EXNO:7	**Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS**
DATE:	

AIM:

To Study Network simulator (NS).and Simulation of Congestion Control Algorithms using NS

ALGORITHM:

Step 1: Initialize pointers elm and elm2, and num_later variable to 1.

Step 2: Start a loop while elm is not NULL and num_later is within bounds.

Step 3: Update elm to the next element in the list.

Step 4: Check if elm is not NULL, if so, find the next data packet in the list.

Step 5: If a data packet is found, start another loop with elm2.

Step 6: In the nested loop, check conditions for seq_num and time.

Step 7: If conditions are met, do nothing; otherwise, remove and delete elm2.

Step 8: Update elm and elm2 for the next iteration of the nested loop.

Step 9: Continue looping until conditions are no longer met in the outer loop or no more elements exist.

Step 10: Optionally, there is a similar function (removeAcksRecvHistory) that focuses on removing elements with a specific type (DCCP_ACK) from the history table. The steps are similar but with this specific type in mind

PROGRAM:

```
#include <wifi_lte/wifi_lte_rtable.vh>

struct r_hist_entry *elm, *elm2;

int num_later = 1;

elm = STAILQ_FIRST(&r_hist_);

while (elm != NULL && num_later <= num_dup_acks_)

{

num_later;

elm = STAILQ_NEXT(elm, linfo_);

}

if (elm != NULL)

{
```

```
elm = findDataPacketInRecvHistory(STAILQ_NEXT(elm, linfo_));
if (elm != NULL)(
elm2 = STAILQ_NEXT(elm, linfo_);
while (elm2 != NULL){
if (elm2->seq_num_ < seq_num && elm2->t_recv_ <
time)
{
}
else
STAILQ_REMOVE(&r_hist_, elm2, r_hist_entry, linfo_);
delete elm2;
elm = elm2;
elm2 = STAILQ_NEXT(elm, linfo_);
}}}}
void DCCPTFRCAgent::removeAcksRecvHistory()
{
struct r_hist_entry *elm1 = STAILQ_FIRST(&r_hist_);
struct r_hist_entry *elm2;
int num_later = 1;
while (elm1 != NULL && num_later <= num_dup_acks_)
{
num_later;
elm1 = STAILQ_NEXT(elm1, linfo_);
}
if (elm1 == NULL)
return;
elm2 = STAILQ_NEXT(elm1, linfo_);
while (elm2 != NULL)
{
if (elm2->type_ == DCCP_ACK)
{
}
```

```
else
{
}
STAILQ_REMOVE(&r_hist_, elm2, r_hist_entry, linfo_);
delete elm2;
elm1 = elm2;
elm2 = STAILQ_NEXT(elm1, linfo_);
}
}
inline r_hist_entry
*DCCPTFRCAgent::findDataPacketInRecvHistory(r_hist_entry *start)
{
while (start != NULL && start->type_ == DCCP_ACK)
start = STAILQ_NEXT(start, linfo_);
return start;
}
```

RESULT:

Thus we have Studied Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.