Chapter Outline

Chapter Experiments and Integrated Validation

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

4 Units in This Section

9.7.1 Experiment Catalog and Minimal Interface Tests

Textbook: Section 9.3.1, interface learning

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation

Textbook: Section 9.3.1; basic use / advanced development

9.7.3 Co-Simulation and Advanced Case Studies

Textbook: Sections 9.3.1–9.3.2, Figure 9.10

9.7.4 Result Interpretation and Integrated Validation

Textbook: Section 9.3, integrated case validation

9.7.1 Experiment Catalog and Minimal Interface Tests: Textbook Experiment List

Experiment Categories and Learning Tasks

Parent directory: [installation directory]/RflySimAPIs/9.RflySimComm/

Category and folder	Question to answer	Representative resources
Interface learning **0.ApiExps**	How are data sent, addressed, and parsed?	RflySim; Data Distribution Service (DDS); Micro Air Vehicle Link (MAVLink); Message Queuing Telemetry Transport (MQTT); Redis; ns-3; multicast
Basic usage **1.BasicExps**	How does multi-vehicle control use communication?	RflySim interfaces and DDS, MQTT, and Redis multi-vehicle examples
Advanced development **2.AdvExps**	How do network constraints affect tasks?	Coarse-grained models, resource allocation, routing simulation

Topics: Index.pdf. Steps: each example's Readme.pdf. Interface reference: API.pdf.

9.7.1 Experiment Catalog and Minimal Interface Tests: One Shared State Dataset

A Default Route Through Four Experiment Stages

Point-to-point (P2P)
Two-vehicle state exchange
→
Multicast
Three-vehicle state sharing
→
Ring following
Four-vehicle state dependency
→
NetSim4Demo
Coarse network constraints

Stage	Inputs and operations	Evidence
Two / three vehicles	Fix vehicle identifiers (IDs), ports, and state fields	Correct source-ID set and continuously updated timestamps
Four / network limits	Fix the trajectory, then vary distance or model parameters	Verify packets traverse the model, then check delivery and freshness

9.7.1 Experiment Catalog and Minimal Interface Tests: Versions and Side Effects

Environment Setup and Startup-Script Effects

Preparation: Use the installed toolchain and the example's supplied environment. Point PSP_PATH to the actual installation directory. Scripts may still default to drive E:.

Version: These checks use the RflySim 5.01 installation. Python38 is a folder name. Check python --version and match libraries to the interpreter version and 32/64-bit architecture.

Side effects: SITLRun*.bat may request elevation and terminate CopterSim, QGC (QGroundControl), RflySim3D, and python processes by name, affecting other sessions.

Network and dependencies: Keep protection enabled. Check application rules, User Datagram Protocol (UDP) ports, multicast adapters, and local-network scope. NetSim needs PyQt5, Matplotlib, and NetworkX.

9.7.1 Experiment Catalog and Minimal Interface Tests: Minimal Send and Receive Operations

Startup Order for Two-Vehicle Point-to-Point Communication

0.ApiExps/e1.RflyNetAPIExps/
1.P2PSendData/

Preparation: Open Readme.pdf. Verify unmanned aerial vehicle (UAV) IDs 1 and 2 and the toolchain environment. This example checks state exchange only; takeoff is unnecessary.

Startup: Run SITLRun2MavlinkFull.bat and wait for both CopterSim/PX4 instances to be ready.

Run: Open two windows with Python38Run.bat. Run python UAV2Ctrl.py first, then python UAV1Ctrl.py.

9.7.1 Experiment Catalog and Minimal Interface Tests: Point-to-Point Observations and Checks

Target ID, UDP Port, and the Source-State Dictionary



Target ID: 2 Receive port: 60002 Source dictionary: UavDict['1']
Equivalent explicit receive call: net.StartNetRec(60002)

Expected message: Got UAV1 Data. Check CopterID = 1 and position units in meters. Local north-east-down (NED) position and global map position are not interchangeable.

Criterion: The intended source state confirms initial connectivity. One printout does not establish sustained packet rate or acceptable delay.

9.7.1 Experiment Catalog and Minimal Interface Tests: One-to-Many Distribution

Three-Vehicle Multicast and Expected Source Sets

0.ApiExps/e1.RflyNetAPIExps/
2.BroadSendData/

Preparation and startup: Open Readme.pdf. Run SITLRun3MavlinkFull.bat, then UavPythonRunALL.bat.

Observe: Each window prints received vehicle IDs and global positions once per second. Default sending and receiving use 224.0.0.10:60000.

