Implementation of ASRAN in NS3

Project Update 1

Presented By :
1705005
Mashiat Mustaq

ASRAN Algorithm

Identifies the cause of segment loss and quickly recovers lost throughput

Causes of Segment Loss :

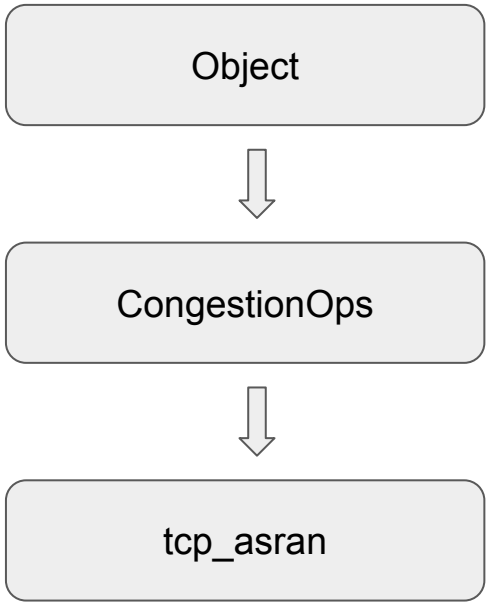
- Congestion Control
- Transient Link Instability

ASRAN Algorithm

- Directory:

  Src / internet / model /

  - Asran.h
  - Asran.cc



ASRAN Algorithm

- Modification from TCPNewReno

```
71
72  private:
73    uint32_t m_cWndCnt {0}; //!< Linear increase counter
74  };
75
76  } // namespace ns3
```

Add another variable :
last_max_cWnd

ASRAN Algorithm

- Modification from TCPNewReno

```
uint32_t
TcpLinuxReno::GetSsThresh (Ptr<const TcpSocketState> state,
                           uint32_t bytesInFlight)
{
  NS_LOG_FUNCTION (this << state << bytesInFlight);

  // In Linux, it is written as:  return max(tp->snd_cwnd >> 1U, 2U);
  return std::max<uint32_t> (2 * state->m_segmentSize, state->m_cWnd / 2);
}
```

ASRAN Algorithm

- Modification from TCPNewReno

```
uint32_t
TcpLinuxReno::GetSsThresh (Ptr<const TcpSocketState> state,
                           uint32_t bytesInFlight)
{
  NS_LOG_FUNCTION (this << state << bytesInFlight);

  // In Linux, it is written as:  return max(tp->snd_cwnd >> 1U, 2U);
  return std::max<uint32_t> (2 * state->m_segmentSize
}
```

Original_ssthresh = /* TCPNewReno one */
Recalculated_thresh = max (last_max_cwnd, original_ssthresh)
Last_max_cwnd = state->m_cwnd
Return recalculated_thresh

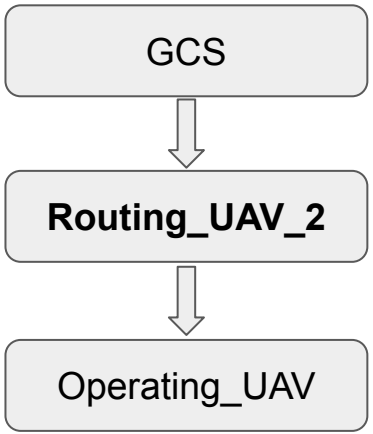
Simulation Environment : UAV Network

- Flying Ad Hoc Network - FANET
- 3D Mobility
- 2D Mobility Model available -> tutorial/third.cc

```
MobilityHelper mobility;

mobility.SetMobilityModel ("ns3::GaussMarkovMobilityModel",
  "Bounds", BoxValue (Box (0, 150000, 0, 150000, 0, 10000)),
  "TimeStep", TimeValue (Seconds (0.5)),
  "Alpha", DoubleValue (0.85),
  "MeanVelocity", StringValue ("ns3::UniformRandomVariable[Min=800|Max=1200]"),
  "MeanDirection", StringValue ("ns3::UniformRandomVariable[Min=0|Max=6.283185307]"),
  "MeanPitch", StringValue ("ns3::UniformRandomVariable[Min=0.05|Max=0.05]"),
  "NormalVelocity", StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=0.0|Bound=0.0]"),
  "NormalDirection", StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=0.2|Bound=0.4]"),
  "NormalPitch", StringValue ("ns3::NormalRandomVariable[Mean=0.0|Variance=0.02|Bound=0.04]"));

mobility.SetPositionAllocator ("ns3::RandomBoxPositionAllocator",
  "X", StringValue ("ns3::UniformRandomVariable[Min=0|Max=150000]"),
  "Y", StringValue ("ns3::UniformRandomVariable[Min=0|Max=150000]"),
  "Z", StringValue ("ns3::UniformRandomVariable[Min=0|Max=10000]"));

mobility.Install (wifiStaNodes);
```

