must be configured with knowledge of the local meaning of the values.

ii. Tunnelling of traffic (e.g., GRE, MPLS, L2TP, IP-in-IP encapsulation) can aggregate flows of different transport classes, complicating individual flow classification with schemes (b) and (c) and incurring further header processing if tunnel contents are inspected.

iii. Use of the TCP/UDP port number makes assumptions about application behaviour and requirements. New applications or protocols can invalidate these assumptions, as can the use of e.g., Network Address Port Translation, where port numbers are remapped [RFC3022].

iv. In IPv6, the entire IPv6 header must be parsed to locate the transport layer protocol, adding complexity to header inspection. Again, this assumes that IPSec payload encryption is not used.

Despite the difficulties in providing a framework for accurate flow identification, approach (a) may be beneficial, and is preferable to adding optimisations that are triggered by inspecting the contents of specific IP packets. Some such optimisations are discussed in detail in [LUD99b].

Flow management is desirable; clear flow identification increases the number of tools available for the link designer, and permits more complex ARQ strategies that may otherwise make misassumptions about traffic requirements and behaviour when flow identification is not done.

Links that are unable to distinguish clearly and safely between delay-sensitive flows, e.g., real-time multimedia, DNS queries or telnet, and delay-insensitive flows, e.g., bulk ftp transfers or reliable multicast file transfer, cannot separate link service classes safely. All flows should therefore experience the same link behaviour.

In general, if separation of flows according to class is not practicable, a low persistency is best for link ARQ.

4. Conclusions

A number of techniques may be used by link protocol designers to counter the effects of channel errors or loss. One of these techniques is Automatic Repeat ReQuest, ARQ, which has been and continues to be used on links that carry IP traffic. An ARQ protocol retransmits link frames that have been corrupted during transmission across a channel. Link ARQ may significantly improve the probability of successful transmission of IP packets over links prone to occasional frame loss.

A lower rate of packet loss generally benefits transport protocols and endhost applications. Applications using TCP generally benefit from Internet paths with little or no loss and low round trip path delay. This reduces impact on applications, allows more rapid growth of TCP’s congestion window during slow start, and ensures prompt reaction to end-to-end protocol exchanges (e.g., retransmission, congestion indications). Applications using other transport protocols, e.g., UDP or SCTP, also benefit from low loss and timely delivery.

A side-effect of link ARQ is that link transit delay is increased when frames are retransmitted. At low error rates, many of the details of ARQ, such as degree of persistence or any resulting out-of-order delivery, become unimportant. Most frame losses will be resolved in one or two retransmission attempts, and this is generally unlikely to cause significant impact to e.g., TCP. However, on shared high-delay links, the impact of ARQ on other UDP or TCP flows may lead to unwanted jitter.

Where error rates are highly variable, high link ARQ persistence may provide good performance for a single TCP flow. However, interactions between flows can arise when many flows share capacity on the same link. A link ARQ procedure that provides flow management will be beneficial. Lower ARQ persistence may also have merit, and is preferable for applications using UDP. The reasoning here is that the link can perform useful work forwarding some complete packets, and that blocking all flows by retransmitting the frames of a single packet with high persistence is undesirable.

During a link outage, interactions between ARQ and multiple flows are less significant; the ARQ protocol is likely to be equally unsuccessful in retransmitting frames for all flows. High persistence may benefit TCP flows, by enabling prompt recovery once the channel is restored.

Low ARQ persistence is particularly useful where it is difficult or impossible to classify traffic flows, and hence treat each flow independently, and where the link capacity can accommodate a large number of simultaneous flows.

Link ARQ designers should consider the implications of their design on the wider Internet. Effects such as increased transit delay, jitter, and re-ordering are cumulative when performed on multiple links along an Internet path. It is therefore very hard to say how many ARQ links may exist in series along an arbitrary Internet path between endhosts, especially as the path taken and its links may change over time.

In summary, when links cannot classify traffic flows and treat them separately, low persistence is generally desirable; preserving packet ordering is generally desirable. Extremely high persistence and perfect persistence are generally undesirable; highly-persistent ARQ is a bad idea unless flow classification and detailed and accurate knowledge of flow requirements make it possible to deploy high persistency where it will be beneficial.

