Eclipse Attacks on Bitcoin's Peer-to-Peer Network

Ethan Heilman, Alison Kendler
Aviv Zohar, Sharon Goldberg

Presented by Joonhyuk Lee
(*slides adapted from Heilman*)

KAIST

CONTENTS

KAIST

1. Introduction

- Bitcoin is thought to be secure if 51% of the mining power is honest.
- Assuming that all miners see all Blocks/transactions: Perfect Information
- Bitcoin relies on its P2P network to deliver this information
- Controlling the network ➔ Controlling the blockchain

Can attacker manipulate node's view on the Bitcoin Network?

1. Introduction - Outline

- **Eclipse Attacks & Implications** — chapter 2
- **How to eclipse a Bitcoin node** — chapter 3
- **How many IPs does the attacker need?** — chapter 4
- **Countermeasures** — chapter 5
- **Eclipse Attack on Ethereum** — chapter 6

Chapter 2

: Eclipse Attacks & Implications

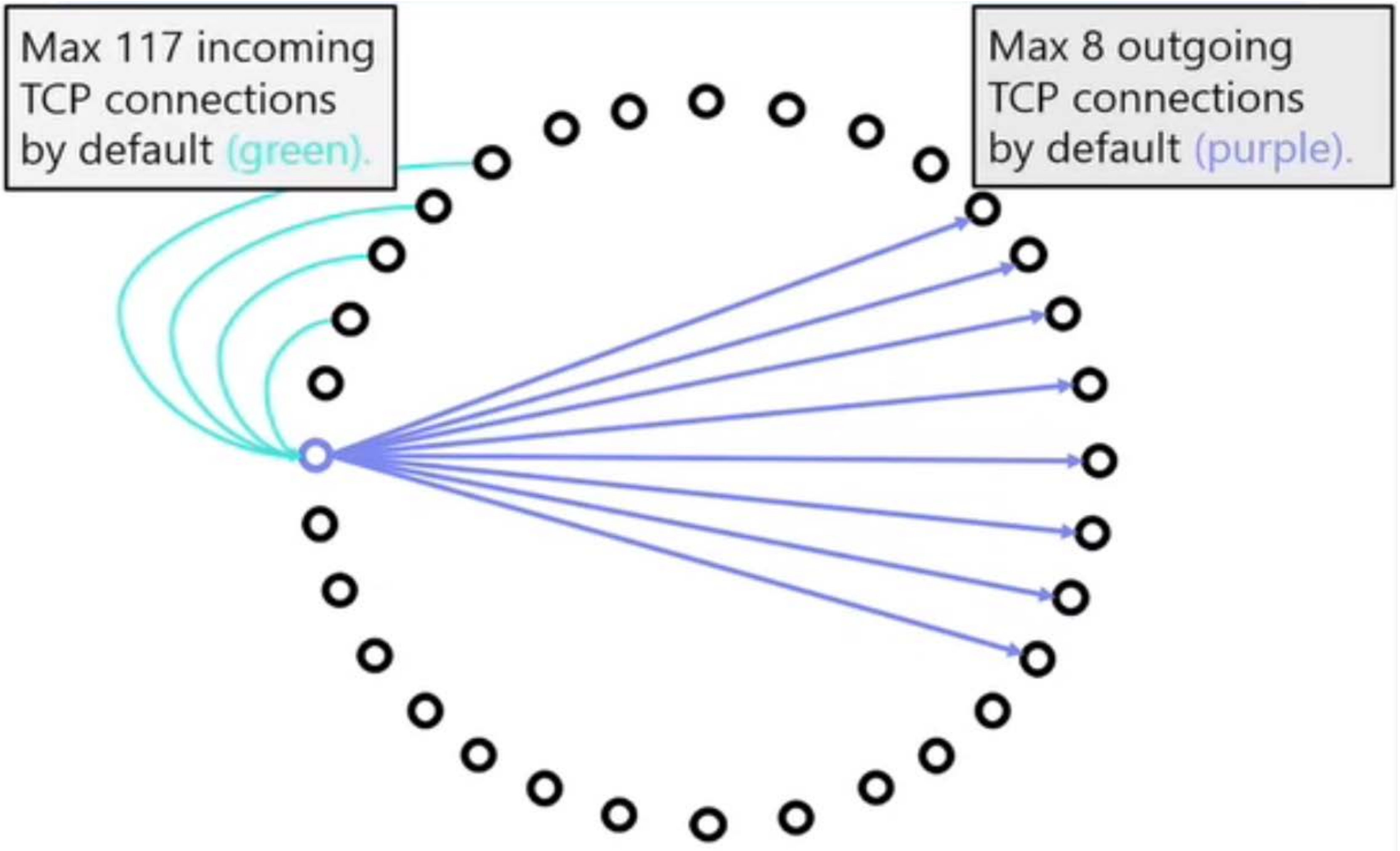
Outline

- **Eclipse Attacks & Implications**
  - Explanation about eclipse attack
  - 51% attack, Selfish Mining
  - N-confirmation double spending
- How to eclipse a Bitcoin node
- How many IPs does the attacker need?
- Countermeasures
- Eclipse Attack on Ethereum

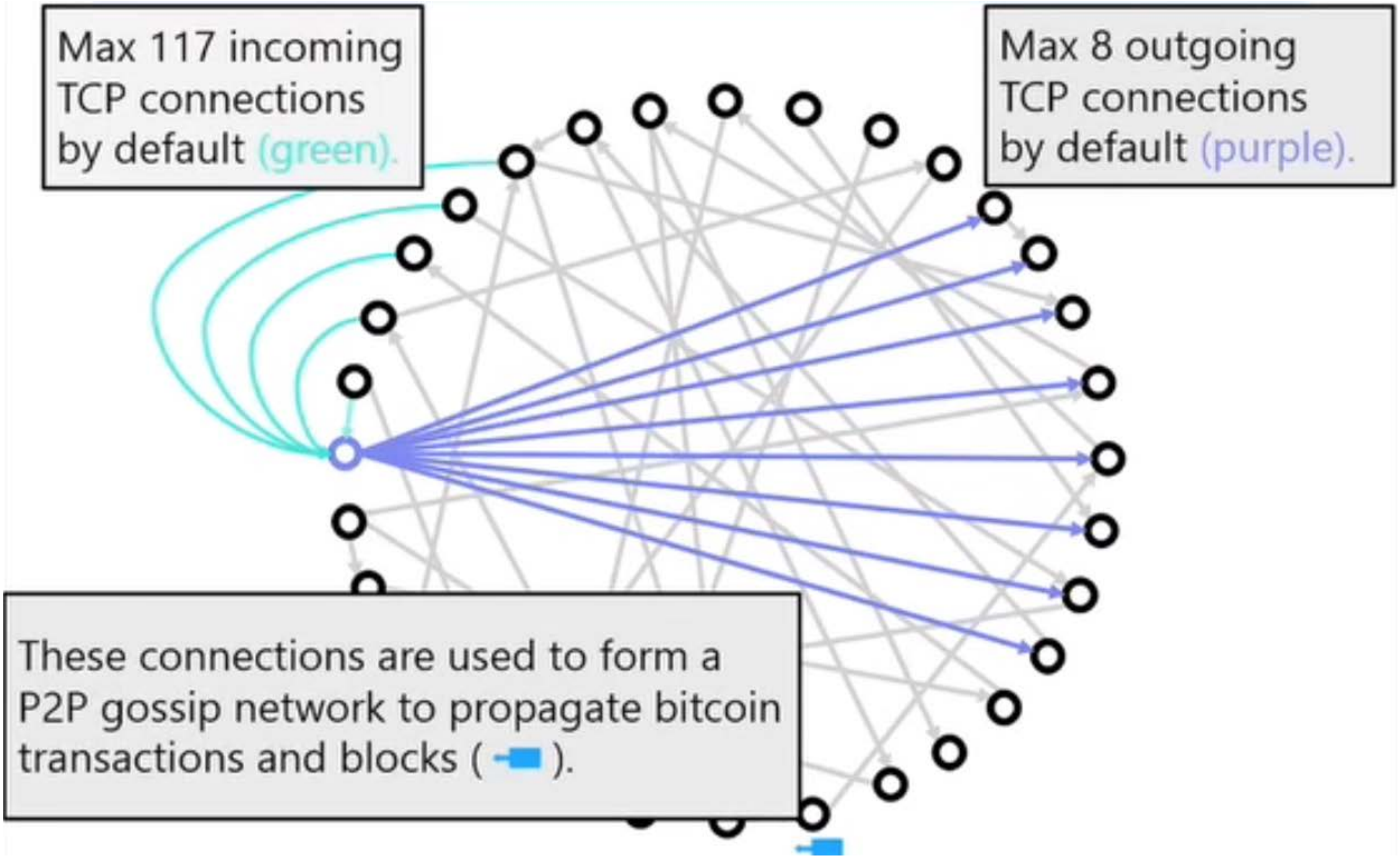
chapter 2
chapter 3
chapter 4
chapter 5
chapter 6

KAIST

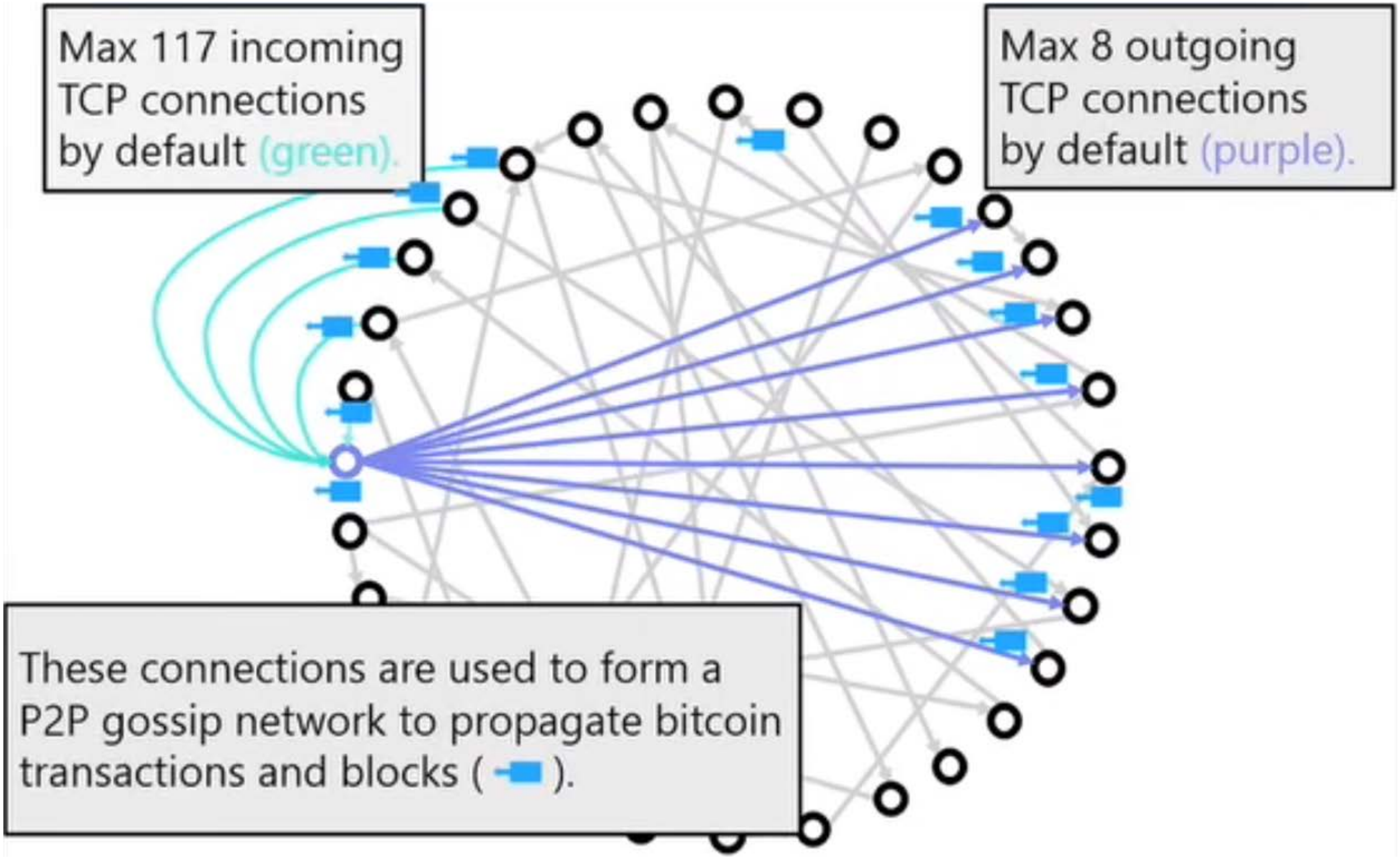
2. Eclipse Attacks & Implications – Bitcoin networking



