MTU and MSS Tutorial

Dr. E. Garcia, admin@miislita.com

Published: November 16, 2009. Last Update: November 16, 2009.

Abstract – This tutorial covers maximum transmission unit (*MTU*), maximum segment size (*MSS*), PING, NETSTAT, and fragmentation. Expressions relevant to these concepts are systematically derived and explained.

Keywords: maximum transmission unit, *MTU*, maximum segment size, *MSS*, PING, NETSTAT

1 *MTU* and *MSS*

As discussed in the ***IP Packet Fragmentation Tutorial*** (http://www.miislita.com/internet-engineering/ip-packet-fragmentation-tutorial.pdf) and elsewhere (1 - 3), the data payload (*DP*) of an IP packet is defined as the packet length (*PL*) minus the length of its IP header (*IPHL*),

$$DP = PL - IPHL \qquad \text{(Eq 1)}$$

where the maximum *PL* is defined as the *Maximum Transmission Unit (MTU)*. This is the largest IP packet that can be transmitted without further fragmentation. Thus, when $PL = MTU$

$$DP = MTU - IPHL \qquad \text{(Eq 2)}$$

However, an IP packet encapsulates a TCP packet such that

$$DP = TCPHL + MSS \qquad \text{(Eq 3)}$$

where *TCPHL* is the length of the TCP header and *MSS* is the data payload of the TCP packet, also known as the *Maximum Segment Size (MSS)*. Combining Equations 2 and 3 leads to

$$MSS = MTU – IPHL – TCPHL \qquad \text{(Eq 4)}$$

Figure 1 illustrates the connection between *MTU* and *MSS* –for an IP packet decomposed into three fragments.

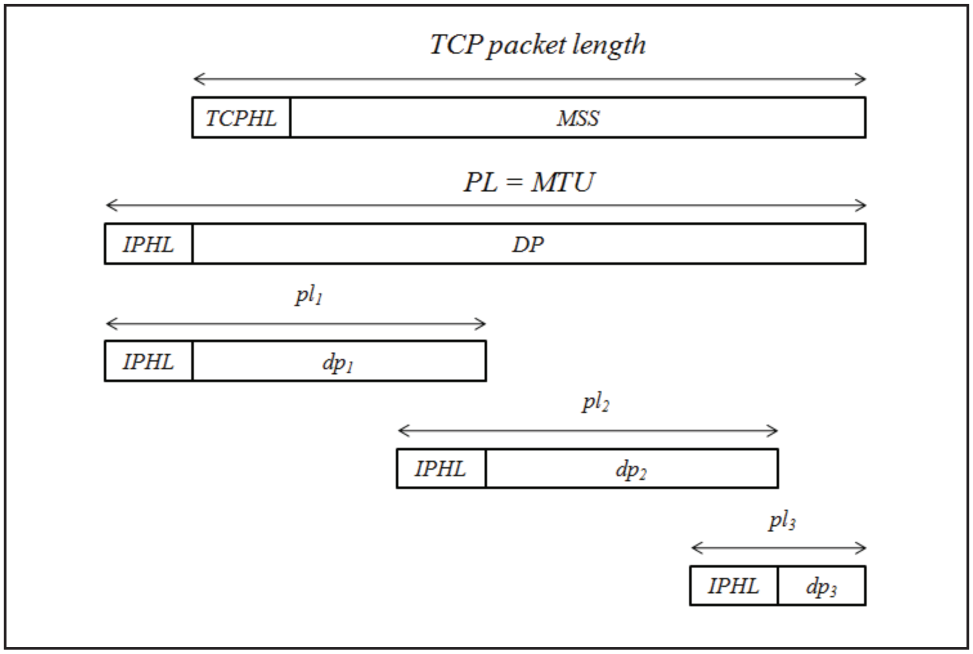


Figure 1. Fragmentation example where $MTU = PL = pl_1 = pl_2 > pl_3$ and $DP = dp_1 + dp_2 + dp_3 = PL – IPHL$.