Simulation Environment : Topology

GCS <-> Operating_UAV



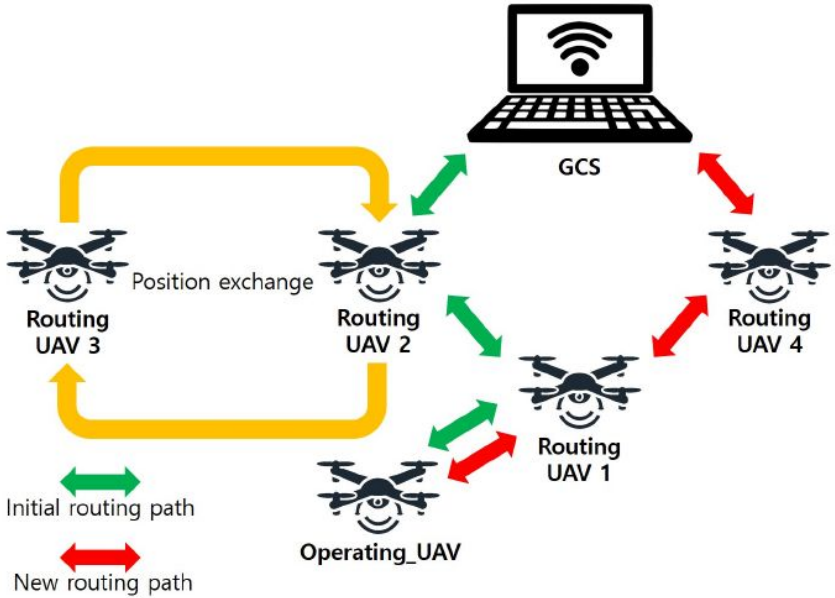
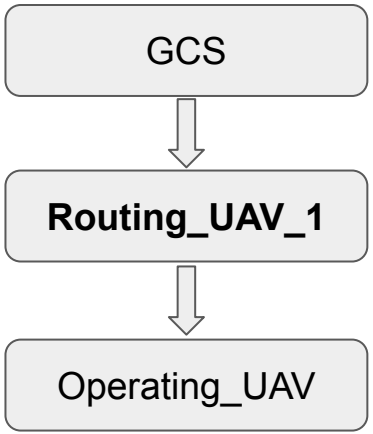


Figure 7. Simulation topology.

Simulation Environment : Topology

GCS <-> Operating_UAV

Routing UAV 2 & 3 exchange positions



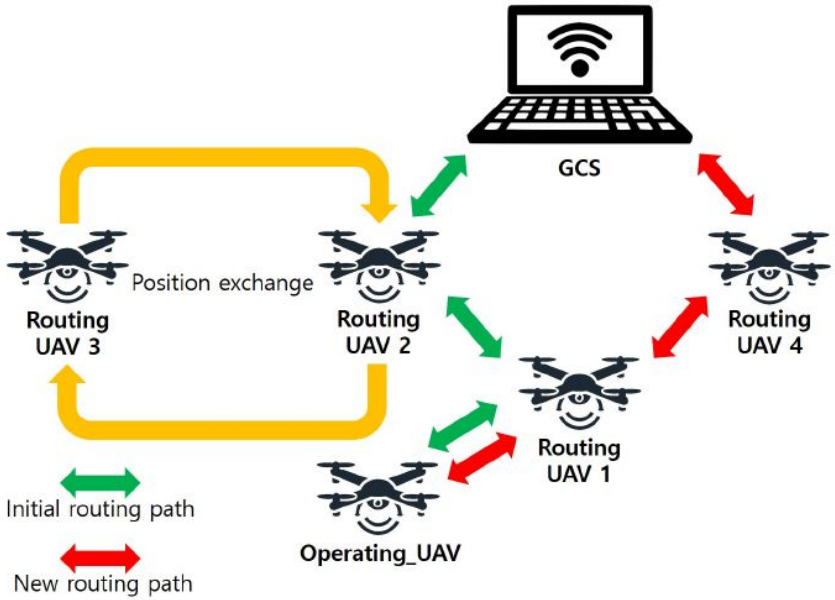


Figure 7. Simulation topology.

