

Congestion Control AODV (CC-AODV)

Software: 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

Follow the instructions specified in the following link to download and setup the Project in NetSim:

<https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects>

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.

Steps to simulate

1. Open the Source codes in Visual Studio by going to Your work-> Workspace Options and Clicking on Open code button in NetSim Home Screen window.
2. In Visual Studio, Right click on the AODV Project and select rebuild.

```

315 //Generate the route reply
316 AODV_GENERATE_RESP();
317 //Free the req packet
318 fn_NetSim_Packet_FreePacket(packet);
319 }
320 }
321 }
322
323 int dev_counter = AODV_DEV_VAR(ptrtoEventDetails->DeviceId)->counter;
324 if (dev_counter > 25)
325 {
326     fn_NetSim_Packet_FreePacket(packet);
327     ptrtoEventDetails->reqpacket = NULL;
328     return 1;
329 }
330
331 if(AODV_HASH_ROUTE_ID00((req->DestinationIPAddress) &&
332 req->SRCID)) { // "1" if destination only flag?
333     if(AODV_GENERATE_RESP_BY_ID())
334     {
335         fn_NetSim_Packet_FreePacket(packet);
336         ptrtoEventDetails->reqpacket=NULL;
337     }
338     else
339     {
340         //Forward the req
341         AODV_FORWARD_REQ();
342     }
343 }
344 return 1;
345 }
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Figure 1: Screen shot of NetSim project source code in Visual Studio

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

Example

1. The CC_AODV_Workspace comes with a sample network configuration that are already saved. To open this example, go to Your work in the Home screen of NetSim and click on the 10Nodes from the list of experiments.

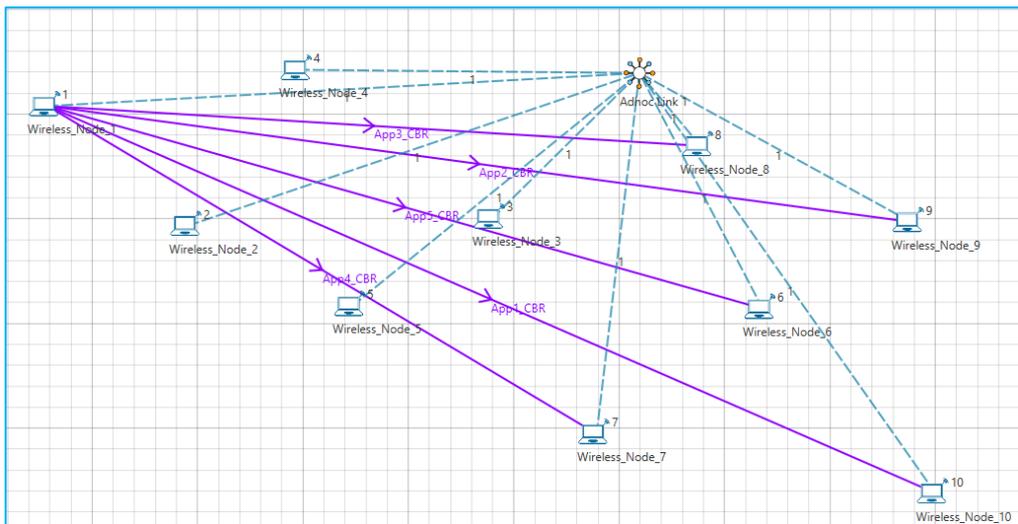


Figure 2: Network topology in this project

2. 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.

Results and discussion

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.

Throughput Comparison of CC-AODV and AODV

Number of Nodes	AODV Aggregate Throughput (Mbps)	CC_AODV Aggregate Throughput (Mbps)
10Nodes	0.28	0.33
30Nodes	0.35	0.40

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.

Graph

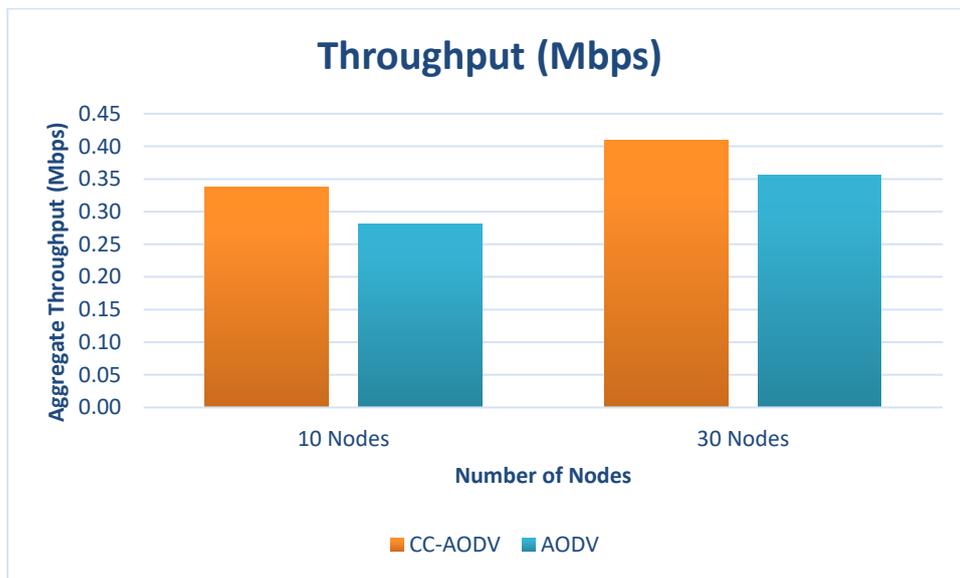


Figure 3: Plot comparison between of Aggregate Throughput AODV and CC_AODV

Delay Comparison of CC-AODV and AODV

Number of Nodes	AODV Average Delay (microseconds)	CC_AODV Average Delay (microseconds)
10Nodes	5462760.29	2004123.19
30Nodes	6534879.47	293415.94

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.

Graph

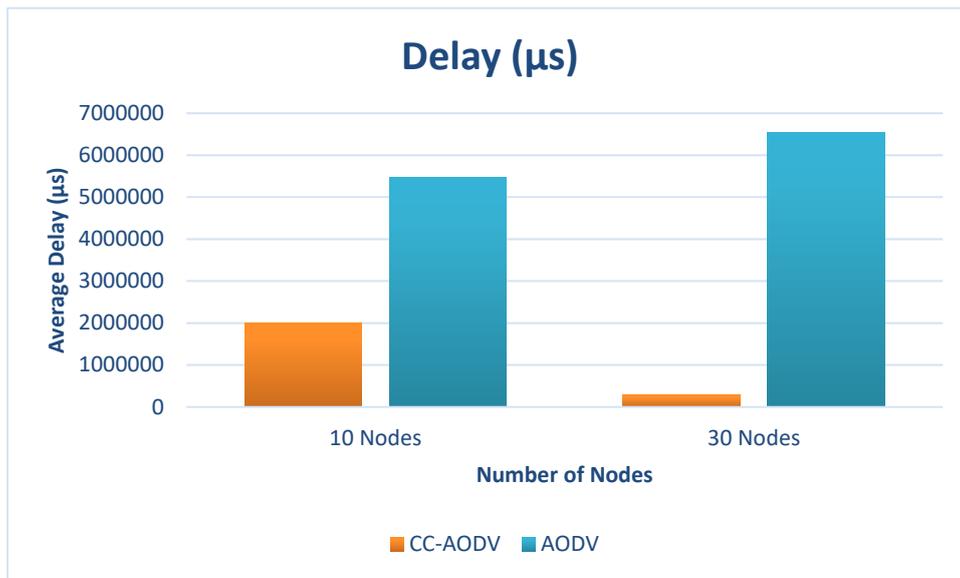


Figure 4: Plot comparison between of Aggregate Throughput AODV and CC_AODV

Appendix: NetSim source code modifications

Changes to RREP structure `stru_NetSim_AODV_RREP`, in `AODV.h` file, within AODV project

