ECE 50863
Project 2 Report Template

Yuan-Yao Lou / lou45@purdue.edu

Submit as a pdf, labeled <firstname.lastname.Project2.first.pdf>, <firstname.lastname.Project2.second.pdf> or <firstname.lastname.Project2.final.pdf>

Instructions

- Please follow this template for all milestones, but update the material with each milestone.

- Answers should be typed. Handwritten documents are not permitted.

- Your report should be submitted in **pdf format only.**

- Please keep your answers to the point.

- Graphs should be carefully plotted, with the **X and Y axis clearly labeled with appropriate legends.**

- For a slide with a graph have a 1 line **"take-home" message (what does the result show?)**

- Avoid 2 graphs on the same slide to ensure clarity.

- For all graphs, **use the file "to_send_large.txt"**

Results for Stop and Go

- Use three configurations. For all three:
  - Use one way propagation delay of 100 ms.
  - Set **Modest loss (2%)** and **modest reordering (2%).**
  - Vary bandwidth as: **200000 bytes/sec, 20000 bytes/sec, 2000 bytes/sec**
- Graphically show both overheads and goodput (efficiency) in two separate graphs. **Insert 2 separate slides following this.**
- Suggested format:
  - Goodput: Bar Chart with 3 bars, one for each bandwidth value.
  - Overhead: Bar Chart with 3 bars, one for each bandwidth value.

Results for Stop and Go - Goodput

**Goodput increases as bandwidth increases.**

The large gap between BW=2000 and BW=20000|200000 is caused by the timeout values. Since lower bandwidth has smaller MAX_PACKETS_QUEUED value, so I set larger timeout values when BW=2000, leading to longer "Total Time". Besides, Goodput also depends on the frequency of dropping packets. Dropping packets too often leads to longer "Total Time".
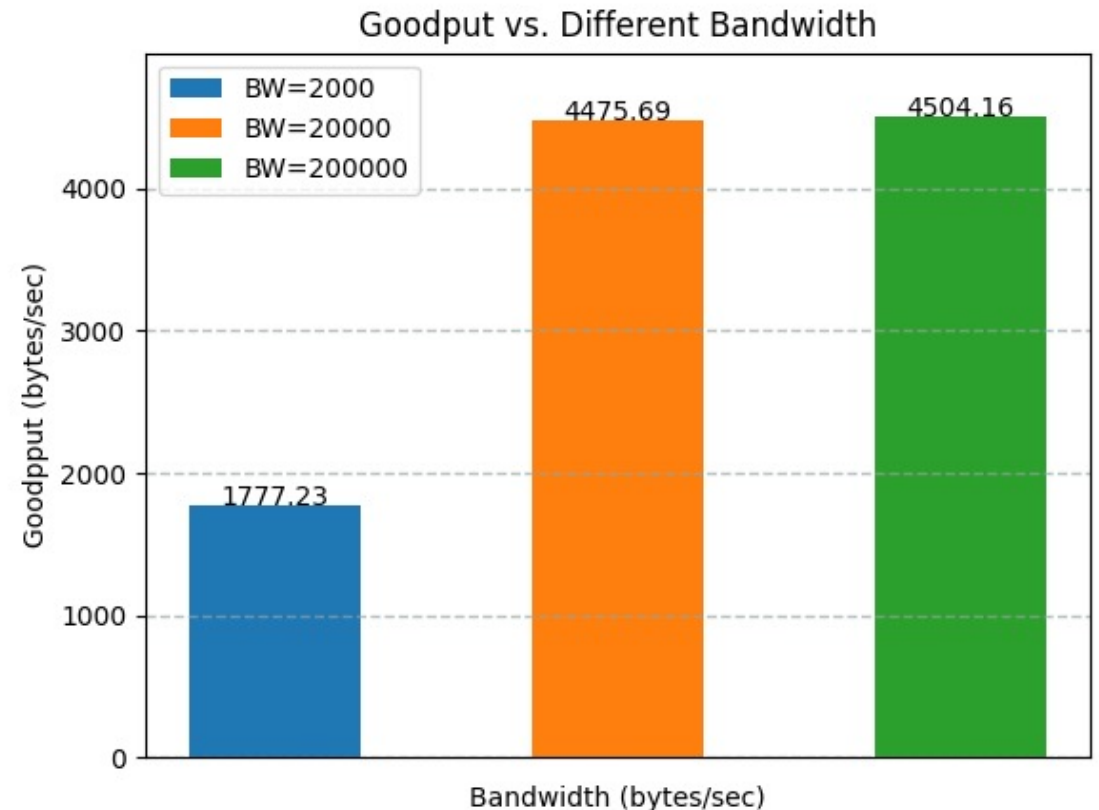
Results for Stop and Go - Overhead

Overhead decreases as bandwidth increases.

