Self-healing Networks Using ORBIT Testbed

sdmay21-36

Matthew Johnson, Adam Tiedeman, Amber Chin, Parnumart Hanthiradej, Yuin-Choon Ng

Client/Adviser: Professor Mohamed Y Selim

Problem Statement

Current cellular networks will portion out a large amount of their budget to spend on repairing and resolving network problems and outages. The current method involves relying on human expertise to identify, diagnose, and resolve any issues with the network.

Solution

Our system can utilize an algorithm to detect a full or partial outage, diagnose the cause, and compensate for the outage. Using an algorithm to solve these issues will cause minimal user downtime during an outage while keeping the cost to maintain the network much low.

Users

With some additional configuration, or self-healing network can be scaled up to larger networks. Everyday cellular network users can benefit from the added coverage.

Functional Requirements

- Detect the loss of communication with a base station and mitigate the effects of the outage so users of the network do not experience an extended outage
- The central controller should be able to assess the network state with limited network congestion
- The self-healing process should make sure network nodes are not overloaded beyond their bandwidth

Non-Functional Requirements

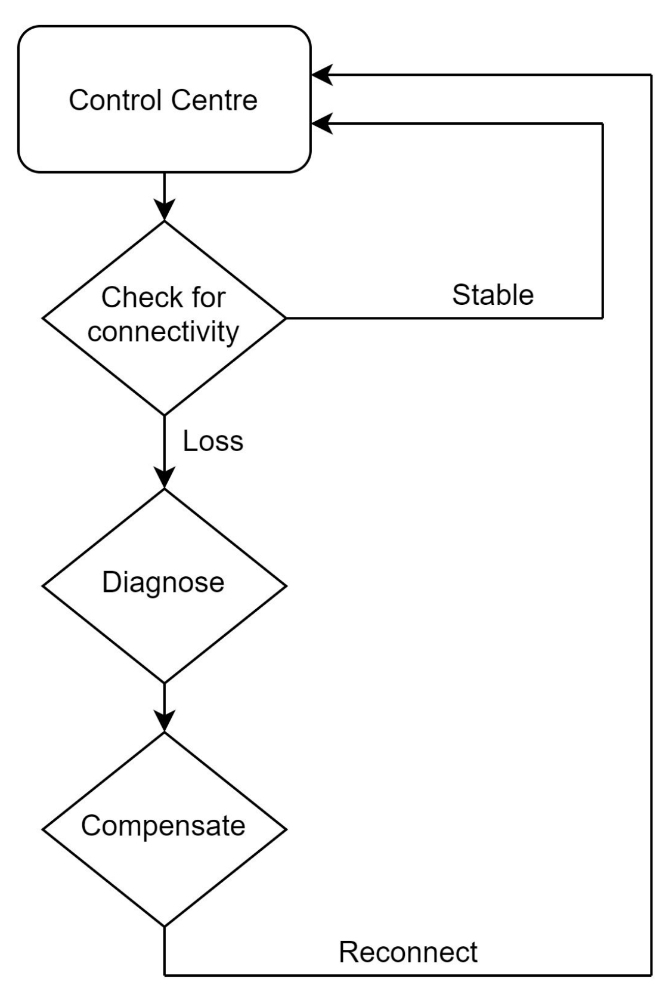
Economical

- We are using the free platform ORBIT

Environmental

- This project should be functional across open-access networks

Conceptual Design Diagram



Failure detection

- checks for network connectivity
- initiates failure diagnosis

Failure diagnosis

- Reroute traffic or attempt to maintain client connection with available servers
- Execute failure compensation

Failure compensation

- Routing error: central controller reroute based on diagnosis
- Total failure: assess clients' connection loss, command servers to reach out to client
- After failure is compensated, resume monitoring

