TCP And UDP Over IPX Networks With Fixed Path MTU

Status of this Memo

IESG Note:

   Internet Engineering Steering Group comment from the Area Director
   for Transport Services: Please note well that this memo is an
   individual product of the author. Implementation experience,
   particularly on the effectiveness of the protocols in dual-stack
   environments, is needed.

1. Introduction

   Most of network applications run on some sort of transports. And, if
   one is to let such applications to run over a foreign network
   protocol, the simplest way would be to allow the applications'
   transports to run over that network protocol. For TCP/IP
   applications, that transport is TCP or UDP. Hence, to let TCP/IP
   applications run over IPX, we would need to have TCP and UDP run
   over IPX. And, once TCP and UDP are allowed to run over IPX, all TCP
   and UDP based applications, such as HTTP for WWW, or NFS, can easily
   be made to work over IPX networks.

   DLsw is another example of such applications. As it is a TCP
   application (and TCP requires IP), the administrator is forced to run
   IP on his network in order to support DLsw. If the site was an IPX
   shop, it means that he now must manage IP protocol/addresses in
   addition to IPX. If TCP could be made to run on IPX, then he would
   not have to add IP to his repertoire of network protocols to manage.

   TCP/IPX allows TCP/IP applications to run over IPX networks by
   letting TCP and UDP run over IPX. And this memo specifies the packet
   format and operational procedures for running TCP and UDP over IPX.

2. Running UDP Over IPX

   Since UDP datagrams can be up to 64K octets long, and the size of IPX
   packet is limited to that of the path MTU, large UDP datagrams must
   be fragmented. And, since IPX does not support fragmentation, large
   UDP datagrams must be fragmented before they are passed to IPX. For
   that purpose, a new protocol called IPXF (IPX Fragmentation layer),
   is invented. UDP must run on IPXF rather than directly on IPX. IPXF
   layer is described in section 4.

   To IPXF service users, IPXF behaves just like IPX except that IPXF
   accepts datagram larger than the IPX path MTU. As such, we describe
   UDP in this section as if it is running on IPX.

   UDP must send and receive the packets on IPX/IPXF socket 0x9092.
   Though it may be possible to send a packet from sockets other than
   0x9092, such sockets cannot receive UDP datagram destined to a well
   known socket 0x9092. Hence, the bidirectional communcation may not
   be established if a socket other than 0x9092 is used to send UDP
   datagram. For that reason. UDP/IPX does not allow source sockets
   other than 0x9092. If a datagram with source socket number other
   than 0x9092 is received, UDP/IPX should discard the packet silently.
   (And increment udpInDatagramErr MIB counter if it is instrumented.)

   UDP over IPX uses the IPX packet type 4, a normal IPX packet type.
   The IPX packet type has no meaning to TCP/IPX protocol. It simply is
   a number required by IPX for general IPX packets.

   See Appendix B.1 and B.2 for UDP over IPX packet format.

   The UDP/IPX checksum uses a pseudo header similar to UDP/IP pseudo
   header. The only difference is that IP addresses and protocol ID are
   replaced by IPX addresses and socket numbers.

   See Appendix B.3 for the UDP/IPX pseudo header format.

3. Running TCP Over IPX

   Unlike UDP, TCP runs directly over IPX. Since IPX does not support
   fragmentation, no TCP segment sent over IPX can be larger than the
   path MTU for the connection. The discovery of the path MTU is
   outside of scope of this paper. If the implementation does not have
   a way to dynamically determine the path MTU for each connection, it
   should at least allow a way to statically configure a reasonable
   value for all connections. For example, if the internetwork made of
   ethernets only, the user may configure the segment size to be 1470
   including the TCP header. If the configuration of the segment size
   is not possible, the implementation should assume that the IPX path

MTU is 576 octects, and not send any TCP segment larger than 546
octets including TCP header. That will result in IPX packet of 576
octets which is the minimum path MTU for IPX. The implementation is
then advised to comunicate the configured/default segment size to the
peer TCP by exchanging MSS option.

Note that this memo does not preclude the possibility of running TCP
over IPXF instead of IPX. Running on IPXF can be done in the same
manner as running UDP over IPXF. However, in general, TCP should
refrain from sending large segments that may result in fragmentation.
Hence, running TCP over IPXF is not recommended.