Overhead is mainly affected by the retransmission times (higher number of retransmissions induce higher "Total Bytes Transmitted"). Thus, overhead value depends on the frequency of dropping packets. Properly set the timeout values could avoid extra retransmissions, reducing the Overhead.
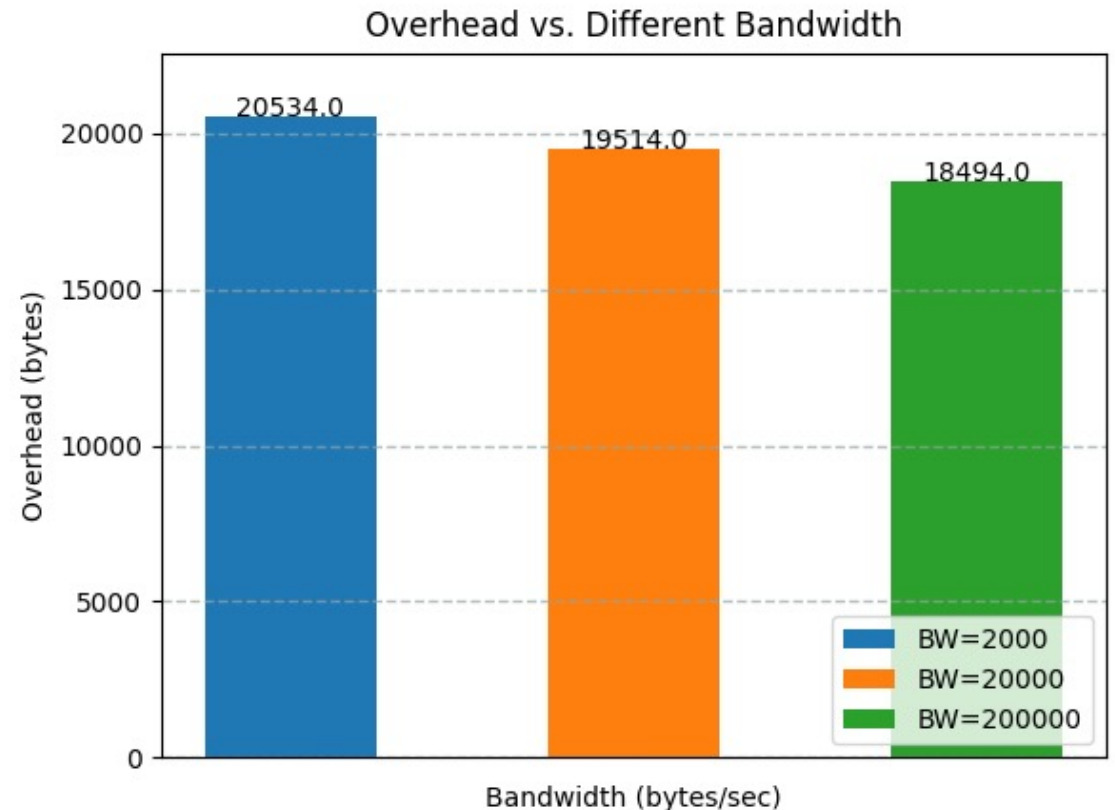
Document your design decisions.

- Please answer each question in the following slide in 1-2 lines each.
- For initial report, answer **what you propose to do and say (Proposed)** to indicate this has not been implemented.
- In the interim report, **update the answers to indicate what you have completed, and what you still propose to do.**
- In the final report, update the answers to indicate what you finally implemented.

