IT 304: Computer Networks

Lab # 3: Performance study of CSMA/CD (IEEE 802.3) mechanism[*]

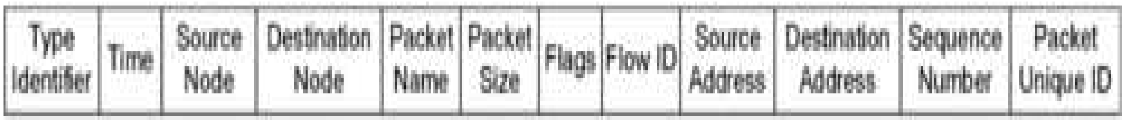
For the Week of August 12, 2013

Pre-lab preparation:

- Read the relevant portions of the textbook on CSMA/CD and Ethernet.
- Go through the Chapter 14 of NS manual to see how a LAN can be created and corresponding parameters be set.

1. **Aim:** The purpose of this lab is to understand the performance of the carrier sense multiple access (CSMA)/ collision detection (CD) (IEEE 802.3 standard is based on this) mechanism that is widely used in local area networks (LANs).

2. **CSMA/CD mechanism and Ethernet:** The carrier sense multiple access (CSMA) mechanism with collision detection (CD) is used in the link layer protocol Ethernet which was standardized as IEEE 802.3. The CSMA mechanism works on the listen before talk principle. Ethernet is the most commonly used link layer protocol in local area networks (LANs). In case of packet collisions Ethernet employs backoff algorithms for retransmissions.



Type Identifier	Time	Source Node	Destination Node	Packet Name	Packet Size	Flags	Flow ID	Source Address	Destination Address	Sequence Number	Packet Unique ID

Figure 1: The figure above shows the format of NS2 trace file.

3. **NS2 trace format:** The file format of NS2 packet trace is given in the Figure 1. A detailed description of the various fields is given below. It is important that you understand what different fields in the NS2 trace file represent for you to actually process the trace file using AWK/Perl scripts for computing the various performance metrics of interest.

   - Type identifier: Consists of a string which depends on the type of packet tracing object. Most widely used type identifiers are
     - "+" which represents a packet enque event.
     - "-" which represents a packet deque event.

[*]© Dr. Laxminarayana S Pillutla

- "r" which represents a packet reception event.
- "d" which represents a packet drop event.
- "c" which represents a packet collision event at the MAC level.
- "h" which represents an event when any node in a LAN gives it to "LAN" node (Note: A LAN bus is modeled in NS2 as a node instead of as a link).

- Time: Denotes the time at which the string corresponding to "Type identifier" is created.
- Source node and Destination node: Contains the source and destination IDs of the source and destination nodes of the tracing object.
- Packet name: Contains the name of packet type.
- Packet size: Contains the size of packet in bytes.
- Flags: Consists of a 7 digit string that is set to `` `-' `` if the corresponding flag is dialed. Otherwise it will be set as follows. The first is set to `` `E' `` if an explicit congestion notification (ECN) echo is enabled. The second is set to `` `P' `` if the priority in the IP header is enabled. The fourth is set to `` `A' `` if the corresponding TCP takes an action on a congestion (e.g., closes the congestion window). The fifth is set to `` `E' `` if the congestion has occurred. The sixth is set to `` `F' `` if the TCP fast start is used. Finally, the seventh is set to `` `N' `` when the transport layer protocol is capable of using ECN.
- Flow ID: Contains the flow ID specified in the field `fid_` of an IP packet header.
- Source address and Destination address: Contains the source and destination address of a packet specified in the IP packet header. For a flat addressing scheme the format of these two fields is `` `a.b' ``, where `` `a' `` is the address and `` `b' `` is the port.
- Sequence number: Contains the sequence number specified in packet header by a transport layer protocol.
- Packet Unique ID: Contains a unique ID stored in common packet header.

4. **Exercises:** As part of this lab we create a typical scenario in LANs where there are multiple users contending for transmission on a common channel. We assume that there are a total of $(N+1)$ nodes in the network of which N nodes are data sources and 1 node is a sink that sinks in the data sent by N sources.

   (a) Design a network simulation scenario with $N = 4$. Creation of nodes, attaching UDP agents and traffic sources/sinks, etc., can be done as per the previous labs. Ensure that these nodes are connected to a common LAN interface using the `newLan` object in NS2. Typical usage is given below:

   `set lan [$ns newLan $nodelist $opt(bw) $opt(delay)  -llType $opt(ll) -ifqType $opt(ifq)  -macType $opt(mac) -chanType $opt(chan)]` where `$ns` is an instance of Simulator class, `nodelist` contains the list of nodes and `opt` contains a list of LAN parameter values.

   Ensure that all the simulation related things are captured in the trace file.

   (b) In this step we shall do post processing of the generated simulation trace file. To accomplish this write an AWK script (a sample is given below) to do the following:

   i. Compute the total number of packets (P) successfully received by the sink node.
   ii. Compute the total number of packets dropped at different sources during the simulation. Also compute the average packet drop percentage at a sender. Explore on the possible reasons for packet drops.

iii. Compute the system throughput (= $\frac{P \times \text{packet\_size} \times 8}{\text{simulation\_time}}$) and efficiency (=ratio of system throughput and LAN data rate).

iv. Compute the individual throughputs of nodes $1, 2, 3$ and 4. Check that the sum of individual throughputs sum to the system throughput computed immediately above.

v. Look at the trace file for the collision profile of first packet (sequence number 0) generated by nodes $1, 2, 3$ and 4 respectively? In other words look at the time when the packet is generated, when the packet got transmitted and if they collided, who won the collision.

(c) In this part we shall run experiments by varying the number of nodes n and compute system efficiency, system throughput and packet drops for each n. Generate three different plots: system efficiency vs number of nodes, system throughput vs number of nodes and packet drops vs number of nodes. Comment on the nature of these plots.

(d) In this part we shall look at the "fairness" of Ethernet to various users in terms of (individual) throughputs. A popular fairness index is the so called Jain's fairness index which is given as

$$J(x_1, x_2, \cdots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \ , \tag{1}$$

where n denotes the number of nodes/users and x_i denotes the throughput of i^{th} user. The Jain's fairness index ranges from $\frac{1}{n}$ (worst case) to 1 (best case). If the value of Jain's fairness index is close to 1 then all the users received the same allocation of resources. **Compute the Jain's fairness index given above using the individual throughput values computed in the sub-section immediately above and comment on the fairness of Ethernet.**

5. **Processing of NS2 simulation trace using AWK scripts:** The following AWK script file computes the total number of packets received by a sink node from all the sources in the network. We assume the corresponding trace file to be `csmacd.tr`. In the `BEGIN` part of the code the `count` variable is initialized which counts the number of packets received. In the `END` part the total number of packets received is printed out onto the screen. In the `IF` condition the first and fifth fields of a given row in the trace file are compared against the corresponding strings to identify the "packet reception event" and the packet type (which in this case was set to CBR).

```
BEGIN
{
count = 0;
}
{
if ($1 == "r" && $5 == "cbr") {
count++;
}
}
END{
pktsize = $6;
printf("Total no of packets received from all nodes is:  %f \n",count);
}
```

If the above AWk script is saved in a file named: `count_pkts.awk` then it can be executed from the terminal using the following command: `awk -f count_pkts.awk csmacd.tr`