The IPX socket number 0x9091 is reserved for the TCP. All TCP packets
must be sent from and received on the socket 0x9091. If the received
TCP/IPX packet has the source IPX socket number other than 0x9091,
the packet should be discarded silently. (And increment tcpInErrs MIB
counter if it is instrumented.)

TCP, like UDP, uses IPX packet type 4. The IPX packet type has no
meaning to TCP/IPX protocol. It is packet type required by IPX for
general IPX packets.

See appendix A.1 for TCP/IPX packet format.

The TCP pseudo header, used in checksuming for TCP over IPX, is
similar to TCP pseudo header for TCP over IP. Again, the difference
is that IPX addresses and IPX socket number are substituted in place
of IP addresses and IP protocol number.

See Appendix A.2 for the TCP/IPX pseudo header format.

4. IPXF Layer

A large UDP datagram cannot be sent directly over IPX as IPX does not
support datagrams larger than the path MTU. Hence, large UDP
datagrams must be fragmented before it can be sent over IPX. To have
large UDP datagrams fragmented, UDP runs over IPXF layer instead of
running directly IPX.

IPXF users treats IPXF as if it is IPX layer. That is, they pass
datagrams to IPXF specifying the destination IPX address/socket along
with the packet. They also must set the source socket number of the
datagram to its actual IPX socket number, as it would when sending
packets to IPX layer. (For UDP, both source and destination sockets
are 0x9092.)

Datagrams passed to IPXF can be upto 64K octets long.

IPXF fragments a datagram as necessary, prepends each fragment with the IPXF header and send them to the IPX socket 0x9093 in the destination IPX address. The actual destination socket number (0x9092 for UDP) in the orignal IPX datagram is preserved in IPXF header. Refer to Appendix B.2 for UDP/IPXF/IPX packet format.

The largest possible IPX datagram that can be sent over the IPX path is limited by the path MTU size. The mechanism to discover the path MTU is outside of the scope of the paper. If an IPXF implementation does not have a mean to determine the path MTU, it should assume that the largest IPX packet size is 576. In that case, any UDP datagram larger than 546 octects will have to be fragmented.

If the datagram does not require fragmentation, IPXF acts as a null layer. That is, the whole packet is directly sent to the actual IPX destination socket without the IPXF fragmentation header. Refer to Appendix B.1 for UDP/IPX packet format without the IPXF header.

An IPXF user receives datagrams by opening a socket with IPXF just as it would with IPX. For example, UDP opens the socket 0x9092 with IPXF to receive UDP datagrams. IPXF, in turn, opens IPX socket of the same number with IPX, so that unfragmented packets directed to that socket will be delivered by IPX directly to the IPXF user.

IPXF fragments are received by IPXF on the IPX socket 0x9093. The receiving IPXF then reassembles the fragments into a complete IPX datagram, restores the actual detination IPX socket number from the IPXF header and delivers the reassembled IPX datagram to its actual recipient designated by the restored socket number.

Upon receiving a fragment, IPXF must ignore the source socket number in the IPX header of the fragment. The source IPX socket field in IPX header contains the actual source of the IPX datagram. As such, the source IPX socket number in IPX header usually is not 0x9093, and it is meaningful only to the actual recepient of the assembled datagram.

The fragmentation/reassembly algorithm used by IPXF is identical to that of IP, except for the following exceptions: 1) the offset of fragments are measured in units of octets rather than in units of 8 octets. 2) if the receiving IPXF does not have sufficient resource for the reassembly, it should discard fragments immediately. The receiving IPXF can determine if it has sufficient resources by looking at the length of the original datagram included in every fragment.

Note that, though it is required only for UDP in this memo, IPXF can also be used by any protocol that requires IPX fragmentation support.

5. TCP/IPX Checksuming

   TCP/IPX is checksummed in exactly same manner as TCP/IP. It uses 16
   bit 1's complement of 1's compliment sum of all 16 bit words in the
   pseudo header and text. See Appendix A.2 and B.3 for the pseudo
   header format for TCP and UDP.

6. Multiplexing

   TCP and UDP data over IPX are delivered to the application in the
   same manner as in TCP/IP. That is, they are delivered to the most
   specific matching endpoint, with the match made on local port, remote
   port, local IPX address and remote IPX address.

   When TCP or UDP is running over both IPX and IP, the connection
   endpoint also identifies the network layer on which the endpoint is.
   Hence, the triplet of network address, network address family, and
   the port number forms the socket. And, the endpoint match must be
   made on the the network address familty as well.