Criterion: List each receiver's source-ID set. It must include the other two vehicles with ongoing state updates. Record local loopback separately; it is not another node.

9.7.1 Experiment Catalog and Minimal Interface Tests: Connectivity and Continuous Updates

Tracing Multicast Failures Layer by Layer

Symptom	Check first	Evidence to recheck
No source IDs	CopterSim state/Python setup, address/port, multicast membership	Sender state changes and source IDs appear at receiver
Only some nodes arrive	Duplicate IDs, source not started, adapter and network scope	Source-ID set at each receiver
Position does not change	Is the vehicle stationary, or have messages stopped?	Do timestamps and receive counts keep increasing?
Works locally, fails across hosts	Required app/UDP rules, adapter, switch multicast policy	Recheck each host under the same sending conditions

Chapter Outline

Chapter Experiments and Integrated Validation

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

4 Units in This Section

9.7.1 Experiment Catalog and Minimal Interface Tests

Textbook: Section 9.3.1, interface learning

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation

Textbook: Section 9.3.1; basic use / advanced development

9.7.3 Co-Simulation and Advanced Case Studies

Textbook: Sections 9.3.1–9.3.2, Figure 9.10

9.7.4 Result Interpretation and Integrated Validation

Textbook: Section 9.3, integrated case validation

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Basic Usage Experiment

Preparing and Starting the Four-Vehicle Ring Example

1.BasicExps/e1.RflyNetSimExps/
3.PythonNetSimAPI-SimpPack/

Preparation: Open Readme.pdf. Confirm four instances with IDs 1–4, the same scene, and consistent initial positions.

Operation: Run SITLRun4MavlinkFull.bat. Wait until all show 3D Fixed, then run UavPythonRunALL.bat.

Run effects: initOffboard changes simulation protection settings; then the scripts arm, take off, and follow.
Coordinates: NED down = −10 m means 10 m above the reference origin.

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Message Topology

A Communication Ring and a Following Chain Differ

Source ID	Destination port	How the receiver uses the state
1	60002, to vehicle 2	Vehicle 2 follows vehicle 1 with a 1 m offset behind its north position
2	60003, to vehicle 3	Vehicle 3 reads and follows vehicle 2
3	60004, to vehicle 4	Vehicle 4 reads and follows vehicle 3
4	60001, to vehicle 1	Vehicle 1 waits for vehicle 4 to be ready, then executes planned motion

Message flow: 1→2→3→4→1. Continuous following chain: 1→2→3→4.

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Update Events and Control

One-Step Calculations for Downsampling and Following Error

Interface: enNetForward([60002], '224.0.0.10', 2). Interval=2 sends once per three processed events.

$$f_{\text{send}} \approx \frac{f_{\text{event}}}{I+1} = \frac{90}{3} = 30\ \text{Hz}$$

Assume 90 Hz processed events; actual rate also depends on coalescing and scheduling.

Teaching inputs: Vehicle 1 is at 10 m north, vehicle 2 at 8.4 m north. Following offset: 1 m behind. Proportional gain: 1 / s.

$$v_{2,N} = 1\ \text{s}^{-1}(10 - 1 - 8.4)\ \text{m} = 0.6\ \text{m/s}$$

Set each velocity component to zero when its magnitude is below 0.2 m/s. The loop runs about every 0.03 s. This is a given-state calculation, not a measured response.

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Coarse-Grained Constraints

Startup Order for NetSim4Demo

2.AdvExps/e1.CoaGraNetSimExps/
1.NetSim4Demo/

Preparation: Open Readme.pdf and readme.txt. Check PSP_PATH, the four-vehicle environment, and PyQt5 / Matplotlib / NetworkX dependencies.

Default sequence: SITLRun4MavlinkFull.bat → all show 3D Fixed → PythonRunALL.bat.

Stepwise troubleshooting: In the example environment, run python NetworkSimulation.py, then UavPythonRunALL.bat, then python PyViaGui.py. Start each only once.