2. Eclipse Attacks & Implications – Bitcoin networking



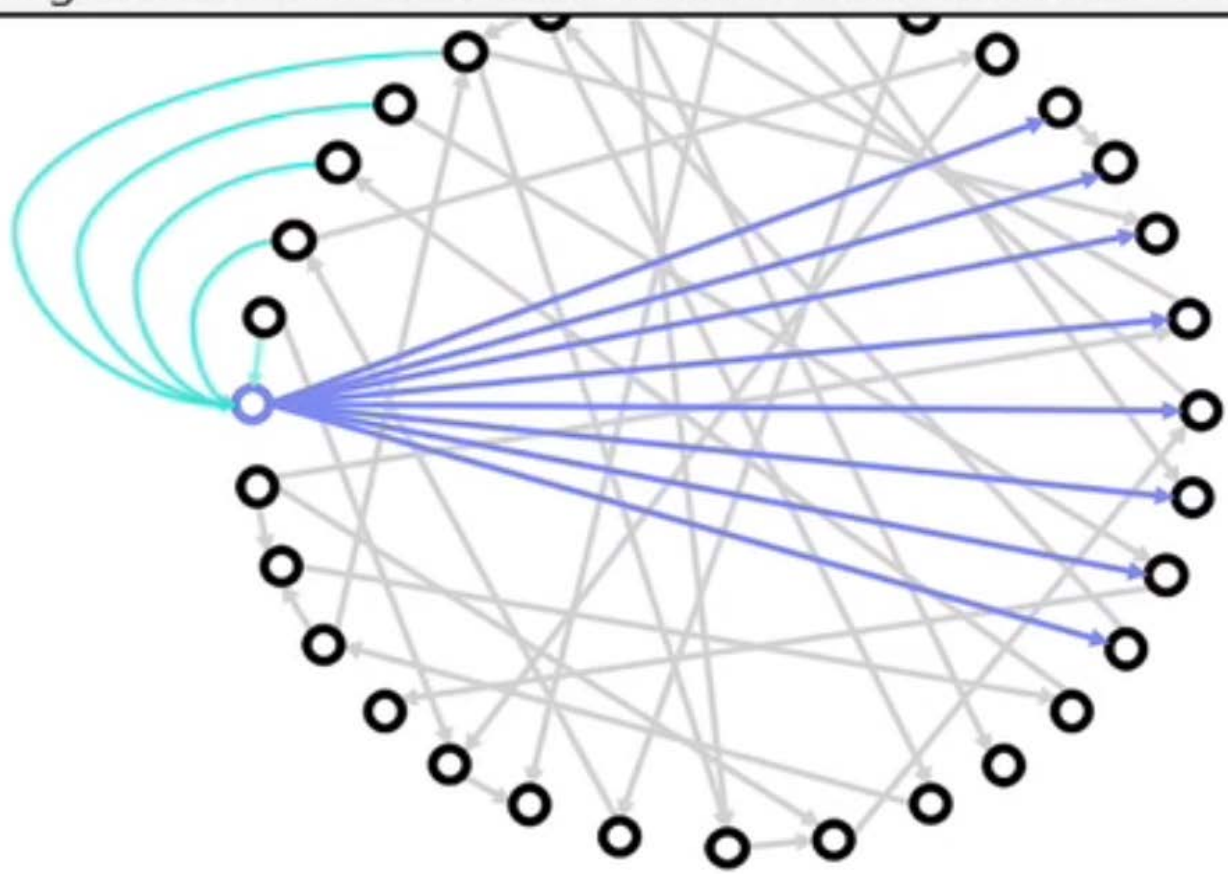
2. Eclipse Attacks & Implications – Bitcoin networking



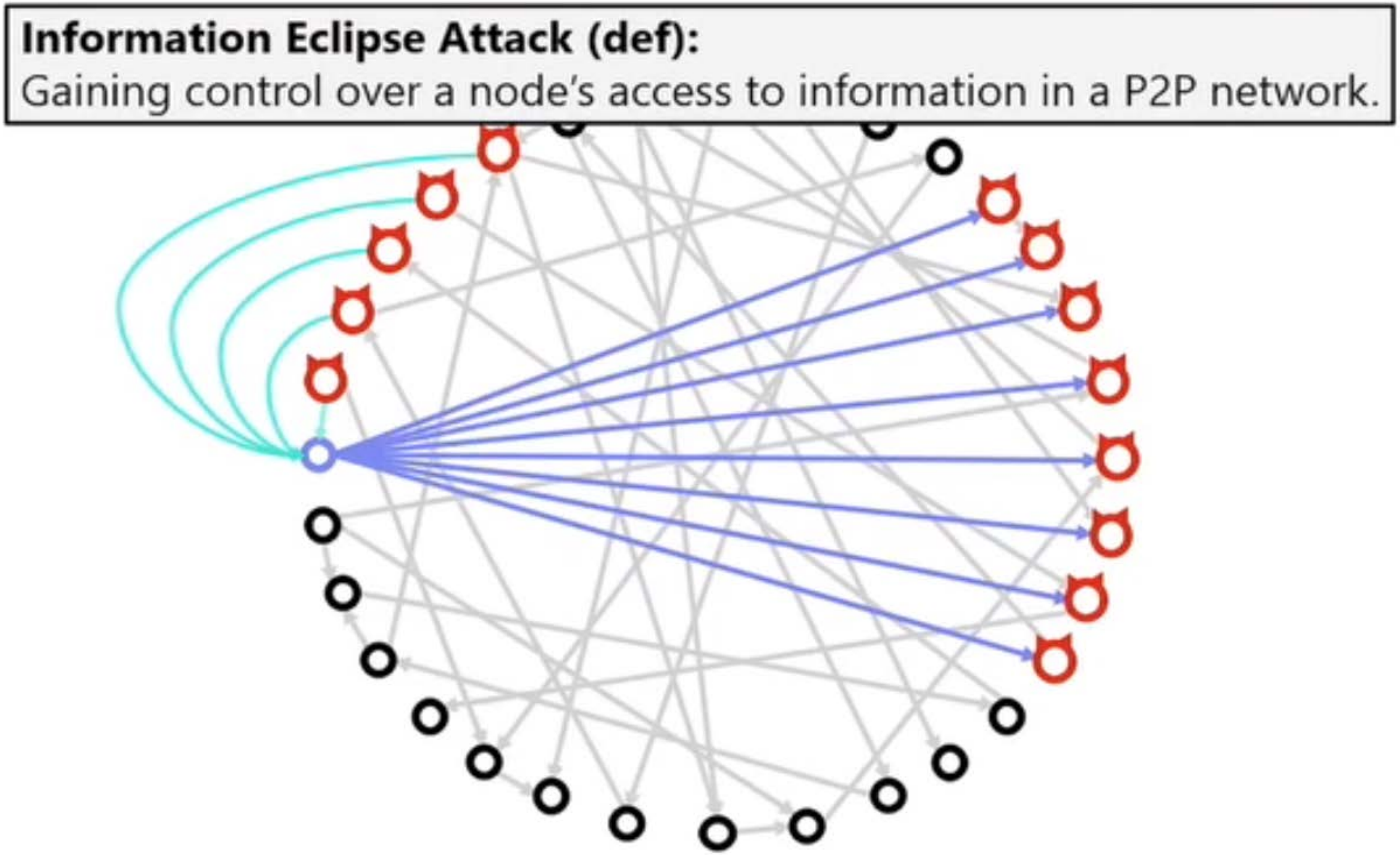
KAIST

2. Eclipse Attacks & Implications – Bitcoin networking

Information Eclipse Attack (def):
Gaining control over a node's access to information in a P2P network.



2. Eclipse Attacks & Implications – Eclipse Attack On Bitcoin



By manipulation the P2P net, the attacker eclipses the node

2. Eclipse Attacks & Implications – Eclipse Attack On Bitcoin

https://youtu.be/J-lF0zxGpu0?t=70

What are the problems?

2. Eclipse Attacks & Implications – Implications

1. Engineering block races
   - engineering & controlling blocks propagation

2. Splitting mining power
   - Making it eaiser to launch mining attacks

3. Selfish Mining
   - By eclipsing miners, the attacker increases gamma
   - Mining Pools -> their gateways to the public bitcoin network

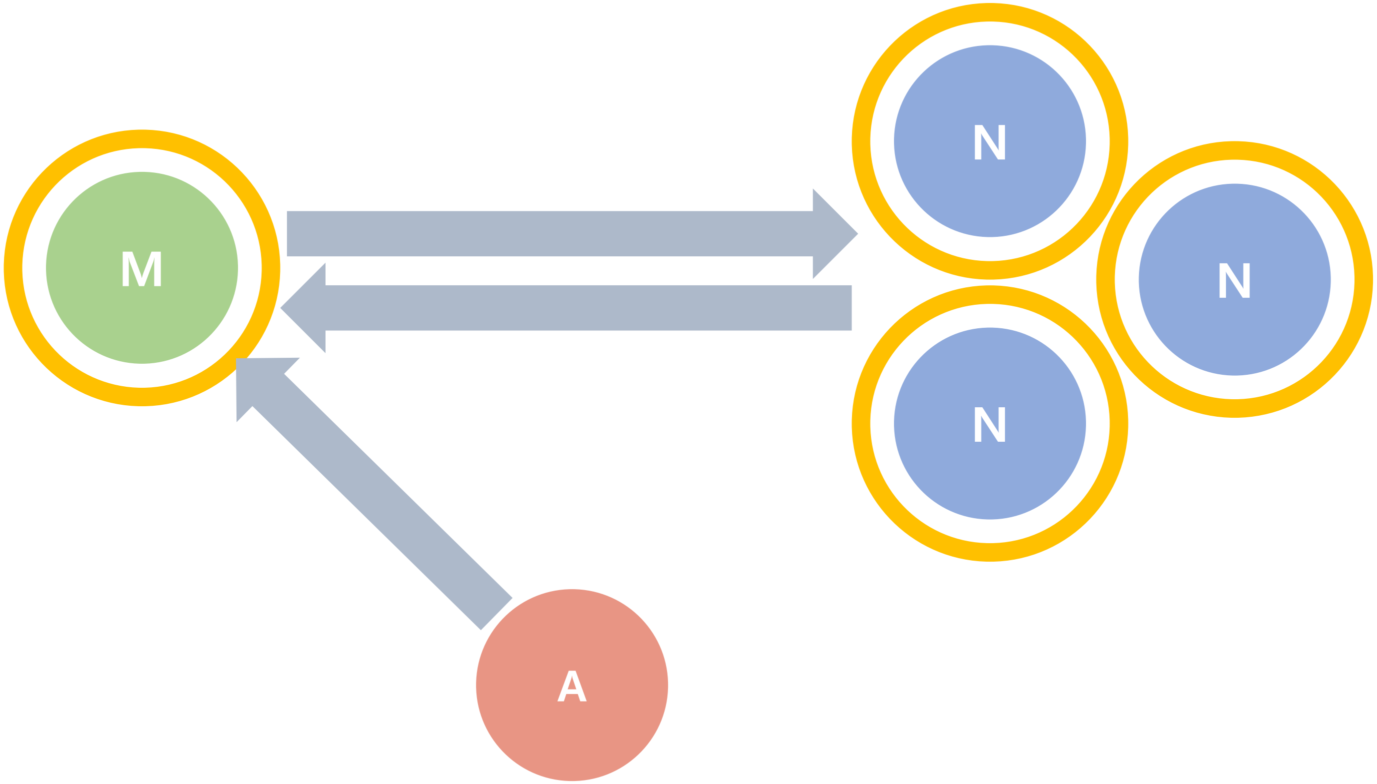
4. 0-Confirmation double spending
   - eclipse the merchant's bitcoin node
   - Send the merchants a tx T, but send T' to the rest of the network.

5. N-Confirmation double spending

KAIST

2. Eclipse Attacks & Implications

- N-Confirmation double spending



Chapter 3

: How to eclipse a Bitcoin node

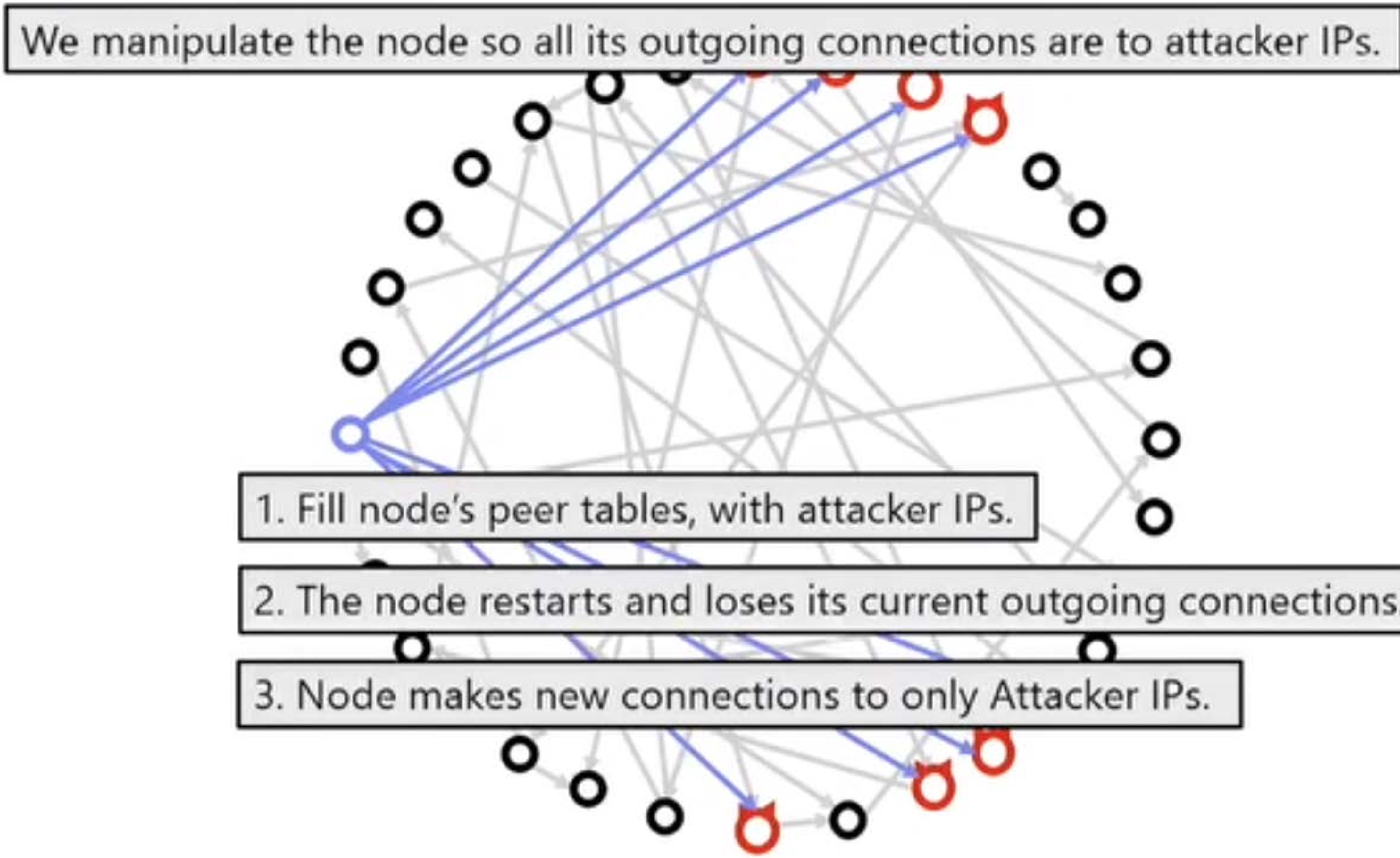
Outline

- Eclipse Attacks & Implications
  - Explanation about eclipse attack
  - 51% attack, Selfish Mining
  - N-confirmation double spending
- **How to eclipse a Bitcoin node**
  - P2P network of Bitcoin
  - How to exploit Bitcoin's P2P networking
- How many IPs does the attacker need?
- Countermeasures
- Eclipse Attack on Ethereum

