Domain Name System (DNS) Tunneling Detection using Structured Occurrence Nets (SONs)

Talal Alharbi
and
Maciej Koutny

Newcastle University

Overview

- Introduction
- Motivation
- Results testing and evaluation
- Concluding remarks
- Future work

Background

A Complex Evolving System (CES)

is our term for a “system-of-systems” that consists of a large number of sub-systems which may proceed concurrently and interact with each other or with the external environment while its behaviour is subject to modification by other systems.

Structured Occurrence nets

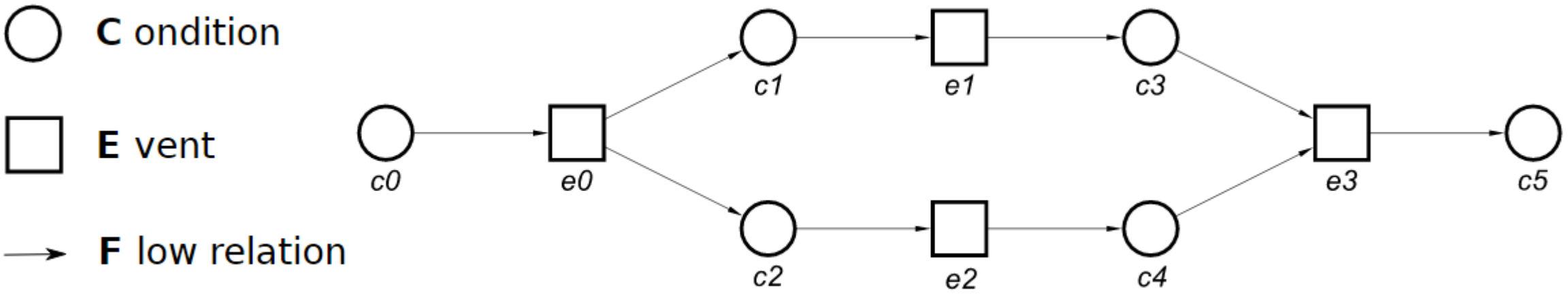
(SONs) are a formalism for representing the activity of a CES, initially introduced for the purpose of failure analysis of complex computer hardware and software systems.

Structured Occurrence Nets (SONs)

- Formalism for describing the behaviour of a CES: communication, evolution.
- They combine multiple occurrence nets into a single structure by using a variety of formal relationships.
- They extend the functionality of occurrence nets, by providing means of representing system evolution.

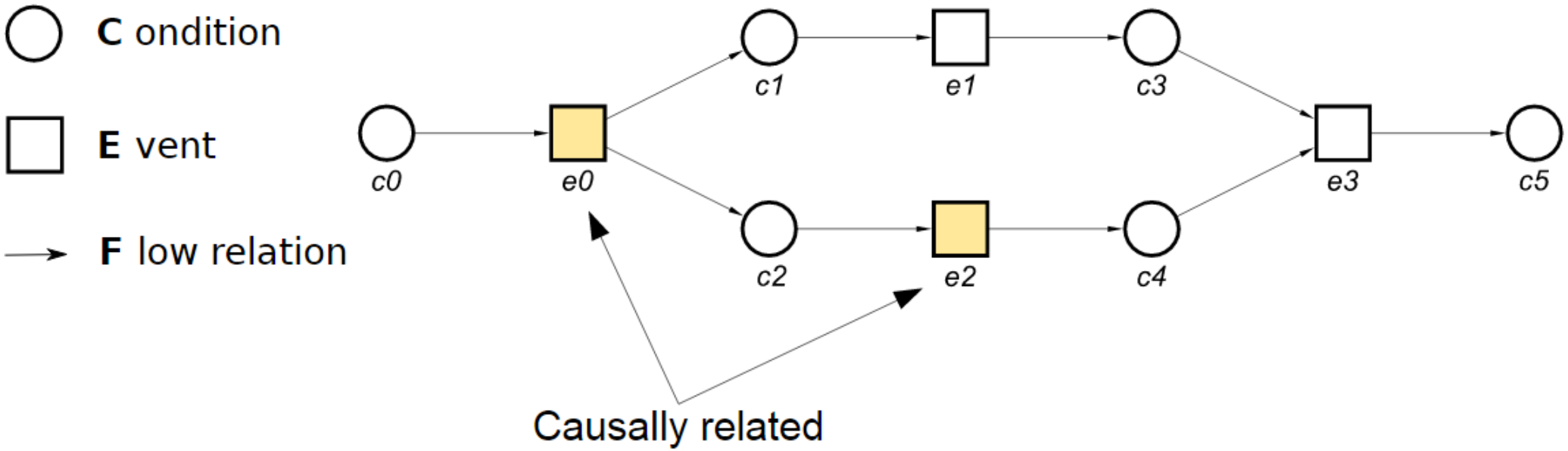
Occurrence Nets

Represent causality and concurrency information about a single run of a (distributed) system.



