Intrusive-Node Detection and Smart Redundancy in Networks

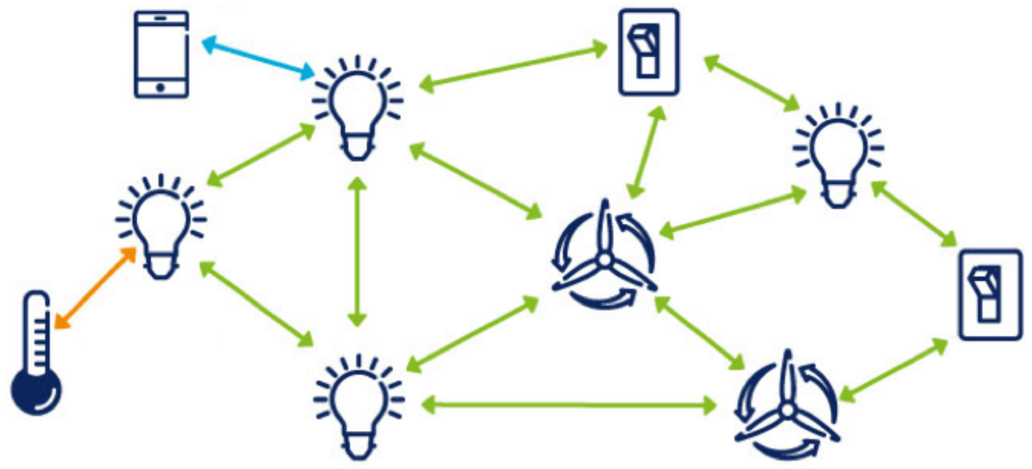
Akshay Madan

Course Instructor - **Dr. Amy Babay**

Adv. Topics in Distributed Information Systems
Spring 2020

Internet of Things

- Internet of Things is widely accepted technology with multitude of Applications
- With tech like Smart homes to Smart Industries to Smart Cities, it is pervading our life more and more
- But, simultaneously the **Security of IoT networks** is becoming a **great concern**



Security Concerns

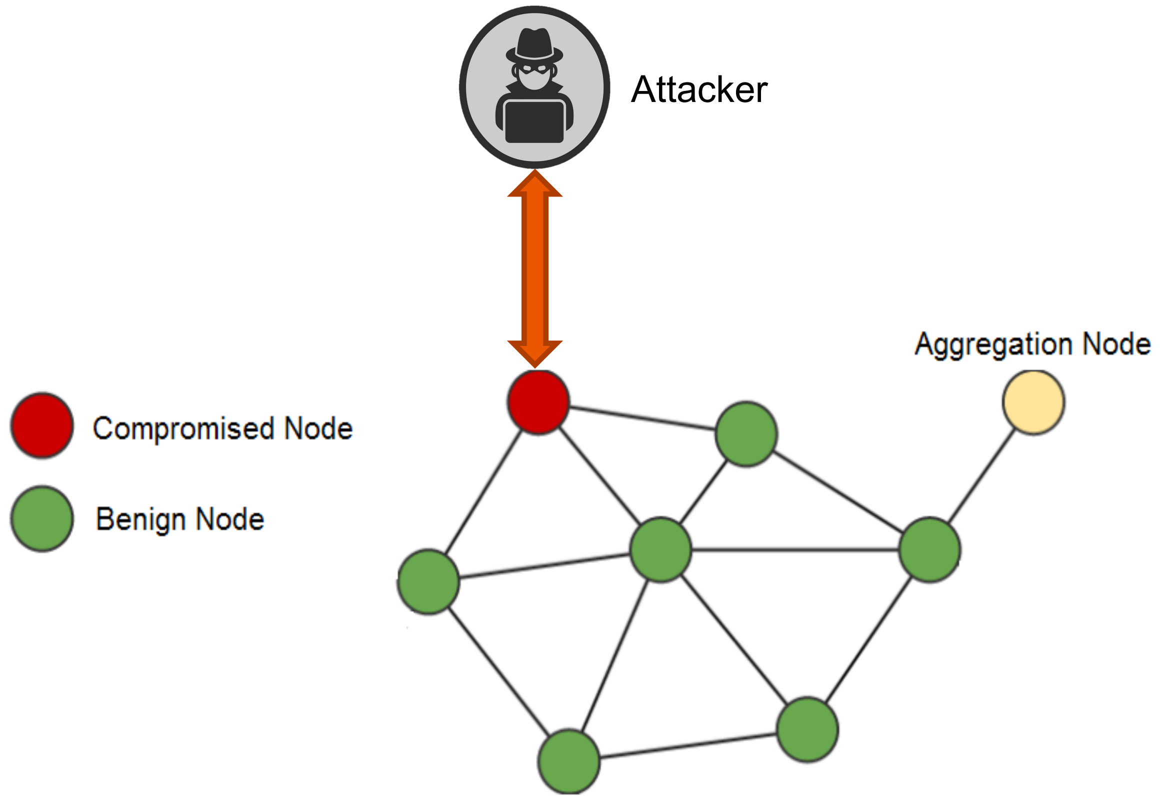
- Networks often **deployed in isolated regions** like forests, disaster sites, etc.
- IoT devices have **poor defense** due to resource constraints
- Nodes **at risk** from malicious actors who could hack into the nodes to **control or disrupt** the network
- **Serious Consequences** in case of sensitive networks like fire alarm systems, patient monitoring systems' networks.
- **Physically monitoring** all the nodes is **not feasible** due to their large number

