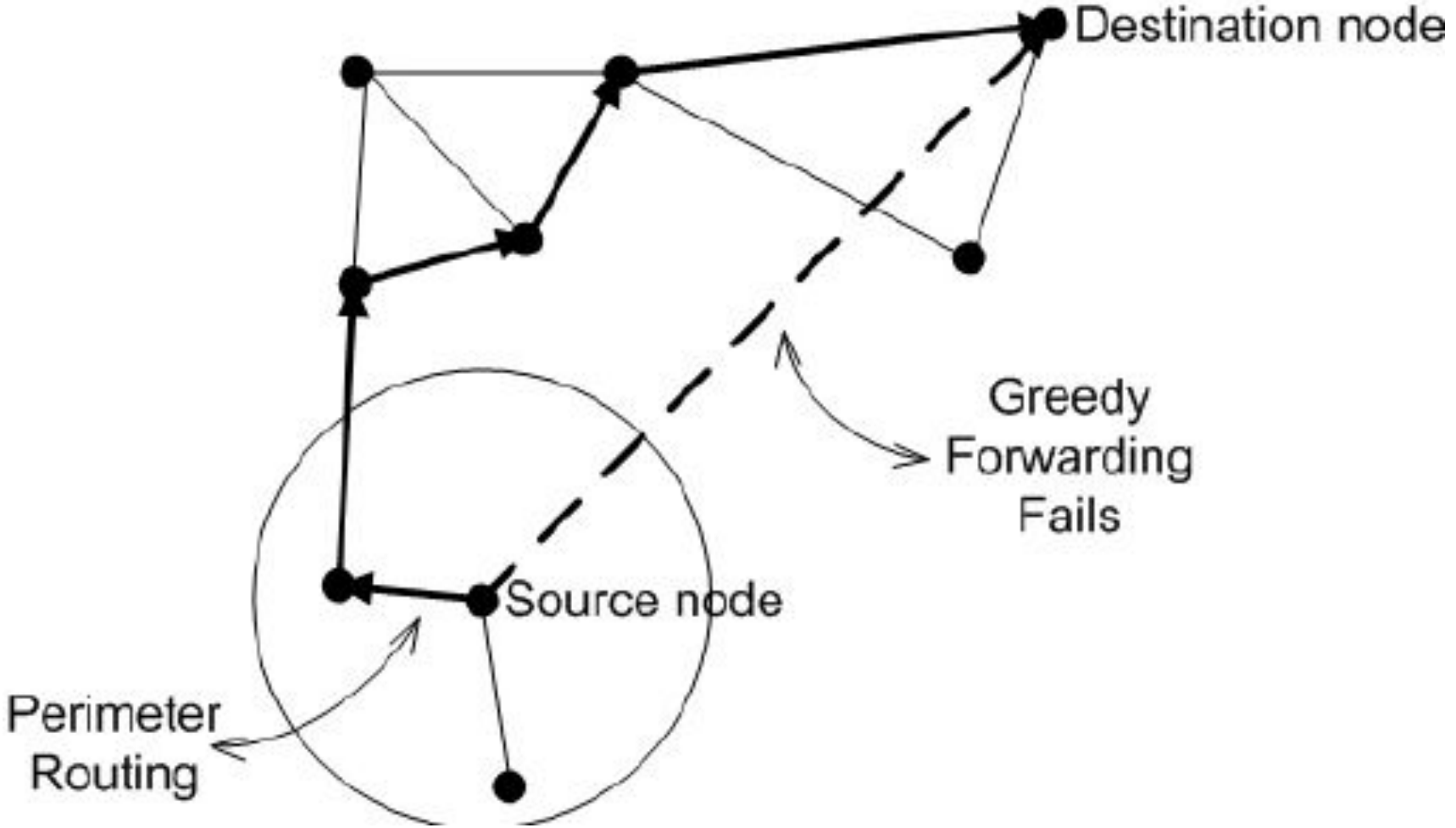
Routing protocol ns-3 implementation - 2

