ASSIGNMENT 2

BLG 337E Principles of Computer Communication

Prof. Dr. Abdül Halim Zaim

R&T Assistant Gülizar Kondel

(kondel16@itu.edu.tr)

R&T Assistant Büşra Bayram

Due Date: January 1, 2026

Cross-Layer Performance Optimization of a Custom ARQ Protocol

This assignment requires to design, implement, and optimize a Selective Repeat Automatic Repeat reQuest (ARQ) protocol within a custom-built network simulator. The simulator must integrate a simplified Transport Layer, operate over a realistic burst-error physical channel, and evaluate performance based on end-to-end Goodput.

To ensure fairness, reproducibility, and comparability across all student submissions, **all baseline physical and system parameters are fixed and provided.**

PHASE 1: SYSTEM DESIGN & CROSS-LAYER IMPLEMENTATION

This phase focuses on building the core simulator and implementing a realistic cross-layer communication stack.

1. Cross-Layer Integration Requirements

Transport Layer Shim

A user interface must be developed to enable users to select:

- **Application Input:** Accepts a large data file (fixed at **100 MB)** from the application layer.
- Segmentation: Breaks the file into transport segments (Transport header size fixed at **8 bytes**).
- **Reassembly:** Reassembles segments at the receiver and verifies integrity.
- **Limited Receiver Buffer:**
  - Application-side buffer capacity is fixed at **256 KB**.
  - If this buffer is full, the Transport Layer must provide **backpressure** to the Link Layer (delayed ACKs, temporary halt of forwarding, etc.).

Selective Repeat ARQ (Link Layer)

The Selective Repeat implementation must include:

- Sending and receiving windows
- Per-frame timers
- Out-of-order buffering
- Selective retransmission
- Proper sliding-window behavior
- Link Layer header overhead fixed at **24 bytes per frame**

2. Physical Layer Model (Fixed Baseline Parameters)

All simulators must use the following shared physical-layer configuration.

Bit Rate: R = 10 Mbps

Asymmetric Propagation Delays:

Table 1: Asymmetric propagation delays used in the simulator

Direction	**Propagation Delay**
Forward path (Data frames)	40 ms
Reverse path (ACK frames)	10 ms
Processing delay (per frame)	2 ms

Burst Error Channel (Gilbert–Elliot Model):

Table 2: Gilbert–Elliot burst error model parameters

Parameter	**Value**
Good-state BER, p_g	1×10^{-6}
Bad-state BER, p_b	5×10^{-3}
Transition probability $P(G \to B)$	0.002
Transition probability $P(B \to G)$	0.05
Average target BER	$\approx 1 \times 10^{-4}$

Errors occur in bursts, and ARQ behavior must handle clustered losses, not only independent random errors.

3. Performance Optimization (Goodput Metric)

Goodput is the primary metric:

$$\text{Goodput} = \frac{\text{Delivered Application Bytes}}{\text{Total Transmission Time}}$$

Students must account for:

- Transport + Link headers
- Retransmission overhead
- Flow control effects
- Dynamic RTT
- Effective pipeline utilization

Parameter Space for Exhaustive Search

All students must evaluate the same design space:

- Send Window Size (W)

$$W \in \{2, 4, 8, 16, 32, 64\}$$

- Frame Payload Size (L)

$$L \in \{128, 256, 512, 1024, 2048, 4096\} \text{ bytes}$$

For each (W, L) pair:

- Run 10 simulations with different RNG seeds
- Total simulations $= 6 \times 6 \times 10 = 360$ runs

Required Output:
A 3D surface plot or heatmap showing:

$$\text{Goodput} = f(W, L)$$

The optimal point

$$(\mathbf{W}, \mathbf{L})$$

must be highlighted.

Students must also explain analytically:

- The trade-off between large frame size (lower overhead) and burst error sensitivity
- The trade-off between large windows (pipeline utilization) and burst-loss amplification
- Why the identified optimum exists

PHASE 2 — AI-ASSISTED OPTIMIZATION (USING GEMINI)

This phase investigates how AI tools can support analytical reasoning and protocol optimization.

A. AI-Assisted Parameter Refinement

Students must:

- Export their complete simulation results to a CSV containing:

  W, L, run_id, goodput, retransmissions, avg_RTT, utilization, buffer_events, ...
- Provide this dataset to Gemini and request:
  - Trend identification
  - Outlier detection
  - A mathematical hypothesis predicting optimal region
  - A refined search space for W and L
  - Additional insights on retransmissions, RTT behavior, and buffer dynamics
- Students must include all prompts and Gemini responses in the final report.

B. Code Critique & Protocol Improvement

Students must request a structured code review from Gemini, including:

- Timer handling
- ACK processing
- Buffer management
- Window advancement logic
- Handling of burst losses
- Timeout calculation strategy

Then:

- One significant protocol improvement must be implemented, such as:
  - Adaptive timeout mechanism
  - Piggybacked ACK optimization
  - Silly Window Syndrome prevention
  - ACK pacing or delayed ACK logic
  - Dynamic window adjustments under burst losses
- After implementing the improvement:
  - Re-run simulations at the predicted optimal (W, L)
  - Compare Goodput before vs. after the enhancement
  - Quantify the improvement (percentage increase)

Students must submit:

1. The Optimized Simulator Codebase
2. Complete Dataset (CSV)

   All 360 runs + improved protocol runs.
3. Goodput Visualization

   Heatmap or 3D surface showing Goodput(W, L).
4. AI Interaction Log

   Full prompt/response history from Gemini.
5. Final Comparative Report

   The report must include:
   - AI-generated theoretical prediction
   - Experimental optimal parameters for the original protocol
   - Performance improvement from the enhanced protocol
   - Interpretation grounded in cross-layer analysis
   - A critical reflection on strengths and limitations of AI-assisted engineering workflows