chapter 2
chapter 3
chapter 4
chapter 5
chapter 6

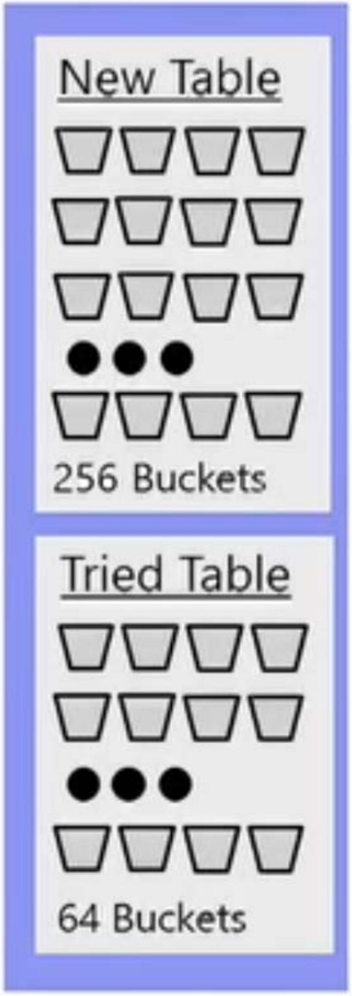
KAIST

3. Eclipse Attack on Bitcoin – Simlpe Overview



KAIST

3. Eclipse Attack on Bitcoin – 2 Tables

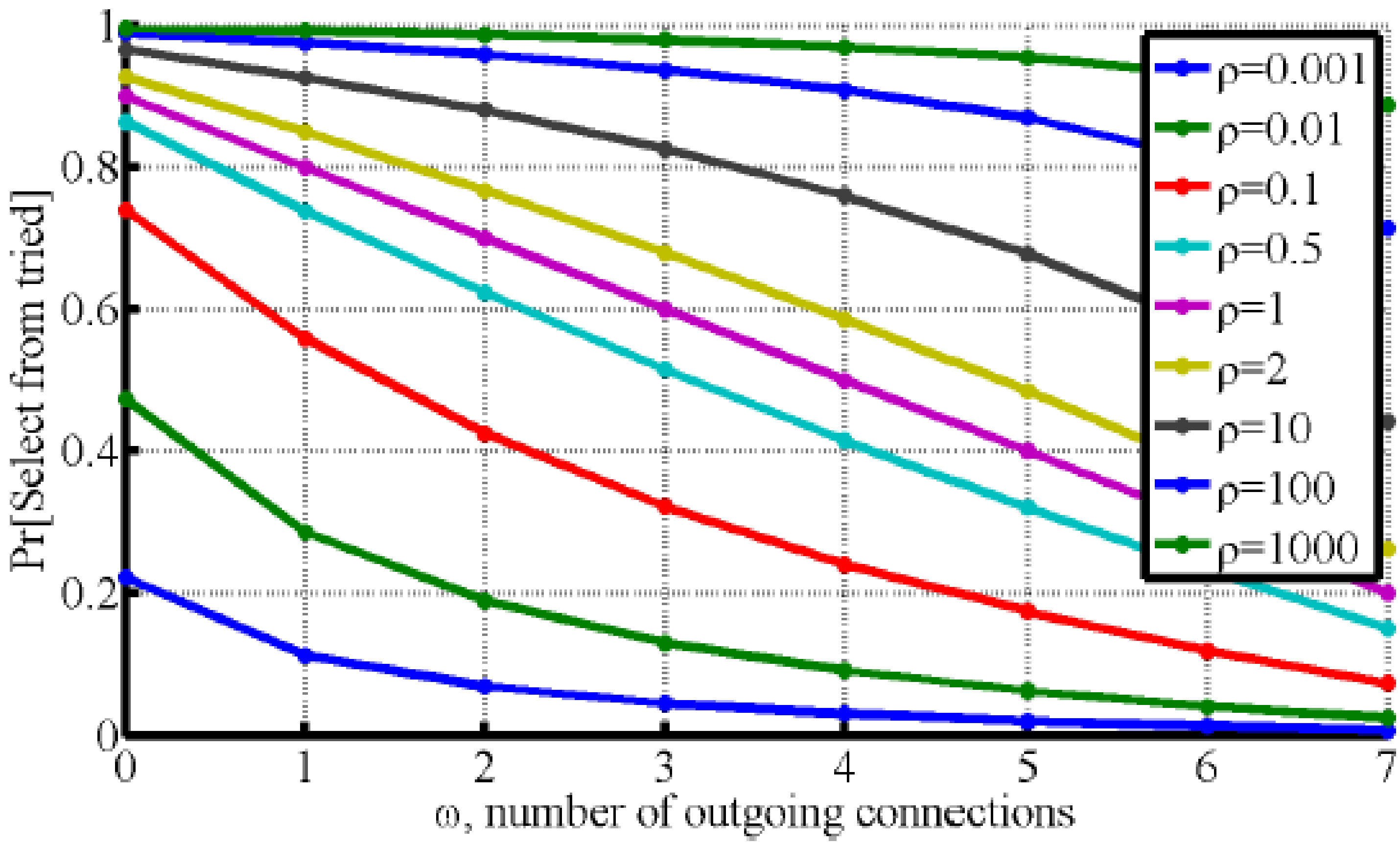


- Each node selects its peers from IP addresses stored in two tables.
  - New table : IPs the node has heard about.
  - Tried table: IPs the node peered with at some point
- Each bucket has 64 unique IP addresses.
- The tables also store a timestamp for each IP
- To find an IP to make an outgoing connection to:
  1. Choose new or tried tables to select from
  2. Select an IP biased toward **"fresher"** timestamps
  3. Attempt an outgoing connection to that IP

Attacker ensures that its IPs are fresher.
They are more likely to be selected as outgoing connection

KAIST

3. Eclipse Attack on Bitcoin – Peer Selection

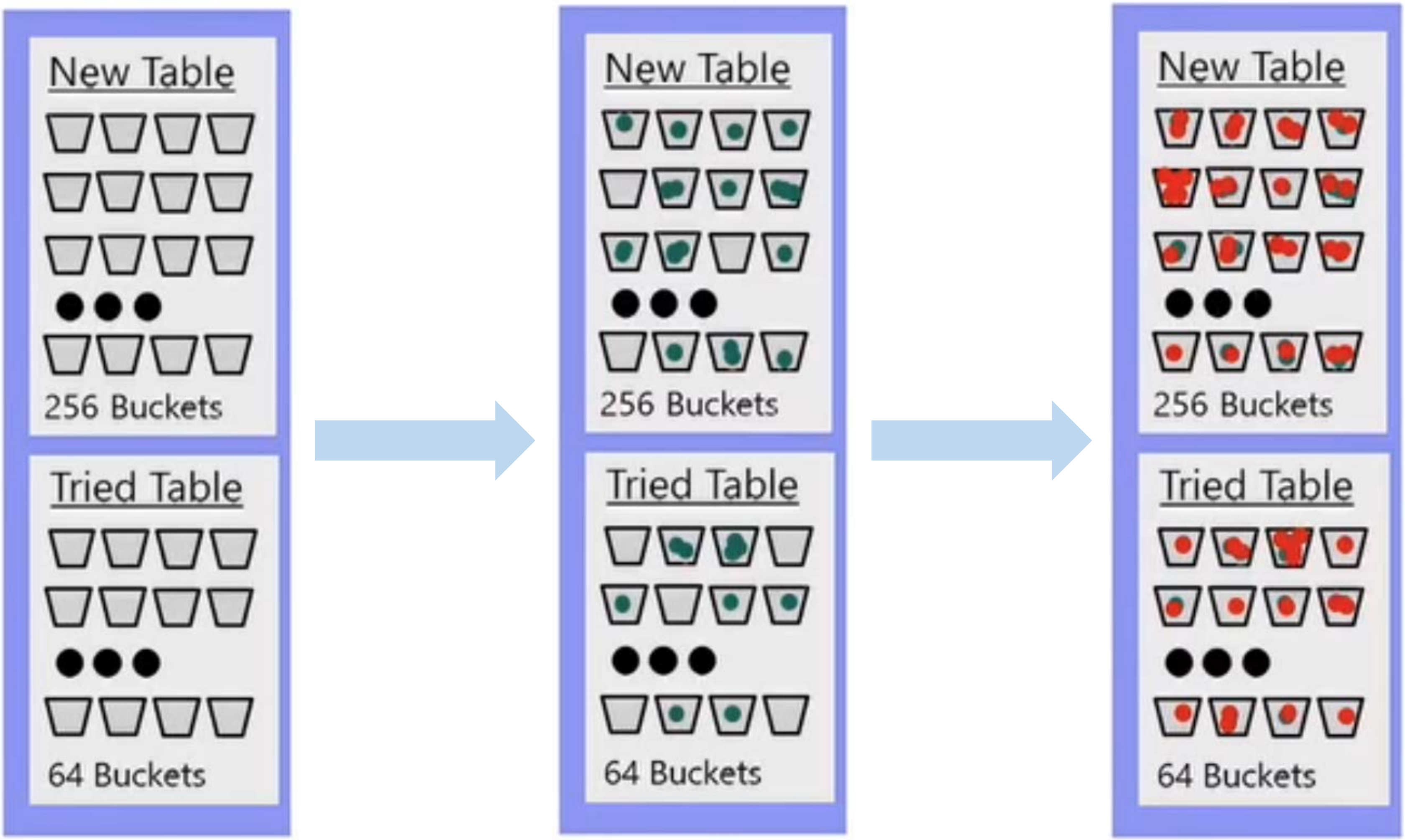


$$\Pr[\text{Select from } \texttt{tried}] = \frac{\sqrt{\rho}(9-\omega)}{(\omega+1)+\sqrt{\rho}(9-\omega)}$$

