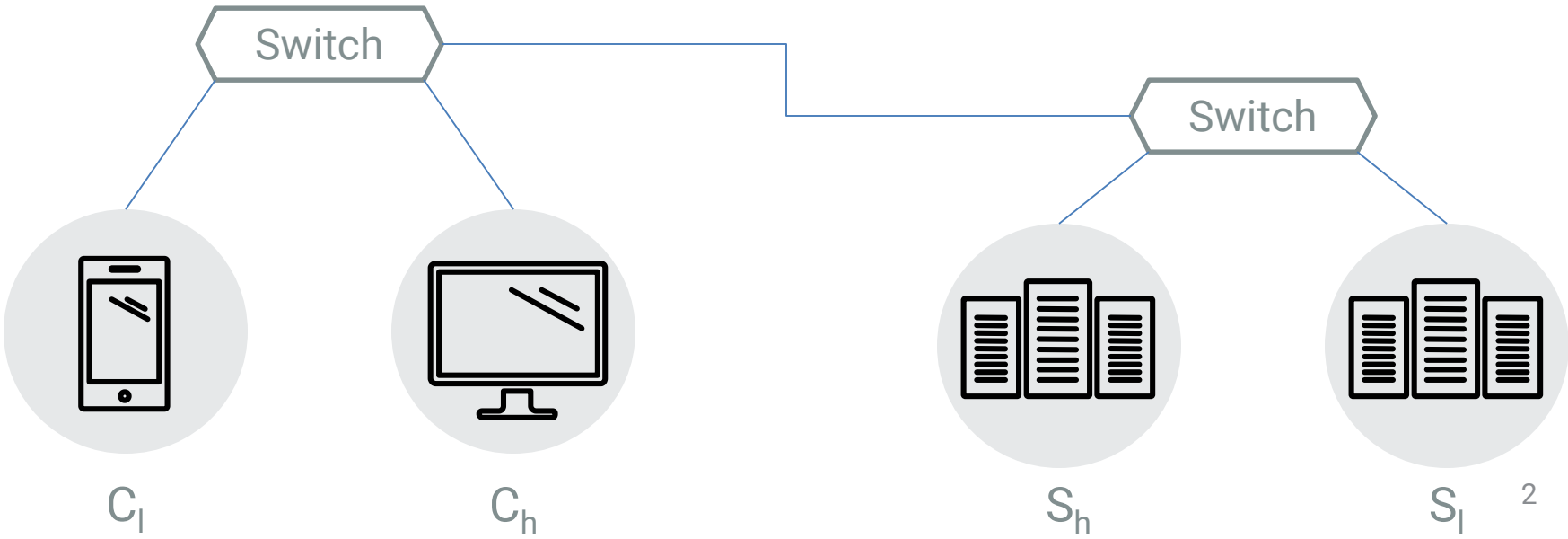
Some Complexity Results for Stateful Network Verification

- Yaron Velner
- **Kalev Alpernas**
- Alexander Rabinovich
- Mooly Sagiv
- Sharon Shoham

- Aurojit Panda
- Scott Shenker

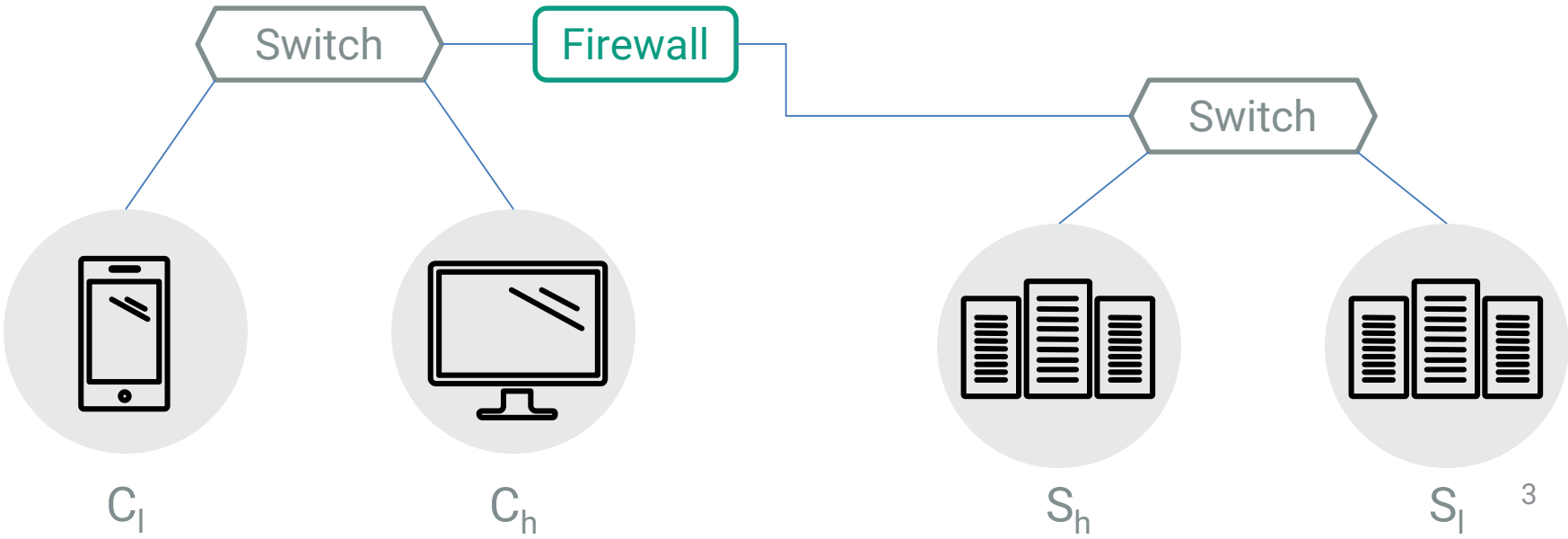
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation



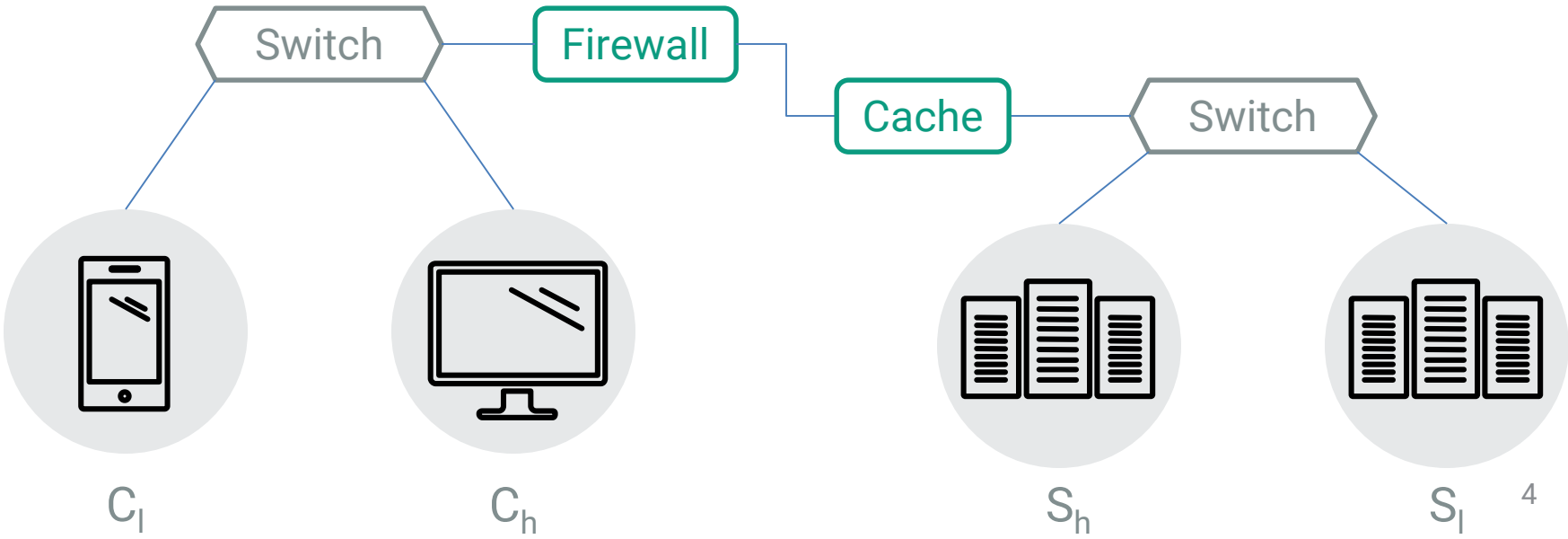
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



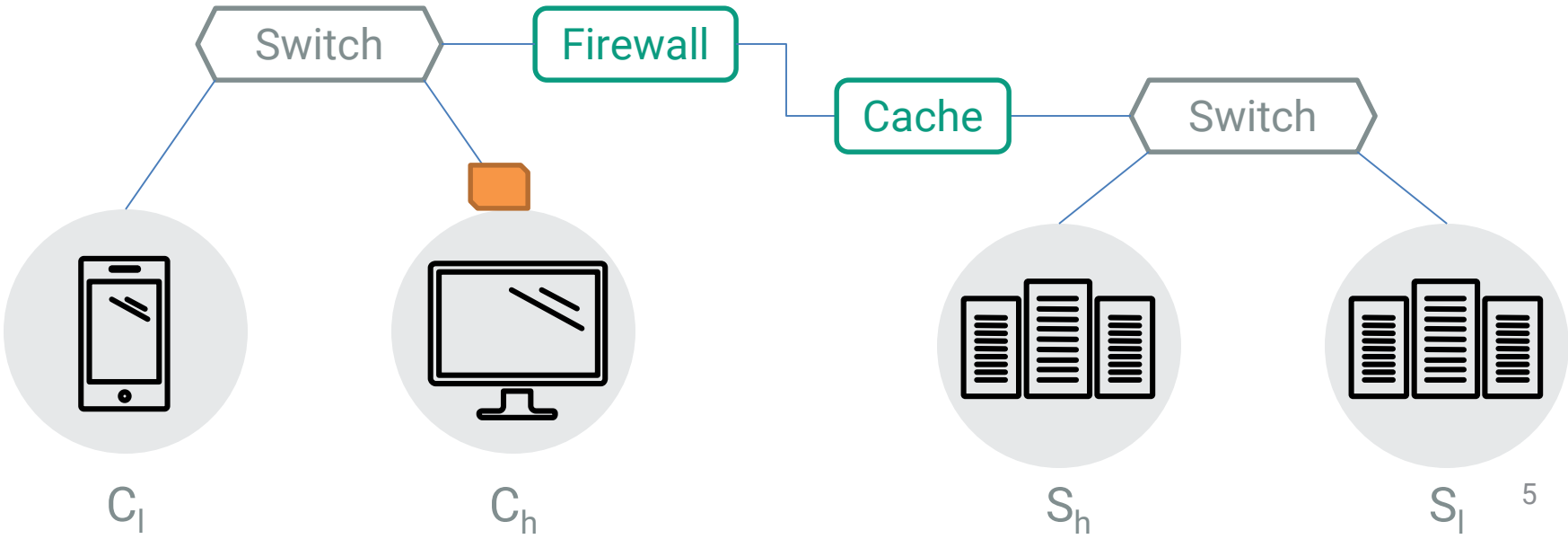
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



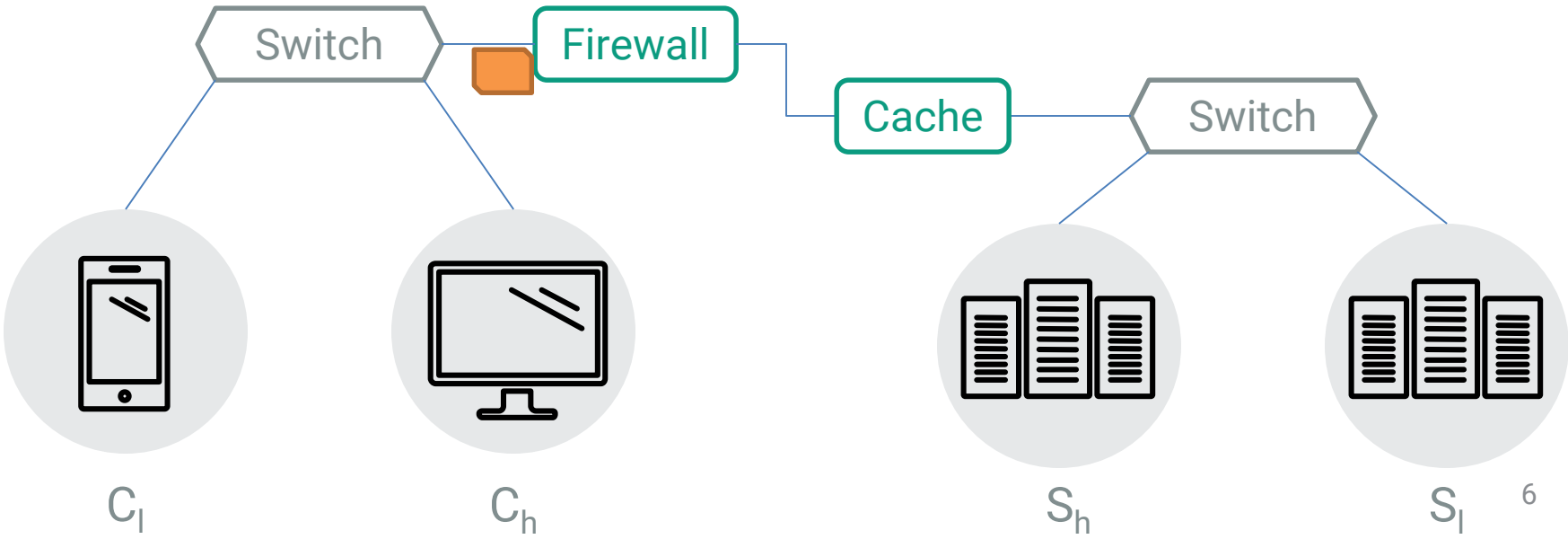
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



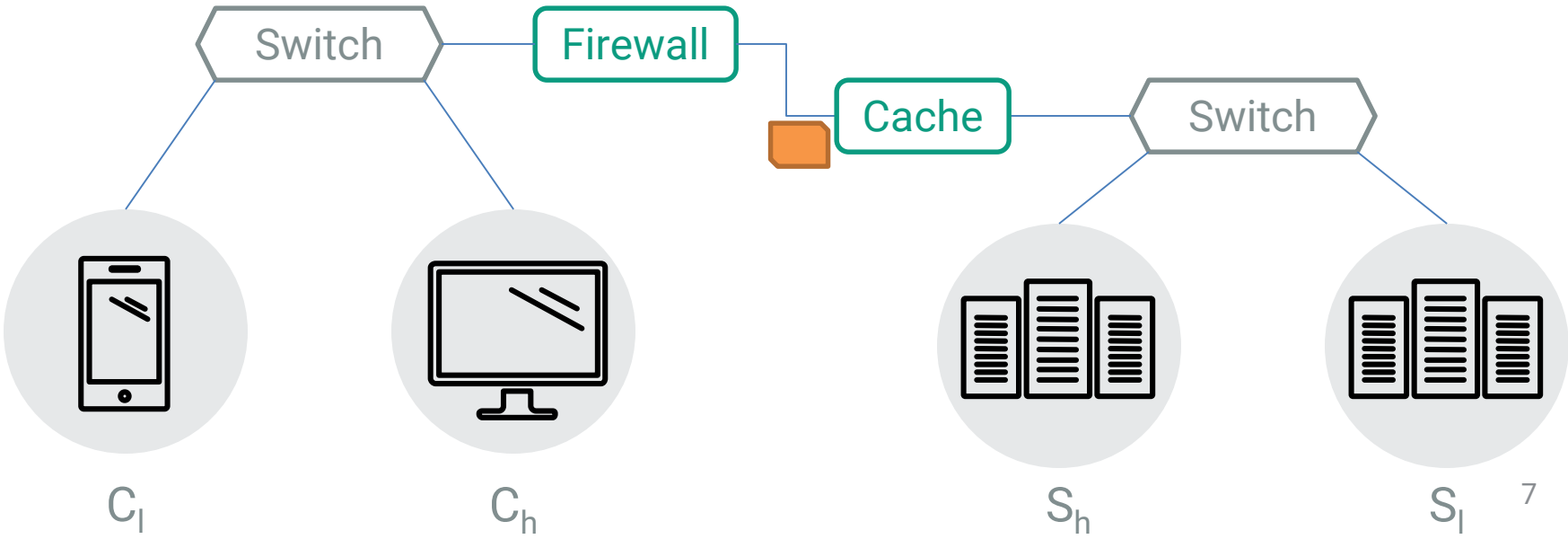
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



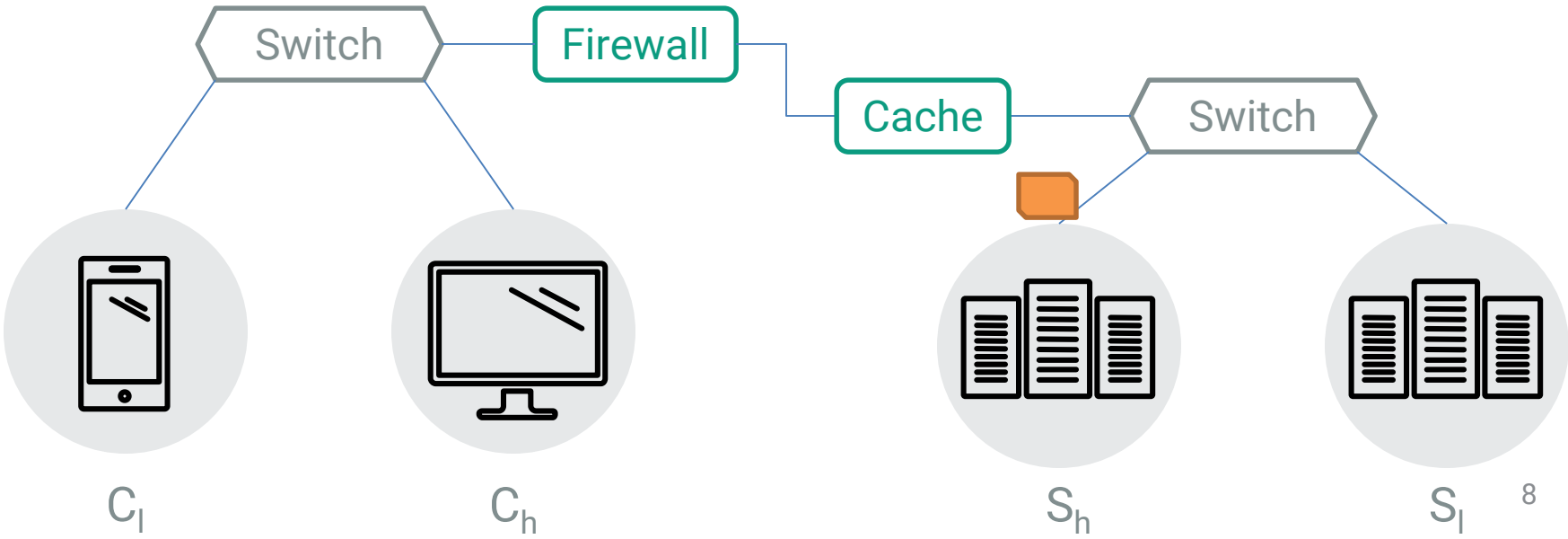
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