Data Integrity Attacks

- Consider an **IoT network** with a **mesh topology** among the nodes that forward data to the **Aggregation Node**

- One or several nodes in the network may get **compromised** and may **manipulate some packets** that go through them

- Such an attack is called **Data Integrity Attack** or Packet Manipulation Attack



Problem Statement

Part 1: To create an **efficient intrusive node detection** technique by detecting the **malicious and erroneous nodes that attempt data integrity attack** in any mesh topology.

Part 2: To **reduce the need of redundancy** in **k paths disjoint** source based dissemination method while ensuring packet delivery.

Approach

Previously done work:

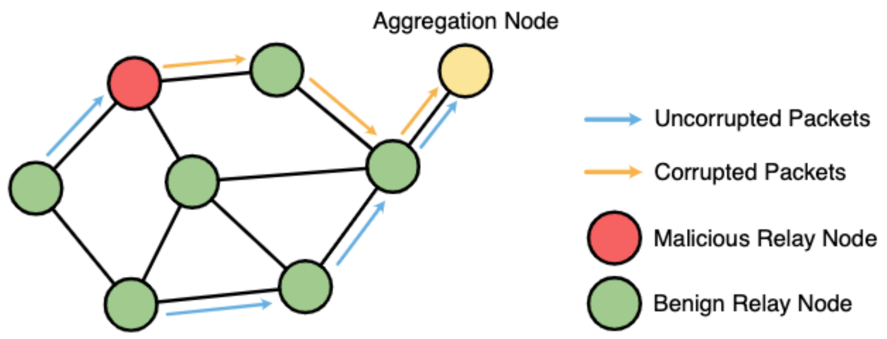
- **Special Nodes** probe the network
- Assign **trust** values to different **paths** and thus derive **node trust values.**

This Term:

- **Part 1:** Investigate the **stream of packets** through paths to derive node nature – malicious or erroneous
- **Part 2:** To use the node trust value to send less number of packets through where we know the malicious nodes reside.

Part 1

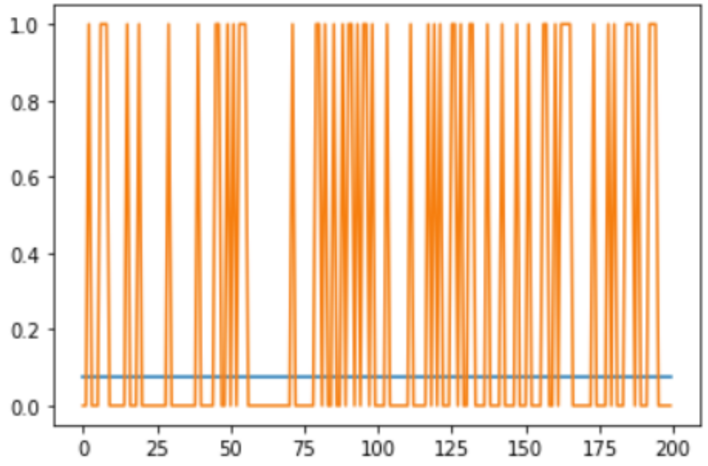
- **Investigate** the **stream of packets** through various paths to derive node nature – **malicious** or **erroneous**
- **Approach**
  - Look in packets received by the destination node through a set of paths for corrupted or not – **stream of 1s and 0s for each path**
  - Derive Node Streams from path packet streams to tell the behavior of the nodes. This is similar to the technique in the previous work with an extra dimension of packets received in the order that they are received.



