

Congestion Control AODV (CC-AODV)

Software Recommended: NetSim Standard v13.0 (64 bit), Visual Studio 2019

Project Download Link:

https://github.com/NetSim-TETCOS/CC_AODV_Project_v13.0/archive/refs/heads/main.zip

Reference: Y. Mai, F. M. Rodriguez and N. Wang, "CC-ADOV: An effective multiple paths congestion control AODV," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, 2018, pp. 1000-1004.

Introduction

Ad hoc On-Demand Distance Vector (AODV) routing is one of the famous routing algorithms. Tremendous amounts of research on this protocol have been done to improve the performance. In this paper, a new control scheme, named congestion control AODV (CC-AODV), is proposed to manage the described routing condition. With this table entry, the package delivery rates are significantly increased while the package drop rate is decreased, however its implementation causes package overhead.

CC-ADOV aims to lower the performance degradation caused by the packets congestion while the data is delivered using AODV. Furthermore, CC-AODV determines a path for the data by using the congestion counter label. This is achieved by checking how stressed the current node is in a table, and once the RREP package is generated and transmitted through the nodes, the congestion counter adds one to the counter. The process of CC-AODV explains how to establish the route. First, the source node performs a flooding broadcast RREQ package in the entire network. When RREQ package arrives to the intermediate node, the router checks the congestion counter whether it is less than a certain predetermined value. If the comparison yields less than the counter, the routing table updates and forwarding to next router; otherwise, the router drops the RREQ package. Once the RREQ arrives to the corresponding destination, the RREP is generated by the router. In CC-AODV, the congestion flag is added to the RREP header. There are two cases of which a RREP is generated corresponding to a RREQ. One is from the source node to establish the route and the other is from the neighbour nodes to maintain the route. When the destination node receives the RREQ from the source node, it generates the RREP with the congestion flag set to true. While the RREP unicast back to the corresponding source node, passing by the intermediate node, the router checks the congestion flag. If it is true, the counter increases; otherwise, the counter keeps the same. Then, the router updates the routing information.

Procedure to implement CC-AODV in NetSim:

In order to implement CC-AODV following code modification done in AODV Protocol

1. The RREP structure `stru_NetSim_AODV_RREP` is defined in `AODV.h` has been modified to include a Congestion flag for implementing CC-AODV

```

176
177
178
179 struct stru_NetSim_AODV_RREP
180 {
181     unsigned int Type;8;
182     char RA[3]; /**<
183
184         R          Repair flag; used for multicast.
185
186         A          Acknowledgment required; see sections 5.4 and 6.7.
187
188     */
189     unsigned int Reserved;9; /**< Sent as 0; ignored on reception.
190     unsigned int PrefixSz;5; /**<
191         If nonzero, the 5-bit Prefix Size specifies that the
192         indicated next hop may be used for any nodes with
193         the same routing prefix (as defined by the Prefix
194         Size) as the requested destination.
195     */
196     unsigned int HopCount;8; /**<
197         The number of hops from the Originator IP Address
198         to the Destination IP Address. For multicast route
199         requests this indicates the number of hops to the
200         multicast tree member sending the RREP.
201     */
202     NETSIM_IPAddress DestinationIPAddress; /**< The IP address of the destination for which a route is supplied.
203     unsigned int DestinationSequenceNumber; /**< The destination sequence number associated to the route.
204     NETSIM_IPAddress OriginatorIPAddress; /**< The IP address of the node which originated the RREQ for which the route is supplied.
205     unsigned int Lifetime; /**< The time in milliseconds for which nodes receiving the RREP consider the route to be valid.
206     NETSIM_IPAddress LastAddress; //NetSim-specific
207     bool congestionflag:true;
208 };
209
210 /**
211
212

```

2. The DeviceVariable Structure stru_AODV_DeviceVariable is defined in AODV.h file has been modified to include a congestion counter for implementing CC-AODV

```

374
375
376 This is the AODV DeviceVariable Structure which contains -
377 FIFO - a packet is added in FIFO buffer if the device does not have route to the target<br>
378 routeTable - this contains the next HOP ip of the routes to the target<br>
379 RREQ_SEEN_TABLE - this contains list differnet RREQ a device encounters.
380
381 struct stru_AODV_DeviceVariable
382 {
383     unsigned int nSequenceNumber;
384     AODV_FIFO* fifo;
385     AODV_ROUTETABLE* routeTable;
386     AODV_RREQ_SEEN_TABLE* rreqSeenTable;
387     AODV_RREQ_SENT_TABLE* rreqSentTable;
388     AODV_PRECURSORS_LIST* precursorsList;
389     double dlastBroadcastTime;
390     unsigned int nRerrCount;
391     double dFirstRerrTime;
392     AODV_METRICS aodvMetrics;
393     unsigned int ncounter;
394 };
395