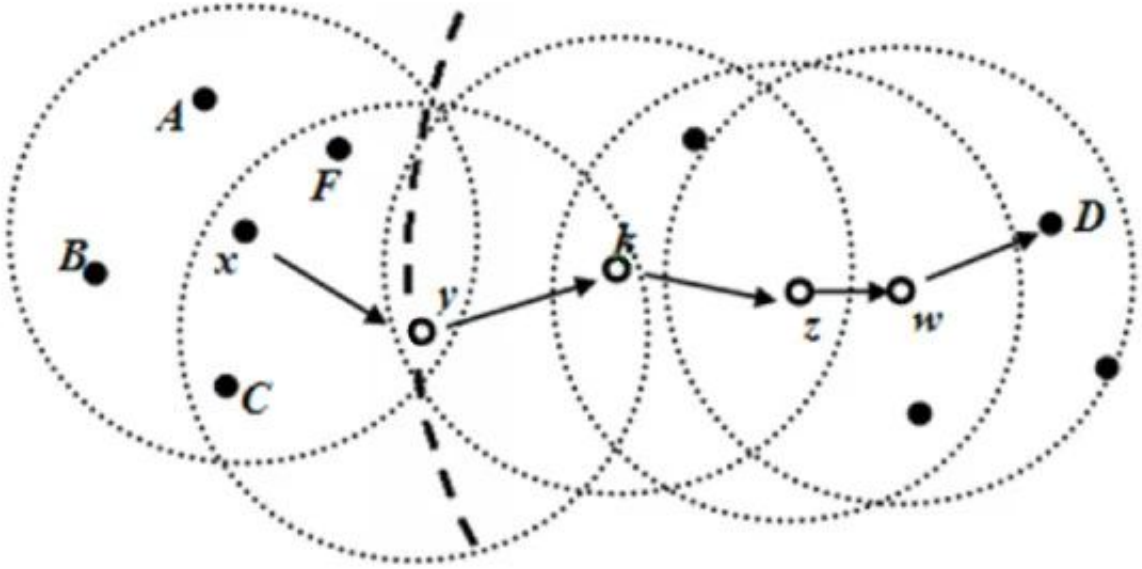
Introduction -- GPSR

- ✔ GPSR 的全稱為 Greedy Perimeter Stateless Routing
- ✔ 使用 GPS 實現路由的一種算法，當節點 S 需要向節點 D 傳送封包的時候，它首先在自己的所有鄰居節點中選擇一個距節點 D 最近的節點作為 Next-hop。該過程一直重複，直到封包到達目的節點 D。
- ✔ 產生或收到數據的節點計算出的最靠近目的節點的 Neighbor 轉發數據，但由於數據會到達沒有比該節點更接近目的點的區域(稱為空洞)，導致數據無法傳輸，利用右手法則沿空洞周圍傳輸來解決此問題。
- ✔ 該協定避免了在節點中建立、維護、存儲路由表，只依賴直接鄰節點進行路由選擇，幾乎是一個無狀態的協定，數據傳輸 delay 小，且保證只要網路連通性不被破壞，一定能夠發現可達路由。。

