NEW ADAPTATION METHOD OF TCP FOR MOBILE AD HOC NETWORKS

BY-

SHREY KHURANA(13BCE0505)

AKARSH GOYAL(13BCE0552)

OBJECTIVE

Here we presents a new approach that tries to adapt the Transmission Control Protocol (TCP) for use in Wireless ad-hoc networks (MANET). This new approach is a Hybrid TCP method where we have modified the original TCP method by using a cross layer solution to the legacy IEEE 802.11 and TCP. This is done because we face a lot of problems when we use TCP in ad-hoc networks.

Challenges

If we want to deploy TCP over MANET we have to face a number of issues these issues can be broadly classified under the following heads:

- **Channel Errors: The main causes for errors in the channel are signal Attenuation, Doppler's Shift, and Multipath Fading.**
- **Path Asymmetry: The wireless link between any mobile node and a base station is asymmetric in nature. This asymmetry can be of three types' bandwidth, loss rate, and route.**
- **Congestion: TCP is a violent Transport Layer Protocol that attempts to utilize the network resources to its fullest, this feature makes an MANET easily undergoes a situation called as Congestion. This Congestion in turn causes overflow of the buffer and increase in the link conflict thus degrading the overall performance of TCP.**

Challenges(cont)

- **Mobility: Mobility feature include increase in overhead as mobility causes link breakages route failure and network Partition between two nodes which in turn causes packet loss. The reestablishment of route from source node to the destination node depends on the factors like Routing Protocol used, Traffic Characteristics of the network and Mobility Pattern of the Mobile Nodes.**
- **Power Constraints: As known mobile nodes works on battery power which is limited, which means that the processing power is also limited also each node is acting as an end node as well as a router at the same time therefore a successful energy scheme must be applied to utilize the resources to its maximum.**

INTRODUCTION

The ad hoc networks are wireless networks without beforehand defined infrastructure, where each entity (node) communicates directly with its neighbor. To communicate with other entities, it is necessary to rely on other entities in order to allow data reach the destination. Therefore, it is very important that the entities are well positioned so they can establish links between them. This connection is achieved by the routing protocol.

A MANET consists of mobile platforms herein simply referred to as "nodes"--which are free to move about arbitrarily.A MANET is an autonomous system of mobile nodes.

TCP has been well tuned to provide services in traditional wired network environment. Due to its wide use in the Internet, it is desirable that TCP remains in use to provide reliable data delivery for communications within MANETs .

INTRODUCTION(CONT)

TCP does not deal properly with the specific effects occurring in MANETs. This is because TCP is a protocol developed initially for wired networks. TCP reacts the same way as wired networks and reduces the bandwidth what degrades the performance of the network. To improve TCP's performance in wireless networks generally and specially on the MANET, many solutions were proposed such as Loss differentiation algorithms with RTT and Solution using the signal power. But the main disadvantage of the first solution (RTT) is the mobility since it is applicable only in a static wireless environment where nodes of the network do not move. This is due to the fact that this solution is based on the round trip time RTT a TCP packet.

So we go for Hybid TCP.

Hybrid TCP

In the proposed solution, the signal strength is taken as a factor to determine the position of nodes in the wireless environment. It is used to calculate the estimated value of round trip time in the normal case (estimated RTT) and compare it with the real RTT to determine the cause of the loss and act according to the analysis.

The strength of this approach is that it is based on a mechanism which has no relation with the number of hops whether it is 1 hop or more. This feature overcomes the multihops problem. With respect to the performances, it allows the detection of the real cause of packet loss.

Our approach assumes that in order to differentiate congestion losses from the wireless losses, it may help to use another formula derived from the TCP formula to estimate the value of next RTT. This formula uses the lower value of signal strength the path that the packet takes.

I. error = measured RTT - prediction

2. new prediction = old prediction + I/a x error
= 7/a x old prediction + I/a x measured RTT

3. new variation = 3/4 x old variation + 1/4 x abs(error)

