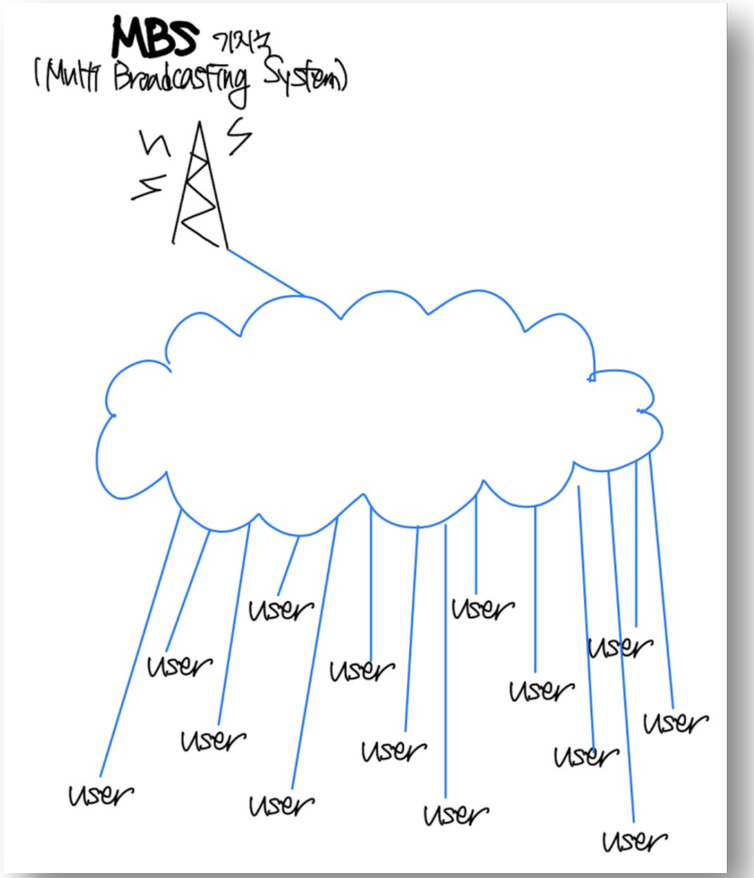
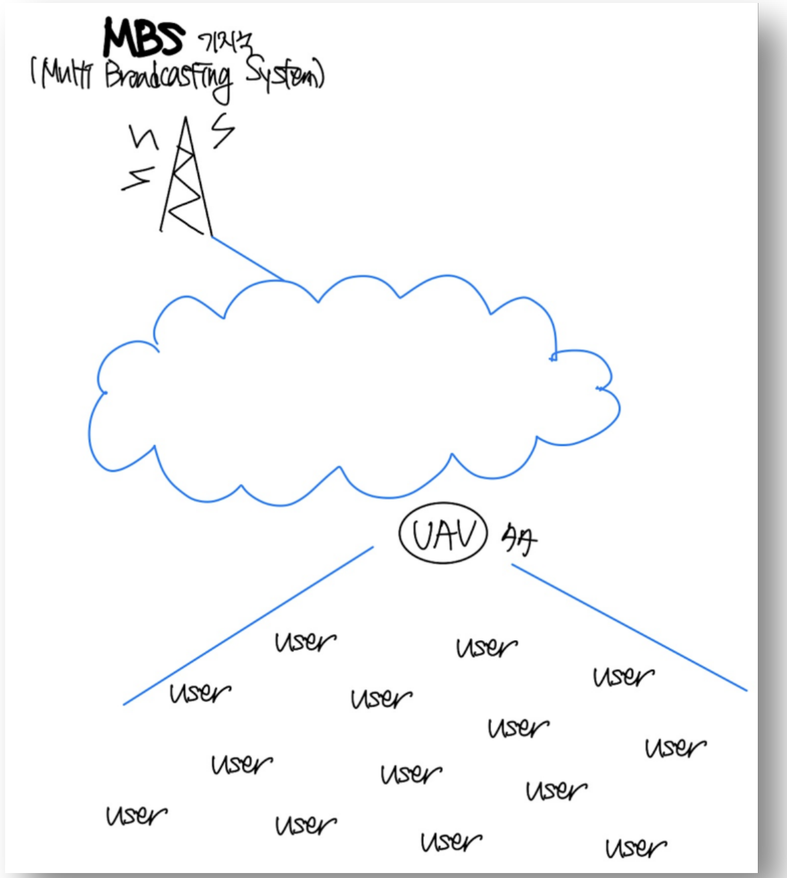
ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

