TCP Congestion Control – Complete Lab Task

P A Mohammed Maaz

2503B05101

Q1. LAB TASK: TCP Congestion Control Simulation

Question:
Using a TCP congestion control simulator (Python or any tool of your choice), **simulate and analyze TCP Reno** under three different packet-loss conditions:

- **0.1% loss**
- **1% loss**
- **2% loss**

For each case:

1. **Plot the Congestion Window (cwnd) vs Time** for at least 20 seconds of simulation.
2. **Compute the average throughput** (in MSS/second or Mbps).
3. **Explain how packet loss affects the behavior of TCP Reno**, specifically discussing:
    - Slow Start
    - Congestion Avoidance
    - Multiplicative Decrease (cwnd halving)
4. **Compare the three cases** and write a short conclusion on how increasing loss impacts network performance.

Reference:

```
# tcp_sim.py
# Simple discrete-time TCP congestion control simulator (Reno-like)
# Usage: python tcp_sim.py
```

```
import numpy as np
import matplotlib.pyplot as plt
import random

# --- Simulation parameters ---
RTT = 0.05              # seconds (50 ms)
sim_time = 20.0         # seconds
mss = 1.0               # normalized MSS (units)
loss_prob = 0.01        # per-packet loss probability (Bernoulli)
packets_per_seg = 1     # one packet per MSS in this model

# TCP behavior params
algorithm = "reno"      # "reno", "tahoe"
initial_cwnd = 1.0      # in MSS
ssthresh_init = 64.0    # in MSS
rto = 1.0               # simplified timeout (s)

# --- State variables ---
t = 0.0
cwnd = initial_cwnd
ssthresh = ssthresh_init
time_steps = []
cwnd_trace = []
sent_packets_total = 0
acked_packets_total = 0
```

```
# For throughput measurement
bytes_acked = 0.0

# Helper: simulate sending cwnd-sized window; return True if loss detected
def send_window(cwnd):
    # we model loss per packet with independent Bernoulli trial
    segs = int(max(1, round(cwnd / mss)))
    lost = False
    acks = 0
    for _ in range(segs):
        sent_packets_total_local = 1
        if random.random() < loss_prob:
            lost = True
        else:
            acks += 1
    return lost, acks

# Main loop: step in RTT-sized intervals
while t < sim_time:
    # record
    time_steps.append(t)
    cwnd_trace.append(cwnd)

    # determine phase: slow start or congestion avoidance
    in_slow_start = (cwnd < ssthresh)
```

```
# send window, get ack/loss (this is simplification; real TCP tracks per-packet ACKs)
lost, acks = send_window(cwnd)

# update bytes acked
bytes_acked += acks * mss

# Simple Reno behavior
if lost:
    # If Tahoe: go to slow start with cwnd = 1
    if algorithm == "tahoe":
        ssthresh = max(cwnd / 2.0, 2.0)
        cwnd = 1.0
    elif algorithm == "reno":
        # Reno: multiplicative decrease, enter congestion avoidance
        ssthresh = max(cwnd / 2.0, 2.0)
        cwnd = max(cwnd / 2.0, 1.0)
    # assume loss consumed this RTT
else:
    # no loss -> increase cwnd
    if in_slow_start:
        cwnd = cwnd * 2.0  # exponential growth (per-RTT doubling)
    else:
        cwnd = cwnd + 1.0  # 1 MSS per RTT (additive increase)
# Cap cwnd to a reasonable value to avoid runaway in this simple sim
cwnd = min(cwnd, 1000.0)
```

```
    # advance time by one RTT
    t += RTT

# Convert to throughputs: bytes_acked / sim_time (in MSS units)
throughput = bytes_acked / sim_time

# Plot cwnd trace
plt.figure(figsize=(10,4))
plt.step(time_steps, cwnd_trace, where='post')
plt.xlabel("Time (s)")
plt.ylabel("cwnd (MSS)")
plt.title(f"TCP {algorithm} simulation: loss={loss_prob}, RTT={RTT}s,
throughput={throughput:.2f} MSS/s")
plt.grid(True)
plt.show()

print(f"Sim done. Throughput (MSS/s): {throughput:.3f}")
```