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Forwarding Data Paths

State, Application Packets, and Display Data Use Different Ports

Source → destination	Port and address	Data role
CopterSim → network model	224.0.0.11:20008	168-byte state packet; position determines distance
UAVCtrl → network model	Multicast 224.0.0.11:30000	Application packet; 24-byte header includes source/target IDs
Network model → receiver	Target host, UDP 22000+ID	Application packets after random loss/delay. Default: print only, no motion control.
Network model → PyViaGui	224.0.0.11:32000	Positions, direct-link quality, and per-source statistics

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Teaching Model

How Distance Maps to Link Quality

Default max_range=0.3 km, with a reliable region through 120 m. Let d be inter-node distance in meters and q the model's single-link delivery quality.

$$q(d) = \begin{cases} 1, & d \le 120 \\ 1 - \dfrac{\log_{10}(d/120)}{\log_{10}(300/120)}, & 120 < d < 300 \\ 0, & d \ge 300 \end{cases}$$

Distance / m	120	180	240	300
Quality q	1.000	0.557	0.244	0.000

These are deterministic teaching calculations, not measurements. The model lacks radio calibration and full wireless access contention.

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: One-Step Model Calculation

Path Reliability and Scheduled Delay

Teaching network: Direct A–C quality is 0.25; A–B is 0.8; B–C is 0.7. These edge qualities are model inputs for this example only.

$$Q(A \to B \to C) = 0.8 \times 0.7 = 0.56 > 0.25$$

$$\tau_{\text{plan}} = \frac{0.001}{Q}\ \text{s} = \frac{0.001}{0.56}\ \text{s} \approx 1.79\ \text{ms}$$

Execution: Enqueue only when a random draw is below path quality. Once due, forward to target 22000+ID. Drop when quality is 0.

This maximum-product model differs from the textbook's minimum-sum Expected Transmission Count (ETX) path and from Optimized Link State Routing (OLSR).

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Reading Model Results

Display Statistics and Implementation Limits

Item	Current rule	Interpretation limit
Loss counter	Per-source target forwarding attempts, plus some format errors	Broadcast may count per target; unregistered targets may skip counts.
Time window	Statistics update and reset about every 1 s	Different windows cannot be added to obtain an end-to-end loss rate.
Bandwidth threshold	data_rate accumulates len(buf)/128; default bandwidth=100.0	Software threshold convention, not calibrated radio throughput
Delay queue	Check the due time of the head in arrival order	Scheduled ≠ actual arrival delay; earlier packets may block later ones.

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation: Controlled Comparison

Criteria for Three Distance Conditions

Condition	Operation and observation	Criterion
Near: ≤120 m	Keep traffic rate fixed. Log distance, quality, receive sequence IDs.	Direct quality = 1; check queues, bandwidth limits and actual receive counts.
Edge: 120–300 m	Compare distance bands. Log relay paths and receive intervals.	Direct model quality falls. Repeat random trials across seeds.
Direct: ≥300 m	Inspect other links and the receiver together	Direct quality is 0. With relays, path quality can still exceed 0.

Fix trajectory, packet size/rate, controller, and window. Change one network factor per test.

Chapter Outline

Chapter Experiments and Integrated Validation

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

4 Units in This Section

9.7.1 Experiment Catalog and Minimal Interface Tests

Textbook: Section 9.3.1, interface learning

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation

Textbook: Section 9.3.1; basic use / advanced development

9.7.3 Co-Simulation and Advanced Case Studies

Textbook: Sections 9.3.1–9.3.2, Figure 9.10

9.7.4 Result Interpretation and Integrated Validation

Textbook: Section 9.3, integrated case validation

9.7.3 Co-Simulation and Advanced Case Studies: Fine-Grained Network Simulation

Prerequisites for ns-3 Co-Simulation

General entry: 0.ApiExps/e6.NS3DepsTestExps/ contains RflyNet.cc, TestSend.py, and PythonAllUavDemo/.

Custom environment: Related 3.CustExps cases require the Full Edition or higher. Current scripts use the Windows Subsystem for Linux (WSL) distribution RflySim-20.04 and a custom ns-3.33 source tree.

Dependencies: Check custom modules such as gpsr, aodvKmeans, and parrot, plus the compiler, network interfaces, real-time scheduling, and UDP port mappings.

Source check: This entry's RflyNet.cc swaps position and velocity fields. Correct the mapping before validation. A standard ns-3 release cannot directly replace the custom tree.

9.7.3 Co-Simulation and Advanced Case Studies: Optional Study with Prerequisites

Custom Entry Points and Directory Dependencies

Common directory: 3.CustExps/e2.NS3_UAVCommExps/
Full Edition or higher only. Every subdirectory starts with OneScript.bat.