Part 1

No Solutions!

- The node streams are all constant.
  - Blue – Node Stream
  - Orange – Actual Packet Stream through the node



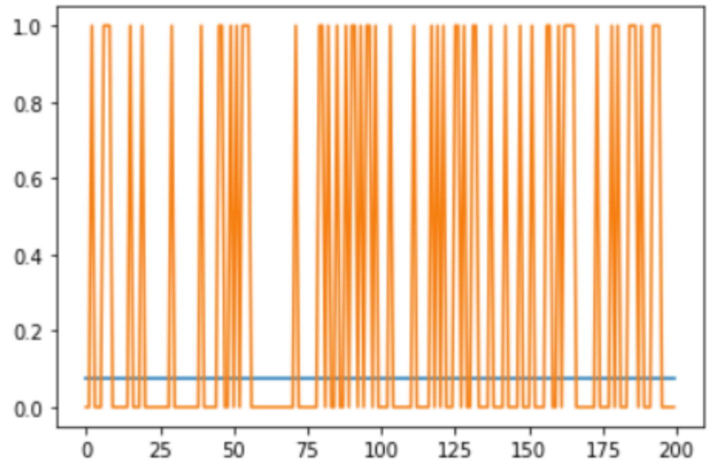
Alternate Solution

- Maybe **other nodes in that paths** are **interfering** with the node in question.
- So, try to get a path to **isolate a corrupted node** in a path – Source -> Corrupted Node -> Destination

Part 1

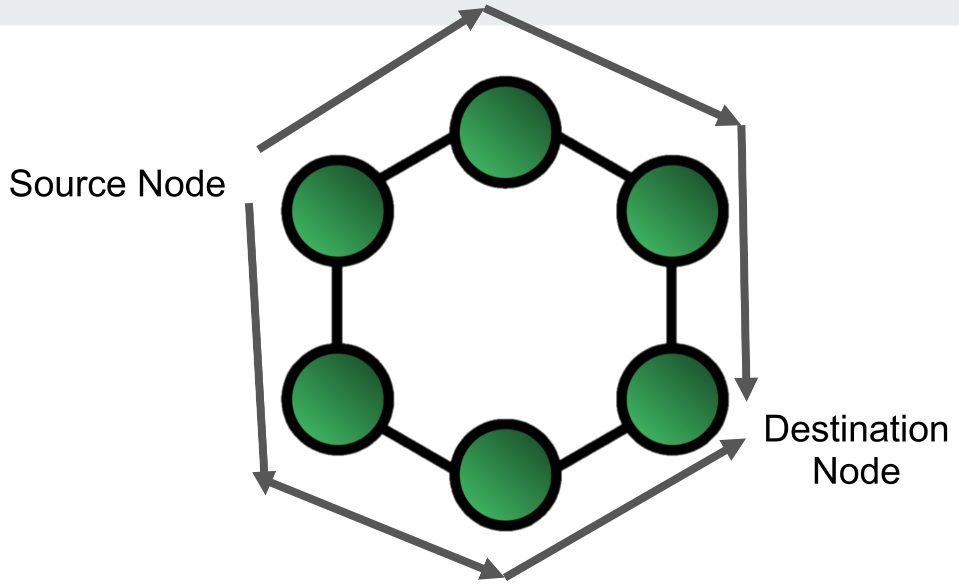
- Involves finding paths that contain one node through the network.
- Setting that node as corrupted
- Simulate!