Typically, IP and TCP headers are 20 bytes long. Thus,

$$MSS = MTU - 40 \quad \text{(Eq 5)}$$

If IP or TCP options are specified, the *MSS* is further reduced by the number of bytes taken up by the options (OP), each of which may be one byte or several bytes in size.

$$MSS = MTU - 40 - OP \quad \text{(Eq 6)}$$

In Windows 2000, Windows XP, and Windows Server 2003, support for additional TCP options, such as time stamps, can increase the typical overhead of the TCP and IP headers. Including padding, the overhead can go up to 52 or more bytes.

2 *MSS* Readjustment

When a TCP connection is established between two hosts, these exchange their *MSS* values. The smaller of the two *MSS* values is used for the connection. If the TCP packets are destined for a remote network, the Don't Fragment (DF) flag is set (DF = 1) in the IP header. This prevents data fragmentation along the path between the two hosts and intermediate links.

If the DF flag is not set (DF = 0) and an intermediate link has an *MTU* smaller than the IP packet being routed ($MTU < PL$), the router should inform the sending host of this *MTU* and that the packet cannot be forwarded further without fragmentation. To inform the host, the router sends an Internet Control Message Protocol, "Destination Unreachable-Fragmentation Needed and DF Set message. This is an ICMP Type 3, Code 4 error message containing the limiting *MTU* encountered.

Upon receiving this ICMP error message, TCP automatically adjusts sender's *MSS* as prescribed by Equations 5 or 6 using the *MTU* specified in the ICMP message, so that any further packets sent on the connection path are no larger than the new *MTU*. The entire process is transparent to end users.

3 *MSS* values from ICMP Messages

A simple technique for experimentally determining *MTU* and *MSS* values along the connection path between two hosts consists in intentionally sending an ICMP message encapsulated as an IP packet, with the Don't Fragment flag set to 1 (DF = 1). As an ICMP message consumes 8 bytes (64 bits) this *ICMPDP* data payload must be deducted from the data payload of the IP packet.

$$DP = PL - IPHL - ICMPDP \quad \text{(Eq 7)}$$

Equation 4 can then be expressed as

$$MSS = MTU - IPHL - ICMPDP \quad \text{(Eq 8)}$$

Figure 2 illustrates this –compare with Figure 1.

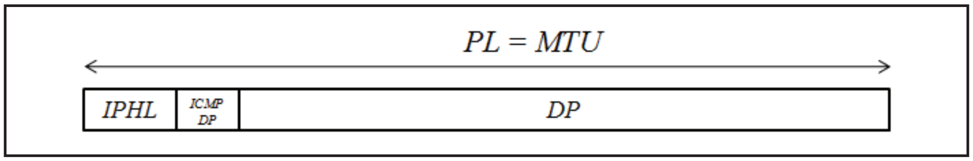


Figure 2. ICMP message payload encapsulated as an IP packet.

Since *IPHL* = 20 bytes and *ICMPDP* = 8 bytes,

$$MSS = MTU - 28 \quad \text{(Eq 9)}$$

That it, the *MSS* value obtained from ICMP data will be 28 bytes smaller than the size of the *MTU* used and 12 bytes (40 – 28) higher than the value obtained from TCP data. Either way, the *MTU* value remains unaltered as it depends on the network type or media used. For *Ethernet v2* networks, this value is 1,500 bytes and for token ring networks is 4,096 bytes.

4 Remote *MTU* values

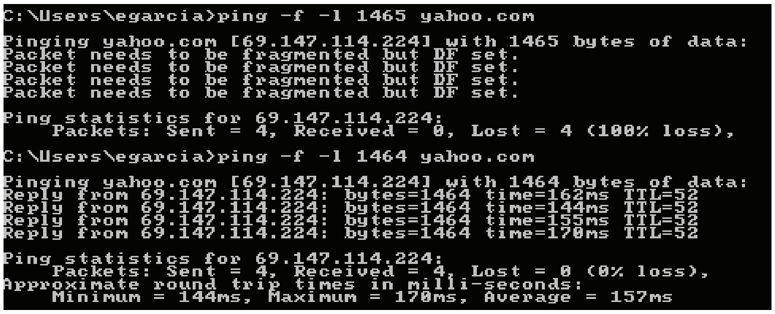
The technique described in Section 3 allows the experimental determination of *MTU* values along the path between two hosts and intermediate links. In practice, this can be implemented with the PING utility. To access its helper, type ***ping/?*** in the *Windows Command Prompt* (WCP) tool. The query that does the analysis is

ping –f –l size host

where ***ping*** invokes the PING utility, ***–f*** sets the Don't fragment flag (DF = 1) of the IP packet, ***-l*** is the length flag of buffer ***size***, and ***host*** is the domain name or IP address of the target host. By default the tool sends 4 packets, but this can be overwritten by adding the ***–n*** flag. Thus, ***ping –n 1 –f –l size host*** instructs PING to send one packet.