CODE:

```
# -*- coding: utf-8 -*-
"""Welcome To Colab

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/notebooks/intro.ipynb
```

```
"""


!pip install python-docx --quiet


import random
import math
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
import shutil
from docx import Document
from docx.shared import Inches


RTT = 0.05
SIM_TIME = 20.0
MSS_BYTES = 1460
LOSS_VALUES = [0.001, 0.01, 0.02]
INITIAL_CWND = 1.0
SSTHRESH_INIT = 64.0
RANDOM_SEED_BASE = 42
MAX_CWND = 1000.0


def simulate_tcp_reno(loss_prob, rtt=RTT, sim_time=SIM_TIME,
seed_offset=0):
    """Simulate a simple TCP Reno on RTT-sized steps.
    Returns: (time_steps_array, cwnd_array, throughput_mss_per_s,
throughput_mbps)"""
    random.seed(RANDOM_SEED_BASE + seed_offset)
```

```
    t = 0.0
    cwnd = float(INITIAL_CWND)
    ssthresh = float(SSTHRESH_INIT)

    time_steps = []
    cwnd_trace = []
    bytes_acked = 0.0

    while t < sim_time - 1e-9:
        time_steps.append(t)
        cwnd_trace.append(cwnd)

        in_slow_start = (cwnd < ssthresh)
        segs = max(1, int(math.floor(cwnd + 1e-9)))

        lost = False
        acks = 0
        for _ in range(segs):
            if random.random() < loss_prob:
                lost = True
            else:
                acks += 1

        bytes_acked += acks * MSS_BYTES

        if lost:
```

```
            ssthresh = max(cwnd / 2.0, 2.0)
            cwnd = max(cwnd / 2.0, 1.0)
        else:

            if in_slow_start:
                cwnd = cwnd * 2.0
            else:
                cwnd = cwnd + 1.0

        cwnd = min(cwnd, MAX_CWND)
        t += rtt

    throughput_mss_per_s = (bytes_acked / MSS_BYTES) / sim_time
    throughput_mbps = (bytes_acked * 8.0) / (sim_time * 1e6)
    return np.array(time_steps), np.array(cwnd_trace),
throughput_mss_per_s, throughput_mbps


output_dir = Path("tcp_sim_output")
output_dir.mkdir(exist_ok=True)

results = []
for i, loss in enumerate(LOSS_VALUES):
    ts, cwnd, thr_mss_s, thr_mbps = simulate_tcp_reno(loss, seed_offset=i)
    results.append((loss, ts, cwnd, thr_mss_s, thr_mbps))

    plt.figure(figsize=(10,4))
    plt.step(ts, cwnd, where='post')
```