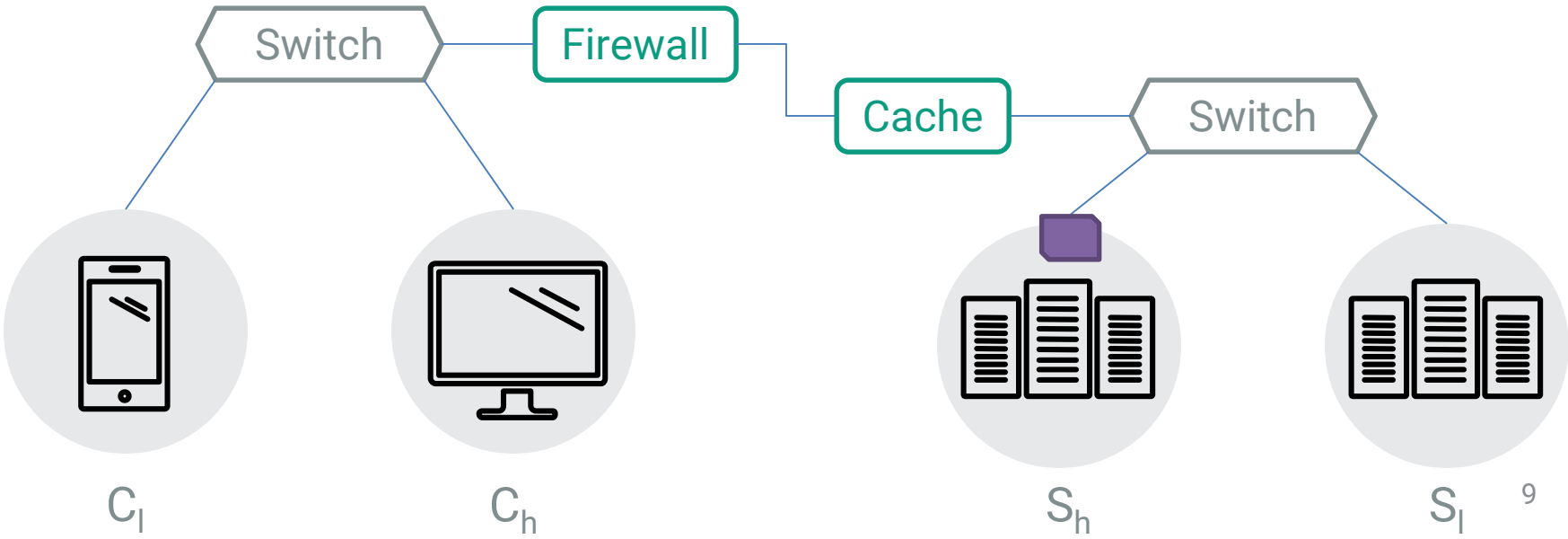
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



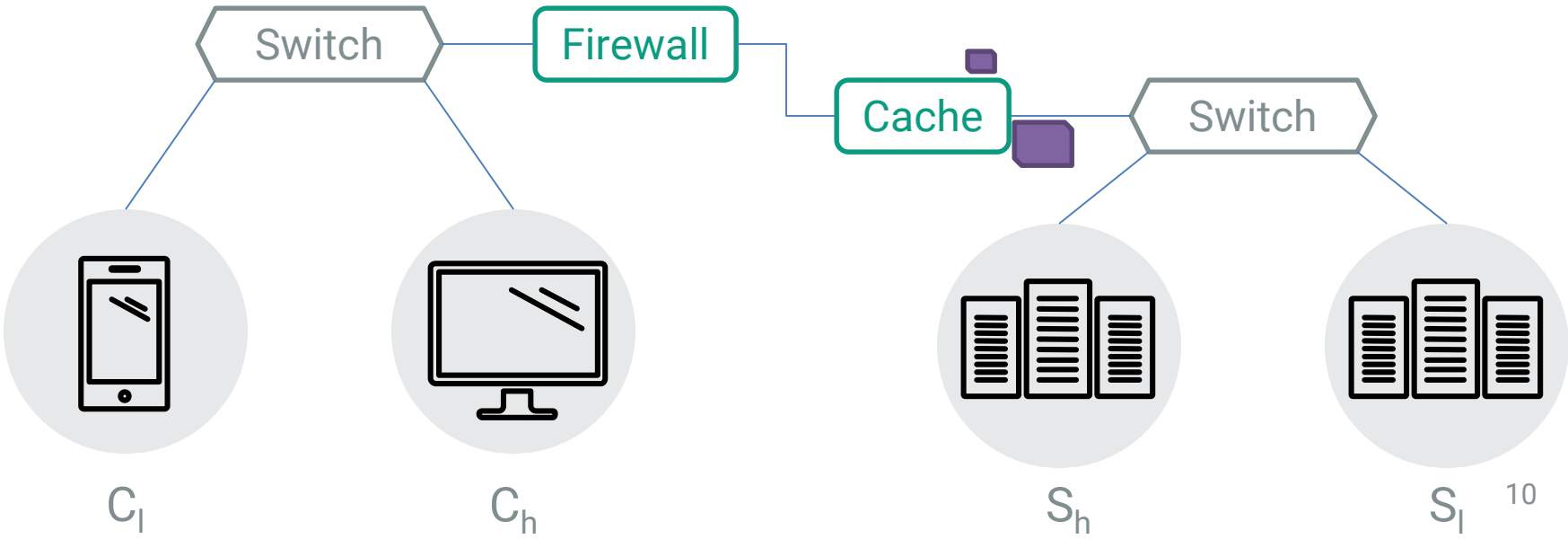
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



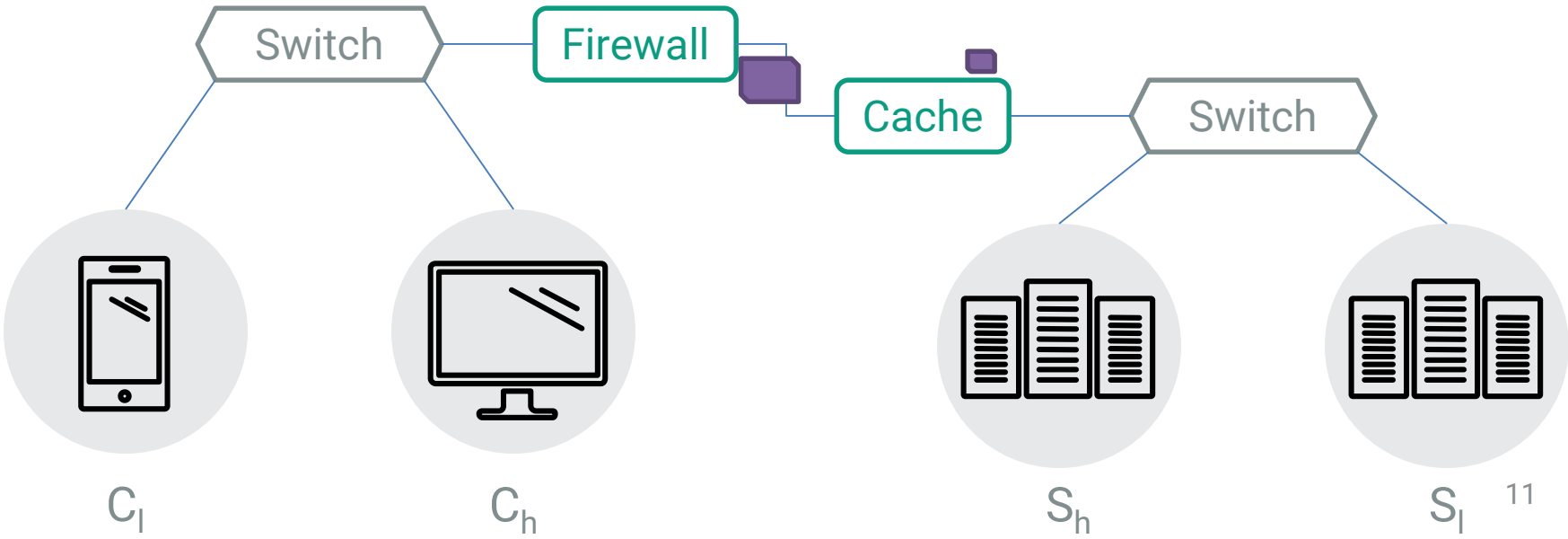
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



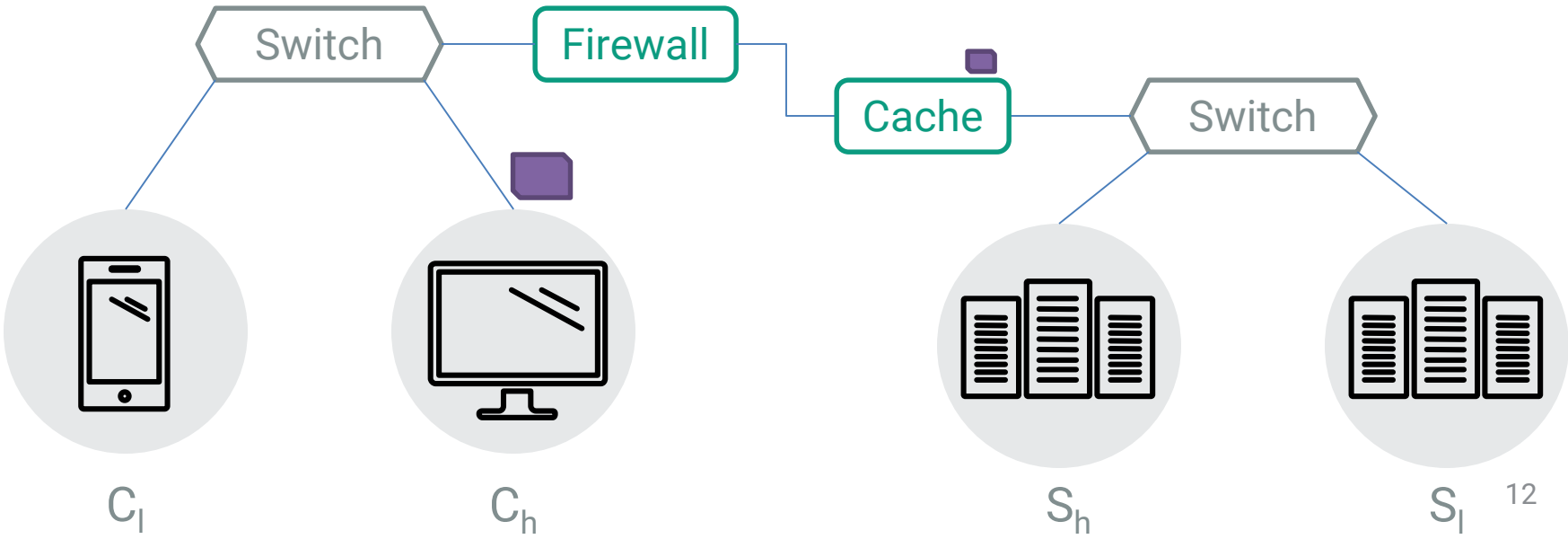
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



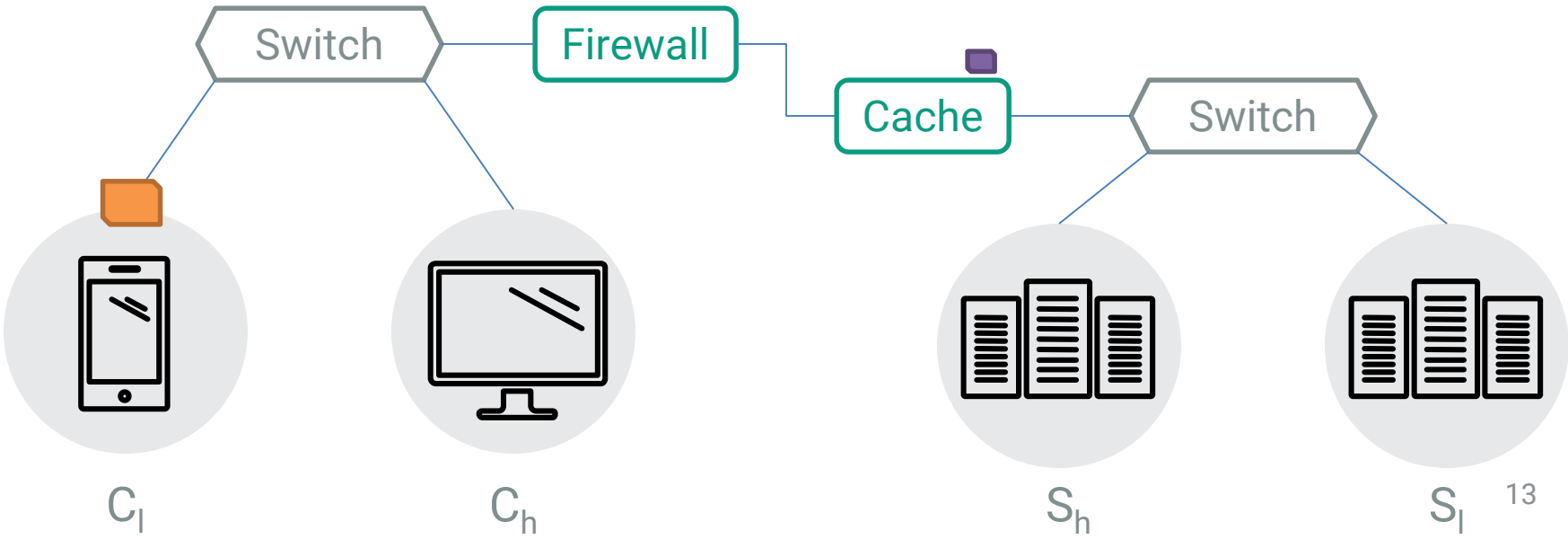
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



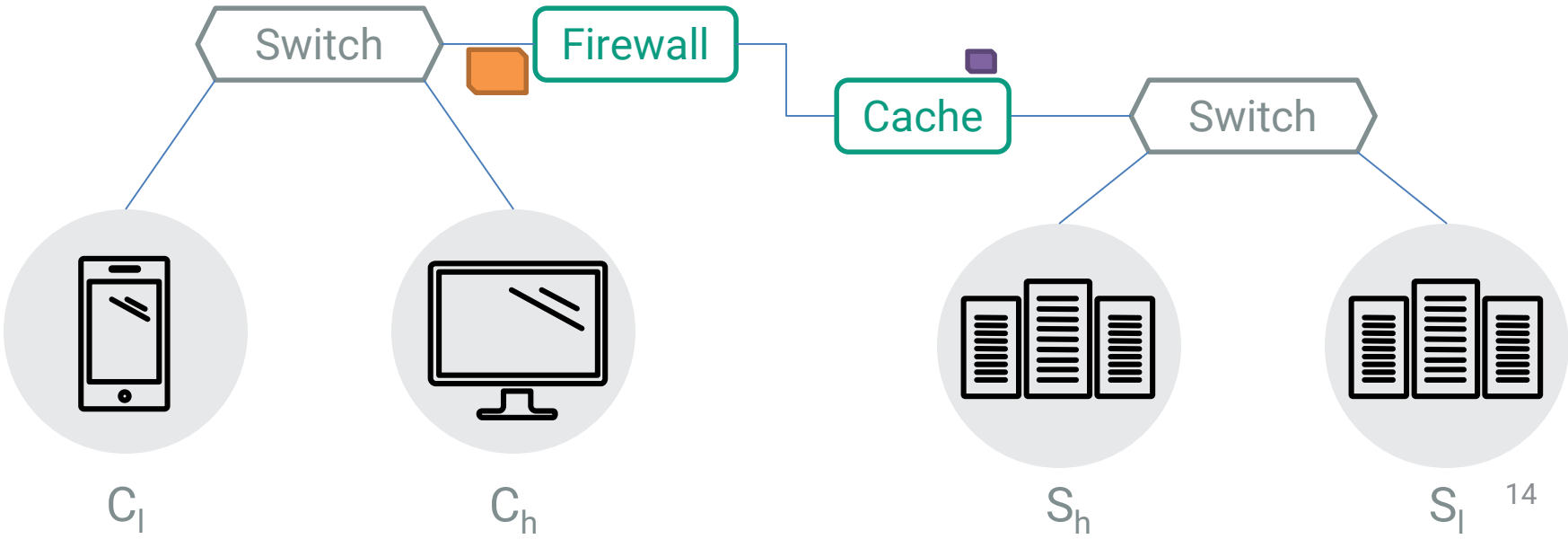
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



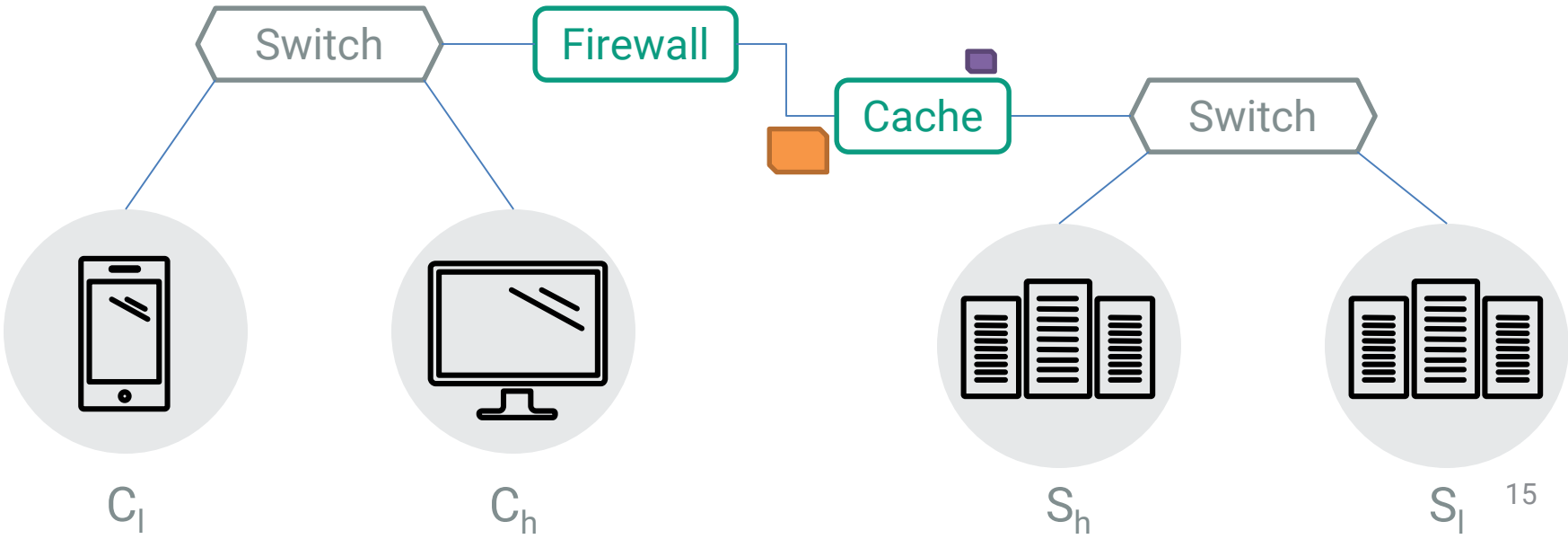
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



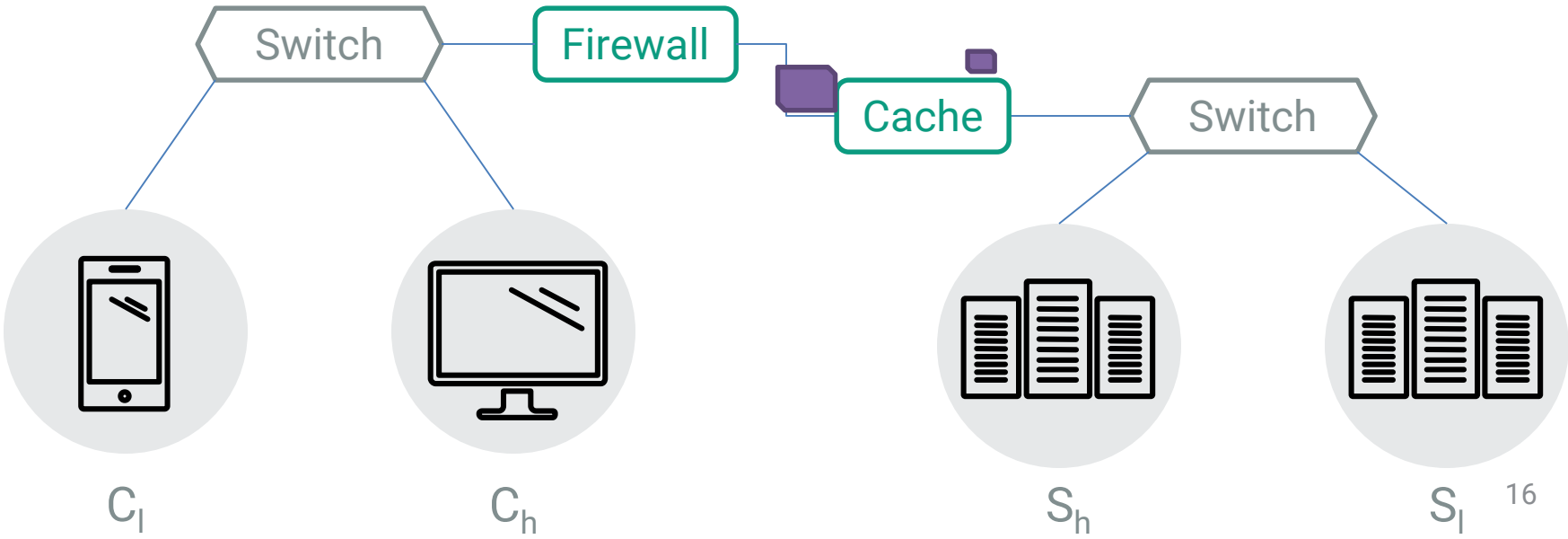
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