Designs Decisions [Ok to split this on 2 slides]

- **How did you set your timeout values?**
  I set timeout values based on 1.65x RTT value with lower bound 0.35s
  = max(0.35, 1.65*(MAX_PACKET_SIZE/LINK_BANDWIDTH + 2*PROP_DELAY))

- **What window size did you use? Why?**
  I set window size to 40 statically by using delay bandwidth product since it is proved to be efficient by the theory.

- **Does your code automatically adjust timeout and window size to network configuration? If so how?**
  I do adjust the timeout automatically by RTT value as described above. For window size, since this stage's experiment uses static link bandwidth, I fix the value to 40 in my code. However, the value is set based on the delay bandwidth product, which is affected by network configuration.

Designs Decisions [Ok to split this on 2 slides]

- **What ACK scheme did you use? Cumulative/Selective/Both? Provide a couple of lines of detail about your ACK scheme.**
I use selective ACK scheme. After I sent out all the packets in the current window, I wait for all expected ACK packets to arrive. If timeout happens, I retransmit packets based on unACKed ones in the current window.

- **Does the sender only retransmit on a timeout, or does any other condition trigger a retransmission in your implementation?**
Yes, the sender only retransmits on a timeout.

- **Please list any other optimizations not listed above that you implemented.**
To ensure correctness, I pre-process the data transmitted by the sender so that sender knows when the transmission is finished. Thus, the sender could inform the receiver by the end and wait for the last ACK message to arrive.
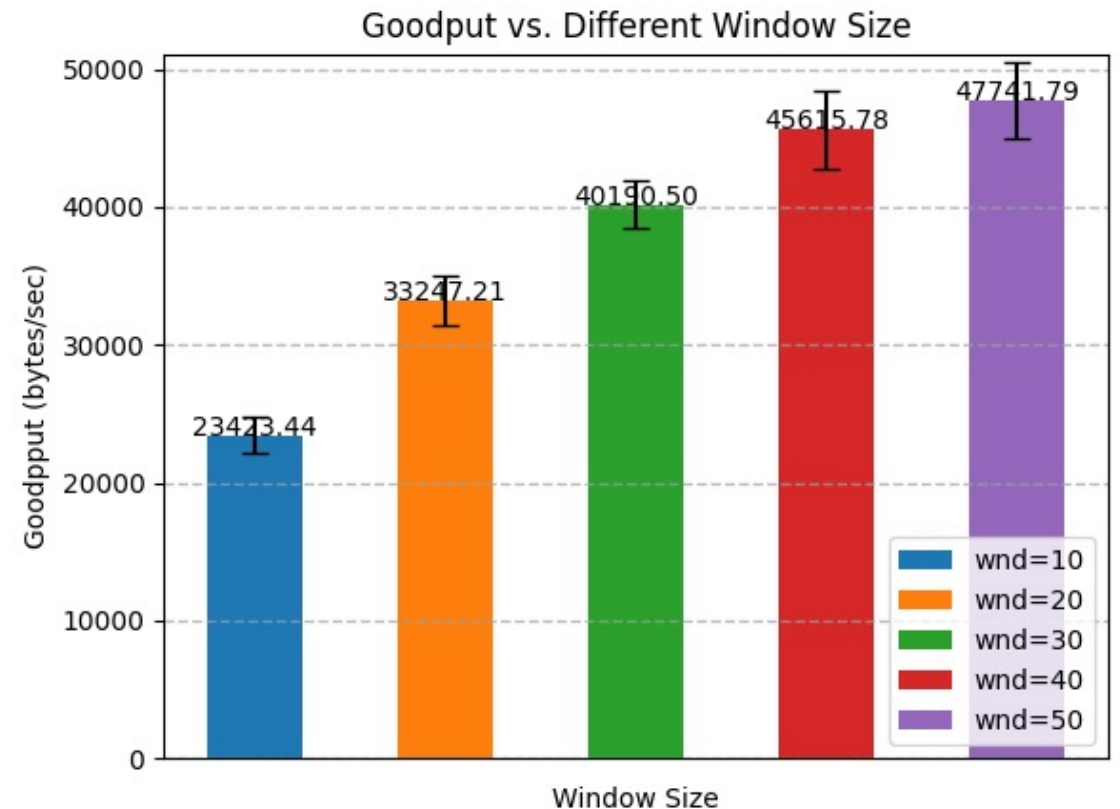
Results with your protocol: Impact of window size