Simulation Environment : Topology

Congestion Control

- Add Operating UAV 3
- Load on Routing UAV 1 increases

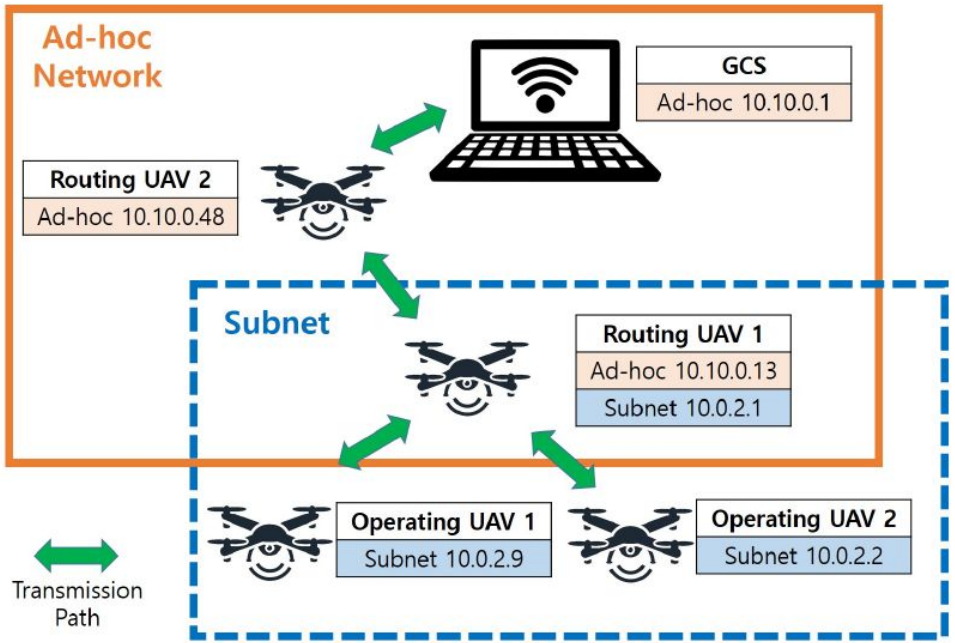


Figure 4. Network topology for experiment.

Simulation Environment : Topology

GCS <-> Operating_UAV

Compare Throughput

- TCPNewReno
- TCPCubic
- ASRAN
- DSR

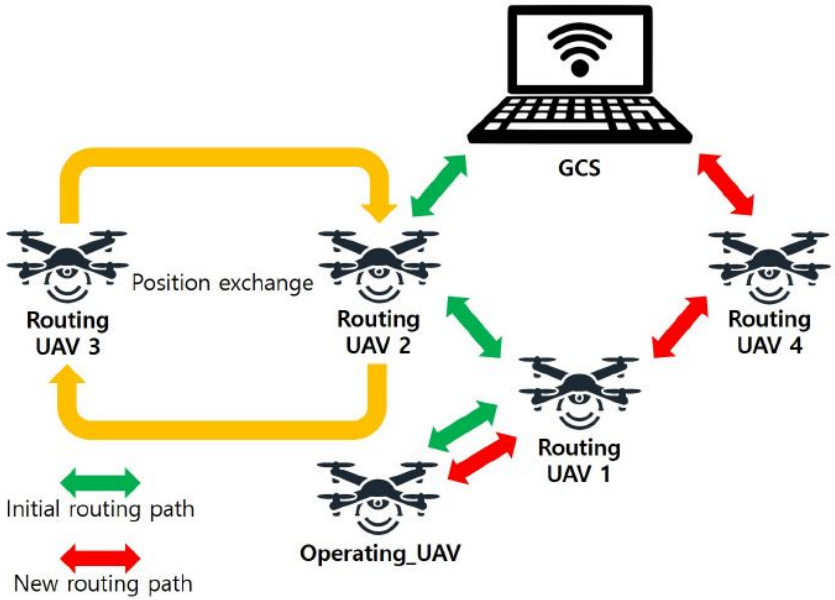


Figure 7. Simulation topology.