4. RTO = prediction + 4 x variation

CONT-

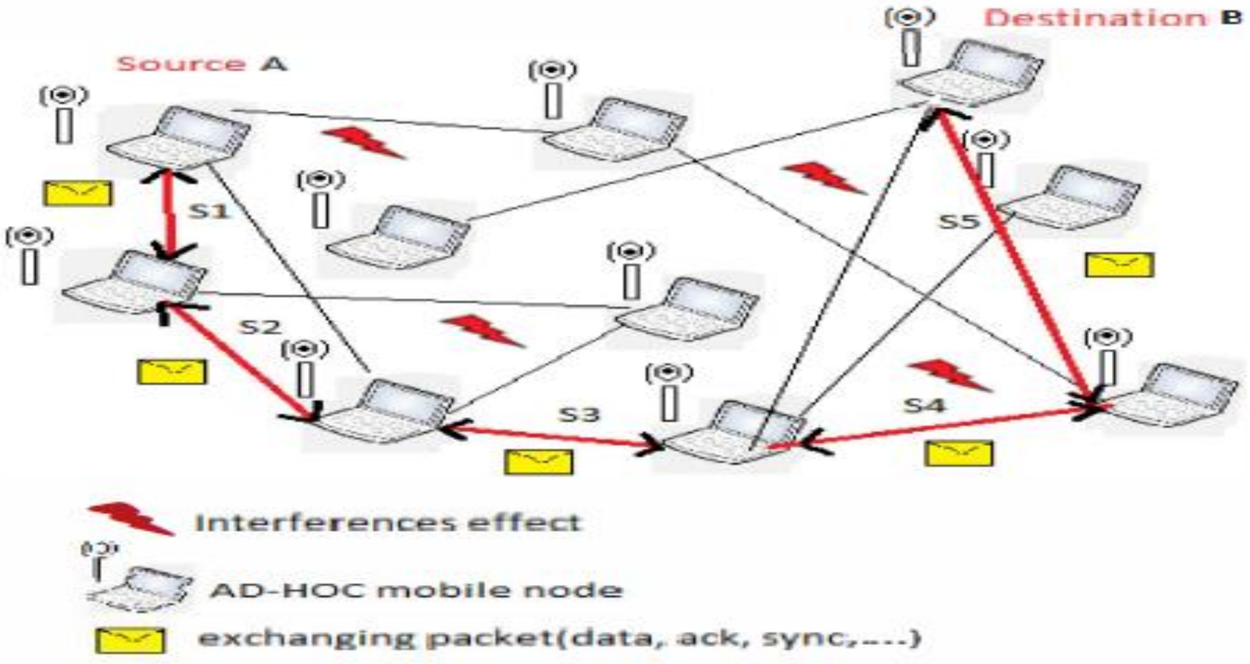
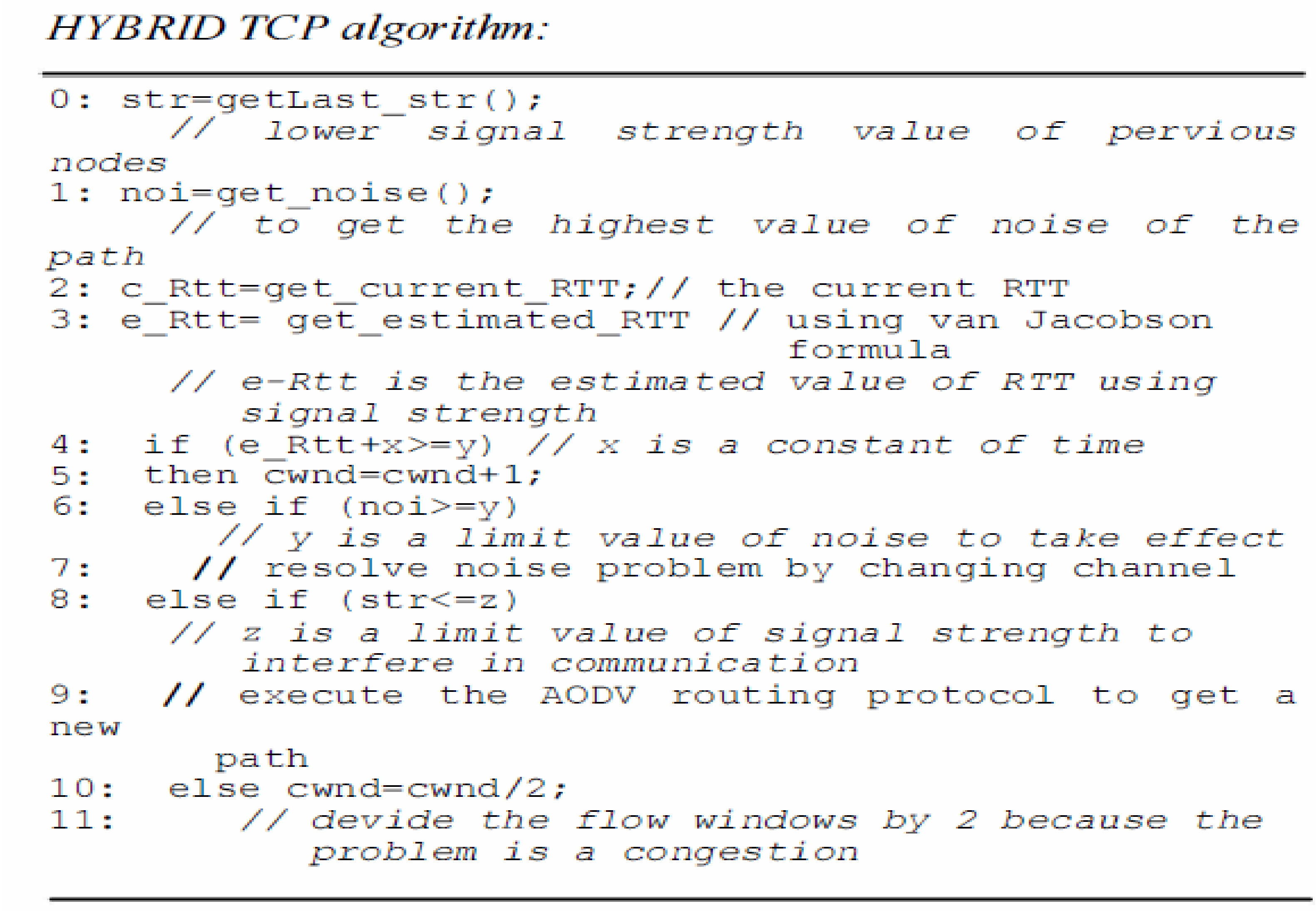


Figure 3. Wireless ad-hoc topology with interferences (Used topology).

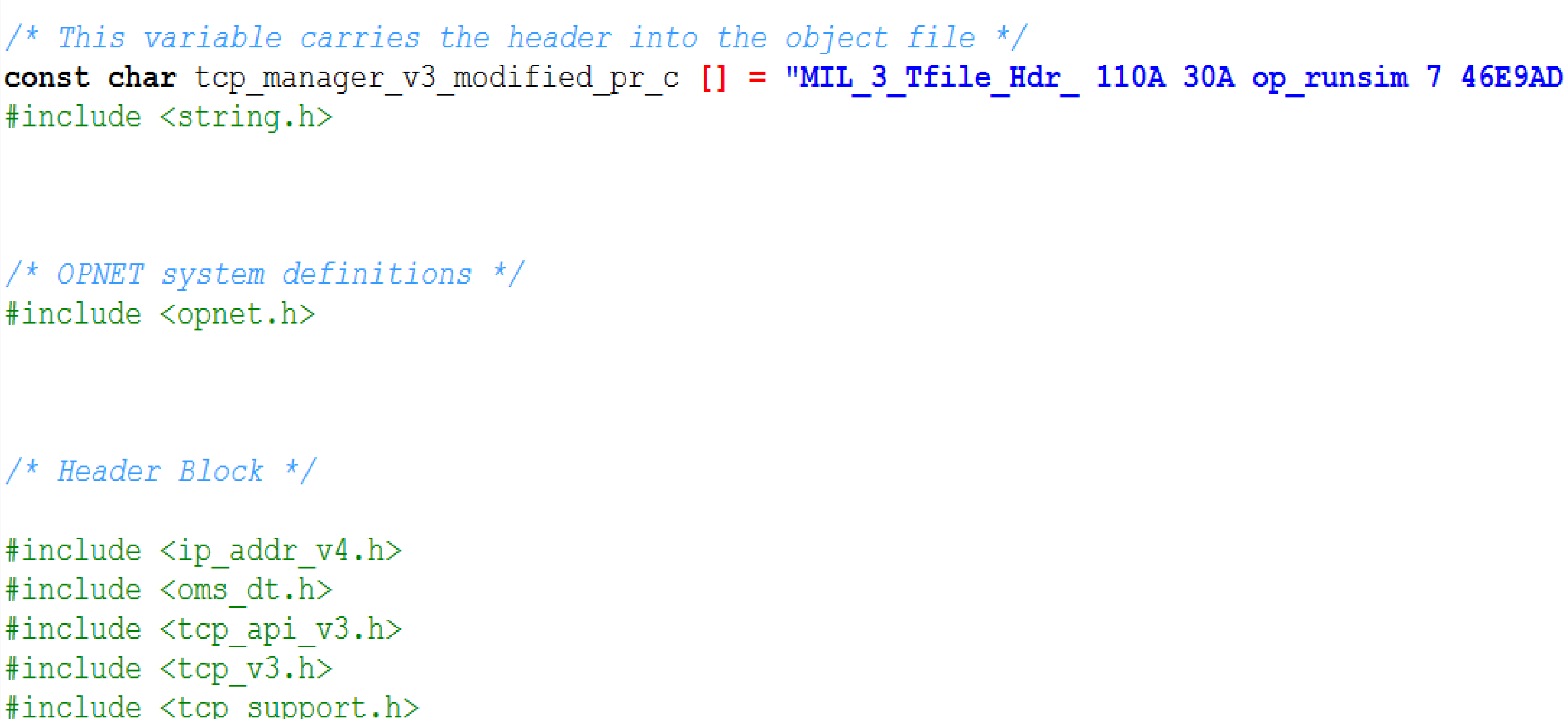
HYBRID TCP ALGORITHM



HYBRID TCP algorithm:

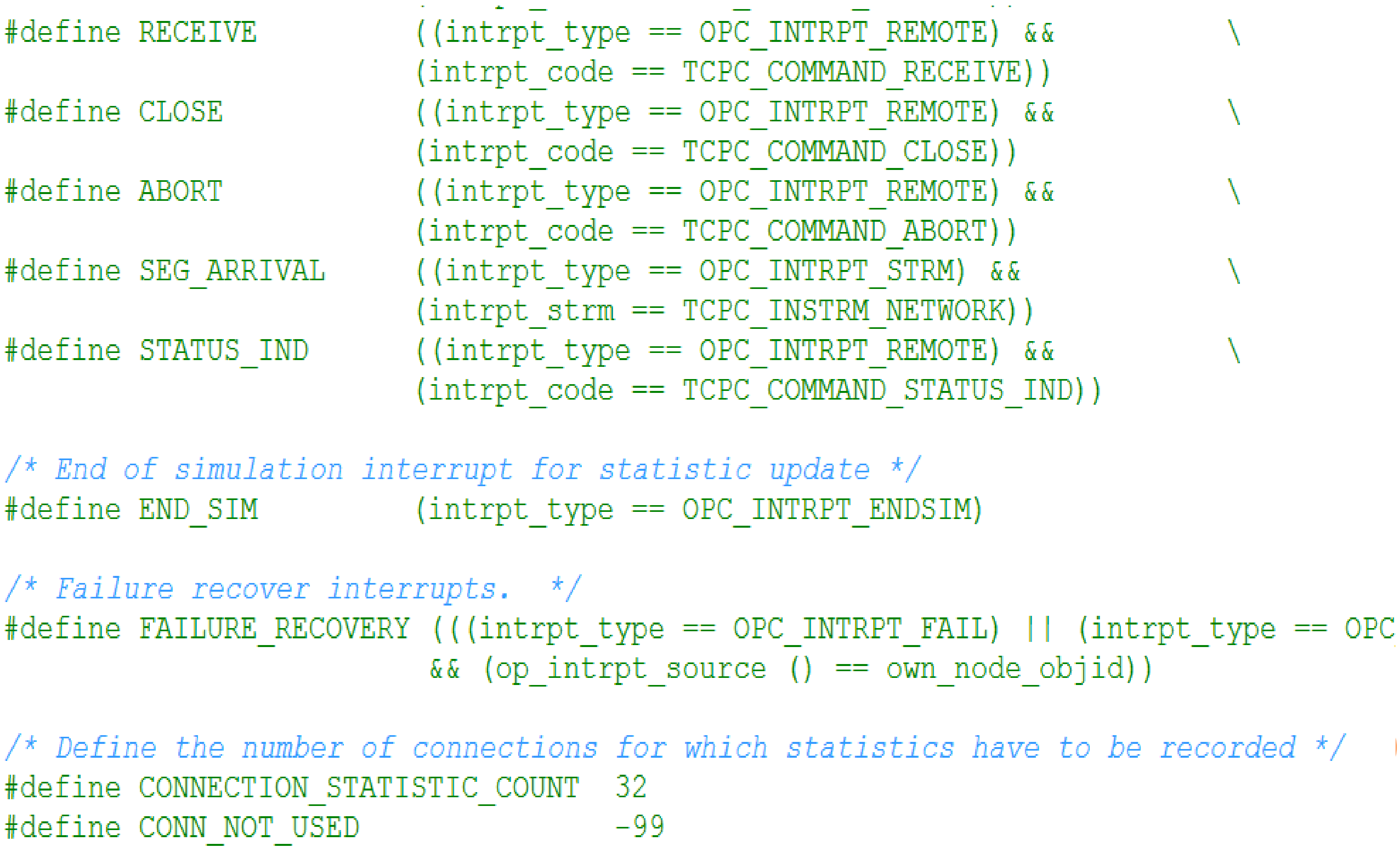
```
str=getLast_str();
    // lower signal strength value of pervious nodes
noi=get_noise();
    // to get the highest value of noise of the path
c_Rtt=get_current_RTT;// the current RTT
e_Rtt= get_estimated_RTT // using van Jacobson formula
    // e-Rtt is the estimated value of RTT using signal strength
 if (e_Rtt+x>=y) // x is a constant of time
 then cwnd=cwnd+1;
 else if (noi>=y)
     // y is a limit value of noise to take effect
   // resolve noise problem by changing channel
 else if (str<=z)
    // z is a limit value of signal strength to interfere in communication
  // execute the AODV routing protocol to get a new path
  else cwnd=cwnd/2;
      // devide the flow windows by 2 because the problem is a congestion
```