Technical Details

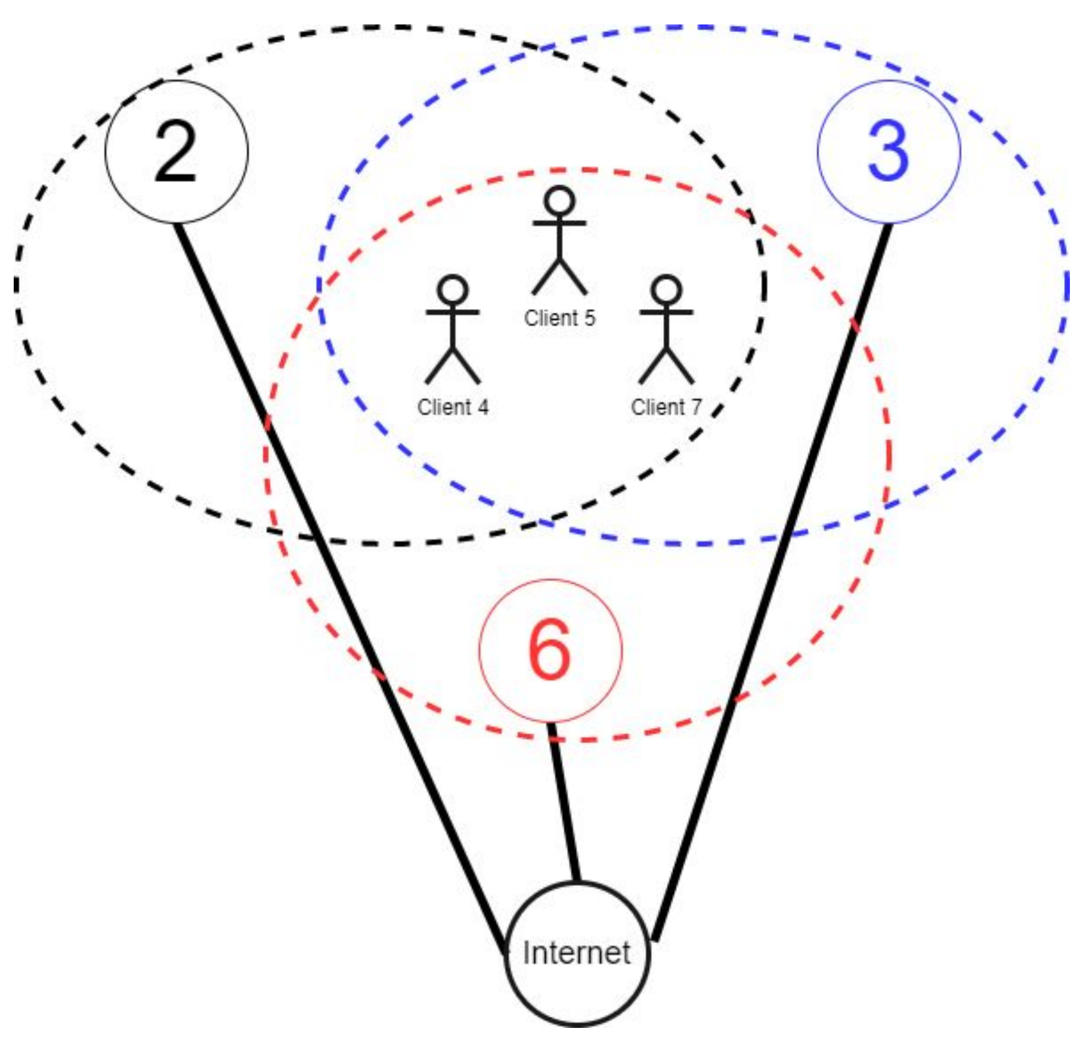
UDP

- UDP sockets are configured with executables that run on each node to do our self-healing
- The Central Controller gives specific commands about how to reroute the network to its assigned servers upon failure

ORBIT

- We use a testbed with 7 network nodes
- These network nodes are configured with our specific network topology using setup scripts
- The topology is configured using linux routing commands

