IEN # 2
Supercedes: None
Replaces: None

Jon Postel
ISI
15 August 1977

2.3.3.2 Comments on Internet Protocol and TCP

Introduction

This memo suggests an approach to protocols used in internetwork systems somewhat different from the main thrust of the work on the Transmission Control Protocol (TCP) [1]. The position taken here is that internetwork communication should be view as having two components: the hop by hop relaying of a message, and the end to end control of the conversation. This leads to a proposal for a hop by hop oriented internet protocol, an end to end oriented host level protocol, and the interface between them.

Discussion

We are screwing up in our design of internet protocols by violating the principle of layering. Specifically we are trying to use TCP to do two things: serve as a host level end to end protocol, and to serve as an internet packaging and routing protocol. These two things should be provided in a layered and modular way. I suggest that a new distinct internetwork protocol is needed, and that TCP be used strictly as a host level end to end protocol. I also believe that if TCP is used only in this cleaner way it can be simplified somewhat. A third item must be specified as well -- the interface between the internet host to host protocol and the internet hop by hop protocol.

> An analogy may be drawn between the internet situation and the ARPANET. The endpoints of message transmissions are hosts in both cases, and they exchange messages conforming to a host to host protocol. In the ARPA subnet there is a IMP to IMP protocol that is primarily a hop by hop protocol, to parallel this the internet system should have a hop by hop internet protocol. In the ARPANET a host and an IMP interact through an inteface, commonly called 1822, which specifies the format of messages crossing the boundary, an equivalent interface in needed in the internet system.

In the rest of this memo i outline first a possible internet host - hop interface, second an internet hop by hop protocol, and third some modifications to TCP so that it can serve as an internet host level protocol.

Internet Host - Hop Interface

It is difficult to present a protocol and an interface separately. In this section i present the fields and functions used in the internet host - hop interface. The discussion will however explain some of the operation of the internet hop by hop protocol as well.

I suggest an internet hop by hop protocol that provides only those functions needed to address and route messages in an arbitrarily structured network, to allow for fragmentation and reassembly of fragments, to provide various types of service, and a moderate level of error control.

The internet host - hop interface is described by discussing the use made of each of the fields in the interface message header.

Version

As time goes by the internetwork system may evolve to a point where the interface format (or the protocol) must be changed. This field provides the handle for simultaneously supporting two (or more) versions of the protocol.

Data Identifier and Acknowledgement Identifier

When a message is sent from a host to a hop module, or between hop modules, or from a hop module to a host, it must be acknowledged. To allow several messages to be in transit simultaneously an identifier is used to associate data messages and acknowledgements.

Type of Service

Because different applications may call for different aspects of communication performance to be emphasised a type of service field is necessary. For example one application may call for reliable delivery and be willing to suffer longer delays or lower throughput for it. Another application may not be able to tolerate the slightest delay but be unconcerned about reliability. The type of service field has the following assigned values:

0 - reliable delivery requested

This means that at each hop the message (or fragment) is retransmitted to the next hop until an acknowledgement is received.

1 - no retransmission

This means that at each hop the message (or fragment) is transmitted exactly once to the next hop, if it is afflicted with errors, too bad.

Addresses

Addresses are variable length strings of 4 bit chunks prefixed by a length. As address chunks are processed they are removed from their position at the head of the address chunk string and placed at the end of the string. This chunk by chunk circular shifting of the address allows each node in the hop by hop processing of a message to examine the part of the address it consumes with out knowing how much address preceeds or follows that part.

Fragmentation

Fragmentation is an internet protocol problem rather than a host level problem. Fragmentation is necessary because of the differing sizes of maximum message allowed by the various networks making up the internetwork system. The possibility of a large message being routed or delivered in a network with a smaller maximum message size requires that the internet protocols provide for fragmentation.

Any where along the transmission path an internet protocol node may fragment a message (which already may be a fragment). Only at a point where all fragments must pass can reassembly of the fragments be done. Usually the only point that meets this constraint is the destination host.

To understand a little about the fragmentation information carried in the internet protocol header we examine the information needed to reassemble fragments and the possible ways to provide that information.

What does the reassembler need to know?

What message is this a fragment of? - Message Identifier

Where in the message does this fragment go? - Fragment Number and Size

Are all the fragments of this message here? - Number of Fragments or Last Fragment Flag

What can the fragmenter (original or intermeadiate) provide?

It may be easier to say what it can not do:

It can't keep a count of total fragments carried in each fragment because some fragments may get split while others do not in a alternate routing environment.

It can't know what fragment identifiers have already been used.

It can do these things:

It can pass on a last fragment flag.