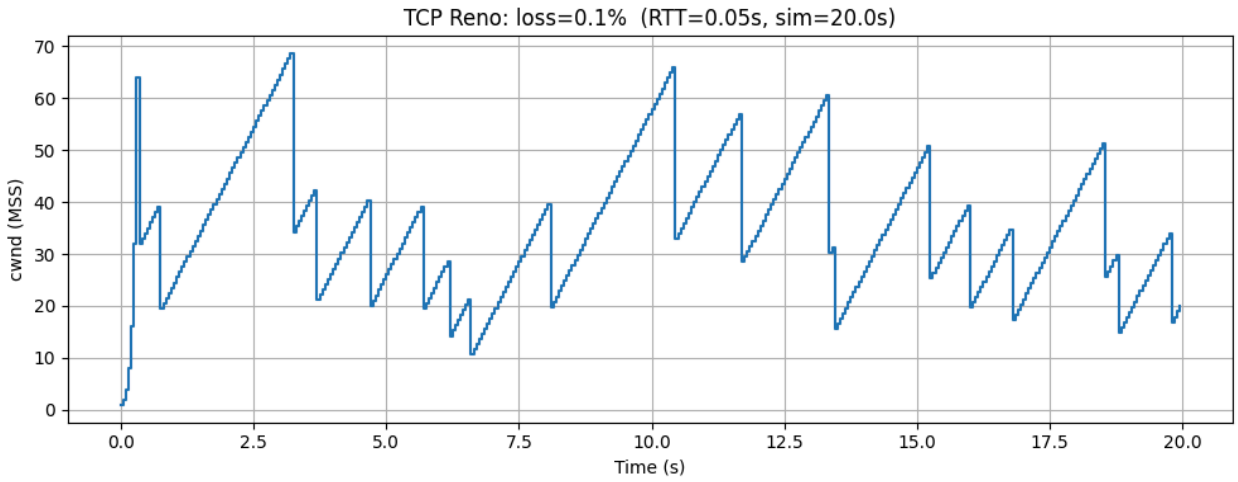
```
    plt.xlabel("Time (s)")
    plt.ylabel("cwnd (MSS)")
    pct = {0.001: "0.1%", 0.01: "1.0%", 0.02: "2.0%"}.get(loss,
f"{loss*100:.3f}%")
    plt.title(f"TCP Reno: loss={pct}  (RTT={RTT}s, sim={SIM_TIME}s)")
    plt.grid(True)
    plt.tight_layout()
    fname = output_dir /
(f"cwnd_{int(loss*10000)}_{int(loss*1000)}pct.png")


    if loss == 0.001:
        fname = output_dir / "cwnd_0.1pct.png"
    elif loss == 0.01:
        fname = output_dir / "cwnd_1pct.png"
    elif loss == 0.02:
        fname = output_dir / "cwnd_2pct.png"
    plt.savefig(fname, dpi=150)
    plt.show()
    plt.close()



plt.figure(figsize=(11,5))
for loss, ts, cwnd, thr_mss_s, thr_mbps in results:
    label = f"{(loss*100):.2f}% loss"
    plt.step(ts, cwnd, where='post', label=label)
plt.xlabel("Time (s)")
plt.ylabel("cwnd (MSS)")
plt.title("TCP Reno cwnd vs Time (comparison)")
```

