LAN Reliability Report

Executive Summary

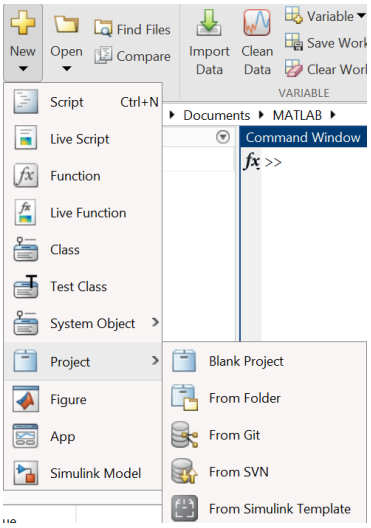
This project explored how many attempts are needed to send messages made up of K packets through various network setups using MATLAB. I tested a single link, two links in series, two parallel links, a compound network, and a custom compound network with different failure probabilities for each link. I used K = 1, 5, 15, 50, 100 (K = 1, 5, 10 for the last part), with failure probabilities from 0 to 0.99, running each test 1000 times for accurate averages. The results are shown in graphs with logarithmic y-axes for clarity, calculated results as solid lines and simulated ones as hollow circles, all color-coordinated.

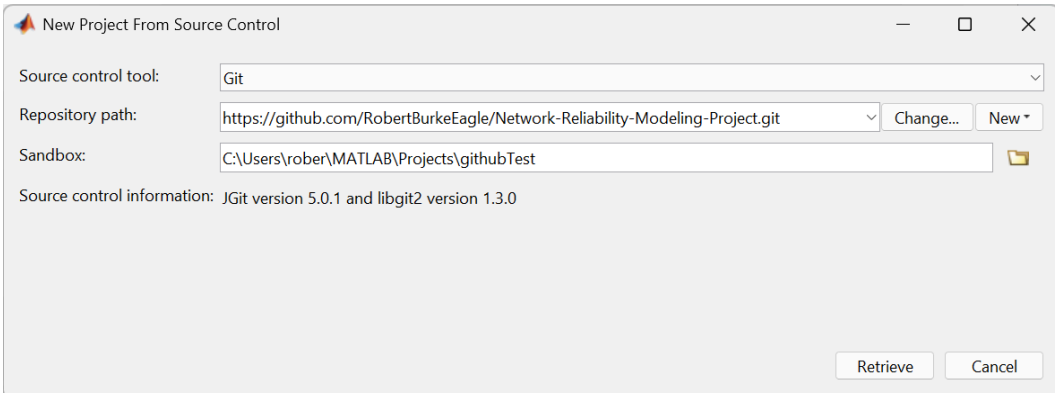
Key takeaways include:

- Single Link: Simulated results closely matched the math, especially with larger K.
- Two Links in Series: Required more tries since both links must succeed.
- Two Parallel Links: Fewer tries needed since only one link has to work.
- Compound Network: A mix of challenges, balancing series and parallel effects.
- Different Probabilities: Changing failure rates showed some links create bigger bottlenecks.

The code is my public GitHub repository at https://github.com/RobertBurkeEagle/Network-Reliability-Modeling-Project.

Instructions to Run:

1. Open Matlab and go to new -> project -> from Git
   a. 

2. Clone the repository: https://github.com/RobertBurkeEagle/Network-Reliability-Modeling-Project.git
    a. 

3. After that the project will open and you can begin running the models

Task 1: Single-Link Network

What I Did

I used the runSingleLinkSim.m function to figure out how many tries it takes to send K packets through one link that might fail with probability p. The math suggests an average of K / (1 - p) tries. I ran 1000 tests for K = 1, 5, 15, 50, 100 and p from 0 to 0.99 (step 0.01). Here's a snippet of the code:

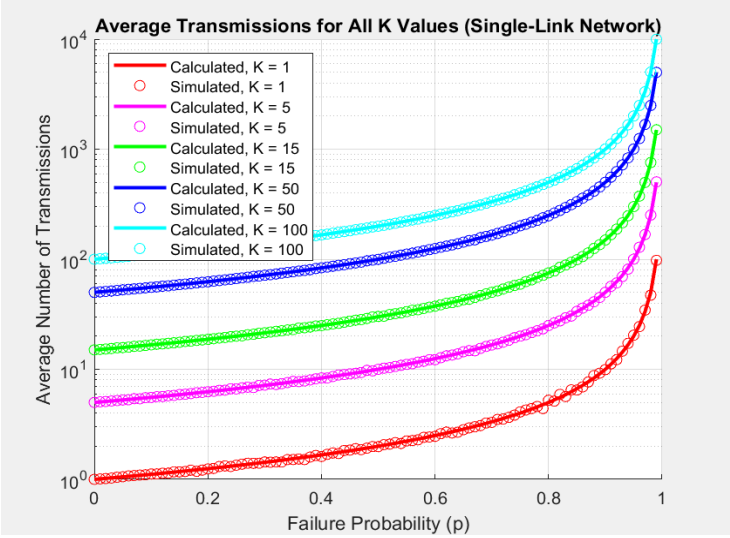
```
for k_i = 1:length(K_values)
    K = K_values(k_i);

    for p_i = 1:length(p_values)
        p = p_values(p_i);

        % Theoretical expected transmissions (single link)
        calculated_results(k_i, p_i) = K / (1 - p);

        % Simulated result using Appendix A function
        simulated_results(k_i, p_i) = runSingleLinkSim(K, p, N);
    end
```

Expected Output

- Individual plots per K showing calculated results as solid lines and simulated results as hollow circles.
- A combined plot overlaying all K values.
- Logarithmic y-axis for readability.

What It Showed

Each K got its own graph with a solid line (math) and hollow circles (simulated), plus one combined graph. The logarithmic y-axis made it easy to see the big jump in tries when p was high. The simulated results were really close to the math, especially for bigger K.



Task 2: Two-Series-Link Network

What I Did

For this, I had packets go through two links in a row, both with failure chance p. The math says it takes K / ((1 - p)^2) tries. I used runTwoSeriesLinkSim.m to simulate this, testing the same K values and p from 0 to 0.9 (step 0.1), with 1000 runs each. The code looked like this:

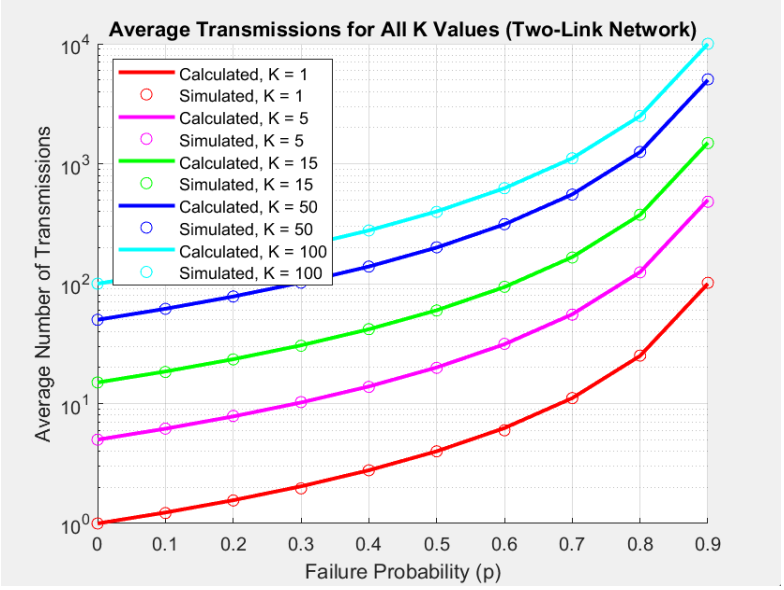
```
for k_i = 1:length(K_values)
    K = K_values(k_i);
    for p_i = 1:length(p_values)
        p = p_values(p_i);

        % Calculated expected transmissions (two links in series)
        calculated_results(k_i, p_i) = K / ((1 - p)^2);

        % Simulated result
        simulated_results(k_i, p_i) = runTwoSeriesLinkSim(K, p, N);
    end
```

Expected Output

- Individual plots per K with calculated vs simulated results.
- A combined plot overlaying all K values.
- Logarithmic y-axis.