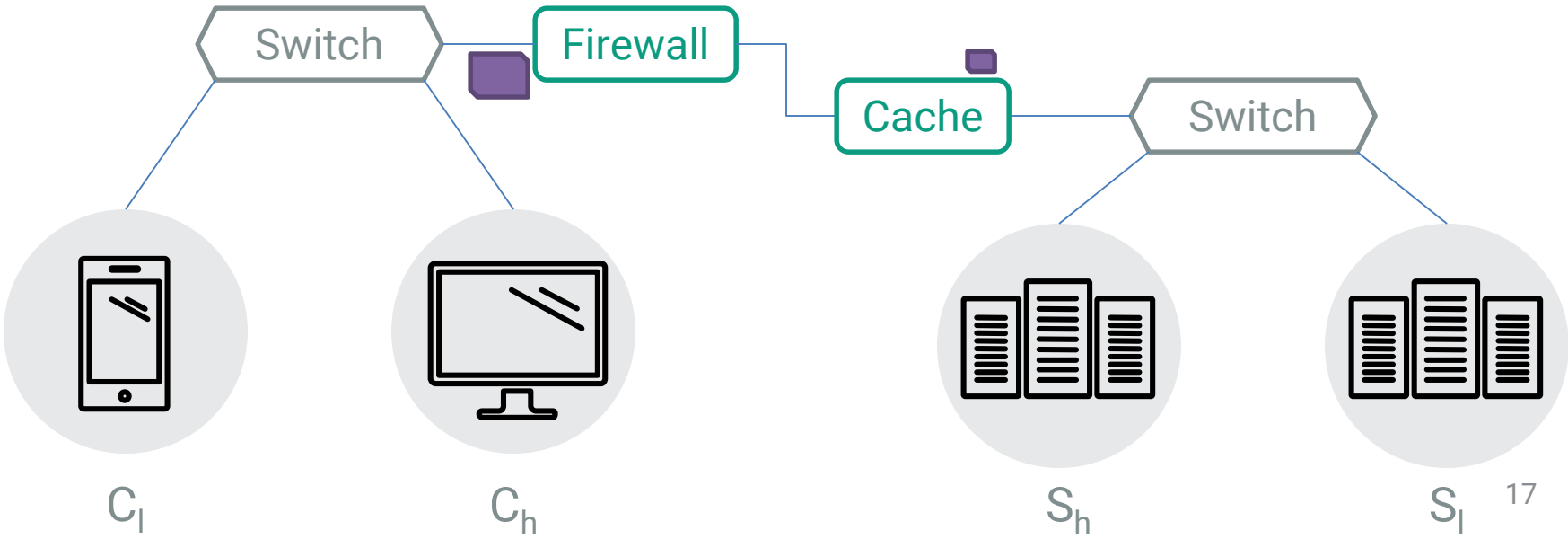
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



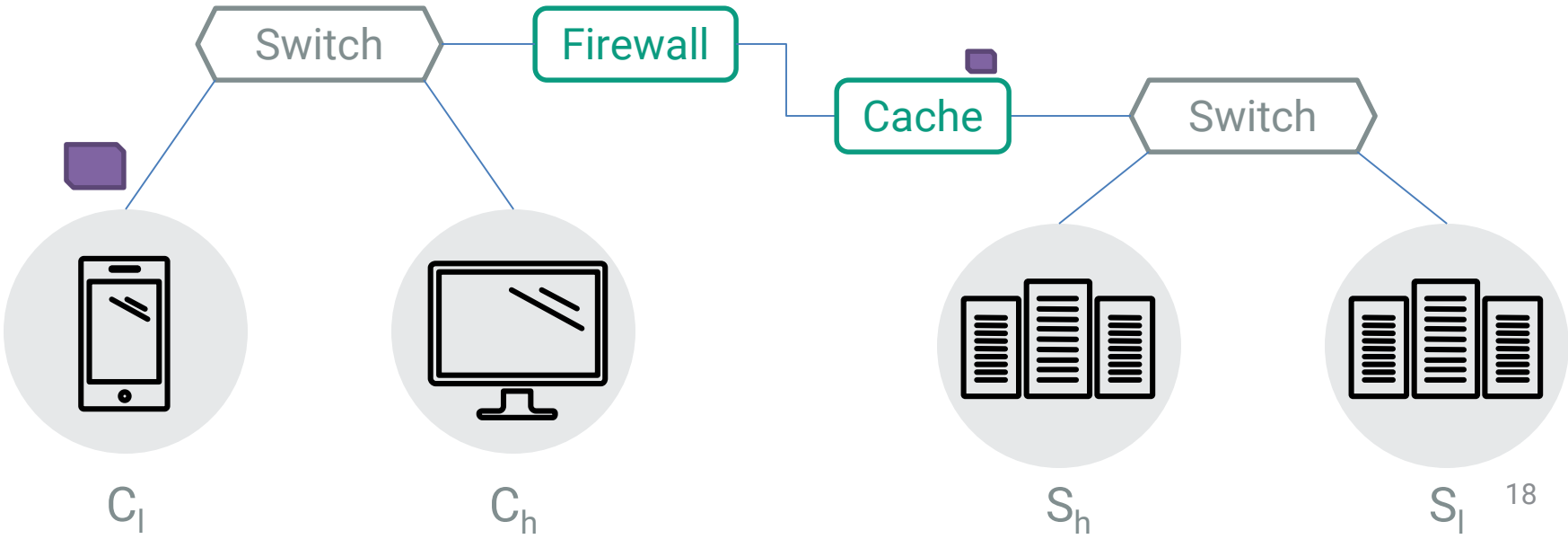
Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



Problem Statement

- Given a network topology
- Task: Verify the safety of the network
  - Isolation
  - In the presence of Middleboxes



Examples of Middlebox

- Network address translators (NAT)
- Firewall
- Traffic shapers
- Intrusion detection systems (IDSs)
- Transparent web proxy caches
- Application accelerators

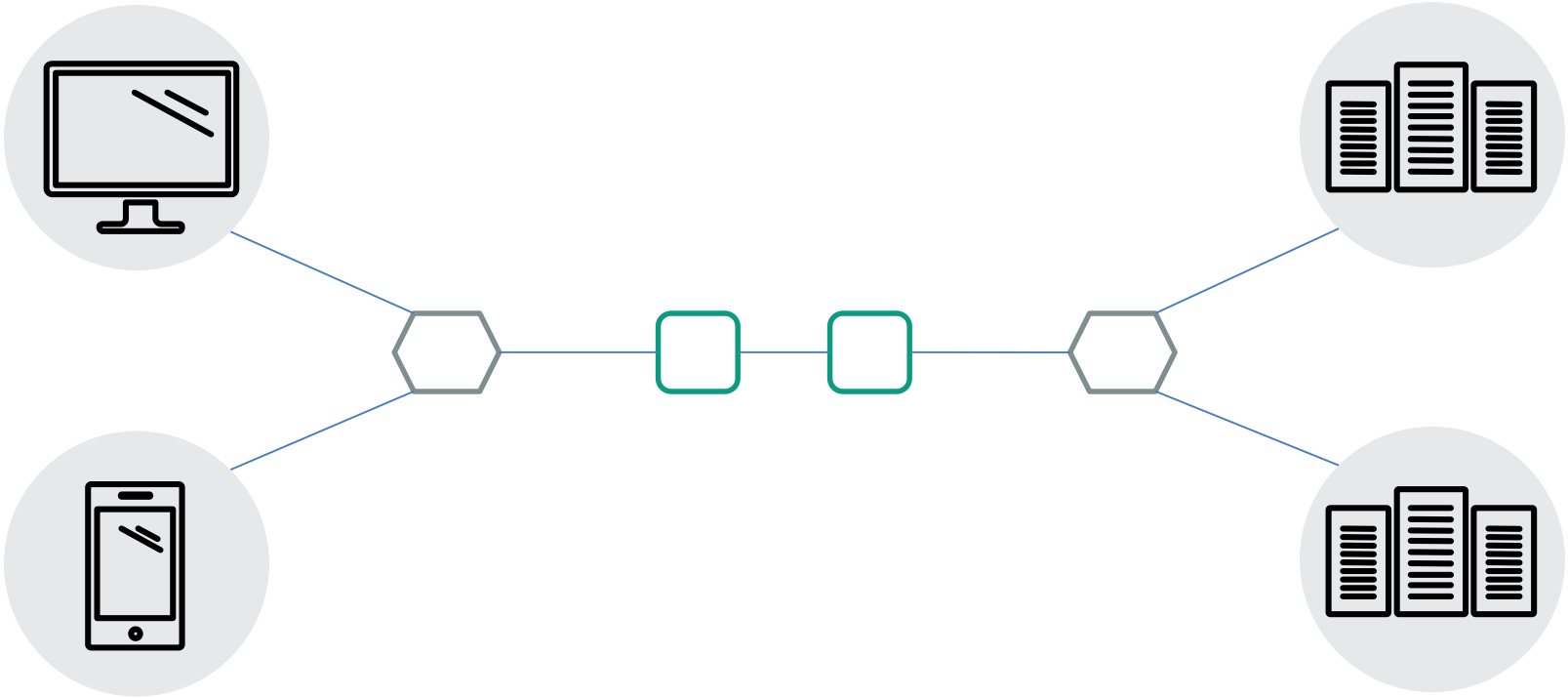
Middlebox ≈ FSM

- For the purpose of proving safety, the behaviour of the middlebox is essentially a finite state machine
  - Reason about middlebox behaviour, not implementation
- Network ≈ Communicating FSMs

Main Results

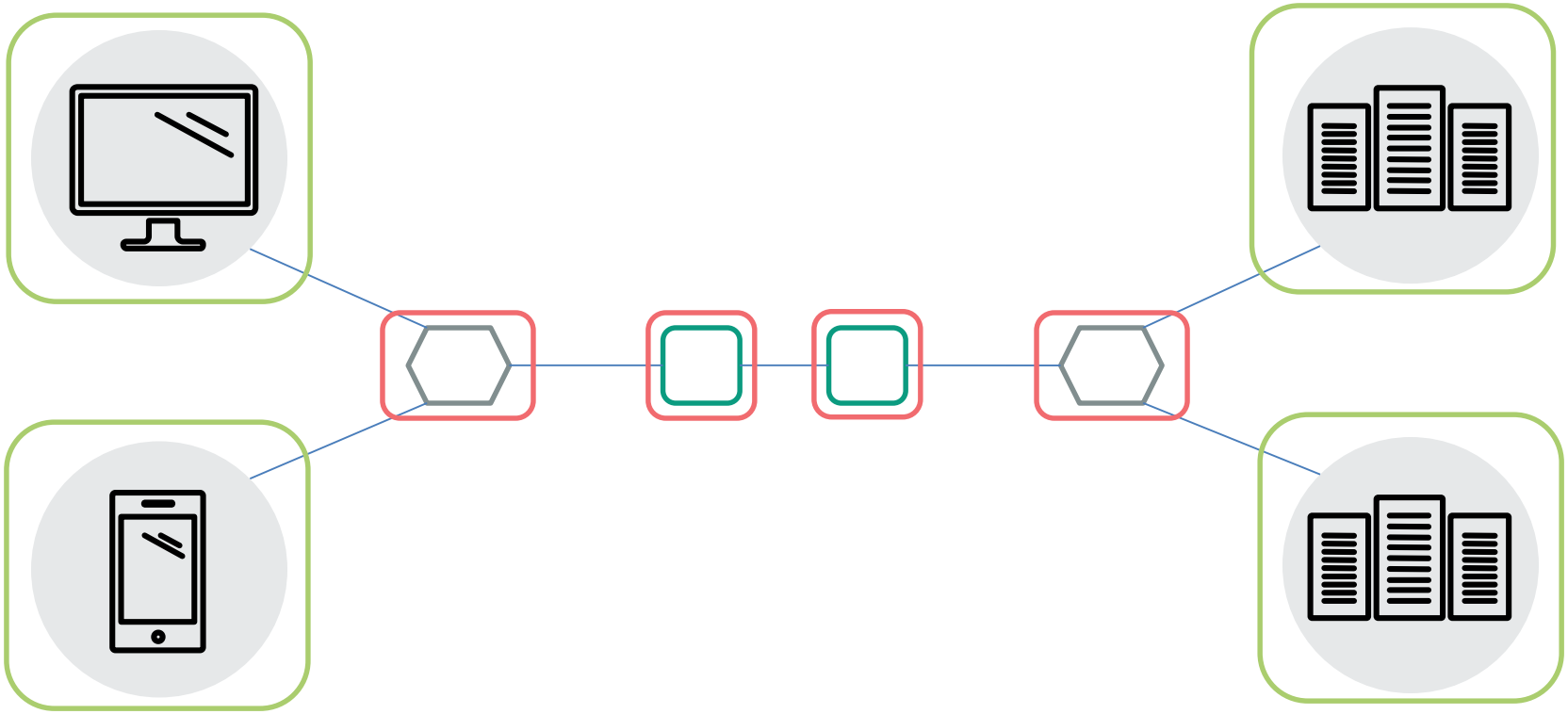
- Classify middleboxes according to the forwarding behaviour
  - Input/Output relation
  - Depends on state (history)
- Tight complexity Results for the different classes
- Compact symbolic representation preserves complexity results
  - Exponential saving in common cases