- Use this configuration:
  - Bandwidth: 200000 bytes/sec
  - One way propagation delay of 100 ms.
  - Modest loss (2%) and modest reordering (2%)
- Graphically **show the goodput and overheads for different window size choices.** Insert 2 slides after this one.
- Suggested format:
  - Goodput: Bar Chart: 5 different window sizes, 1 bar per window size.
  - Overhead: Bar Chart: 5 different window sizes, 1 bar per window size.
  - **Pick window sizes to cover interesting points, and show the right trend.**

Results with your protocol: Goodput

Goodput increases as window size increases.

Although window size 50 has the highest goodput, it does have tradeoff between goodput and overhead.

Results with your protocol: Overhead

Overhead remains almost the same except the window size 50.

I retransmit the unACKed packets in the current window, so the overhead only depends on the probability of dropping packets (= 2%).

However, when the window size is larger than 40, it might cause the buffer to be full and lead to extra packet drop and higher overhead. Besides, the standard deviation is larger.
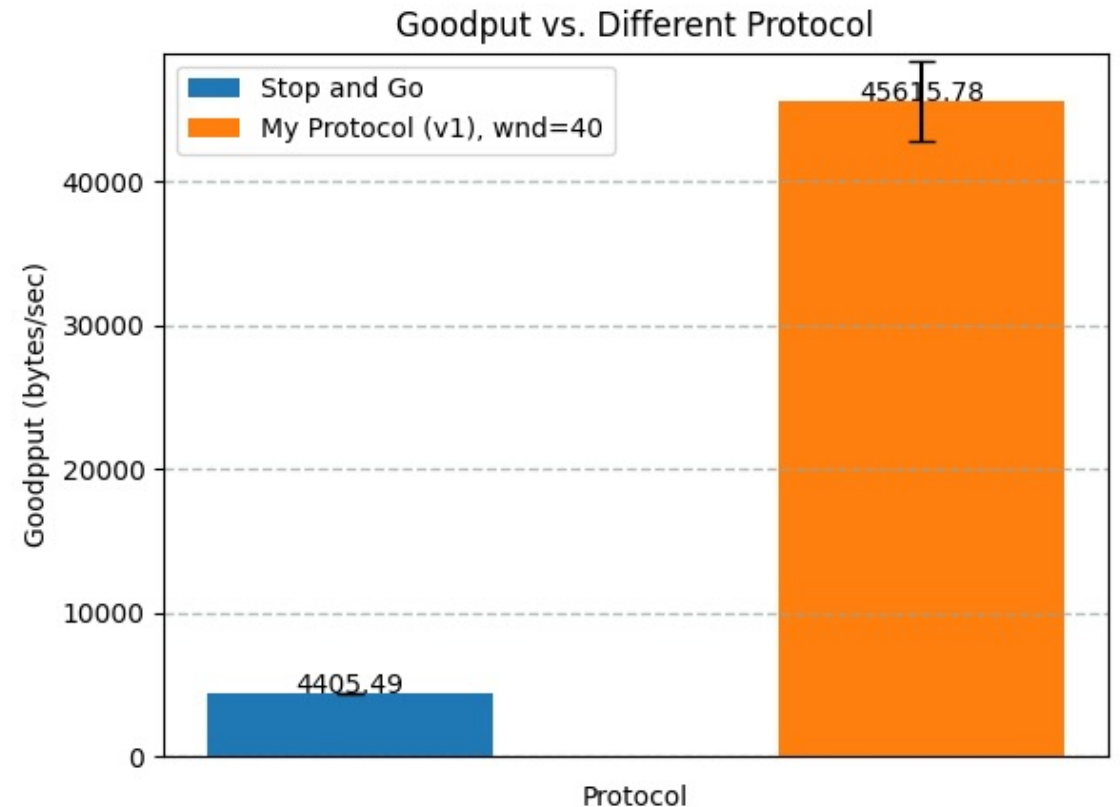