```
/* This is used to add a Congestion flag for implementing CC-AODV*/

struct stru_NetSim_AODV_RREP
{
  unsigned int Type:8;//2
  char RA[3]; /**<
  ~~~~~
  R      Repair flag; used for multicast.

  A      Acknowledgment required; see sections 5.4 and 6.7.
  ~~~~~
  */
  unsigned int Reserved:9; ///< Sent as 0; ignored on reception.
  unsigned int PrefixSz:5; /**<
  If nonzero, the 5-bit Prefix Size specifies that the
  indicated next hop may be used for any nodes with
  the same routing prefix (as defined by the Prefix
  Size) as the requested destination.
  */
  unsigned int HopCount:8; /**<
  The number of hops from the Originator IP Address
  to the Destination IP Address. For multicast route
  requests this indicates the number of hops to the
  multicast tree member sending the RREP.
  */
  NETSIM_IPAddress DestinationIPAddress;///< The IP address of the destination for which a route is
  supplied.
  unsigned int DestinationSequenceNumber;///< The destination sequence number associated to the route.
  NETSIM_IPAddress OriginatorIPAddress;///< The IP address of the node which originated the RREQ for
  which the route is supplied.
  unsigned int Lifetime;///< The time in milliseconds for which nodes receiving the RREP consider the route to
  be valid.
  NETSIM_IPAddress LastAddress; //NetSim-specific
  bool congestionflag : true;
};
```

Changes to RREP structure `stru_AODV_DeviceVariable`, in `AODV.h` file, within AODV project

```
/* This is used to add a Congestion congestion counter for implementing CC-AODV*/

struct stru_AODV_DeviceVariable
{
  unsigned int nSequenceNumber;
  AODV_FIFO* fifo;
  AODV_ROUTETABLE* routeTable;
  AODV_RREQ_SEEN_TABLE* rreqSeenTable;
  AODV_RREQ_SENT_TABLE* rreqSentTable;
  AODV_PRECURSORS_LIST* precursorsList;
  double dLastBroadcastTime;
  unsigned int nRerrCount;
  double dFirstRerrTime;
  AODV_METRICS aodvMetrics;
  unsigned int ncounter;