Network Model



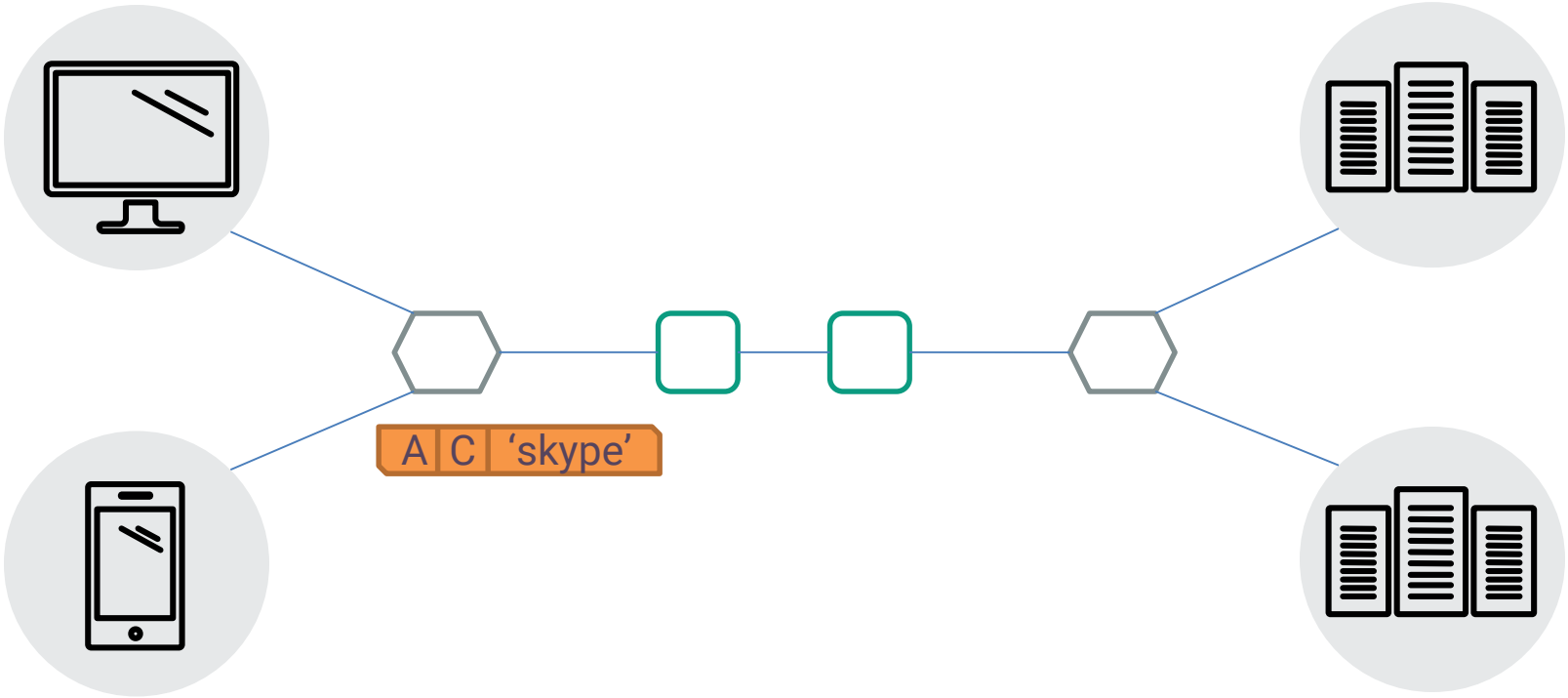
- An undirected graph

Network Model



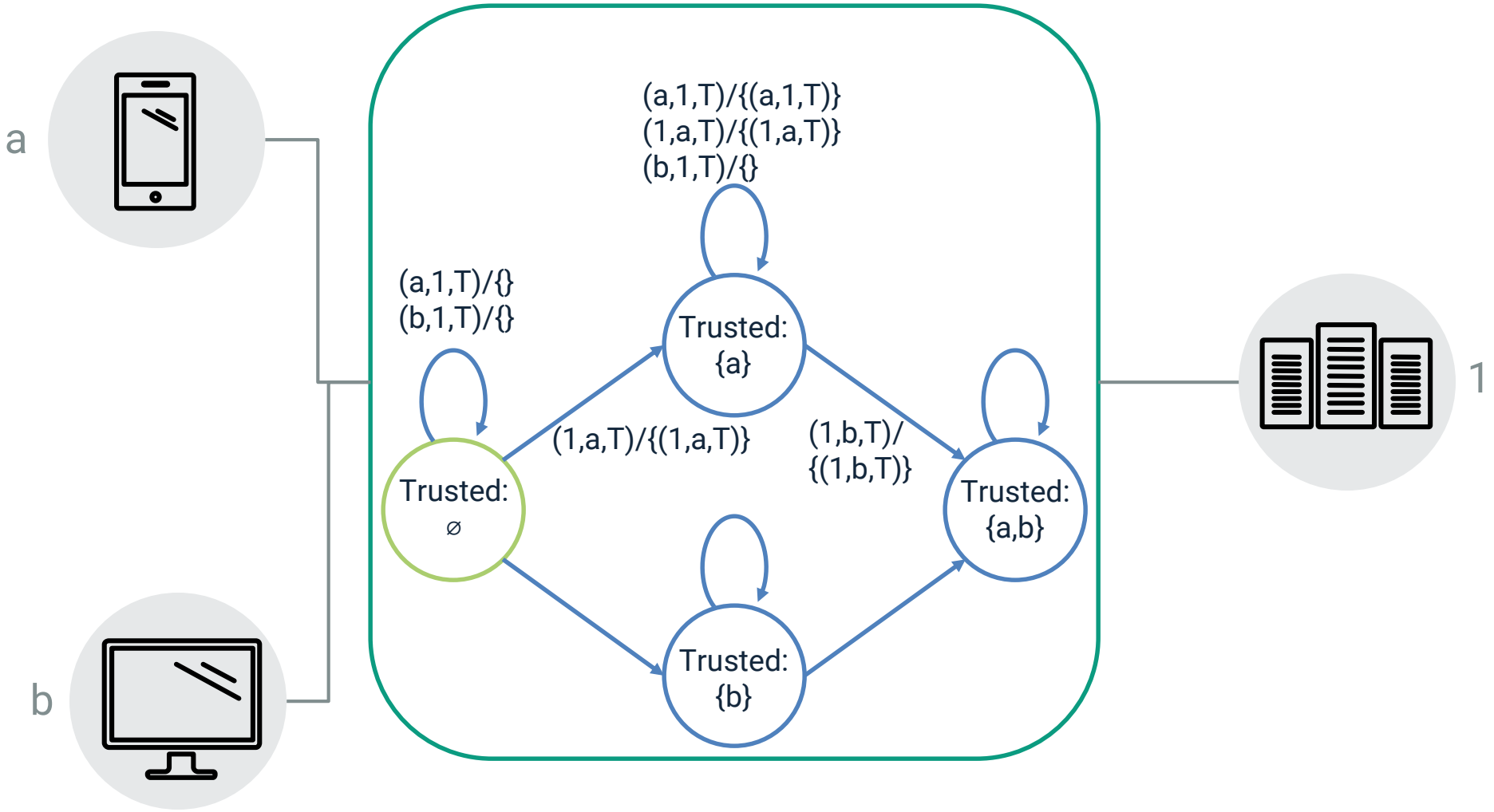
- Vertices are hosts and Middleboxes

Network Model



- Packets are <Source, Destination, Tag> tuples

Transducer Middlebox – Hole Punching Firewall



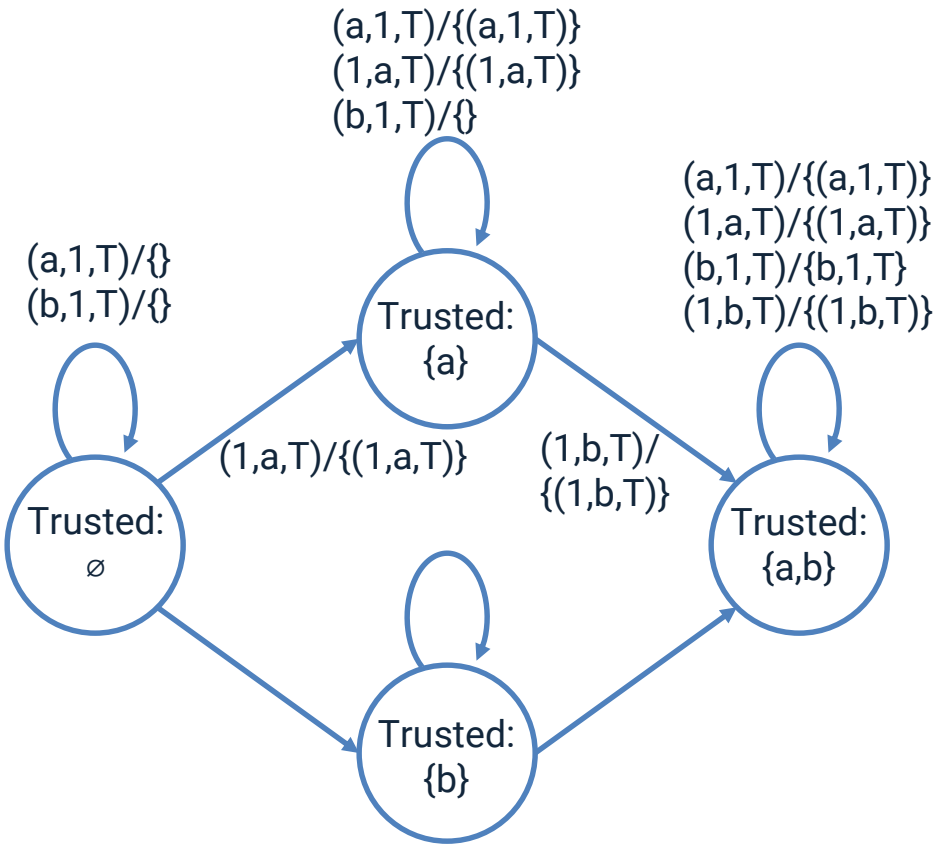
Symbolic Representation Hole Punching Firewall

```
input(src, dst, tag, prt):
  prt = INTERN ⇒
    // hosts within organization
    trusted.insert dst;
    output {(src, dst, tag, EXTERN)}
  prt = EXTERN ∧ src in trusted ⇒
    // trusted hosts outside organization
    output {(src, dst, tag, INTERN)}
  prt = EXTERN ∧ ¬(src in trusted) ⇒
    output ∅ // untrusted hosts
```

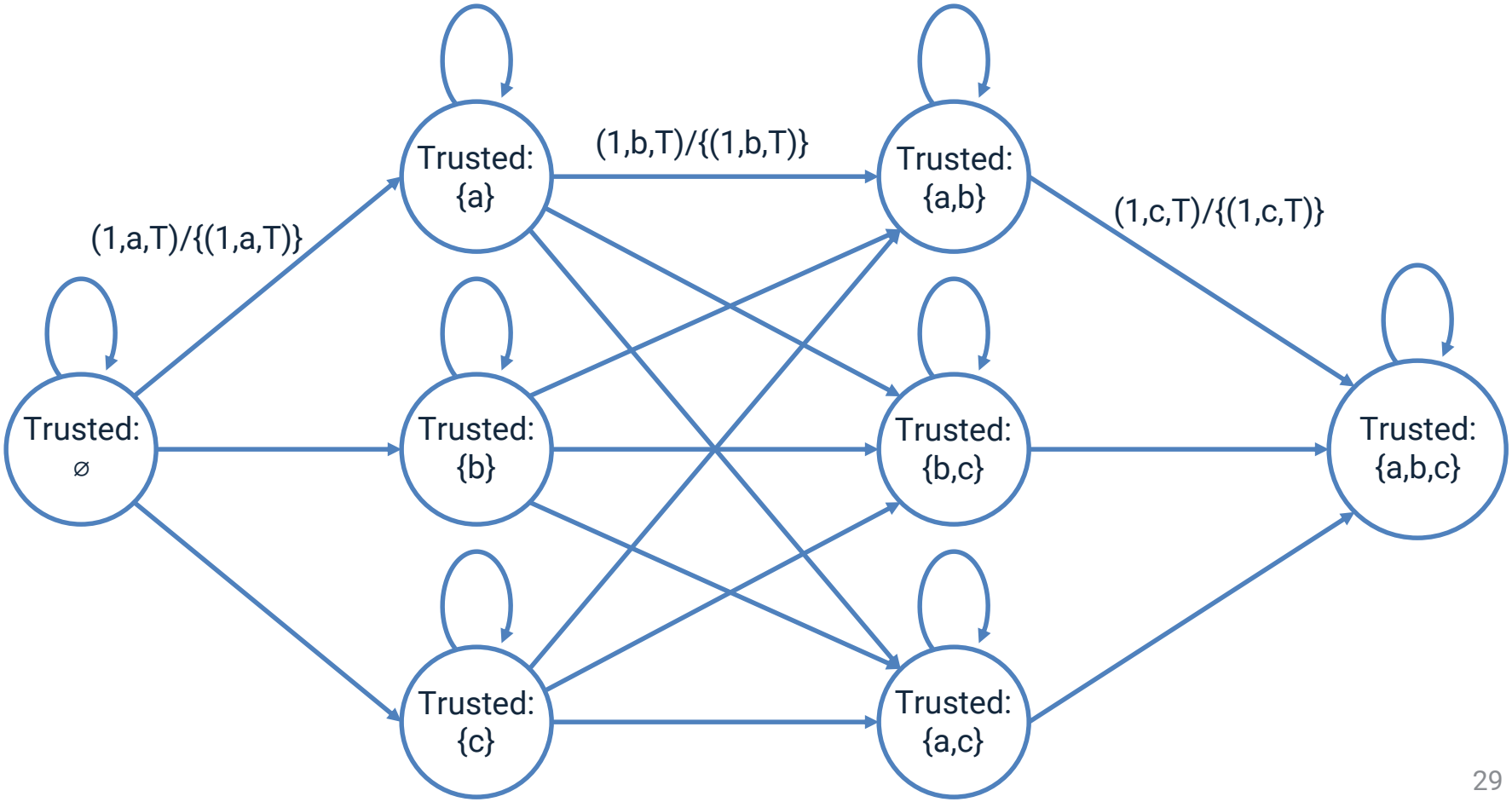
Symbolic Representation Hole Punching Firewall

```
input(src, dst, tag, prt):
  prt = INTERN ⇒
    // hosts within organization
    trusted.insert dst;
    output {(src, dst, tag, EXTERN)}
  prt = EXTERN ∧ src in trusted ⇒
    // trusted hosts outside organization
    output {(src, dst, tag, INTERN)}
  prt = EXTERN ∧ ¬(src in trusted) ⇒
    output ∅ // untrusted hosts
```

Explicit Representation – Hole Punching Firewall (2 Hosts)



Explicit Representation – Hole Punching Firewall (3 Hosts)

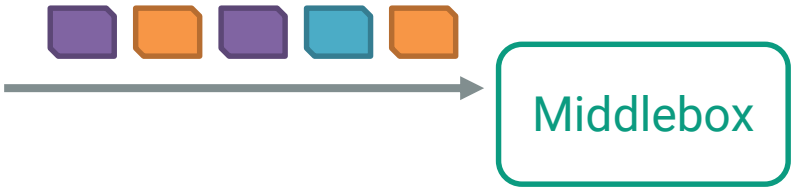


Property Middleboxes - Isolation

```
input(src, dst, tag, prt):
  prt = HOST ⇒
    // host for which isolation is checked
    output {(src, dst, tag, NET)}
  prt = NET ∧ ¬(src in forbidden) ⇒
    // no isolation violation
    output {(src, dst, tag, HOST)}
  prt = NET ∧ src in forbidden⇒
    // isolation violation
    abort
```

Network Semantics

- FIFO

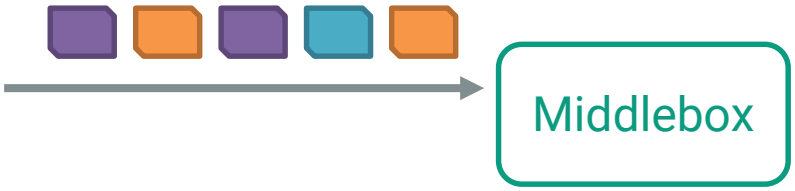


  - Network verification is undecidable
    - Can simulate a Turing Machine
    - Even without forwarding loops

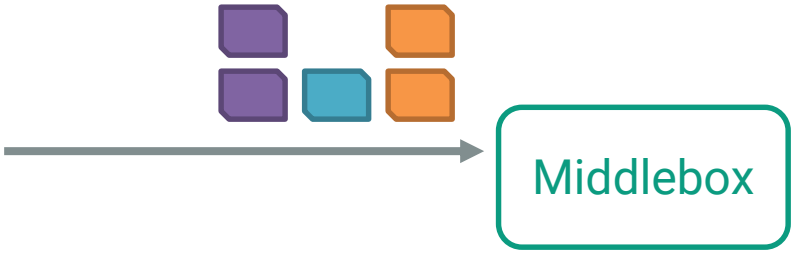
[Daniel Brand, and Pitro Zafiropulo. *JACM* '83]

Network Semantics

- FIFO



- Unordered



  - Network verification is decidable

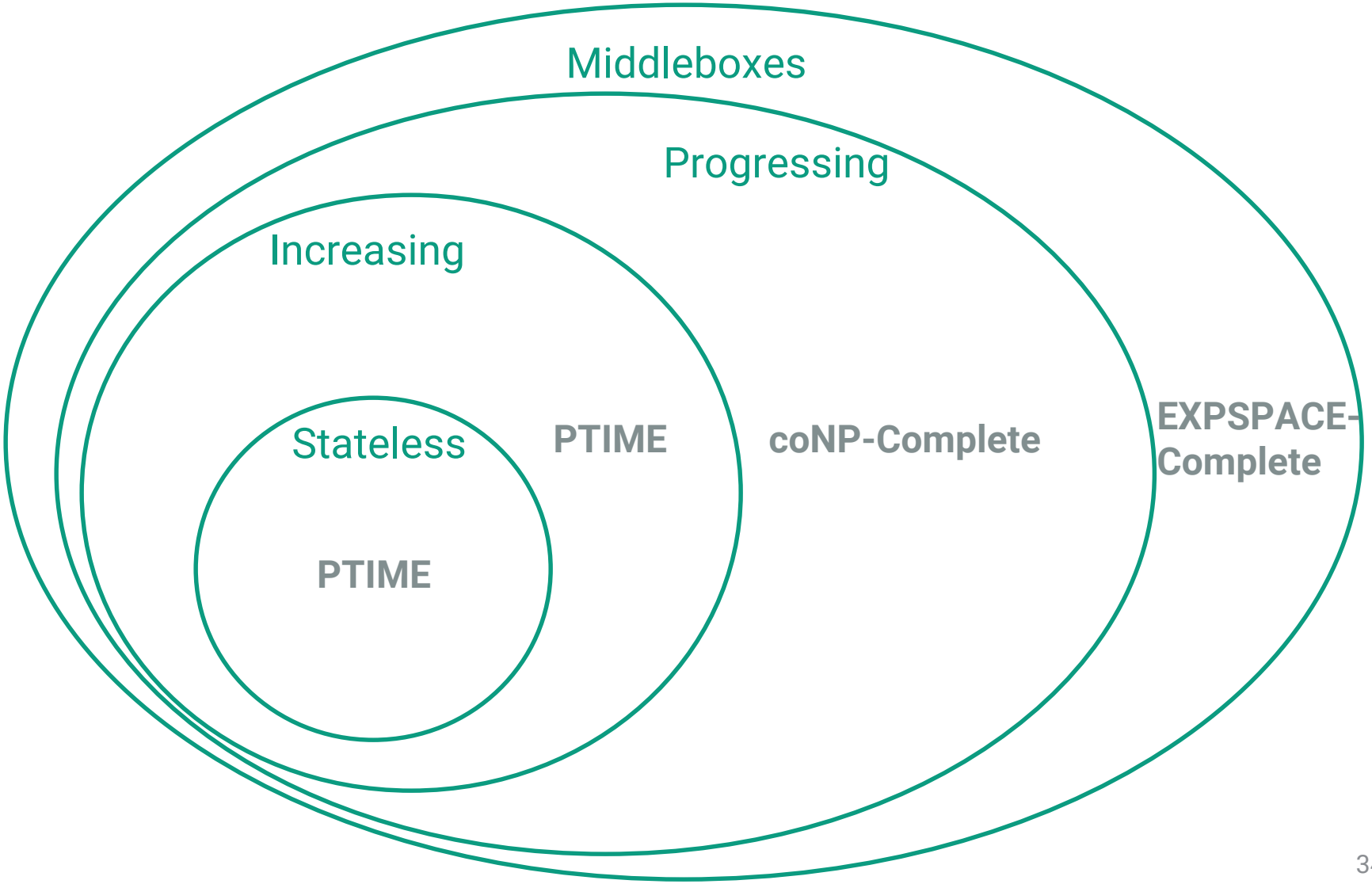
[Abdulla et al. *LICS* '93]

[Finkel, A., Schnoebelen, P. *Theoretical Computer Science* '01]

Verification Complexity

- **EXPSPACE-Complete**
  - Equivalent to Petri Net coverability
- Can we do better?
- In practice very few middlebox types are used
  - Explore ‘Good’ middleboxes

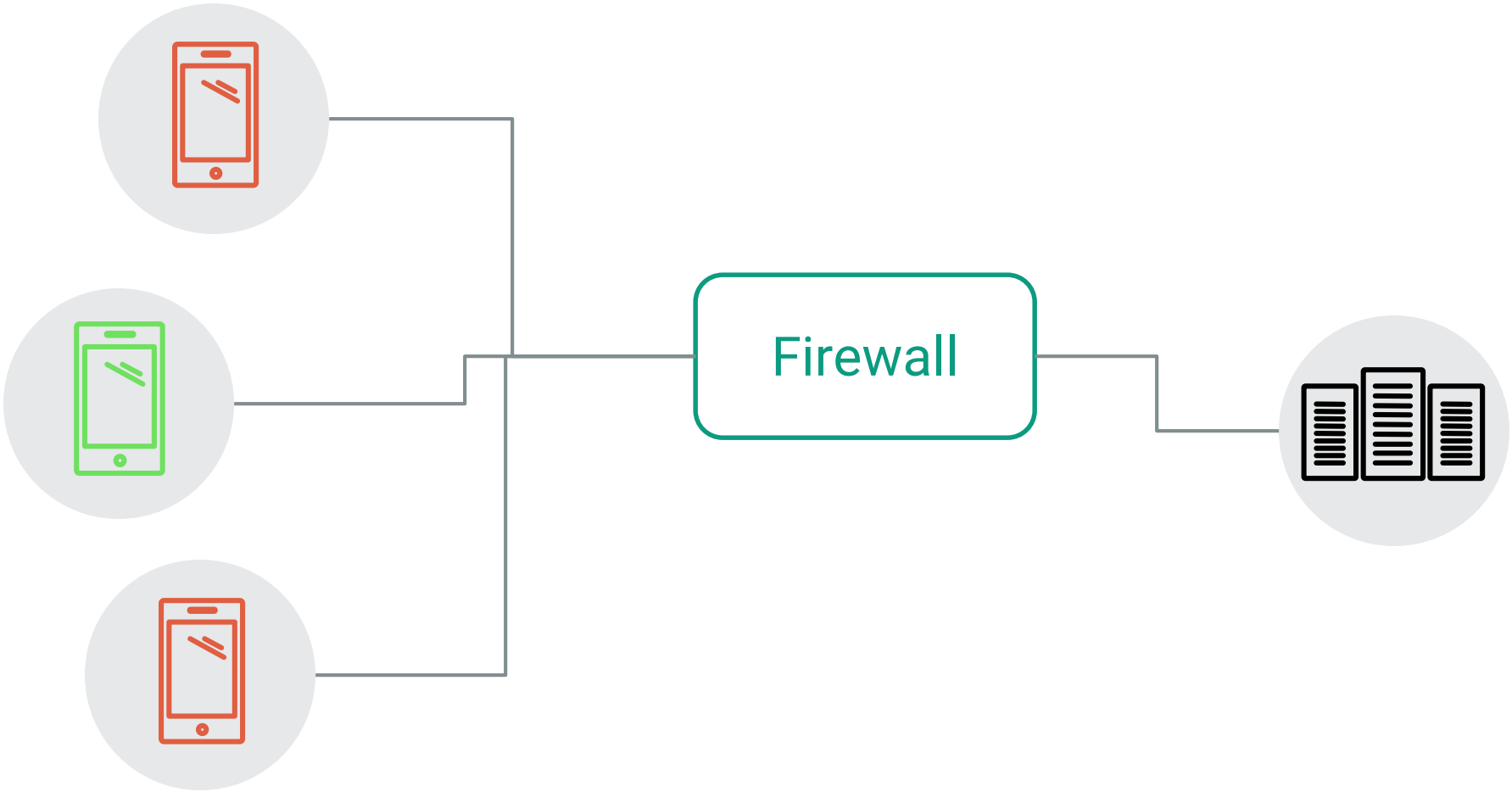
Middlebox Classification



Middlebox Classification

- According to forwarding behaviour
  - Effects of history on the forwarding behaviour
- Equivalently according to syntactic restrictions