MBS only(user와 직접 통신)



MBS + UAV (MBS↔UAV, UAV↔user)



ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

1. 연구 배경
- 드론(UAV)는 위기 상황이나 인구 밀집도가 높은 곳에서 빠르게 통신 서비스를 제공할 수 있음
- 기존 기술은 사용자가 어디에 있는지, 어떻게 이동하는지 고려하지 않고 있어, 드론의 위치와 캐시(콘텐츠 저장)를 똑똑하게 결정하는 기술이 필요함

2. 연구 설계
(1) 강화학습 활용 UAV의 위치와 캐시 전략 최적화
→ UAV는 사용자 근처로 이동하고, 인기 콘텐츠를 미리 저장해 빠르게 응답
[사용자 ↔ UAV ↔ 기지국(MBS)]
(2) DRL 학습 요소
- 상태(State): 사용자 위치, 요청 정보 등
- 행동(Action): UAV의 이동 위치, 어떤 콘텐츠를 저장할지
- 보상(Reward): 사용자의 체감 품질(QoE), 지연 시간 등

3. 결론
- DRL을 이용하면 사용자 밀집 지역으로 UAV가 자동 이동
- 기존 방식보다 지연 시간 감소, 사용자 만족도(QoE) 향상 확인

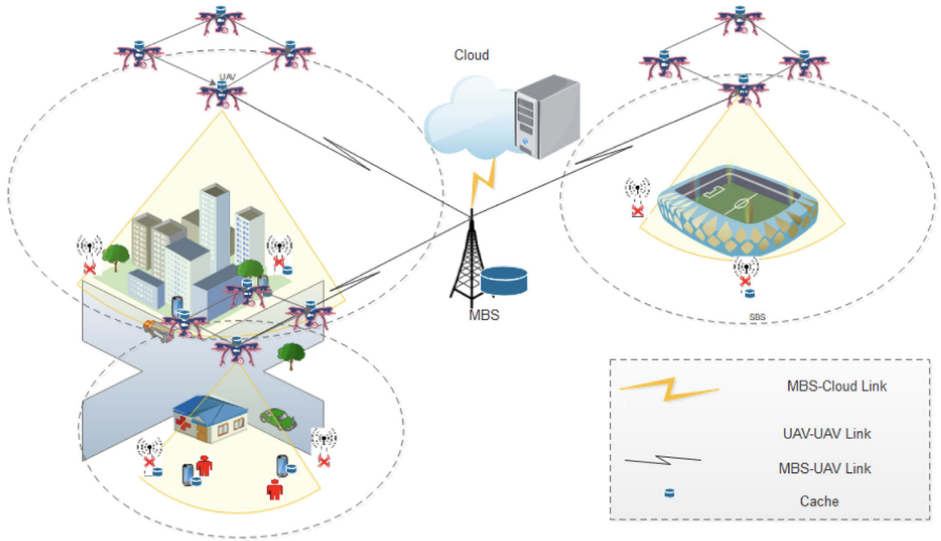


Fig. 1. Network architecture.

ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

uav1.cc

```cpp
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/netanim-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/applications-module.h"  // UDP Echo 애플리케이션을 위한 헤더

#define CONTENT_NUM 10 // 콘텐츠 수
#define USER_NUM 20 // user(이동 노드) 수
#define UAV_NUM 1 // UAV 수
#define MBS_Tx_POWER 30 // MBS 송신 전력(dBm)
#define UAV_Tx_POWER 20 // UAV 송신 전력(dBm)
#define UAV_HEIGHT 100 // UAV 높이(m)
#define AREA 2000 // simulation 영역 크기(2000m * 2000m)
#define RUNTIME 200 // simulation time

using namespace ns3;

NS_LOG_COMPONENT_DEFINE("UAVNetworkExample");

/* 전체적인 구조

[Cloud]
    |
  (Backhaul)
    |
[MBS] ---WiFi(MBS-Connection)---> [UAV #1]
                                      |
                                WiFi(UAV-Connection)
                                      |
                                [ Users (랜덤 이동) ]
*/
```

```cpp
/*
가정 :
USER와 UAV, UAV와 UAV, MBS와 UAV는 WiFi 연결로 통신
MBS와 Backbone은 P2P 연결로 통신(시뮬레이션에는 미구현)
*/

int main(int argc, char* argv[])
{
    // UDP Echo 앱 실행 로그 출력
    LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
    LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
    Time::SetResolution(Time::NS);

    // user(이동 노드) 생성
    NodeContainer userNodes;
    userNodes.Create(USER_NUM);

    // uav 생성
    NodeContainer uavNodes;
    uavNodes.Create(UAV_NUM);

    // MBS 생성
    NodeContainer mbsNodes;
    mbsNodes.Create(1);

    // 더미 노드 생성
        NodeContainer dummy;
        dummy.Create(2);
        Ptr<ListPositionAllocator> cornerPos = ns3::CreateObject<ListPositionAllocator> ();
            cornerPos->Add (Vector (0, 0, 0));
            cornerPos->Add (Vector (AREA, AREA, 0));
            MobilityHelper mobilityC;
            mobilityC.SetPositionAllocator (cornerPos);
            mobilityC.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
            mobilityC.Install (dummy);
```

ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

```
73      // UAV 배치
74      MobilityHelper uavMob;
75      // UAV 좌표 설정
76      Ptr<ListPositionAllocator> positionAlloc = ns3::CreateObject<ListPositionAllocator> ();
77      positionAlloc->Add (Vector(AREA*0.5, AREA*0.5, UAV_HEIGHT));
78
79      uavMob.SetPositionAllocator(positionAlloc);
80      uavMob.SetMobilityModel("ns3::ConstantPositionMobilityModel");
81      uavMob.Install(uavNodes);
82
83      // MBS 배치
84      MobilityHelper mbsMob;
85      Ptr<ListPositionAllocator> mbsPositionAlloc = ns3::CreateObject<ListPositionAllocator> ();
86      mbsPositionAlloc->Add(Vector(0,0,0));
87
88      mbsMob.SetPositionAllocator(mbsPositionAlloc);
89      mbsMob.SetMobilityModel("ns3::ConstantPositionMobilityModel");
90      mbsMob.Install(mbsNodes);
91
92      // user 배치
93      MobilityHelper userMob;
94      ObjectFactory points;
95      points.SetTypeId("ns3::RandomRectanglePositionAllocator");
96      std::string xStr = std::string("ns3::UniformRandomVariable[Min=1000.0|Max=") + std::to_string(AREA) + "]";
97      points.Set("X", StringValue(xStr));
98      std::string yStr = std::string("ns3::UniformRandomVariable[Min=1000.0|Max=") + std::to_string(AREA) + "]";
99      points.Set("Y", StringValue(yStr));
100     Ptr<PositionAllocator> waypos = points.Create() -> GetObject<PositionAllocator>();
101
102     userMob.SetMobilityModel("ns3::RandomWaypointMobilityModel",
103     "Speed",StringValue("ns3::UniformRandomVariable[Min=0.0|Max=20.0]"), // 랜덤으로 속도(0~20m/s) 이동
104     "Pause", StringValue("ns3::UniformRandomVariable[Min=0.0|Max=2.0]"), // 목표지점에 도착하면 0~2초 쉬었다가 다시 이동
105     "PositionAllocator",PointerValue(waypos)
106     );
107     userMob.SetPositionAllocator(waypos);
108     userMob.Install(userNodes);
```

ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

```
110     // UAV와 users를 연결하는 WiFi
111     YansWifiChannelHelper uavChannel = YansWifiChannelHelper::Default();
112     uavChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
113     uavChannel.AddPropagationLoss("ns3::RangePropagationLossModel",
114                                   "MaxRange", DoubleValue(1000000.0));  // 1000km로 설정
115
116     YansWifiPhyHelper uavPhy;
117     uavPhy.SetChannel(uavChannel.Create());
118     uavPhy.Set("TxPowerStart", DoubleValue(UAV_Tx_POWER));
119     uavPhy.Set("TxPowerEnd", DoubleValue(UAV_Tx_POWER));
120
121     WifiHelper uavWifi;
122     uavWifi.SetRemoteStationManager("ns3::ConstantRateWifiManager");
123
124     WifiMacHelper uavMac;
125     Ssid ssid = Ssid("UAV-Connection");
126     uavMac.SetType("ns3::ApWifiMac",
127     "Ssid",SsidValue(ssid));
128     NetDeviceContainer uavDevices;
129     uavDevices = uavWifi.Install(uavPhy,uavMac,uavNodes);
130
131     WifiMacHelper userMac;
132     userMac.SetType("ns3::StaWifiMac",
133     "Ssid",SsidValue(ssid),
134     "ActiveProbing",BooleanValue(false));
135     NetDeviceContainer userDevices;
136     userDevices = uavWifi.Install(uavPhy,userMac,userNodes);
```

```
139     // MBS와 UAV를 연결하는 WiFi
140     YansWifiChannelHelper mbsChannel = YansWifiChannelHelper::Default();
141     mbsChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
142     mbsChannel.AddPropagationLoss("ns3::RangePropagationLossModel",
143                                   "MaxRange", DoubleValue(1000000.0));  // 1000km로 설정
144
145     YansWifiPhyHelper mbsPhy;
146     mbsPhy.SetChannel(mbsChannel.Create());
147     mbsPhy.Set("TxPowerStart", DoubleValue(MBS_Tx_POWER));
148     mbsPhy.Set("TxPowerEnd", DoubleValue(MBS_Tx_POWER));
149
150     WifiHelper mbsWifi;
151     Ssid ssid2 = Ssid("MBS-Connection");
152     mbsWifi.SetRemoteStationManager("ns3::ConstantRateWifiManager");
153
154     WifiMacHelper mbsMac;
155     mbsMac.SetType("ns3::ApWifiMac",
156     "Ssid",SsidValue(ssid2));
157     NetDeviceContainer mbsDevices;
158     mbsDevices = mbsWifi.Install(mbsPhy,mbsMac,mbsNodes);
159
160     WifiMacHelper uavMac2;
161     uavMac2.SetType("ns3::StaWifiMac",
162     "Ssid",SsidValue(ssid2),
163     "ActiveProbing",BooleanValue(false));
164     NetDeviceContainer uavDevices2;
165     uavDevices2 = mbsWifi.Install(mbsPhy,uavMac2,uavNodes);
```

ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

```
167     // Internet 설치
168     InternetStackHelper stack;
169     stack.Install(uavNodes);
170     stack.Install(mbsNodes);
171     stack.Install(userNodes);
172
173     //PCAP 파일 생성 설정
174     uavPhy.EnablePcap("uav-node", uavDevices.Get(0));
175     uavPhy.EnablePcap("user-node", userDevices.Get(0));
176     mbsPhy.EnablePcap("mbs-node", mbsDevices.Get(0));
177
178     // IP 주소 부여
179     Ipv4AddressHelper address;
180
181     // User-UAV 네트워크 주소 할당 (10.1.1.0/24)
182     address.SetBase("10.1.1.0","255.255.255.0");
183     Ipv4InterfaceContainer userInterfaces = address.Assign(userDevices);
184     Ipv4InterfaceContainer uavInterfaces = address.Assign(uavDevices);
185
186     // MBS-UAV 네트워크 주소 할당 (10.1.2.0/24)
187     address.SetBase("10.1.2.0","255.255.255.0");
188     Ipv4InterfaceContainer mbsInterfaces = address.Assign(mbsDevices);
189     Ipv4InterfaceContainer uavInterfaces2 = address.Assign(uavDevices2);
```

```
191     // 정적 라우팅 설정
192     Ipv4StaticRoutingHelper staticRouting;
193
194     // UAV 노드들의 라우팅 설정
195     for (uint32_t i = 0; i < uavNodes.GetN(); i++) {
196         Ptr<Ipv4> ipv4 = uavNodes.Get(i)->GetObject<Ipv4>();
197         Ptr<Ipv4StaticRouting> uavStaticRouting = staticRouting.GetStaticRouting(ipv4);
198
199         // User 네트워크로 가는 경로
200         uavStaticRouting->AddNetworkRouteTo(Ipv4Address("10.1.1.0"), Ipv4Mask("255.255.255.0"), 1);
201         // MBS 네트워크로 가는 경로
202         uavStaticRouting->AddNetworkRouteTo(Ipv4Address("10.1.2.0"), Ipv4Mask("255.255.255.0"), 2);
203     }
204
205     // User 노드들의 라우팅 설정
206     for (uint32_t i = 0; i < userNodes.GetN(); i++) {
207         Ptr<Ipv4> ipv4 = userNodes.Get(i)->GetObject<Ipv4>();
208         Ptr<Ipv4StaticRouting> userStaticRouting = staticRouting.GetStaticRouting(ipv4);
209         userStaticRouting->AddNetworkRouteTo(Ipv4Address("10.1.1.0"), Ipv4Mask("255.255.255.0"), 1);
210     }
211
212     // MBS 노드의 라우팅 설정
213     Ptr<Ipv4> ipv4 = mbsNodes.Get(0)->GetObject<Ipv4>();
214     Ptr<Ipv4StaticRouting> mbsStaticRouting = staticRouting.GetStaticRouting(ipv4);
215     mbsStaticRouting->AddNetworkRouteTo(Ipv4Address("10.1.2.0"), Ipv4Mask("255.255.255.0"), 1);
```

ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

```
    // UAV to User application 설치
    uint16_t port = 9;

    NS_LOG_INFO("Creating UDP Echo Server on UAV nodes");
    UdpEchoServerHelper echoServer(port);
    ApplicationContainer serverApps = echoServer.Install(uavNodes);
    serverApps.Start(Seconds(1.0));
    serverApps.Stop(Seconds(RUNTIME-1));

    NS_LOG_INFO("Creating UDP Echo Client on User nodes");
    UdpEchoClientHelper echoClient(uavInterfaces.GetAddress(0), port);
    echoClient.SetAttribute("MaxPackets", UintegerValue(100));
    echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
    echoClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer clientApps;
    for (uint32_t i = 0; i < userNodes.GetN(); i++) {
        NS_LOG_INFO("Installing client on user node " << i);
        clientApps.Add(echoClient.Install(userNodes.Get(i)));
    }
    clientApps.Start(Seconds(2.0));
    clientApps.Stop(Seconds(RUNTIME-1));
```

```
    // MBS to UAV application 설치
    uint16_t mbs_port = 10;

    NS_LOG_INFO("Creating UDP Echo Server on MBS node");
    UdpEchoServerHelper mbsServer(mbs_port);
    ApplicationContainer mbsServerApps = mbsServer.Install(mbsNodes.Get(0));
    mbsServerApps.Start(Seconds(1.0));
    mbsServerApps.Stop(Seconds(RUNTIME-1));

    NS_LOG_INFO("Creating UDP Echo Client on UAV nodes for MBS communication");
    UdpEchoClientHelper mbsClient(mbsInterfaces.GetAddress(0), mbs_port);
    mbsClient.SetAttribute("MaxPackets", UintegerValue(10));
    mbsClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
    mbsClient.SetAttribute("PacketSize", UintegerValue(1024));

    ApplicationContainer uavClientApps;
    for (uint32_t i = 0; i < uavNodes.GetN(); i++) {
        NS_LOG_INFO("Installing client on UAV node " << i);
        uavClientApps.Add(mbsClient.Install(uavNodes.Get(i)));
    }
    uavClientApps.Start(Seconds(2.0));
    uavClientApps.Stop(Seconds(RUNTIME-1));
```

ns-3

Deep Reinforcement Learning-Based Mobility-Aware UAV Content Caching and Placement in Mobile Edge Networks

```
263     // NetAnim 설정
264     AnimationInterface anim("uav1.xml");
265     anim.SetMobilityPollInterval(Seconds(1.0));
266
267     for(uint32_t i=0; i<uavNodes.GetN(); i++){
268         anim.UpdateNodeDescription(uavNodes.Get(i),"UAV");
269         anim.UpdateNodeColor(uavNodes.Get(i),255,0,0);
270         anim.UpdateNodeSize(uavNodes.Get(i),100,100);
271     }
272     for(uint32_t i=0; i<userNodes.GetN(); i++){
273         anim.UpdateNodeDescription(userNodes.Get(i),"User");
274         anim.UpdateNodeColor(userNodes.Get(i),0,255,0);
275         anim.UpdateNodeSize(userNodes.Get(i),40,40);
276     }
277     anim.UpdateNodeDescription(mbsNodes.Get(0),"MBS");
278     anim.UpdateNodeColor(mbsNodes.Get(0),0,0,255);
279     anim.UpdateNodeSize(mbsNodes.Get(0),100,100);
280
281     // simulator 실행
282     Simulator::Stop(Seconds(RUNTIME));
283     Simulator::Run();
284     Simulator::Destroy();
285
286     return 0;
287 }
```

감사합니다 :)