Code



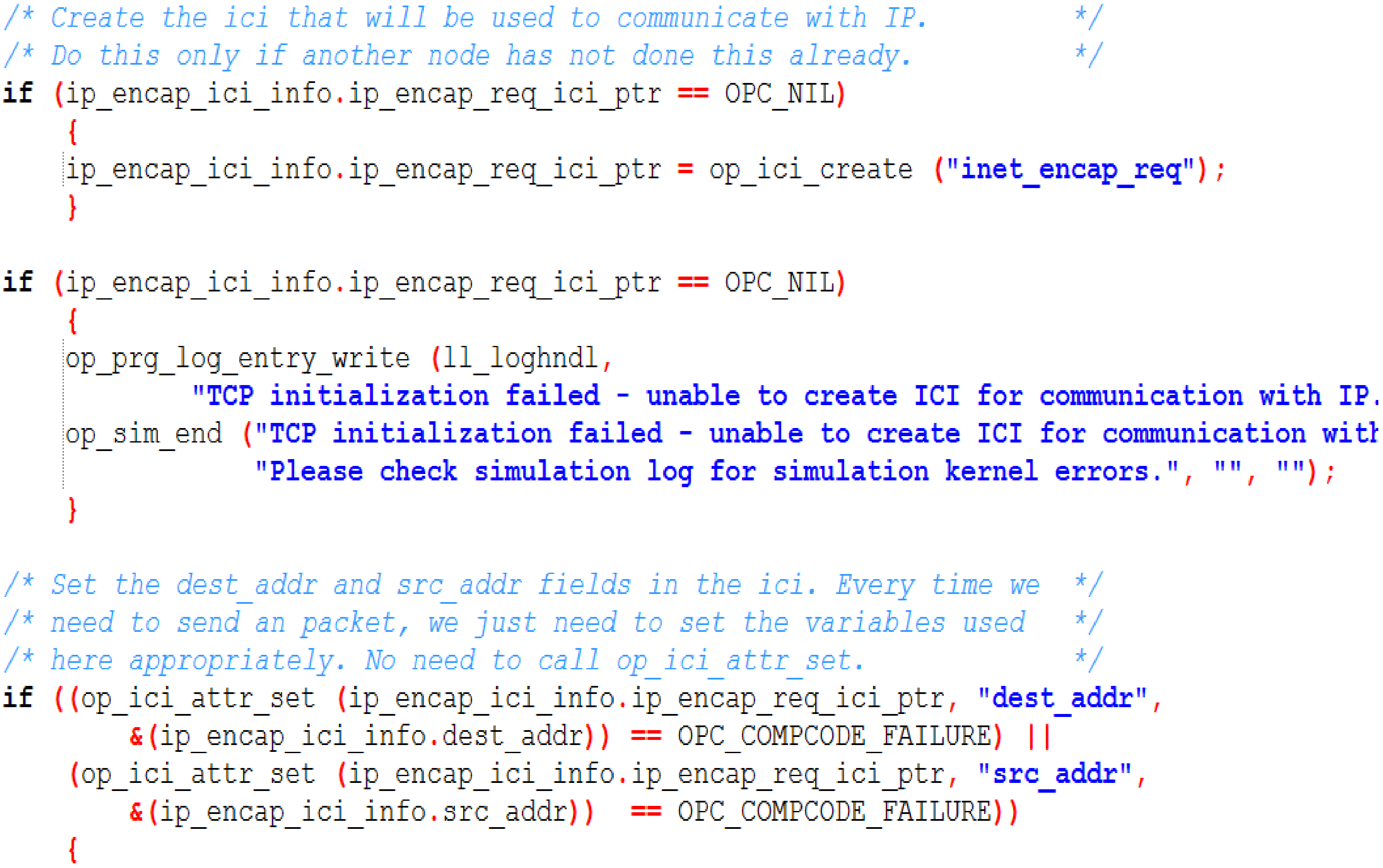
```
/* This variable carries the header into the object file */
const char tcp_manager_v3_modified_pr_c [] = "MIL_3_Tfile_Hdr_ 110A 30A op_runsim 7 46E9AD
#include <string.h>



/* OPNET system definitions */
#include <opnet.h>



/* Header Block */

#include <ip_addr_v4.h>
#include <oms_dt.h>
#include <tcp_api_v3.h>
#include <tcp_v3.h>
#include <tcp_support.h>
```

CODE (CONT-)



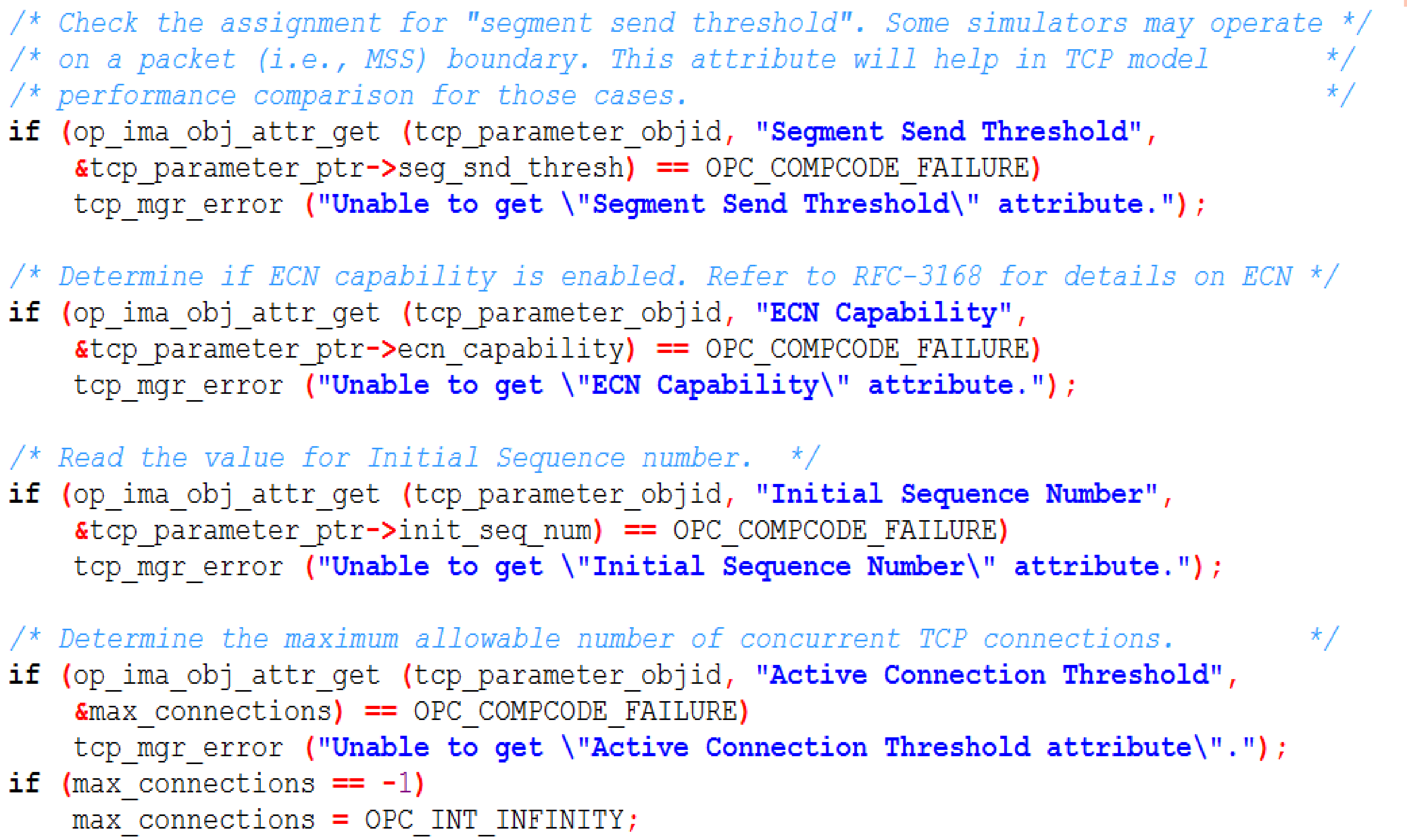
```
#define RECEIVE            ((intrpt_type == OPC_INTRPT_REMOTE) &&          \
                           (intrpt_code == TCPC_COMMAND_RECEIVE))
#define CLOSE              ((intrpt_type == OPC_INTRPT_REMOTE) &&          \
                           (intrpt_code == TCPC_COMMAND_CLOSE))
#define ABORT              ((intrpt_type == OPC_INTRPT_REMOTE) &&          \
                           (intrpt_code == TCPC_COMMAND_ABORT))
#define SEG_ARRIVAL        ((intrpt_type == OPC_INTRPT_STRM) &&            \
                           (intrpt_strm == TCPC_INSTRM_NETWORK))
#define STATUS_IND         ((intrpt_type == OPC_INTRPT_REMOTE) &&          \
                           (intrpt_code == TCPC_COMMAND_STATUS_IND))

/* End of simulation interrupt for statistic update */
#define END_SIM            (intrpt_type == OPC_INTRPT_ENDSIM)

/* Failure recover interrupts.  */
#define FAILURE_RECOVERY (((intrpt_type == OPC_INTRPT_FAIL) || (intrpt_type == OPC
                            && (op_intrpt_source () == own_node_objid))

/* Define the number of connections for which statistics have to be recorded */
#define CONNECTION_STATISTIC_COUNT  32
#define CONN_NOT_USED               -99
```