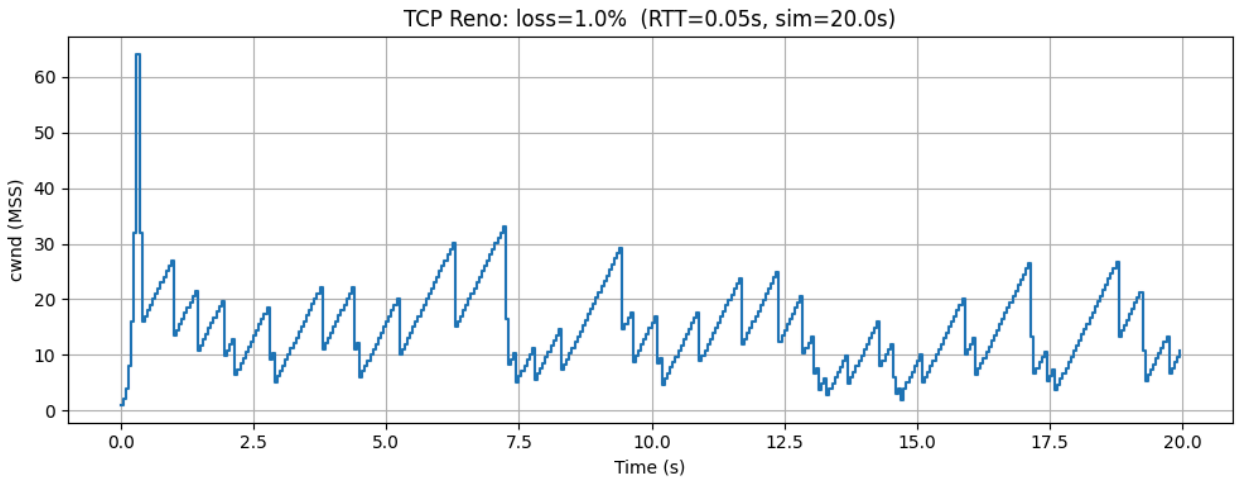
```
plt.legend()

plt.grid(True)

plt.tight_layout()

combined_fname = output_dir / "cwnd_comparison.png"

plt.savefig(combined_fname, dpi=150)

plt.show()

plt.close()




print("Simulation summary:")

for loss, ts, cwnd, thr_mss_s, thr_mbps in results:

    pct = {0.001: "0.1%", 0.01: "1.0%", 0.02: "2.0%"}.get(loss,
f"{loss*100:.3f}%")

    print(f"- Loss = {pct:<6} | Avg throughput = {thr_mss_s:.3f} MSS/s  |
{thr_mbps:.3f} Mbps")


report_doc = Document()

report_doc.add_heading("TCP Reno Simulation Report", level=1)

report_doc.add_paragraph(f"Simulated for RTT={RTT}s, simulation
time={SIM_TIME}s, MSS={MSS_BYTES} bytes.")

report_doc.add_paragraph("Loss scenarios: 0.1%, 1%, 2%")


for loss, ts, cwnd, thr_mss_s, thr_mbps in results:

    pct = {0.001: "0.1%", 0.01: "1.0%", 0.02: "2.0%"}.get(loss,
f"{loss*100:.3f}%")

    report_doc.add_heading(f"Loss = {pct}", level=2)

    report_doc.add_paragraph(f"Average throughput: {thr_mss_s:.3f} MSS/s
({thr_mbps:.3f} Mbps)")
```

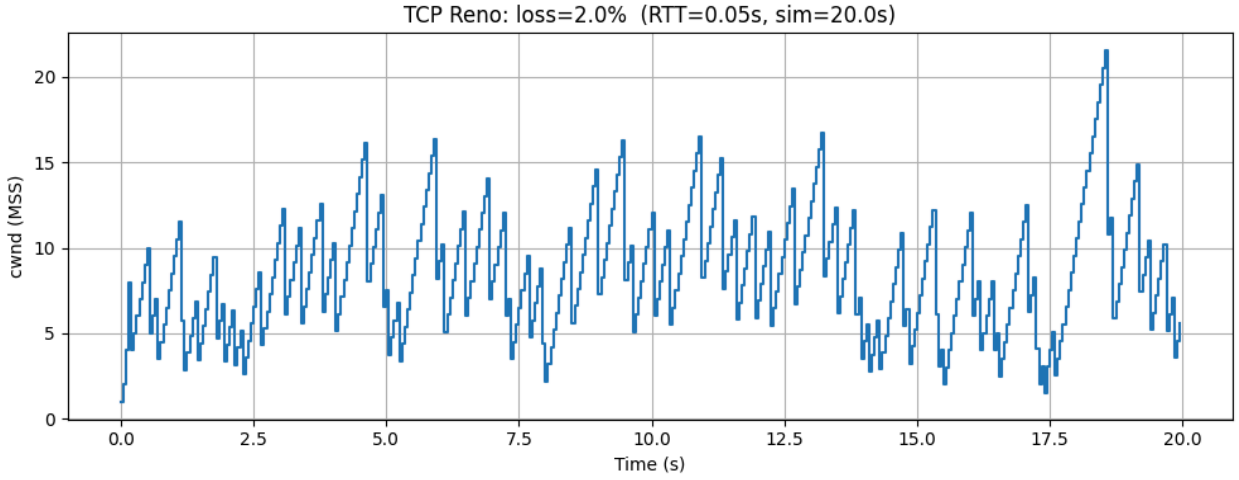
```
    obs = ("Observation: With higher loss the congestion window
experiences "
           "more frequent multiplicative decreases (cwnd halving),
reducing average cwnd and throughput.")
    report_doc.add_paragraph(obs)
    img_name = None
    if loss == 0.001:
        img_name = output_dir / "cwnd_0.1pct.png"
    elif loss == 0.01:
        img_name = output_dir / "cwnd_1pct.png"
    elif loss == 0.02:
        img_name = output_dir / "cwnd_2pct.png"
    if img_name and img_name.exists():
        report_doc.add_picture(str(img_name), width=Inches(6))


report_doc.add_heading("Comparison: cwnd vs Time", level=2)
if combined_fname.exists():
    report_doc.add_picture(str(combined_fname), width=Inches(6))



report_doc.add_heading("Conclusion", level=2)
report_doc.add_paragraph(


)


report_path = output_dir / "TCP_Reno_Report.docx"
report_doc.save(report_path)
print(f"\nSaved report to: {report_path}")
```