```

3. The source codes of functions in RREP.c, RouteTable.c and AODV_RouteError.c has been modified suitably to Increment, Decrement the congestion counter accordingly

```

RREP.c  AODV.c  AODV_CheckRouteFound.c  AODV_RouteError.c  RouteTable.c  RREQ.c  AODV.h
AODV
61  Deletes the RREP entry from sent table and forwards the rrep if the device is not
62  the source node.
63  */
64  int fn_NetSim_AODV_ProcessRREP(Netsim_EVENTDETAILS* pstruEventDetails)
65  {
66      AODV_ROUTETABLE* table = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->routeTable;
67      AODV_RREP* rrep = (AODV_RREP*)pstruEventDetails->pPacket->pstruNetworkData->Packet_RoutingProtocol;
68      //Update the routing table
69      if (rrep->DestinationIPaddress == aodv_get_curr_ip())
70          return 0;
71
72      if(rrep->congestionflag == true)
73          AODV_DEV_VAR(pstruEventDetails->nDeviceId)->ncounter++;
74
75      AODV_INSERT_ROUTE_TABLE(rrep->DestinationIPaddress,
76                             rrep->DestinationSequenceNumber,
77                             rrep->HopCount,
78                             rrep->LastAddress,
79                             pstruEventDetails->dEventTime+AODV_ACTIVE_ROUTE_TIMEOUT);
80      //Transmit the buffer
81      AODV_TRANSMIT_FIFO(AODV_DEV_VAR(pstruEventDetails->nDeviceId));
82      //Update the precursor list
83      AODV_INSERT_PRECURSOR(rrep->LastAddress);
84      AODV_UPDATE_ROUTE_TABLE(rrep->LastAddress,rrep->Lifetime);
85      if(!IP_COMPARE(aodv_get_curr_ip(),rrep->OriginatorIPaddress))
86      {
87          //Delete entry from sent table
88          AODV_RREQ_SENT_TABLE* table = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->rreqSentTable;
89          while(table)
90          {
91              if(!IP_COMPARE(table->DestAddress,rrep->DestinationIPaddress))
92              {
93                  IP_FREE(table->DestAddress);
94                  LIST_FREE((void*)&AODV_DEV_VAR(pstruEventDetails->nDeviceId)->rreqSentTable,table);
95                  break;
96              }
97              table = (AODV_RREQ_SENT_TABLE*)LIST_NEXT(table);
98          }
99      }

```

```

RREP.c  AODV.c  AODV_CheckRouteFound.c  AODV_RouteError.c  RouteTable.c  RREQ.c  AODV.h
AODV
163  /**
164  This function adds the timeout event of a Route Table which is equal to the table_LifeTime
165  */
166  int fn_NetSim_AODV_ActiveRouteTimeout(Netsim_EVENTDETAILS* pstruEventDetails)
167  {
168      int flag = 0;
169      NETSIM_IPAddress dest = (NETSIM_IPAddress)pstruEventDetails->szOtherDetails;
170      AODV_ROUTETABLE* table = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->routeTable;
171      while(table)
172      {
173          if(!IP_COMPARE(table->DestinationIPaddress,dest))
174          {
175              if(table->Lifetime <= pstruEventDetails->dEventTime)
176              {
177                  AODV_ROUTETABLE* temp = LIST_NEXT(table);
178                  IP_FREE(table->DestinationIPaddress);
179                  IP_FREE(table->NextHop);
180                  LIST_FREE(&AODV_DEV_VAR(pstruEventDetails->nDeviceId)->routeTable,table);
181                  AODV_DEV_VAR(pstruEventDetails->nDeviceId)->ncounter--;
182                  table = temp;
183                  continue;
184              }
185              else
186              {
187                  //Add new time out event
188                  pstruEventDetails->dEventTime = table->Lifetime;
189                  fnpAddEvent(pstruEventDetails);
190                  flag = 1;
191              }
192              table=(AODV_ROUTETABLE*)LIST_NEXT(table);
193          }
194      }
195      if(!flag)
196          IP_FREE(dest);
197      return 1;
198  }
199