It can pass on fragment numbers that are hierarchial or based of data length such as data sequence numbers.

It can pass on a message identifier.

It can pass on a data length.

The difficult issue is how best to identify where this fragment fits with respect to other fragments when trying to reassemble a message. Two schemes seem workable: a hierachial fragment naming scheme, and a sequence number scheme.

The hierachial scheme would call for a fragment name to be composed of a variable number of chunks. Each time a fragment was split each new fragment would get a copy of the old fragment name with an added chunk specifing the order between the just created fragments. These fragment names would then be equivalent to the names of the leaves of a tree. At the destination the reassembly would proceed by combining sibling leaves and replacing the mother node by the combined fragment. This scheme would require a last fragment flag for each set of siblings unless the degree of branching were fixed. The variable length fragment names are a complexity it would be nice to avoid.

The sequence number scheme would call for each fragment to carry a fragment sequence number and a data length. Each original message could start with the fragment sequence number zero. When it was necessary to fragment a message the first fragment would carry the original fragment sequence number, while the second fragment would carry a fragment sequence number equal to the fragment sequence number of the first fragment plus the data length of the first fragment, and so on. The last fragment

would have to carry a last fragment flag. At the reassembly point adjacent fragments could be combined until the message was complete. Two messages are adjacent when the fragment sequence number of the second is equal to the fragment sequence number of the first plus the data length of the first.

Message Identifier

Each message to be fragmented must have an identifier unique to this end to end address pair, so that the fragments of one message may be distinguished from the fragments of another message.

There is no need to coordinate the choice of this identifier between the source and destination. The source should choose the identifiers on messages it sends so as to avoid having two distinct messages to the same destination concurrently in circulation with the same identifier.

The choice of message identifier could be clock based or a simple sequence and the same sequence could be spread across all outgoing messages or separate sequences could be used for each destination address.

Data

Ah the data, at last a space for the reason we are going through all this nonsense. There is a data length field, and then that much data.

Error Control

Only hop to hop error control should be attempted in the internet protocol. Specific host level protocols such as TCP can provide for end to end error control. The same checksum field is used to protect the communication between the source host and the first hop, and between the last hop and the destination host, as is used between hops.

Message Format

The internet protocol message format has the following fields:

Version
Flags
Data Identifier
Acknowledgement Identifier
Type of Service
Destination Address
Fragment Information
  Message Identifier
  Fragment Sequence Number
  Last Fragment Flag
Data Length
Data
Checksum

The Version field is a 4 bit field which indicates the version of the internet protocol the remainder of the message conforms to. This proposal describes version 0.

The Type of Service field is a 4 bit field specifing which handling type is desired.

The Flags are a set of bits which specify the presence or absence of values in certain fields, in particular there is a data flag which indicates the data identifier field is meaningful (and that the data length is nonzero), and a acknowledgement flag which indicates the acknowledgement identifier field is meaningful.

The Data Identifier is a 16 bit field that distinguishes this message from other active messages on this host to hop, hop to hop, or hop to host, link.

The Acknowledgement Identifier is a 16 bit field that specifies which message among the active messages on this host to hop, hop to hop, or hop to host, link this acknowledgement pertains to.

The Destination Address field is a variable length extensible address composed of 4 bit chunks. The first chunk is the length of the address in chunks, except that the value 0 indicates that the next two chunks as a 8 bit number indicate the length.

This address extension scheme actually goes on indefinitely: we

Internet Host - Hop Interface

start with the length of the length equal to one, ll=1, if the value of the length is zero, l=0, then the next two chunks, ll←ll+1, are taken to be the length, if that length value is zero, l=0, then the next ll←ll+1 chunks are taken to be the length, and so on.

The Fragmentation Information consists of three things: the message identifier which is a 16 bit field, the fragment sequence number which is a 16 bit field, and the last fragment flag which requires one bit.

The Data Length is a 16 bit field whose value is the number of octets in the Data field.

The Data field is as many octets of arbitrary data as specified in the data length field.

The Checksum is a 16 bit field whose value is a hop by hop computed checksum that covers the entire message.

Internet Hop Protocol

The internet hop by hop protocol has nearly been described, at least by implication, in the preceeding section. In this section i will try to add a few details about the use of the fields in the hop by hop protocol.

The Type of Service field specifies the handling type is desired. The intent here is for the hop module to tailor its treatment of this messages according to the value of this field. As previously indicated one kind of tailoring is a trade off between delay and reliability. I expect substantially more thought will be needed before a reasonable set of cases can be established for type of service.

The Flags are bits which specify the presence or absence of values in certain fields:

data flag - indicates the data identifier is meaningful

acknouledgement flag - indicates the acknouledgement identifier is meaningful.