$$p(r,\tau) = \min(1, \tfrac{1.2^r}{1+\tau})$$

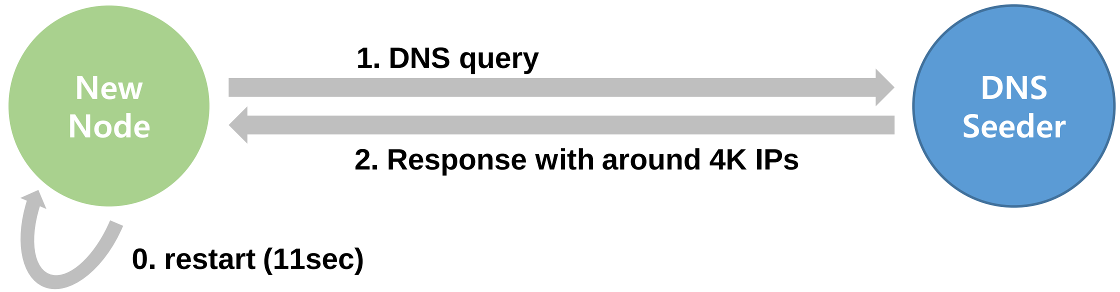
KAIST

3. Eclipse Attack on Bitcoin – Polluting 2 Tables

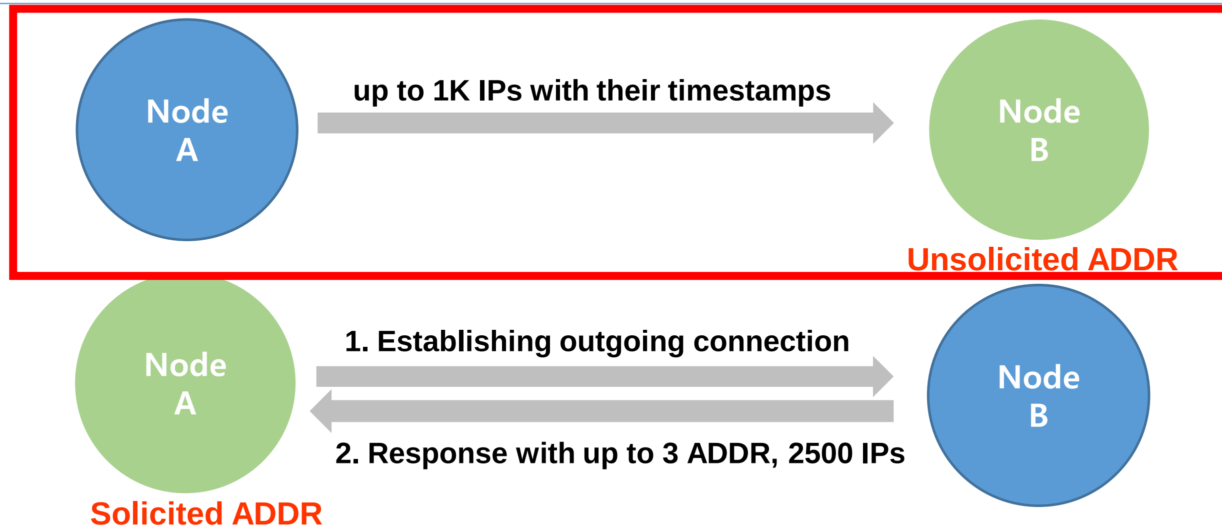


3. Eclipse Attack on Bitcoin – Propagating network information

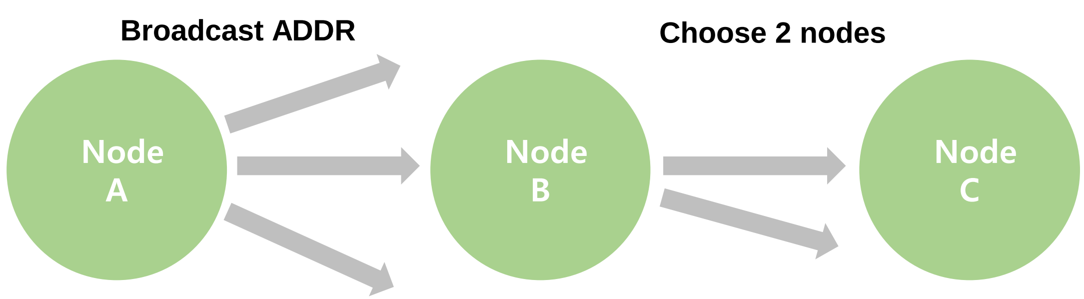
- Join/Re-join Case



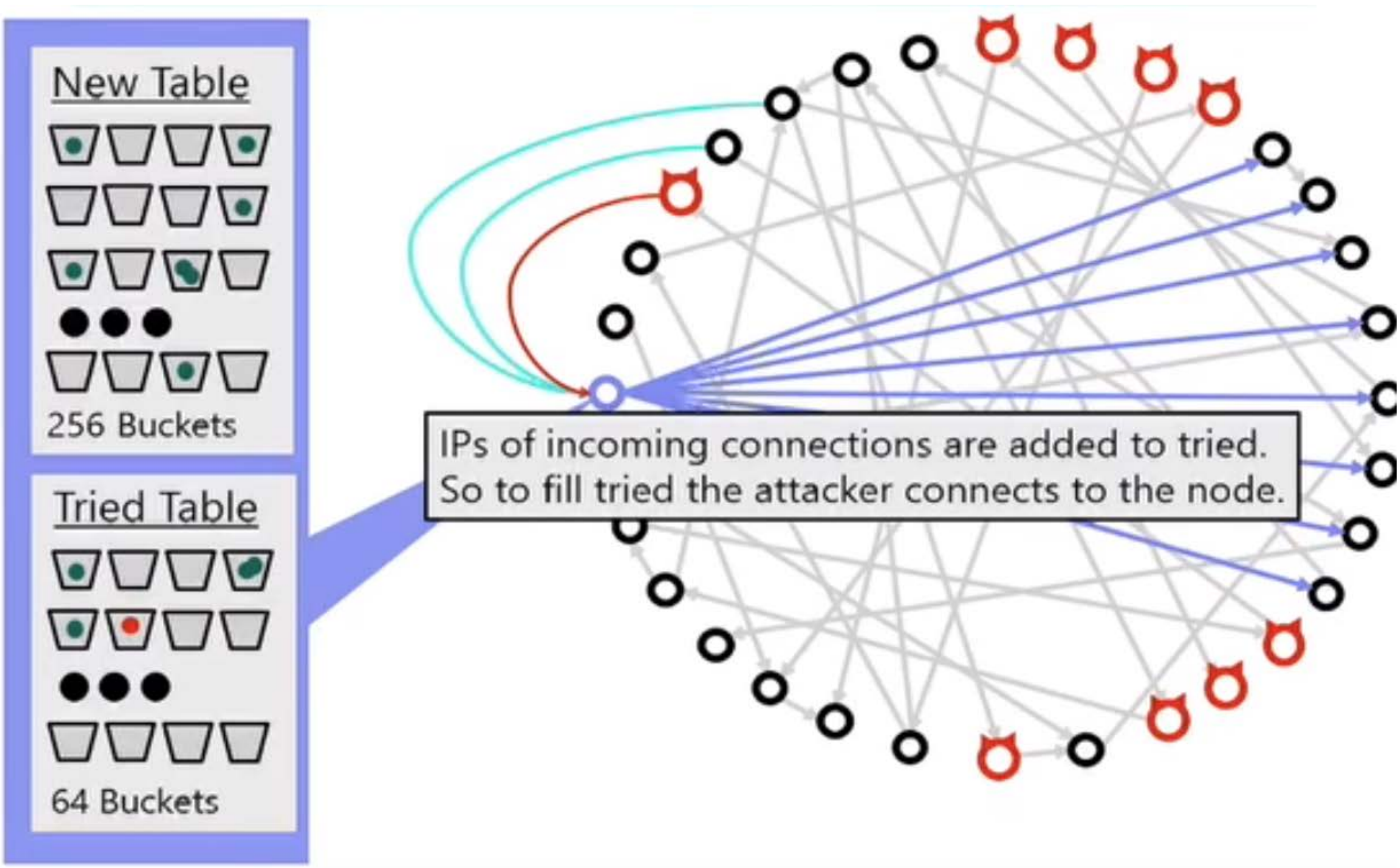
- ADDR msg



- Periodical Hello msg



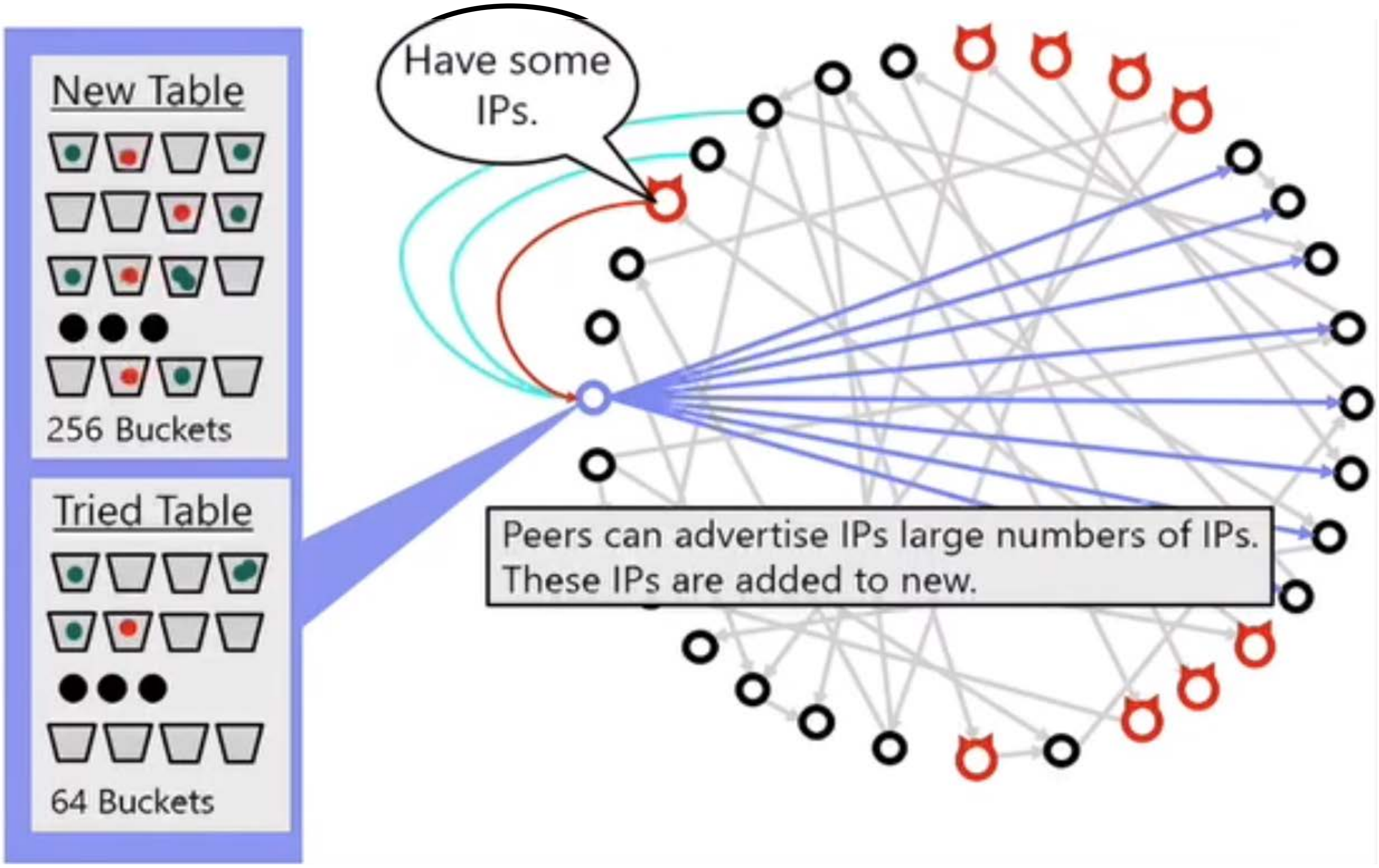
3. Eclipse Attack on Bitcoin – Tried Table polluting



1 slot per 1 incoming connection

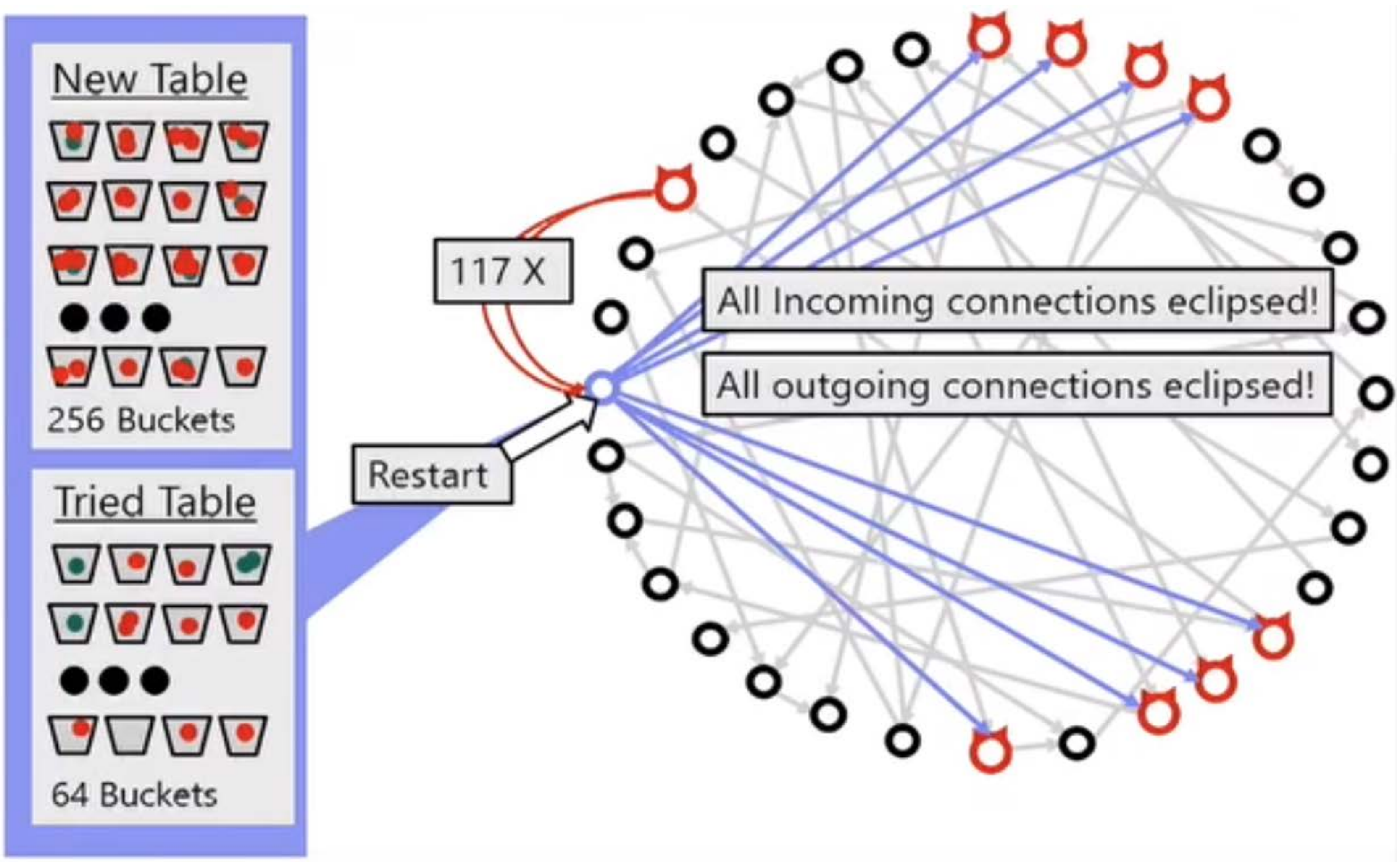
KAIST

3. Eclipse Attack on Bitcoin – New table polluting



1000 slots of New table per 1 ADDR message
-> Use trash IPs for New table pollution

3. Eclipse Attack on Bitcoin – Eclipsing target node



Polluting entire New table & almost Tried Table
Not finished!

