# Simulare Comunicare in Inel

**Membru echipa:** Rafa Ioana-Sorina

Contents

[Simulare Comunicare in Inel 1](#_Toc196695040)

[1. Introducere 1](#_Toc196695041)

[2. Prezentarea topologiei 1](#_Toc196695042)

[3. Justificarea limbajului de programare ales 2](#_Toc196695043)

[4. Cod relevant si explicatii 2](#_Toc196695044)

[5. Capturi Wireshark 3](#_Toc196695045)

[6. Bibliografie 3](#_Toc196695046)

[7. Anexa - Cod sursa complet 4](#_Toc196695047)

## 1. Introducere

Acest proiect are ca scop simularea comunicarii intr-o topologie de tip inel folosind trei noduri, cu transmiterea si incrementarea unei valori numerice pana la atingerea valorii 100. Comunicarea se realizeaza utilizand socket-uri TCP pe adrese IP din intervalul de bucla locala (127.0.0.x).

## 2. Prezentarea topologiei

Topologia utilizata este un **inel** format din trei noduri:

* Nod 1 (127.0.0.1:5001) ➔ Nod 2 (127.0.0.2:5002)
* Nod 2 (127.0.0.2:5002) ➔ Nod 3 (127.0.0.3:5003)
* Nod 3 (127.0.0.3:5003) ➔ Nod 1 (127.0.0.1:5001)

Primul nod initiaza comunicarea cu valoarea 1. La fiecare primire, valoarea este incrementata si retransmisa mai departe. Simularea se opreste cand valoarea atinge 100.

## 3. Justificarea limbajului de programare ales

Am ales **Python** datorita avantajelor sale:

* Cod scurt si clar pentru programare de retea
* Suport excelent pentru programarea cu socket-uri
* Integrare usoara cu Wireshark pentru capturarea traficului
* Portabilitate si rulare simpla pe Windows

## 4. Cod relevant si explicatii

**Structura clasei RingNode:**

* \_\_init\_\_(): seteaza IP-ul, porturile si rolul de initiator.
* start(): porneste listenerul si, daca este initiator, trimite valoarea initiala.
* listen(): asteapta conexiuni si proceseaza mesajele primite.
* send(): trimite valorile catre nodul urmator, cu retry automat in caz de eroare.

**Exemplu cod fragment:**

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind((self.listen\_ip, self.listen\_port))

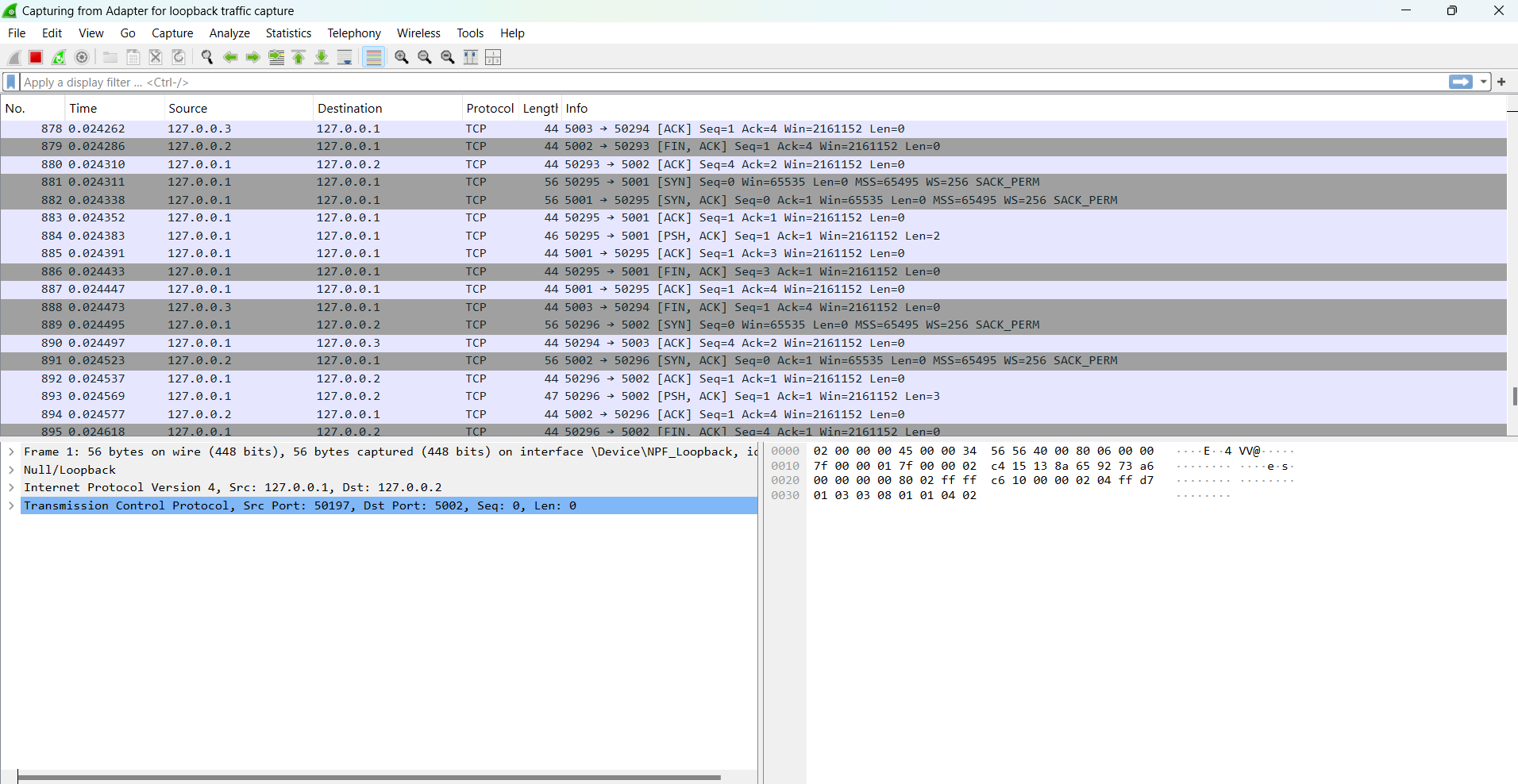
server\_socket.listen(1)