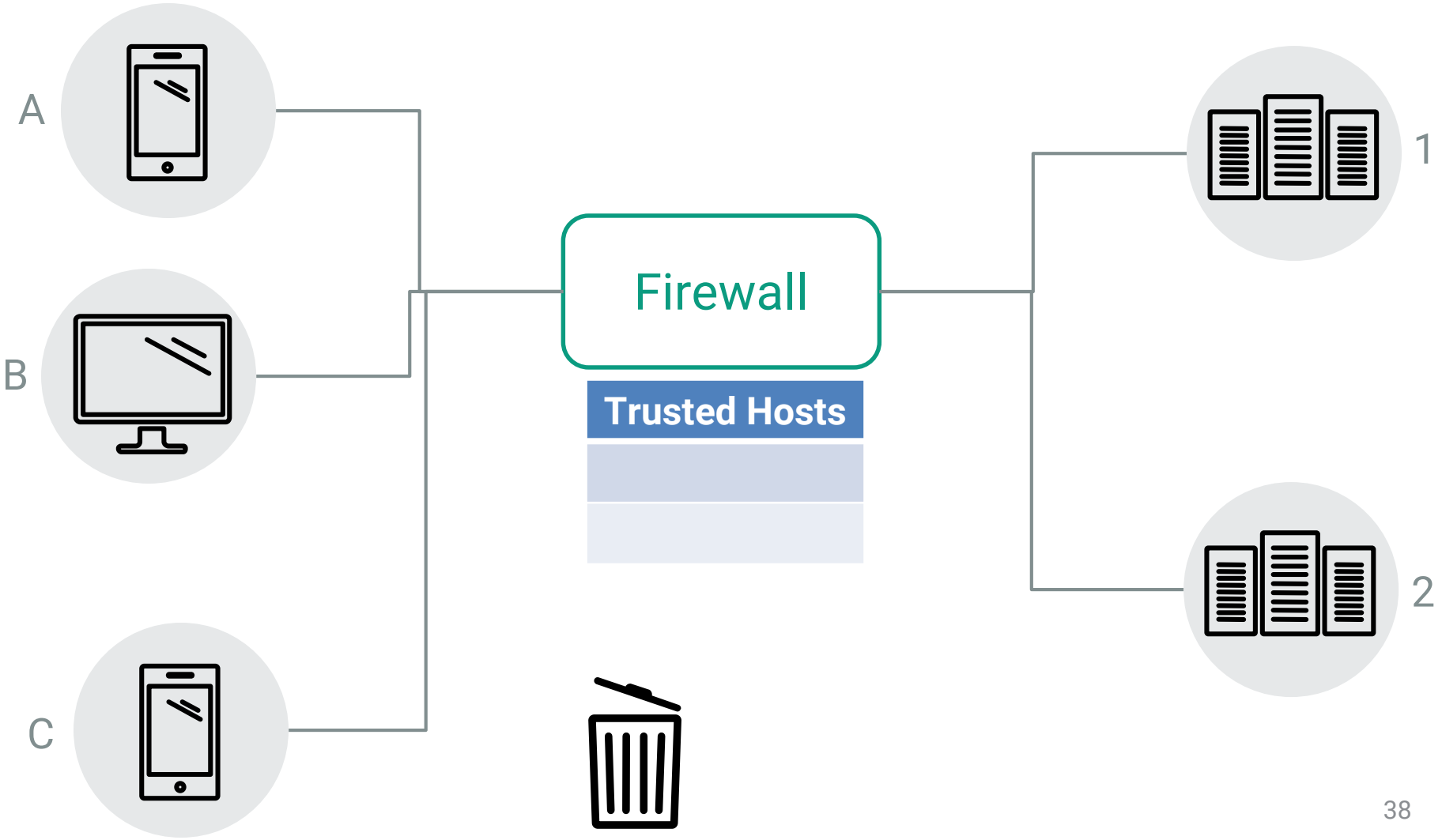
Stateless Middlebox – ACL Firewall



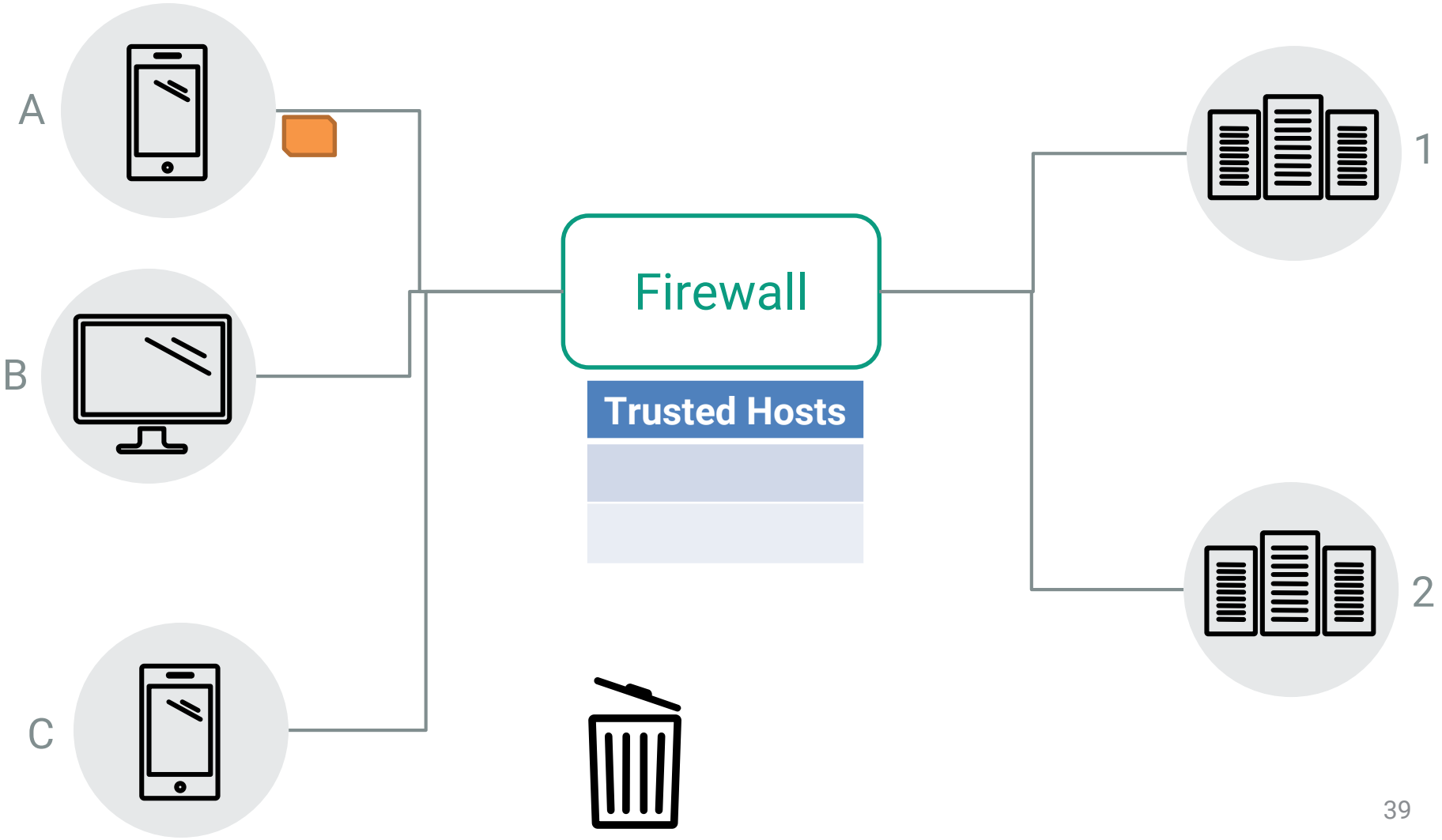
Stateless

- Transducer has only a single state
- Forwarding behaviour is history agnostic
- Syntactic restriction – no changes to the relations

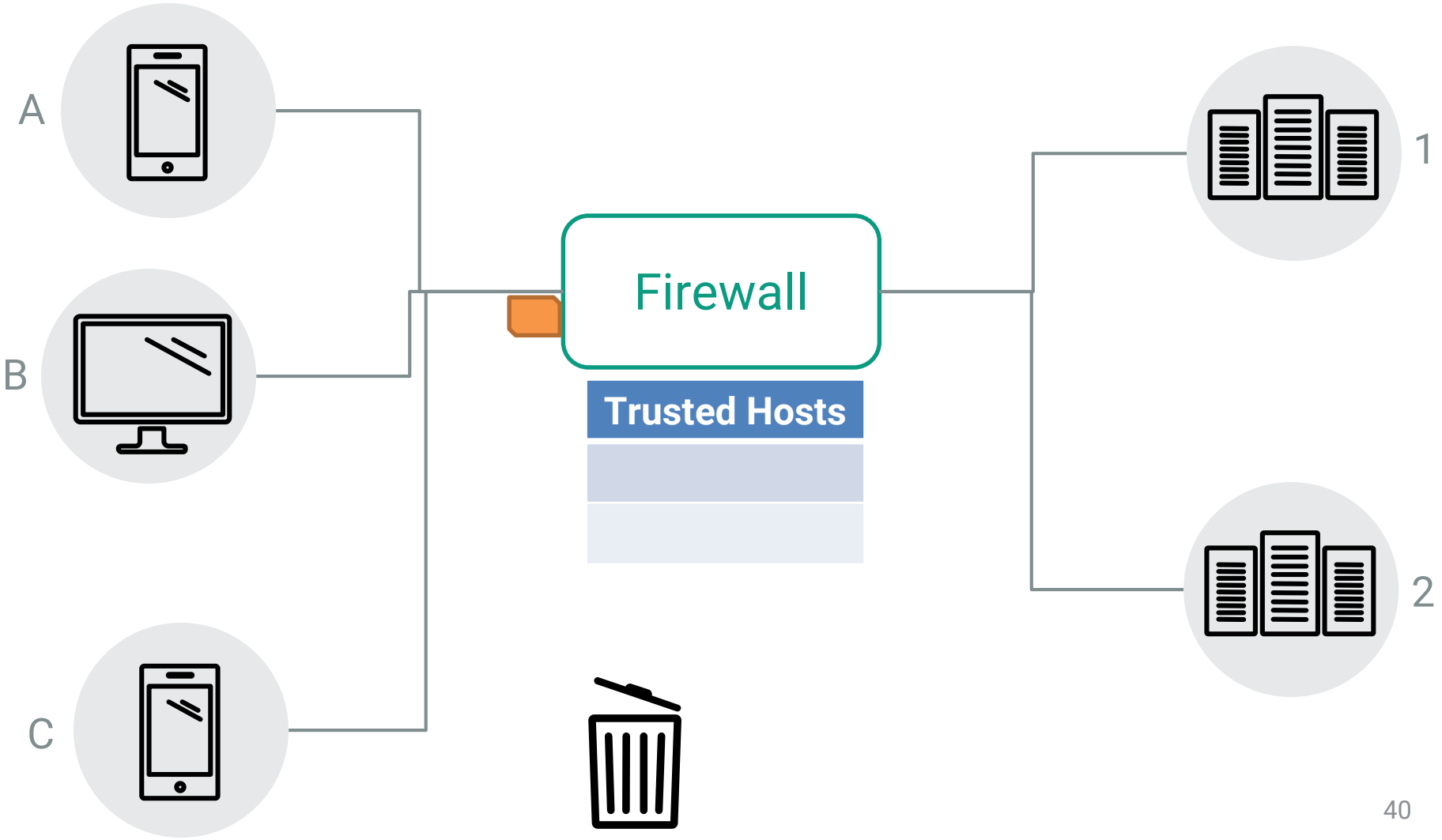
Increasing Middlebox – Hole Punching Firewall



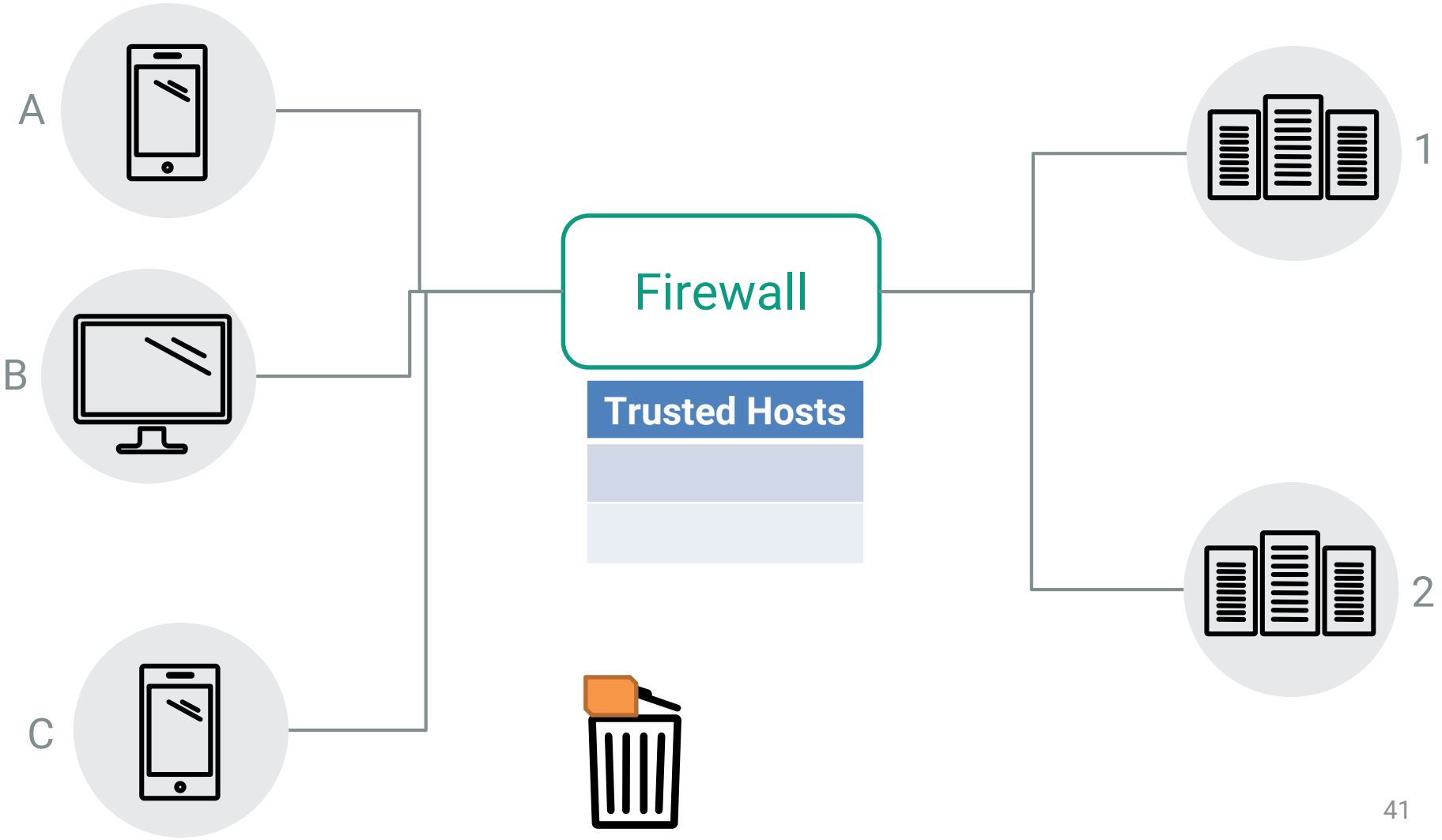
Increasing Middlebox – Hole Punching Firewall



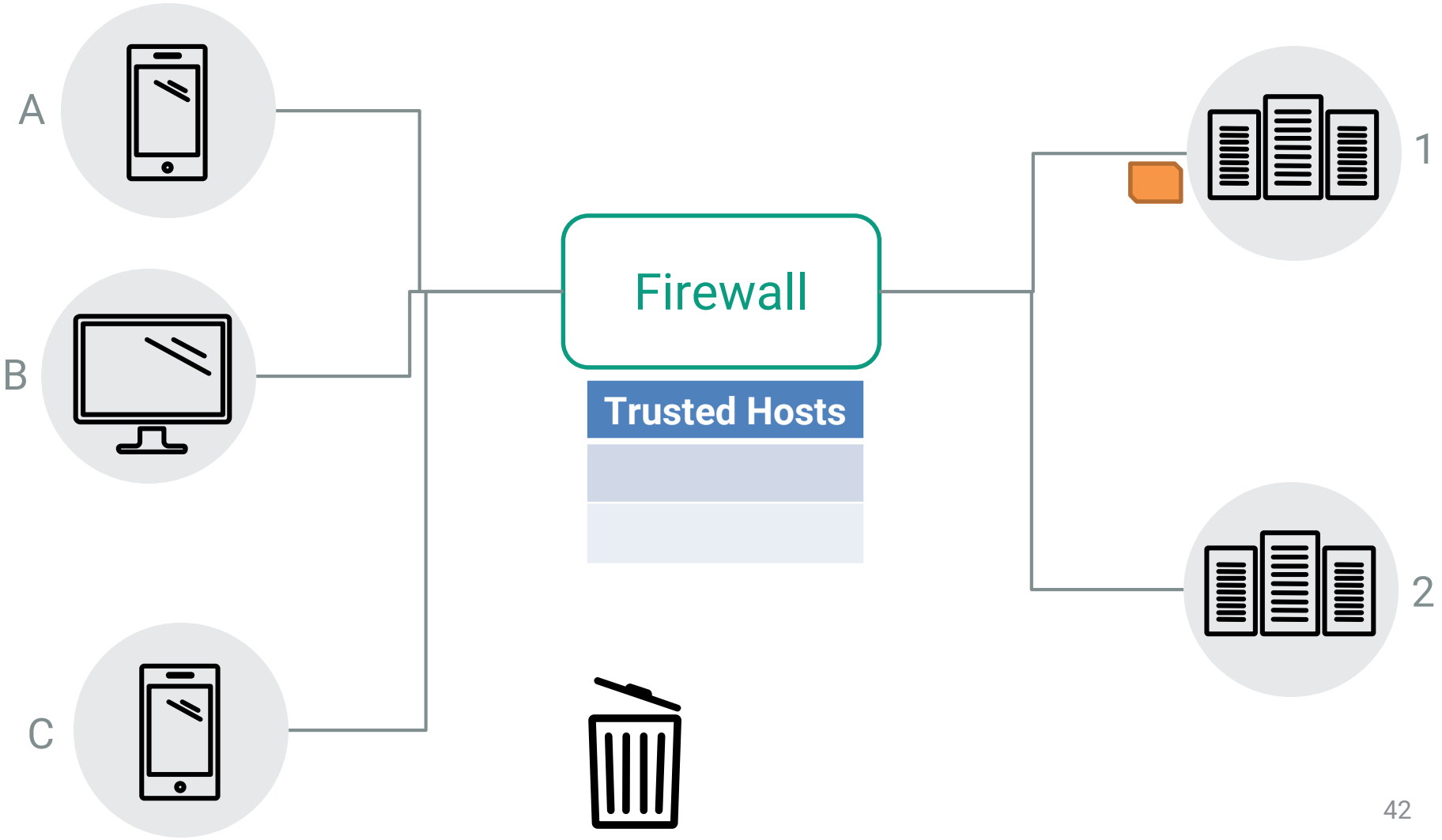
Increasing Middlebox – Hole Punching Firewall



