AWK Scripts are very good in processing the data from the log (trace files) which we get from NS2. If you want to process the trace file manually, here is the detail
Here is a sample of trace file from NS2 (However ns2 supports a new type of trace file also), but this post will make you understand the old trace format only.

r 0.030085562 _0_ MAC --- 0 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
r 0.030110562 _0_ RTR --- 0 message 32 [0 ffffffff 1 800] ------- [1:255 -1:255 32 0]
s 1.119926192 _0_ RTR --- 1 message 32 [0 0 0 0] ------- [0:255 -1:255 32 0]

AWK Scripts are very good in processing the data column wise. For example
the first column in the above trace file represents r, s which indicates receive, sent respectively. If we want to trace the entire r and s alone from this trace file we can represent it as $1

S0
$1 represents ACTION
$2 Time
$3 Node ID
$4 Layer
$5 Flags
$6 seqno
$7 type
$8 Size
$14 Energy (if the network nodes includes EnergyModel)

To run the awk script in Linux,
gawk –f filename.awk filename.tr

So, it is necessary for the researchers to know the basics of awk scripts before they are used.
Here this post will let you know some of the scripts that were used to process the data (NB: all these codes were taken from various websites and you can refer those websites for further information).

To find the throughput of the Network

```
 1: BEGIN {
 2:       recvdSize = 0
 3:       startTime = 400
 4:       stopTime = 0
 5:   }
 6:
 7:   {
 8:            event = $1
 9:            time = $2
10:             node_id = $3
11:             pkt_size = $8
```

```
 12:             level = $4
 13:
 14:   # Store start time
 15:   if (level == "AGT" &;& event == "s" && pkt_size >= 512) {
 16:     if (time <; startTime) {
 17:             startTime = time
 18:             }
 19:       }
 20:
 21:   # Update total received packets' size and store packets arrival time
 22:   if (level == "AGT" &;& event == "r" && pkt_size >= 512) {
 23:       if (time >; stopTime) {
 24:             stopTime = time
 25:             }
 26:       # Rip off the header
 27:       hdr_size = pkt_size % 512
 28:       pkt_size -= hdr_size
 29:       # Store received packet's size
 30:       recvdSize += pkt_size
 31:       }
 32:   }
 33:
 34:   END {
 35:       printf("Average Throughput[kbps] = %.2f\t\t StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/
(stopTime-startTime))*(8/1000),startTime,stopTime)
 36:   }
```

To print the Congestion window size

```
 1: BEGIN {
 2:
 3: }
 4: {
 5: if($6=="cwnd_") {
 6:     printf("%f\t%f\n",$1,$7);
 7: }
 8: }
 9: END {
10:
11: }
```

To print packet Delivery ratio

```
BEGIN {
        sendLine = 0;
        recvLine = 0;
        fowardLine = 0;
}

$0 ~/^s.* AGT/ {
        sendLine ++ ;
}

$0 ~/^r.* AGT/ {
        recvLine ++ ;
}

$0 ~/^f.* RTR/ {
        fowardLine ++ ;
}

END {
        printf "cbr s:%d r:%d, r/s Ratio:%.4f, f:%d \n", sendLine, recvLine,
(recvLine/sendLine),fowardLine;
}
```

AWK Script for calculating the Send, Received, Dropped Packets, Received Packets, Packet Delivery Ratio and Average end to End Delay

```
 1: BEGIN {
 2: seqno = -1;
 3: droppedPackets = 0;
 4: receivedPackets = 0;
 5: count = 0;
 6: }
 7: {
 8: #packet delivery ratio
 9: if($4 == "AGT" &2amp;;& $1 == "s" && seqno < $6) {
10: seqno = $6;
11: } else if(($4 == "AGT") && ($1 == "r")) {
12: receivedPackets++;
13: } else if ($1 == "D" && $7 == "tcp" && $8 > 512){
14: droppedPackets++;
15: }
16: #end-to-end delay
17: if($4 == "AGT" &;& $1 == "s") {
18: start_time[$6] = $2;
19: } else if(($7 == "tcp") && ($1 == "r")) {
20: end_time[$6] = $2;
21: } else if($1 == "D" && $7 == "tcp") {
22: end_time[$6] = -1;
23: }
24: }
25:
26: END {
27: for(i=0; i<=seqno; i++) {
28: if(end_time[i] >; 0) {
29: delay[i] = end_time[i] - start_time[i];
30: count++;
31: }
32: else
33: {
34: delay[i] = -1;
35: }
36: }
```

```
37: for(i=0; i<count; i++) {
38: if(delay[i] >; 0) {
39: n_to_n_delay = n_to_n_delay + delay[i];
40: }
41: }
42: n_to_n_delay = n_to_n_delay/count;
43: print "\n";
44: print "GeneratedPackets = " seqno+1;
45: print "ReceivedPackets = " receivedPackets;
46: print "Packet Delivery Ratio = " receivedPackets/(seqno+1)*100
47: "%";
48: print "Total Dropped Packets = " droppedPackets;
49: print "Average End-to-End Delay = " n_to_n_delay * 1000 " ms";
50: print "\n";
51: }
```