KAIST

3. Eclipse Attack on Bitcoin – Eclipsing target node

https://youtu.be/J-lF0zxGpu0?t=425

3. Eclipse Attack on Bitcoin – Restart Target Node

- Eclipse Attack requires the target/victim node restart.

- Software/Security Updates
  - Predictable for the attacker, users are notified of upcoming updates
  - lose for the victim, restart or remain vulnerable

- Packets of Death/Dos Attacks
  - Ten Dos CVEs in Bitcoin[1], many more on underlying OSes.

- Power/Network Failures
  - Bitcoin nodes have 25% chance of going offline within 10 hours[2]

After restart, victim node select new outgoing connections from the tables!

[1]: https://en.bitcoin.it/wiki/Common_Vulnerabilities_and_Exposures
[2]: Biryukov, A. et al., Deanonymisation of clients in Bitcoin P2P network

KAIST

Chapter 4

: How many IPs does the attacker need?

KAIST

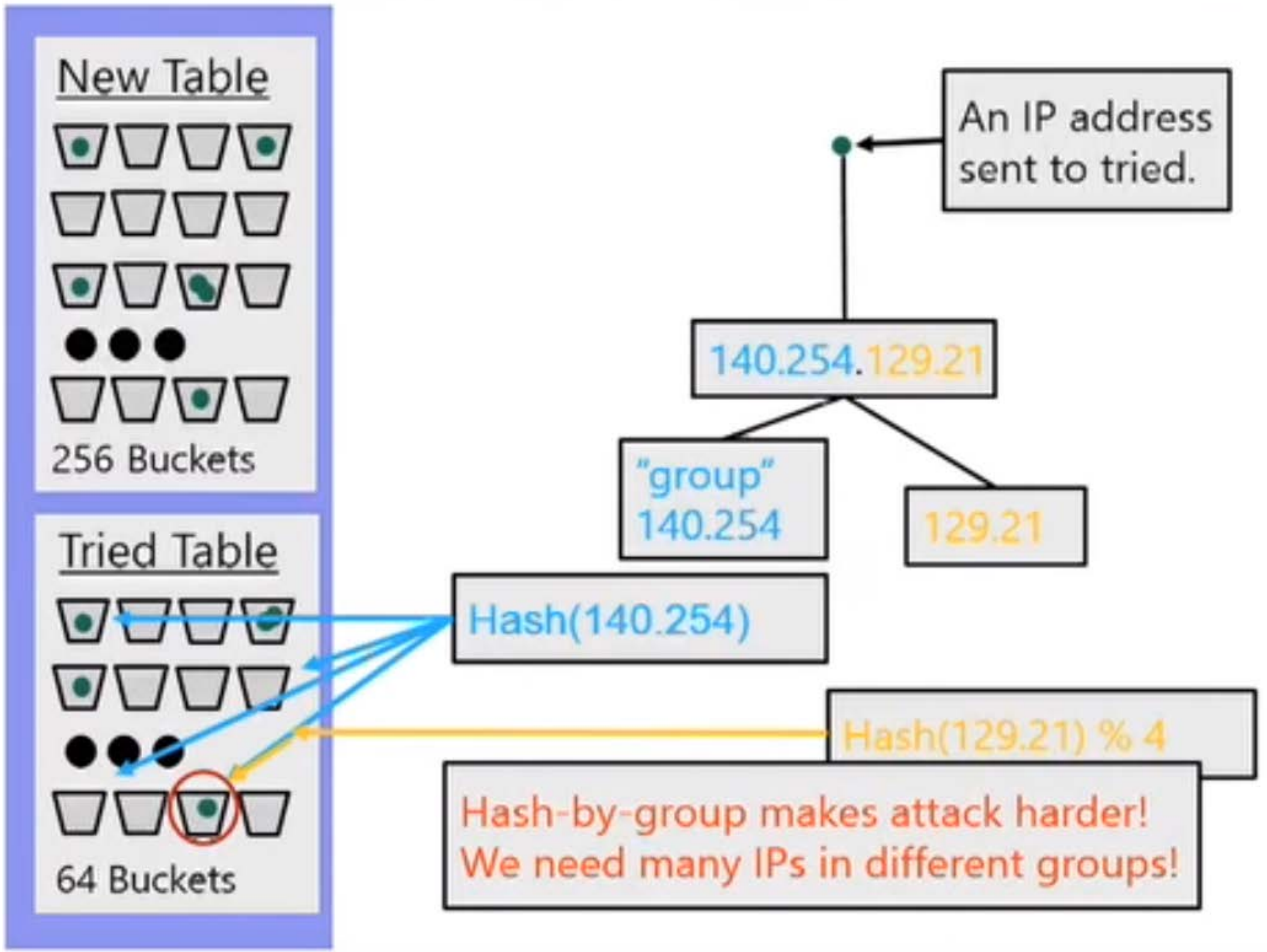
Outline

- **Eclipse Attacks & Implications**
  - Explanation about eclipse attack
  - 51% attack, Selfish Mining
  - N-confirmation double spending
- **How to eclipse a Bitcoin node**
  - P2P network of Bitcoin
  - How to exploit Bitcoin's P2P networking
- **How many IPs does the attacker need?**
  - Models & Experimental Results
  - Botnets, Infrastructure attack
- **Countermeasures**
- **Eclipse Attack on Ethereum**

chapter 2
chapter 3
chapter 4
chapter 5
chapter 6

KAIST

4. Eclipse Attack on Bitcoin – IP Insertion



Filling New table is easy to do, even though it also does Hash-by-group

KAIST

4. Eclipse Attack on Bitcoin – Use limited # of IPs

- The attack gets eaiser IF

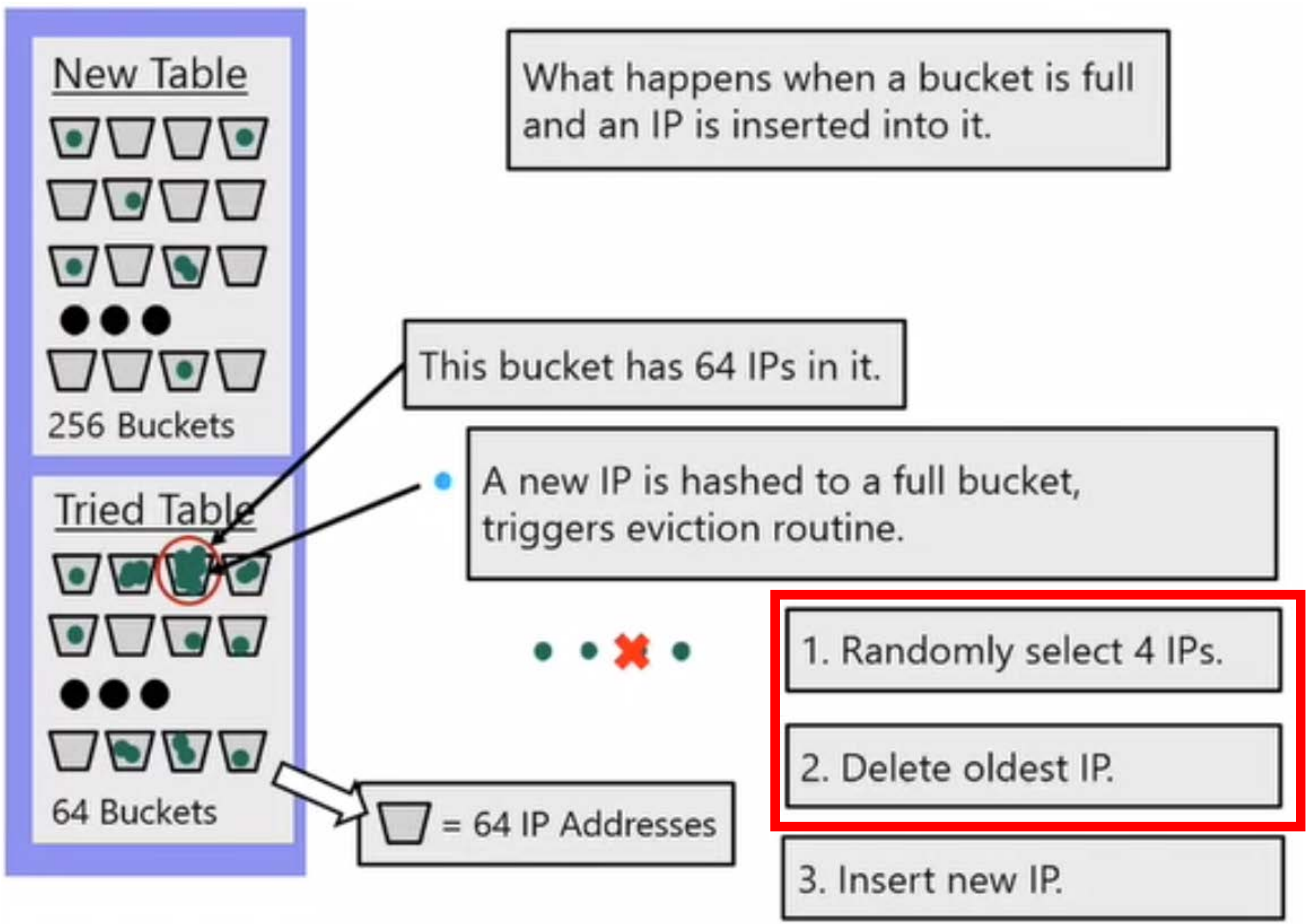
1. More attacker IPs in distinct groups
2. Few honest IPs in the tried table

- **Due to Hash-by-Group. Need many IPs in different group**

3. Stale honest IPs in the tried table
4. Fresh attacker IPs in the tried table

- **can age honest IPs by investing more time**
- **can ensure fresh IPs by continually filling the new table**

4. Eclipse Attack on Bitcoin – Bucket Eviction by Investing Time



Actually, move to New and deleted

KAIST

4. Eclipse Attack on Bitcoin – Modeling and Simulating

- Approach

1. Model Bitcoin with probability analysis & Monte-Carlo simulations
2. Use these models to determine effective attack parameters.
3. Experimentally verify these parameters against Bitcoin nodes

- Botnet vs Infrastructure

1. Botnet attacker holds several IPs, each in a distinct group
2. Infrastructure attacker holds several IPs blocks (same group)

