Real-time Transport Protocol을 이용한 Video Streaming Application

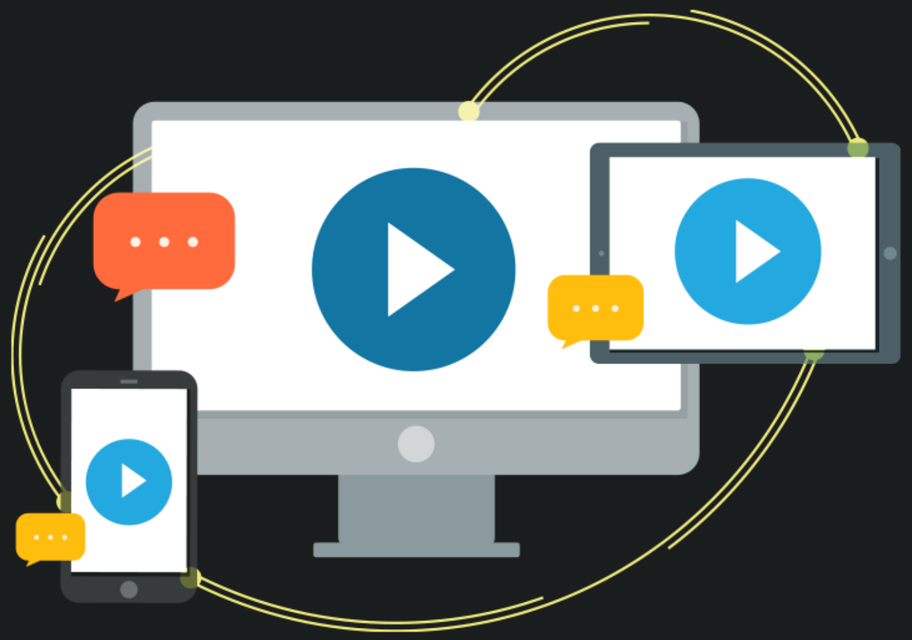
Network Project Team Project with NS-3

Team 11
전영훈
정명희
조재훈
주재현
최승규
한지명

Contents

Introduction of our Project

Real-time Transport Protocol Video Streaming



Model

Real-time Transport Protocol

Original RTP (Real-time Transport Protocol)

RTP

Ethernet (18 바이트)	IP (20 바이트)	UDP (8 바이트)	RTP (12 바이트)	Datagram (음성 / 영상)

RTP Packet Header

Offsets	Octet	0								1								2								3							
Octet	Bit [a]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version			P	X		CC		M				PT				Sequence number															
4	32	Timestamp																															
8	64	SSRC identifier																															
12	96	CSRC identifiers ...																															
12+4×CC	96+32×CC	Profile-specific extension header ID																Extension header length															
16+4×CC	128+32×CC	Extension header ...																															

Our RTP

RTP

Ethernet (18 바이트)	IP (20 바이트)	UDP (8 바이트)	RTP (8 바이트)	Datagram (음성 / 영상)

RTP Packet Header

Offsets	Octets	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Sequence number																															
4	32	Last Sequence number for current frame																															

Sequence Number 크기 32 비트로 확장
프레임의 마지막 Sequence number 추가
간단하게 구성해 헤더 크기 8바이트로 축소

Our RTP

rtp-header.h

```
private:
  uint32_t m_sequence; //!< Sequence number
  uint32_t m_lastFrameSequence; //!< Last sequence number of the frame
```

rtp-header.cc

```
RtpHeader::RtpHeader ()
  : m_sequence (0) // default value is 0 : no retransmition required.
  , m_lastFrameSequence (0)
{
}
```