Subdirectory	Purpose and observation	Dependency
1.NS3	Basic coupling; instructions specify no flight	Own ns-3.33.zip and netsim/
2.NS3-Udp-PyQtViz	UDP state with PyQt visualization	Reuses adjacent 1.NS3 tree
3.NS3-PyViz	PyViz topology inside ns-3	Reuses adjacent 1.NS3 tree
4.NS3-Ctrl	Network and control coupling	Own source tree and archive

9.7.3 Co-Simulation and Advanced Case Studies: Installation and Startup Effects

OneScript Sequence and Its File Changes

Scope: 3.CustExps/e2.NS3_UAVCommExps/, Full Edition or higher.

Check Readme.pdf
Back up experiment copy

➡

Open selected subdirectory
Run OneScript.bat

➡

Check build and processes
Then verify port traffic

Script actions: Start uav simulation, extract ns-3.33.zip in WSL, copy netsim/RflyNet.cc to scratch, then sudo ./waf configure / build / --run.

Effects: Source replacement, build-file writes, elevation, and multiple processes. If communication fails, check application and port rules while keeping protection enabled.

9.7.3 Co-Simulation and Advanced Case Studies: Preparation, Operation, and Criteria

Staged Acceptance for Co-Simulation

Stage	Operation and observation	Pass criterion
Build and configuration	Record source version, modules, and build output	Successful build exit; target program exists
Minimal packet forwarding	TestSend.py or the case's uav entry; source/target IDs and UDP ports	Match the same message sequence ID at the target
Motion and network	Input a known trajectory; check coordinates and units	Motion changes map to ns-3 node positions
Control and comparison	Switch network conditions in a control-specific case	Control reads the model exit. One window for network/task metrics.

9.7.3 Co-Simulation and Advanced Case Studies: Real-Virtual Coupling Case

Communication Between Real Vehicles and a Virtual Scene

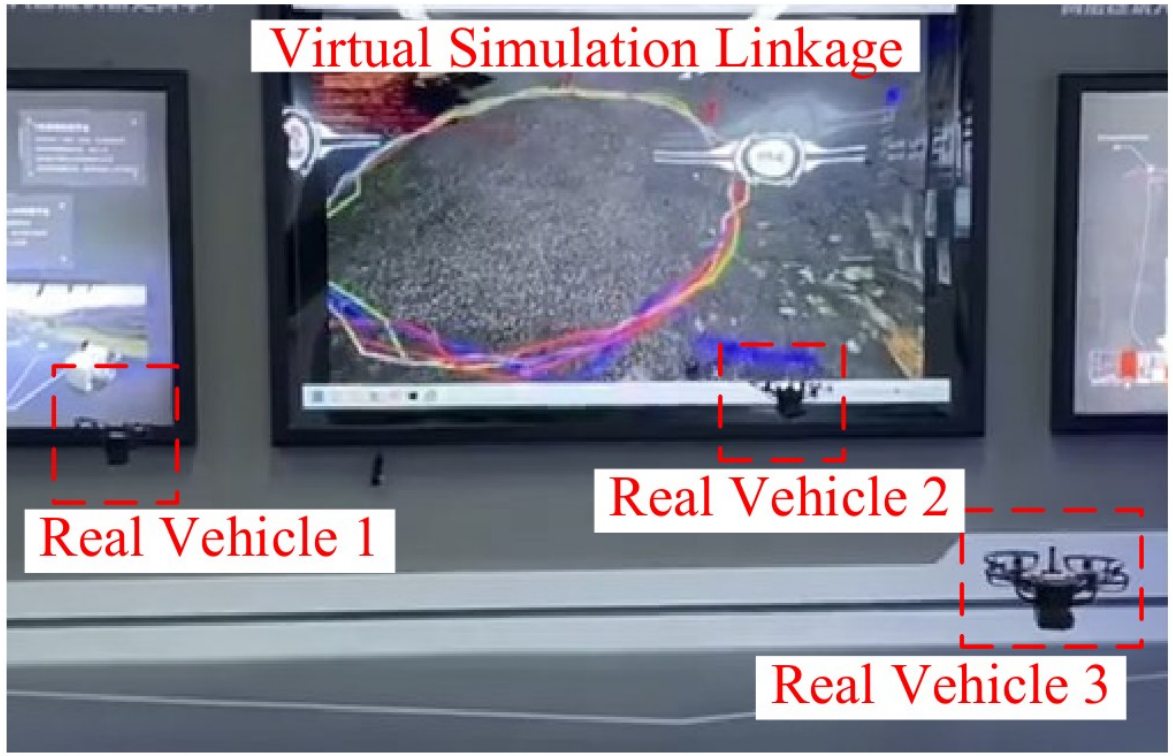


Figure 9.10(a) Real-virtual communication simulation

Objects: Real nodes link over a local network to simulated nodes. The virtual scene provides the task environment.