4. Eclipse Attack on Bitcoin – Botnet Attack

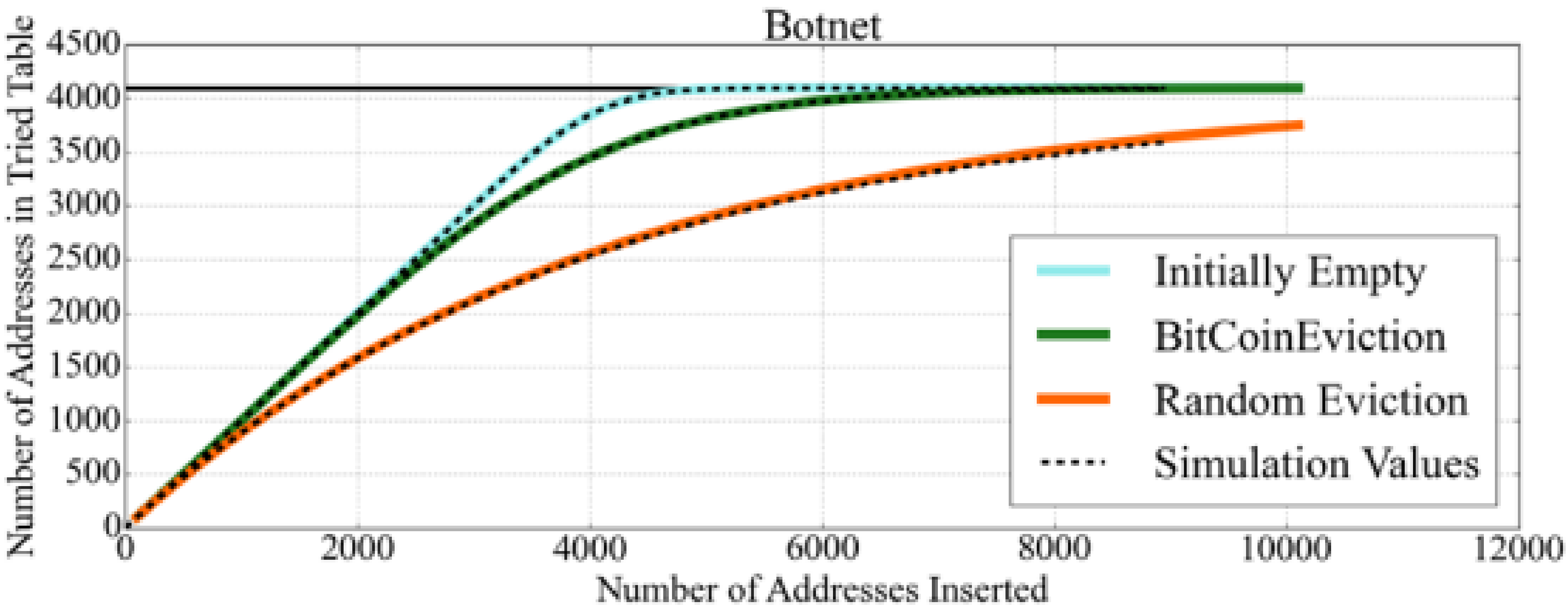


Figure. Botnet Attack simulation results

4. Eclipse Attack on Bitcoin – Infrastructure Attack

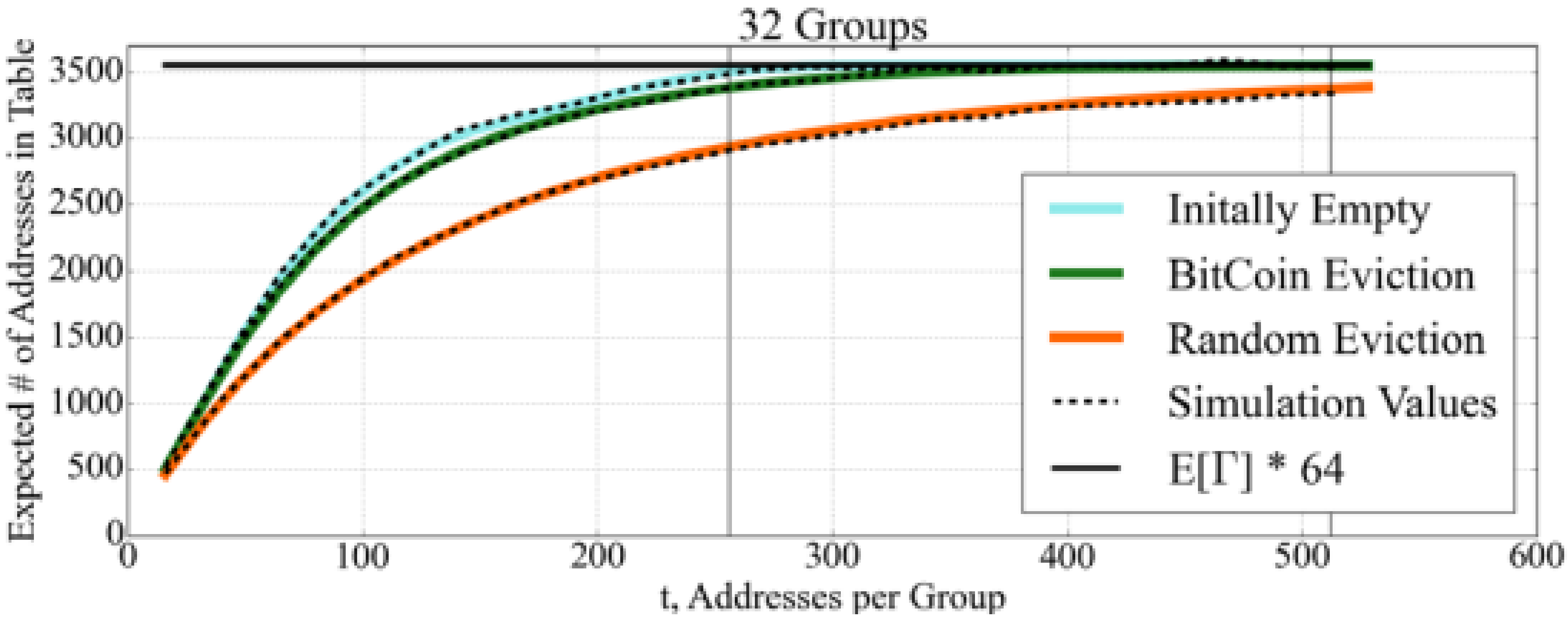


Figure. Infrastructure Attack simulation results

4. Eclipse Attack on Bitcoin – Botnet Results (Worst case)

Before Attack

- Artificially fill a node's tried table
- Tried table is 99.9% full of honest IPs (4093 IPs)

- Botnet of 4600 IPs, 2 IPs per group
- 5 hours invested, 26 min interval

After Attack

- Node's tried table is now 98.8% attacker IPs

- 100% attacker success rate, all 8 outgoing connections eclipsed

KAIST

4. Eclipse Attack on Bitcoin – Infrastructure Results (Worst case)

Before Attack

- Artificially fill a node's tried table
- Tried table is 99.8% full of honest IPs (4090 IPs)

- 32 groups of size /24(256 addresses), total 8192 IPs
- 10 hours invested, 43 min interval

After Attack

- Node's tried table is now 83.1% attacker IPs

- 98% attacker success rate, all 8 outgoing connections eclipsed

KAIST

4. Eclipse Attack on Bitcoin – Live Experiment

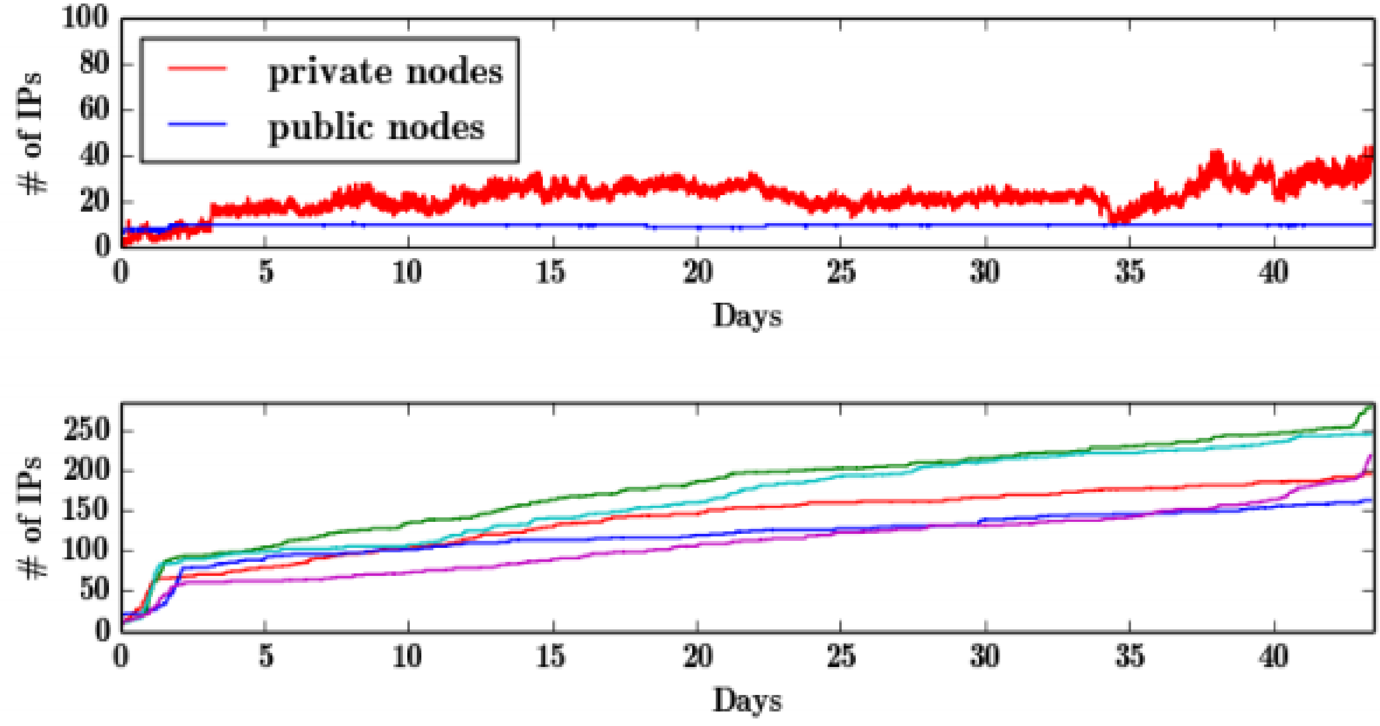


Figure. # of Connections, Tried entries

KAIST

4. Eclipse Attack on Bitcoin – Botnet Results (Live)

Before Attack

- Connected a node to the bitcoin network for +50 days
- Tried table has 298 honest IPs

- Botnet of 400 IPs, 1 IPs per group
- 1 hours invested, 90 sec interval

After Attack

- Node’s tried table is still mostly empty, but 57% are attacker IPs

- 84% attacker success rate, all 8 outgoing connections eclipsed

4. Eclipse Attack on Bitcoin – Infrastructure Results (Live)

Before Attack

- Connected a node to the bitcoin network for +50 days
- Tried table has 346 honest IPs

- 20 groups of size /24(256 addresses), total 5120 IPs
- 1 hours invested, 27 min interval

After Attack

- 94% are attacker IPs (Tried Table)

- 84% attacker success rate, all 8 outgoing connections eclipsed

4. Eclipse Attack on Bitcoin

Attack Type	Attacker resources					Experiment							Predicted		
	grps s	addrs/ grp t	total addrs	τ_ℓ, time invest	τ_a, round	Total pre-attack new	Total pre-attack tried	Total post-attack new	Total post-attack tried	Attack addrs new	Attack addrs tried	Wins	Attack addrs new	Attack addrs tried	Wins
Infra (Worstcase)	32	256	8192	10 h	43 m	16384	4090	16384	4096	15871	3404	**98%**	16064	3501	**87%**
Infra (Transplant)	20	256	5120	1 hr	27 m	16380	278	16383	3087	14974	2947	**82%**	15040	2868	**77%**
Infra (Transplant)	20	256	5120	2 hr	27 m	16380	278	16383	3088	14920	2966	**78%**	15040	2868	**87%**
Infra (Transplant)	20	256	5120	4 hr	27 m	16380	278	16384	3088	14819	2972	**86%**	15040	2868	**91%**
Infra (Live)	20	256	5120	1 hr	27 m	16381	346	16384	3116	14341	2942	**84%**	15040	2868	**75%**
Bots (Worstcase)	2300	2	4600	5 h	26 m	16080	4093	16384	4096	16383	4015	**100%**	16384	4048	**96%**
Bots (Transplant)	200	1	200	1 hr	74 s	16380	278	16384	448	16375	200	**60%**	16384	200	**11%**
Bots (Transplant)	400	1	400	1 hr	90 s	16380	278	16384	648	16384	400	**88%**	16384	400	**34%**
Bots (Transplant)	400	1	400	4 hr	90 s	16380	278	16384	650	16383	400	**84%**	16384	400	**61%**
Bots (Transplant)	600	1	600	1 hr	209 s	16380	278	16384	848	16384	600	**96%**	16384	600	**47%**
Bots (Live)	400	1	400	1 hr	90 s	16380	298	16384	698	16384	400	**84%**	16384	400	**28%**

Table 2: Summary of our experiments.

Which one is better?

Is Bitcoin safe?

Chapter 5

: Countermeasures

Outline

- **Eclipse Attacks & Implications** — chapter 2
  - Explanation about eclipse attack
  - 51% attack, Selfish Mining
  - N-confirmation double spending
- **How to eclipse a Bitcoin node** — chapter 3
  - P2P network of Bitcoin
  - How to exploit Bitcoin's P2P networking
- **How many IPs does the attacker need?** — chapter 4
  - Models & Experimental Results
  - Botnets, Infrastructure attack
- **Countermeasures** — chapter 5
  - Effectiveness of countermeasures
  - Current deployment
- **Eclipse Attack on Ethereum** — chapter 6

5. Countermeasures : Random Selection

- Vunlerability 1 – Selection Bias

Attacker ensures its IPs are fresher so they are more likely to be selected

- Countermeasure : Random Selection

Randomly select IPs with no bias toward fresher timestamps

$$p(r, \tau) = \min(1, \frac{1.2^r}{1+\tau})$$

5. Countermeasures : Deterministic Random Eviction

- Vunlerability 2 – Eviction Bias

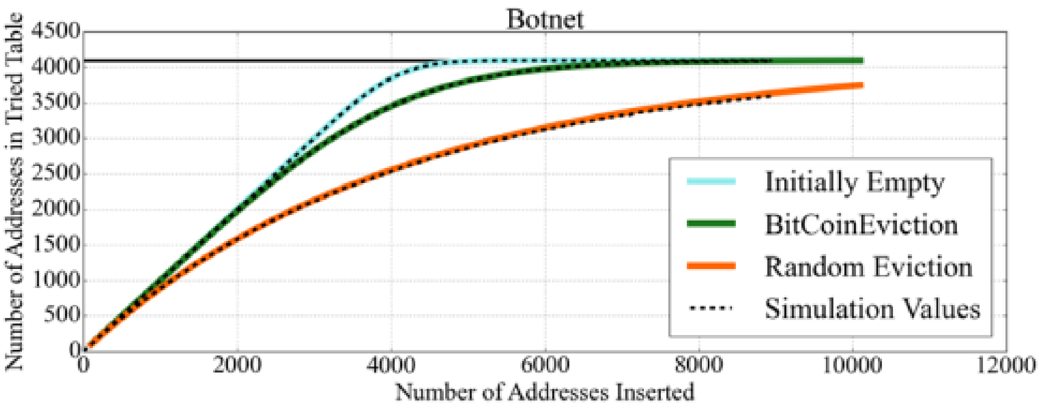
Attacker exploited Eviction bias toward older IPs

- Vunlerability 3 – Try Again

Attacker exploited randomness in eviction process to improve odds of stuffing tried table by running the attack multiple times

- Countermeasure : Deterministic Random Eviction

Deterministically map IPs to buckets and positions in buckets, evicting whatever happens to be in that position



5. Countermeasures : Feelers & Test-Before-Evict

- Problem:

Tried table fills up very slowly and contain mostly dead IPs.
The fewer honest IPs in tried

- Countermeasure : Feeler Connection

Make test connections to IPs in new to fill tried table faster

- Problem:

Good IP addresses from tried get evicted

- Countermeasure : Test Before Evict

Test IPs in tried before evicting them, if online do not evict

5. Countermeasures : Deployment

- Countermeasurement

1. Deterministic Random Eviction
2. Random Selection
6. More Buckets

→ **Bitcoind 10.1 version**

3. Test-Before-Evict
4. Feeler Connections

→ **In a Patch, awaiting review**

5. Anchor Connections

And More!