};
```

Changes to fn_NetSim_AODV_ProcessRREP(), in RREP.c file, within AODV project

/* This is used to modified suitably to Increment, Decrement the congestion counter accordingly */

```
int fn_NetSim_AODV_ProcessRREP(NetSim_EVENTDETAILS* pstruEventDetails)
{
    AODV_ROUTETABLE* table = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->routeTable;
    AODV_RREP* rrep = (AODV_RREP*)pstruEventDetails->pPacket->pstruNetworkData->Packet_RoutingProtocol;
    //Update the routing table
    if(rrep->DestinationIPAddress == aodv_get_curr_ip())
        return 0;
    if (rrep->congestionflag == true)
        AODV_DEV_VAR(pstruEventDetails->nDeviceId)->ncounter++;
    AODV_INSERT_ROUTE_TABLE(rrep->DestinationIPAddress,
        rrep->DestinationSequenceNumber,
        rrep->HopCount,
        rrep->LastAddress,
        pstruEventDetails->dEventTime+AODV_ACTIVE_ROUTE_TIMEOUT);
}
```

Changes to fn_NetSim_AODV_ActiveRouteTimeout(), in RouteTable.c file, within AODV project

/*This function adds the timeout event of a Route Table which is equal to the table_LifeTime*/

```
int fn_NetSim_AODV_ActiveRouteTimeout(NetSim_EVENTDETAILS* pstruEventDetails)
{
    int flag = 0;
    NETSIM_IPAddress dest = (NETSIM_IPAddress)pstruEventDetails->szOtherDetails;
    AODV_ROUTETABLE* table = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->routeTable;
    while(table)
    {
        if(!IP_COMPARE(table->DestinationIPAddress,dest))
        {
            if(table->Lifetime <= pstruEventDetails->dEventTime)
            {
                AODV_ROUTETABLE* temp = LIST_NEXT(table);
                IP_FREE(table->DestinationIPAddress);
                IP_FREE(table->NextHop);
                LIST_FREE(&AODV_DEV_VAR(pstruEventDetails->nDeviceId)->routeTable,table);
                AODV_DEV_VAR(pstruEventDetails->nDeviceId)->ncounter--;
                table = temp;
                continue;
            }
            else
            {
                //Add new time out event
                pstruEventDetails->dEventTime = table->Lifetime;
                pstruEventDetails->szOtherDetails = dest;
                fnpAddEvent(pstruEventDetails);
                flag = 1;
            }
        }
        table=(AODV_ROUTETABLE*)LIST_NEXT(table);
    }
    if(!flag)
        IP_FREE(dest);
    return 1;
}
```

```
}
```

Changes to `fn_NetSim_AODV_GenerateRERR()`, in `AODV_RouteError.c` file, within AODV project

```
/*This function Generates a route error and sends it to the previous HOP.*/
```

```
int fn_NetSim_AODV_GenerateRERR(NETSIM_ID nDeviceId,  
NETSIM_IPAddress UnreachableIP,  
NetSim_EVENTDETAILS* pstruEventDetails)  
{  
AODV_DEV_VAR(nDeviceId)->ncounter--;  
int DestCount=0;  
NETSIM_IPAddress* DestinationList=NULL;  
unsigned int* DestinationSequence=NULL;  
AODV_DEVICE_VAR* pstruDeviceVar = AODV_DEV_VAR(nDeviceId);  
AODV_ROUTETABLE* routeTable = pstruDeviceVar->routeTable;  
AODV_PRECURSORS_LIST* precursorList = pstruDeviceVar->precursorList;
```

Changes to `fn_NetSim_AODV_ProcessRREQ()`, in `RREQ.c` file, within AODV project

```
/* modified suitably to check the value of the congestion counter in the received RREQ packet and  
accordingly forward or drop the packet */
```

```
{  
    int dev_counter = AODV_DEV_VAR(pstruEventDetails->nDeviceId)->ncounter;  
    if (dev_counter > 25)  
    {  
        fn_NetSim_Packet_FreePacket(packet);  
        pstruEventDetails->pPacket = NULL;  
        return 1;  
    }  
    if (AODV_CHECK_ROUTE_FOUND(rreq->DestinationIPAddress) &&  
        rreq->JRGDU[3] != '1' /* Destination only flag*/)  
    {  
        if (AODV_GENERATE_RREP_BY_IN())  
        {  
            fn_NetSim_Packet_FreePacket(packet);  
            pstruEventDetails->pPacket = NULL;  
        }  
        else  
        {  
            //Forward the rreq  
            AODV_FORWARD_RREQ();  
        }  
    }  
}  
  
return 1;
```