دانشگاه شهید بهشتی

دانشکده مهندسی و علوم کامپیوتر

درس شبکه‌های کامپیوتری پیشرفته

گزارش پروژه‌ی نهایی

استاد درس:

جناب آقای دکتر مقصود عباسپور

توسط :

امیرمحمد پیرحسینلو 401443029

نیلوفر بهرام‌زاده 401443022

1401

برای ساخت و ایجاد پروژه ابتدا به تعریف مقادیر اولیه در تابع main می‌پردازیم.

کاربر می‌تواند با استفاده از ترمینال، ۲ پارامتر را مقداردهی کند:

- متغیر **p2pDataRateCmd**: برابر است با میزان گذردهی لینک‌های نقطه به نقطه.
- متغیر **p2pDelayMSCmd**: برای تعیین مقدار تاخیر لینک‌های نقطه به نقطه بر حسب میلی ثانیه است.

```
void parseCMD(
    int argc,
    char *argv[],
    string &p2pDataRateCmd,
    uint64_t &p2pDelayMSCmd)
{
    CommandLine cmd;
    cmd.AddValue("p2pDataRate", "p2p data rate", p2pDataRateCmd);
    cmd.AddValue("p2pDelay", "p2p delay(ms)", p2pDelayMSCmd);

    cmd.Parse(argc, argv);
}
```

حال با کمک کتابخانه‌ی NS3 به ساخت متغیرهای توپولوژی میپردازیم:

```
    NodeContainer p2pNodes,
        northCsmaNodes,
        southCsmaNodes,
        northStationNodes,
        southStationNodes;

    Ptr<Node> northCsmaGateway,
        southCsmaGateway,
        northAccessPoint,
        southAccessPoint;

    vector<NodeContainer> p2pLinks;

    vector<NetDeviceContainer> p2pDevices,
        northCsmaDevices,
        southCsmaDevices,
        northWifiDevices,
```

```
        southWifiDevices;

    vector<Ipv4InterfaceContainer> p2pInterfaces,
        northCsmaInterfaces,
        southCsmaInterfaces,
        northWifiInterfaces,
        southWifiInterfaces;

    MobilityHelper mobilityHelper;
```

حال کلاسی به نام **Edge** می‌سازیم. ویژگی‌های این کلاس نمایانگر تنظیمات لینک‌های نقطه به نقطه هستند. این تنظیمات شامل موارد زیر است:
گره‌ی مبدا، گره‌ی مقصد، پیشوند شبکه و network mask.

```
class Edge
{
public:
    Ptr<Node> source, sink;
    string network, mask;

    Edge(Ptr<Node> source, Ptr<Node> sink, string network, string mask)
    {
        this->source = source;
        this->sink = sink;
        this->network = network;
        this->mask = mask;
    }
};
```

کلاس دیگری به نام **OnOffScenario** برای ساخت رویدادهای بیان شده ایجاد می‌کنیم.

```
class OnOffScenario
{
public:
    Ptr<Node> source;
    Ptr<Node> sink;
    Ipv4Address sinkIP;
    uint16_t sinkPort;
```

```
    StringValue onTime, offTime;
    StringValue dataRate;
    UintegerValue packetSize;
    Time start, end;

    OnOffScenario(
        Ptr<Node> source,
        Ptr<Node> sink,
        Ipv4Address sinkIP,
        uint16_t sinkPort,
        StringValue onTime,
        StringValue offTime,
        StringValue dataRate,
        UintegerValue packetSize,
        Time start,
        Time end)
    {
        this->source = source;
        this->sink = sink;
        this->sinkIP = sinkIP;
        this->sinkPort = sinkPort,
        this->onTime = onTime;
        this->offTime = offTime;
        this->dataRate = dataRate;
        this->packetSize = packetSize;
        this->start = start;
        this->end = end;
    }
};
```

پس از تعریف متغیرهای انواع اپلیکیشن‌های مورد نیاز (sourceApps, sinkApps, udpClientApps, udpServerApps) به سراغ ساخت توپولوژی می رویم.

```
vector<OnOffScenario> scenarios;
    vector<ApplicationContainer> sourceApps;
    vector<ApplicationContainer> sinkApps;
    vector<ApplicationContainer> udpClientApps;
    vector<ApplicationContainer> udpServerApps;
```