By virtue of Equation 9, 28 bytes will be added to the buffer ***size*** specified. By modifying on a trial-and-error basis the ***size*** value, eventually an ICMP error message Type 3, Code 4 will be received. This ICMP error is indicative of the fact that the *Maximum Segment Size (MSS)* and therefore *MTU* have been exceeded. Since for *Ethernet v2* networks *MTU* = 1,500 bytes, the *MSS* should be around the 1,472 bytes mark. As *MTU* values are networks-specific (e.g., for token ring networks *MTU* = 4,096 bytes), the technique can be used to identify the type of network involved.

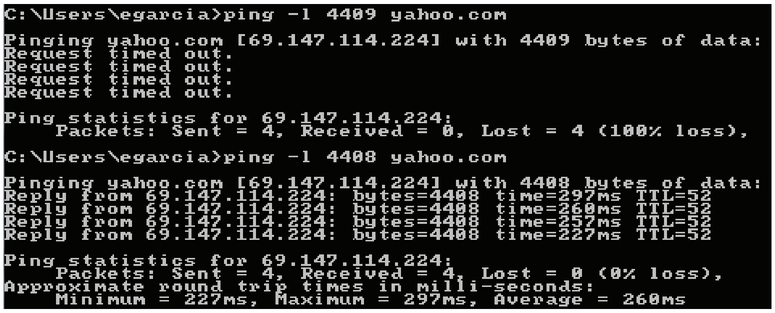
Figure 3 shows PING records, obtained by pinging Yahoo.com. An *MSS* value of 1,464 bytes was obtained. Thus from Equation 9, *MTU* = 1,492 bytes. This value is close to the 1,500 bytes mark, obtainable when no additional overhead due to options, padding, or noisy conditions is consumed. Note the ICMP Type 3, Code 4 error message triggered by incrementing the buffer size by 1 byte.



```
C:\Users\egarcia>ping -f -l 1465 yahoo.com

Pinging yahoo.com [69.147.114.224] with 1465 bytes of data:
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.
Packet needs to be fragmented but DF set.

Ping statistics for 69.147.114.224:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\egarcia>ping -f -l 1464 yahoo.com

Pinging yahoo.com [69.147.114.224] with 1464 bytes of data:
Reply from 69.147.114.224: bytes=1464 time=162ms TTL=52
Reply from 69.147.114.224: bytes=1464 time=144ms TTL=52
Reply from 69.147.114.224: bytes=1464 time=155ms TTL=52
Reply from 69.147.114.224: bytes=1464 time=170ms TTL=52

Ping statistics for 69.147.114.224:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 144ms, Maximum = 170ms, Average = 157ms
```

Figure 3. PING records for Yahoo.com with DF = 1.

It should be stressed that not declaring the ***–f*** flag in the above queries implies that DF = 0; i.e., that the packets can be fragmented. Accordingly, the buffer size threshold determined will correspond to that of the largest packet that can be sent along the network path, without exceeding the buffer reassembling timer. Exceeding this threshold should trigger a ' Request timed out' or Type 11, Code 1 error message (Time Exceeded, Fragment Reassembly Time Exceeded). This is illustrated in Figure 4.



```
C:\Users\egarcia>ping -l 4409 yahoo.com

Pinging yahoo.com [69.147.114.224] with 4409 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 69.147.114.224:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

C:\Users\egarcia>ping -l 4408 yahoo.com

Pinging yahoo.com [69.147.114.224] with 4408 bytes of data:
Reply from 69.147.114.224: bytes=4408 time=297ms TTL=52
Reply from 69.147.114.224: bytes=4408 time=260ms TTL=52
Reply from 69.147.114.224: bytes=4408 time=257ms TTL=52
Reply from 69.147.114.224: bytes=4408 time=227ms TTL=52

Ping statistics for 69.147.114.224:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 227ms, Maximum = 297ms, Average = 260ms
```

Figure 4. PING records for Yahoo.com with DF = 0.