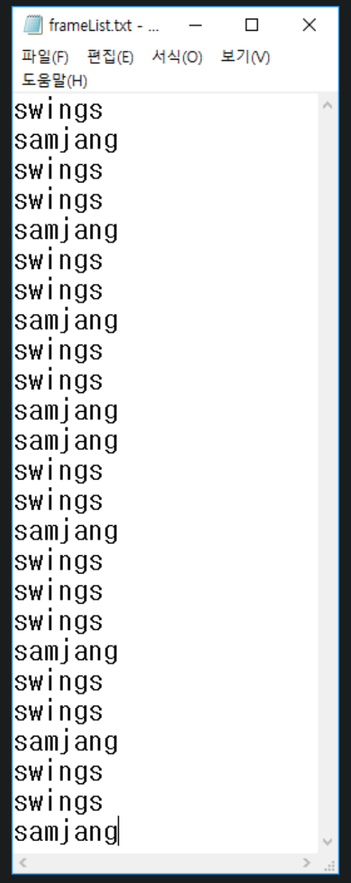
Video Streaming

reliable streaming (server side)

Input -〉

목적: 서버-〉클라 영상 전달

FrameList



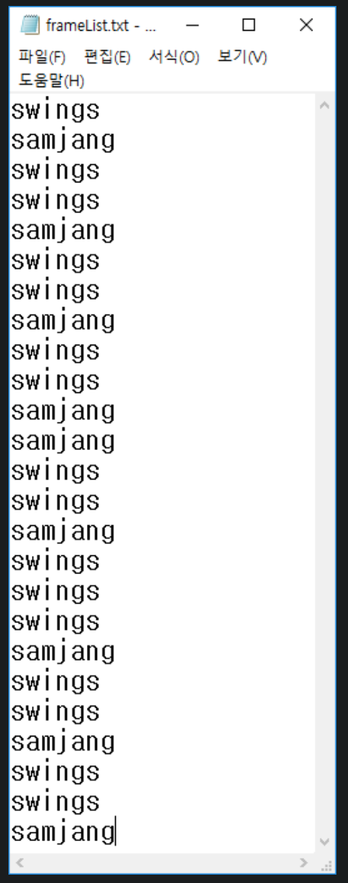
각각 1/2/3/4/5/6 ver 존재

reliable streaming (server side)

Input –〉

목적: 서버–〉클라 영상 전달

FrameList



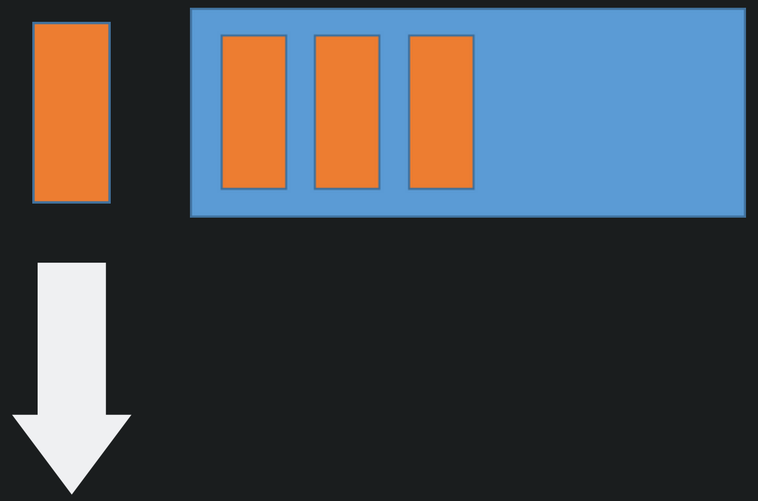
저는
1초짜리
영상이에욧

각각 1/2/3/4/5/6 ver 존재

reliable streaming (client side)

