

# Leveraging NoPASARAN to Test DNS Resolvers

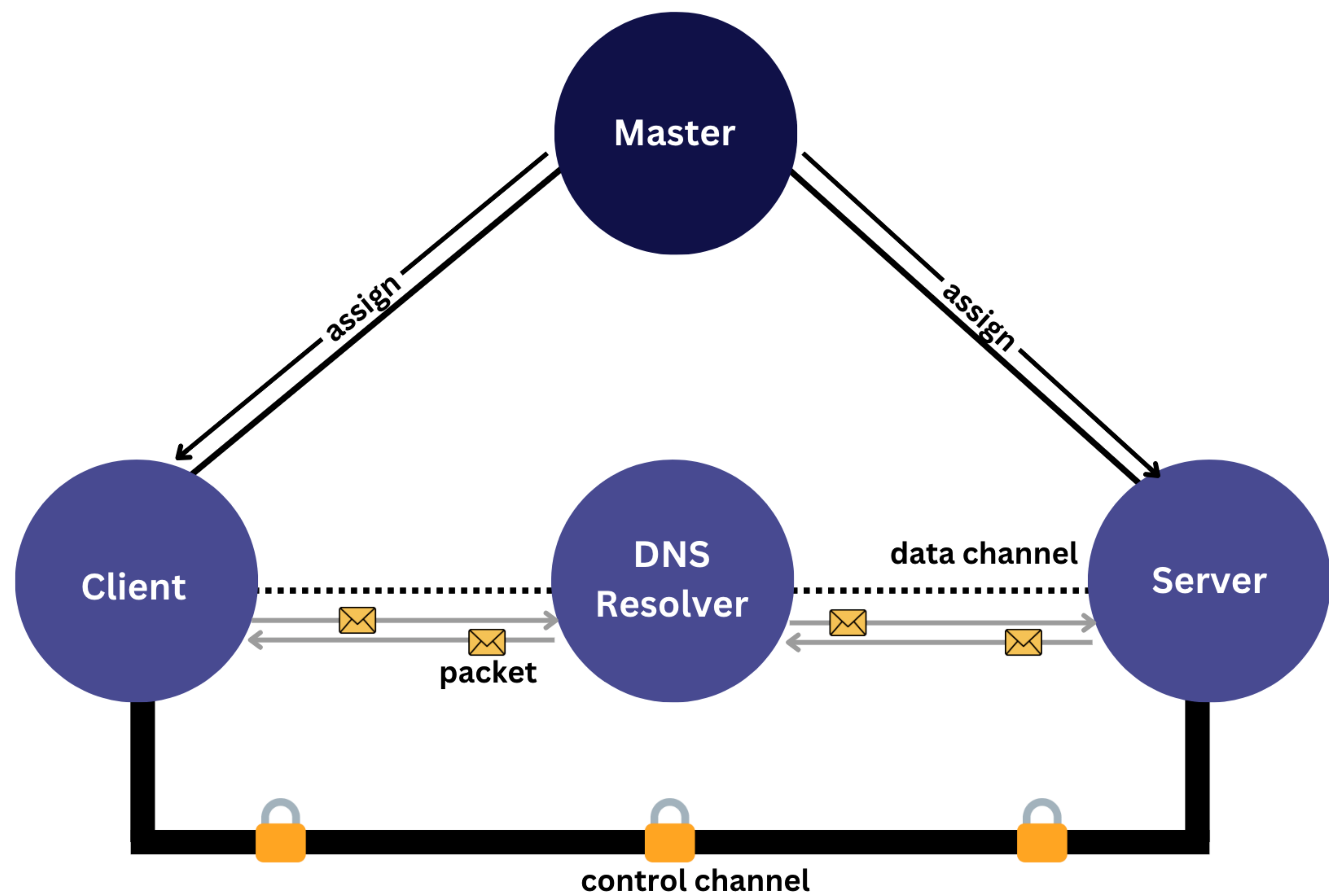
Mayan Bashehab, Ilies Benhabbour, Prof. Marc Dacier  
King Abdullah University of Science and Technology

## Objective

The goal is to conduct DNS tests using NoPASARAN [1] to identify potential implementation issues in DNS resolvers or proxying possibilities.

## Introduction

- When a client needs to resolve a domain name, it entrusts a third party, the resolver, to handle the resolution correctly for them.
- The resolver acts as a network middlebox, making the resolution process invisible to the client.
- A potential problem with DNS resolvers is that they might not support essential resolver features such as DNS port randomization, glue records policy, etc.
- We employed the capabilities of NoPASARAN to detect compliance and potential vulnerabilities of network middleboxes.



- The main features implemented in NoPASARAN are as follows:
  - Scenarios
  - Data Channel
  - Control Channel
  - Predefined Primitives

## DNS Measurement Tests

We leveraged certain tests from Netalyzr [2] into NoPASARAN, comprising the following:

- Port Randomization
- 0x20 Encoding
- Glue Records Policy
- Last Resolver IP address

## Results

Using bind9 as the DNS resolver in our setup, we obtained the following results:

- The resolver randomized source ports.
- The resolver respected 0x20 encoding.
- The resolver stripped glue records.
- The test returned the IP address of the last proxy.

The same scenarios will be used to flag resolvers that would not give these results.

