IT200: CCN PROJECT

Harini Thirunavukkarasan - 191IT221
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:harinithiru.191it221@nitk.edu.in

Aadil Khalifa - 191IT101
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:aadil.191it101@nitk.edu.in

Gayathri Nisha - 191IT116
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:gayathrinisha.191it116@nitk.edu.in

Prasanthi Bolimera - 191IT240
Information Technology
National Institute of Technology Karnataka
Surathkal, India 575025
Email:prasanthibolimera.191it240@nitk.edu.in

***Abstract*—Device-to-device (D2D) communication is expected to play a significant role in upcoming cellular networks as it promises ultra-low latency for communication among users. This new mode may operate in a licensed or unlicensed spectrum. It is a novel addition to the traditional cellular communication paradigm. Its benefits are, however, accompanied by many technical and business issues that must be resolved before integrating it into the cellular ecosystem.The project consists of four D2D pairs and two Cellular Users and they are sharded using K-Nearest Neighbors algorithm which clusters the UEs on the basis of distance between them**

I. Introduction

Cellular network is now four generations old. Need for fast multimedia-rich data exchange along with high quality voice calls has been the primary motivation in this forward journey. As newer and more demanding applications arise and subscriber base increases exponentially, there is an urgent requirement for more novel techniques to boost data rates and reduce latency. D2D communication is a new paradigm in cellular networks. It allows user equipment (UEs) in close proximity to communicate using a direct link rather than having their radio signal travel all the way through the base station (BS) or the core network. One of its main benefits is the ultra-low latency in communication due to a shorter signal traversal path. Various short-range wireless technologies like Bluetooth, WiFi Direct and LTE Direct (defined by the Third Generation Partnership Project (3GPP))can be used to enable D2D communication.

The Device to Device (D2D) pairs are sharded/clustered based on distance between the UE's.For Sharding, we have used the K-NN (K-Nearest Neighbour) Algorithm. KNN Algorithm is a simple algorithm which stores all available cases and classifies new cases based on a similarity measure. Example, distance between the UEs in our case. The new data points will be assigned a value based on how closely it matches the points in the training set.

II. Problem Statement

The available D2D pairs should be sharded based on any suitable criteria using any clustering algorithm.Any number of D2D pairs is allowed in a shard with at least one Cellular User(CU).The D2D pairs should start communication at regular intervals.Calculate the SINR of each D2D pair accordingly.

A. Objectives

- To successfully create D2D pairs, Cellular Users and an eNB.
- To shard the D2D pairs based on distance using K-NN algorithm.
- To establish communication at regular intervals.
- To calculate the SINR of each D2D pair.

III. Methodology

One node for eNb and ten UE nodes are created out of which 2 nodes are Cellular Users.

```
//Create nodes (eNb + UEs)
NodeContainer enbNode;
enbNode.Create (1);
NS_LOG_INFO ("eNb node id = [" << enbNode.Get (0)->GetId () << "]");
NodeContainer ueNodes;
ueNodes.Create (10);
```

Position for each node is allocated, i.e. coordinates are assigned to the eNb node and the cellular nodes in three-dimensional coordinate system.

```
//Position of the nodes
Ptr<ListPositionAllocator> positionAllocEnb = CreateObject<ListPositionAllocator> ();
positionAllocEnb->Add (Vector (0.0, 0.0, 30.0));

Ptr<ListPositionAllocator> positionAllocUe1 = CreateObject<ListPositionAllocator> ();
positionAllocUe1->Add (Vector (10.0, 0.0, 0.0));
Ptr<ListPositionAllocator> positionAllocUe2 = CreateObject<ListPositionAllocator> ();
positionAllocUe2->Add (Vector (-10.0, 0.0, 0.0));
 Ptr<ListPositionAllocator> positionAllocUe3 = CreateObject<ListPositionAllocator> ();
```

Mobility of each node is restricted by making it's position constant.

```
//Install mobility
MobilityHelper mobilityeNodeB;
mobilityeNodeB.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobilityeNodeB.SetPositionAllocator (positionAllocEnb);
mobilityeNodeB.Install (enbNode);

MobilityHelper mobilityUe1;
mobilityUe1.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobilityUe1.SetPositionAllocator (positionAllocUe1);
mobilityUe1.Install (ueNodes.Get (0));
```