BLUE: BUFFER

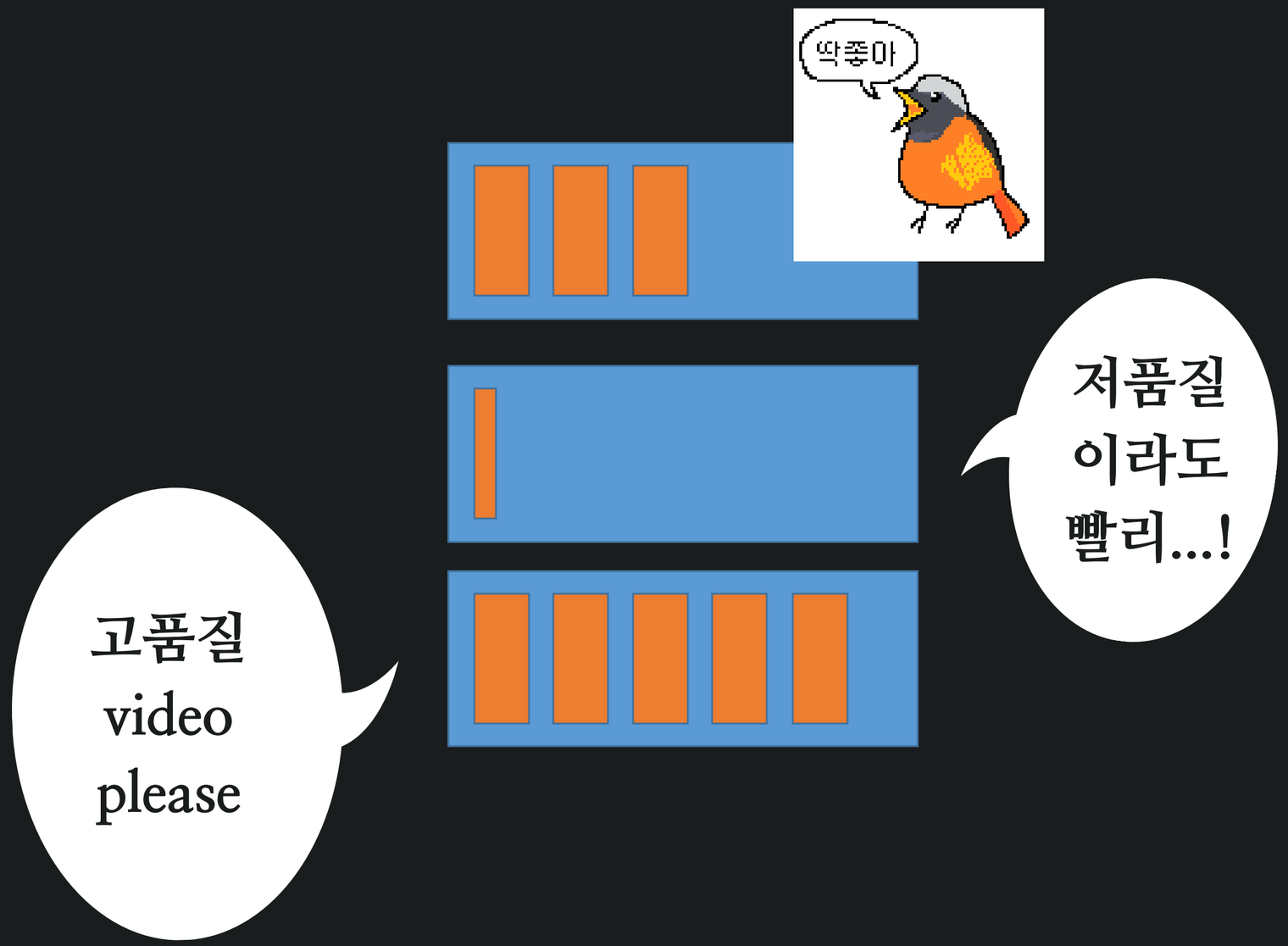
ORANGE: 25 frame



Save them as 25 PNG files
(per second)

reliable streaming (client side)

BLUE: BUFFER
ORANGE: 25 frame



reliable streaming
(rtp -specific)