Mechanism: The network simulator relays node messages to control and display.

Criterion: Identify constrained messages and check for an ideal direct path bypassing the model.

9.7.3 Co-Simulation and Advanced Case Studies: Digital-Twin Visualization

What 3D Communication Effects Represent

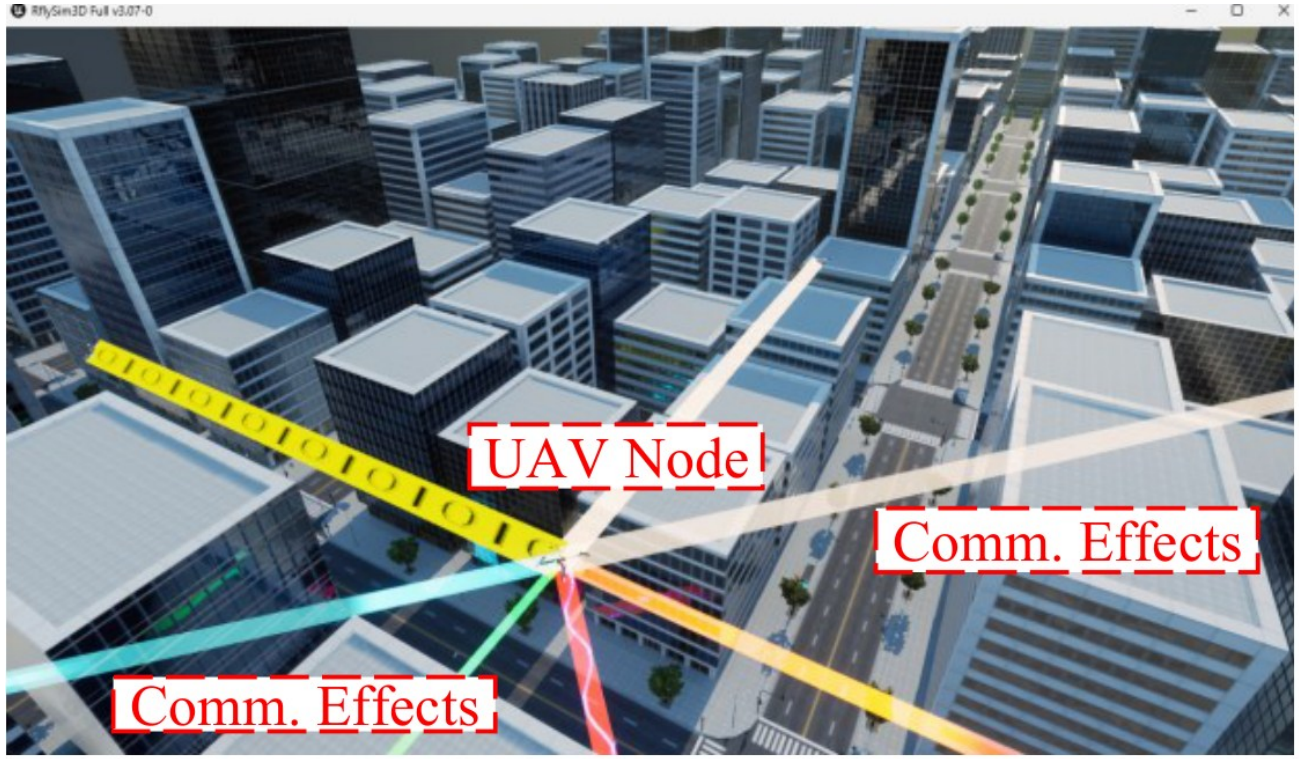


Figure 9.10(b) Digital-twin communication visualization

Objects: Unmanned nodes, lines, and regions in the scene show communication state.

Mechanism: Unreal Engine (UE) Blueprint interfaces visualize connectivity, rate, delay, or loss.

Criterion: Identify the data behind colors/lines, then match the display to logs at the same time.