   For exmple, an endpoint bound to IPX network layer would be
   identified by AF_IPX, IPX address and TCP port number. On the other
   hand, endpoints bound to IP network layer would be identified by
   AF_IP, IP address, and TCP port. Finally, endpoints not bound to any
   network layer would be identified by AF_UNSPEC and TCP port.

   First, an attempt is made to deliver the data to the most specific
   endpoint that is bound to the network layer that the packet arrived
   from. If there is no such endpoint, then the packet is delivered to
   the best matching endpoint that is not bound to any network layer at
   all. For example, if the packet arrived over IPX network, then the
   packet is delivered to the most specific matching endpoint that is
   bound to IPX. If there is no matching endpoint over IPX, then it is
   delivered to an endpoint that did not specify any network layer.

   The use of endpoints not bound to any network layer is similar to
   TCP/IP endpoints with no IP address bound to it. Such endpoints are
   usually used for listening for connection requests from any of the
   interfaces within the host. Similarly, endpoints with no network
   layer bound to it are used to field the connection requests from any
   of the network layers.

Acknowledgement

   The author wishes to thank following folks, in alphabetical order,
   and others for their helpful comments and contributions to the work:
   Lester Bird, Doug Kogan, Greg Minshall and Don Provan.

Security Considerations

    Security issues are not discussed in this memo.

Author's Address

    Tae Sung
    Novell, Inc.
    2180 Fortune Drive
    San Jose, California, 95131

    Phone: (408)577-8439
    EMail: tae@novell.Com

Appendix A.1 - TCP/IPX Packet Format

   A TCP/IPX Packet has following format:

```
        +-------+-------+-------+-------+
        | IPX Checksum  | IPX Pkt Len   |
        +-------+-------+-------+-------+
        | Zero  |IPX PT | IPX Dest -
        +-------+-------+-------+-------+
          Network | IPX Dest -
        +-------+-------+-------+-------+
          Node                         |
        +-------+-------+-------+-------+
        | IPX Dest Skt  | IPX Src -
        +-------+-------+-------+-------+
          Network       | IPX Src -
        +-------+-------+-------+-------+
          Node                         |
        +-------+-------+-------+-------+
        | IPX Src Skt   | TCP Header and
        +--------------+-------+-------+
          Data...
        +----...
```

   IPX PT field contains the IPX packet type. It is set to 4 for
   TCP/IPX packet.

   Both Src Skt and Dest Skt field in IPX header must be set to 0x9091
   for TCP/IPX packet. If the Src Skt is not set to 0x9091, the
   receiving TCP/IPX should discard the packet silently. (And increment
   tcpInErrs mib object if it is instrumented.)

Appendix A.2 - TCP/IPX Pseudo Header Format

    TCP/IPX uses following pseudo header to compute checksum:

```
            +-------+-------+-------+-------+
            | IPX Src Network               |
            +-------+-------+-------+-------+
            | IPX Src Node
            +-------+-------+-------+-------+
                            | IPX Src Skt   |
            +-------+-------+-------+-------+
            | IPX Dest Network              |
            +-------+-------+-------+-------+
            | IPX Dest Node
            +-------+-------+-------+-------+
                            | IPX Dest Skt  |
            +-------+-------+-------+-------+
            | Zero          | TCP Length    |
            +-------------+-------------+
```

    IPX Src/Dest Network/Node/Skt are the fields from the IPX header.
    TCP Length is the IPX Pkt Len minus the IPX header length in octets.

    Note that IPX Src Skt is expected to be 0x9091 for TCP. As such, one
    may insert 0x9091 in IPX Src Skt field rather than getting the value
    from IPX header. Then the implementation will not have to check the
    IPX Src Skt field in the fast path since the checksum failure will
    also cover the unexpected value. In that case, the implementation
    may want to examine if the checksum failure was due to the IPX Src
    Skt value other than 0x9091, so that it can increment appropriate
    counter, if proprietary counters other than tcpInErrs are used.