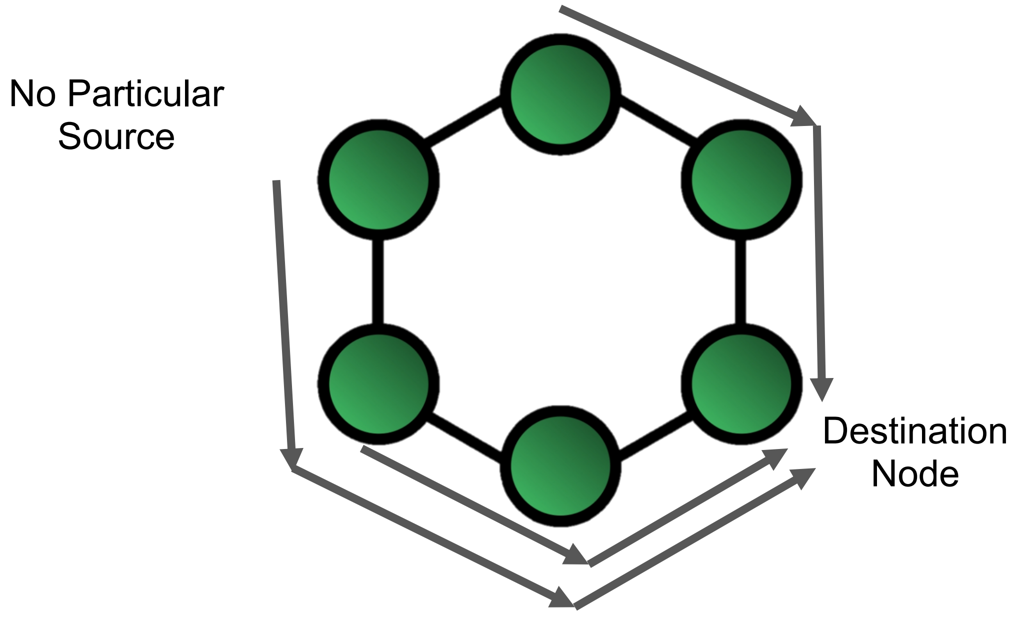
- **Still No Solution!**



Part 0

- **This is because the Part 1 did not work**
- Look into working with other topologies like ring
- **Problem** – Only 2 paths in Ring from Source to Destination
- **Solution**
  - No Source Node
  - Requires assumption that every node responsibly sends probe packets to a destination node
  - Thus can include multiple paths
- **Results**
  - Similar Accuracy as the mesh: 95.6% s.t. parameters





Part 2

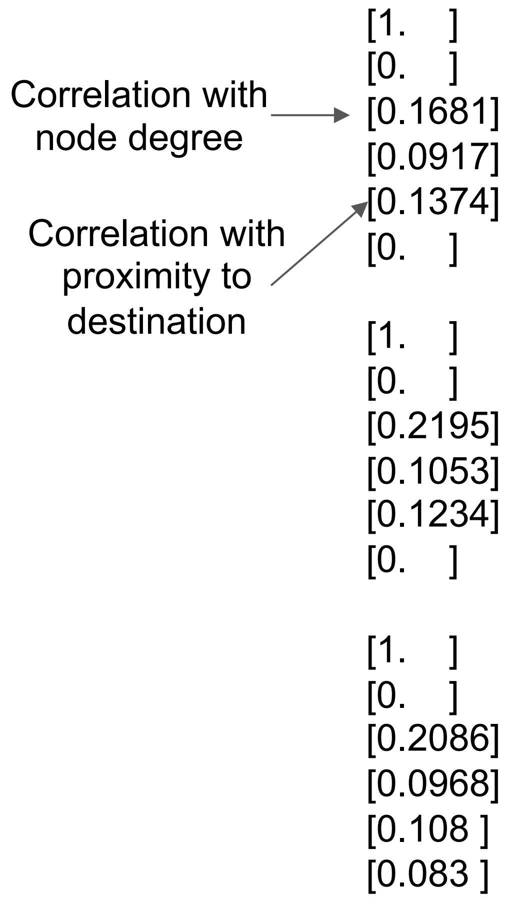
- **Idea:** To **improve** upon the **k disjoint path redundant transmission** technique mentioned in the paper: "**Practical Intrusion Tolerant Networks**" by Daniel Obenshain, Amy Babay, et al.
- k – node disjoint paths based Source Dissemination method
    - Source based redundancy technique
    - Send k packets through the network across k disjoint paths
    - Ensures packet transfer even if k-1 adversaries.

Part 2

- To use the predictions of node metrics to reduce the redundancy, yet ensure delivery
- Divide the nodes into 3 cases:
  - If it is **highly likely** that a node is **compromised** then **no point in sending out packets** through that node.
  - If it is **highly likely** that a node is **benign** then **send as few packets** as possible. because even a singular correct transmission is sufficient.
  - If we are **unsure** then send the message through the node.