9.7.3 Co-Simulation and Advanced Case Studies: Coarse-Grained Result Illustration

Node Positions and Direct-Link Quality

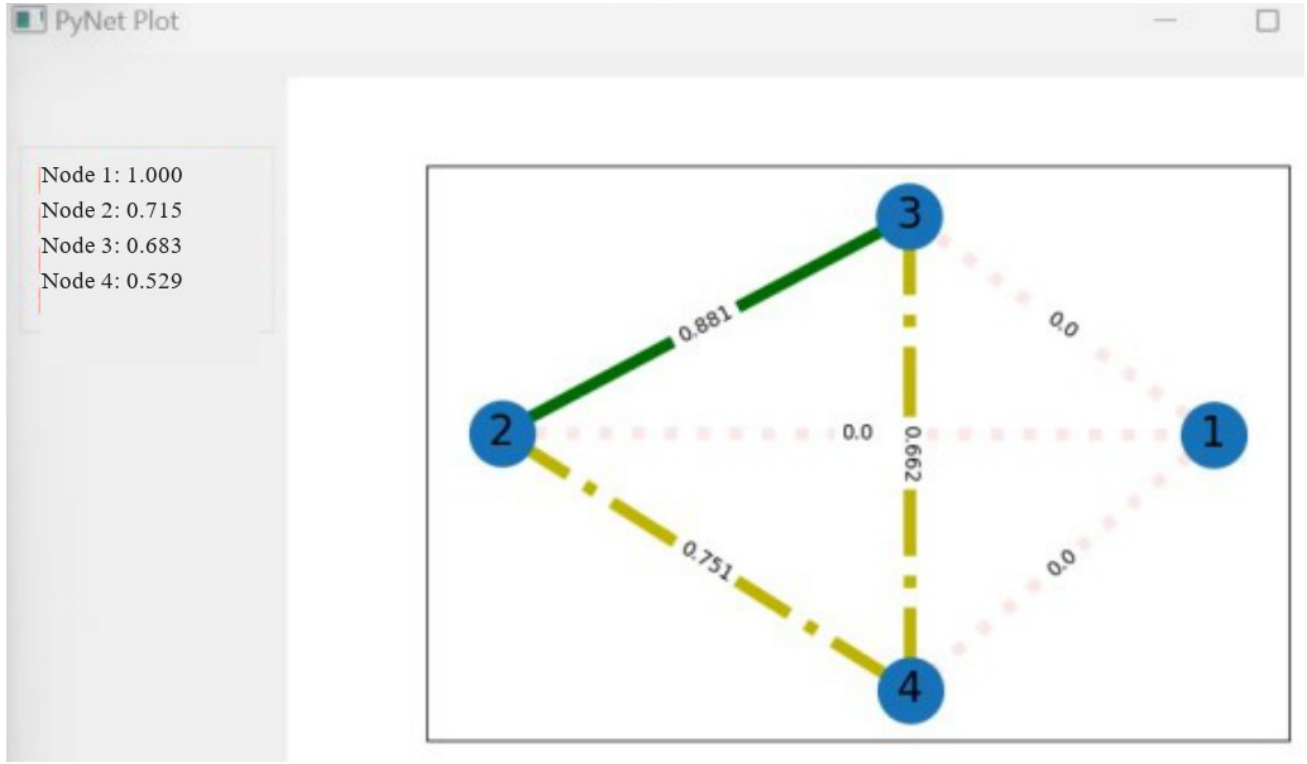


Figure 9.10(c) Python coarse-grained network simulation

Objects: Node positions and connection quality for each node pair.

Mapping: Distance determines quality. Quality affects path selection and delivery probability.

Criterion: Distinguish a direct link from an end-to-end path. Check both in the same time window.