Appendix B.1 - UDP/IPX Packet Format without Fragmentation

    IPXF transmits UDP packets over IPX in this format if the UDP
    datagram does not have to be fragmented:

```
                +-------+-------+-------+-------+
                | IPX Checksum  | IPX Pkt Len   |
                +-------+-------+-------+-------+
                | Zero  |IPX PT | IPX Dest -
                +-------+-------+-------+-------+
                  Network       | IPX Dest -
                +-------+-------+-------+-------+
                  Node                          |
                +-------+-------+-------+-------+
                | IPX Dest Skt  | IPX Src -
                +-------+-------+-------+-------+
                  Network       | IPX Src -
                +-------+-------+-------+-------+
                  Node                          |
                +-------+-------+-------+-------+
                | IPX Src Skt   | UDP Header and
                +--------------+-------+-------+
                  Data...
                +----...
```

    The IPX PT field contains IPX packet type. It should be set to 4 for
    all UDP/IPX packets.

    Both IPX Src Skt and IPX Dest Skt field must be set 0x9092. The
    receiving UDP/IPX should discard the packet silently if the IPX Src
    Skt field is not set to 0x9092. (And increment udpInErrors mib
    object if it is instrumented.)

Appendix B.2 - UDP/IPX Packet Format With Fragmentation

   IPXF transmits fragmented datagrams over IPX in the following format:

```
           +-------+-------+-------+-------+
           | IPX Checksum  | IPX Pkt Len   |
           +-------+-------+-------+-------+
           | Zero  |IPX PT | IPX Dest -
           +-------+-------+-------+-------+
             Network       | IPX Dest -
           +-------+-------+-------+-------+
             Node                  |
           +-------+-------+-------+-------+
             IPX Dest Skt  | IPX Src -
           +-------+-------+-------+-------+
             Network       | IPX Src -
           +-------+-------+-------+-------+
             Node                  |
           +-------+-------+-------+-------+
           | IPX Src Skt   | IPXF Offset   |
           +---------------+-------+-------+
           | IPXF Frag Identification      |
           +-------------------------------+
           | IPXF Dest Skt | IPXF DG Len   |
           +-------------------------------+
           | UDP Header and Data ...
           +--------...
```

   The IPX PT field contains IPX packet type. It is set to the value
   set by the IPXF user in the IPX packet passed to IPXF. (UDP sets it
   to 4.)

   IPX Dest Skt field must be set to 0x9093 for all IPXF Packets.

   The value for IPX Src Skt field is variable, and must be set to the
   actual IPX socket number of the IPXF user. (For example, it must be
   set to 0x9092 for UDP.)

   IPXF Offset field indicates where the fragment belongs in the
   datagram. The offset is measured is octet from the begining of the
   UDP datagram. The first fragment has the offset of 0.

   IPXF Frag Identification field is assigned a same value by the sender
   for all fragements belonging to the same datagram. The receiver then
   uses this field to reassemble all fragments with same ID into a
   datagram.

IPXF Dest Skt field contains the IPX socket number of the actual
recipient that the reassembled datagram will be delivered to. (It is
0x9092 for UDP.) All fragments of a datagram must have the same
value in this field.

IPXF DG Len field is the total length of the IPX datagram before the
fragmentation. The sender should set it to the value of IPX Pkt Len
of the original IPX datagram. All fragments of a IPX datagram must
have the same value in this field.

Appendix B.3 - UDP/IPX Pseudo Header Format

   UDP/IPX uses following pseudo header for computing the checksum:

```
              +-------+-------+-------+-------+
              | IPX Src Network               |
              +-------+-------+-------+-------+
              | IPX Src Node
              +-------+-------+-------+-------+
                              | IPX Src Skt   |
              +-------+-------+-------+-------+
              | IPX Dest Network              |
              +-------+-------+-------+-------+
              | IPX Dest Node
              +-------+-------+-------+-------+
                              | IPX Dest Skt  |
              +-------+-------+-------+-------+
              | Zero          | UDP Length    |
              +-------------+-------------+
```

   IPX Src/Dest Network/Node/Skt fields are from the IPX packet. Note
   that, if UDP is running over IPXF, the IPX Dest Skt field in IPX
   packet header is copied over from IPXF header before the reassembled
   IPX packet is delivered to UDP, Hence, the pseudo header must be
   derived from the reassembled IPX header.

   UDP Length is from UDP header.

   Note that IPX Src Skt is expected to be 0x9092 for UDP. As such, one
   may insert 0x9092 in IPX Src Skt field rather than getting the value
   from IPX header. Then the implementation will not have to check the
   IPX Src Skt field in the fast path since the checksum failure will
   also cover the unexpected value. In that case, the implementation
   may want to examine if the checksum failure was due to the IPX Src
   Skt value other than 0x9092, so that it can increment appropriate
   counter, if proprietary counters other than udpInDatagramErr are
   Datagr