KAIST

5. Countermeasures : How Effective?

	No countermeasures	1,2,6 countermeasures Bitcoin 0.10.1	1,2,3,4,6 countermeasures Bitcoin 0.10.1 + patch
Full tried table (worst case)	**4600** IPs **100%** attacker success	**41,000** IPs (model) **~50%** attacker success	test-before-evict keeps attacker IPs out. **0%** attacker success
Live node (298 IPs)	**400** IPs **84%** attacker success	**3,700** IPs (model) **~50%** attacker success	

Better Security →

5. Summary

- Eclipse Attacks violate Bitcoin's core security gurantees
  - N-Confirmation double spending
  - 51% attack, Selfish mining, and so on

- The paper develop practical attacks
  - A botnet of 400 IPs is sufficient
  - In an attackers worst case >> a botnet of 4600 IPs

- The paper have effective countermeasures to resist these attacks
  - Some of the countermeasures have already been deployed
  - Others are awaiting review

KAIST

Chapter 6

: Eclipse Attack on Ethereum

KAIST

Outline

- **Eclipse Attacks & Implications**
  - Explanation about eclipse attack
  - 51% attack, Selfish Mining
  - N-confirmation double spending
- **How to eclipse a Bitcoin node**
  - P2P network of Bitcoin
  - How to exploit Bitcoin's P2P networking
- **How many IPs does the attacker need?**
  - Models & Experimental Results
  - Botnets, Infrastructure attack
- **Countermeasures**
  - Effectiveness of countermeasures
  - Current deployment
- **Eclipse Attack on Ethereum**
  - simple case study of Ethereum Attack

chapter 2

chapter 3

chapter 4

chapter 5

chapter 6

KAIST

6. Eclipse Attack on Ethereum – Overview

1. Monopolizing Connection

2. Table Poisoning

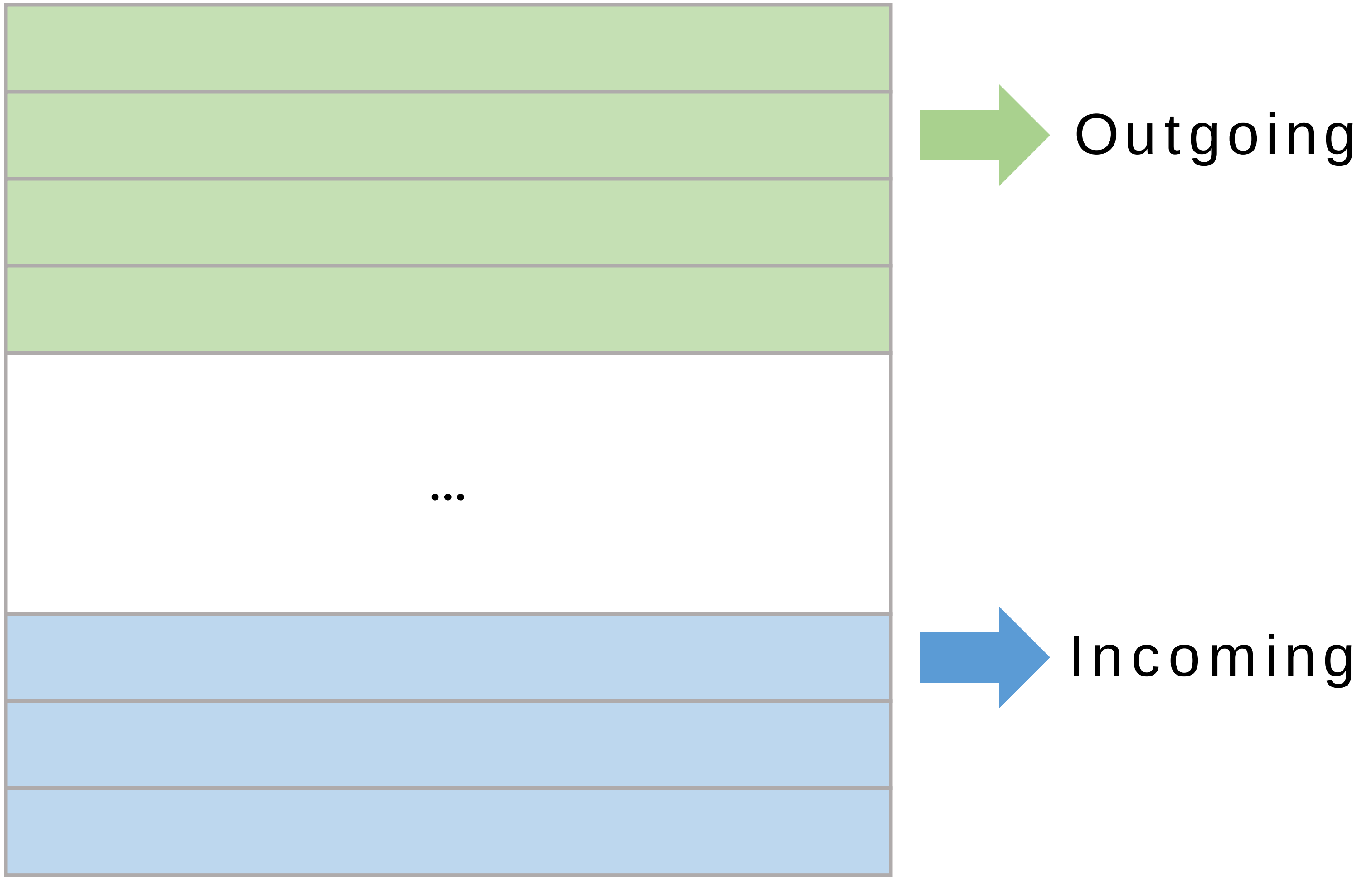
6. Eclipse Attack on Ethereum – Overview about Data Structure

- Table
  1. empty when the client reboot
  2. 256 buckets, 16 entries
  3. store node information prior to db
- DB
  1. DB is stored on disk, persistant
  2. information about nodes the client has seen

KAIST

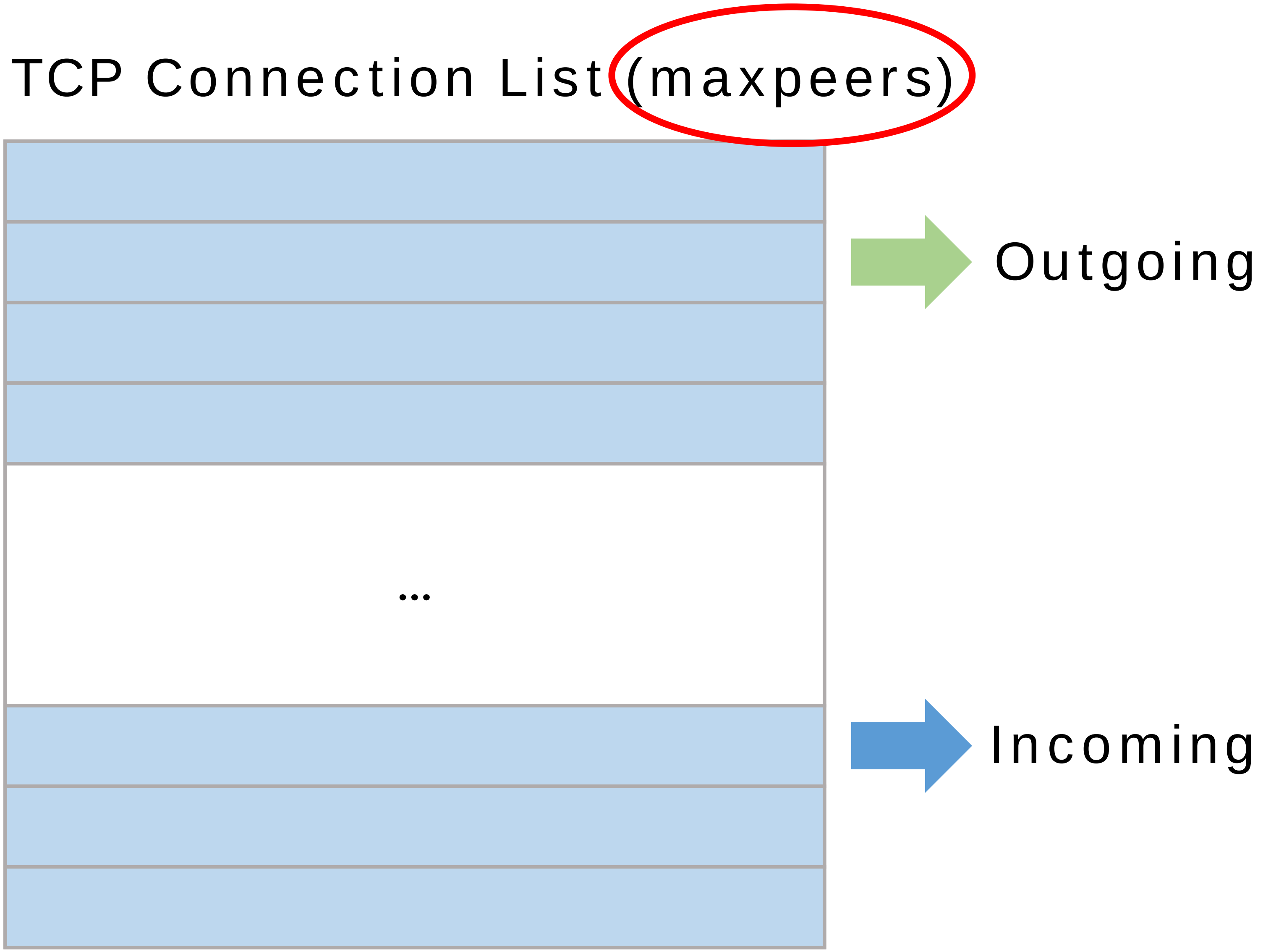
6. Eclipse Attack on Ethereum – Monopolizing Connections

TCP Connection List (maxpeers)



Using TCP to exchange blockchain information

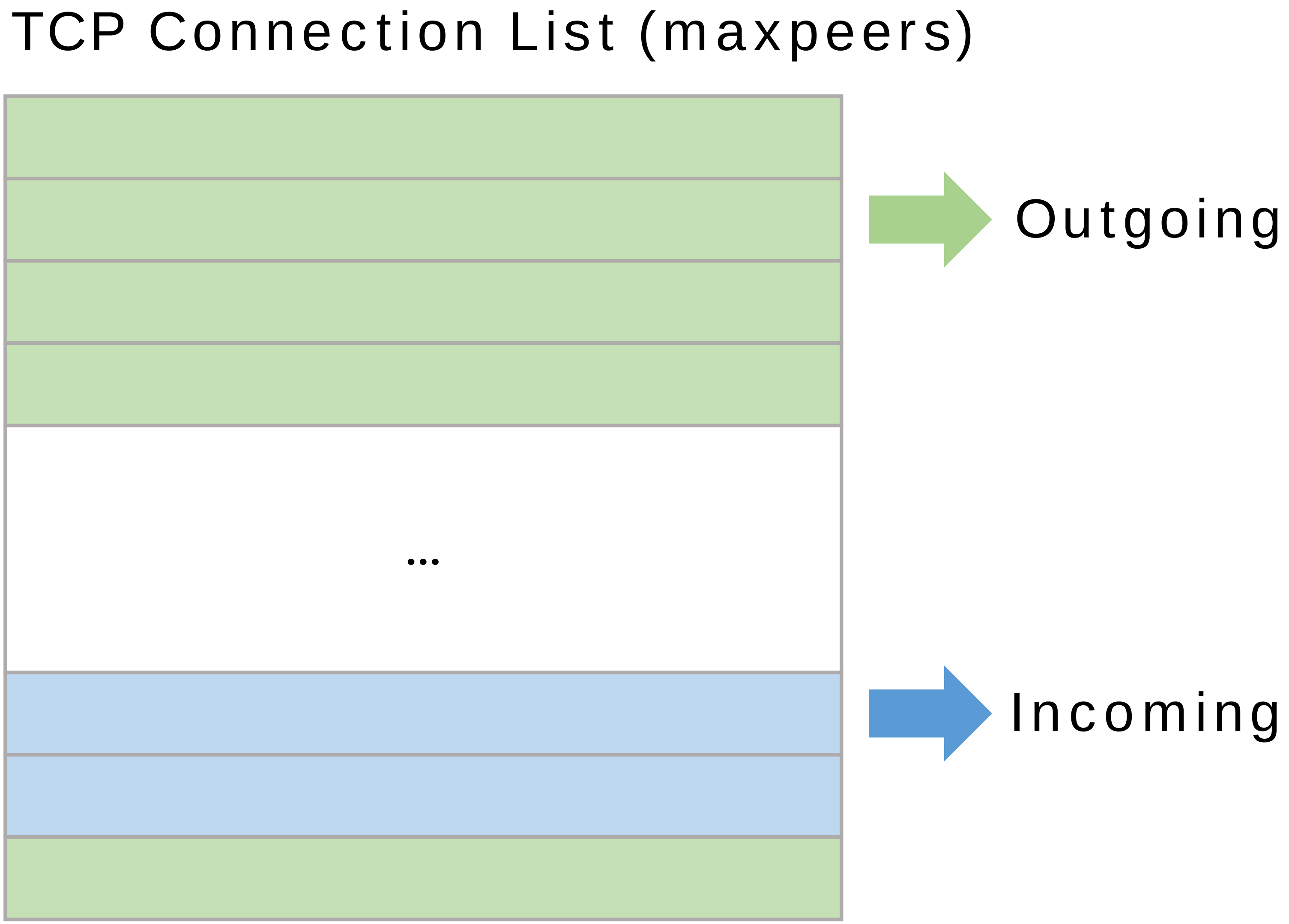
6. Eclipse Attack on Ethereum – Monopolizing Connections



When a client **reboot**, no incoming/outgoing
Establishing incoming is faster than outgoing(db)