Part 2

- **Confidence of prediction** for a node defined as a linear combination of
  - proximity to the destination + node degree + difference of node metric from avg
- Decided on the basis of observing each of their **correlation coefficient with the Accuracy**
- Confidence is high when greater than 0.5 (default).
- Lastly, **multiply** all **node wise packets** sent, in a path, to get packets sent on that path.


Correlation with node degree
Correlation with proximity to destination

```
[1.    ]
[0.    ]
[0.1681]
[0.0917]
[0.1374]
[0.    ]

[1.    ]
[0.    ]
[0.2195]
[0.1053]
[0.1234]
[0.    ]

[1.    ]
[0.    ]
[0.2086]
[0.0968]
[0.108 ]
[0.083 ]
```

K-disjoint pathfinding heuristic

Was unable to implement the classical k node disjoint path search algorithm.
Instead look for such paths in a **heuristic way:**

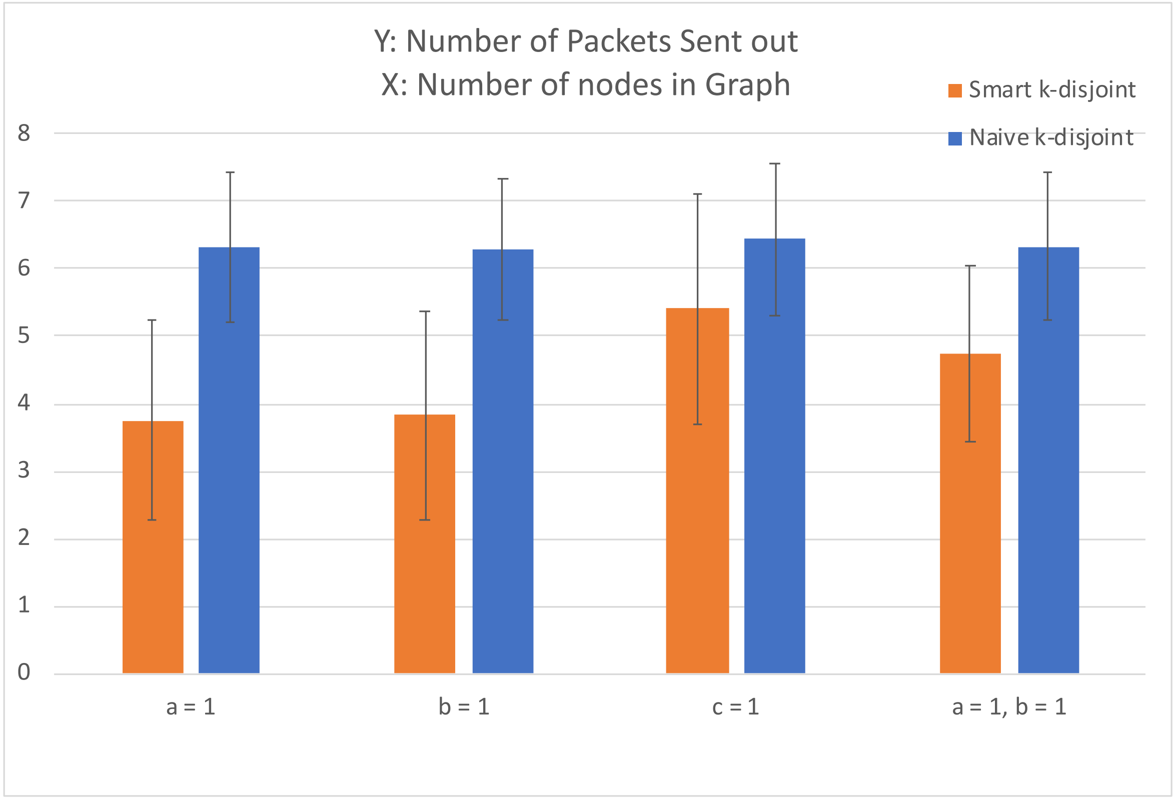
1. Assign a root path that is the shortest path from source to destination and add it to path set.
2. Find next 1000 shortest paths and 1000 randomly generated paths from the source to destination
3. Look in these 2000 paths, for a paths that is node disjoint to all the relay nodes in the path set.
4. When found, add to the path set
5. Repeat step 3 until have k paths.