CODE (CONT-)



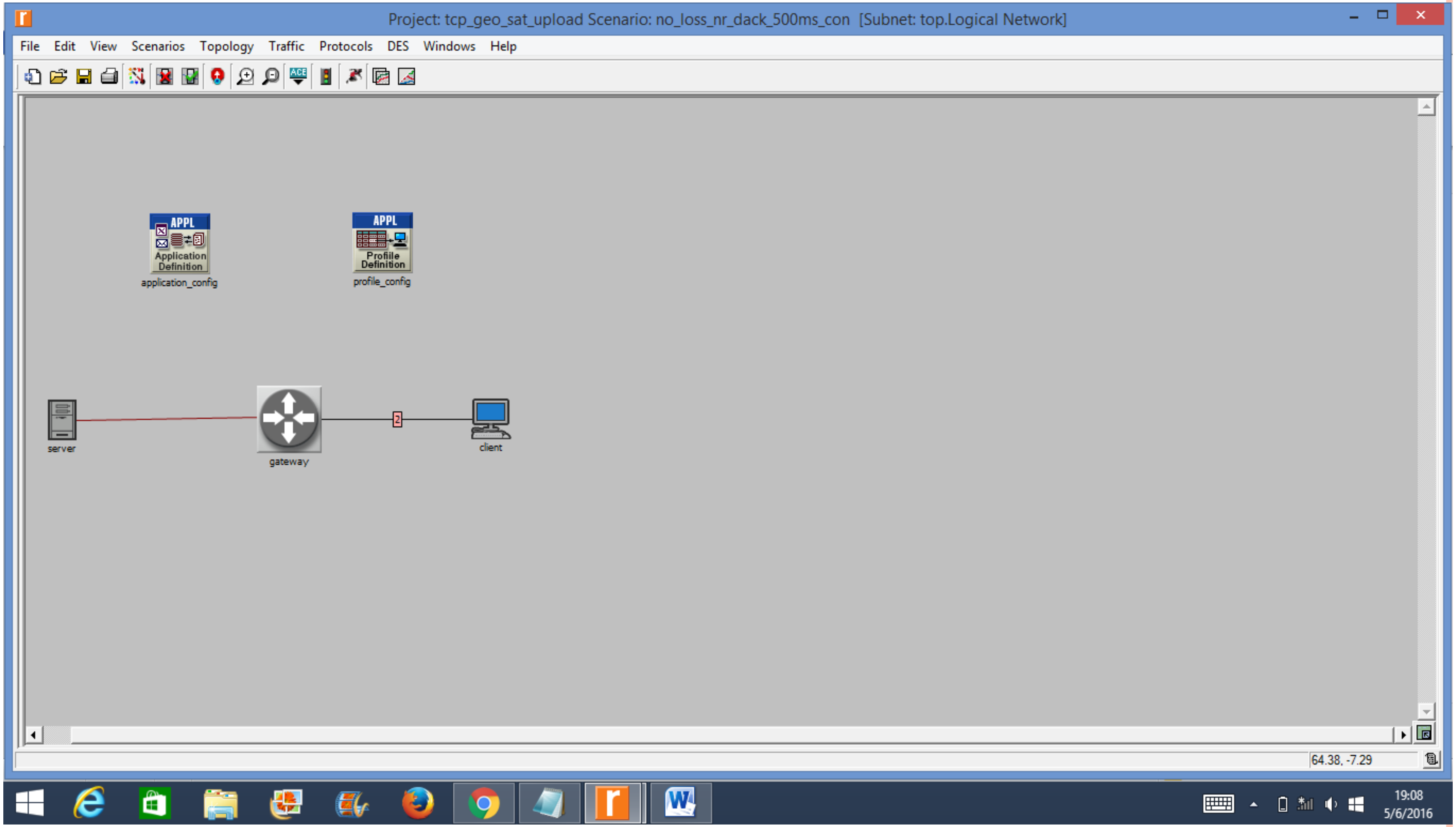
```
/* Create the ici that will be used to communicate with IP.        */
/* Do this only if another node has not done this already.         */
if (ip_encap_ici_info.ip_encap_req_ici_ptr == OPC_NIL)
    {
    ip_encap_ici_info.ip_encap_req_ici_ptr = op_ici_create ("inet_encap_req");
    }

if (ip_encap_ici_info.ip_encap_req_ici_ptr == OPC_NIL)
    {
    op_prg_log_entry_write (ll_loghndl,
            "TCP initialization failed - unable to create ICI for communication with IP.
    op_sim_end ("TCP initialization failed - unable to create ICI for communication witl
                "Please check simulation log for simulation kernel errors.", "", "");
    }

/* Set the dest_addr and src_addr fields in the ici. Every time we  */
/* need to send an packet, we just need to set the variables used   */
/* here appropriately. No need to call op_ici_attr_set.             */
if ((op_ici_attr_set (ip_encap_ici_info.ip_encap_req_ici_ptr, "dest_addr",
        &(ip_encap_ici_info.dest_addr)) == OPC_COMPCODE_FAILURE) ||
    (op_ici_attr_set (ip_encap_ici_info.ip_encap_req_ici_ptr, "src_addr",
        &(ip_encap_ici_info.src_addr))  == OPC_COMPCODE_FAILURE))
    {
```

CODE (CONT-)