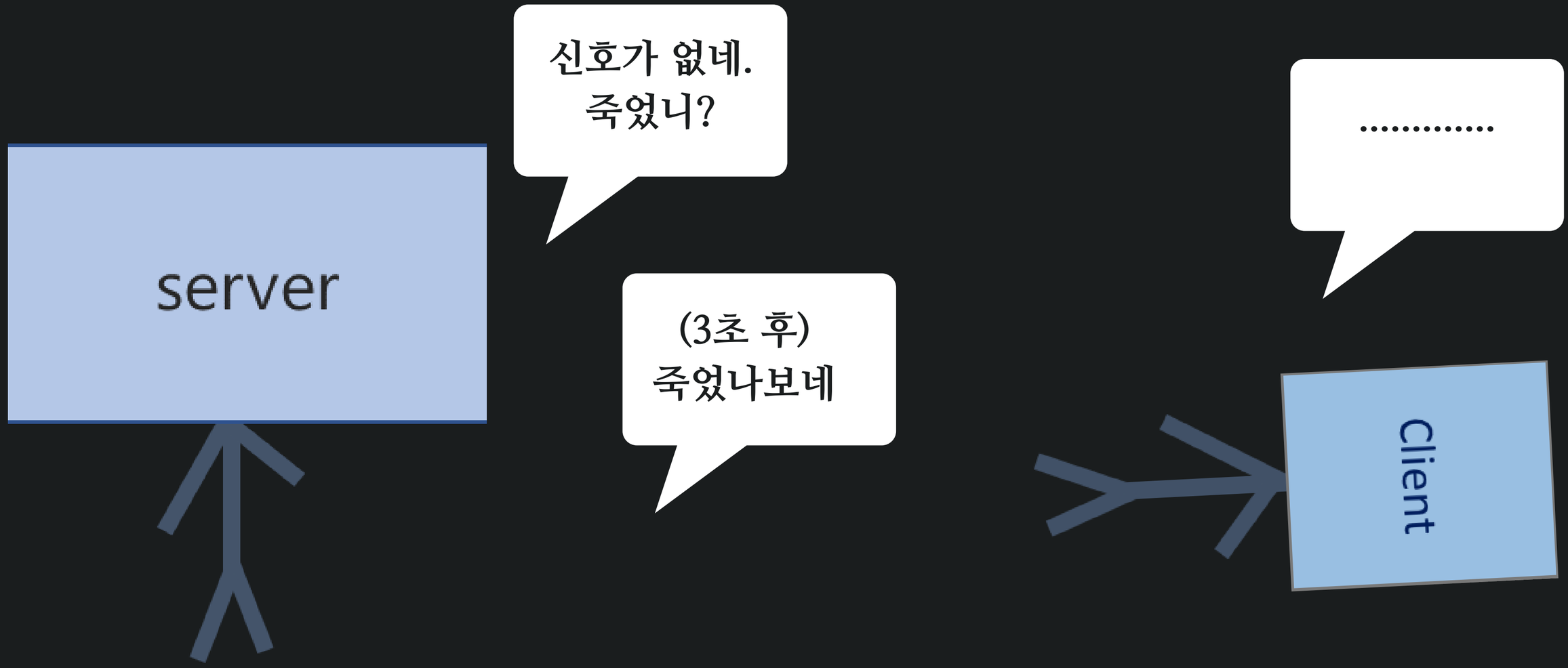
reliable streaming (rtp -specific)

reliable streaming (rtp -specific)