```

```

RREP.c  AODV.c  AODV_CheckRouteFound.c  AODV_RouteError.c  RouteTable.c  RREQ.c  AODV.h
AODV
14 #include "main.h"
15 #include "AODV.h"
16 #include "List.h"
17 /**
18  * This function Generates a route error and sends it to the previous HOP.
19  */
20 int fn_NetSim_AODV_GenerateRERR(NETSIM_ID nDeviceId,
21     NETSIM_IPAddress UnreachableIP,
22     NetSim_EVENTDETAILS* pstruEventDetails)
23 {
24     AODV_DEV_VAR(nDeviceId)->ncounter--;
25
26     int DestCount=0;
27     NETSIM_IPAddress* DestinationList=NULL;
28     unsigned int* DestinationSequence=NULL;
29     AODV_DEVICE_VAR* pstruDeviceVar = AODV_DEV_VAR(nDeviceId);
30
31     AODV_ROUTE_TABLE* routeTable = pstruDeviceVar->routeTable;
32     AODV_PRECURSORS_LIST* precursorList = pstruDeviceVar->precursorList;
33     while(routeTable)
34     {
35         if(!IP_COMPARE(routeTable->NextHop,UnreachableIP))
36         {
37             routeTable->routingFlags = AODV_RoutingFlag_Invalid;
38             routeTable->Lifetime = pstruEventDetails->dEventTime+AODV_DELETE_PERIOD;
39             DestCount++;
40             DestinationList = realloc(DestinationList, DestCount*(sizeof* DestinationList));
41             DestinationSequence = realloc(DestinationSequence, DestCount*(sizeof* DestinationSequence));
42             DestinationList[DestCount-1] = IP_COPY(routeTable->DestinationIPAddress);
43             DestinationSequence[DestCount-1] = routeTable->DestinationSequenceNumber;
44         }
45         routeTable = LIST_NEXT(routeTable);
46     }
47     if(precursorList->count)
48     {
49         int loop;
50         bool flag=false;
51         for(loop=0; loop<precursorList->count; loop++)

```

4. The source codes and functions related to Route request are defined in the file **RREQ.c**. The **fn_NetSim_AODV_ProcessRREQ()** function that is part of this file has been modified suitably to check the value of the congestion counter in the received RREQ packet and accordingly forward or drop the packet

```

RREP.c  AODV.c  AODV_RouteError.c  RouteTable.c  RREQ.c  AODV.h
Miscellaneous Files
317 //Free the rreq packet
318 fn_NetSim_Packet_FreePacket(packet);
319 pstruEventDetails->pPacket=NULL;
320 }
321 else
322 {
323     int dev_counter = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->ncounter;
324     if (dev_counter > 25)
325     {
326         fn_NetSim_Packet_FreePacket(packet);
327         pstruEventDetails->pPacket = NULL;
328         return 1;
329     }
330
331     if(AODV_CHECK_ROUTE_FOUND(rreq->DestinationIPAddress) &&
332         rreq->JRGDU[3] != '1' /* Destination only flag*/)
333     {
334         if(AODV_GENERATE_RREP_BY_IN())
335         {
336             fn_NetSim_Packet_FreePacket(packet);
337             pstruEventDetails->pPacket=NULL;
338         }
339     }
340

```

Steps:

1. After you unzip the downloaded project folder, Open NetSim Home Page click on **Your Work**

NetSim Standard

Network Simulation/Emulation Platform
Version 13.0.21 (64 Bit)

New Simulation Ctrl+N

Your work Ctrl+O

Examples Ctrl+E

License Settings

Exit Alt+F4

Current workspace: CC_AODV_Workspace

Choose a Network

C1 Internetworks (Base)

Features TCP, IP, Routing, Wi-Fi, Ethernet, RF, Propagation, Application Models, Network Stack, Simulation Kernel, Animator and Plot Engine

C2 Legacy Networks

Features Pure Aloha, Slotted Aloha, GSM and CDMA. Legacy Networks run standalone and do not interface with Internetworks library

C4 Mobile Adhoc Networks

Features DSR, AODV, OLSR and ZRP routing protocols, and multiple MANETs with bridge nodes. MANETs interface with Internetworks library

C5 Wireless Sensor Networks

Based on 802.15.4 MAC/PHY interfaces with MANETs library

C6 Internet of Things

Features WSN, RPL and 6LoWPAN with MANET routing. IoT interfaces with Internetworks and MANETs libraries