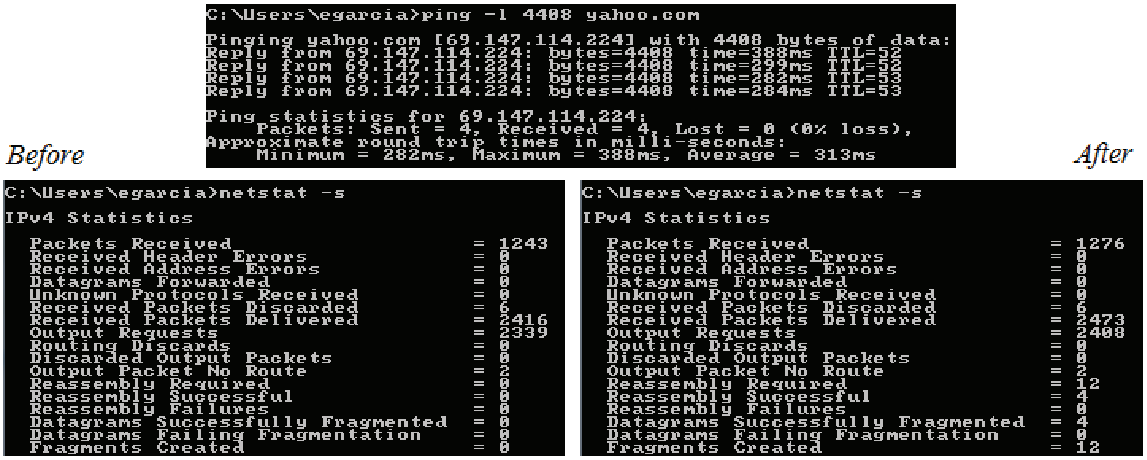
If the *MTU* along the connection path is known, the number of fragments of a packet can be determined as described in the ***IP Packet Fragmentation Tutorial*** (http://www.miislita.com/internet-engineering/ip-packet-fragmentation-tutorial.pdf).

The number of fragments can also be computed directly with NETSTAT, a network statistics tool, by issuing the ***netstat –s*** query and checking the IP statistic sections (IPv4 or IPv6) of the output. The default is **IPv4 Statistics** which is the first section of the output. The entries to check are **Fragments Created** (number of fragments) and **Datagrams Successfully Fragmented** (number of packets). The ratio of these two entries is the number of fragments per packet. However, since NETSTAT displays cumulative statistics per TCP/IP session, it is necessary to run NETSTAT before and after pinging a host with fragmentable packets of same size.

This is what we recommend to do:

1. Start a fresh TCP/IP session by restarting the local host.
2. Run NETSTAT by typing the query ***netstat –s***.
3. From the IPv4 section of the output, record the number of **Fragments Created** (*FC*) and number of **Datagrams Successfully Fragmented** (*DSF*). Each entry should be 0 unless packets have been already sent and fragmented. If you elect not to start a fresh TCP/IP session, these entries might not be 0.
4. Ping a remote host by querying ***ping –l size host***, where ***size*** > *MTU*. For instance in the previous example, ***size*** was 4,408, ***host*** was yahoo.com, and 4 packets were sent. If there is no packet lost, four packets should be successfully fragmented.
5. Repeat Steps 2 and 3, record the new *FCs* and *DSF*s, and compute the number of fragments per packet as shown in Figure 5.