9.7.3 Co-Simulation and Advanced Case Studies: ns-3 Result Illustration

Topology and Packet Events in Fine-Grained Simulation

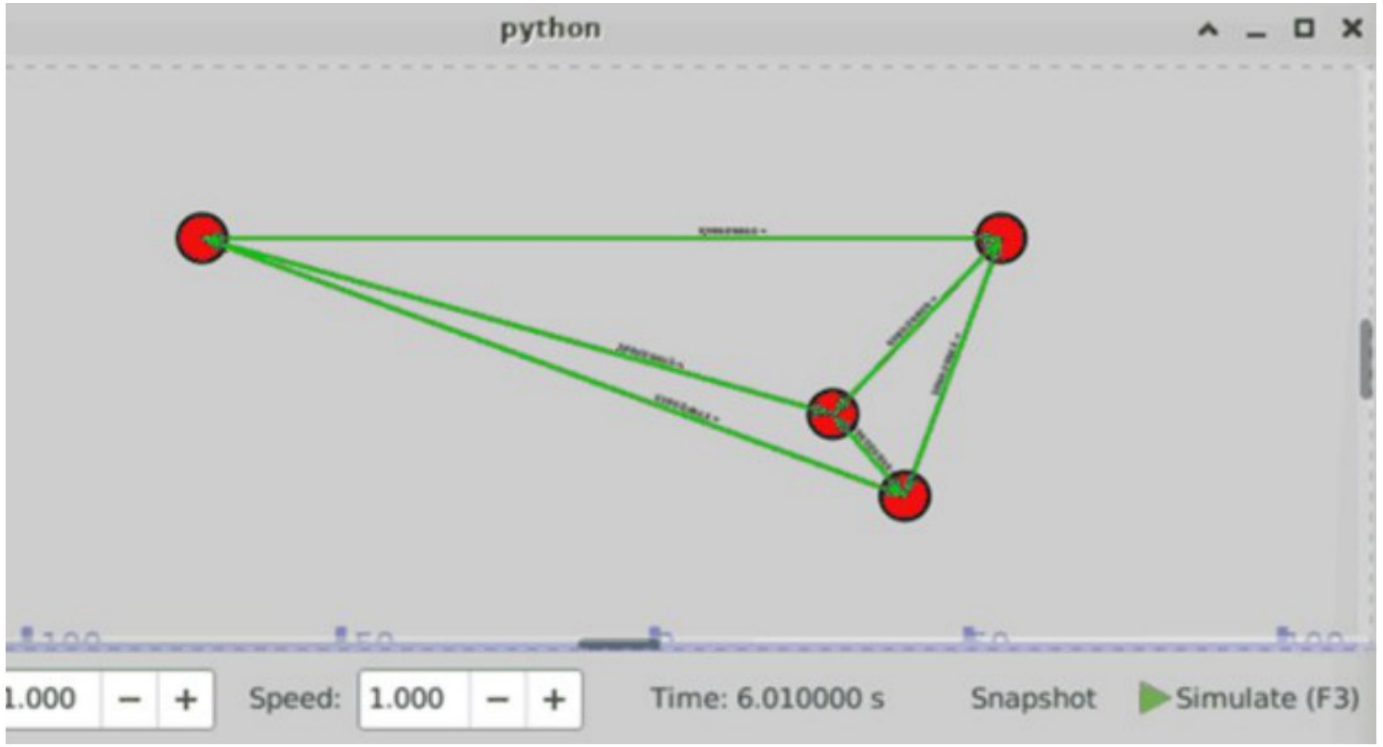


Figure 9.10(d) ns-3 fine-grained communication simulation

Objects: Network nodes, connections, and communication visualization.

Mechanism: Devices, channels, applications, and routing interact through discrete events.

Criterion: Use packet events, routing tables, and receive statistics to judge delay. Links alone are insufficient.

9.7.3 Co-Simulation and Advanced Case Studies: Resource Allocation and Middleware

Optional Alternatives and Their Limits

Directory or entry	What can be studied	Conditions and limits
0.ApiExps/e2.DDSDepsTestExps/ Demo/HelloWorldExample.py	Pub/sub matching; flags: -p subscriber / -p publisher	Check DDS/Python binding versions and 32/64-bit libraries.
0.ApiExps/e4.MQTTDepsTestExps/ 0.ApiExps/e5.RedisDepsTestExps/	Broker distribution, publish/subscribe, or key-value polling	Match extra services and versions; optional for the default route.
2.AdvExps/e2.NetResAlloc/ main.py	Random policy interacts with environment and interface	Main entry fixes random; other branches remain TODO.

Chapter Outline

Chapter Experiments and Integrated Validation

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

4 Units in This Section

9.7.1 Experiment Catalog and Minimal Interface Tests

Textbook: Section 9.3.1, interface learning

9.7.2 Multi-Vehicle Communication and Coarse-Grained Simulation

Textbook: Section 9.3.1; basic use / advanced development

9.7.3 Co-Simulation and Advanced Case Studies

Textbook: Sections 9.3.1–9.3.2, Figure 9.10

9.7.4 Result Interpretation and Integrated Validation

Textbook: Section 9.3, integrated case validation

9.7.4 Result Interpretation and Integrated Validation: Metric Definitions

Communication Metrics and What They Count

Convention: Fix one source–target application flow and a sending window of length T. Deduplicate, exclude warmup, and allow an agreed drain time after sending stops.

$$P_{\text{loss}} = 1 - \frac{N_{\text{rx,unique}}}{N_{\text{tx}}}$$

N_{tx}: sent-message count; $N_{\text{rx,unique}}$: unique received-message count.

$$G = \frac{8\sum L_{\text{payload,rx}}}{T} \quad [\text{bit/s}]$$

$L_{\text{payload,rx}}$: payload bytes per successful message; T: seconds.