C7 Cognitive Radio Networks

Based on 802.22 standard and interfaces with Internetworks library

C8 LTE / LTE-A

Based on 3GPP 36 series standards. LTE/LTE-A interfaces with Internetworks library

C9 Vehicular Adhoc Networks

Based on IEEE1609 Wave and IEEE802.11p with MANET routing. VANETs interface with Internetworks library and SUMO

C10 5G NR

Based on 3GPP 38 series standards. 5G NR interfaces with Internetworks library

C11 Satellite Comm. Networks

Support

[Answers/FAQ](#)
[Contact Technical Support](#)
 Email - support@tetcos.com

Learn

[Videos](#)
[Experiment Manual](#)

Documentation

[User Manual](#)
[Technology Libraries](#)
[Source Code Help](#)

Contact Us

 Email - sales@tetcos.com
 Phone - +91 767 605 4321
2. Click on **Workspace options**

NetSim Standard

Network Simulation/Emulation Platform
Version 13.0.21 (64 Bit)

New Simulation Ctrl+N

Your work Ctrl+O

Examples Ctrl+E

License Settings

Exit Alt+F4

Current workspace: CC_AODV_Workspace

Experiment name	Date modified	Network type	Size
10Nodes	01-03-2021	Mobile_Adhoc_Networks	5,197 KB
30Nodes	27-02-2021	Mobile_Adhoc_Networks	6,877 KB

[View Results](#)[Export](#)[View Results](#)[Export](#)

Workspace options...

[Import Experiment](#)

Support

[Answers/FAQ](#)
[Contact Technical Support](#)
 Email - support@tetcos.com

Learn

[Videos](#)
[Experiment Manual](#)

Documentation

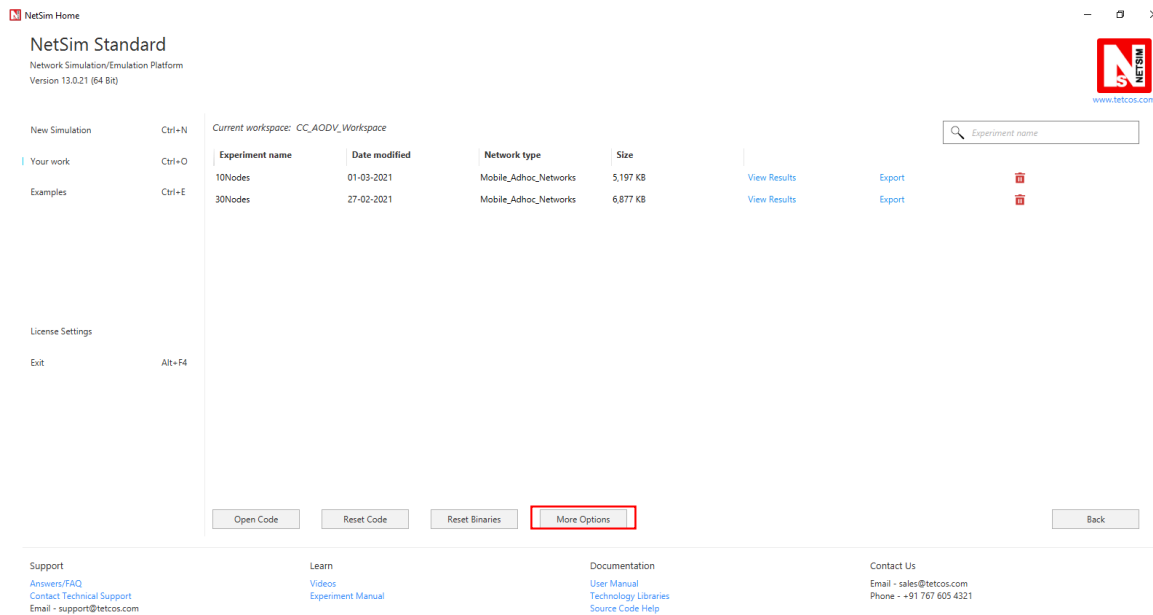
[User Manual](#)
[Technology Libraries](#)
[Source Code Help](#)