The number of cellular users in our case are 2,

```
uint32_t n_CellularUsers=2;
uint32_t n_d2dPairs=(ueNodes.GetN()-n_CellularUsers)/2;
std::cout<<"Number of Cellular users: "<<n_CellularUsers<<"\nNumber of D2D pairs: "<<n_d2dPairs<<"\n";
```

In the above picture, the number of D2D pairs are found by dividing the total number of UEs (Number of UE nodes - number of cellular users) by 2. A two-dimensional array is created where the coordinates of each UE is stored in it. Next, the distance vector for each pair of D2D users is found by taking the average of the distance vectors of both.

Now, we use the KNN (K- nearest neighbour) algorithm to do the sharding. The function distance() is used to calculate distance between two nodes. In this, we take ueNode 0 and 1 as cellular users.

```
int result[n_d2dPairs];
NS_LOG_INFO(result);
for (uint32_t i=0; i<n_d2dPairs; i++){
  int r=0;
NS_LOG_INFO(r);

double b[]={du[3*i+0],du[3*i+1],du[3*i+2]};
double dist=distance( CU_coordinates[0] , b );
double minimum=distance( CU_coordinates[0] , b );
for (uint32_t j=0;j<n_CellularUsers;j++){
  dist=distance( CU_coordinates[j] , b );
  //std::cout<<"Distance: "<<dist<<"\n";
  if(dist<minimum){
  minimum=dist;
  r=j;
  //std::cout<<"changed\n";
  }
}
result[i]=r;
//std::cout<<"\t\t"<<result[i]<<"\n";
}
```

In this algorithm, we take the distance between the cellular node 0 and the present d2d device and is assigned as the minimum value. Next, the distance between the same d2d device is calculated with both the cellular nodes and if the current distance is less than the initial distance assigned to minimum, the current distance is assigned to the minimum. The variable 'r' is assigned the value of 'j' which is basically the node which has the minimum distance from the current d2d device and this value of 'r' is assigned to the result (array). The same is repeated for all the d2d devices and result has the CU node number (whether it belongs to shard of CU node 0 or CU node 1) and are finally sharded.

In our case, we are using Class A of Ipv4 addresses and of Multicast.

The InetSocketAddress class implements an IP Socket Address (IP Address + Port number) to the Client (remote) and Server(local), each d2d pair is connected to the same port. Parameters of OnOffHelper(): protocol-the name of the

```
remote[0] = InetSocketAddress (groupAddress4, 8000);
local[0] = InetSocketAddress (Ipv4Address::GetAny (), 8000);

remote[1] = InetSocketAddress (groupAddress4, 8001);
local[1] = InetSocketAddress (Ipv4Address::GetAny (), 8001);

remote[2] = InetSocketAddress (groupAddress4, 8002);
local[2] = InetSocketAddress (Ipv4Address::GetAny (), 8002);

remote[3] = InetSocketAddress (groupAddress4, 8003);
local[3] = InetSocketAddress (Ipv4Address::GetAny (), 8003);

tft = Create<LteSlTft> (LteSlTft::BIDIRECTIONAL, groupAddress4, groupL2Address);
```

protocol to use to send traffic by the applications. This string identifies the socket factory type used to create sockets for the applications. A typical value would be ns3::UdpSocketFactory. address-the address of the remote node to send traffic to.

```
OnOffHelper sidelinkClient0 ("ns3::UdpSocketFactory", remote[0]);
sidelinkClient0.SetConstantRate (DataRate ("16kb/s"), 200);
```

```
OnOffHelper sidelinkClient1 ("ns3::UdpSocketFactory", remote[1]);
sidelinkClient1.SetConstantRate (DataRate ("16kb/s"), 200);
```