Delay: Receive time minus send time for successful messages. Report median, 95th percentile (P95), maximum, and clock error.

Task: Also log following error, disconnection duration, and degradation count. Define separate criteria for communication and task success.

9.7.4 Result Interpretation and Integrated Validation: Teaching Record, Not Measurements

A Six-Message Record You Can Recalculate

Seq. ID	Sent / ms	Received / ms	Delay / ms
1	0	10	10
2	20	35	15
3	40	65	25
4	60	75	15
5	80	Lost	Exclude from delay
6	100	140	40

Assume generation equals sending, one clock, a 1 s window, and 100 B per packet. Sent 6, received 5: loss 16.7%; goodput 4,000 bit/s.

Delivered: mean 21 ms, median 15 ms; nearest-rank 95th percentile (P95): sorted item 5 = 40 ms.

9.7.4 Result Interpretation and Integrated Validation: Clocks and State Freshness

Timestamps and Age of Information

$$\hat{d} = t_{\mathrm{rx}}^{(B)} - t_{\mathrm{tx}}^{(A)} = d + \Delta_{BA}$$

d is true delay. Δ_{BA} is receiver B's clock offset relative to sender A. Align the time bases before subtracting timestamps across hosts.

$$A(t) = t - t_{\mathrm{gen,latest}}; \qquad A(100\ \mathrm{ms}) = 100 - 60 = 40\ \mathrm{ms}$$

At 100 ms on the previous page, message 4 has the latest generation time among received states: 60 ms. At a constant 10 m/s, motion during these 40 ms is 0.4 m.

The 0.4 m is displacement while state is stale. Actual following error also depends on prediction, feedback, and the controller.

9.7.4 Result Interpretation and Integrated Validation: Integrated Validation Design

Comparing an Idealized Path with a Constrained Network

Comparison stage	Keep consistent	Added mechanisms / attribution
Idealized direct link	Reference trajectory, initial state, packet size/rate, controller, window, clock basis	Baseline interfaces and task performance
Coarse-grained model	Same application flow and reference trajectory	Distance quality, random loss, scheduled delay, and software thresholds
ns-3 fine-grained model	Same node identities and application inputs	Device/channel/access/routing events explain queuing and delivery differences

Output: Parameter table, packet logs, actual trajectories, task errors; conditions and repetition count.

9.7.4 Result Interpretation and Integrated Validation: Failure Diagnosis and Recovery

Ending an Experiment and Keeping Traceable Records

Record: Versions, script Secure Hash Algorithm (SHA) hashes, parameters, trajectories, random seeds, measurement window, time base, and sent/received sequence IDs for every flow.

Shutdown: In each experiment window, use Ctrl+C to stop control and sending first. Then stop the network model and interface. Close this experiment's simulators last.

Blocking: endNetLoop() may wait for the receive thread. If needed, identify the window and process ID (PID), then terminate only that experiment process.

Recovery: Confirm ports are released and retain failure logs. Restore the experiment copy's parameters before retrying. Global termination of python is not a routine shutdown method.

Exercises and Discussion

Class Discussion: What Do the Results Establish?

Question 1: Direct-link quality is 0, but the target still receives packets. How can logs distinguish a working relay, stale buffers, and a model bypass?

Question 2: Design A averages 15 ms delay with 30% loss. Design B averages 25 ms with 1% loss. Does mean delay alone justify choosing A?

Question 3: Subtracting timestamps across hosts gives negative delay. Should you change the network model first, or check clocks, units, and message identity?

Reference Points

Discussion Reference: Evidence Along the Same Flow

Question	Reference interpretation	Required evidence
1 Direct link down; packets arrive	Possible relay or stale sample, or an ideal-path bypass.	Fresh matching sequence IDs, path quality, model-exit and target receive logs
2 Low mean delay	Higher loss may leave only fast samples. Evaluate against task requirements.	Deduplicated loss, P95, information age, following error, outage duration
3 Negative delay	Check clocks and message pairing first	Units, clock-offset bounds, source/target IDs, and message sequence IDs

Section Summary

From Initial Connectivity to Validation with Clear Limits

Method: Add constraints along the default four-stage route. Preserve message identities and records that support recalculation.

Interpretation: Communication metrics explain delivery. Task metrics explain control using the delivered data. Tie conclusions to versions, conditions, and measurement windows.

Common errors: An ID is not a UDP port. Multicast does not imply fully connected control. Model quality and display colors are not measured link performance.

Next: Section 9.8 recalculates links and routes, then designs integrated validation with explicit capacity, delay, and failure conditions.