Contact Us

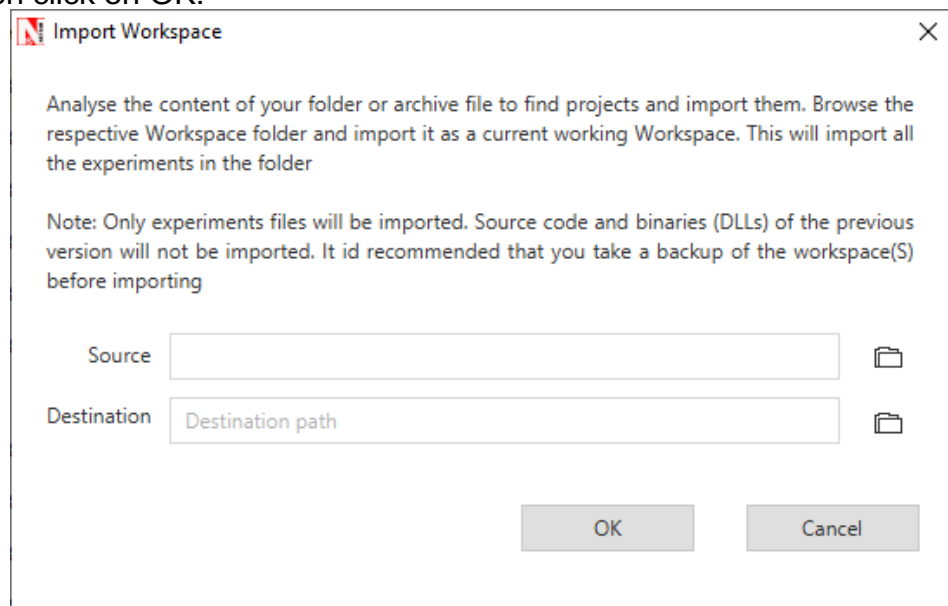
 Email - sales@tetcos.com
 Phone - +91 767 605 4321