Aceasta bucata de cod initializeaza socketul de ascultare pentru fiecare nod.

## 5. Capturi Wireshark

A screenshot of a computer

AI-generated content may be incorrect.



## 6. Bibliografie

* [Python socket documentation](https://docs.python.org/3/library/socket.html)
* Wireshark User Guide

## 7. Anexa - Cod sursa complet

import socket

import threading

import time

class RingNode:

    def \_\_init\_\_(self, listen\_ip, listen\_port, next\_ip, next\_port, is\_initiator=False):

        self.listen\_ip = listen\_ip

        self.listen\_port = listen\_port

        self.next\_ip = next\_ip

        self.next\_port = next\_port

        self.is\_initiator = is\_initiator

        self.running = True

    def start(self):

        listener\_thread = threading.Thread(target=self.listen)

        listener\_thread.start()

        if self.is\_initiator:

            time.sleep(1)

            self.send(1)

    def listen(self):

        server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

        server\_socket.bind((self.listen\_ip, self.listen\_port))

        server\_socket.listen(1)

        print(f"[{self.listen\_ip}:{self.listen\_port}] Listening...")

        while self.running:

            conn, addr = server\_socket.accept()

            data = conn.recv(1024)

            if data:

                value = int(data.decode())

                print(f"[{self.listen\_ip}:{self.listen\_port}] Received: {value}")

                if value >= 100:

                    print(f"[{self.listen\_ip}:{self.listen\_port}] Value 100 reached, stopping.")

                    self.running = False

                else:

                    self.send(value + 1)

            conn.close()

    def send(self, value):

        success = False

        attempts = 0

        while not success and attempts < 10:

            try:

                client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

                client\_socket.connect((self.next\_ip, self.next\_port))

                client\_socket.send(str(value).encode())

                print(f"[{self.listen\_ip}:{self.listen\_port}] Sent: {value} to [{self.next\_ip}:{self.next\_port}]")

                client\_socket.close()

                success = True

            except ConnectionRefusedError:

                print(f"[{self.listen\_ip}:{self.listen\_port}] Connection refused, retrying in 1 second...")

                attempts += 1

                time.sleep(1)

            except Exception as e:

                print(f"[{self.listen\_ip}:{self.listen\_port}] Unexpected error: {e}")

                break

if \_\_name\_\_ == "\_\_main\_\_":

    import sys

    if len(sys.argv) != 2:

        print("Usage: python ring\_communication.py <node\_number (1/2/3)>")

        sys.exit(1)

    node\_number = int(sys.argv[1])

    if node\_number == 1:

        node = RingNode('127.0.0.1', 5001, '127.0.0.2', 5002, is\_initiator=True)

    elif node\_number == 2:

        node = RingNode('127.0.0.2', 5002, '127.0.0.3', 5003)

    elif node\_number == 3:

        node = RingNode('127.0.0.3', 5003, '127.0.0.1', 5001)

    else:

        print("Node number must be 1, 2 or 3.")

        sys.exit(1)

    node.start()