با فراخوانی تابع **create_nodes**، تمامی مراحل ساخت ۷ گره‌ی موجود در شبکه و تعریف و نامگذاری آن ها طی می‌شود.

```
void create_nodes(
   NodeContainer &p2pNodes,
   NodeContainer &northCsmaNodes,
   NodeContainer &southCsmaNodes,
   NodeContainer &northStationNodes,
   NodeContainer &southStationNodes,
   Ptr<Node> &northCsmaGateWay,
   Ptr<Node> &southCsmaGateWay,
   Ptr<Node> &northAccessPoint,
   Ptr<Node> &southAccessPoint,
   unordered_map<int, Ptr<Node>> &indexToNode)
{
   p2pNodes.Create(7);
   northCsmaNodes.Create(2);
   southCsmaNodes.Create(2);
   northStationNodes.Create(2);
   southStationNodes.Create(2);

   indexToNode.insert({
       {0, p2pNodes.Get(0)},
       {1, p2pNodes.Get(1)},
       {2, p2pNodes.Get(2)},
       {3, p2pNodes.Get(3)},
       {4, p2pNodes.Get(4)},
       {11, p2pNodes.Get(5)},
       {31, p2pNodes.Get(6)},
       {12, northCsmaNodes.Get(0)},
       {13, northCsmaNodes.Get(1)},
       {32, southCsmaNodes.Get(0)},
       {33, southCsmaNodes.Get(1)},
       {21, northStationNodes.Get(0)},
       {22, northStationNodes.Get(1)},
       {41, southStationNodes.Get(0)},
       {42, southStationNodes.Get(1)},
   });
```

```
    northCsmaGateWay = indexToNode[11];
    southCsmaGateWay = indexToNode[31];
    northAccessPoint = indexToNode[2];
    southAccessPoint = indexToNode[4];

    InternetStackHelper inetStackHelper;
    inetStackHelper.Install(p2pNodes);
    inetStackHelper.Install(northCsmaNodes);
    inetStackHelper.Install(southCsmaNodes);
    inetStackHelper.Install(northStationNodes);
    inetStackHelper.Install(southStationNodes);
}
```

سپس با فراخوانی تابع **create_edges** تمامی لینک‌ها را ایجاد و با کمک تابع **setup_p2p** پهنای باند و تاخیر لینک‌های برگرفته از کلاس Edge را تعریف می‌کنیم و لینک‌ها را به توپولوژی و شبکه می افزاییم.

```
void create_edges(vector<Edge> &edges, unordered_map<int, Ptr<Node>>
&indexToNode)
{
    edges.push_back(
        Edge(
            indexToNode[0],
            indexToNode[1],
            "192.168.1.0",
            "255.255.255.0"));
    edges.push_back(
        Edge(
            indexToNode[0],
            indexToNode[2],
            "192.168.2.0",
            "255.255.255.0"));
    edges.push_back(
        Edge(indexToNode[0],
             indexToNode[3],
             "192.168.3.0",
             "255.255.255.0"));
    edges.push_back(
        Edge(
            indexToNode[0],
```

```
            indexToNode[4],
            "192.168.4.0",
            "255.255.255.0"));
    edges.push_back(
        Edge(
            indexToNode[1],
            indexToNode[11],
            "192.168.5.0",
            "255.255.255.0"));
    edges.push_back(
        Edge(
            indexToNode[3],
            indexToNode[31],
            "192.168.6.0",
            "255.255.255.0"));
}
```

پروتکل CSMA را با تابع **setup_csma** به زیرشبکه‌های LAN اعمال می‌کنیم.

```
void setup_csma(
    NodeContainer &nodes,
    Ptr<Node> &gateway,
    vector<NetDeviceContainer> &devices,
    vector<Ipv4InterfaceContainer> &interfaces,
    StringValue dataRate,
    TimeValue delay,
    string network,
    string mask)
{
    nodes.Add(gateway);

    CsmaHelper csmaHelper;
    csmaHelper.SetChannelAttribute("DataRate", dataRate);
    csmaHelper.SetChannelAttribute("Delay", delay);

    NetDeviceContainer device = csmaHelper.Install(nodes);

    Ipv4AddressHelper ipv4Helper;
```

```
    ipv4Helper.SetBase(network.c_str(), mask.c_str());
    Ipv4InterfaceContainer interface = ipv4Helper.Assign(device);

    devices.push_back(device);
    interfaces.push_back(interface);

    string pcapFileNamePrefix = PATH_PREFIX + "csma/" + network + "-";
    csmaHelper.EnablePcap(pcapFileNamePrefix, device, 0);

    AsciiTraceHelper ascii;
    csmaHelper.EnableAsciiAll(ascii.CreateFileStream(PATH_PREFIX + "csma/"
+ network + ".tr"));
}
```