이 Frame 범위
(1~)8
해당 SeqNum
7

reliable streaming
(rtp -specific)

reliable streaming



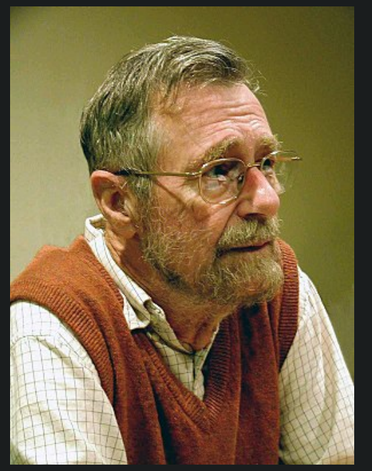
Scenario

Routing Algorithm Based on Dijkstra

Dijkstra

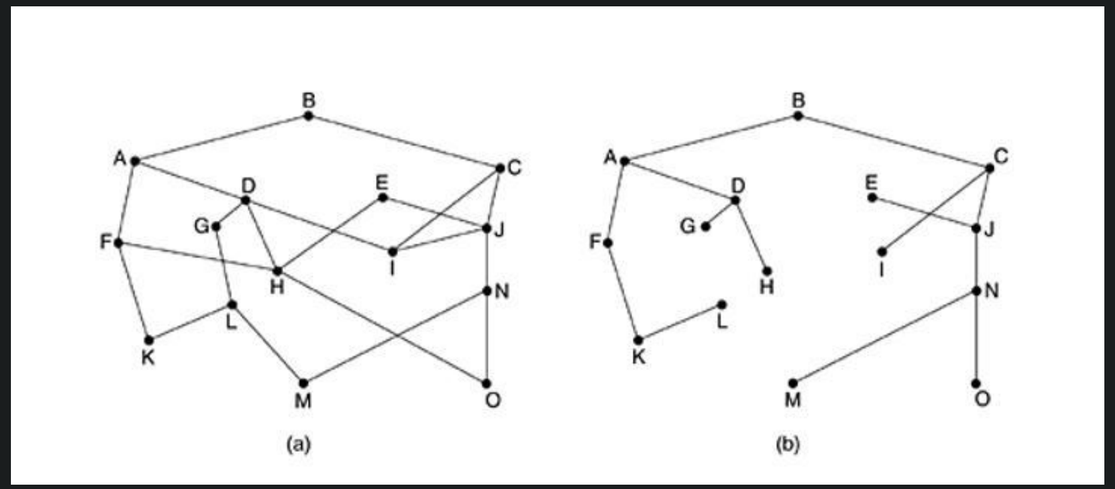
다이나믹 프로그래밍을 활용한 대표적 최단 경로 탐색 알고리즘

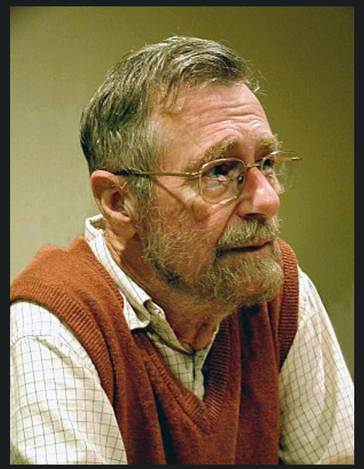
Non-adaptive (static) algorithm
  Shortest path routing
  Flooding: selective flooding
  Flow-based routing
Adaptive (dynamic) algorithm
  Distance vector routing
  Link state routing
  Hierarchical routing



