Networking for Big Data - Data Center

Homework 1

Group 30 - Kleinrock

Chiara Sammarco 1913440
Elisa Pierini 1859043
Giacomo Bellini 1970896

17/04/2023

1 Part 1

- **Complexity versus number of nodes K, for the three connectivity checking algorithms:**

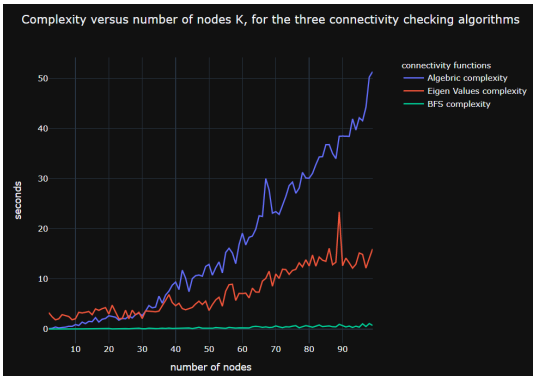

Figure 1: By looking at the plot we can see that the best performing one is the BFS algorithm that is stable as **K** increases.

- **Probability of a connected ER random graph as a function of p for K = 100 nodes**

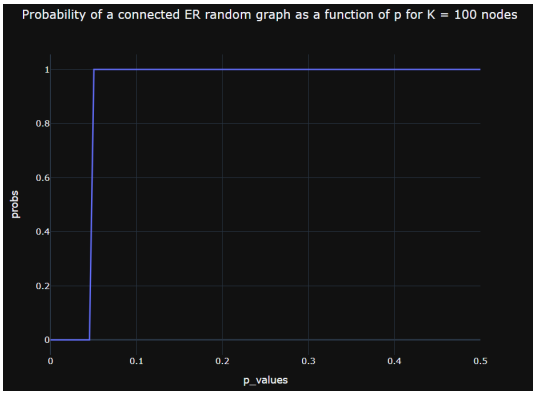

Figure 2: As the p value increases, the probability of the ER graph to be connected increases as well until 1.

- **Probability of a connected r-regular random graph as a function of K for r = 2 and 8.**

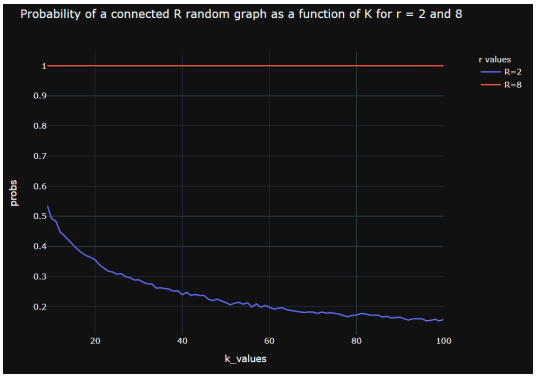

Figure 3: As the number of nodes increase, the probability of the r-regular random graph to be connected is fixed at 1 when the with r (degree) = 8, instead with r=2 the probability, as expected, slowly goes down to 0.

2 Part 2

Mean response time and job running cost:
Given our parameters: $C = 10Gbit/s, \tau = 5\mu_s, L_f = 4TB, L_o = 4TB, E[X] = 8hours, T_0 = 30sec, \xi = 0.1, f = 48/1500, n = 64$. Given the computation time X = negative exponential random variable with E[X] = 8 hours and L_{oi} = uniform random variable $[0, 2\ \frac{L_o}{N}]$ as our random generated output file size, we did **20000 Montecarlo simulations** for every server number. We generated the random computation time X together with the random output file size and computed the average throughput for all the servers $\mathbf{E[T}_i] = \left(C \times \frac{1/Ti}{\sum_{i=1}^{N}(1/Ti)}\right)$. Considering the **RTT** of every server, we calculated the **time transmission** from and to each server, adding the overhead **TCP** and taking the normalizing mean w.r.t. the baseline. To calculate the response time **R**, we summed up T_0 with the job-time of the servers L_a and the transmission times T_{from} and T_{to}, where $T_{from} = \frac{(Lo_i)+(f*Lo_i)}{E[T_i]}$ and $T_{to} = \frac{(L_f/N)+(f*L_f/N)}{E[T_i]}$. Given $L_a = T_0 + X$ as the computational job time of A, the **average response time** is equal to $T_0 + mean(L_a) + mean(T_{to}) + mean(T_{from})$. The normalized average response time is given by taking the mean from all the simulations response times normalized by $E[R_{baseline}]$ which is the average response time for N=1.