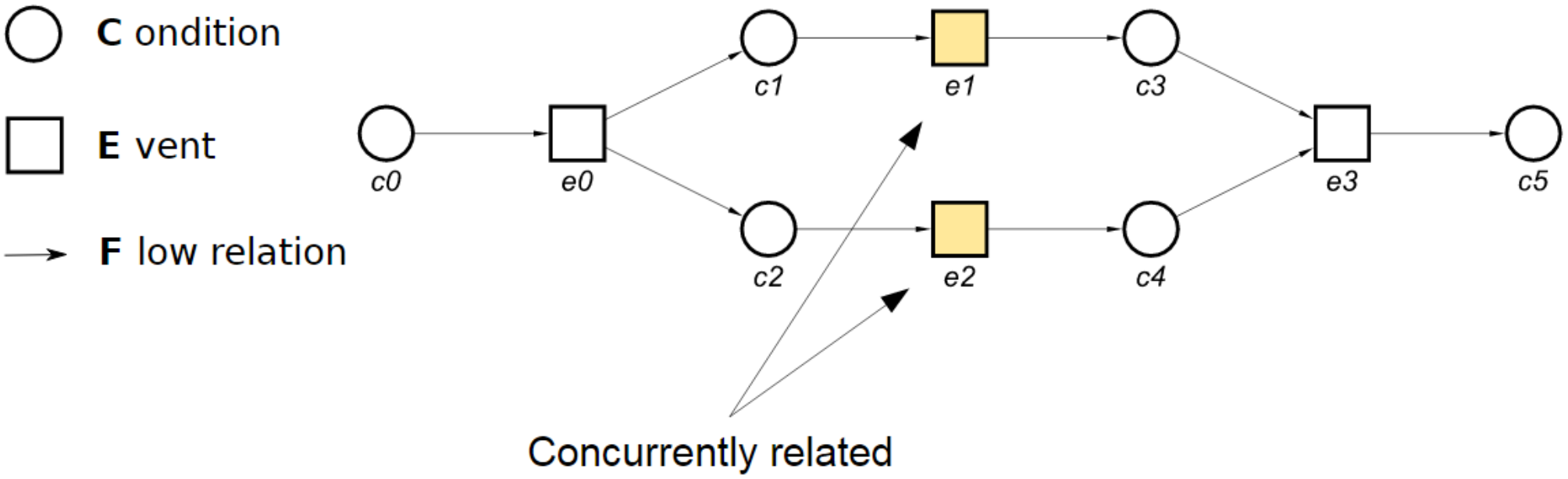
Occurrence Nets

Represent causality and concurrency information about a single run of a (distributed) system