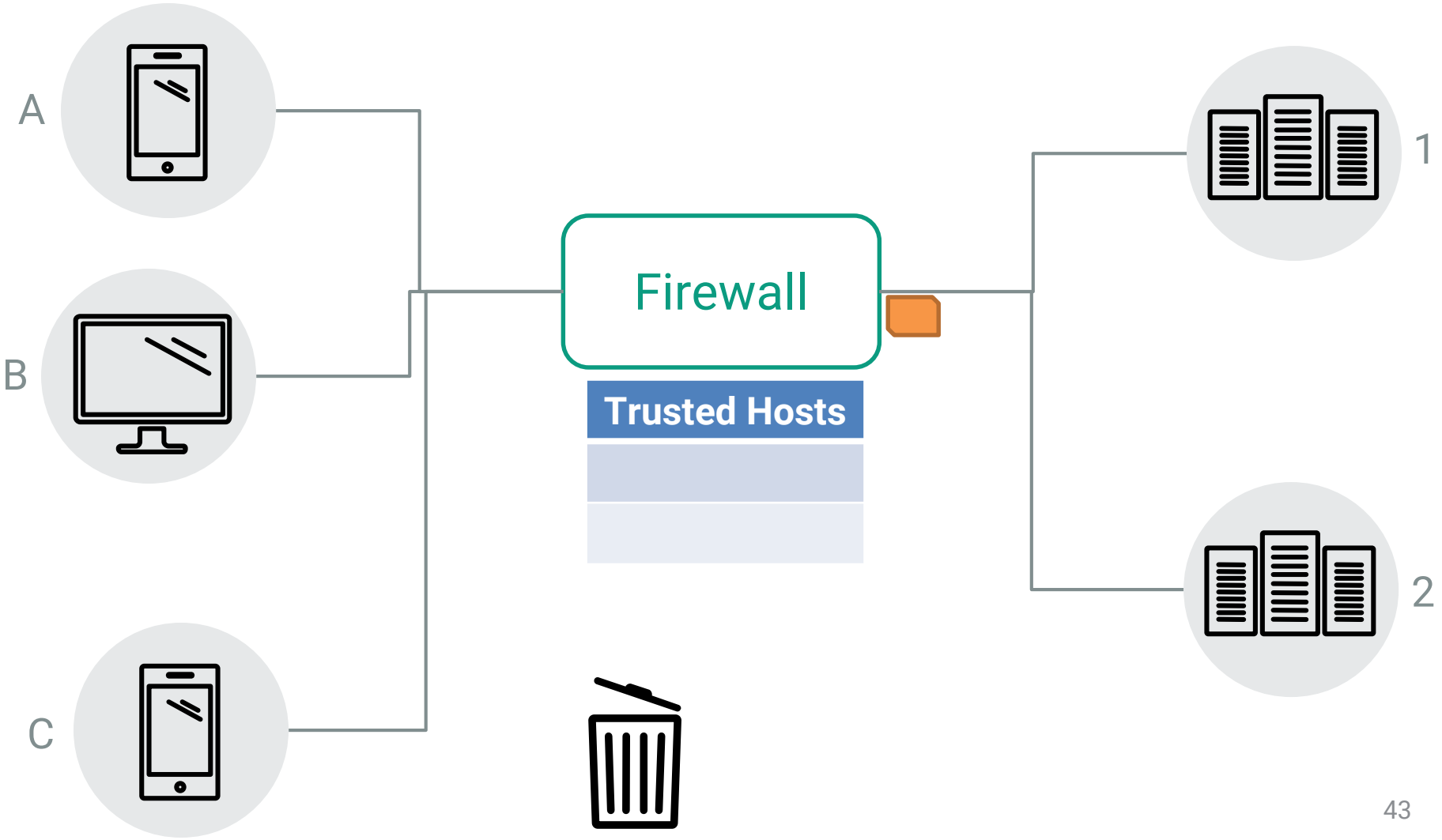
Increasing Middlebox – Hole Punching Firewall



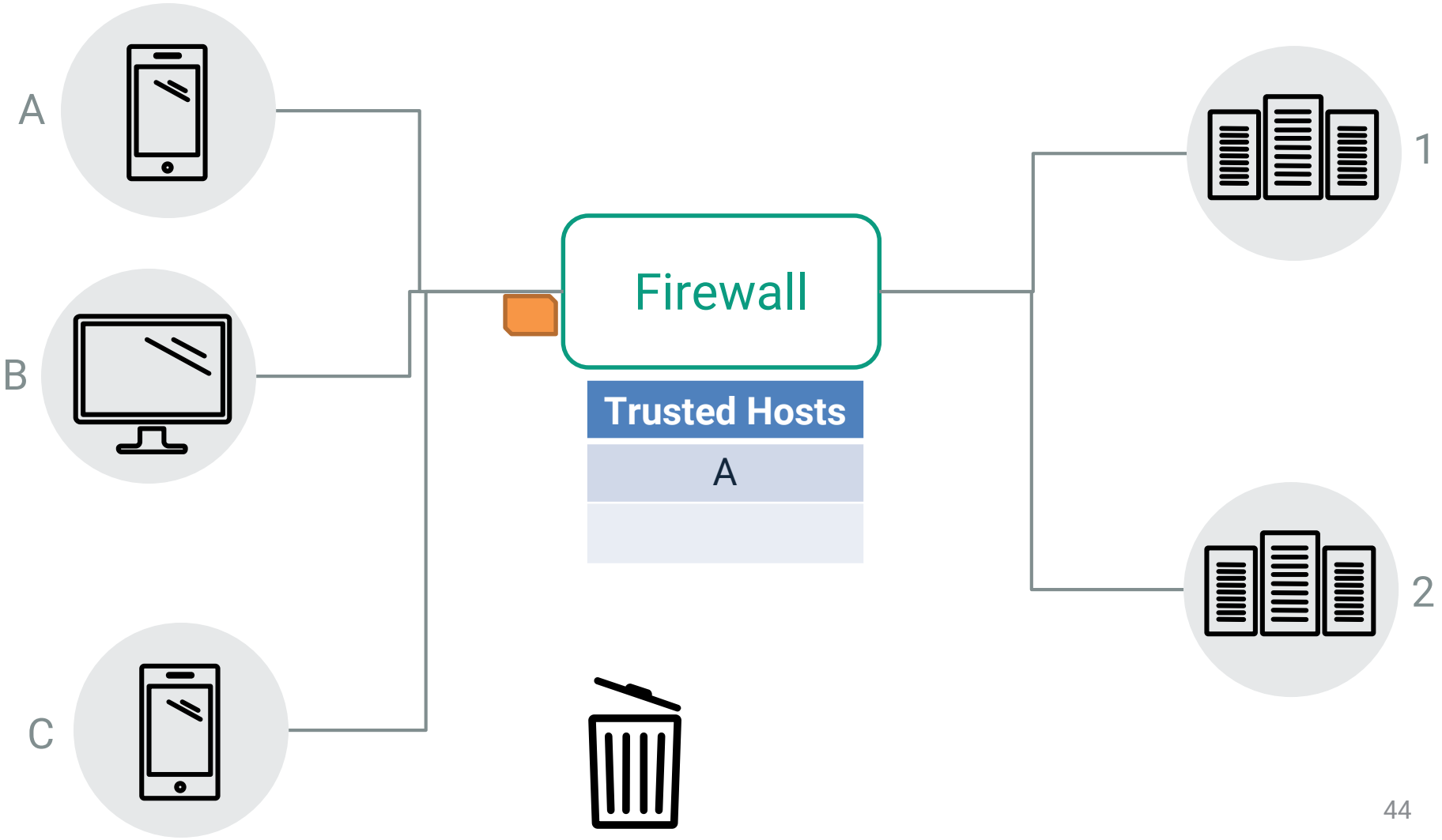
Increasing Middlebox – Hole Punching Firewall



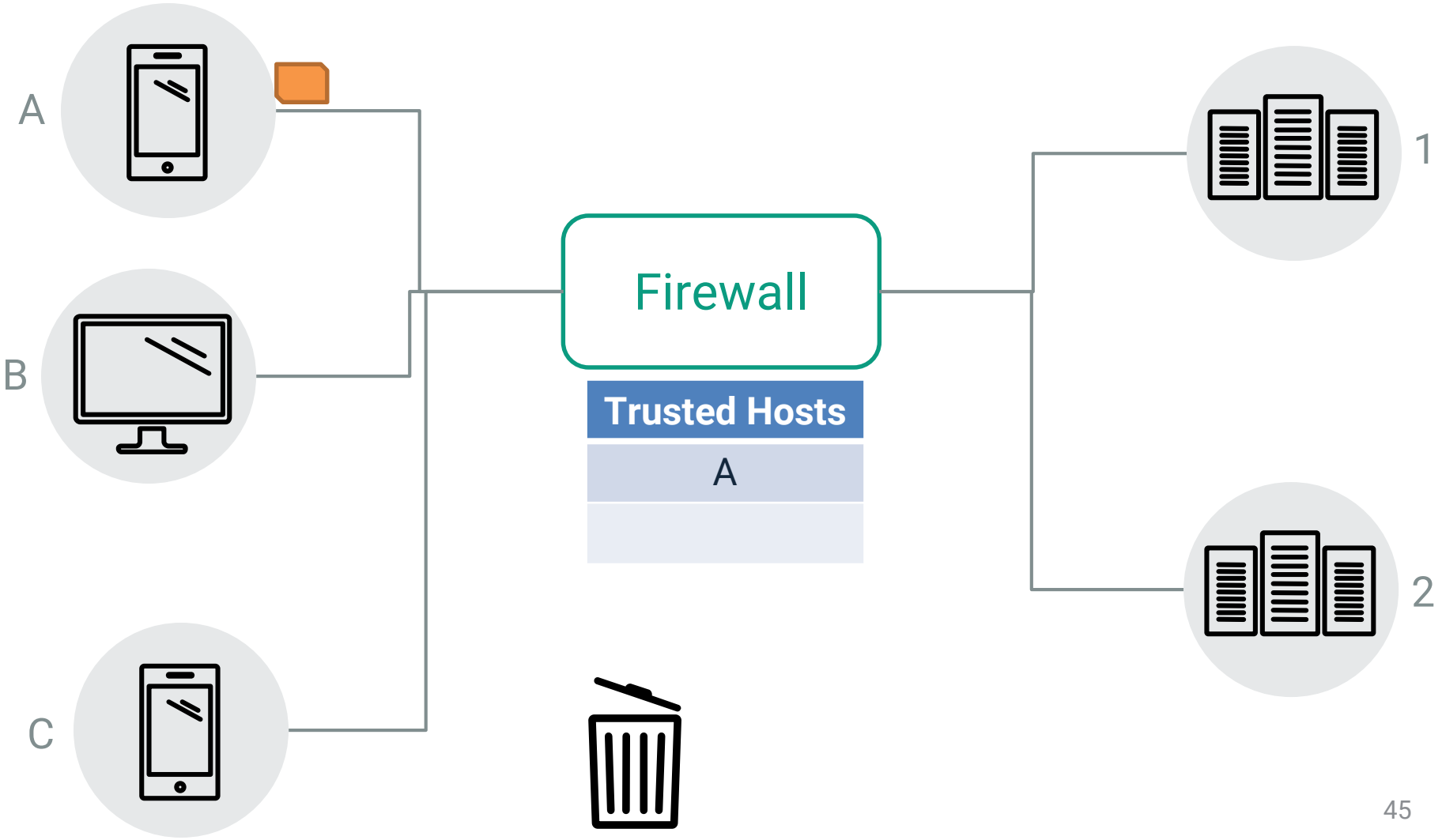
Increasing Middlebox – Hole Punching Firewall



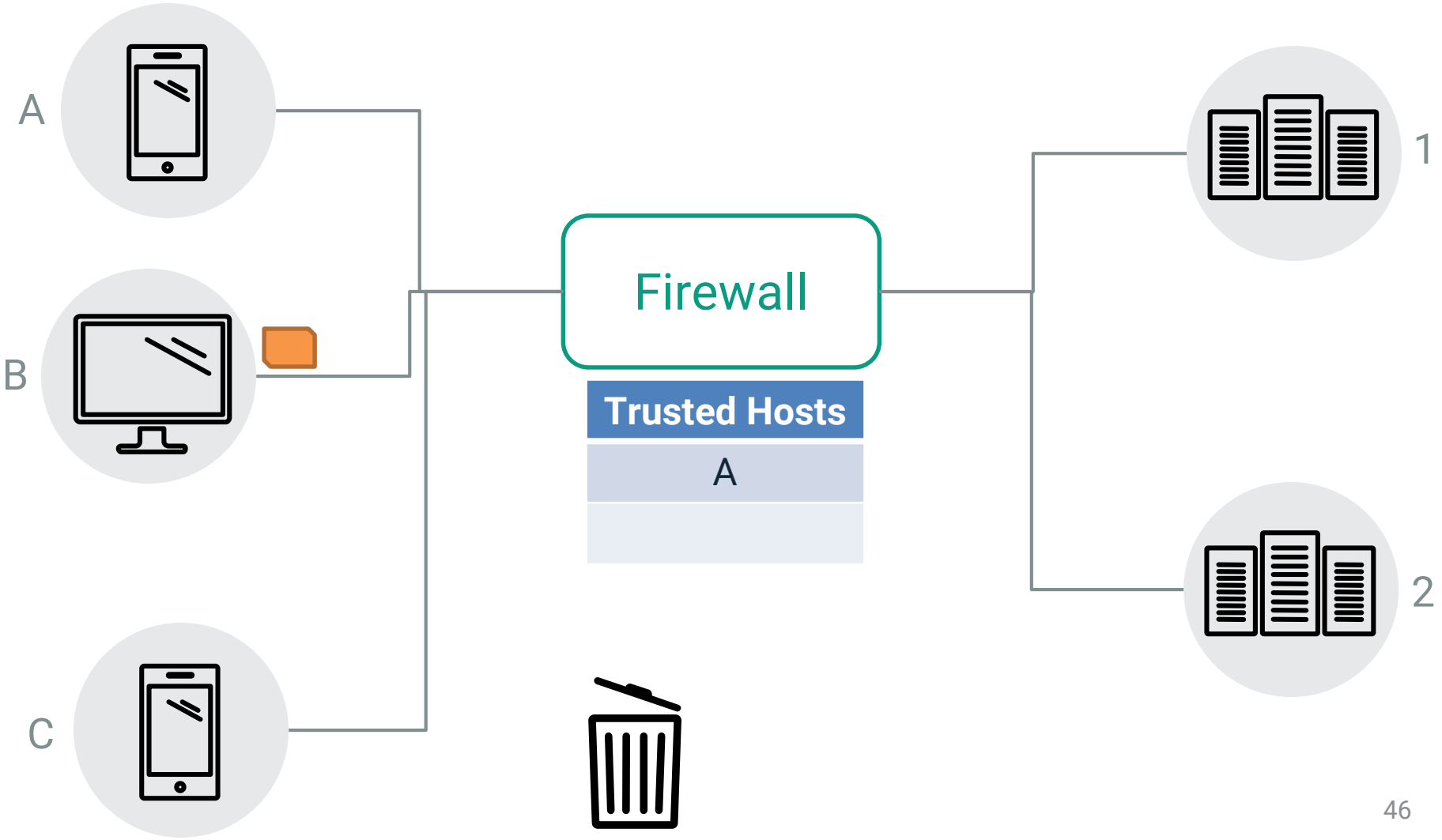
Increasing Middlebox – Hole Punching Firewall



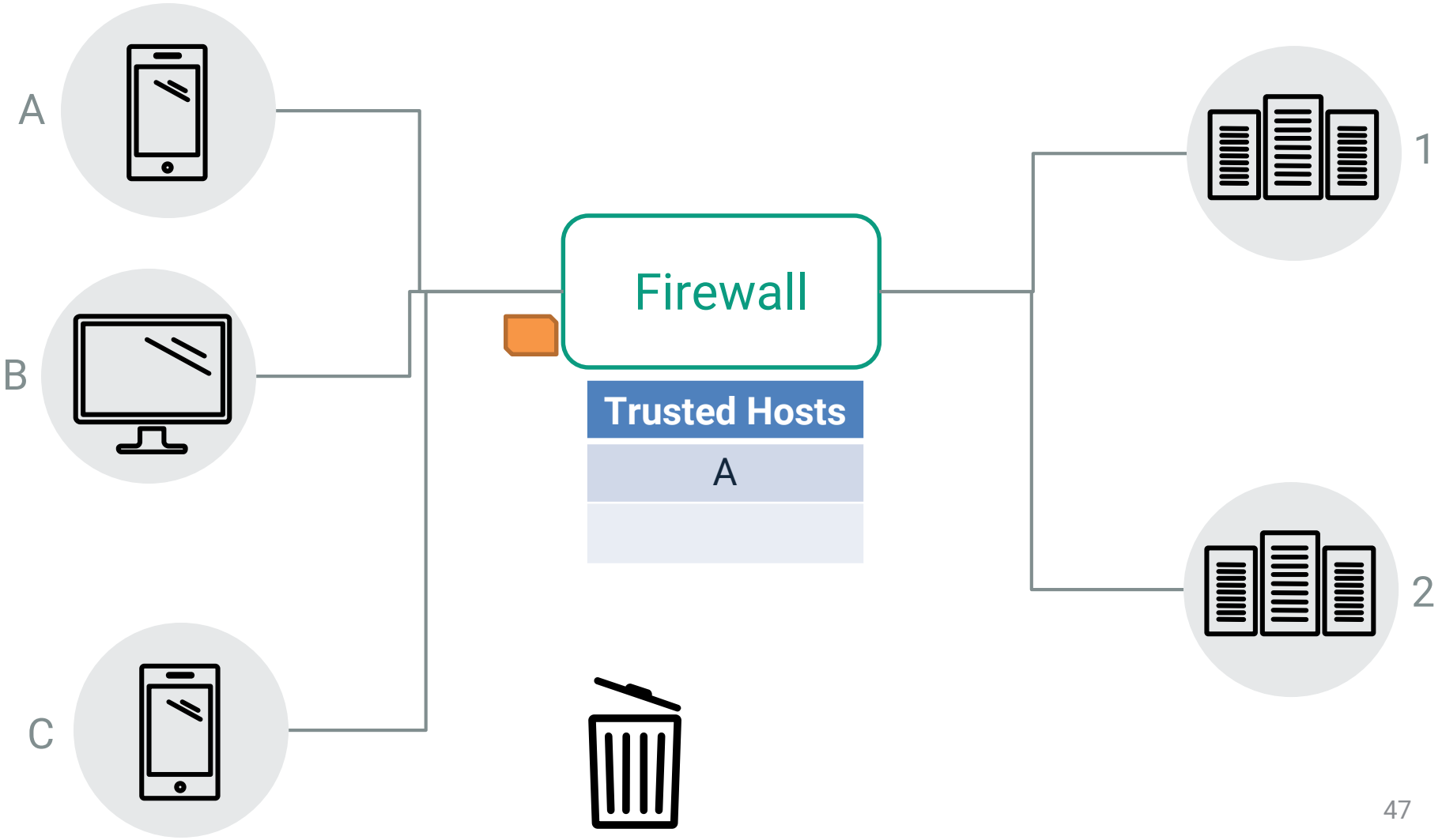
Increasing Middlebox – Hole Punching Firewall



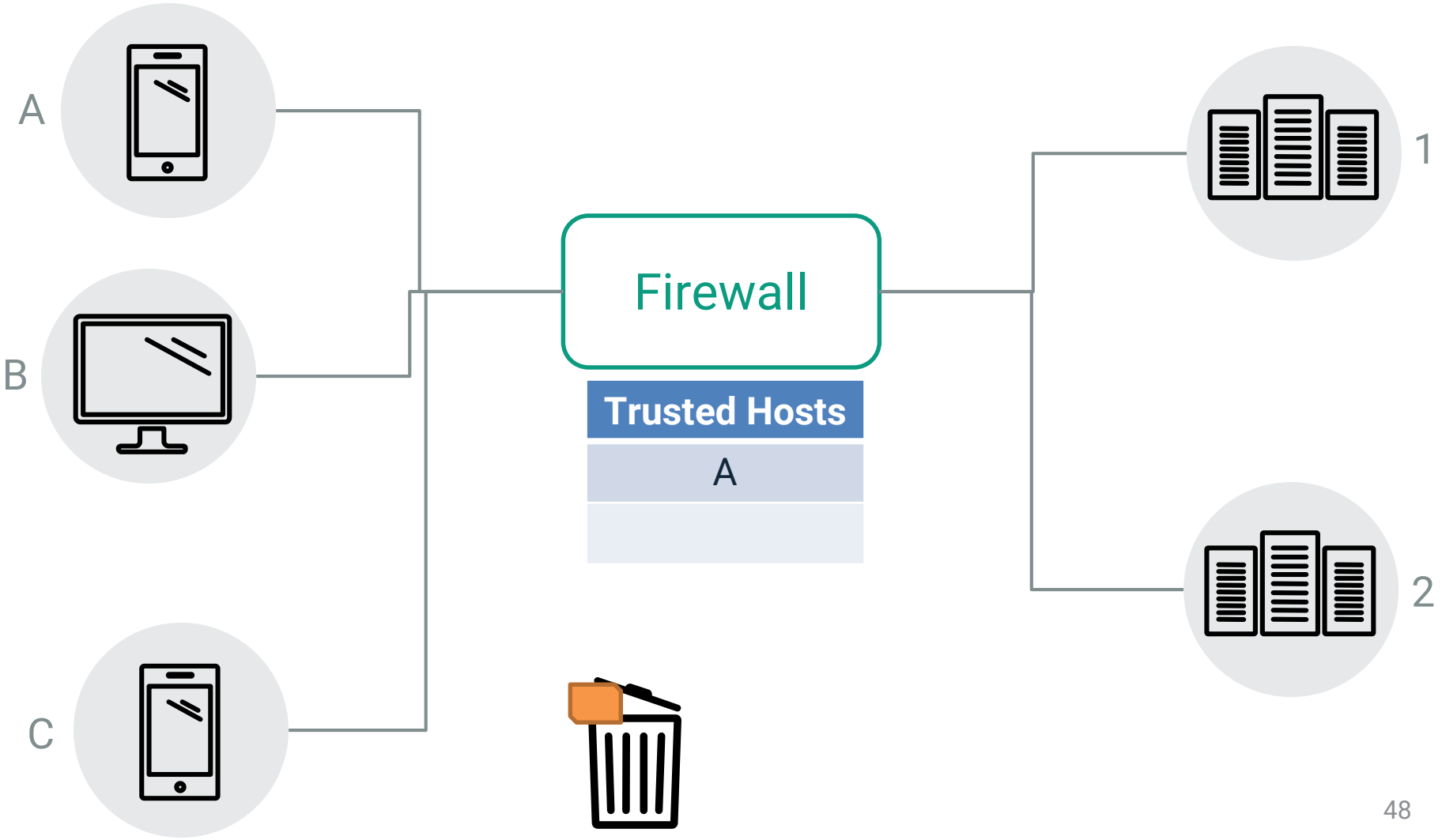
Increasing Middlebox – Hole Punching Firewall



