JEDI:- A VOTE WITH NO VOTE APPROACH TO DEFEND SDN's FROM MALICIOUS ADMINISTRATORS

UCSDCSE
Computer Science and Engineering

Sriram Manohar, Aditya Suresh kumar, Rakesh Karanth, Vikas Lokesh

The "MISBEHAVING" Administrator Problem

- Administrators can affect SDN routing by mis-configuring correctly functioning controller upon link failure
- Human Error is responsible for **50 - 80%** of all network outages
- Fleet (Matsumoto et al) proposes a single configuration approach using threshold signatures and voting wherein the problem is resolved **only if at least n - k administrators (out of n) are not malicious**.

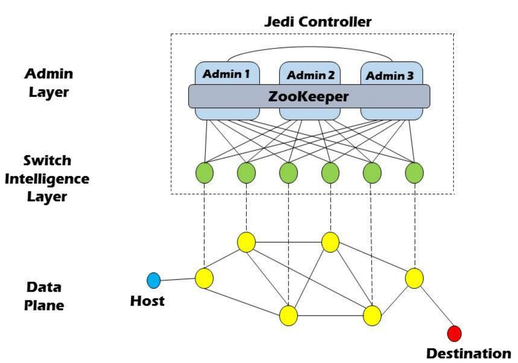
Key Contributions

- Propose Jedi - an extension to Fleet that uses a **multi-configuration approach and a Vote with No Vote** protocol.
- The best path is chosen as long as at least 1 administrator is non-malicious contrary to Fleet's k-malicious admin adversary model.
- Jedi is more resilient and allows administrators to create network configurations more independently of each other.

Jedi's Approach

- Administrators are
  - time-synchronised
  - communicating with Zookeeper via SSL / TLS. This is to implement distributed consensus and to ensure each administrator votes only once
  - having the same view of the network topology
  - sharing the same routing policy if not malicious
- Vote - a tuple (path, confidence score) is sent to switch intelligence layer for validation via Zookeeper

Jedi Model



New Switch Intelligence Layer

- Intermediary layer between the Administrators and the Controller
- Initiates the voting process upon detecting link failure
- Validates the confidence score proposed by each Admin
- Accepts vote only if the proposed vote has a confidence score greater than the current maximum score among the other Admins
- Applies the best configuration onto the network at the end of the voting process

Implementation

- Prototype implemented in Python-based POX controller and Mininet SDN framework.
- Flow rules propagated to switches through OpenFlow
- Voting simulated using Zookeeper Kazoo library
- Python scripts simulate admin behavior and switch intelligence layer
- OVS-OFCTL Switches operating in Kernel mode, Remote Controller operating in out-of-band mode

Workloads and Experimental Setup

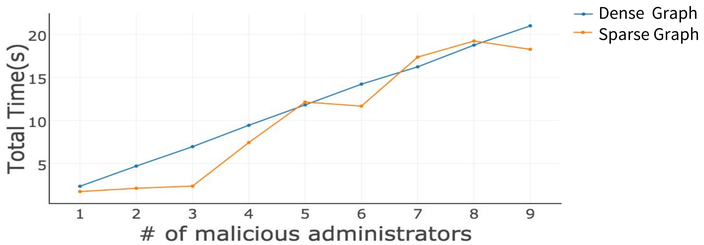
- Random graphs were generated with edge weights being either bottleneck bandwidth, latency or minimum number of hops
- Two types of graphs: Sparse and Dense internet-like graphs. Probability of link between 2 switches is 20% in sparse graphs while 80% in dense graphs. Topologies had 4-25 switches
- The number of administrators in the network ranges from 3 to 10. Three basic classes of admins: Good, Malicious and Naive.
- Zookeeper responsible for establishing consensus amongst administrators and storing the highest validated confidence score configurations.

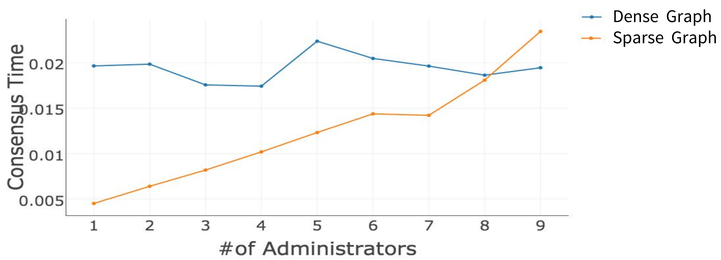
Evaluation

KEY INSIGHT:- As number of malicious admins increase the total time increases. Sparse graphs do not follow this linear relation. Other graphs explained in paper.

Metrics considered for best alternate path:
- Maximum bottleneck bandwidth
- Minimum Latency
- Minimum number of hops

KEY INSIGHT:- This is an Anomalous graph depicting the probabilistic nature of Jedi, where the number of malicious administrators (k) is chosen randomly among the total number of administrators(n).





TOTAL TIME = Link failure detection time + Admin Path recovery time + Vote Time + New Path establishment time

CONSENSUS TIME = Time taken for Zookeeper to poll the admins + Time taken to verify the confidence score

Comparison with Fleet

- Requires more changes to the existing SDN architecture than Fleet.
- The Switch Intelligence layer needs to maintain more information because it stores each administrator's configuration.
- More resilient and allows administrators to create configurations independently of each other.
- More powerful as n-1 admins can be malicious

Limitations and Future Work

- Metrics evaluated on the static Network.
- If evaluated dynamically it would become reactive than preventive. Is that optimal?
- What is the correct metric? That in itself is a research question!
- Reactive approach based on application Flow. Is it possible to make decisions based on the application which is making most of use of the network?