Assignment 2- **Jammers**

AM.EN.P2CSN18001

Details of Code:

- The Assignment has been implemented with the help of 4 programs
  - **'inc_tcl.tcl'** to implement constant jamming
  - **'ran_tcl.tcl'** to implement random jamming
  - **'per_tcl.tcl'** to implement periodic jamming
  - **'jamgen.py'** to call the tcl scripts and set the number of nodes
- The assignment is executed by running the python code
  - Syntax: python jamgen.py <number of nodes>

jamgen.py:

- This program is called first. It is responsible for running all the tcl scripts with varying number of malicious nodes and calculated the throughput and delay for each case and create graph file.
- Output files created by jamgen.py include:
  - out_const : stores the throughput for constant jamming for varying percentage of malicious nodes.
  - out_ran : stores the throughput for random jamming for varying percentage of malicious nodes.
  - out_per : stores the throughput for periodic jamming for varying percentage of malicious nodes.
  - const_delay : stores the delay for constant jamming for varying percentage of malicious nodes.
  - ran_delay : stores the delay for random jamming for varying percentage of malicious nodes.
  - per_delay : stores the delay for periodic jamming for varying percentage of malicious nodes.
- Each of these files can be plotted using 'xgraph'

Constant Jamming:

- Implemented using 'inc_tcl.tcl' script.

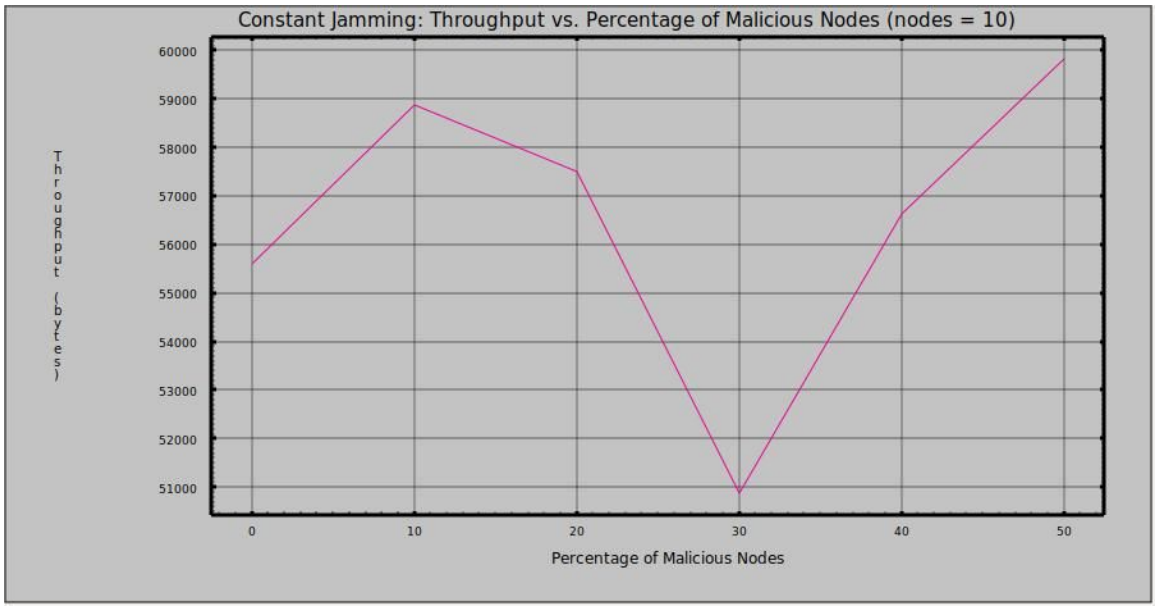

Figure: Graph showing "Constant Jamming: Throughput vs. Percentage of Malicious Nodes (nodes = 10)"

- When the malicious nodes start jamming the benign nodes, the throughput of the benign nodes decreases drastically as seen through the drop between 10-30 percentage of malicious nodes.
- After the percent of malicious nodes rise above 30, the benign nodes start dropping the packets of the malicious nodes due to which the throughput goes back up.