What It Showed

Each K got a graph with solid lines (math) and hollow circles (simulated), plus a combined graph. The logarithmic y-axis showed a steep rise in tries when p got big. The simulated results followed the math well, with a bit more variation at higher p.



Task 3: Two-Parallel-Link Network

What I Did

Here, I had packets go through two links at once, succeeding if either worked. The math suggests K / (1 - p^2) tries. I used runTwoParallelLinkSim.m to test this for the same K and p values (0 to 0.99, step 0.01), running 1000 times:

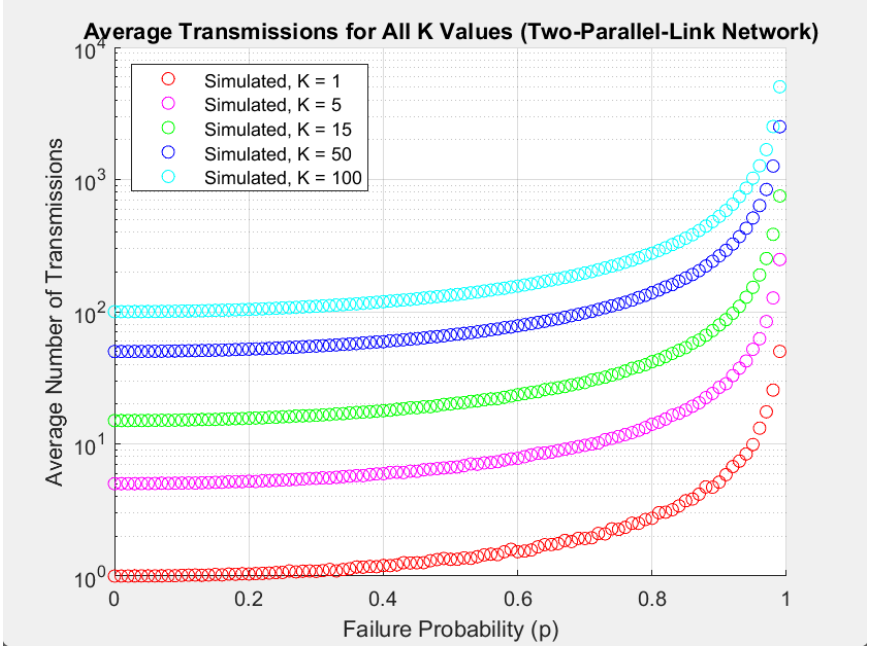
```
% Run simulations
for k_i = 1:length(K_values)
    K = K_values(k_i);
    for p_i = 1:length(p_values)
        p = p_values(p_i);
        simulated_results(k_i, p_i) = runTwoParallelLinkSim(K, p, N);
    end
```

Expected Output

- Individual and combined plots for all K values.
- Logarithmic y-axis.

What It Showed

Each K got a graph with hollow circles for simulated results, plus a combined graph. The logarithmic y-axis showed fewer tries than the series setup, which made sense since only one link needed to work.