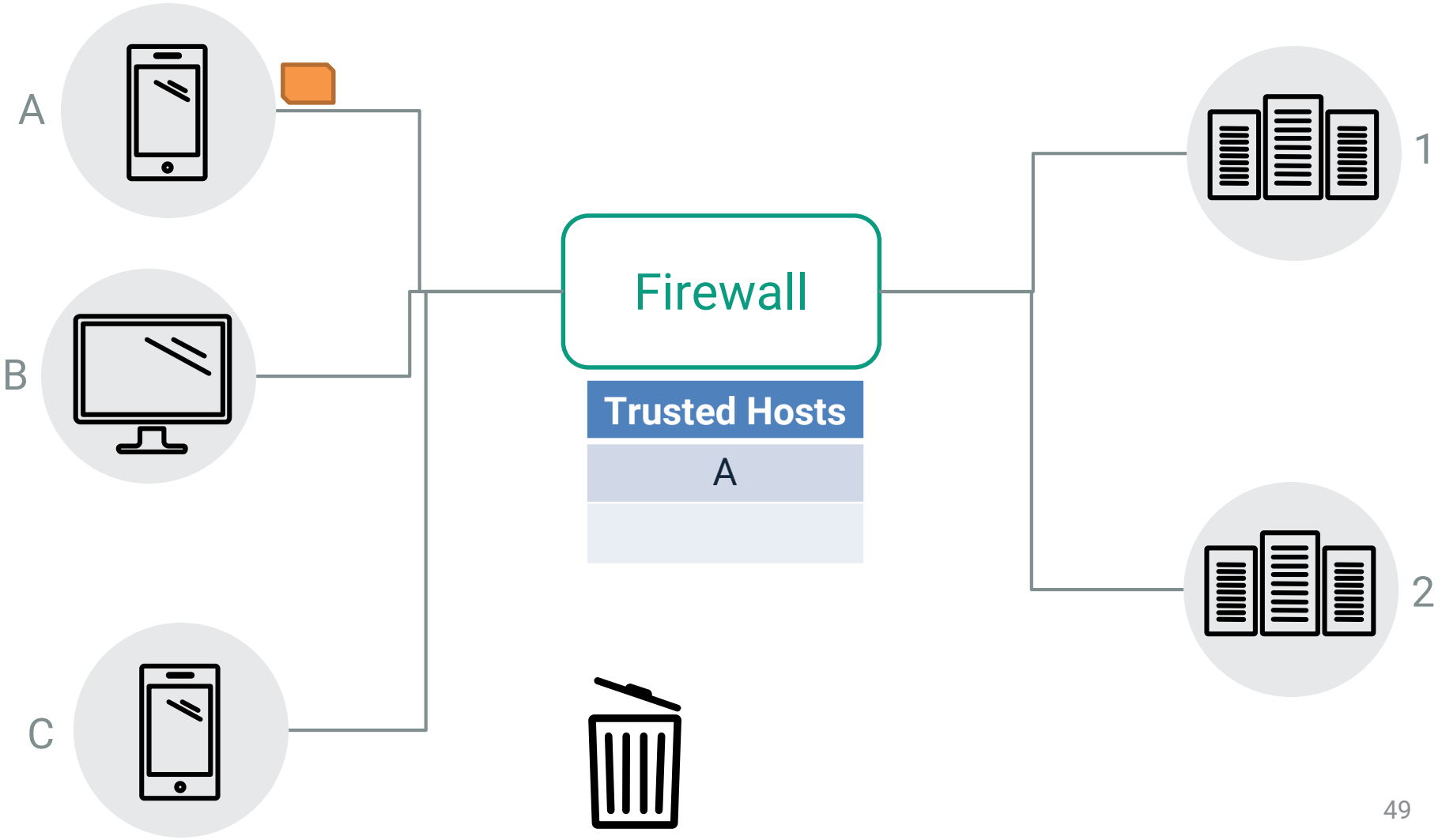
Increasing Middlebox – Hole Punching Firewall



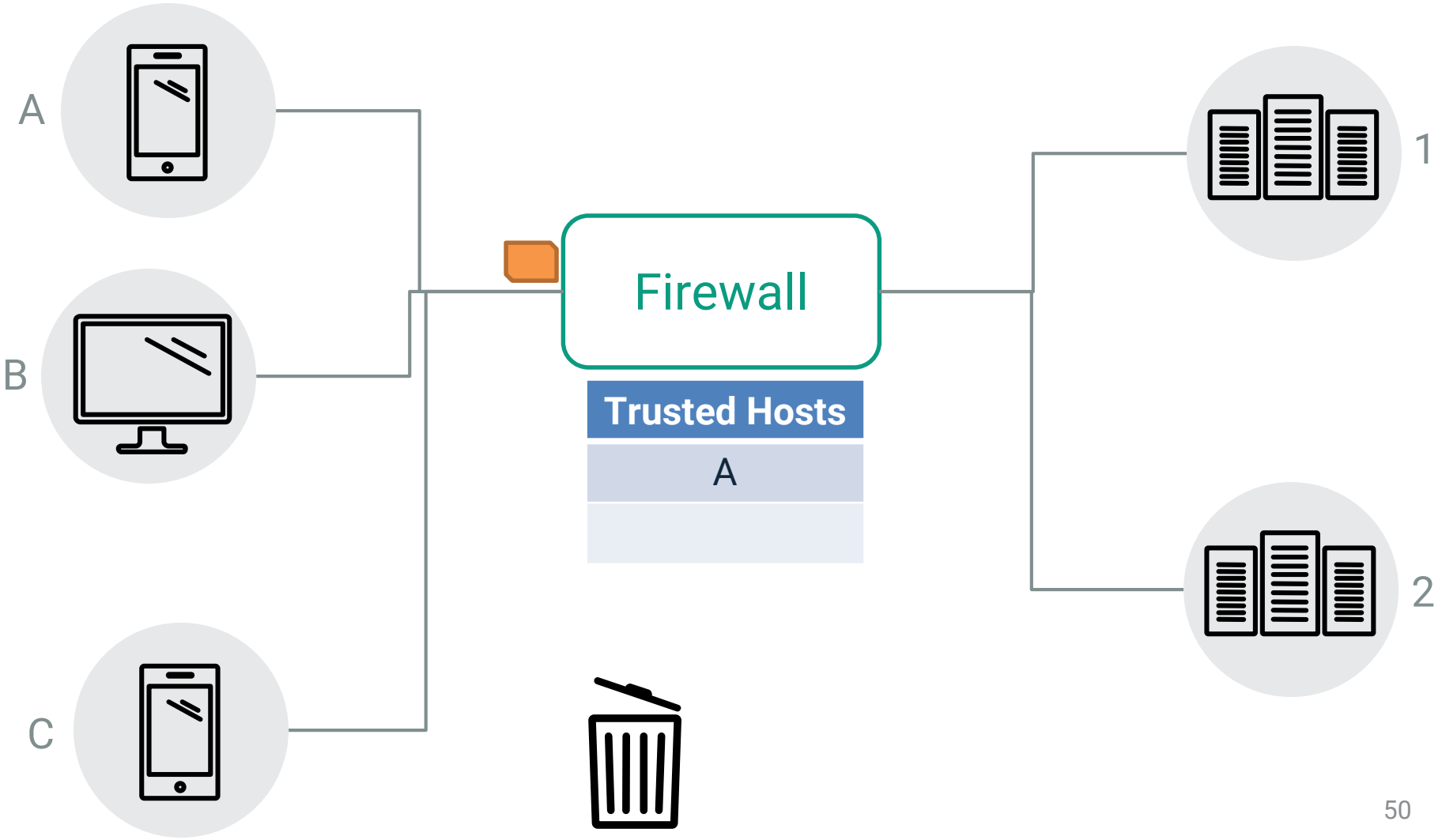
Increasing Middlebox – Hole Punching Firewall



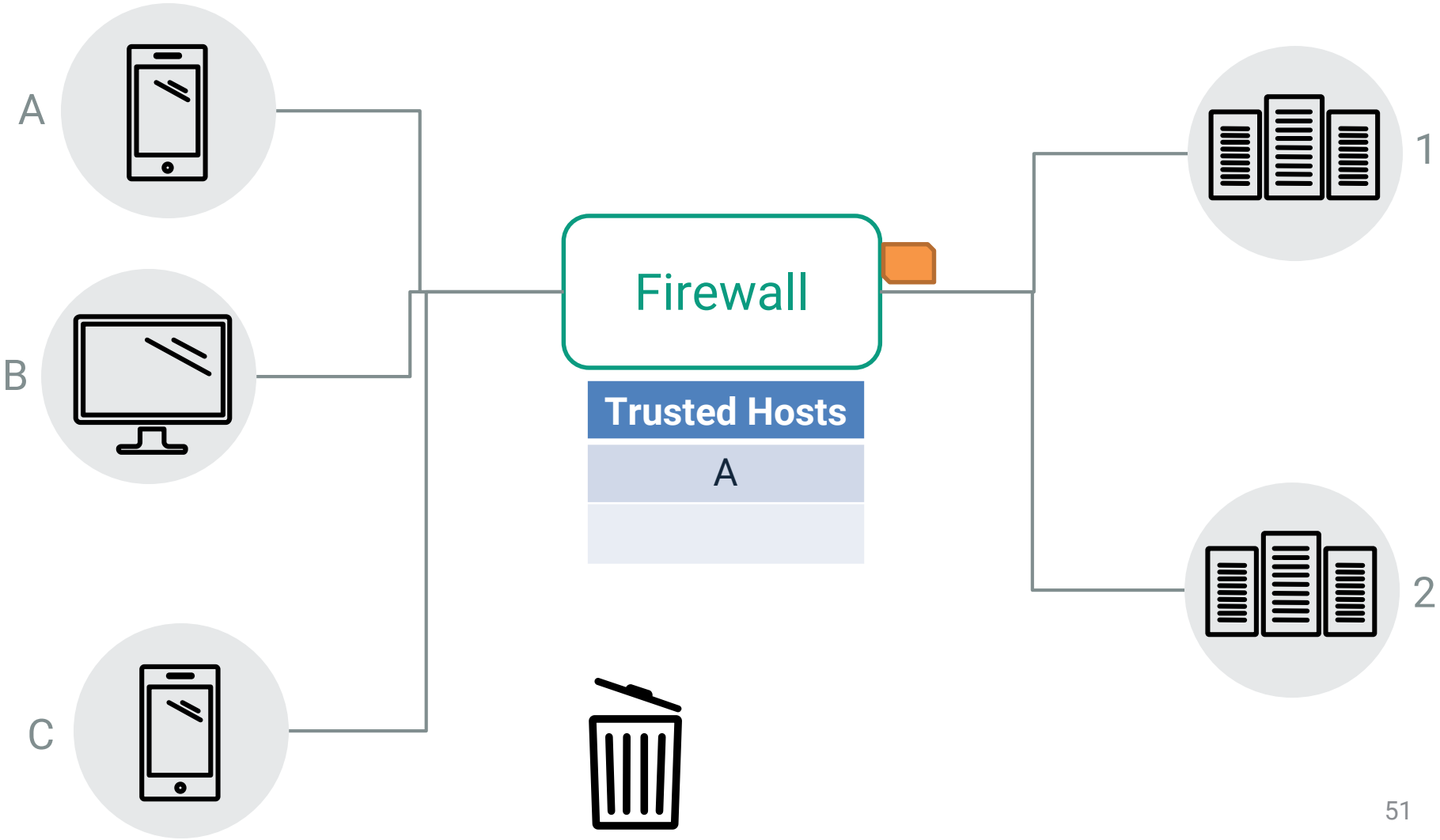
Increasing Middlebox – Hole Punching Firewall



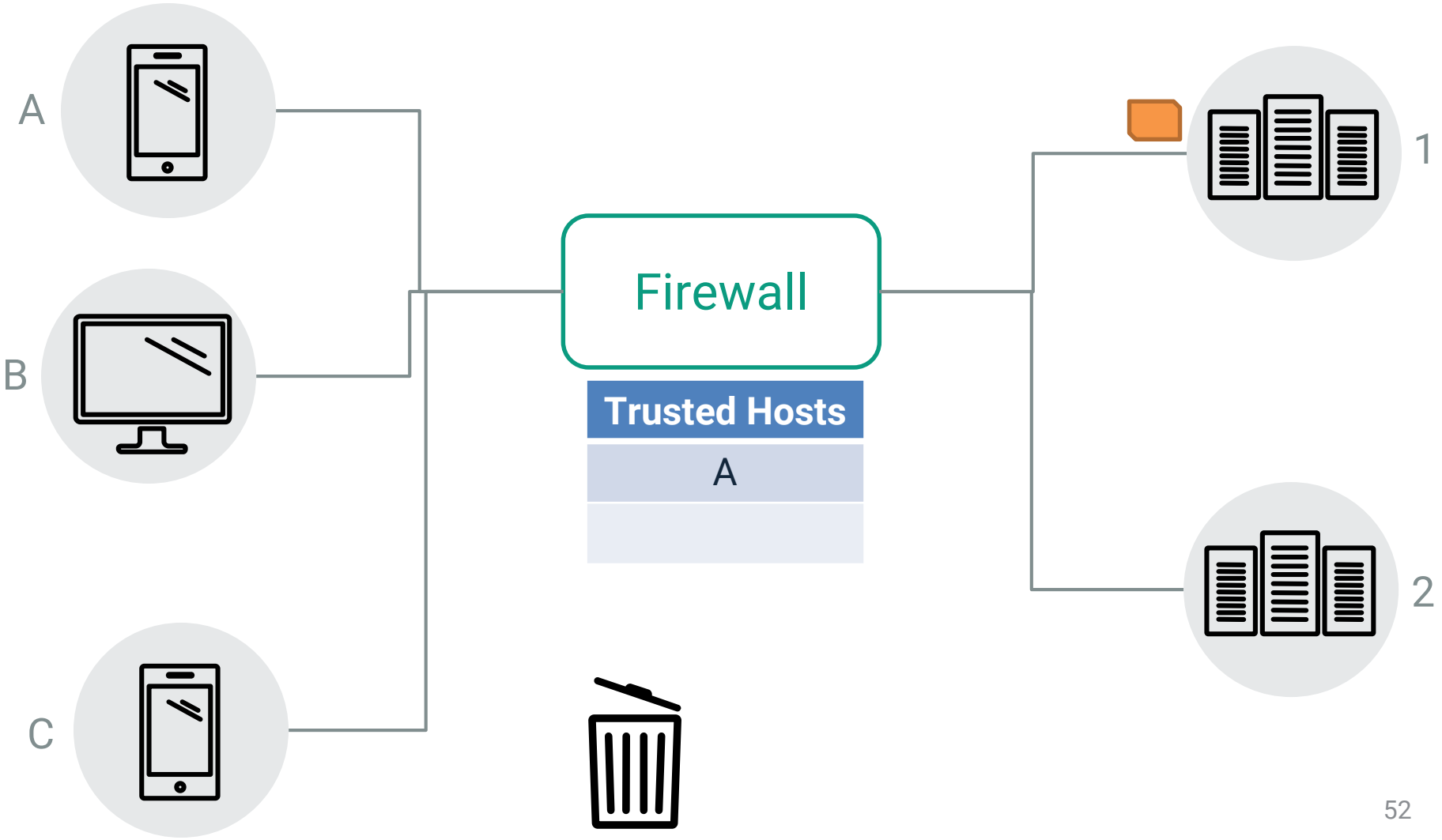
Increasing Middlebox – Hole Punching Firewall



Increasing Middlebox – Hole Punching Firewall



Increasing Middlebox – Hole Punching Firewall



Increasing

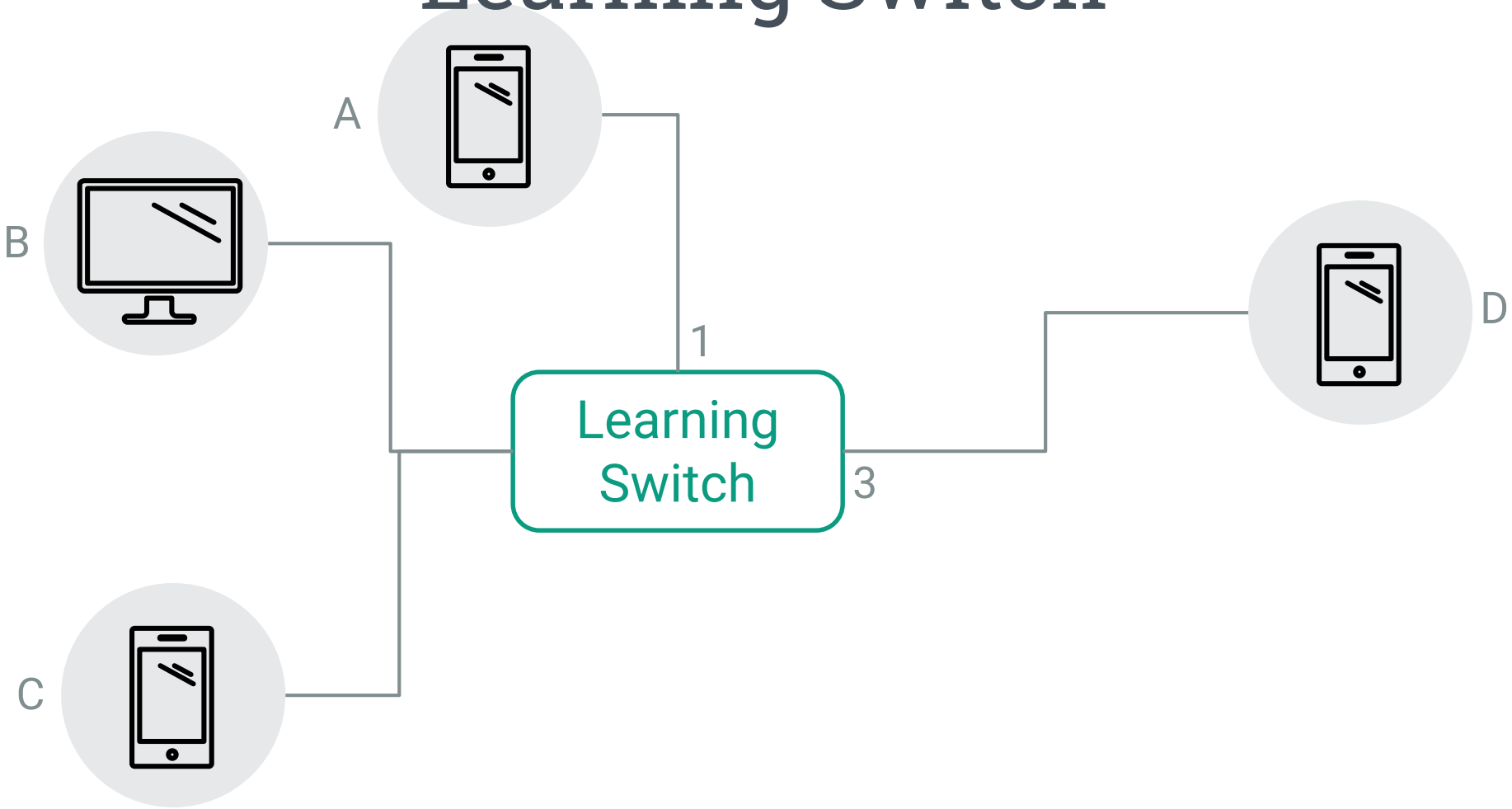
- Forwarding behaviour increases over time
- Future instance increases the output
- Syntactic restriction – no negative conditions or removals from relations
  - Monotonic guard ‘truth state’
  - Monotonic middlebox state