Overhead vs. Different Window Size
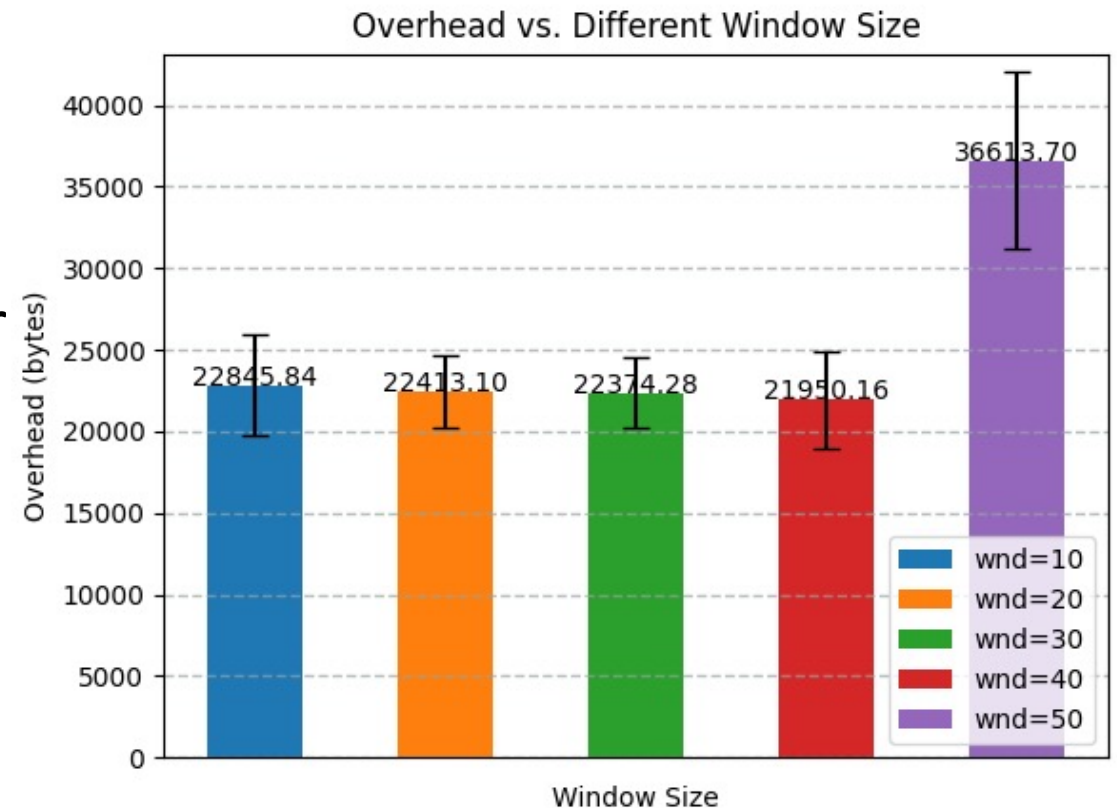
Results: Your protocol Vs. Stop and Go

- Use this configuration:
  - Bandwidth: 200000 bytes/sec
  - One way propagation delay of 100 ms.
  - Modest loss (2%) and modest rebuffering (2%)
  - **Pick what you think is the best window choice for your scheme.**
- Graphically compare (a) your final protocol (b) your interim protocol and (c) Stop and Go with respect to goodput and overheads (insert 2 slides, one for each metric).
- Suggested format:
  - Bar Chart: 3 bars. The bars correspond to the 3 protocols.
  - Interim report need **have only 2 bars (for Stop and Go and your interim protocol),** and you can update with additional bar (for final protocol) in the final report.
  - For the best credit on this, **please run each experiment multiple times** and include the **mean and error bar.** This is because performance can be variable across multiple experiments.

Results: Your protocol Vs. Stop and Go - Goodput

My protocol is significantly better than the protocol "Stop and Go" (around 10.35x)

Results: Your protocol Vs. Stop and Go - Overhead

My protocol is slightly better than the protocol "Stop and Go", but the standard deviation is much more stable.