به همین ترتیب به ساخت زیرشبکه‌های wifi از طریق تابع **setup_wifi** می‌پردازیم.

```
void setup_wifi(
    NodeContainer &stationNodes,
    Ptr<Node> &accessPointNode,
    vector<NetDeviceContainer> &devices,
    vector<Ipv4InterfaceContainer> &interfaces,
    MobilityHelper &mobilityHelper,
    Ssid ssid,
    string network,
    string mask)
{
    YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
    YansWifiPhyHelper phy = YansWifiPhyHelper::Default();
    phy.SetChannel(channel.Create());

    WifiHelper wifiHelper;
    wifiHelper.SetRemoteStationManager("ns3::AarfWifiManager");

    WifiMacHelper macHelper;
    macHelper.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid),
                      "ActiveProbing", BooleanValue(false));

    NetDeviceContainer stationDevice = wifiHelper.Install(phy, macHelper,
stationNodes);
```

```
   macHelper.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));

   NetDeviceContainer accessPointDevice = wifiHelper.Install(phy,
macHelper, accessPointNode);

   Ipv4AddressHelper ipv4Helper;
   ipv4Helper.SetBase(network.c_str(), mask.c_str());
   Ipv4InterfaceContainer stationInterface =
ipv4Helper.Assign(stationDevice);
   Ipv4InterfaceContainer accessPointInterface =
ipv4Helper.Assign(accessPointDevice);

   mobilityHelper.Install(stationNodes);
   mobilityHelper.Install(accessPointNode);

   devices.push_back(stationDevice);
   devices.push_back(accessPointDevice);
   interfaces.push_back(stationInterface);
   interfaces.push_back(accessPointInterface);

   string stationPcapFileNamePrefix = PATH_PREFIX + "wifi/" + network +
"-station-";
   phy.EnablePcap(stationPcapFileNamePrefix, stationDevice, 0);
   string accessPointPcapFileNamePrefix = PATH_PREFIX + "wifi/" + network
+ "-accesspoint-";
   phy.EnablePcap(accessPointPcapFileNamePrefix, accessPointDevice, 0);

   AsciiTraceHelper ascii;
   phy.EnableAsciiAll(ascii.CreateFileStream(PATH_PREFIX + "wifi/" +
network + ".tr"));
}
```

پس از تعریف و ساخت توپولوژی شبکه، در تابع fill_on_off_scenarios رویدادهای خواسته شده را ساخته و با تابع setup_on_off_application آن‌ها را اجرا می‌کنیم.

تابعی نیز به نام setup_link_failure تعریف می‌کنیم تا زمان‌های قطعی لینک‌های n0n2 ،n0n1 و n0n3را در شبکه مدیریت کنیم.

در نهایت خروجی را با توجه به انتخاب‌های اولیه‌ی کاربر با استفاده از ترمینال، نمایش می‌دهیم.

خروجی ترمینال به شکل زیر است:

```
p2pDataRateCmd = 2Mbps, p2pDelayMSCmd = 30
-----
10.1.3.1
10.1.3.2
10.1.3.3
10.1.4.1
10.1.4.2
10.1.4.3
10.1.5.1
10.1.5.2
10.1.5.3
-----
-----
10.1.4.1
10.1.4.2
-----
-------
Remove n0 --- > n1: 192.168.1.1
Remove n1 --- > n0: 192.168.1.2
Remove n0 --- > n2: 192.168.2.1
Remove n2 --- > n0: 192.168.2.2
Remove n0 --- > n3: 192.168.3.1
Remove n3 --- > n0: 192.168.3.2
-------
-----
context, /NodeList/8/ApplicationList/0/$ns3::UdpEchoClient/Tx
udp echo client Tx Trace, 1
-----
At time 1s client sent 1024 bytes to 10.1.4.2 port 5432
At time 1.11379s server received 1024 bytes from 10.1.2.2 port 49153
At time 1.11379s server sent 1024 bytes to 10.1.2.2 port 49153
At time 1.22385s client received 1024 bytes from 10.1.4.2 port 5432
-----
context, /NodeList/8/ApplicationList/0/$ns3::UdpEchoClient/Rx
udp echo client Rx Trace, 1.22385
RTT = 0.223846 seconds
-----
```

نکته مهم: برای اجرا کد، نیاز است در دایرکتوری‌ای که waf قرار دارد، دایرکتوری‌های زیر ایجاد شوند:

- final-project-traces/

- final-project-traces/p2p/
- final-project-traces/csma/
- final-project-traces/wifi/

مانند شکل زیر:

```
:~/Downloads/ns-allinone-3.29/ns-3.29/final-project-traces$ tree ./
./
├── csma
│   ├── 10.1.2.0--5-2.pcap
│   ├── 10.1.2.0--7-1.pcap
│   ├── 10.1.2.0--8-1.pcap
│   ├── 10.1.2.0.tr
│   ├── 10.1.5.0--10-1.pcap
│   ├── 10.1.5.0--6-2.pcap
│   ├── 10.1.5.0--9-1.pcap
│   └── 10.1.5.0.tr
├── p2p
│   ├── 0-1--0-1.pcap
│   ├── 0-1--1-1.pcap
│   ├── 0-2--0-2.pcap
│   ├── 0-2--2-1.pcap
│   ├── 0-3--0-3.pcap
│   ├── 0-3--3-1.pcap
│   ├── 0-4--0-4.pcap
│   ├── 0-4--4-1.pcap
│   ├── 1-5--1-2.pcap
│   ├── 1-5--5-1.pcap
│   ├── 3-6--3-2.pcap
│   ├── 3-6--6-1.pcap
│   └── p2p.tr
└── wifi
    ├── 10.1.3.0-accesspoint--2-2.pcap
    ├── 10.1.3.0-station--11-1.pcap
    ├── 10.1.3.0-station--12-1.pcap
    ├── 10.1.3.0.tr
    ├── 10.1.4.0-accesspoint--4-2.pcap
    ├── 10.1.4.0-station--13-1.pcap
    ├── 10.1.4.0-station--14-1.pcap
    └── 10.1.4.0.tr

3 directories, 29 files
:~/Downloads/ns-allinone-3.29/ns-3.29/final-project-traces$
```

تعداد پکت‌های ارسالی از n3 به n22:

command:

tcpdump -nn -tt -r p2p/0-3--3-1.pcap | grep '10.1.3.2'

output:

reading from file p2p/0-3--3-1.pcap, link-type PPP (PPP)

3.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

6.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

9.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

12.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

18.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

21.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

24.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

27.000000 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

با توجه به تنظیمات انجام شده برای اپلیکیشن n3 به n22 (نرخ ارسال 512Bps و اندازه‌ی بسته‌ی 512B)، هر ۳ ثانیه، باید یک بسته از n3 به n22 فرستاده شود. همانطور که دیده می‌شود، به دلیل قطعی لینک بین n0 وn3 بین زمان‌های ۱۵ تا ۱۸، در ثانیه‌ی ۱۵ بسته‌ای ارسال نشده است (بسته از بین رفته است). لینک بین n0 و n2 بین زمان‌های ده ثانیه تا ۱۲ ثانیه نیز قطع است منتهی اثری ندارد زیرا در این بازه، n3 بسته‌ای ارسال نمی‌کند. اتفاقات از دید n22:

command:

tcpdump -nn -tt -r wifi/10.1.3.0-station--12-1.pcap | grep '10.1.3.2'

output:

reading from file wifi/10.1.3.0-station--12-1.pcap, link-type IEEE802_11 (802.11)

3.064448 ARP, Request who-has 10.1.3.2 (ff:ff:ff:ff:ff:ff) tell 10.1.3.3, length 32

3.064491 ARP, Reply 10.1.3.2 is-at 00:00:00:00:00:14, length 32

3.065570 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

6.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

9.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

12.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

18.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

21.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

24.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

27.065128 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

همانطور که مشاهده می‌شود، در ثانیه‌ی ۱۵ بسته‌ای دریافت نشده است (از بین رفته است.).
در نتیجه، در کل یک بسته drop شده است.

تعداد پکت‌های رسیده به n2 در هر یک ثانیه:

command:
tcpdump -nn -tt -r p2p/0-2--2-1.pcap | grep -v '192.168.2.2'

output:
reading from file p2p/0-2--2-1.pcap, link-type PPP (PPP)
3.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
6.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
9.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
12.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
18.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
21.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
24.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512
27.064335 IP 192.168.3.2.49153 > 10.1.3.2.5443: UDP, length 512

در هر کدام از فواصل زیر، ۵۱۲ بایت داده به n2 رسیده‌است:

3s-4s 6s-7s 9s-10s 12s-13-s 18s-19s
21s-22s 24s-25s 27s-28s

این داده‌ها در نهایت به n22 می‌رسند.
در سایر فواصل، میزان داده‌ی دریافتی برابر صفر است.