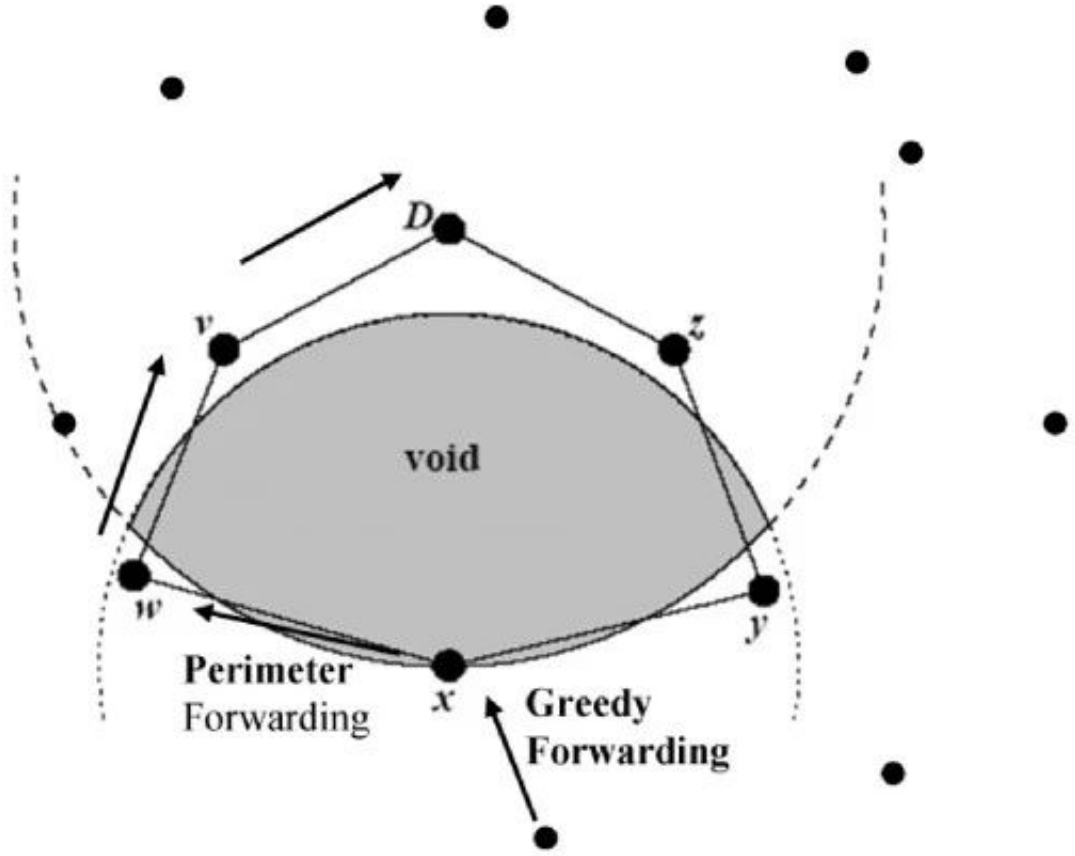
Introduction -- GPSR



Introduction -- GPSR



(a)