There is currently insufficient experience to recommend a specific ARQ scheme for any class of link. It is also important to realize that link ARQ is just one method of error recovery, and that other complementary physical-layer techniques may be used instead of, or together with, ARQ to improve overall link throughput for IP traffic.

The choice of potential schemes includes adapting the data rate, adapting the signal bandwidth, adapting the transmission power, adaptive modulation, adaptive information redundancy / forward error control, and interleaving. All of these schemes can be used to improve the received signal energy per bit, and hence reduce error, frame loss and resulting packet loss rates given specific channel conditions.

There is a need for more research to more clearly identify the importance of and trade-offs between the above issues over various types of link and over various types of channels. It would be useful if researchers and implementers clearly indicated the loss model, link capacity and characteristics, link and end-to-end path delays, details of TCP, and the number (and details) of flows sharing a link when describing their experiences. In each case, it is recommended that specific details of the link characteristics and mechanisms also be considered; solutions vary with conditions.

5. Security Considerations

No security implications have been identified as directly impacting IP traffic. However, an unreliable link service may adversely impact some existing link-layer key management distribution protocols if link encryption is also used over the link.

Denial-of-service attacks exploiting the behaviour of the link protocol, e.g., using knowledge of its retransmission behaviour and propagation delay to cause a particular form of jamming, may be specific to an individual link scenario.

6. IANA Considerations

No assignments from the IANA are required.

7. Acknowledgements

Much of what is described here has been developed from a summary of a subset of the discussions on the archived IETF PILC mailing list. We thank the contributors to PILC for vigorous debate.

In particular, the authors would like to thank Spencer Dawkins, Aaron Falk, Dan Grossman, Merkourios Karaliopoulos, Gary Kenward, Reiner Ludwig and Jean Tourrilhes for their detailed comments.

8. References

References of the form RFCnnnn are Internet Request for Comments (RFC) documents available online at http://www.rfc-editor.org/.

8.1 Normative References

[RFC768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol", RFC 793, September 1981.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -- Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2988] Paxson, V. and M. Allman, "Computing TCP’s Retransmission Timer", RFC 2988, November 2000.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G. and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, June 2001.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.

8.2 Informative References

[BAL95] Balakrishnan, H., Seshan, S. and R. H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks", ACM MOBICOM, Berkeley, 1995.

[BAL97] Balakrishnan, H., Padmanabhan, V. N., Seshan, S. and R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", IEEE/ACM Transactions on Networking, 5(6), pp. 756-759, 1997.

[BEN00] Bennett, J. C., Partridge, C. and N. Schectman, "Packet Reordering is Not Pathological Network Behaviour", IEEE/ACM Transactions on Networking, 7(6), pp. 789-798, 2000.

[BHA97] Bhagwat, P., Bhattacharya, P., Krishna A. and S. K. Tripathi, "Using channel state dependent packet scheduling to improve TCP throughput over wireless LANs", ACM/Baltzer Wireless Networks Journal, (3)1, 1997.

[CHE00] Cheng, H. S., G. Fairhurst et al., "An Efficient Partial Retransmission ARQ Strategy with Error Codes by Feedback Channel", IEE Proceedings - Communications, (147)5, pp. 263-268, 2000.

[DRAFTKARN02] Karn, P., Ed., "Advice for Internet Subnetwork Designers", Work in Progress.

[DRAFTHAN01] Handley, M., Floyd, S. and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", Work in Progress.

[ECK98] Eckhardt, D. A. and P. Steenkiste, "Improving Wireless LAN Performance via Adaptive Local Error Control", IEEE ICNP, 1998.

[FER99] Ferrero, A., "The Eternal Ethernet", Addison-Wesley, 1999.

[ISO4335a] HDLC Procedures: Specification for Consolidation of Elements of Procedures, ISO 4335 and AD/1, International Standardization Organization, 1985.

[ISO4335b] HDLC Procedures: Elements of Procedures, Amendment 4: Multi-Selective Reject Option, ISO 4335/4, International Standards Organization, 1991.