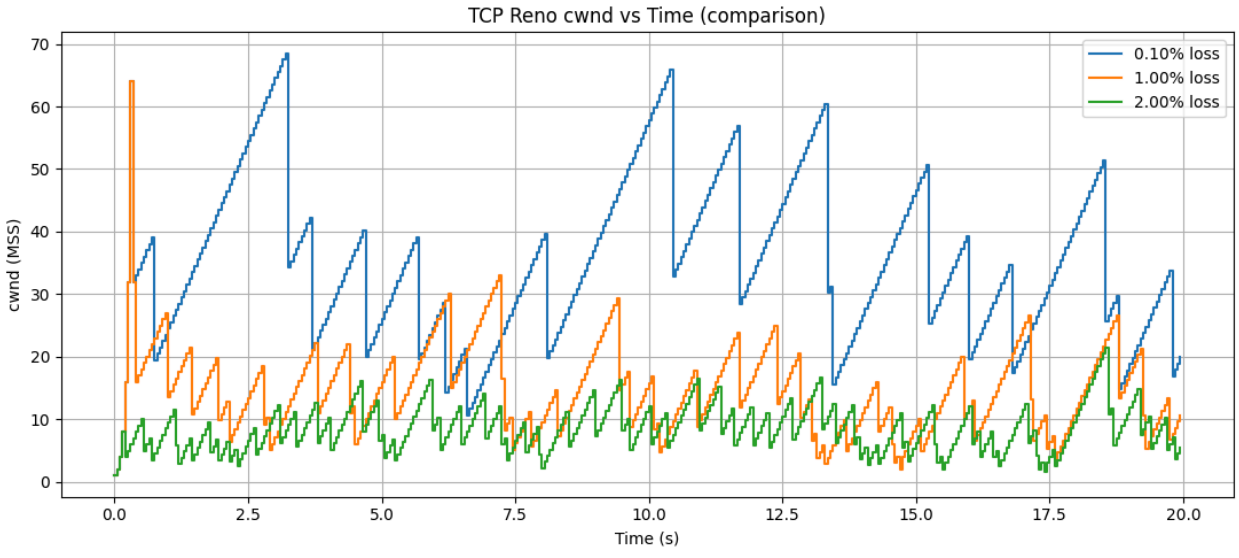
```
uploaded_path = Path("/mnt/data/Assignment-6.docx")
if uploaded_path.exists():
    shutil.copy(uploaded_path, output_dir / uploaded_path.name)
    print(f"Copied uploaded file to: {output_dir / uploaded_path.name}")
else:
    print(f"Uploaded path not found at {uploaded_path} (this path was
included per your session history).")
```

OUTPUT:









Simulation summary:

- Loss = 0.1% | Avg throughput = 687.200 MSS/s | 8.026 Mbps
- Loss = 1.0% | Avg throughput = 279.550 MSS/s | 3.265 Mbps
- Loss = 2.0% | Avg throughput = 156.800 MSS/s | 1.831 Mbps