Task 4: Compound Network

What I Did

This network has one link (A) followed by two parallel links (B and C). A packet needs A and at least one of B or C to succeed. I used runCompoundNetworkSim.m to simulate this for the same K and p values, running 1000 times:

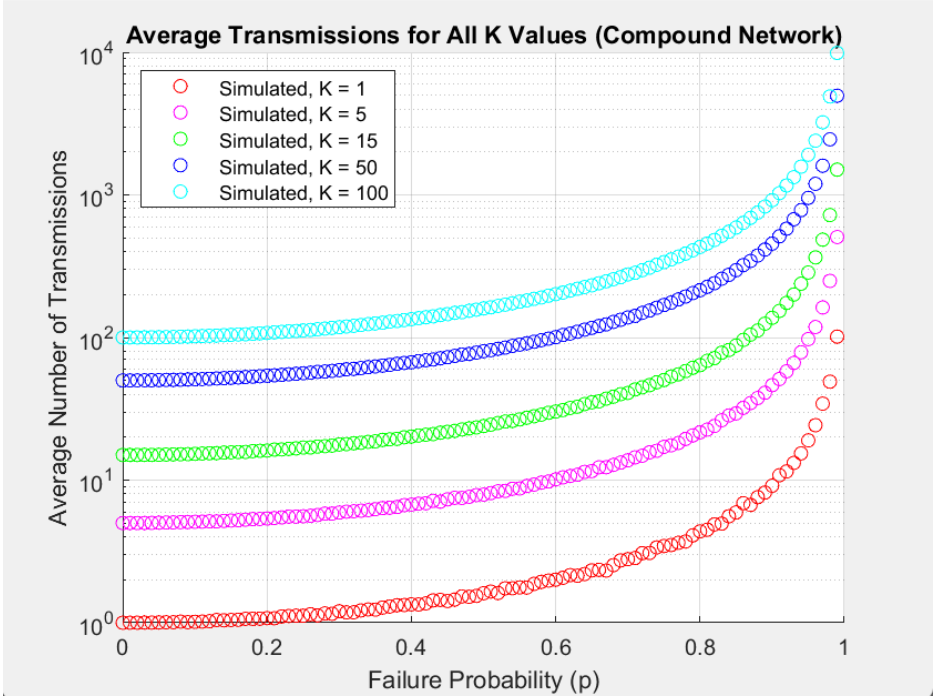
```
for k_i = 1:length(K_values)
    K = K_values(k_i);
    for p_i = 1:length(p_values)
        p = p_values(p_i);
        simulated_results(k_i, p_i) = runCompoundNetworkSim(K, p, N);
    end
```

Expected Output

- Individual and combined plots for all K values.
- Logarithmic y-axis.

What It Showed

Each K got a graph with hollow circles, plus a combined graph. The logarithmic y-axis showed more tries than the parallel setup but fewer than the series one, with a noticeable increase when p was high.



Task 5: Compound Network with Different Probabilities

What I Did

Now, each link (A, B, C) had its own failure chance (p1, p2, p3). I updated runCustomCompoundNetworkSim.m and created six graphs, varying one probability while fixing the others (e.g., p1 = 0.1, p2 = 0.6, p3 from 0.01 to 0.99) for K = 1, 5, 10. Here's a bit of the code:

```
    % Run simulations
    for k_i = 1:length(K_values)
        K = K_values(k_i);
        for p_i = 1:length(p_vals)
            p_var = p_vals(p_i);
            [p1_val, p2_val, p3_val] = deal(p1, p2, p3); % Default to fixed values
            if var_param == 'p1', p1_val = p_var;
            elseif var_param == 'p2', p2_val = p_var;
            else, p3_val = p_var;
            end
            results(k_i, p_i) = runCustomCompoundNetworkSim(K, p1_val, p2_val, p3_val, N);
        end
    end

    % Store results
    simulated_results{fig_i} = results;
end
```

Expected Output

- Six figures showing the effect of varying probabilities for K = 1, 5, 10.
- Logarithmic y-axis.

What It Showed

The six graphs showed simulated results (hollow circles) for different setups, with logarithmic y-axes. When the first link (p1) had a high failure chance, it slowed things down a lot. Changing the parallel links (p2 or p3) had less impact due to the backup.

Conclusion

This project showed how different network setups handle packet failures. Single and series links struggled more with high failure rates, while parallel links were more forgiving. The compound network was a mix, and adjusting failure chances highlighted key bottlenecks. The logarithmic graphs helped spot these trends easily.