The overhead of "Stop and Go" strongly relates to the timeout value. Since packet transmission time varies across different experiment iterations, that's why "Stop and Go" has a higher standard deviation.
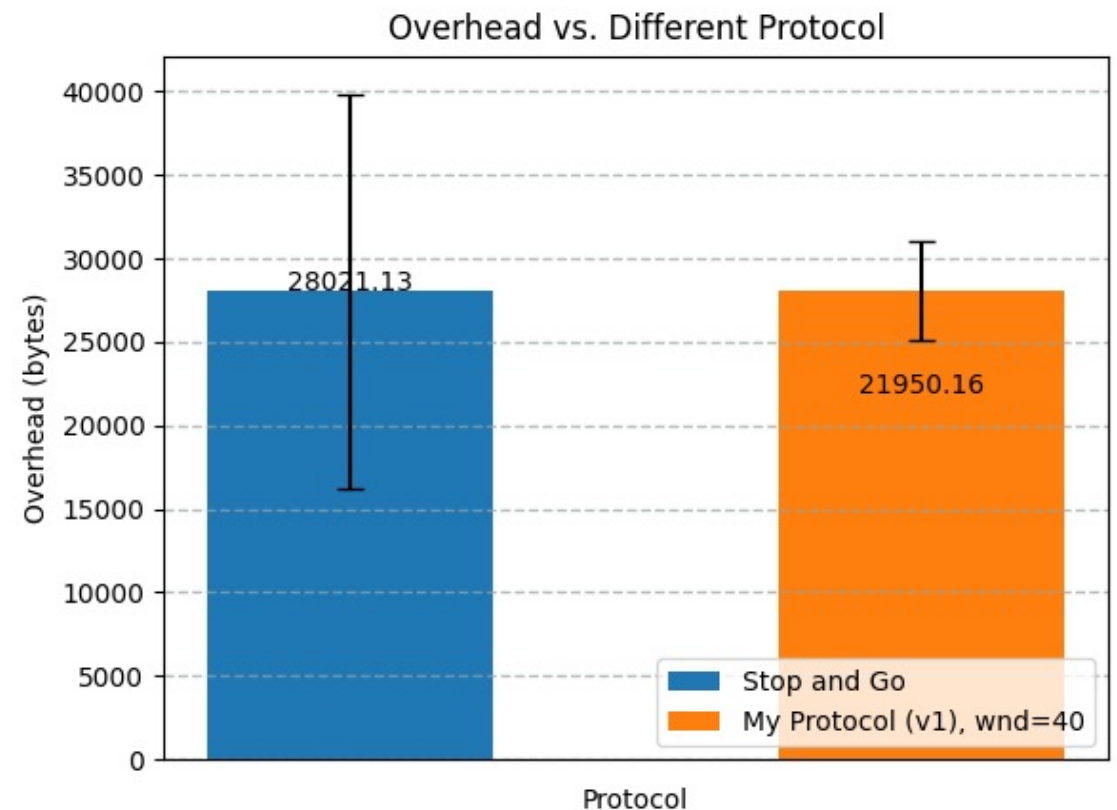
Sensitivity Testing [Final Report only]

- Use this configuration:
  - Bandwidth: 200000 bytes/sec
  - One way propagation delay of 100 ms.
  - Pick what you think is the best window choice for your scheme.
  - Pick 3 configurations:
    - Higher loss (5%) and higher reordering (5%)
    - Modest loss (2%) and modest reordering (2%)
    - No loss, and no reordering
- Graphically compare your final protocol and Stop and Go with respect to goodput and overheads (insert 2 slides, one for each metric).
- Suggested format:
  - Bar Chart: 3 groups of 2 bars each. Each group corresponds to a configuration. In each group, the 2 bars correspond to Your Protocol, and Stop and Go respectively.

Discussion/Conclusion - For Interim Milestone

- For Interim Milestone, **do you think your results are satisfactory? How do you plan to improve results for the final report?**

  I strongly believe that my results could be improved, for both goodput and overhead. Waiting for all packets in the current window to be ACKed wastes time. I plan to keep moving my window once receiving ACK from the receiver.

  However, for the packets that do not receive corresponding ACK, I need to make sure receiver does receive them correctly.

Discussion/Conclusion - For Final Milestone

- For Final milestone:
  - Any interesting findings? Were particular optimizations particularly useful? Were some optimizations not as useful?
  - Optimizations you should try in the future but could not get to?