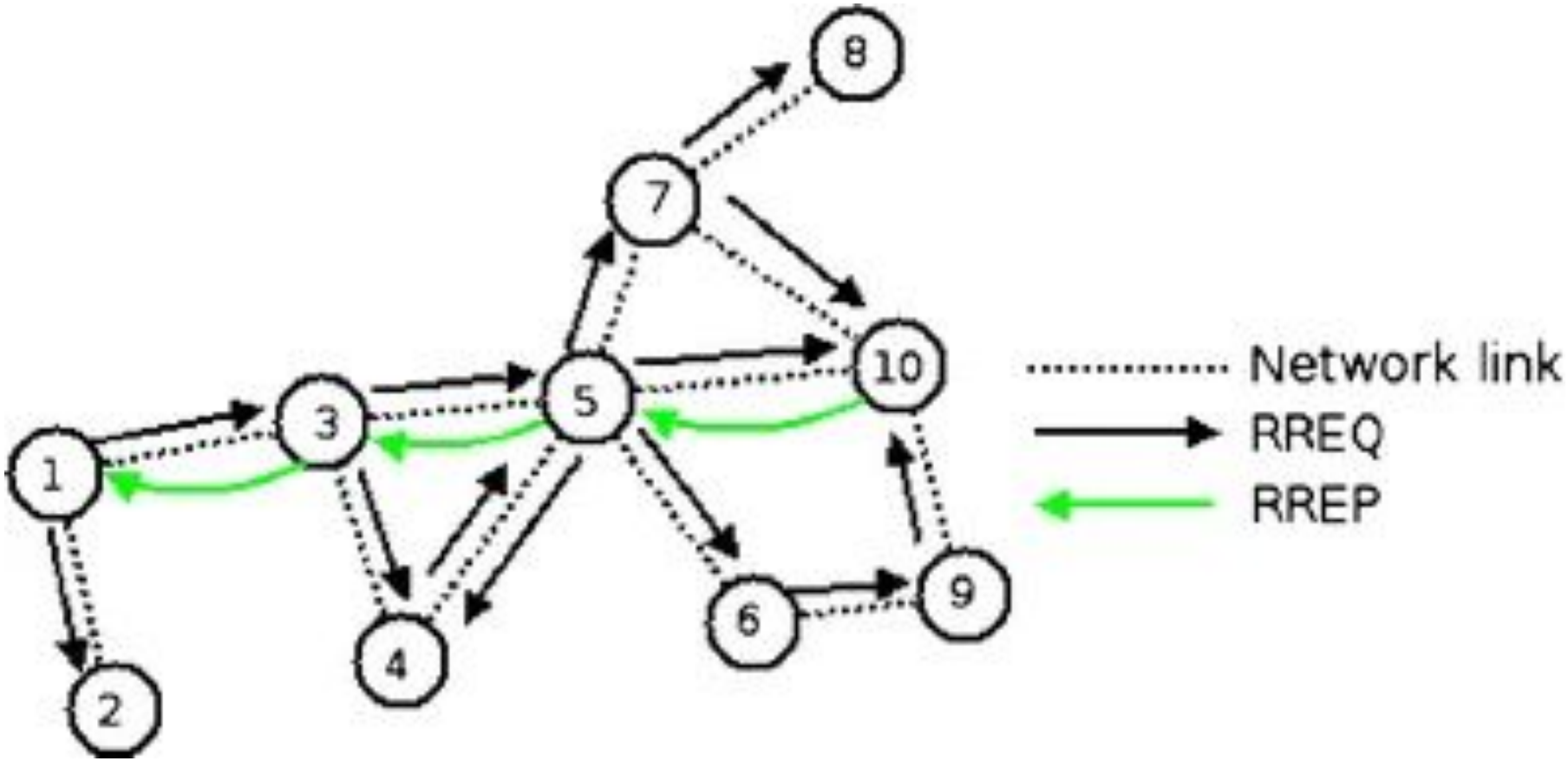


(b)

Introduction -- AODV

- ✔ AODV 的全稱為 Ad-hoc On-demand Distance Vector
- ✔ 此種路由方式特別之處是只有當要發送網路封包的時候，看看目的地在哪裡，才去研究最佳路由是什麼，所以才叫做 Ad-hoc On-demand，每一個路由請求會有一個序號，大家使用這個序號以免重複發送相同的請求。當然每個請求會有一段「生存時間」，一旦逾時，這個請求就會失效。
- ✔ 因為這種路由方式的設計，大家應該可以想像得到，一旦這個的網路環境 沒有任何的封包發送行為時，整個網路是非常靜止的，不像其他路由方式會一直想要互相發送更新以便於維護最新的路由表

Distance vector algorithm



GPSR 程式碼說明

```
38
39  // Define enumeration for PayLoad type
40  enum
41  {
42      HELLO,
43      STANDARD,
44  };
45
```

GPSR 程式碼說明

```
602  for (double t = 0; t < simulationTime; t += helloSendAfter)
603  {
604      for (uint32_t i = 0; i < numNodes; i++)
605      {
606          Ipv4InterfaceAddress iaddrSender = c.Get(i)->GetObject<Ipv4>()->GetAddress(1, 0);
607          Ipv4Address ipSender = iaddrSender.GetLocal();
608
609          Ptr<Socket> socket = Socket::CreateSocket(c.Get(i), tid);
610
611          PayLoadConstructor payload = PayLoadConstructor(HELLO);
612          payload.setTtl(TTL);
613          payload.setUid(0);
614          payload.setNextHopAddress(ipSender);
615          payload.setNeighborId(i);
616          payload.setDestinationAddress(ipSender);
617          payload.setDestinationId(i);
618          Ptr<Packet> packet = payload.toPacket();
619
620          InetSocketAddress remote = InetSocketAddress(Ipv4Address("255.255.255.255"), 80);
621          socket->Connect(remote);
622          socket->SetAllowBroadcast(true);
623
624          Ptr<UniformRandomVariable> x = CreateObject<UniformRandomVariable>();
625          double randomPause = x->GetValue(0, 0.5);
626
627          Simulator::Schedule(Seconds(t + randomPause), &GenerateTraffic, socket, packet, payload.getUid(), TTL);
628      }
629  }
```

GPSR 程式碼說明

```
639  for (double t = warmingTime; t < simulationTime - sendUntil; t += sendAfter)
640  {
641      // Create socket
642      Ptr<Socket> socket = Socket::CreateSocket(c.Get(sourceNode), tid);
643      NodeHandler *currentNode = &nodeHandlerArray[socket->GetNode()->GetId()];
644
645      PayLoadConstructor payload = PayLoadConstructor(STANDARD);
646      payload.setTtl(TTL);
647      payload.setUid(UID);
648      payload.setNextHopAddress(ipSender);
649      payload.setNeighborId(sourceNode);
650      payload.setDestinationAddress(ipReceiver);
651      payload.setDestinationId(sinkNode);
652      Ptr<Packet> packet = payload.toPacket();
653
654      PacketLogData dataPacket = {false, -1, 0.00, 0};
655      dataForPackets.push_back(dataPacket);
656
657      Ptr<UniformRandomVariable> x = CreateObject<UniformRandomVariable>();
658      double randomPause = x->GetValue(0, 1);
659
660      Simulator::Schedule(Seconds(t + randomPause), &ScheduleNeighbor, socket, packet, currentNode, sinkNode);
661      UID += 1;
662  }
```