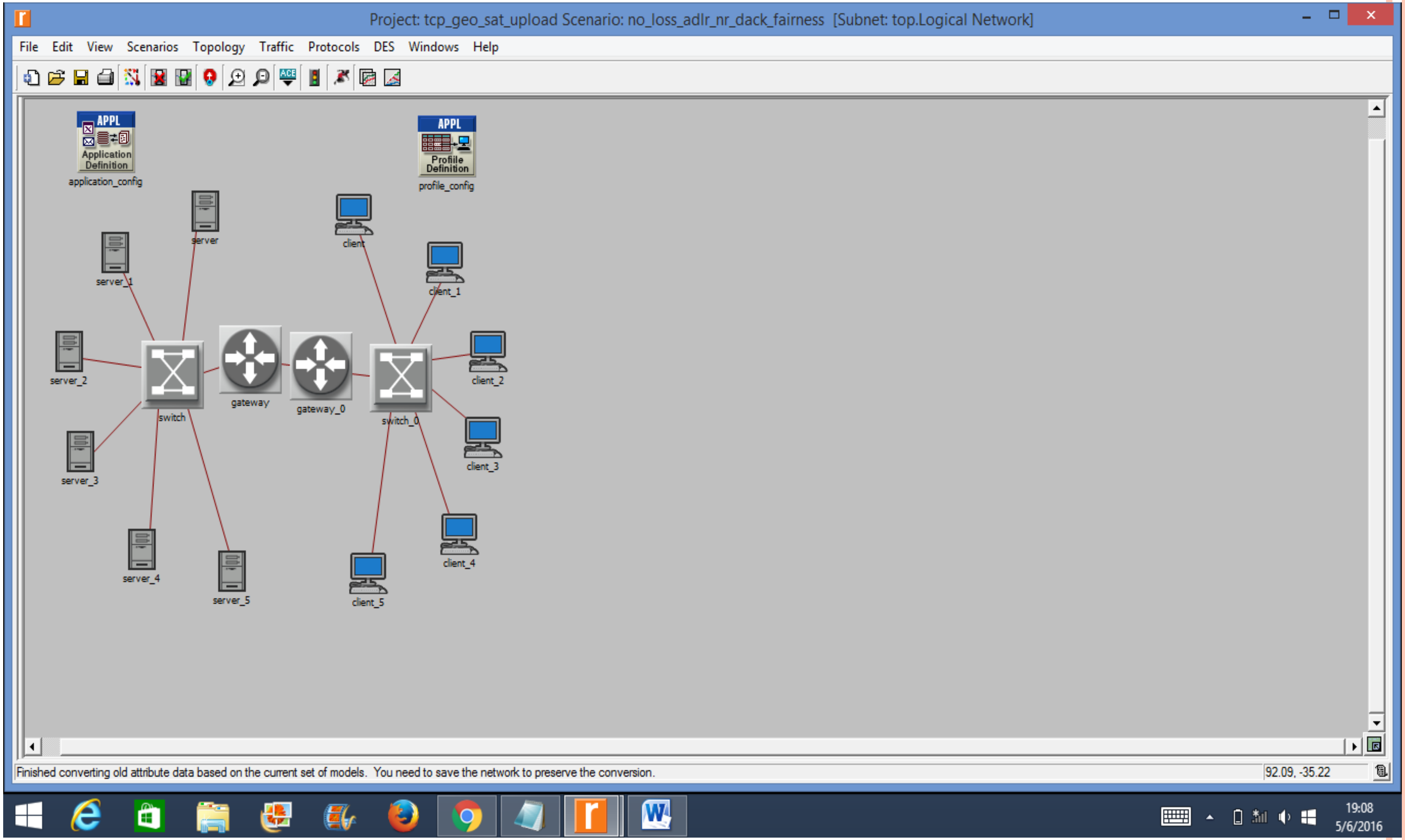


```
/* Check the assignment for "segment send threshold". Some simulators may operate */
/* on a packet (i.e., MSS) boundary. This attribute will help in TCP model        */
/* performance comparison for those cases.                                        */
if (op_ima_obj_attr_get (tcp_parameter_objid, "Segment Send Threshold",
    &tcp_parameter_ptr->seg_snd_thresh) == OPC_COMPCODE_FAILURE)
    tcp_mgr_error ("Unable to get \"Segment Send Threshold\" attribute.");

/* Determine if ECN capability is enabled. Refer to RFC-3168 for details on ECN */
if (op_ima_obj_attr_get (tcp_parameter_objid, "ECN Capability",
    &tcp_parameter_ptr->ecn_capability) == OPC_COMPCODE_FAILURE)
    tcp_mgr_error ("Unable to get \"ECN Capability\" attribute.");

/* Read the value for Initial Sequence number.  */
if (op_ima_obj_attr_get (tcp_parameter_objid, "Initial Sequence Number",
    &tcp_parameter_ptr->init_seq_num) == OPC_COMPCODE_FAILURE)
    tcp_mgr_error ("Unable to get \"Initial Sequence Number\" attribute.");

/* Determine the maximum allowable number of concurrent TCP connections.        */
if (op_ima_obj_attr_get (tcp_parameter_objid, "Active Connection Threshold",
    &max_connections) == OPC_COMPCODE_FAILURE)
    tcp_mgr_error ("Unable to get \"Active Connection Threshold attribute\".");
if (max_connections == -1)
    max_connections = OPC_INT_INFINITY;
```

SCREENSHOTS OF SCENARIO

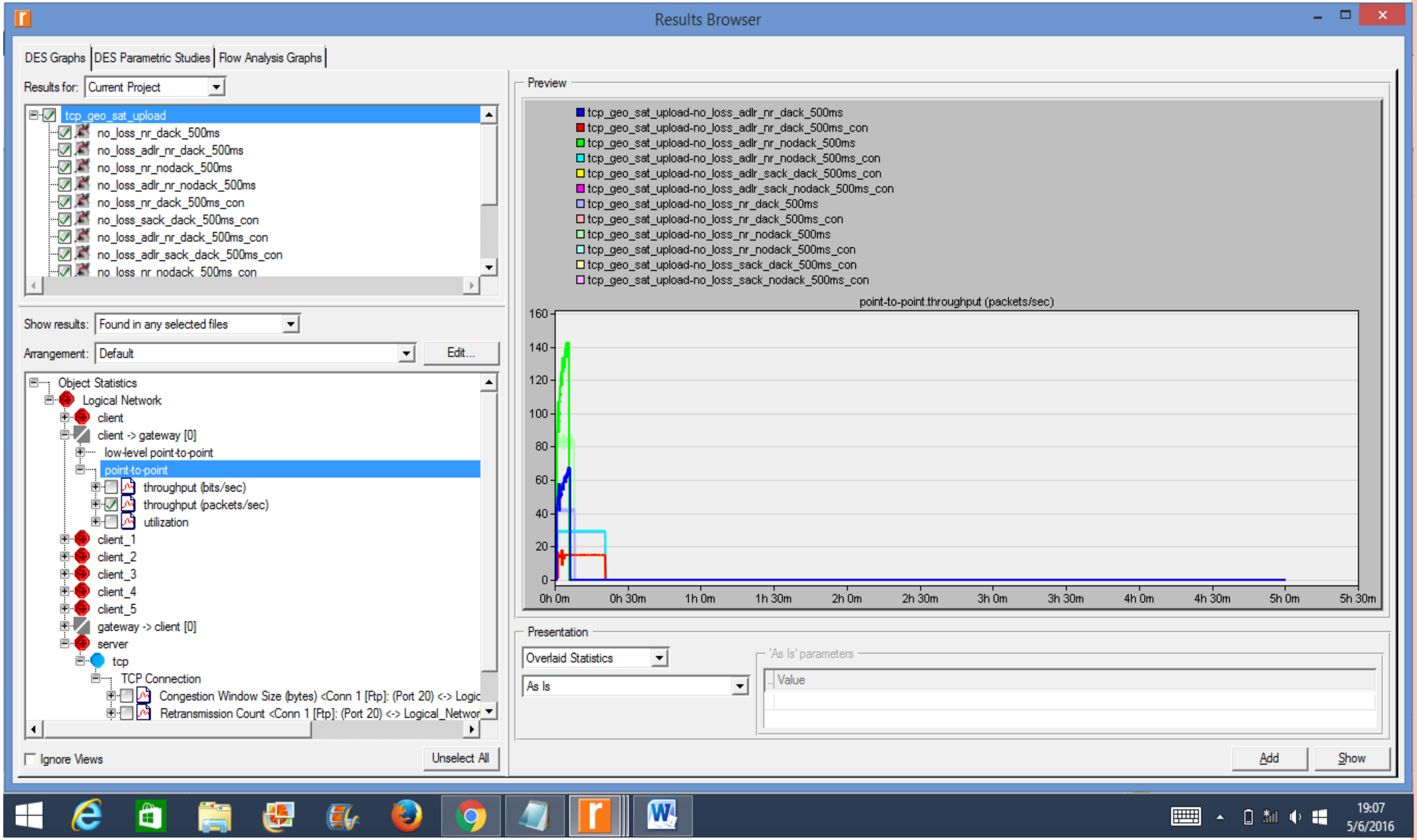


SCREENSHOTS OF SCENARIO



Finished converting old attribute data based on the current set of models. You need to save the network to preserve the conversion.