- **Normalized mean response time as a function of N for fat-tree and jellyfish topology:**

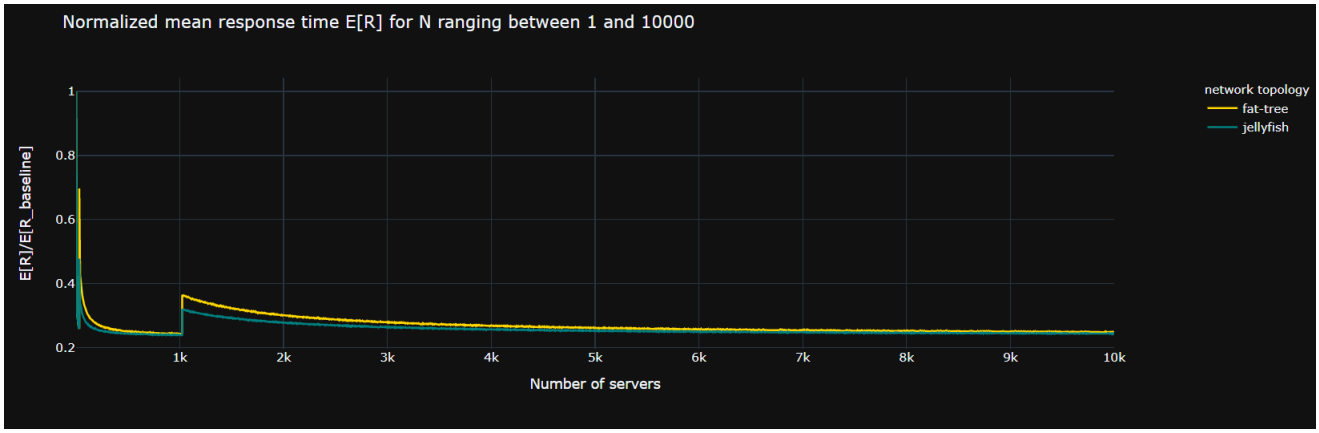

Figure 4: Normalized mean response time for fat-tree and jellyfish topology

It can be seen from the image that the Jellyfish topology as the number of servers increases has a lower average response time than the Fat tree type. Both after passed a threshold of 4500 servers settle on a constant response time.

- **Normalized job running cost for fat-tree and jellyfish topology:**

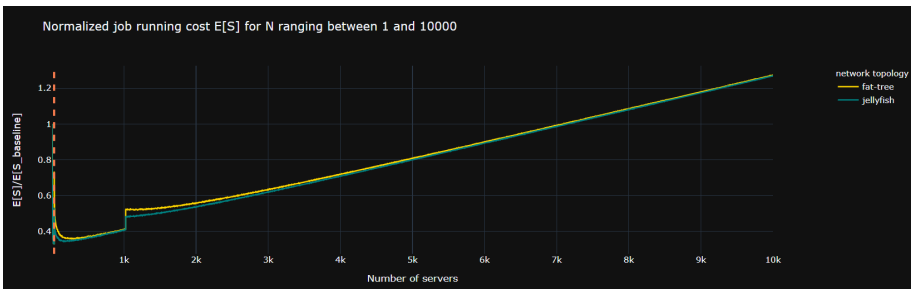

Figure 5: Normalized job running cost for fat-tree and jellyfish topology

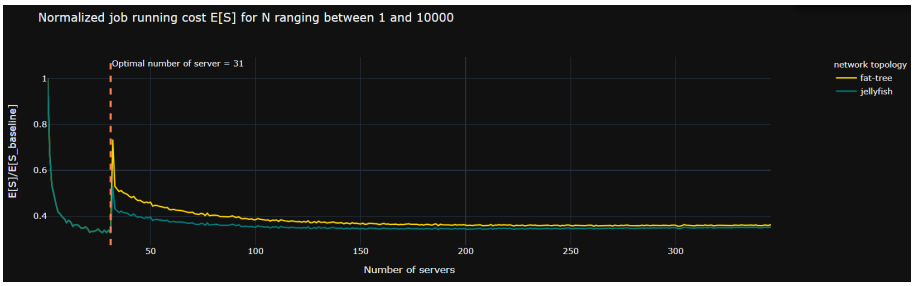

Figure 6: Normalized job running cost for fat-tree and jellyfish topology

- **Analysis of results and takeaways**

As shown in figure 5 and 6, the average response time in the case of two data center network topologies: Jellyfish and Fat tree, with 64-port switches. In the case of the Fat tree we can see that the average response time is slightly higher due to the characteristics of the topology. With 64-port switches we have 1024 switches on the core layer and 64 pods with 32 aggregation and boundary switches inside respectively connected to 32 servers. This shows us that the number of hops required to reach the various servers from server A are 2 to server number 31, 4 to 1024, and 6 to 10000. In the Jellyfish topology, on the other hand, server A is connected to a switch that is also connected to 32 different switches. Therefore in this case the number of hops results to be two up to server number 31, 3 up to server 1024 and 4 up to 10000. To calculate the job running cost, we took the previous results and we added up the average job running time performed by all servers. The final formula: $S = E[R] + \xi * E[\theta]$ where Θ is the time that server A is used, if the job runs locally on A. Otherwise, Θ is the sum of the times that all N servers are used to run their respective tasks. Figure 5 shows the job running cost normalised to the value obtained for N = 1, cost initially decrease and reaches its minimum with 31 servers for the two topologies. After 1200 servers the normalized job running cost starts to increase linearly.