## Future Work

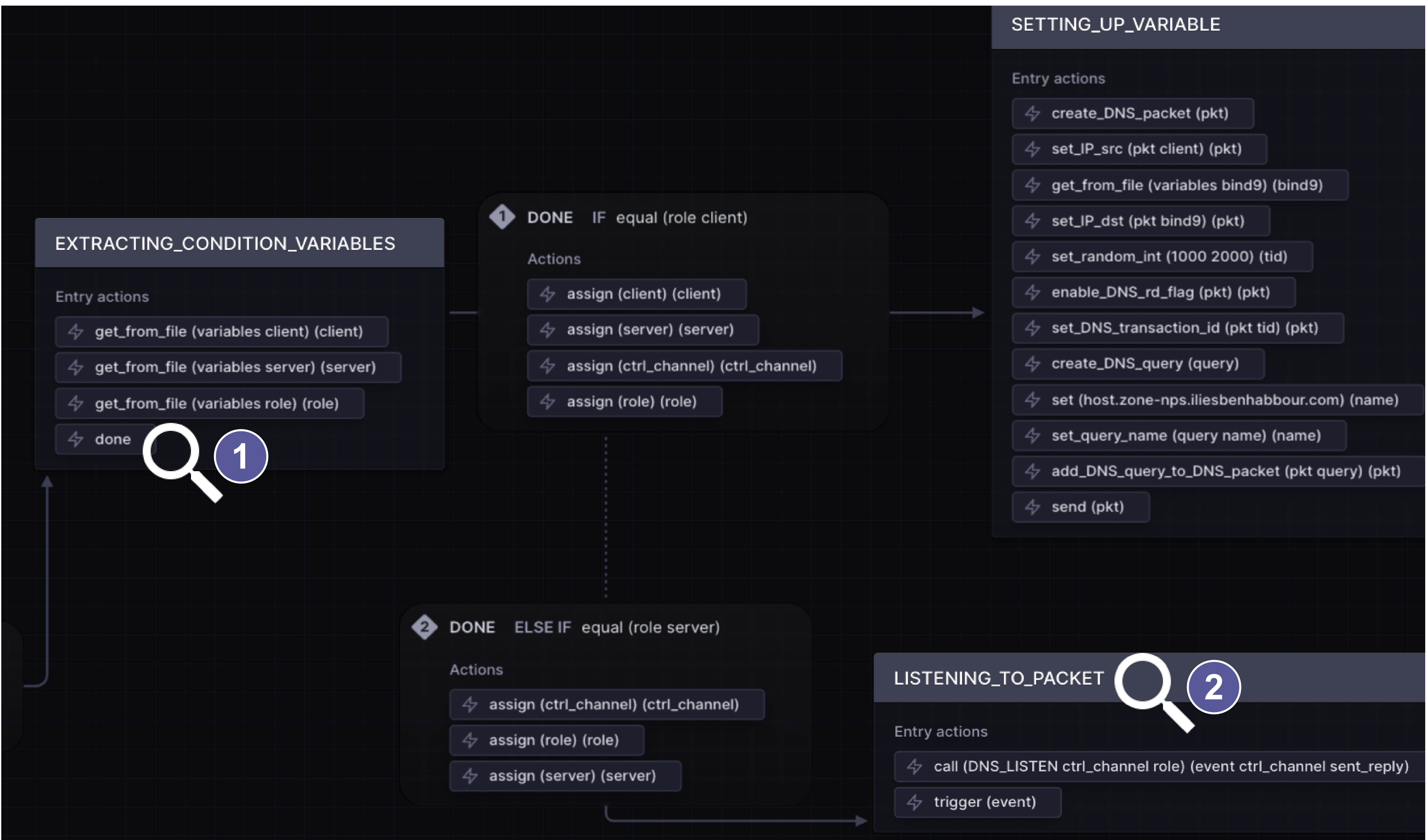
Design scenarios for the remaining of the Netalyzr DNS tests including:

- Respect for short TTL
- NXDOMAIN Wildcarding
- IPv6 Support

## References

- [1] Benhabbour, I., & Dacier, M. (2022). NoPASARAN: a Novel Platform to Analyse Semi Active elements in Routes Across the Network. Applied Cybersecurity & Internet Governance, 1(1), 73–96.
- [2] Kreibich, C., Weaver, N., Nechaev, B., & Paxson, V. (2010). Netalyzr: Illuminating the edge network. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (pp. 246-259).

## Finite State Machine



## The Client

```
[Primitive - get_from_file] Expecting 2 input(s) and 1 output(s). Optional outputs: False. Optional outputs: False
[Primitive - get_from_file] Received inputs: ['variables', 'role']
[Primitive - get_from_file] Received outputs: ['role']
chinese - MAIN-dd54b4] Setting variable role to: 192.168.123.31
chinese - MAIN-dd54b4] Executing action: {'EXECUTE_ACTION': 'done'}
[Primitive - done] Expecting 0 input(s) and 0 output(s). Optional outputs: False
[Primitive - done] Received inputs: []. Received outputs: []
chinese - MAIN-dd54b4] Event DONE triggered
[Primitive - equal] Expecting 2 input(s) and 0 output(s). Optional outputs: False
```

## The Server

```
State Machine - MAIN-516ab3] Variables assigned: {'ctrl_channel': <ProtocolController.protocol.WorkerServerProtocol object at 0x7f8a7908b7f0>, 'role': 'server', 'server': '192.168.123.11'}
State Machine - MAIN-516ab3] Executing action: {'SET_STATE': 'LISTENING TO PACKET'}
State Machine - MAIN-516ab3] Setting state to: LISTENING TO PACKET
State Machine - MAIN-516ab3] Executing action: {'EXECUTE_ACTION': 'done'}
[Primitive - call] Expecting 1 input(s) and 0 output(s). Optional outputs: True
[Primitive - call] Received inputs: ['DNS_LISTEN', 'ctrl_channel']
```