KAIST

6. Eclipse Attack on Ethereum – Monopolizing Connections



Set "Upper limit" on number of incoming TCPs
(geth v1.8.0, limit = 8)

KAIST

6. Eclipse Attack on Ethereum – Overview

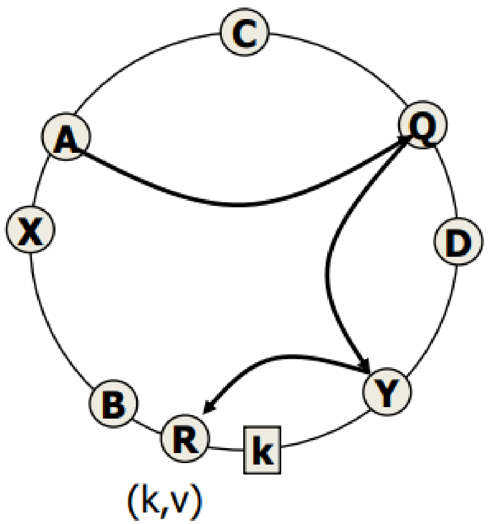
1. Monopolizing Connection

2. Table Poisoning

6. Eclipse Attack on Ethereum – Brief Review

DHT: Terminologies

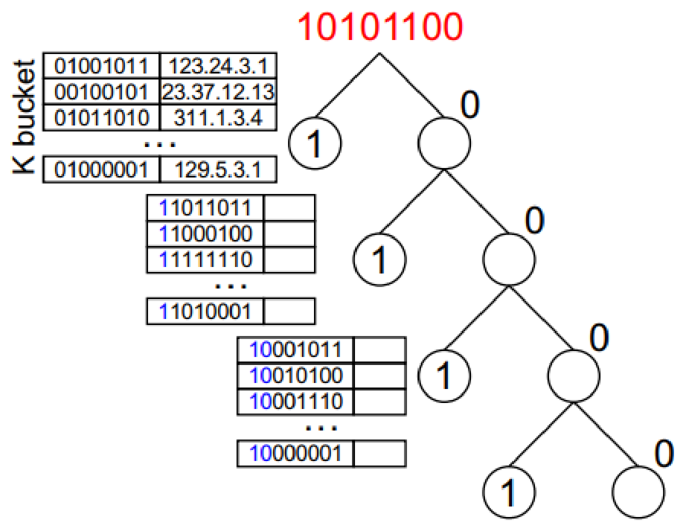
- Every node has a unique ID: *nodeID*
- Every object has a unique ID: *key*
- Keys and nodeIDs are logically arranged on a *ring* (*ID space*)
- A data object is stored at its *root(key)* and several *replica roots*
  - Closest nodeID to the key (or successor of k)

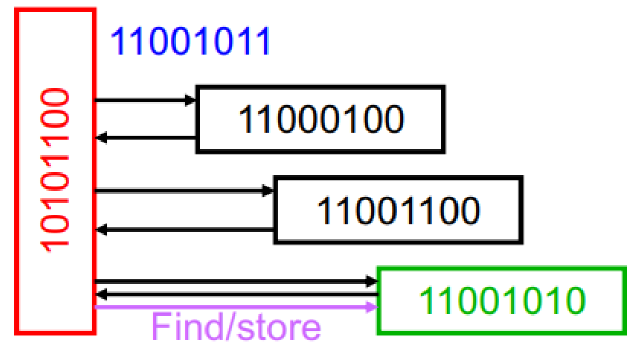


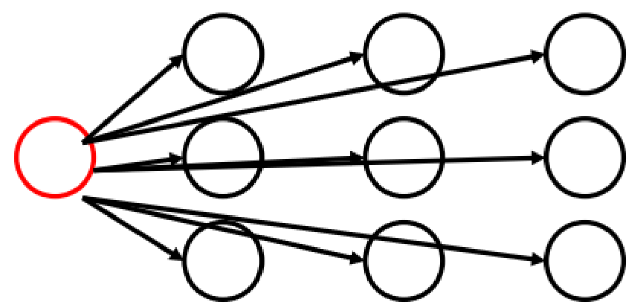
- *Range:* the set of keys that a node is responsible for
- Routing table size: O(log(N))
- Routing delay: O(log(N)) hops
- Content addressable!

6. Eclipse Attack on Ethereum – Brief Review

Kademlia Protocol







- d(X, Y) = X XOR Y
- An entry in k-bucket shares at least k-bit prefix with the nodeID
  - k=20 in overnet
- Add new contact if
  - k-bucket is not full

- Parallel, iterative, prefix-matching routing
- Replica roots: k closest nodes

KAIST

- **Ethereum is based on Kademlia,**

The Purpose of Kademlia Algorithm?

- For Neighboring, not find/store contents

The Problem of Kademlia Algorithm?

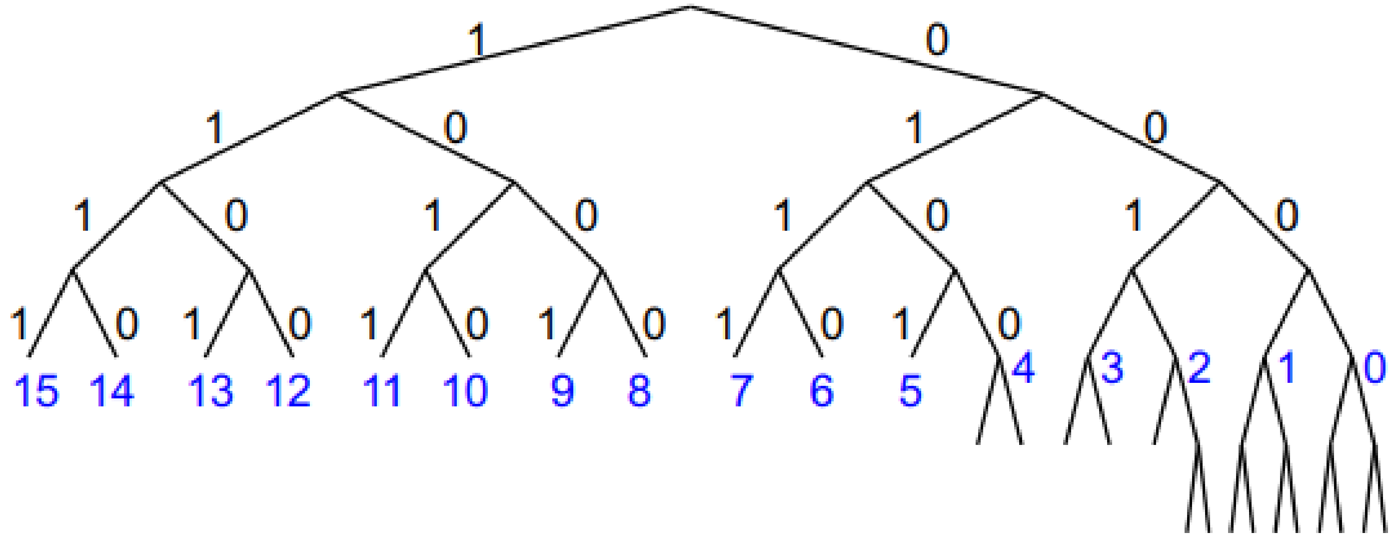
- Using Node Id, not IP
- Using XOR distance

6. Eclipse Attack on Ethereum – problem of this approach(1)

Attack Type	Attacker resources					Experiment							Predicted		
	grps s	addrs/ grp t	total addrs	τ_ℓ, time invest	τ_a, round	Total pre-attack new	Total pre-attack tried	Total post-attack new	Total post-attack tried	Attack addrs new	Attack addrs tried	Wins	Attack addrs new	Attack addrs tried	Wins
Infra (Worstcase)	32	256	8192	10 h	43 m	16384	4090	16384	4096	15871	3404	**98%**	16064	3501	**87%**
Infra (Transplant)	20	256	5120	1 hr	27 m	16380	278	16383	3087	14974	2947	**82%**	15040	2868	**77%**
Infra (Transplant)	20	256	5120	2 hr	27 m	16380	278	16383	3088	14920	2966	**78%**	15040	2868	**87%**
Infra (Transplant)	20	256	5120	4 hr	27 m	16380	278	16384	3088	14819	2972	**86%**	15040	2868	**91%**
Infra (Live)	20	256	5120	1 hr	27 m	16381	346	16384	3116	14341	2942	**84%**	15040	2868	**75%**
Bots (Worstcase)	2300	2	4600	5 h	26 m	16080	4093	16384	4096	16383	4015	**100%**	16384	4048	**96%**
Bots (Transplant)	200	1	200	1 hr	74 s	16380	278	16384	448	16375	200	**60%**	16384	200	**11%**
Bots (Transplant)	400	1	400	1 hr	90 s	16380	278	16384	648	16384	400	**88%**	16384	400	**34%**
Bots (Transplant)	400	1	400	4 hr	90 s	16380	278	16384	650	16383	400	**84%**	16384	400	**61%**
Bots (Transplant)	600	1	600	1 hr	209 s	16380	278	16384	848	16384	600	**96%**	16384	600	**47%**
Bots (Live)	400	1	400	1 hr	90 s	16380	298	16384	698	16384	400	**84%**	16384	400	**28%**

Table 2: Summary of our experiments.

6. Eclipse Attack on Ethereum – problem of this approach(2)



- **Using SHA3 to make Node ID 256 bits.**
- **“Table” -> 256 buckets, 16 entries each**
- **Since Kademlia uses XOR distance, “Table” info is public**

6. Eclipse Attack on Ethereum – Table Poisoning

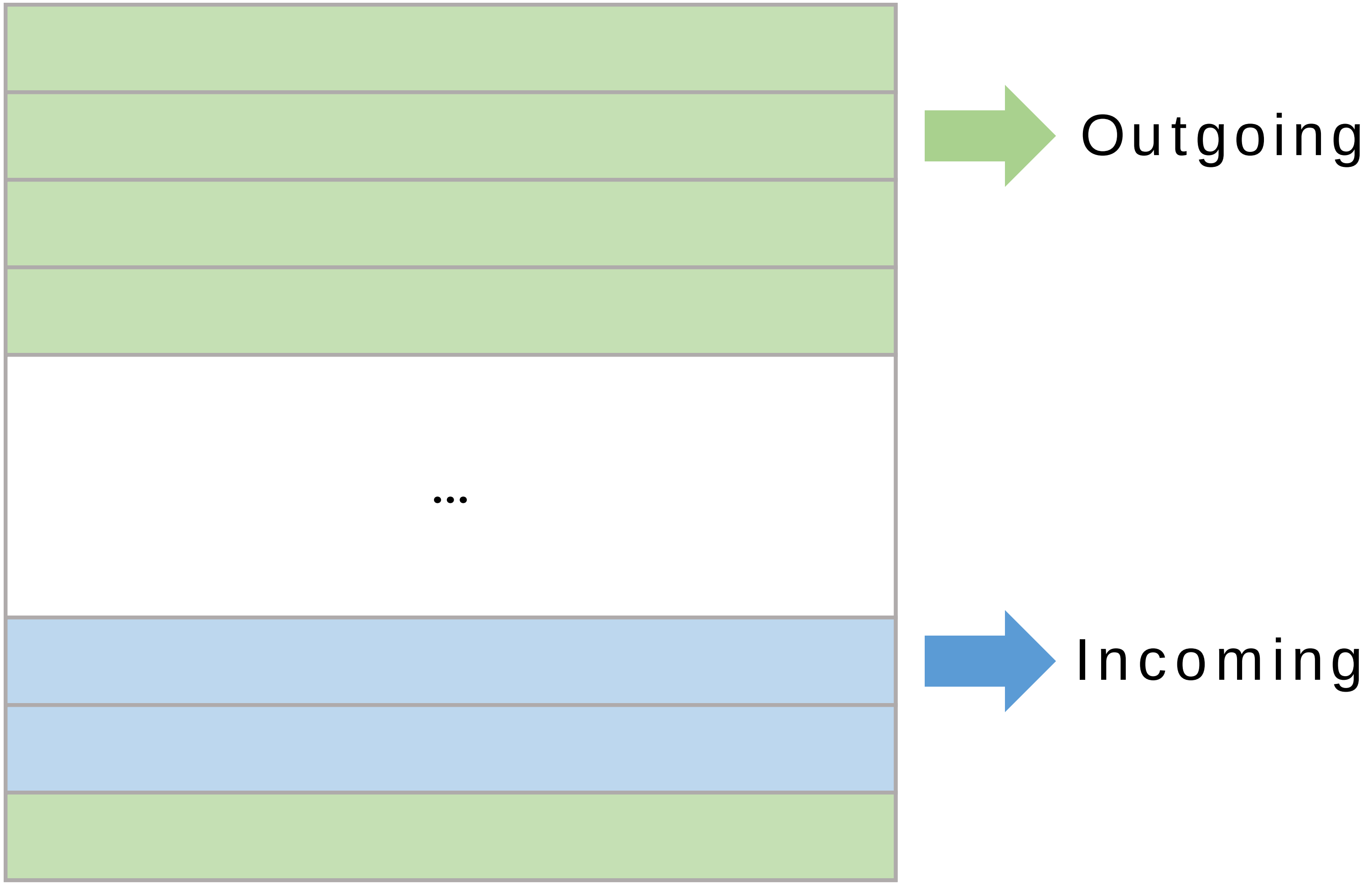
1. **Craft Attacker node IDs**
   - Make a lot of IDs using 1 or 2 IPs
   - Use rejection sampling

2. **Insert Attacker node IDs into db**
   - Send ping msg to the victim
   - Every 24 hours
   - Response to ping, findnode

3. **Reboot and eclipse the victim**
   - Do seeding to fill in Table
   - Seeding use info from db

Do Monopolizing agian

TCP Connection List (maxpeers)



THANK YOU