NetSim Home

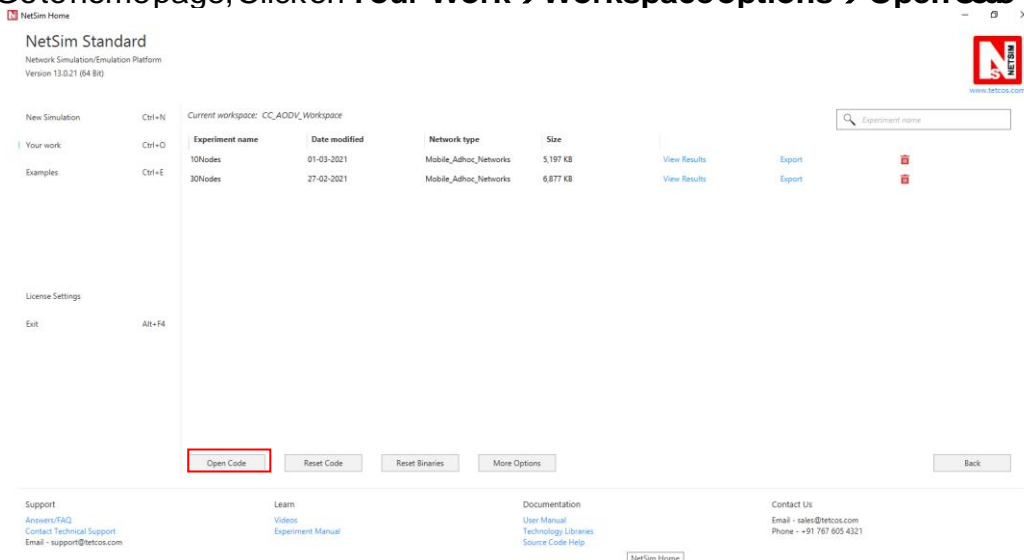
3. Click on **More Options**,



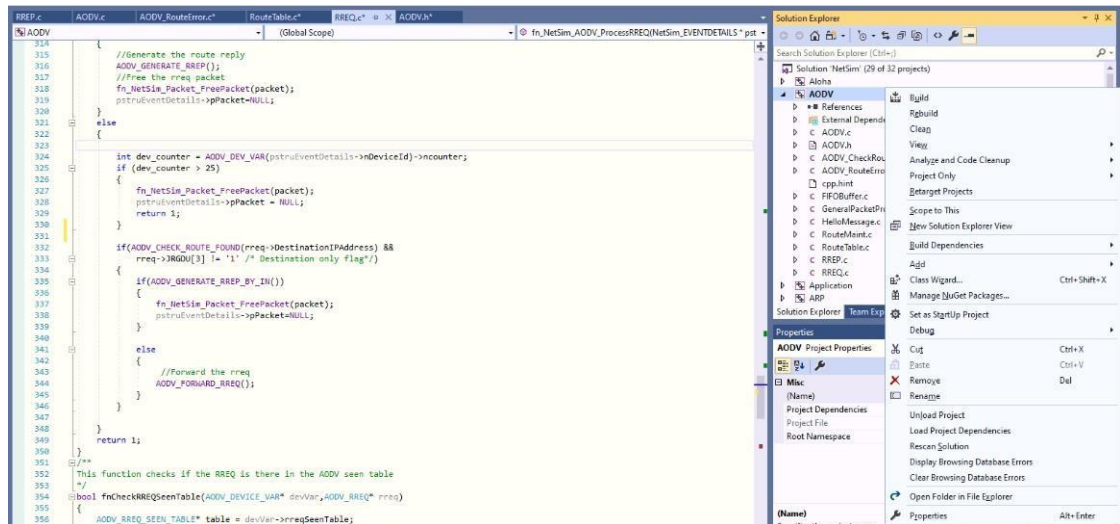
- Click on Import, browse the extracted folder and go into the `Workspace_Performance_Analysis_CC_AODV.netsim_wsp`, click on select folder and then click on OK.



- Goto homepage, Click on **Your Work** → **Workspace options** → **Open Code**



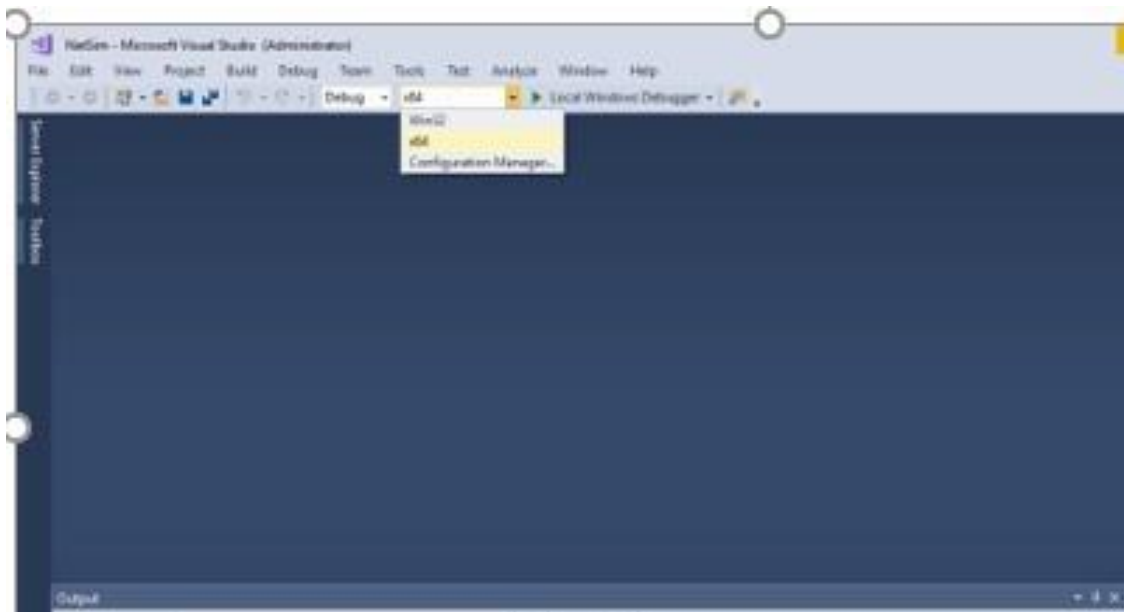
- Right click on the AODV Project and select rebuild.