Dijkstra

다이나믹 프로그래밍을 활용한 대표적 최단 경로 탐색 알고리즘



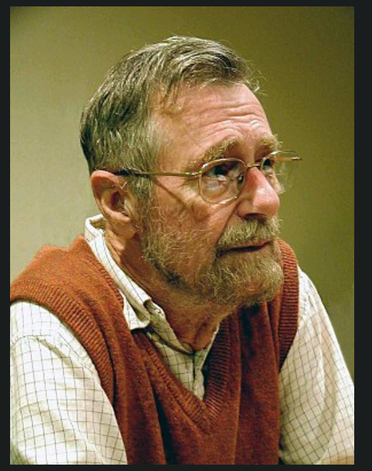


Dijkstra - Input & Output

다이나믹 프로그래밍을 활용한 대표적 최단 경로 탐색 알고리즘

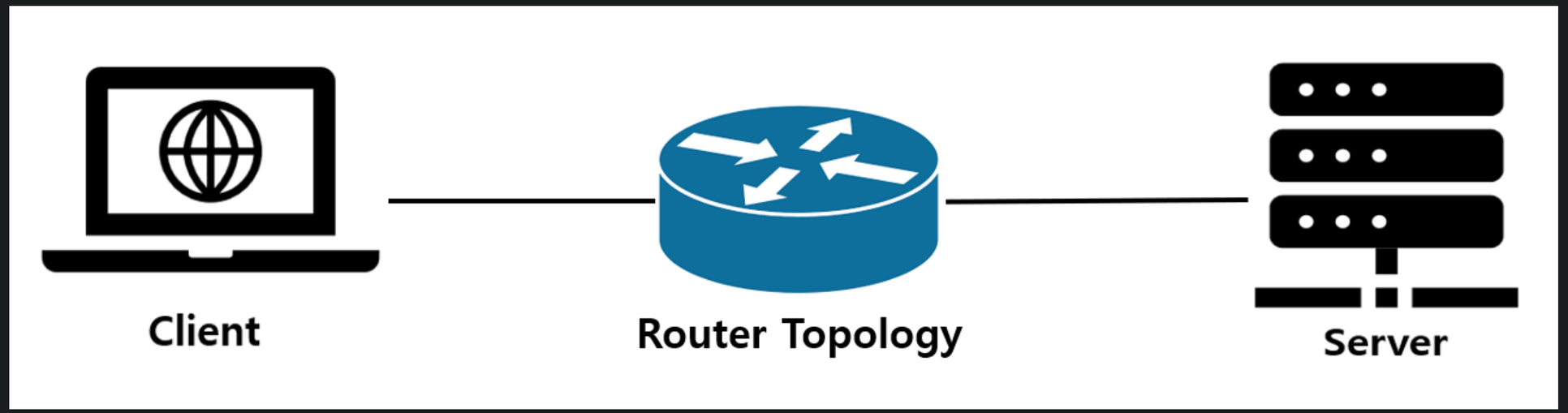
```
1    5 5
2    4 5 4
3    1 2 3
4    2 3 1
5    3 4 2
6    2 4 1
7
```

```
1    0 1 1
2    1 2 3
3    2 4 1
4    4 5 4
5    5 6 1
6
```



Network Topology - Wi-Fi & P2P

Network Topology



Network Topology - Wi-Fi

```
for(uint32_t m=0; m<nAp; m++)
{
  ApplicationContainer serverApps = videoServer.Install (wifiApNode.Get (m));
  serverApps.Start (Seconds (0.0));
  serverApps.Stop (Seconds (100.0));
}
for(uint32_t k=0; k<bridgeNum; k++)
{
  VideoStreamClientHelper videoClient (apInterfaces.GetAddress (route[k][1]), 5000);
  ApplicationContainer clientApps =
  videoClient.Install (wifiStaNodes.Get (route[k][0]));
  clientApps.Start (Seconds (0.5));
  clientApps.Stop (Seconds (100.0));
}
```

Network Topology - P2P

```
// dif delay
std::vector<PointToPointHelper> p2pvector(bridgeNum);
for(uint i=0; i<bridgeNum; i++){
   p2pvector[i].SetDeviceAttribute("DataRate", StringValue("100Mbps"));
   delaytime = std::to_string(route[i][2])+"ms";
   p2pvector[i].SetChannelAttribute("Delay", StringValue(delaytime));
}
```

Network Topology - P2P

```
Ipv4AddressHelper address;
std::string address_value = "10.1.1.0";
for(uint i=0; i<bridgenum; i++){
    int num = route[i][1];
    address_value = "10.1."+std::to_string(num)+".0";
    address.SetBase(Ipv4Address(address_value.c_str()), "255.255.255.0");
    address.Assign(netvector[i]);
}

std::vector<Ipv4InterfaceContainer> interfacevector(bridgenum);
for(uint i=0; i<bridgenum; i++){
    interfacevector[i] = address.Assign(netvector[i]);
}
```