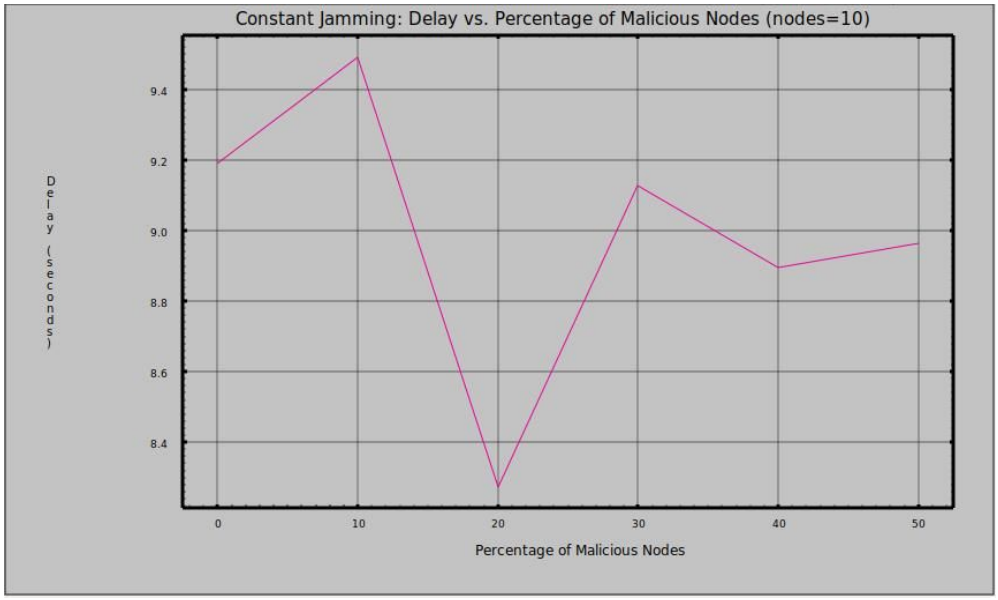

Figure: Graph showing "Constant Jamming: Delay vs. Percentage of Malicious Nodes (nodes = 10)"

- As you see, the delay due to constant jamming shows a decrease between 10 - 20 percent of malicious nodes before going on a rise.

Periodic Jamming:

- Implement using ‘per_tcl.tcl’ script

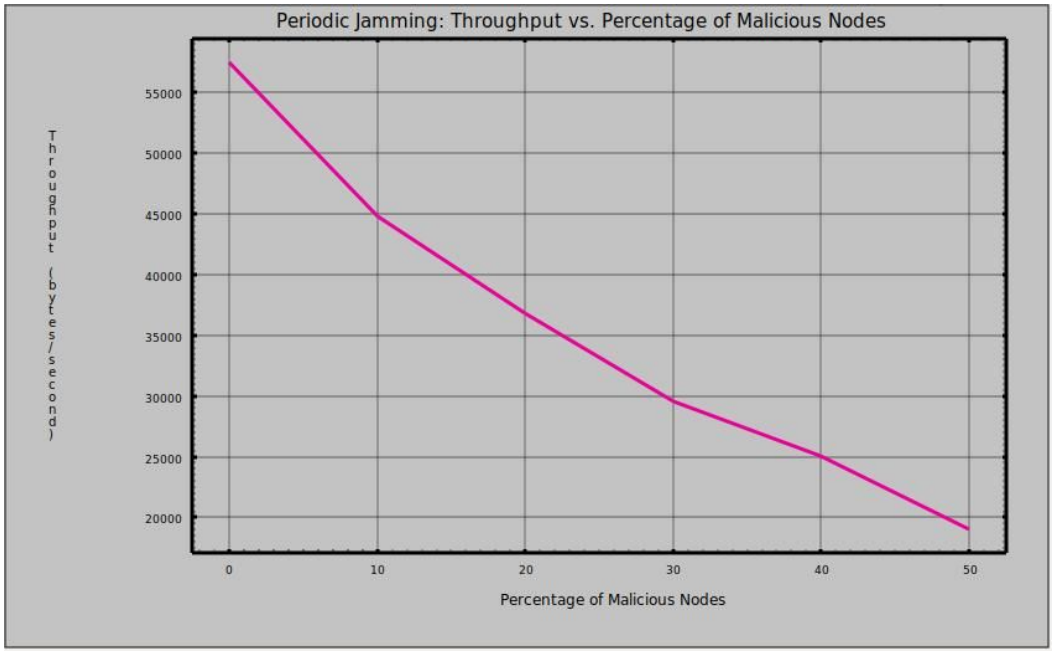

Figure: Graph showing “Periodic Jamming: Throughput vs. Percentage of Malicious Nodes (nodes = 10)”

- Steady decrease in throughput due to increase in the number of malicious nodes.