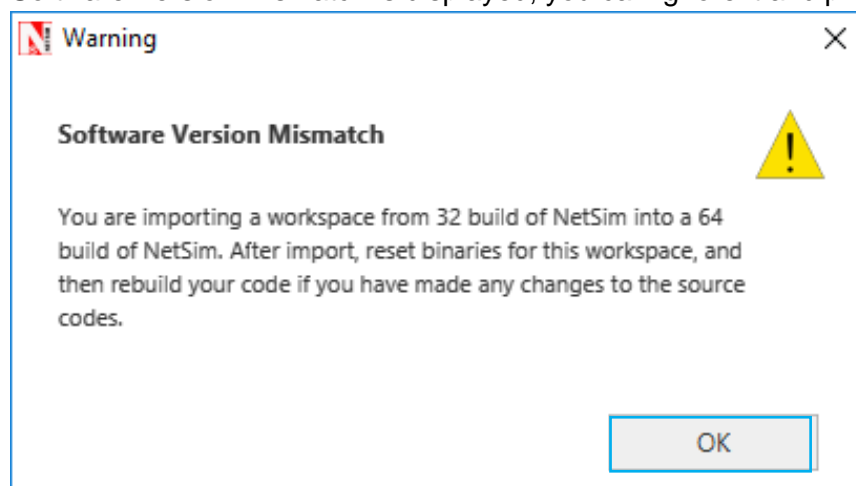
- Upon rebuilding, **libAodv.dll** will automatically get updated in the respective bin folder of the current workspace.

Note:

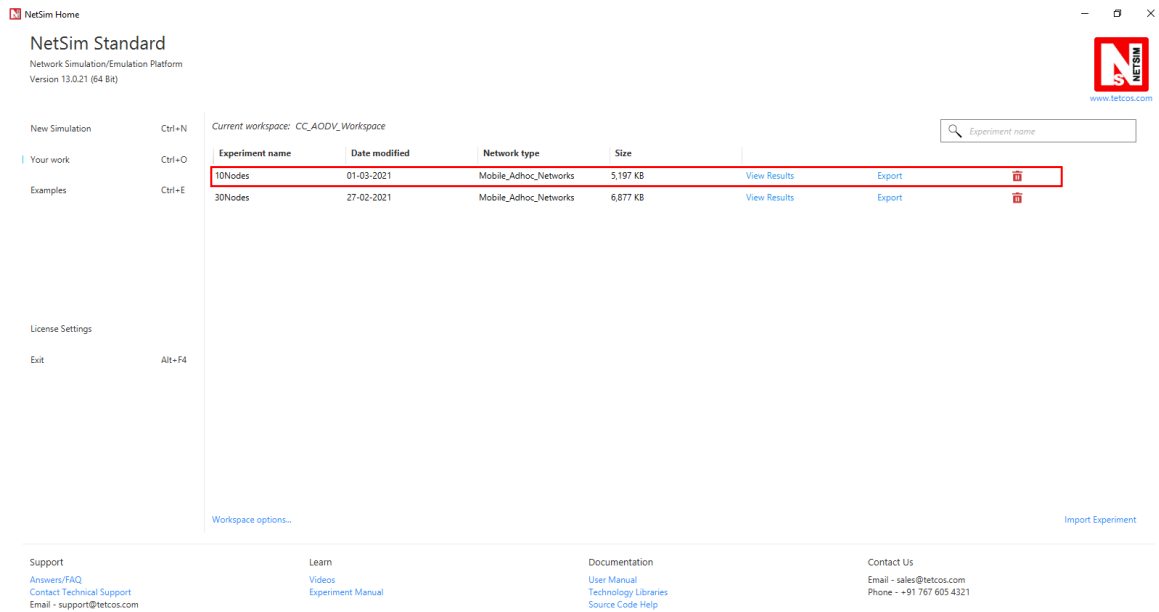
- Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit DLL files respectively as shown below:



- While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.