GPSR 程式碼說明

```
435     while (packet = socket->RecvFrom(from))
436     {
437         NodeHandler *currentNode = &nodeHandlerArray[socket->GetNode()->GetId()];
438         Ipv4Address ipSender = InetSocketAddress::ConvertFrom(from).GetIpv4();
439 
440         currentNode->increasePacketsReceived(1);
441         PayLoadConstructor payload = PayLoadConstructor(HELLO);
442         payload.fromPacket(packet);
443 
444         Ipv4Address nextHopAddress = payload.getNextHopAddress();
445         uint32_t neighborId = payload.getNeighborId();
446         uint32_t UID = payload.getUid();
447         uint32_t TTL = payload.getTtl();
448 
449         if ((payload.getDestinationAddress() == ipReceiver) && payload.getUid() != 0)
450         {
451             if (dataForPackets[payload.getUid()].delivered != true)
452             {
453                 dataForPackets[payload.getUid()].delivered = true;
454                 dataForPackets[payload.getUid()].delivered_at = Simulator::Now().GetSeconds();
455                 dataForPackets[payload.getUid()].ttl = payload.getTtl();
456 
457                 NS_LOG_UNCOND(Simulator::Now().GetSeconds() << "s\t PKT DESTINATION REACHED, UID: " << payload.getUid());
458             }
459             else
460             {
461                 NS_LOG_UNCOND(Simulator::Now().GetSeconds() << "s\t " << ipReceiver << "\tRE-received the package with uid: " << UID);
462             }
463 
464             continue;
465         }
466 
```

GPSR 程式碼說明

```
466
467     // receive the packet
468     if (payload.getType() == HELLO)
469     {
470         if (ipSender == nextHopAddress)
471         {
472             currentNode->increaseBytesReceived();
473             currentNode->setFindNeighbor(neighborId);
474         }
475     }
476
```

GPSR 程式碼說明

```
476
477         // selected neighbor node receive the packet
478         else if (payload.getType() == STANDARD)
479         {
480             if (ipReceiver == nextHopAddress && currentNode->searchInStack(UID) == false)
481             {
482                 currentNode->increaseBytesReceived();
483                 currentNode->increaseBuffer();
484
485                 // if ((int)socket->GetNode()->GetId() == 100)
486                 // NS_LOG_UNCOND(time << "s\t" << ipReceiver << "\t" << socket->GetNode()->GetId() << "\tReceived pkt type: " << payload.getType() << "\twith uid: " << UID << "\tfrom: " << ipSend
487
488                 if ((dataForPackets[UID].start + (double)TTL >= Simulator::Now().GetSeconds()) && currentNode->checkBufferSize())
489                 {
490                     destinationId = payload.getDestinationId();
491                     destinationAddress = payload.getDestinationAddress();
492                     currentNode->savePacketsInBuffer(payload);
493                     forwarding += 1;
494
495                     Ptr<UniformRandomVariable> x = CreateObject<UniformRandomVariable>();
496                     double randomPause = x->GetValue(0, 1);
497                     Simulator::Schedule(Seconds(randomPause), &ScheduleNeighbor, socket, packet, currentNode, destinationId);
498                 }
499             }
500         }
501     }
502 }
```

GPSR 程式碼說明

```
345  void ScheduleNeighbor(Ptr<Socket> socket, Ptr<Packet> packet, NodeHandler *currentNode, uint32_t destinationId)
346  {
347      Ptr<MobilityModel> current_mob = c.Get(currentNode->getNodeID())->GetObject<MobilityModel>();
348      float src_X = current_mob->GetPosition().x;
349      float src_Y = current_mob->GetPosition().y;
350
351      Ptr<MobilityModel> dst_mob = c.Get(destinationId)->GetObject<MobilityModel>();
352      float dst_X = dst_mob->GetPosition().x;
353      float dst_Y = dst_mob->GetPosition().y;
354
355      Ipv4Address nextHopAddress;
356      bool check = false;
357      double rec_distance = 100000;
358
359      PayLoadConstructor payload = PayLoadConstructor(HELLO);
360      payload.fromPacket(packet);
361
362      uint32_t UID = payload.getUid();
363      uint32_t ttl = payload.getTtl();
364      int temp_distance, validation;
365
366      for (uint32_t i = 0; i < numNodes; ++i)
367      {
```

GPSR 程式碼說明

```
366     for (uint32_t i = 0; i < numNodes; ++i)
367     {
368         uint32_t node = i;
369 
370         if (currentNode->getFindNeighbor(node) == 0)
371             continue;
372 
373         Ptr<Ipv4> ipv4 = c.Get(node)->GetObject<Ipv4>();
374         Ipv4InterfaceAddress iaddr = ipv4->GetAddress(1, 0);
375         Ipv4Address ipSender = iaddr.GetLocal();
376 
377         Ptr<MobilityModel> node_mob = c.Get(node)->GetObject<MobilityModel>();
378         NodeHandler *neighborNode = &nodeHandlerArray[node];
379 
380         float node_X = node_mob->GetPosition().x;
381         float node_Y = node_mob->GetPosition().y;
382 
383         temp_distance = dist(node_X, node_Y, dst_X, dst_Y);
384         validation = dist(node_X, node_Y, src_X, src_Y);
385 
386         if (payload.getDestinationAddress() == ipSender && validation <= distance)
387         {
388             nextHopAddress = ipSender;
389             check = true;
390             break;
391         }
392 
393         if (temp_distance < rec_distance && validation <= distance && neighborNode->searchInStack(UID) == false)
394         {
395             nextHopAddress = ipSender;
396             rec_distance = temp_distance;
397             check = true;
398         }
399     }
400 
```

GPSR 程式碼說明

```
345  void ScheduleNeighbor(Ptr<Socket> socket, Ptr<Packet> packet, NodeHandler *currentNode, uint32_t destinationId)
346  {
347      Ptr<MobilityModel> current_mob = c.Get(currentNode->getNodeID())->GetObject<MobilityModel>();
348      float src_X = current_mob->GetPosition().x;
349      float src_Y = current_mob->GetPosition().y;
350
351      Ptr<MobilityModel> dst_mob = c.Get(destinationId)->GetObject<MobilityModel>();
352      float dst_X = dst_mob->GetPosition().x;
353      float dst_Y = dst_mob->GetPosition().y;
354
355      Ipv4Address nextHopAddress;
356      bool check = false;
357      double rec_distance = 100000;
358
359      PayLoadConstructor payload = PayLoadConstructor(HELLO);
360      payload.fromPacket(packet);
361
362      uint32_t UID = payload.getUid();
363      uint32_t ttl = payload.getTtl();
364      int temp_distance, validation;
```

GPSR 程式碼說明

```
561     // Set it to adhoc mode
562     wifiMac.SetType("ns3::AdhocWifiMac");
563     NetDeviceContainer devices = wifi.Install(wifiPhy, wifiMac, c);
564
565     MobilityHelper mobility;
566     mobility.SetPositionAllocator("ns3::RandomRectanglePositionAllocator",
567                                   "X", StringValue("ns3::UniformRandomVariable[Min=0.0|Max=3000.0]"),
568                                   "Y", StringValue("ns3::UniformRandomVariable[Min=0.0|Max=1500.0]"));
569
570     mobility.SetMobilityModel("ns3::SteadyStateRandomWaypointMobilityModel",
571                               "MinSpeed", DoubleValue(5),
572                               "MaxSpeed", DoubleValue(10),
573                               "MinX", DoubleValue(0.0),
574                               "MaxX", DoubleValue(3000.0),
575                               "MinPause", DoubleValue(10),
576                               "MaxPause", DoubleValue(20),
577                               "MinY", DoubleValue(0.0),
578                               "MaxY", DoubleValue(1500.0));
579     mobility.Install(c);
580
```

作業繳交說明

GPSR _example

gpsr_example 檔案在ftp可以下載

將檔案放入至 ns-3.35/scratch 資料夾

```
$ cd && cd ns-allinone-3.35/ns-3.35/
```

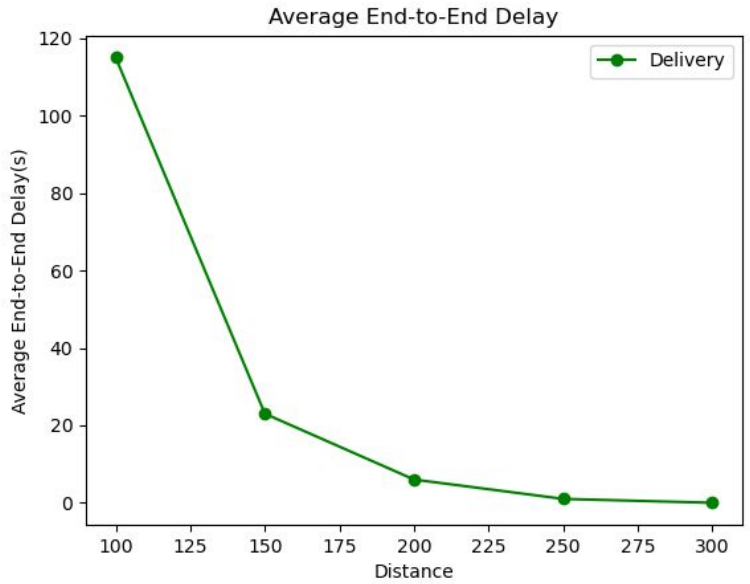
```
$  ./waf --run gpsr_example
```

作業說明 GPSR

- ✔ distance (傳輸半徑)
- ✔ numNodes (節點個數)
- ✔ TTL (封包存活時間)
- ✔ 挑選兩種參數，區間 [a1, a2, a3, a4, a5]，進行模擬
- ✔ 比較 delivery ratio
- ✔ 比較 end2end delay
- ✔ 比較 overhead (PacketsReceived / PacketsSent)
- ✔ 畫成折線圖

NS-3 作業繳交內容

GPSR折線圖(圖一)



(圖一)

作業說明 AODV

檔案位置 : /ns-allinone-3.35/ns-3.35/src/aodv/examples/aodv.cc

將檔案放入至 ns-3.35/scratch 資料夾

```
$ cd && cd ns-allinone-3.35/ns-3.35/
```

更改參數與 GPSR 相同

```
$  ./waf --run aodv
```

截圖執行結果

作業說明 AODV

把GPSR的參數帶入AODV

```
int main(int argc, char *argv[])
{
    std::string phyMode("DsssRate11Mbps");
    distance = 200;
    helloSendAfter = 1;

    double simulationTime = 200.00;
    double sendUntil = 50.00;
    double warmingTime = 10.00;
    uint32_t seed = 123;

    numNodes = 150;
    uint32_t sendAfter = 1;
    uint32_t sinkNode = 20;
    uint32_t sourceNode = 21;

    uint32_t TTL = 10;
    uint32_t UID = 1;
    double rss = -80;
```

GPSR程式內的參數

作業說明 AODV

把GPSR的參數帶入AODV

```
115  //-----------------------------------------------------------------------
116  AodvExample::AodvExample () :
117    size (150),
118    step (50),
119    totalTime (100),
120    pcap (true),
121    printRoutes (true)
122  {
123  }
124
125  bool
126  AodvExample::Configure (int argc, char **argv)
127  {
128    // Enable AODV logs by default. Comment this if too noisy
129    // LogComponentEnable("AodvRoutingProtocol", LOG_LEVEL_ALL);
130
131    SeedManager::SetSeed (12345);
132    CommandLine cmd (__FILE__);
133
134    cmd.AddValue ("pcap", "Write PCAP traces.", pcap);
135    cmd.AddValue ("printRoutes", "Print routing table dumps.", printRoutes);
136    cmd.AddValue ("size", "Number of nodes.", size);
137    cmd.AddValue ("time", "Simulation time, s.", totalTime);
138    cmd.AddValue ("step", "Grid step, m", step);
139
140    cmd.Parse (argc, argv);
141    return true;
142  }
```

AODV程式內的參數

NS-3 作業繳交內容

AODV的截圖(圖二)

```
64 bytes from 10.0.0.10: icmp_seq=32 ttl=56 time=+7.31102ms
64 bytes from 10.0.0.10: icmp_seq=33 ttl=56 time=+7.38302ms
--- 10.0.0.10 ping statistics ---
100 packets transmitted, 34 received, 66% packet loss, time +1e+05ms
rtt min/avg/max/mdev = 7/99.82/2057/390 ms
duckie@duckie-VirtualBox:~/ns-allinone-3.35/ns-3.35$
```

(圖二)

NS-3 作業繳交內容

內容必須有GPSR與Epidemic比較折線圖(圖一)以及AODV的截圖(圖二)加以描述、心得。

需繳交檔案:1. 紙本檔案、2. 電子檔案上傳 FTP、3. 檔名為 : 學號_HW5.pdf (補交檔名 :學號_HW5-2)

上傳 : 120.107.172.19 使用者名稱 : 1132VANET 密碼 : 1132student