The Data Identifier field distinguishes this message from other active messages on this host to hop, hop to hop, or hop to host, link.

The intention is that for each hop module to select its own data identifier to put on a messages for transmission to the next hop module. That each hop modules should choose identifiers to avoid having two active messages on this link with the same identifier should be obvious.

The Acknowledgement Identifier specifies which message among the active messages on this host to hop, hop to hop, or hop to host, link this acknowledgement pertains to.

The hop module copies the data identifier of a message, lets call it A, it receives into the acknowledgement identifier field and sets the acknowledgement identifier flag in a message, lets call it B, traveling to the sender of message A to acknowledge correct receipt and take responsibility for message A.

The Destination Address field is processed by the hop module and the portion of the address consumed by the hop module is (chunk by chunk) circular shifted to the end of the address, bring the part of the address to be processed by the next hop to the beginning of the address string.

Fragmentation is performed if nesessary. Reassembly is left for the destination host.

Fragment Handling Procedures

Necessary Fragment Information Fields

Message Identifier
Data Length
Fragment Sequence Number
(initially zero in each new message)
Last Fragment Flag
(initially set in each new message)

Fragmentation Procedure

Split the data.

Copy the internet header from original message (or fragment) to each fragment.

Replace the data length field of each fragment by the new data length.

Replace the fragment sequence number in each fragment by the correct value.

The correct value of the fragment sequence number for fragment N+1 is the fragment sequence number of fragment N plus the data length of fragment N.

Reset the last fragment flag in all but the last fragment.

Reassembly Procedure

If two fragments are of the same message and adjacent assemble them.

If a fragment has fragment sequence number zero and the last fragment flag set then it is a whole message.

Where assemble means form one fragment with

fragment sequence number set to the fragment sequence number of the first fragment

the data length set to the data length of the first fragment plus the data length of the second fragment

last fragment flag set to the last fragment flag of the second fragment

Where adjacent means the fragment sequence number plus the data length of the first fragment equals the fragment sequence number of the second fragment.

The Data Length is only modified if fragmentation is done.

The Data field is not examined at all.

The Checksum must be checked as the first step in processing each message. If the checksum is not correct the message is discarded. When a message is forwarded to the next hop the checksum is recomputed if any change has been made. Since almost always the destination address is changed the checksum must be recomputed. And, of course, if a message has been fragmented a new checksum is necessary for each fragment.

To review, the internet hop by hop protocol message format has the following fields:

FIELD	BITS
Version	4
Flags	
Data Identifier	1
Acknowledgement Identifier	1
Data Identifier	16
Acknowledgement Identifier	16
Type of Service	4
Destination Address	variable
Fragment Information	
Message Identifier	16
Fragment Sequence Number	16
Last Fragment Flag	1
Data Length	16.
Data	variable
Checksum	16

Internet Host Protocol

The internet host protocol is a slightly simplified TCP. The principal simplifications are that TCP no longer concerns itself with fragmentation, and that the addresses are of the same form as used in the internet hop protocol.

The TCP should use the extensible addresses used in the hop by hop protocol. This means that the TCP need not carry a distinct copy of the destination address since it can be reconstructed from the internet host - hop message format. However since the TCP needs both the source and destination addresses in each message it may be useful to carry both in the host level header (which is treated as part of the data by the internet hop by hop protocol).

Since TCP is no longer concerned about fragmentation the End of Letter flag is not necessary to any part of the TCPs internal workings. It may be a desirable feature for purposes of the TCP - user interface though.

Model of Communication

To pull all this together perhaps it would be helpful to have a scenario of a message transversing this system.

Model of Communication

First a source process turns some data over to a source host protocol module. The host protocol module packages the data up in the host level protocol. Then that package is wrapped up in the internet interface format and forwarded to an intenet hop module. The internet hop module then forwards the message according to the address processing rules.

At any hop along the way the message may be fragmented, and the fragments separately forwarded toward the destination.

When the message arrives at the destination it is first verified according to the internet hop by hop protocol (i.e. is the checksum correct?). Next the address is checked to be sure this really is the destination. Once this initial checking is done, the next step is to reassemble the fragments if that is necessary. Once the whole message is assembled it can be turned over to the host protocol module for processing. The host protocol module unpackages the data and passes it to the destination process.

Summary

A model of internetwork commuication has been presented that is based on components that separate and modularize the distinct functions of the host to host interaction controls and the hop by hop addressing and routing functions. An interface between these functional modules has been specified. It is argued that this approach is more appropriate than the attempts to make a single protocol cover both functions.

References

[1] Cerf, V. "Specification of TCP version 2," March 1977.