SCREENSHOTS OF OUTPUT



Results for: Current Project

- ☑ tcp_geo_sat_upload
  - ☑ no_loss_nr_dack_500ms
  - ☑ no_loss_adlr_nr_dack_500ms
  - ☑ no_loss_nr_nodack_500ms
  - ☑ no_loss_adlr_nr_nodack_500ms
  - ☑ no_loss_nr_dack_500ms_con
  - ☑ no_loss_sack_dack_500ms_con
  - ☑ no_loss_adlr_nr_dack_500ms_con
  - ☑ no_loss_adlr_sack_dack_500ms_con
  - ☑ no_loss_nr_nodack_500ms_con

Show results: Found in any selected files

Arrangement: Default

- Object Statistics
  - Logical Network
    - client
    - client -> gateway [0]
      - low-level point-to-point
      - point-to-point
        - □ throughput (bits/sec)
        - ☑ throughput (packets/sec)
        - □ utilization
    - client_1
    - client_2
    - client_3
    - client_4
    - client_5
    - gateway -> client [0]
    - server
      - tcp
        - TCP Connection
          - □ Congestion Window Size (bytes) <Conn 1 [Ftp]: (Port 20) <-> Logic
          - □ Retransmission Count <Conn 1 [Ftp]: (Port 20) <-> Logical_Networ

□ Ignore Views

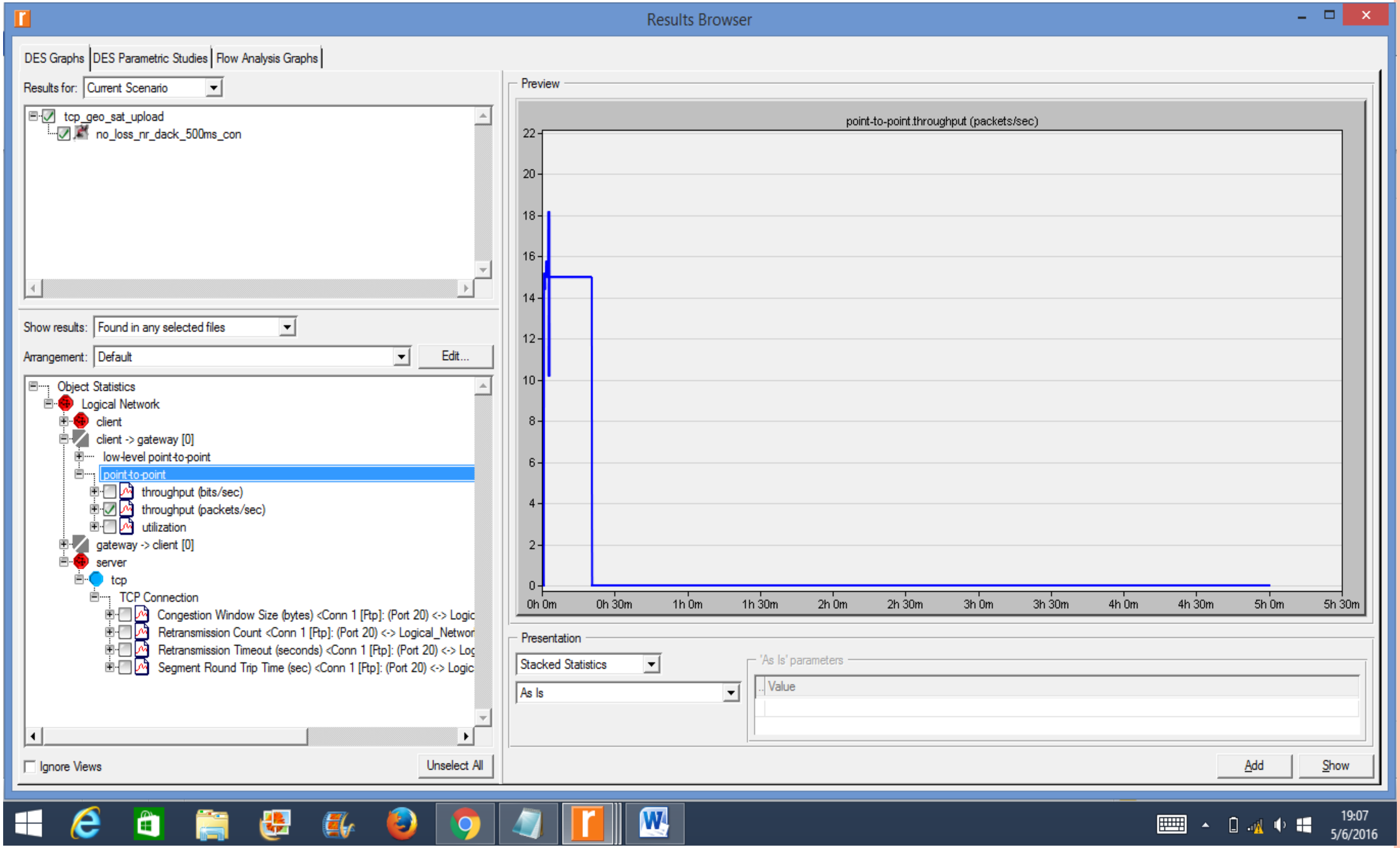
- 🟦 tcp_geo_sat_upload-no_loss_adlr_nr_dack_500ms
- 🟥 tcp_geo_sat_upload-no_loss_adlr_nr_dack_500ms_con
- 🟩 tcp_geo_sat_upload-no_loss_adlr_nr_nodack_500ms
- ■ tcp_geo_sat_upload-no_loss_adlr_nr_nodack_500ms_con
- 🟨 tcp_geo_sat_upload-no_loss_adlr_sack_dack_500ms_con
- 🟪 tcp_geo_sat_upload-no_loss_adlr_sack_nodack_500ms_con
- ■ tcp_geo_sat_upload-no_loss_nr_dack_500ms
- ■ tcp_geo_sat_upload-no_loss_nr_dack_500ms_con
- ■ tcp_geo_sat_upload-no_loss_nr_nodack_500ms
- ■ tcp_geo_sat_upload-no_loss_nr_nodack_500ms_con
- ■ tcp_geo_sat_upload-no_loss_sack_dack_500ms_con
- ■ tcp_geo_sat_upload-no_loss_sack_nodack_500ms_con