Results

Varying the parameters for node metric Confidence

- a – effect of **proximity to the destination**
- b – effect of **node degree**
- c – effect of difference between **node metric from average.**

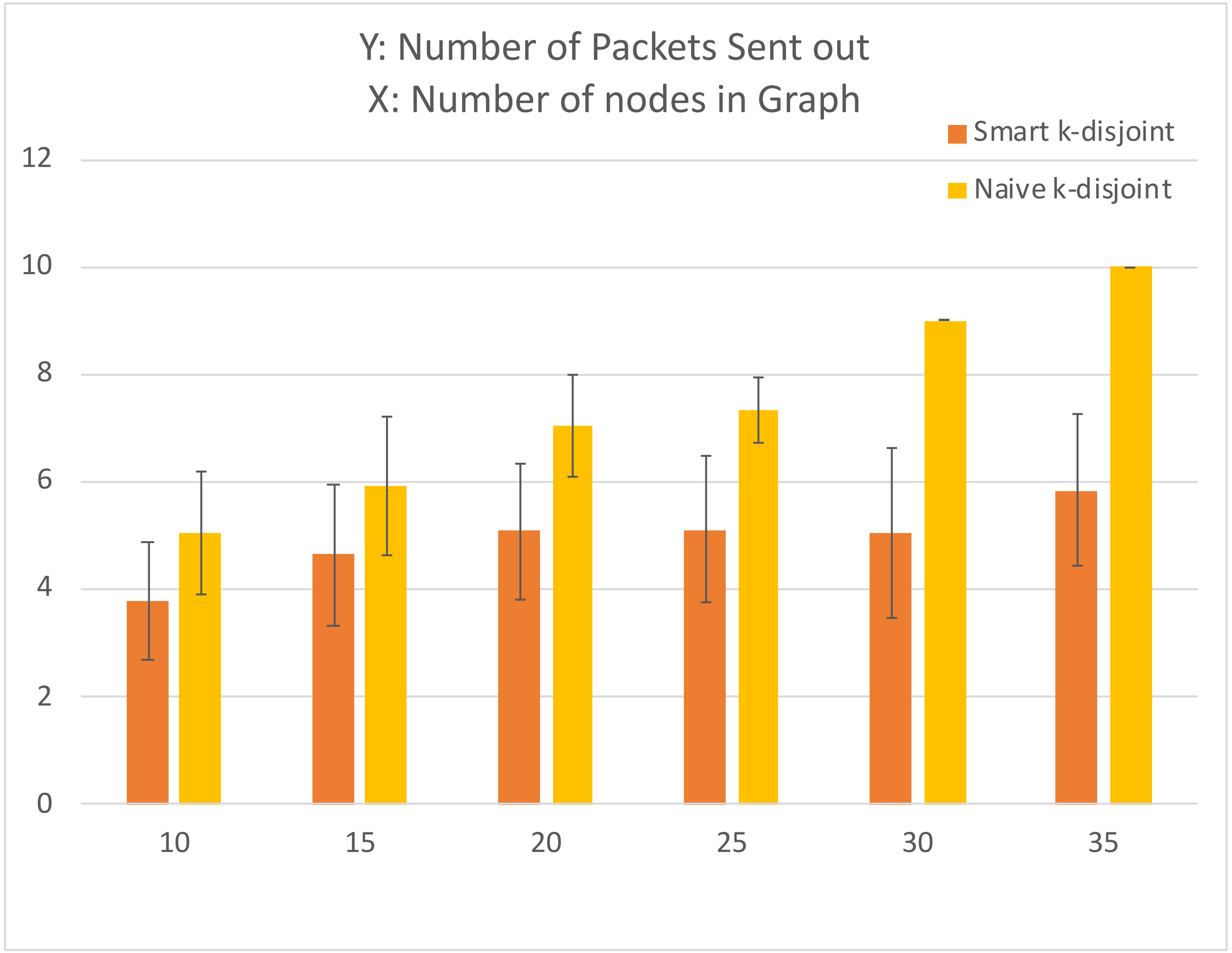
Note: For all experiments k is as many paths as could be found for a graph using the heuristic method described before, which also decides the number of malicious nodes in the graph (k-1)



Results: Varying Number of Nodes

For each bar in the bar graph:

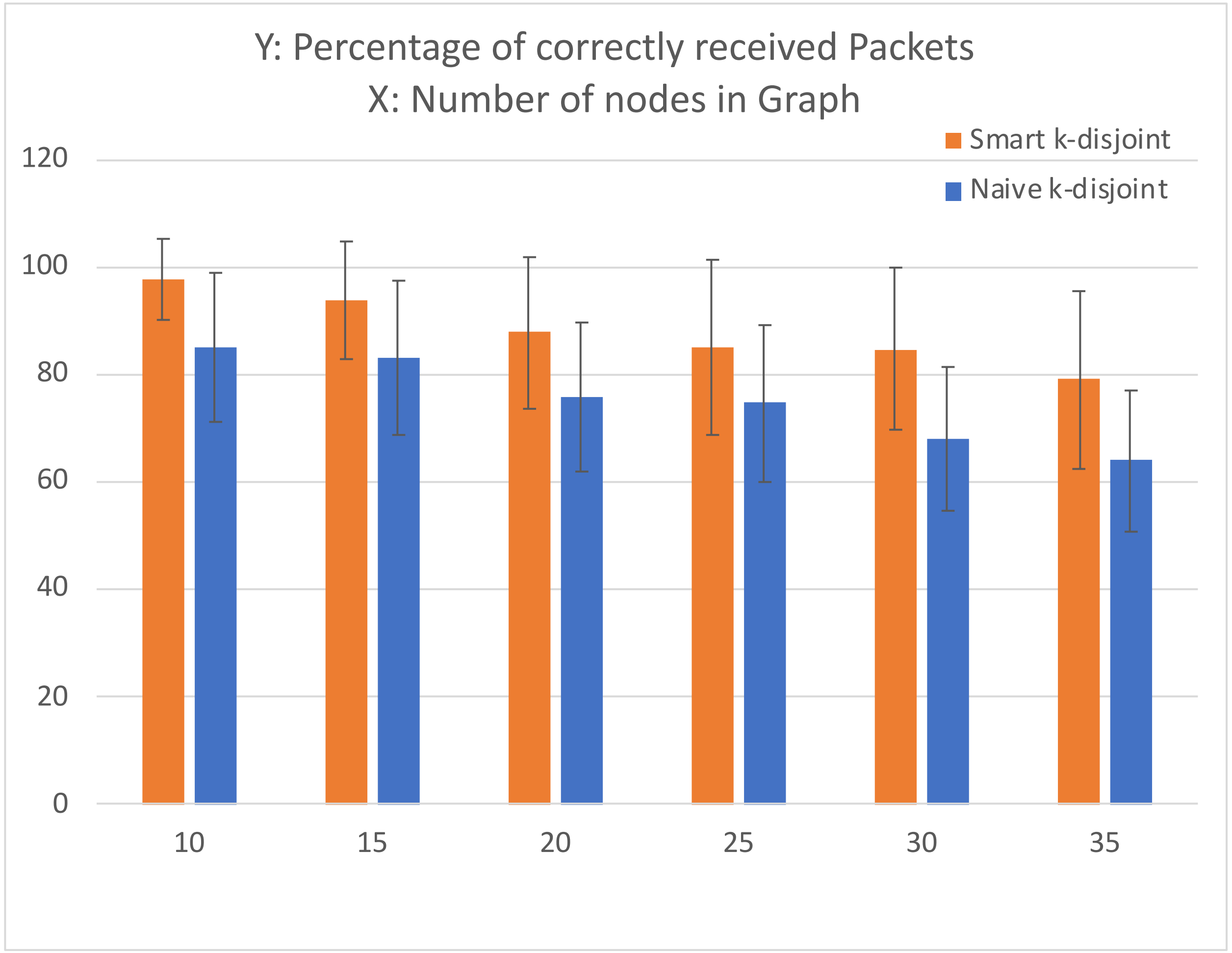
- 1000 Graphs simulated over
- 10 repetitions of the experiment.
- X axis: No of nodes in the graph
- Y axis: No of packets needed to be sent
- Error Bars: Standard Deviation of Y



Results: Varying Number of Nodes

For each bar in the bar graph:

- 1000 Graphs simulated over
- 10 repetitions of the experiment.
- X axis: No of nodes in the graph
- Y axis: Percentage of packets correctly received
- Error Bars: Standard Deviation of Y



Thank You. Any Questions?