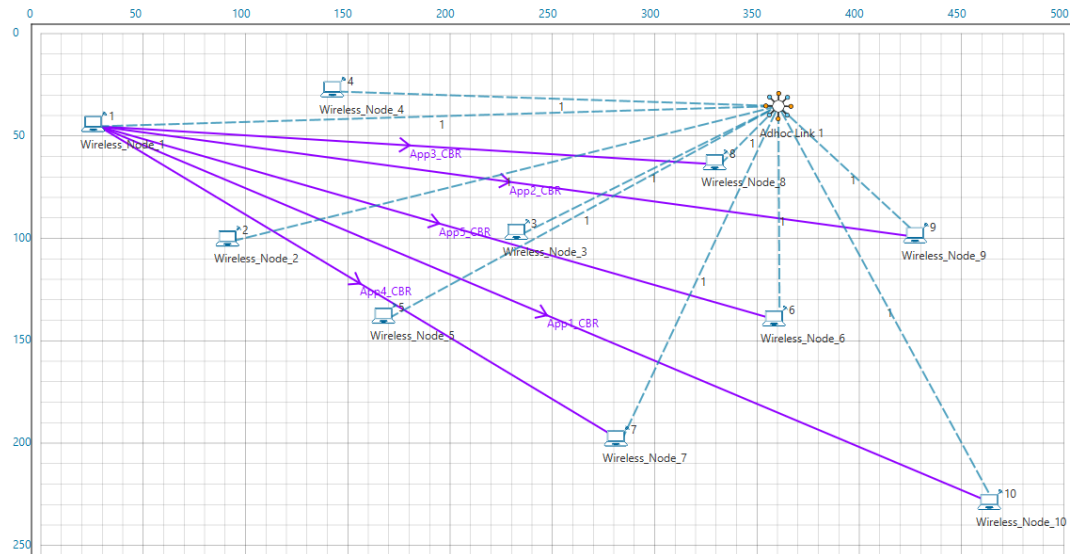


8. Go to NetSim home page, click on **Your work**, Click on **10Nodes_Example**.



- Now create a Network Scenario in NetSim Mobile Adhoc Network as per the screenshot below.

A sample Configuration.netsim file is also provided in the Config_File folder along with this project which can be opened in NetSim directly.



- Run the simulation for 30 sec

Simulations have been carried out using a different number of nodes in a network to symbolize different practical applications of wireless network. For example, 10 nodes symbolize a small network that can be used in an agricultural setup. 30 nodes symbolize a medium size network that can be used in an industrial setup.

Result:

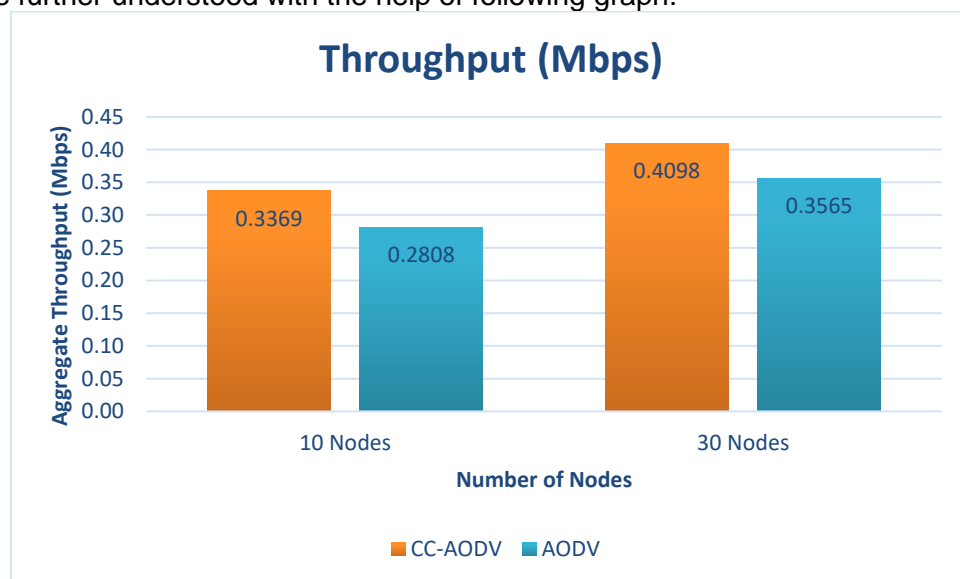
Performance of CC-AODV has been compared with other reactive protocol AODV based on different performance metrics such as Throughput, End to End delay etc.

Number of Nodes	AODV Aggregate Throughput (Mbps)	CC_AODV Aggregate Throughput (Mbps)
10Nodes	0.2808	0.3369
30Nodes	0.3565	0.4098

Table 1: Aggregate Throughput comparison between AODV and CC_AODV

As per the Table 1 the proposed CC-AODV has higher throughput than the AODV. In CC-AODV, the internal nodes can be utilized much efficiently than AODV because the counter helps to reroute the path if the internal node is busy. This can increase the network channel utilization.

This can be further understood with the help of following graph:



Number of Nodes	AODV Average Delay (microseconds)	CC_AODV Average Delay (microseconds)
10Nodes	5462760.299	2004123.199
30Nodes	6534879.477	293415.948

Table 2: End to End delay comparison between AODV and CC_AODV

Table 2 demonstrate that AODV has higher End-to-End performance than the CC-AODV, the result is achieved by rerouting the path of the data if the router is on a busy state.

This can be further understood with the help of following graph:

