Chapter Outline

RflySim Networking Framework and Communication Simulation

Chapter 9 Multi-Vehicle Communication and Intelligent Networking

9.1 Networking Fundamentals

9.2 Hardware and Link Mechanisms

9.3 Software Layers and Protocols

9.4 Channel and Interference Modeling

9.5 RflySim Networking and Simulation

9.6 Protocols and Development Interfaces

9.7 Experiments and Validation

9.8 Exercises and Classroom Discussion

2 Units in This Section

9.5.1 Overall Architecture and Data Categories

Textbook: Section 9.2.1, item 1

9.5.2 Communication Forwarding and Simulation Granularity

Textbook: Section 9.2.1, items 2–3

9.5.1 Overall Architecture and Data Categories: Data Content

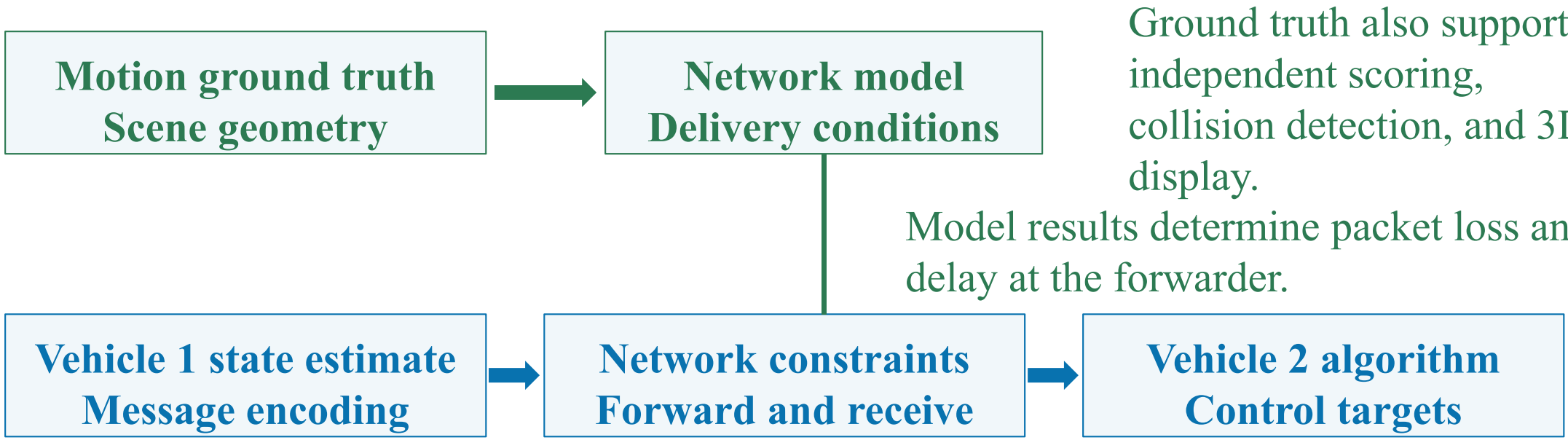
Vehicle State Can Come from Different Sources

Data category	Producer and meaning	Typical users
Simulation ground truth	Exact pose and velocity from the motion model	Scene display, collision detection, channel models, scoring
Observations and estimates	Camera, inertial measurement unit (IMU), and positioning observations plus autopilot state estimates	Perception, cooperative decision-making, state exchange
Control and task information	Targets, tasks, or control commands from the algorithm	Autopilot, actuators, or other nodes

A field named “position” does not identify its source. Check the generating module, coordinate frame, and timestamp.

9.5.1 Overall Architecture and Data Categories: Two Data Paths

Ground Truth Drives the Model; Messages Feed the Algorithm



Vehicle 2 algorithm → vehicle 2 autopilot → vehicle 2 motion. The ground-truth branch computes network conditions and supports independent scoring.

9.5.1 Overall Architecture and Data Categories: Algorithm Input Boundaries

Estimated and True Positions Require Separate Checks

```
if msg.get_type() == "LOCAL_POSITION_NED":
    self.uavPosNED[0] = msg.x
    self.uavPosNED[1] = msg.y
    self.uavPosNED[2] = msg.z
```

Source: Autopilot position estimate via Micro Air Vehicle Link (MAVLink).

Meaning: North-east-down (NED) coordinates in meters. Verify the origin.

Comparing two vehicles: Align origins, axis directions, and times before computing relative position.

Converting uavGlobalPos to a global reference does not give it ground-truth accuracy.

Separate estimation error from communication effects to understand why an algorithm fails.

Chapter Outline

RflySim Networking Framework and Communication Simulation

Chapter 9 Multi-Vehicle Communication and Intelligent Networking

9.1 Networking Fundamentals

9.2 Hardware and Link Mechanisms

9.3 Software Layers and Protocols

9.4 Channel and Interference Modeling

9.5 RflySim Networking and Simulation

9.6 Protocols and Development Interfaces

9.7 Experiments and Validation

9.8 Exercises and Classroom Discussion

2 Units in This Section

9.5.1 Overall Architecture and Data Categories

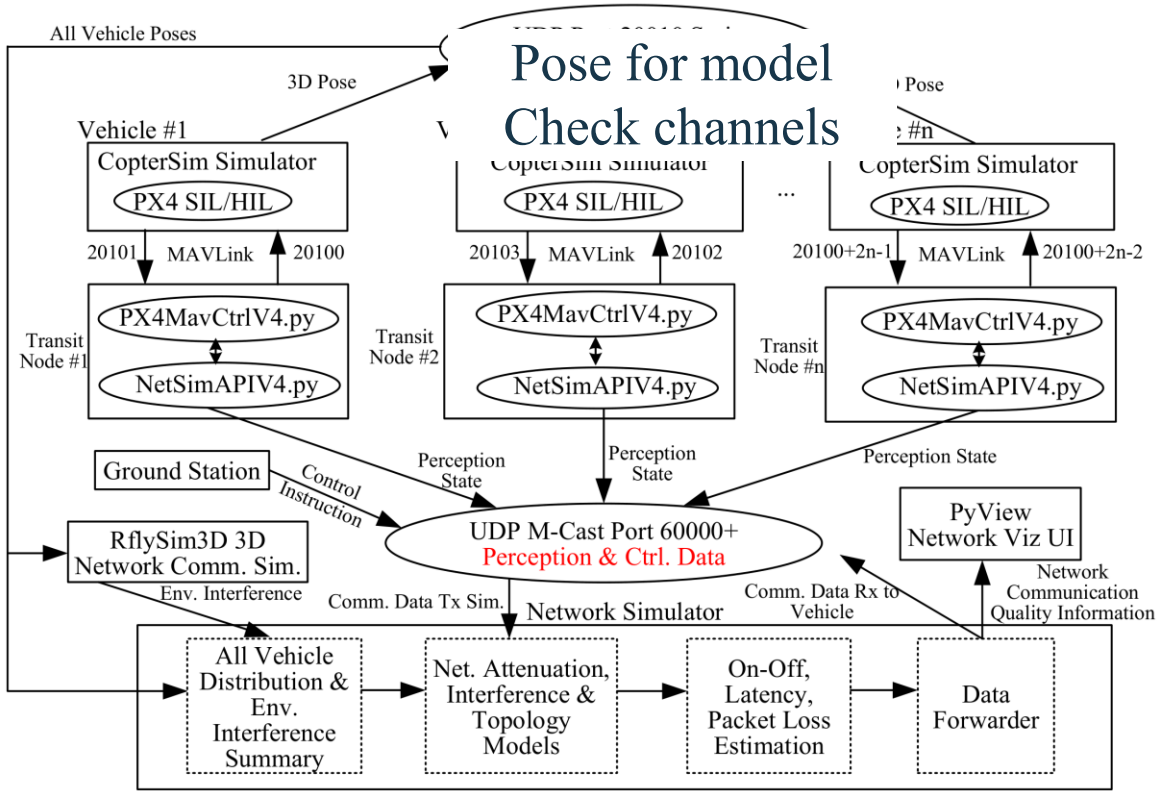
Textbook: Section 9.2.1, item 1

9.5.2 Communication Forwarding and Simulation Granularity

Textbook: Section 9.2.1, items 2–3

9.5.2 Communication Forwarding and Simulation Granularity: Forwarding Architecture

Figure 9.7 Links Motion, Communication, and Network Models



Vehicle side: PX4 and CopterSim exchange state and control. Python reads and forwards state.

Network side: Pose and environment determine conditions for application packet delivery.

The 20100-series interfaces carry autopilot state and control. Identify the separate ground-truth channels.

Figure 9.7 Multi-vehicle communication, networking, and forwarding framework

9.5.2 Communication Forwarding and Simulation Granularity: Direct 20100-Series Links

Direct Connections Bypass the Network Loss Model

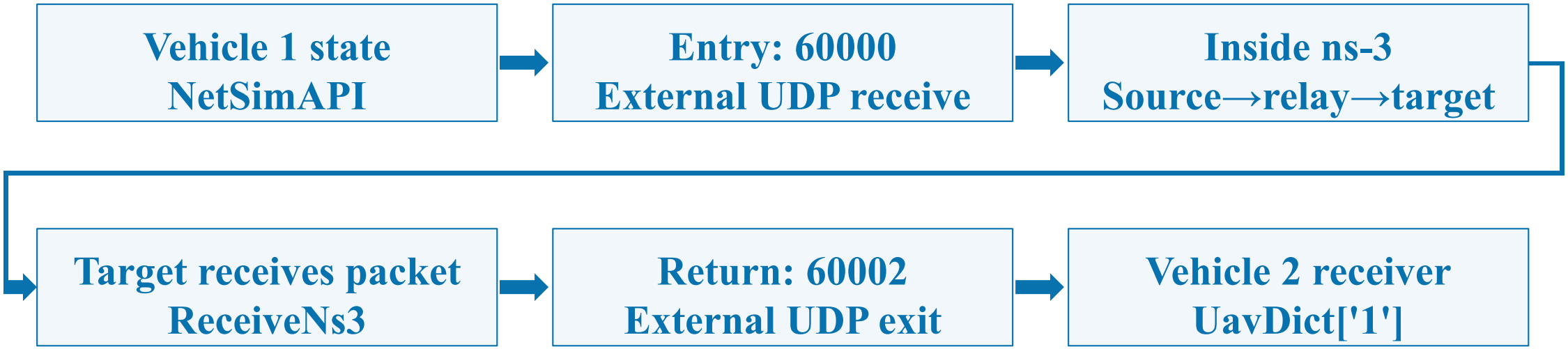
Vehicle ID i	Python sends control to CopterSim	Python listens for returned state
1	20100	20101
2	20102	20103
Default mapping	$20100 + 2(i - 1)$	$20101 + 2(i - 1)$

Idealized direct link: No simulated distance loss, blockage, or packet loss. Host and real-network overhead remains.

Default User Datagram Protocol (UDP) ports are shown. In constrained-network tests, remote state must enter through the constrained receive interface.

9.5.2 Communication Forwarding and Simulation Granularity: 60000-Series Forwarding

Packets Must Traverse the Entry, Simulated Network, and Exit



The connector joins both rows. Return to vehicle 2 occurs only after reception at the target node.

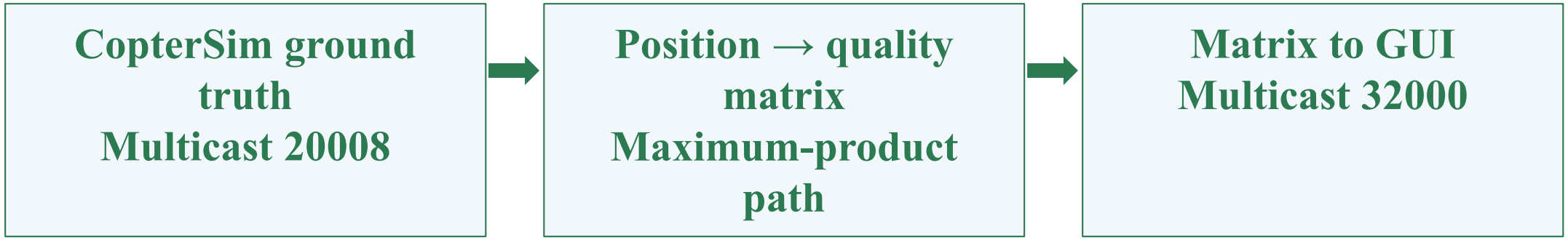
Pose enters RecvSim separately on 20008/20009/20010 to update Mobility, not algorithm messages.

9.5.2 Communication Forwarding and Simulation Granularity: NetSim4Demo Data Paths

The Coarse-Grained Python Example Uses Its Own Ports



The model's success probability controls delivery above. The green path does not directly supply remote state to the cooperative algorithm. GUI: graphical user interface.



9.5.2 Communication Forwarding and Simulation Granularity: Numerical Example

Coarse-Grained Forwarding Converts Path Quality into Delivery

Candidate path	Given link success probabilities	Path success probability
1→2	0.50	0.50
1→3→2	0.80; 0.90	0.72, selected by this model

$$P_{\text{path}} = 0.8 \times 0.9 = 0.72, \qquad \tau = \frac{0.001}{0.72}\ \text{s} \approx 1.389\ \text{ms}$$

P_{path}: dimensionless path success probability; τ: model delay; 0.001 s: baseline time.

Assumptions: Ignore bandwidth threshold; independent link deliveries; teaching probabilities. Aggregate path effects only. No Optimized Link State Routing (OLSR) maintenance or per-hop medium access control (MAC) contention.

9.5.2 Communication Forwarding and Simulation Granularity: Fine-Grained Network Simulator

Figure 9.8 Maps External Vehicles to Virtual Network Nodes

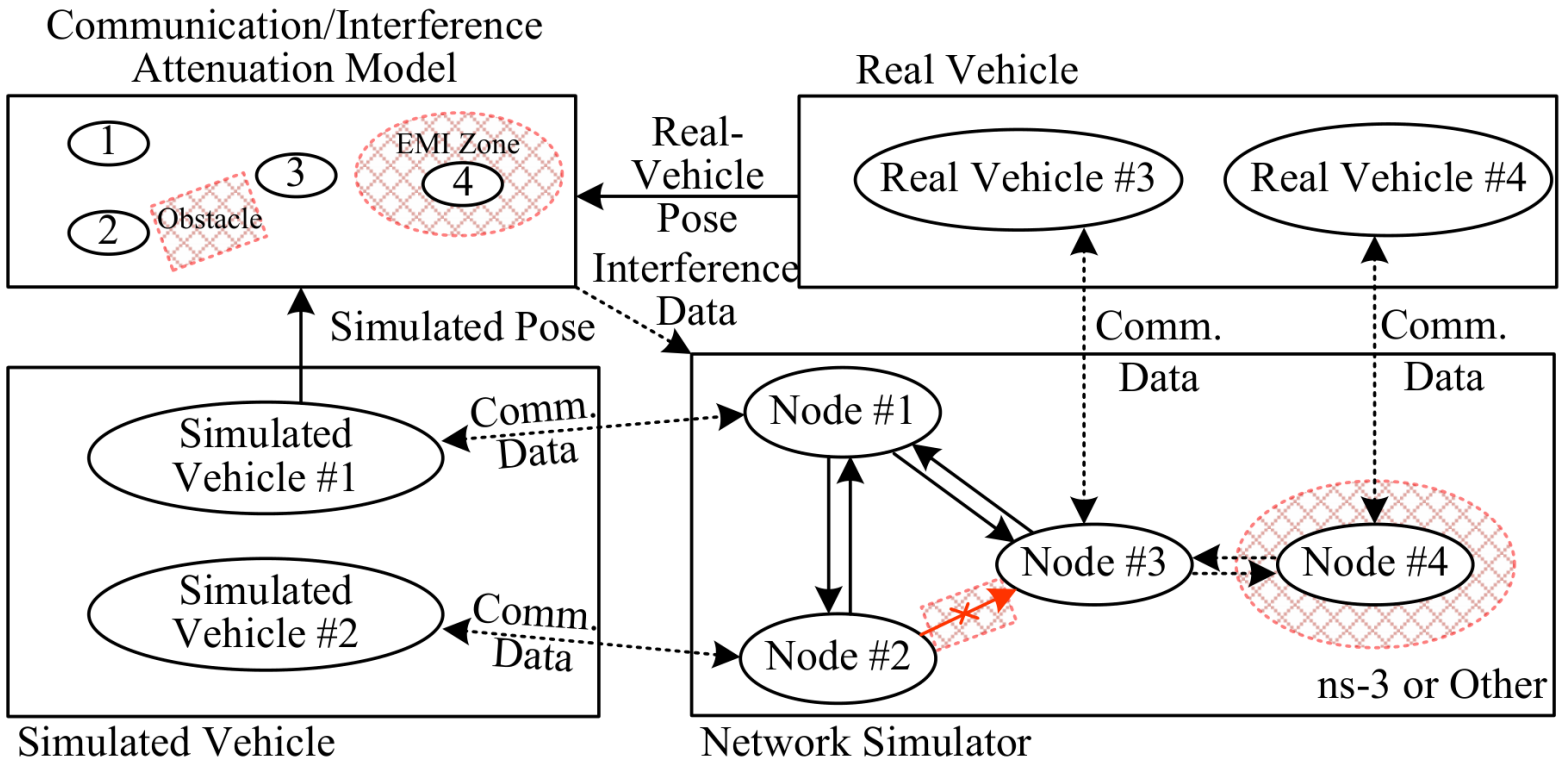


Figure 9.8 RflySim and the ns-3 fine-grained network simulator

9.5.2 Communication Forwarding and Simulation Granularity: Fine-Grained Paths and Clocks

Routing and Radio Contention in ns-3 Determine Packet Delivery



Current integration code: The entry double buffer is polled every 5 ms. Each buffer holds up to 100 entries and removes older entries when full.

3D and real time: The current position update sets altitude to 0. Check the mapping for 3D scenes. Falling behind real time adds delay.

Timing boundary: Add external UDP, bridge buffering, and algorithm processing to packet delay inside ns-3. IP: Internet Protocol.

9.5.2 Communication Forwarding and Simulation Granularity: Distributed Groups and Environment

Distributed Groups Need Intergroup Links and Synchronization

Organization	Required inputs or connections	Required checks
Multiple network instances	Intergroup gateways, packet paths, clocks	Cross-group packets neither bypass nor duplicate loss modeling
Unreal Engine (UE) geometry	Building, terrain, and pose parameters	Geometry supported by the selected ns-3 propagation model
Loss values from UE	Computed attenuation	Units, frequency, update time, model interface
UDP or Redis interface	Application data connected as deployment permits	Whether shared data actually traverse the network model

Group size depends on hardware, models, and packet rate. “10 nodes per group” is not a universal capacity guarantee.

Exercises and Discussion

Why Vehicle 2 Is Unaffected by Simulated Network Delay

Scenario: Vehicle 2 has both a constrained network path and a direct state connection to vehicle 1 on 20101. Its algorithm always reads the direct connection.

Question 1: Why might increasing network loss barely change algorithm performance? Which data sources and times should be logged?

Question 2: What do direct, coarse-grained, and fine-grained tests each examine? Can delay inside the model represent end-to-end delay?

Reference Points

Trace the Data the Algorithm Actually Reads

Path: Remove the bypass at the algorithm input. Log source ID, input interface, and sampling, reception, and use times separately.

Granularity: Direct links test path logic. Coarse-grained models test aggregate delivery conditions. Fine-grained models test device, channel, and routing events.

Timing: Log external communication, bridge buffering, internal network delay, and algorithm processing separately. Use the same application flow and aligned clocks.

Section Summary

Data Paths Determine Whether the Network Model Takes Effect

Data sources: Label ground truth, estimates, observations, and control separately. Align local coordinate frames and time bases first.

Forwarding boundary: Only application packets that traverse the model are affected. Separate coarse- and fine-grained results and external and internal delay.

Timing boundary: Internal model delay differs from end-to-end delay. Choose simulation granularity for the task.

Next section: MAVLink encoding, Data Distribution Service (DDS) matching, and OLSR maintenance of multihop routes.