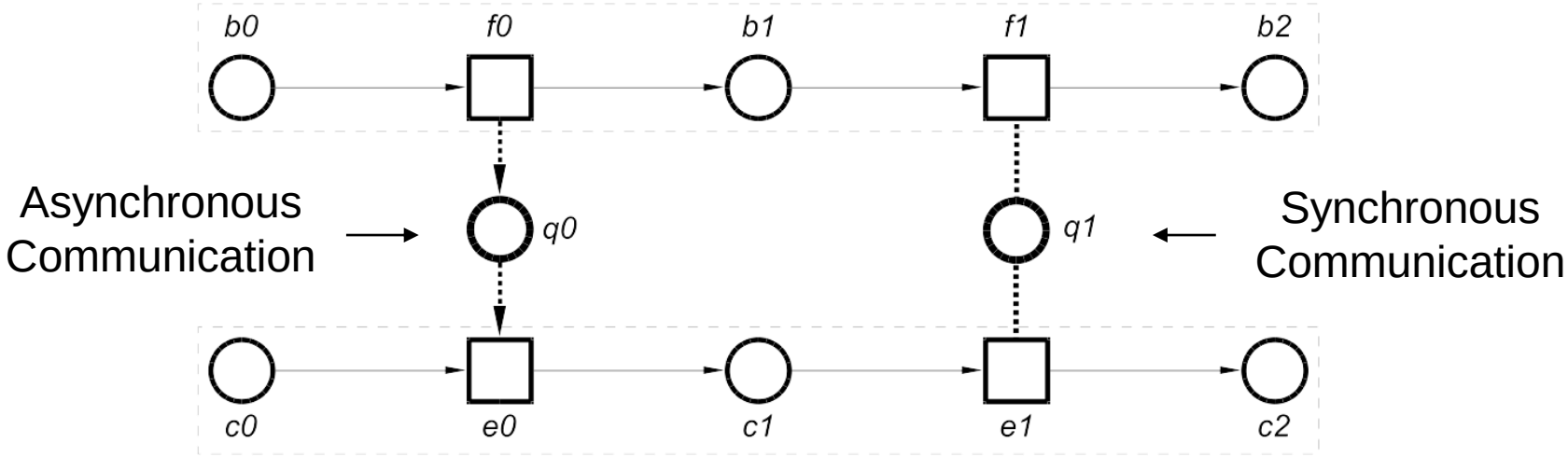
Occurrence Nets

Represent causality and concurrency information about a single run of a (distributed) system



Communication Structured Occurrence Nets (CSONs)

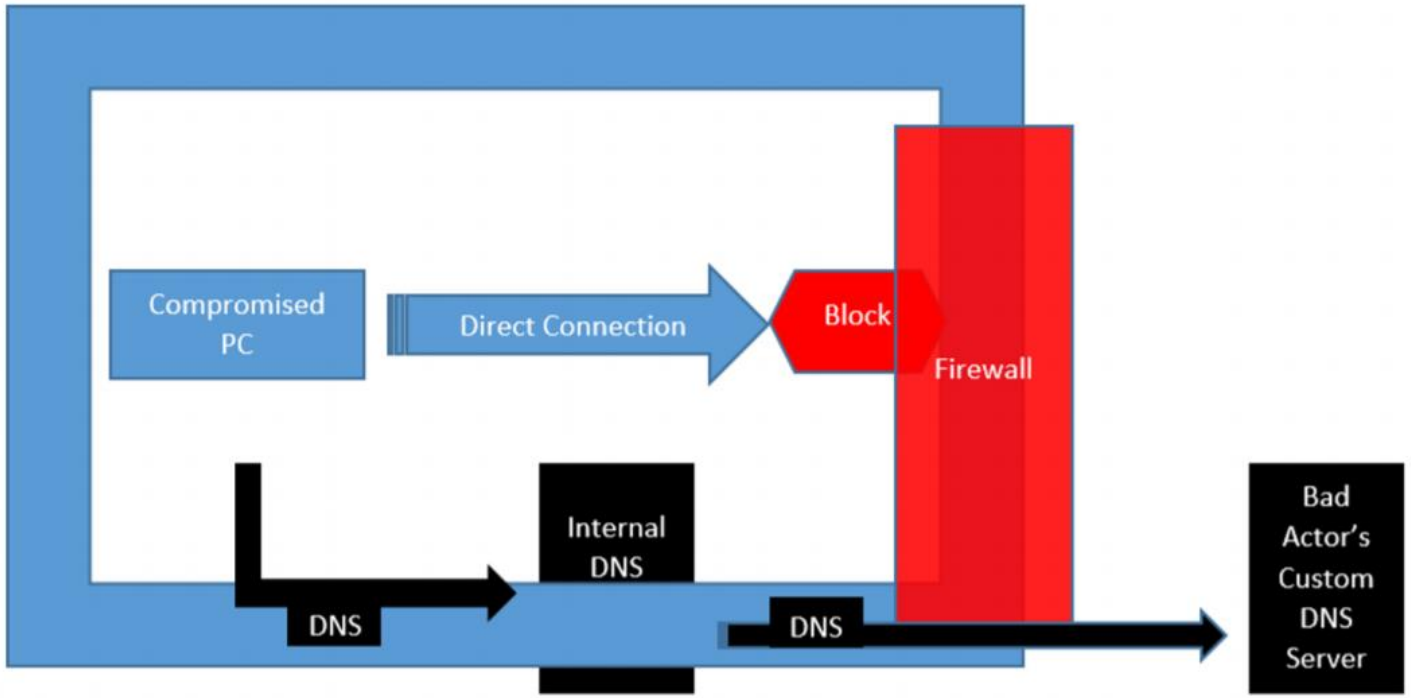
- Model asynchronous and synchronous communication between interacting systems.
- Combine multiple occurrence nets using channel places.