[ISO7776] Specification for X.25 LAPB-Compatible DTE Data Link Procedures, ISO 4335/4, International Standards Organization, 1985.

[KEN87] Kent, C. A. and J. C. Mogul, "Fragmentation Considered Harmful", Proceedings of ACM SIGCOMM 1987, ACM Computer Communications Review, 17(5), pp. 390-401, 1987.

[LIN93] Lin, S. and D. Costello, "Error Control Coding: Fundamentals and Applications", Prentice Hall, 1993.

[LUD99a] Ludwig, R., Rathonyi, B., Konrad, A., Oden, K., and A. Joseph, "Multi-Layer Tracing of TCP over a Reliable Wireless Link", ACM SIGMETRICS, pp. 144-154, 1999.

[LUD99b] Ludwig, R., Konrad, A., Joseph, A. and R. H. Katz, "Optimizing the End-to-End Performance of Reliable Flows over Wireless Links", ACM MobiCOM, 1999.

[MEY99] Meyer, M., "TCP Performance over GPRS", IEEE Wireless Communications and Networking Conference, 1999.

[PAR00] Parsa, C. and J. J. Garcia-Luna-Aceves, "Improving TCP Performance over Wireless Networks at the Link Layer", ACM Mobile Networks and Applications Journal, (5)1, pp. 57-71, 2000.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC1323] Jacobson, V., Braden, R. and D. Borman, "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC1350] Sollins, K., "The TFTP Protocol (Revision 2)", STD 33, RFC 1350, July 1992.

[RFC1435] Knowles, S., "IESG Advice from Experience with Path MTU Discovery", RFC 1435, March 1993.

[RFC1981] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2488] Allman, M., Glover, D. and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", BCP 28, RFC 2488, January 1999.

[RFC2757] Montenegro, G., Dawkins, S., Kojo, M., Magret V. and N. Vaidya, "Long Thin Networks", RFC 2757, January 2000.

[RFC2760] Allman, M., Dawkins, S., Glover, D., Griner, J., Tran, D., Henderson, T., Heidemann, J., Touch, J., Kruse, H., Ostermann, S., Scott K. and J. Semke "Ongoing TCP Research Related to Satellites", RFC 2760, February 2000.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC3155] Dawkins, S., Montenegro, G., Kojo, M., Magret, V. and N. Vaidya, "End-to-end Performance Implications of Links with Errors", BCP 50, RFC 3155, August 2001.

[SALT81] Saltzer, J. H., Reed, D. P. and D. Clark, "End-to-End Arguments in System Design", Second International Conference on Distributed Computing Systems, pp. 509-512, 1981. Published with minor changes in ACM Transactions in Computer Systems (2)4, pp. 277-288, 1984.

[SAM96] Samaraweera, N. and G. Fairhurst, "Robust Data Link Protocols for Connection-less Service over Satellite Links", International Journal of Satellite Communications, 14(5), pp. 427-437, 1996.

[SAM98] Samaraweera, N. and G. Fairhurst, "Reinforcement of TCP/IP Error Recovery for Wireless Communications", ACM Computer Communications Review, 28(2), pp. 30-38, 1998.

[STE94] Stevens, W. R., "TCP/IP Illustrated, Volume 1", Addison-Wesley, 1994.

[STONE00] Stone, J. and C. Partridge, "When the CRC and TCP Checksum Disagree", Proceedings of SIGCOMM 2000, ACM Computer Communications Review 30(4), pp. 309-321, September 2000.

[WARD95] Ward, C., et al., "A Data Link Control Protocol for LEO Satellite Networks Providing a Reliable Datagram Service", IEEE/ACM Transactions on Networking, 3(1), 1995.

Authors’ Addresses

Godred Fairhurst
Department of Engineering
University of Aberdeen
Aberdeen AB24 3UE
United Kingdom

EMail: gorry@erg.abdn.ac.uk
http://www.erg.abdn.ac.uk/users/gorry/

Lloyd Wood
Cisco Systems Ltd
4 The Square
Stockley Park
Uxbridge UB11 1BY
United Kingdom

EMail: lwood@cisco.com
http://www.ee.surrey.ac.uk/Personal/L.Wood/

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.