In Figure 5, the subscripts indicate that NETSTAT queries were issued before and after pinging the host. The output shows that each packet sent was broken up into 3 fragments for a total of 12 fragments. Based on this output only, one should be able to work backward the above equations and recalculate the *MTU* and *MSS* values.



```
C:\Users\egarcia>ping -l 4408 yahoo.com

Pinging yahoo.com [69.147.114.224] with 4408 bytes of data:
Reply from 69.147.114.224: bytes=4408 time=388ms TTL=52
Reply from 69.147.114.224: bytes=4408 time=299ms TTL=52
Reply from 69.147.114.224: bytes=4408 time=282ms TTL=53
Reply from 69.147.114.224: bytes=4408 time=284ms TTL=53

Ping statistics for 69.147.114.224:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 282ms, Maximum = 388ms, Average = 313ms
```

```
C:\Users\egarcia>netstat -s

IPv4 Statistics

  Packets Received                   = 1243
  Received Header Errors             = 0
  Received Address Errors            = 0
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 6
  Received Packets Delivered         = 2416
  Output Requests                    = 2339
  Routing Discards                   = 0
  Discarded Output Packets           = 0
  Output Packet No Route             = 2
  Reassembly Required                = 0
  Reassembly Successful              = 0
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 0
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 0
```

```
C:\Users\egarcia>netstat -s

IPv4 Statistics

  Packets Received                   = 1276
  Received Header Errors             = 0
  Received Address Errors            = 0
  Datagrams Forwarded                = 0
  Unknown Protocols Received         = 0
  Received Packets Discarded         = 6
  Received Packets Delivered         = 2473
  Output Requests                    = 2408
  Routing Discards                   = 0
  Discarded Output Packets           = 0
  Output Packet No Route             = 2
  Reassembly Required                = 12
  Reassembly Successful              = 4
  Reassembly Failures                = 0
  Datagrams Successfully Fragmented  = 4
  Datagrams Failing Fragmentation    = 0
  Fragments Created                  = 12
```

$$number\ of\ fragments\ per\ packet\ =\ n\ =\ \frac{FC_{After}\ -\ FC_{Before}}{DSF_{After}\ -\ DSF_{Before}}\ =\ \frac{12\ -\ 0}{4\ -\ 0}\ =\ 3$$

Figure 5. Experimental determination of fragments with NETSTAT.

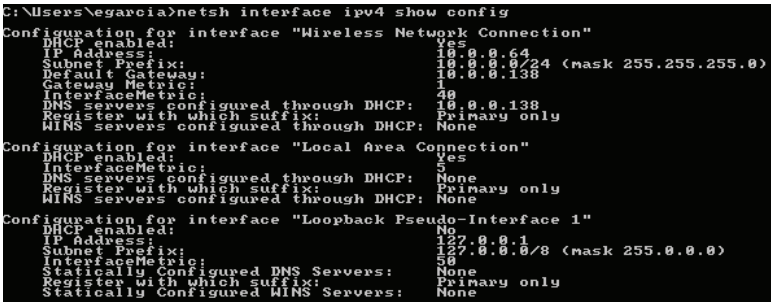
5 Local *MTU* values

The technique described in Sections 3 and 4, however, cannot be used to estimate *MTU* values of local hosts because a given machine can have more than one network interface, each with their own *MTU* value. In addition, the PING utility accepts buffer sizes within the 1 to 65,500 range and the *MTU* of a loopback pseudo-interface is arbitrarily preset to $2^{32} – 1$ = 4294967295.

Fortunately, the several *MTUs* used by a local host can be determined with NETSH, a network shell tool. This is a Microsoft utility that allows local or remote configuration of network settings. Its menu helper is invoked by typing ***netsh/?*** in the WCP. The interface records of a local host using IPv4 can be retrieved with the query

netsh interface ipv4 show config

Figure 6 shows the output produced by this query.



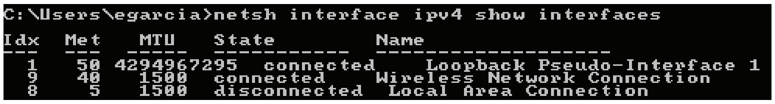
```
C:\Users\egarcia>netsh interface ipv4 show config

Configuration for interface "Wireless Network Connection"
    DHCP enabled:                         Yes
    IP Address:                           10.0.0.64
    Subnet Prefix:                        10.0.0.0/24 (mask 255.255.255.0)
    Default Gateway:                      10.0.0.138
    Gateway Metric:                       1
    InterfaceMetric:                      40
    DNS servers configured through DHCP:  10.0.0.138
    Register with which suffix:           Primary only
    WINS servers configured through DHCP: None

Configuration for interface "Local Area Connection"
    DHCP enabled:                         Yes
    InterfaceMetric:                      5
    DNS servers configured through DHCP:  None
    Register with which suffix:           Primary only
    WINS servers configured through DHCP: None

Configuration for interface "Loopback Pseudo-Interface 1"
    DHCP enabled:                         No
    IP Address:                           127.0.0.1
    Subnet Prefix:                        127.0.0.0/8 (mask 255.0.0.0)
    InterfaceMetric:                      50
    Statically Configured DNS Servers:    None
    Register with which suffix:           Primary only
    Statically Configured WINS Servers:   None
```