Network Topology

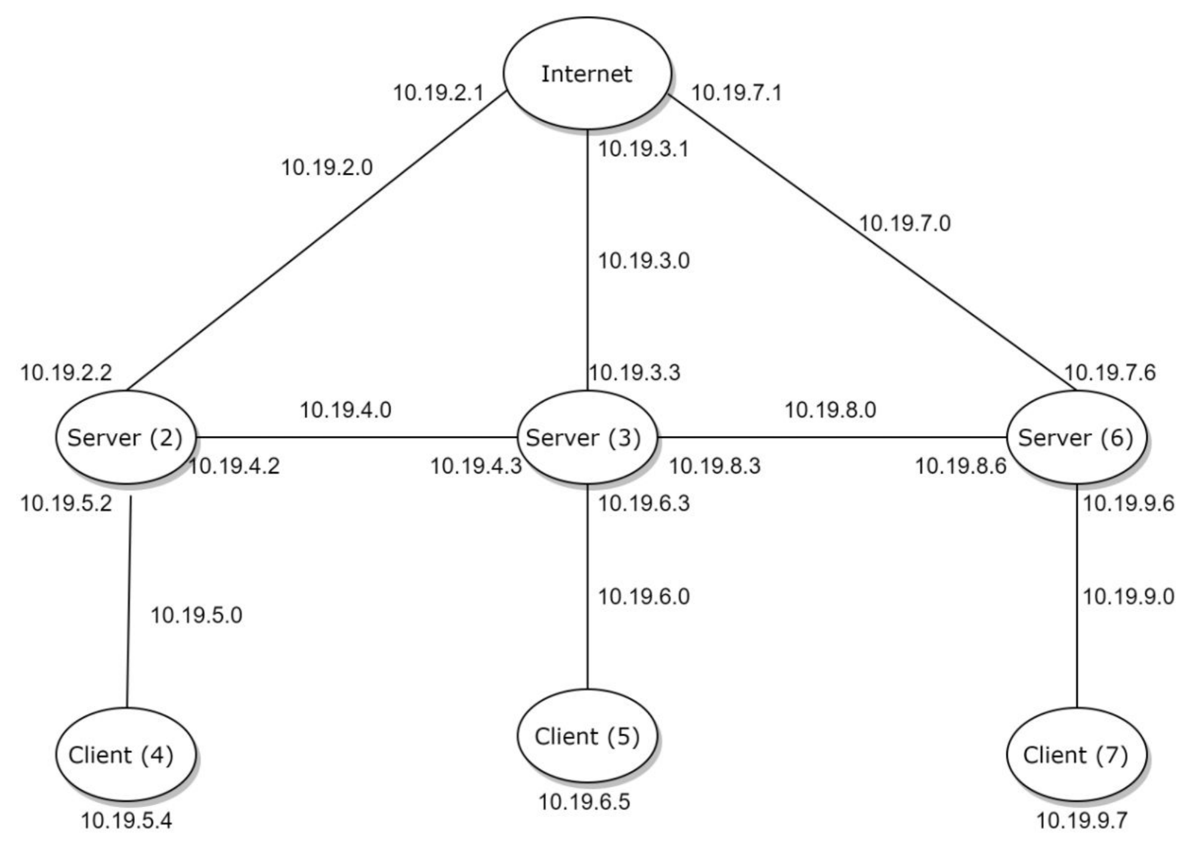


Wireless Representation

- Represents the wireless environment we are trying to replicate
- Three servers and three clients all within "wireless range"

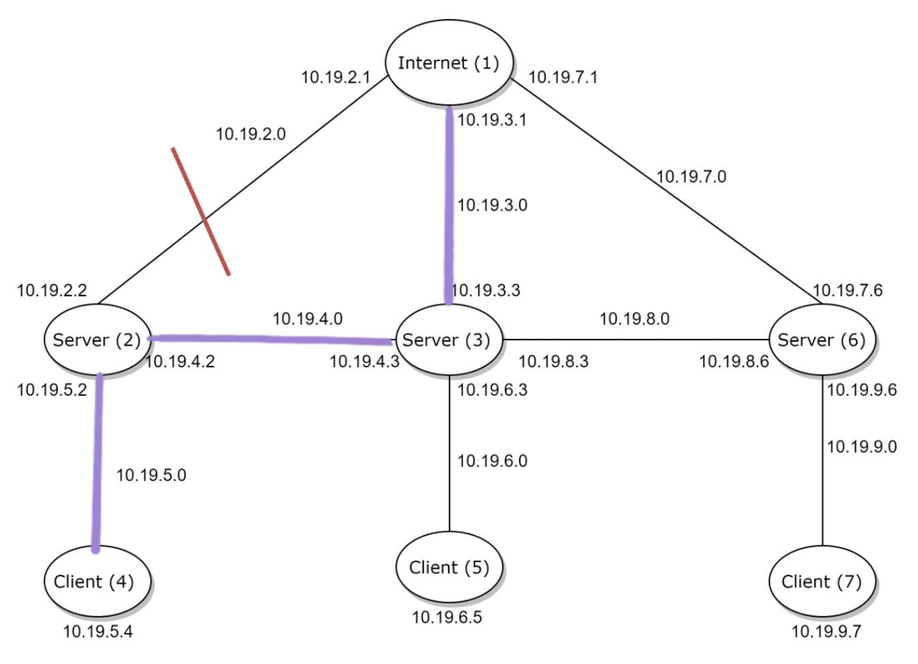
Physical Network

- Actual physical network created on ORBIT testbed
- Uses ethernet interface to create connections between nodes