Overlaid Statistics

As Is

'As Is' parameters: Value

SCREENSHOTS OF OUTPUT



Results Browser

DES Graphs | DES Parametric Studies | Flow Analysis Graphs

Results for: Current Scenario

- ⊟ ☑ tcp_geo_sat_upload
  - ☑ no_loss_nr_dack_500ms_con

Show results: Found in any selected files

Arrangement: Default

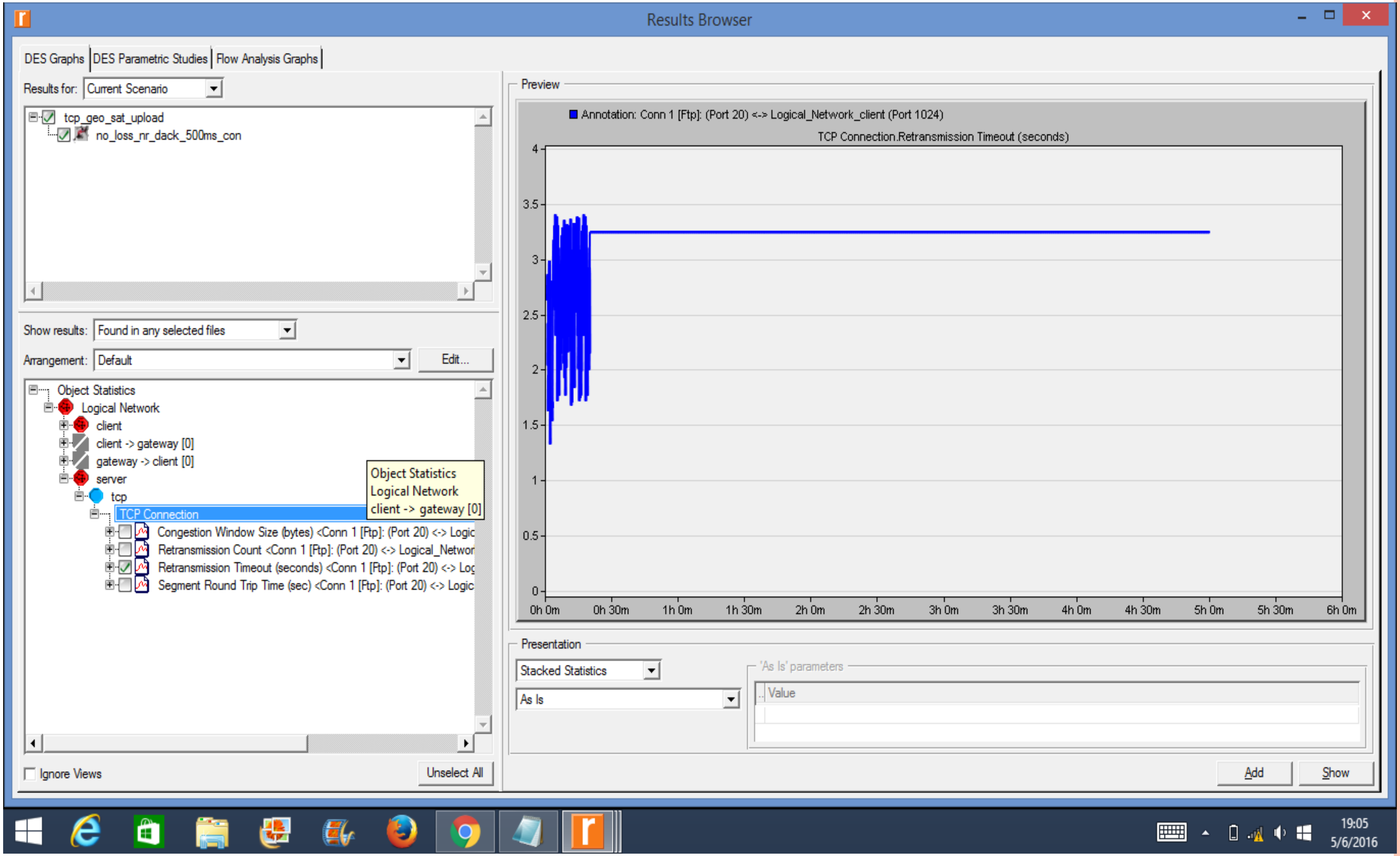
- ⊟ Object Statistics
  - ⊟ Logical Network
    - ⊞ client
    - ⊟ client -> gateway [0]
      - ⊞ low-level point-to-point
      - ⊟ point-to-point
        - ⊞ □ throughput (bits/sec)
        - ⊞ ☑ throughput (packets/sec)
        - ⊞ □ utilization
    - ⊞ gateway -> client [0]
    - ⊟ server
      - ⊟ tcp
        - ⊟ TCP Connection
          - ⊞ □ Congestion Window Size (bytes) <Conn 1 [Ftp]: (Port 20) <-> Logic
          - ⊞ □ Retransmission Count <Conn 1 [Ftp]: (Port 20) <-> Logical_Networ
          - ⊞ □ Retransmission Timeout (seconds) <Conn 1 [Ftp]: (Port 20) <-> Log
          - ⊞ □ Segment Round Trip Time (sec) <Conn 1 [Ftp]: (Port 20) <-> Logic

□ Ignore Views

Presentation: Stacked Statistics; As Is

'As Is' parameters: Value

SCREENSHOTS OF OUTPUT



Results for: Current Scenario

☑ tcp_geo_sat_upload
☑ no_loss_nr_dack_500ms_con

Show results: Found in any selected files

Arrangement: Default

Object Statistics
Logical Network
client
client -> gateway [0]
gateway -> client [0]
server
tcp
TCP Connection
☐ Congestion Window Size (bytes) <Conn 1 [Ftp]: (Port 20) <-> Logic
☐ Retransmission Count <Conn 1 [Ftp]: (Port 20) <-> Logical_Networ
☑ Retransmission Timeout (seconds) <Conn 1 [Ftp]: (Port 20) <-> Log
☐ Segment Round Trip Time (sec) <Conn 1 [Ftp]: (Port 20) <-> Logic

Object Statistics
Logical Network
client -> gateway [0]

☐ Ignore Views

■ Annotation: Conn 1 [Ftp]: (Port 20) <-> Logical_Network_client (Port 1024)

TCP Connection.Retransmission Timeout (seconds)

Stacked Statistics

As Is

REFERENCES

- A Survey of TCP over Mobile ADHOC Networks by Harpreet Singh Chawla, M. I. H. Ansari, Ashish Kumar, Prashant Singh Yadav
- Analysis of TCP Performance over Mobile Ad Hoc Networks by Gavin Holland
- New adaptation method of TCP for mobile ad hocnetworks by Yassine Douga and Malika Bourenane