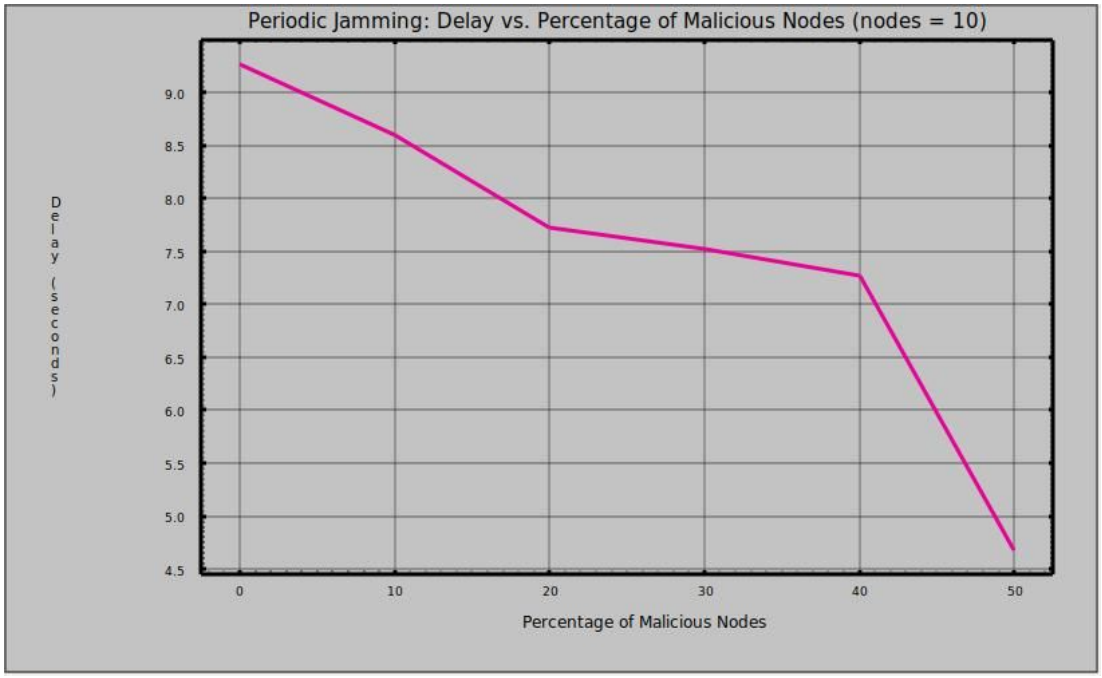

Figure: Graph showing “Periodic Jamming: Delay vs. Percentage of Malicious Nodes (nodes = 10)”

- Graph shows decrease in delay with increase in no. of malicious nodes. This indicates that an increase in percentage of periodic jammer nodes make the entire jamming process less effective due to interference.

Random Jamming:

- Implement using ‘ran_tcl.tcl’ script

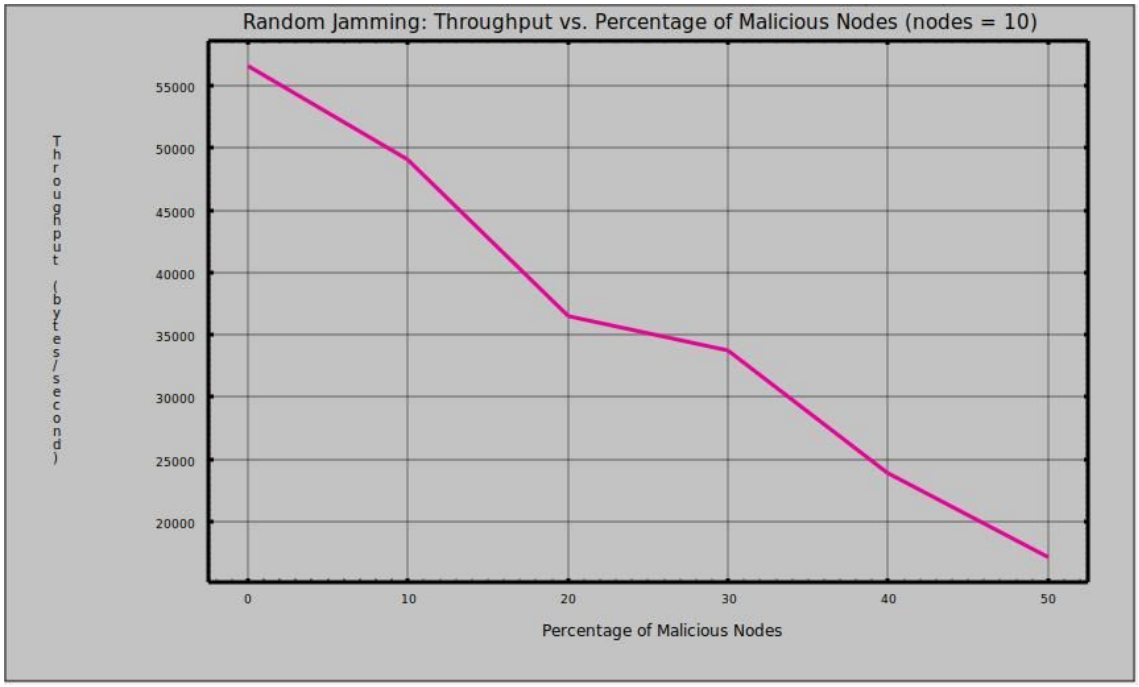

Figure: Graph showing “Random Jamming: Throughput vs. Percentage of Malicious Nodes (nodes = 10)”

- Steady decrease in throughput due to increase in the number of malicious nodes.

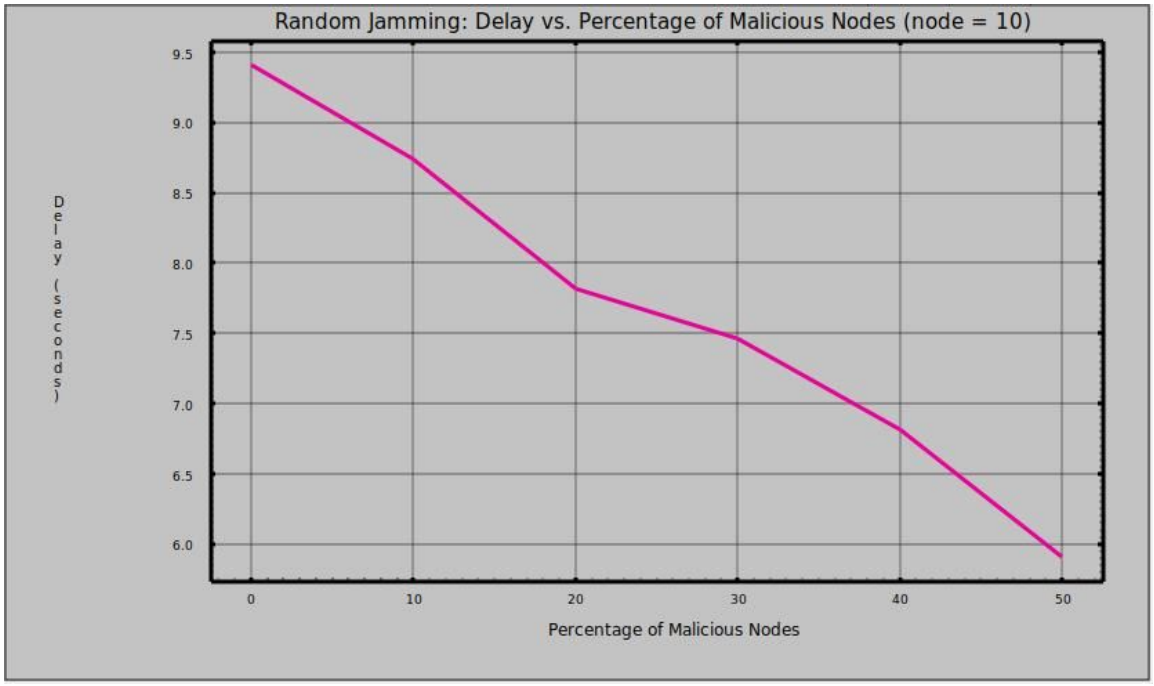

Figure: Graph showing “Random Jamming: Delay vs. Percentage of Malicious Nodes (nodes = 10)”

- Similar to periodic jamming, graph shows decrease in delay with increase in no. of malicious nodes. This indicates that an increase in percentage of periodic jammer nodes make the entire jamming process less effective due to interference.

Conclusion:

From this we can conclude that the presence of jammers has affected the system causing decrease in throughput due to increase in number of malicious nodes.
Secondly the delay even though expected to rise shows a drop due to interference.