Network Topology - P2P

```
for(uint k=0; k<bridgenum; k++){
    VideoStreamClientHelper videoClient (interfacevector[k].GetAddress (0), 5000);
    ApplicationContainer clientApps = videoClient.Install (nodes.Get (route[k][0]));
    clientApps.Start (Seconds (0.0));
    clientApps.Stop (Seconds (100.0));
}

VideoStreamServerHelper videoServer (5000);
videoServer.SetAttribute ("MaxPacketSize", UintegerValue (1400));
videoServer.SetAttribute ("FrameFile", StringValue ("./scratch/videoStreamer/frameList.txt"));
// videoServer.SetAttribute ("FrameSize", UintegerValue (4096));

for(uint k=0; k<nodeNum+2; k++){
    ApplicationContainer serverApps = videoServer.Install(nodes.Get(k));
    serverApps.Start(Seconds(0.0));
    serverApps.Stop(Seconds(100.0));
}
    serverApps.Stop(Seconds(100.0));
}
```

Execution Result

Execution Result - in Wi-Fi

```
root@2e37b0ed5587:~/ns-allinone-3.29/ns-3.29# ls scratch/videoStreamer/videos
root@2e37b0ed5587:~/ns-allinone-3.29/ns-3.29#
```

```
root@2e37b0ed5587:~/ns-allinone-3.29/ns-3.29# ls scratch/videoStreamer/videos
0.0.png   0.11.png  0.14.png  0.17.png  0.2.png   0.22.png  0.3.png  0.6.png  0.9.png
0.1.png   0.12.png  0.15.png  0.18.png  0.20.png  0.23.png  0.4.png  0.7.png
0.10.png  0.13.png  0.16.png  0.19.png  0.21.png  0.24.png  0.5.png  0.8.png
```

Execution Result - in P2P

```
root@2e37b0ed5587:~/ns-allinone-3.29/ns-3.29# ls scratch/videoStreamer/videos
root@2e37b0ed5587:~/ns-allinone-3.29/ns-3.29#
```

```
root@2e37b0ed5587:~/ns-allinone-3.29/ns-3.29# ls scratch/videoStreamer/videos
0.0.png   0.17.png  0.3.png  1.10.png  1.19.png  1.5.png   2.12.png  2.20.png  2.7.png   3.14.png  3.22.png  3.9.png
0.1.png   0.18.png  0.4.png  1.11.png  1.2.png   1.6.png   2.13.png  2.21.png  2.8.png   3.15.png  3.23.png
0.10.png  0.19.png  0.5.png  1.12.png  1.20.png  1.7.png   2.14.png  2.22.png  2.9.png   3.16.png  3.24.png
0.11.png  0.2.png   0.6.png  1.13.png  1.21.png  1.8.png   2.15.png  2.23.png  3.0.png   3.17.png  3.3.png
0.12.png  0.20.png  0.7.png  1.14.png  1.22.png  1.9.png   2.16.png  2.24.png  3.1.png   3.18.png  3.4.png
0.13.png  0.21.png  0.8.png  1.15.png  1.23.png  2.0.png   2.17.png  2.3.png   3.10.png  3.19.png  3.5.png
0.14.png  0.22.png  0.9.png  1.16.png  1.24.png  2.1.png   2.18.png  2.4.png   3.11.png  3.2.png   3.6.png
0.15.png  0.23.png  1.0.png  1.17.png  1.3.png   2.10.png  2.19.png  2.5.png   3.12.png  3.20.png  3.7.png
0.16.png  0.24.png  1.1.png  1.18.png  1.4.png   2.11.png  2.2.png   2.6.png   3.13.png  3.21.png  3.8.png
```

Thank You