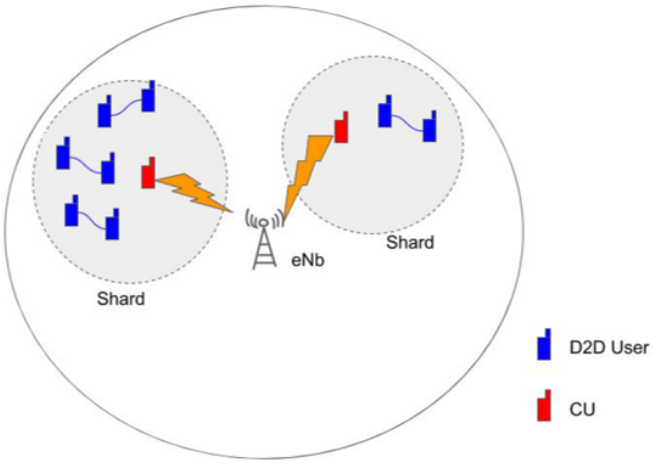
Both the sidelink clients have a data rate of 16kbps and packet size of 200 bytes.

```
ApplicationContainer clientApps[4];
clientApps[0] = sidelinkClient0.Install (ueNodes.Get (2));
//onoff application will send the first packet at :
//(2.9 (App Start Time) + (1600 (Pkt size in bits) / 16000 (Data rate)) = 3.0 sec
int t=2+x[result[0]]++;
//std::cout<<"\t"<<t<<"\n";
clientApps[0].Start (Seconds (t));
clientApps[0].Stop (Seconds (10));
```

Trace file is created and transmission and received traces are set.

IV. Results and Analysis

D2D Pairs,Cellular Users and eNB have been successfully created with default x,y and z coordinates in the program.The available Device to Device (D2D) pairs have been sharded (cluster) based on the distance between the UE's and CU's using clustering algorithm K-nearest neighbors (KNN) algorithm.The shard contents which include the shard no and the x,y,z coordinates of the UEs and CUs are displayed in the terminal mode.



Visualisation after sharding

The D2D pairs start communication in regular intervals i.e, the D2D 1 pair starts data transfer at 2 second, the D2D 2 pair starts data transfer at 3 second, the D2D 3 pair starts data transfer at 4 seconds D2D 4 pair starts data transfer at 5 seconds.The results with respect to the trace files is shown below;

S1.No	Seconds	Active D2D Pairs
1	2	$D2D_1$
2	3	$D2D_1$,$D2D_2$
3	4	$D2D_1$,$D2D_2$,$D2D_3$
4	5	$D2D_1$,$D2D_2$,$D2D_3$,$D2D_4$

Fig. 1

```
gayathri@DESKTOP-BKBH31T:~/psc-ns3-3.0.1$ ./waf --run scratch/final
Waf: Entering directory `/home/gayathri/psc-ns3-3.0.1/build'
[2939/2994] Compiling scratch/final.cc
[2940/2994] Compiling scratch/lte-sl-in-covrg-comm-mode1.cc
[2941/2994] Compiling scratch/scratch-simulator.cc
[2942/2994] Compiling scratch/subdir/scratch-simulator-subdir.cc
[2951/2994] Linking build/scratch/scratch-simulator
[2952/2994] Linking build/scratch/subdir/subdir
[2953/2994] Linking build/scratch/lte-sl-in-covrg-comm-mode1
[2954/2994] Linking build/scratch/final
Waf: Leaving directory `/home/gayathri/psc-ns3-3.0.1/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (35.257s)
Number of Cellular users: 2
Number of D2D pairs: 4
Cellular User: (10,0,0)
Cellular User: (-10,0,0)
D2D User at: (15,0,0)(25,0,0)   Average: (20,0,0)
D2D User at: (-15,0,0)(-25,0,0) Average: (-20,0,0)
D2D User at: (35,5,0)(45,5,0)   Average: (40,5,0)
D2D User at: (-35,10,0)(-45,10,0)       Average: (-40,10,0)

Number of shards created:  2
SHARD 1:
Celluar User present at location: (10,0,0)
D2D pair present at location: (20,0,0)
D2D pair present at location: (40,5,0)

SHARD 2:
Celluar User present at location: (-10,0,0)
D2D pair present at location: (-20,0,0)
D2D pair present at location: (-40,10,0)

Tx address: 7.0.0.4
Tx address: 7.0.0.6
Tx address: 7.0.0.8
Tx address: 7.0.0.10
Rx address: 7.0.0.5
Rx address: 7.0.0.7
Rx address: 7.0.0.9
Rx address: 7.0.0.11
```