Figure 6. Network interfaces configuration settings of a local host.

Note that three interfaces are used by the local host: wireless, LAN, and the loopback pseudo-interface. The corresponding *MTU* values are retrieved by replacing the ***config*** keyword with ***interfaces***; i.e., by typing

netsh interface ipv4 show interfaces

Figure 7 displays the new output produced by this query.



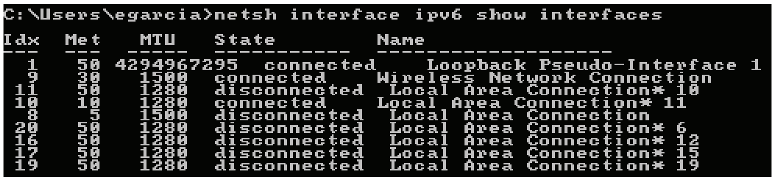
```
C:\Users\egarcia>netsh interface ipv4 show interfaces

Idx  Met   MTU   State        Name
---  ---  ------  ------------  ---------------------
  1   50 4294967295  connected    Loopback Pseudo-Interface 1
  9   40   1500  connected    Wireless Network Connection
  8    5   1500  disconnected  Local Area Connection
```

Figure 7. Networking Interface *MTU* values for IPv4.

The figure lists the Identification of the Network Interface (IDx), Interface Metric (Met), *MTU*, and the Status and Name of the interface. When the host has one Networking Interface Card (NIC) the IDx of this card will be Local Area Connection. A similar output can be retrieved for IPv6 by querying

netsh interface ipv6 show interfaces

The corresponding output is given in Figure 8.



```
C:\Users\egarcia>netsh interface ipv6 show interfaces

Idx  Met   MTU   State        Name
---  ---  ------  ------------  ---------------------
  1   50 4294967295  connected    Loopback Pseudo-Interface 1
  9   30   1500  connected    Wireless Network Connection
 11   50   1280  disconnected  Local Area Connection* 10
 10   10   1280  connected    Local Area Connection* 11
  8    5   1500  disconnected  Local Area Connection
 20   50   1280  disconnected  Local Area Connection* 6
 16   50   1280  disconnected  Local Area Connection* 12
 17   50   1280  disconnected  Local Area Connection* 15
 19   50   1280  disconnected  Local Area Connection* 19
```

Figure 8. Networking Interface *MTU* values for IPv6.

6 Exercises

1. What is the *MSS* of each of the packets mentioned in Exercises 1 and 2 of the ***IP Packet Fragmentation Tutorial*** (http://www.miislita.com/internet-engineering/ip-packet-fragmentation-tutorial.pdf)? Assume default values for the TCP/IP headers and ICMP messages.

2. Determine the *MSS* value between your host and three search engine hosts other than Yahoo.com. Compare results.

3. Determine the network interfaces and *MTU* values used by your local machine.

4. Using NETSTAT, determine the maximum number of fragments per packet required to ping Darpa.org.

5. Identify a remote host wherein the buffer size threshold required for sending pings is the same regardless of the DF flag. How does this impact the analysis? What does this tell you about the connection path between the hosts?

References

1. Garcia, E. *IP Packet Fragmentation Tutorial*. Mi Islita.com; 2009.
   [On-line]. Available: http://www.miislita.com/internet-engineering/ip-packet-fragmentation-tutorial.pdf
2. Internet Engineering Task Force.
   [On-line]. Available: http://www.ietf.org
3. Microsoft.
   [On-line]. Available: http://msdn.microsoft.com/en-us/library/ms817967.aspx