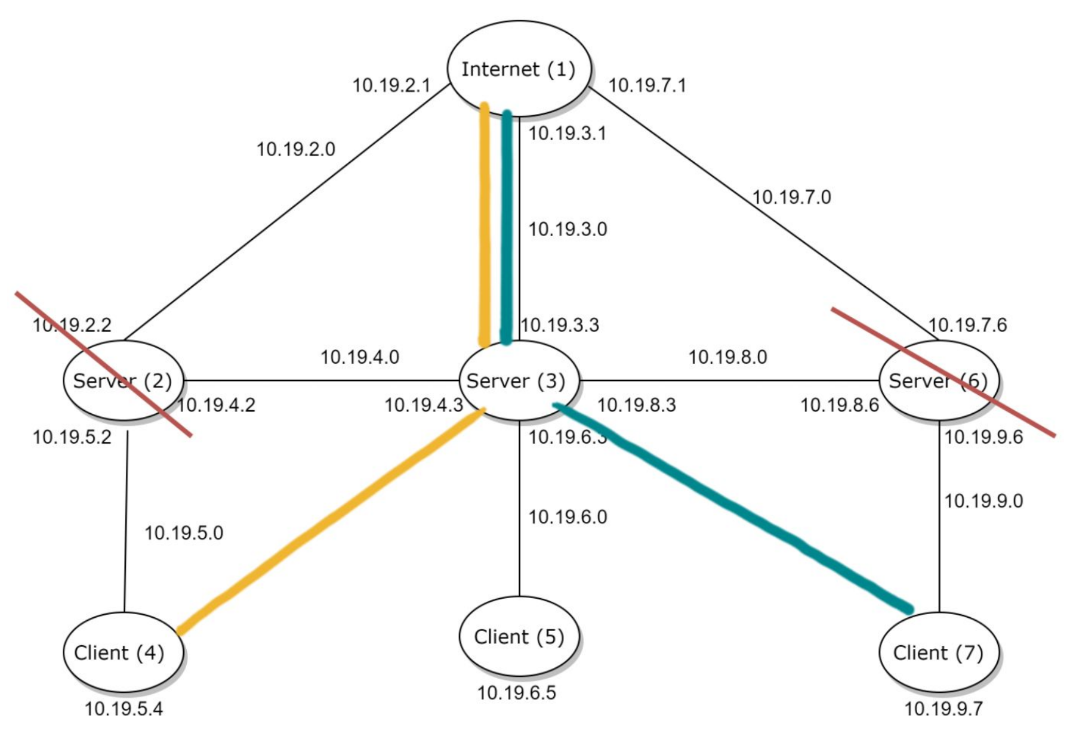
Testing

- Testing strategy



The downstream routing failure here shows a connection being broken between server 2 and the broader network (internet). In order to reconnect client 4, the central controller will evaluate available bandwidth from server 3, then connect it back to server 2 and finally to the client 4.

A second scenario is shown where multiple servers fail. This demonstrates that our algorithm is able to search out any remaining paths and distribute the remaining server bandwidth to the clients.



Engineering Standards

- IEEE 1703-2012 - Local Area Network/Wide Area Network (LAN/WAN) Node Communication Protocol
- ANSI C12.22 - The American National Standard for Protocol Specification for Interfacing to Data Communication Networks
- RFC 768 - User Datagram Protocol Standard

Engineering Constraints

- Our project must be able to run on an ORBIT testbed consisting of 7 network nodes
- Our central controller must be able to quickly assess network state without flooding the network
- Our wired connections must correspond to our topology to simulate the wireless network
- We are limited by the setup of the orbit sandbox (speed, interface, etc.)