Fig. 2: Output

```
time(sec)       tx/rx   NodeID  IMSI    PktSize(bytes)  IP[src] IP[dst]
2.1     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.1     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.17693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.2     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.2     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.29693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.3     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.3     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.37693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.4     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.4     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.49693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.5     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.5     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.57693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.6     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.6     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.69693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.7     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.7     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.77693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.8     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.8     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.89693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
2.9     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
2.9     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
2.97693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
3       tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
3       tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
3.09693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
3.1     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
3.1     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
3.1     tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
3.17693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
3.17693 rx      9       6       200     7.0.0.6:49153   7.0.0.7:8001
3.2     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
3.2     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
3.2     tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
```

Fig. 3: Tracemetrics File

V. Conclusion

Device to Device (D2D) pairs have been sharded based on the distance using the K-nearest neighbors (KNN) algorithm.After which they start communication in regular intervals.Finally the SINR of each D2D pair is calculated.D2D

```
4.7     tx      10      7       200     7.0.0.8:49153   225.0.0.0:8002
4.77693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
4.77693 rx      9       6       200     7.0.0.6:49153   7.0.0.7:8001
4.77693 rx      11      8       200     7.0.0.8:49153   7.0.0.9:8002
4.8     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
4.8     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
4.8     tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
4.8     tx      10      7       200     7.0.0.8:49153   225.0.0.0:8002
4.89693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
4.89693 rx      9       6       200     7.0.0.6:49153   7.0.0.7:8001
4.89693 rx      11      8       200     7.0.0.8:49153   7.0.0.9:8002
4.9     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
4.9     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
4.9     tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
4.9     tx      10      7       200     7.0.0.8:49153   225.0.0.0:8002
4.97693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
4.97693 rx      9       6       200     7.0.0.6:49153   7.0.0.7:8001
4.97693 rx      11      8       200     7.0.0.8:49153   7.0.0.9:8002
5       tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
5       tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
5       tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
5       tx      10      7       200     7.0.0.8:49153   225.0.0.0:8002
5.09693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
5.09693 rx      9       6       200     7.0.0.6:49153   7.0.0.7:8001
5.09693 rx      11      8       200     7.0.0.8:49153   7.0.0.9:8002
5.1     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
5.1     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
5.1     tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
5.1     tx      10      7       200     7.0.0.8:49153   225.0.0.0:8002
5.17693 rx      7       4       200     7.0.0.4:49153   7.0.0.5:8000
5.17693 rx      9       6       200     7.0.0.6:49153   7.0.0.7:8001
5.17693 rx      11      8       200     7.0.0.8:49153   7.0.0.9:8002
5.2     tx      6       3       200     7.0.0.4:49153   225.0.0.0:8000
5.2     tx      12      9       200     7.0.0.10:49153  225.0.0.0:8003
5.2     tx      8       5       200     7.0.0.6:49153   225.0.0.0:8001
```

Fig. 4: Tracemetrics File

communication can play a pivotal role in realizing the ambitious goals of 5G wireless networks. Preliminary systems employing D2D communication have already started to arrive. More developments and industry standards are in the pipeline.

REFERENCES

[1] Asadi A., Wang Q., Mancuso V.A survey on device-to-device communication in cellular networks.IEEE Commun. Surv. Tutor., 16 (4) (2014), pp. 1801-1819

[2] S. Hakola, T. Chen, J. Lehtomäki and T. Koskela, "Device-To-Device (D2D) Communication in Cellular Network - Performance Analysis of Optimum and Practical Communication Mode Selection," 2010 IEEE Wireless Communication and Networking Conference, Sydney, NSW, 2010, pp. 1-6, doi: 10.1109/WCNC.2010.5506133.

[3] https://www.geeksforgeeks.org/k-nearest-neighbours/

[4] J. Lee and J. H. Lee, "Performance Analysis and Resource Allocation for Cooperative D2D Communication in Cellular Networks With Multiple D2D Pairs," in IEEE Communications Letters, vol. 23, no. 5, pp. 909-912, May 2019, doi: 10.1109/LCOMM.2019.2907252.

[5] W. Xueli, J. Zhiyong and Y. Dahai, "An Improved KNN Algorithm Based on Kernel Methods and Attribute Reduction," 2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC), Qinhuangdao, 2015, pp. 567-570, doi: 10.1109/IMCCC.2015.125.