DNS Tunnelling

- DNS tunnelling is a covert communication channel, which allows encapsulating the traffic of other protocols (E.g.: HTTP, Telnet, FTP, SSH) within DNS packets
- DNS tunnelling is very suitable to be used for malicious activities such as data exfiltration as well as command and control call-backs from within restricted internal networks

DNS Tunnelling



Preprocessing Dataset

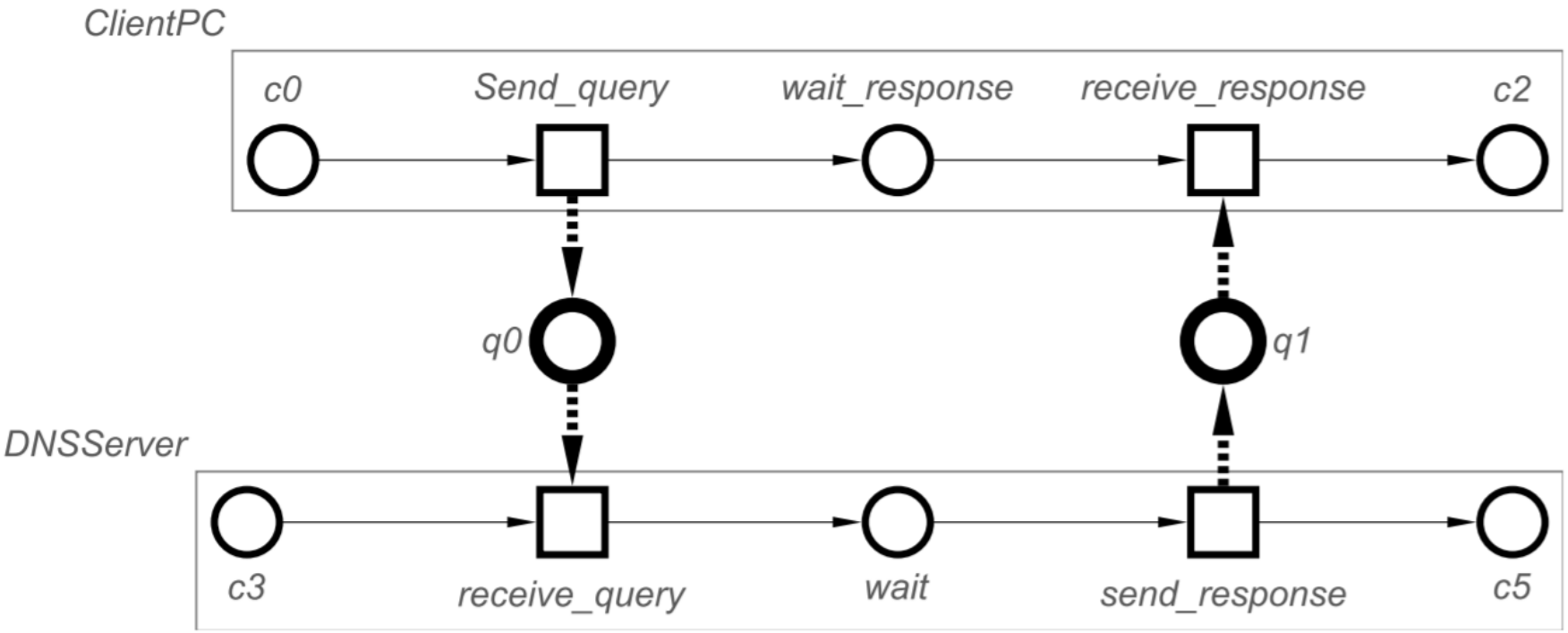
DNS Packets

No.	Source	Destination	Info
26	172.20.10.2	172.20.10.1	query 0x81fe A google.com
29	172.20.10.1	172.20.10.2	response 0x81fe A google.com

Preprocessing Dataset

No.	Source	Destination	Operation	PacketID	Domain
1	172.20.10.2	172.20.10.1	query	0x81fe	google.com
2	172.20.10.1	172.20.10.2	response	0x81fe	google.com

Preprocessing Dataset



Preprocessing Dataset

No.	Source	Destination	Operation	PacketID	Domain
1	172.20.10.2	146.185.138.197	query	0xc7cb	paaac5ay.tunnel.carn.us.to
2	146.185.138.197	172.20.10.2	response	0xc7cb	NULL paaac5ay.tunnel.carn.us.to

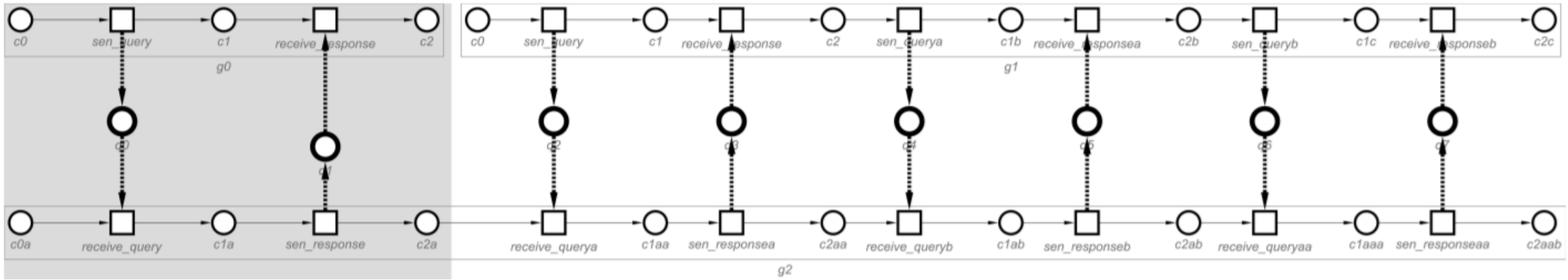
Preprocessing Dataset

No.	Source	Destination	Operation	PacketID	GroupID	Domain
1	172.20.10.2	172.20.10.1	query	0x81fe	001	google.com
2	172.20.10.1	172.20.10.2	response	0x81fe	001	google.com A 216.58.206.206
3	172.20.10.2	172.20.10.1	query	ox98c1	001	maps.google.com
4	172.20.10.1	172.20.10.2	response	ox98c1	001	maps.google.com A
5	172.20.10.2	146.185.138.197	query	0xc7cb	002	paaac5ay.tunnel.carn.us.to
6	146.185.138.197	172.20.10.2	response	0xc7cb	002	NULL paaac5ay.tunnel.carn.us.to
7	172.20.10.2	146.185.138.197	query	0x0320	002	paaydani.tunnel.carn.us.to
8	146.185.138.197	172.20.10.2	response	0x0320	002	paaydani.tunnel.carn.us.to","190392

Detection DNS Tunneling using (SONs)

- The main idea behind the algorithm is first to detect any ON input, whether it is a normal input or an ON representing a DNS attack
- We assume that each packet is a unique ON.
- And the local DNS server is one ON
- Examine each ON input and count its events which communicate with the DNS Server ON. If the number of those ON events is less than threshold value, we flag it as a normal ON. Otherwise, we flag it as an abnormal ON.

Detection DNS Tunneling using (SONs)



Detection DNS Tunneling using (SONs)

- Then, we check whether or not we had abnormal ONs; if so, then we know a DNS attack has occurred
- We have used threshold via **logistic regression**

Result Testing and Evaluation

The sensitivity and specificity methods:

1. True positive result: attack transactions were correctly identified as attacks.
2. False positive result: normal transactions were incorrectly identified attacks.
3. True negative result: normal transactions were identified as normal transactions
4. False negative result: attack transactions were incorrectly identified as normal transactions.

Result Testing and Evaluation

Table results

No.	% of normal and attack packets	True Positives	False Positives	True Negatives	False negatives
1	0% attack, 100 normal	N/A	9.7%	90.92%	N/A
2	1% attack, 99 normal	0%	9.77%	90.92%	0%
3	5% attack, 95 normal	0%	8.9%	91.1%	0%
4	10% attack, 90 normal	7.1%	9.4%	90.6%	2.9%
5	20% attack, 80 normal	82.9%	5.3%	94.7%	17.1%
6	30% attack, 70 normal	86.7%	5.3%	94.7%	13.3%
7	40% attack, 60 normal	88.5%	0%	100%	11.5%
8	50% attack, 50 normal	90.9%	0%	100%	9.1%
9	60% attack, 40 normal	91.5%	0%	100%	8.5%
10	70% attack, 30 normal	91.1%	0%	100%	8.9%
11	80% attack, 20 normal	91.5%	0%	100%	8.5%
12	90% attack, 10 normal	91.8%	0%	100%	8.2%
13	95% attack, 5 normal	92%	0%	100%	8%
14	99% attack, 1 normal	92.3%	0%	100%	7.7%
15	100% attack, 0 normal	92.5%	N/A	N/A	7.5%

Concluding Remarks

- The DNS tunneling has been examined.
- A solution to address this attack has been proposed.
- The design and development of the corresponding algorithm have been discussed.
- Test and evaluation have been shown.

Future Work

- Examine other different tunnelling tools' behaviours.
- Investigate more than one local DNS servers
- Evaluate the algorithm with large data sets

Questions

Thank you

Questions