Increasing

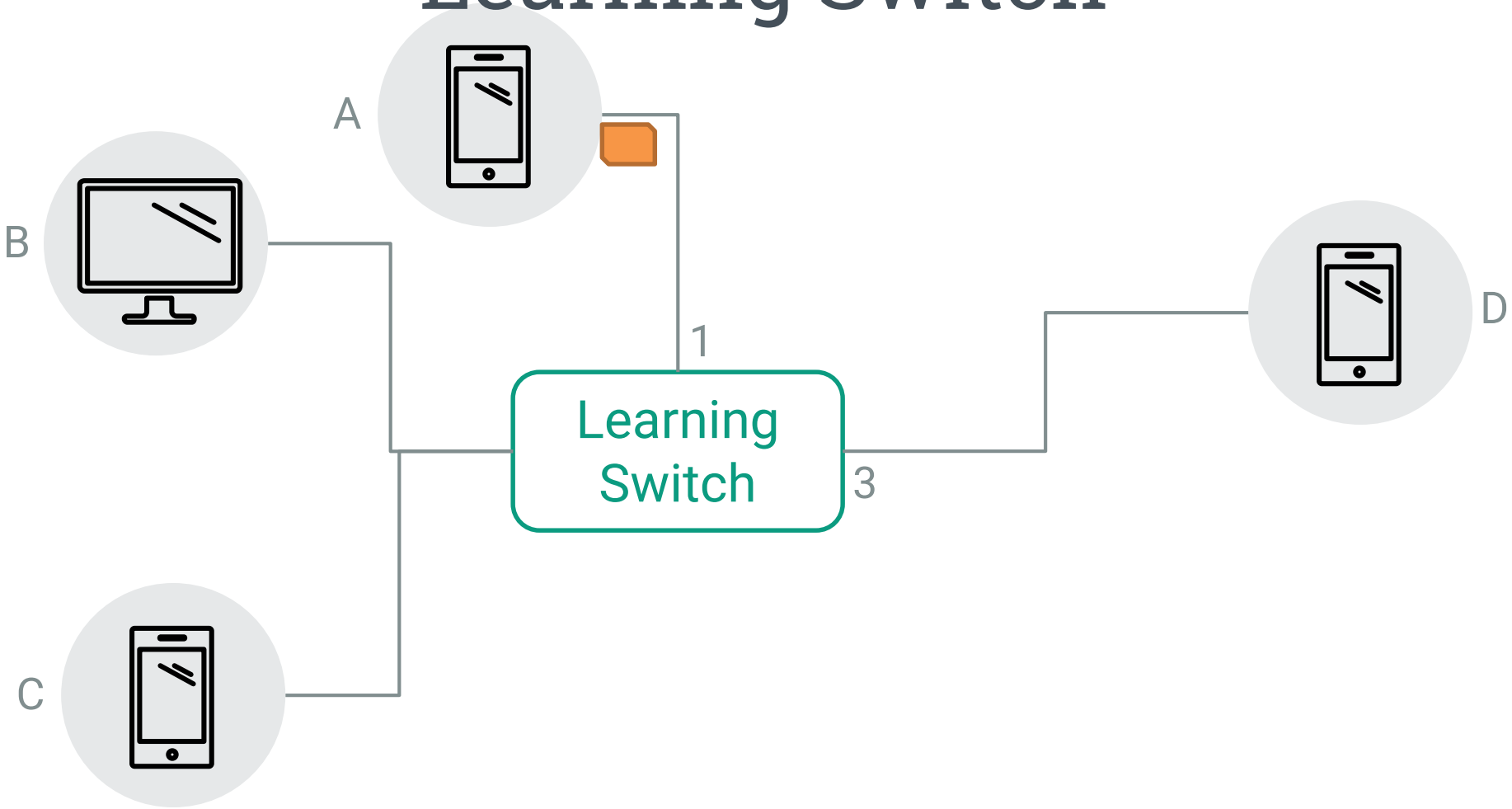
Theorem:

Safety verification of Increasing networks is in **PTIME**

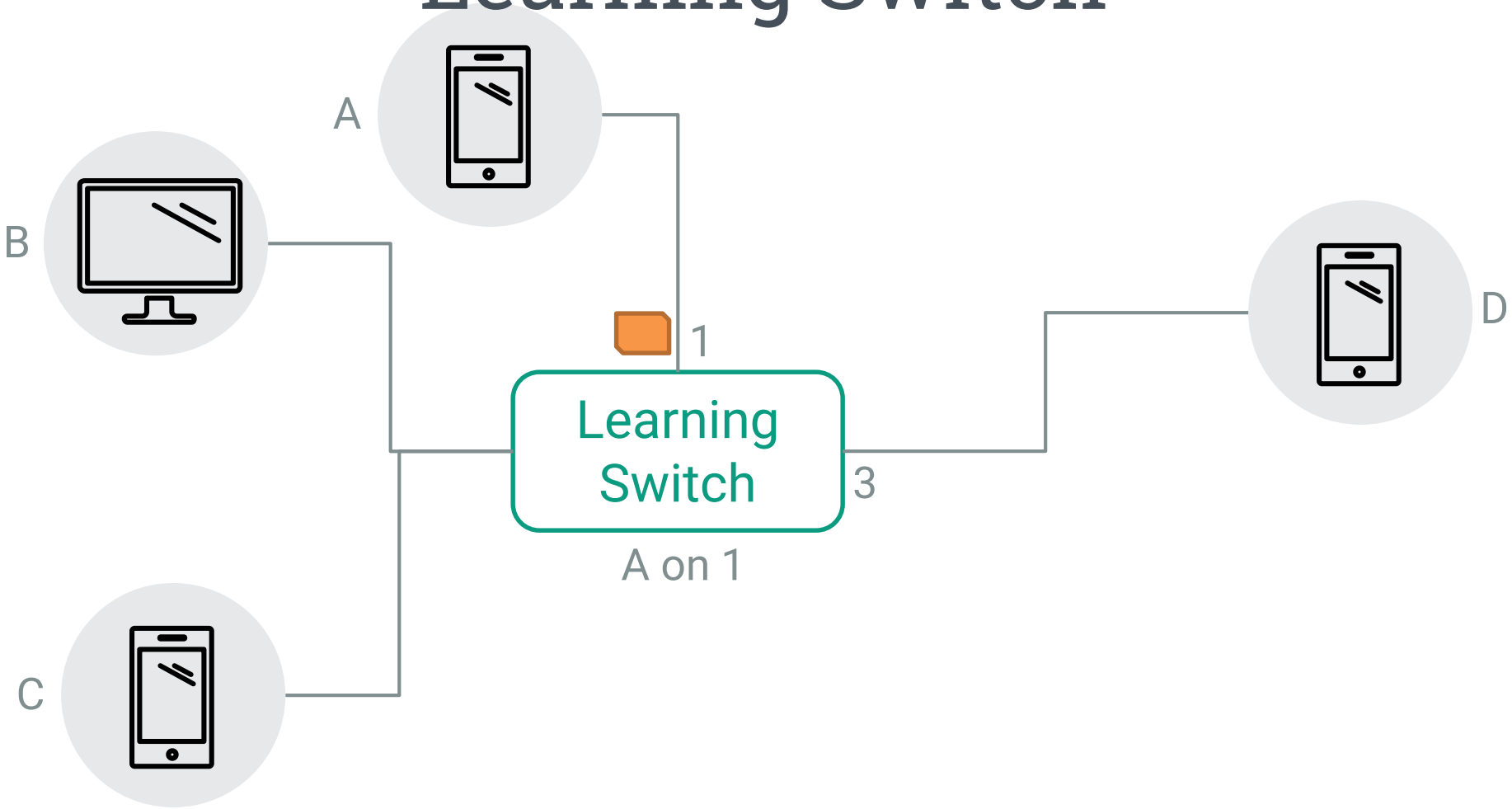
Progressing Middlebox – Learning Switch



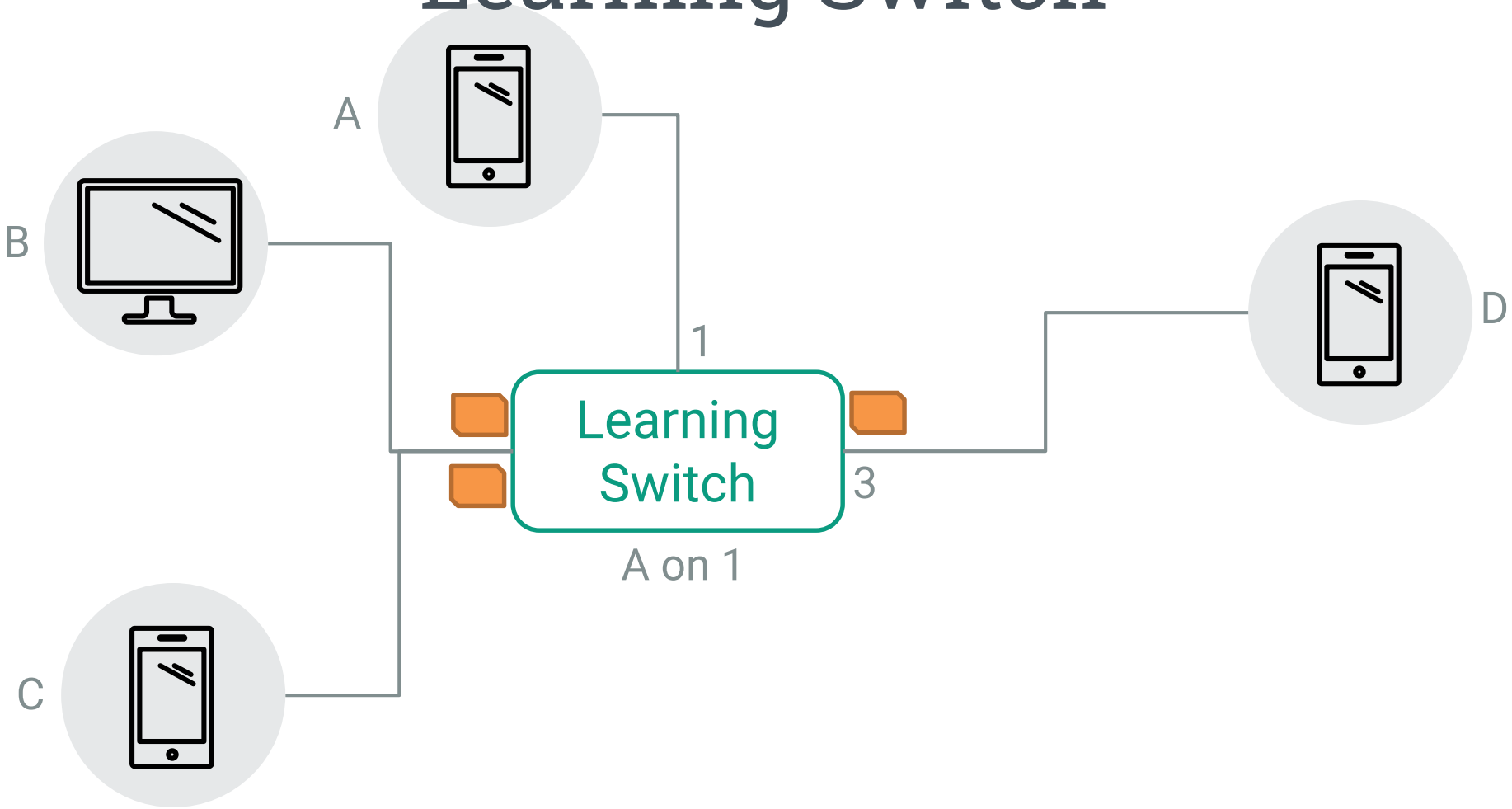
Progressing Middlebox – Learning Switch



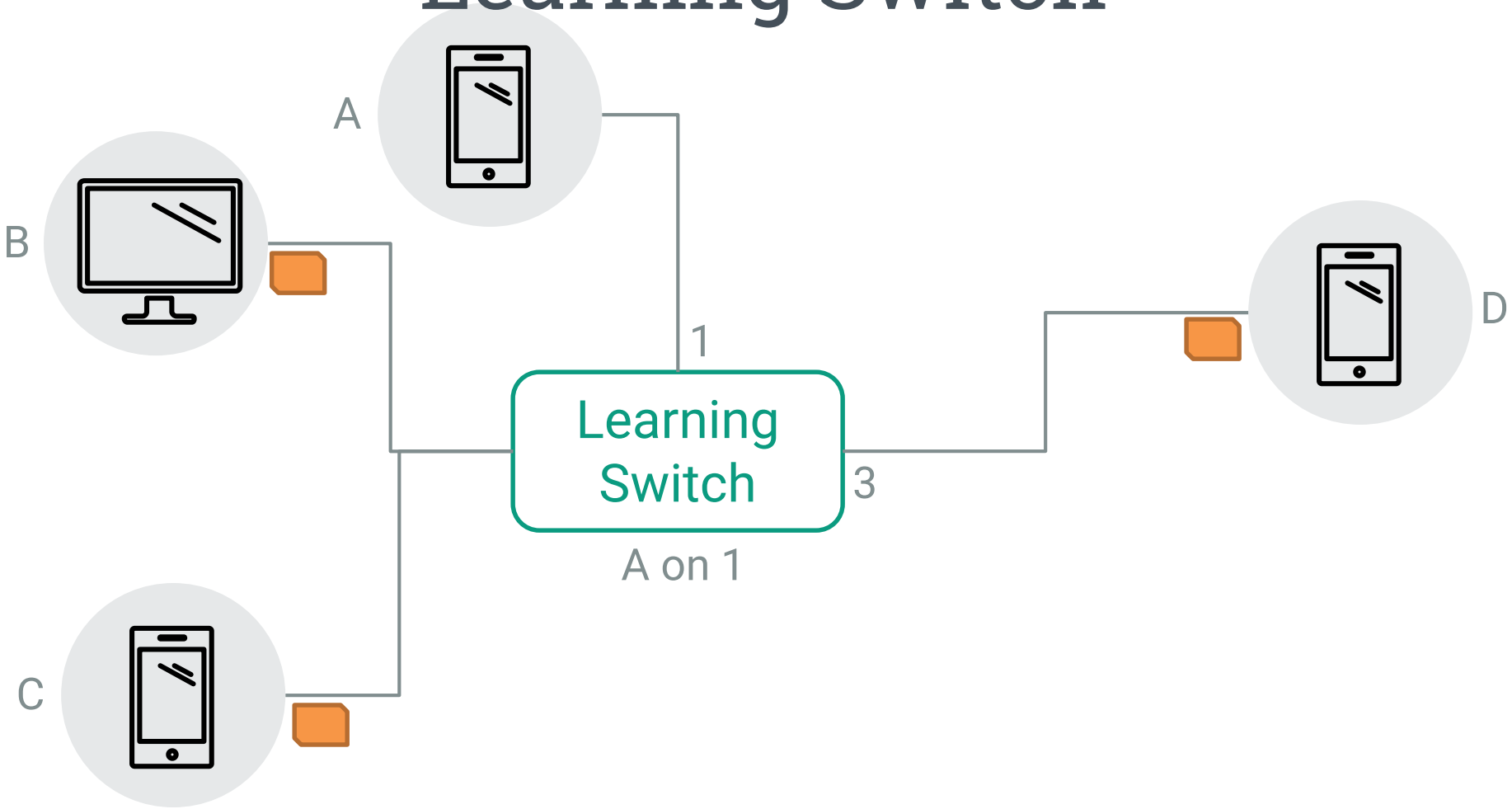
Progressing Middlebox – Learning Switch



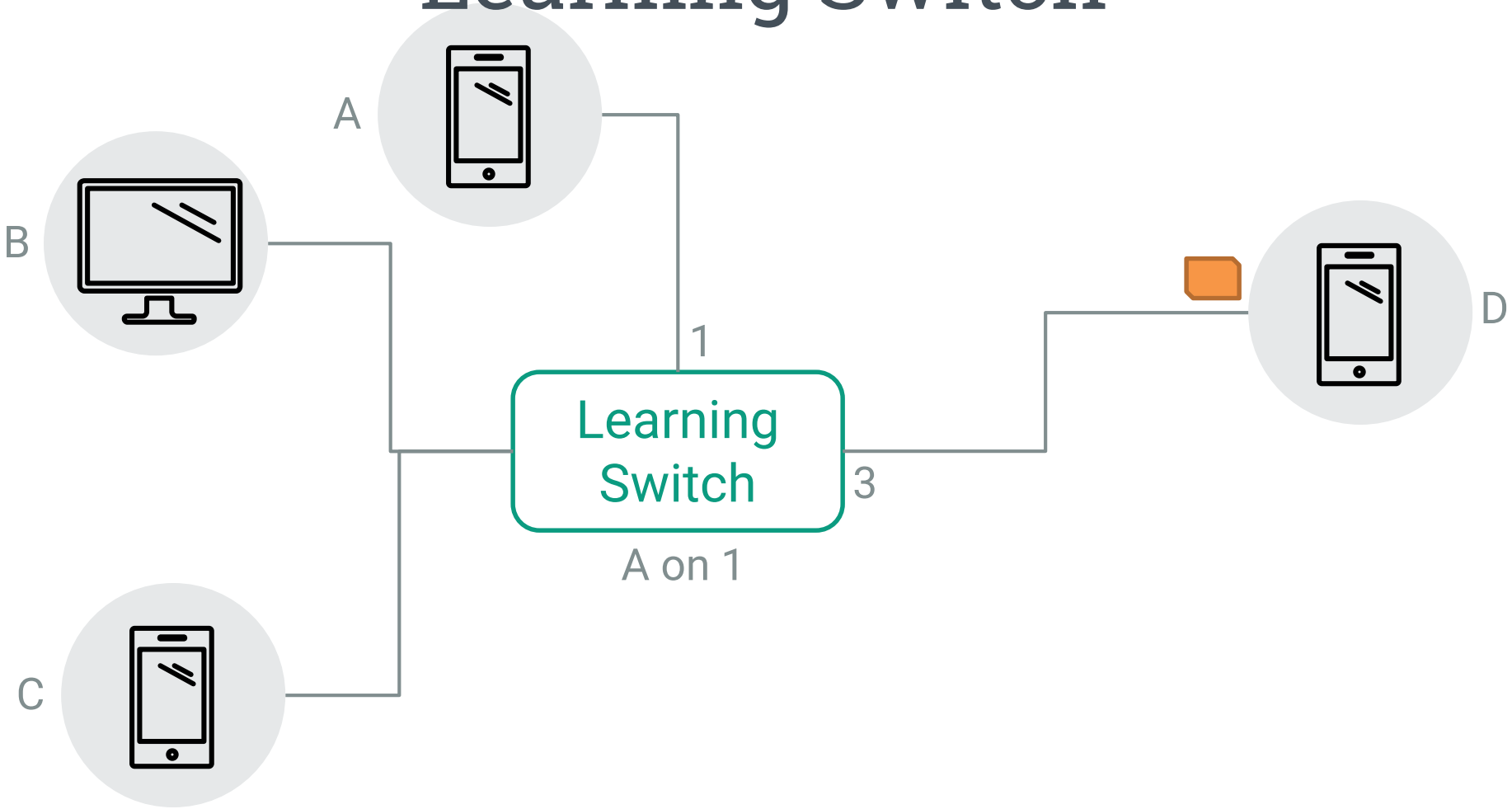
Progressing Middlebox – Learning Switch



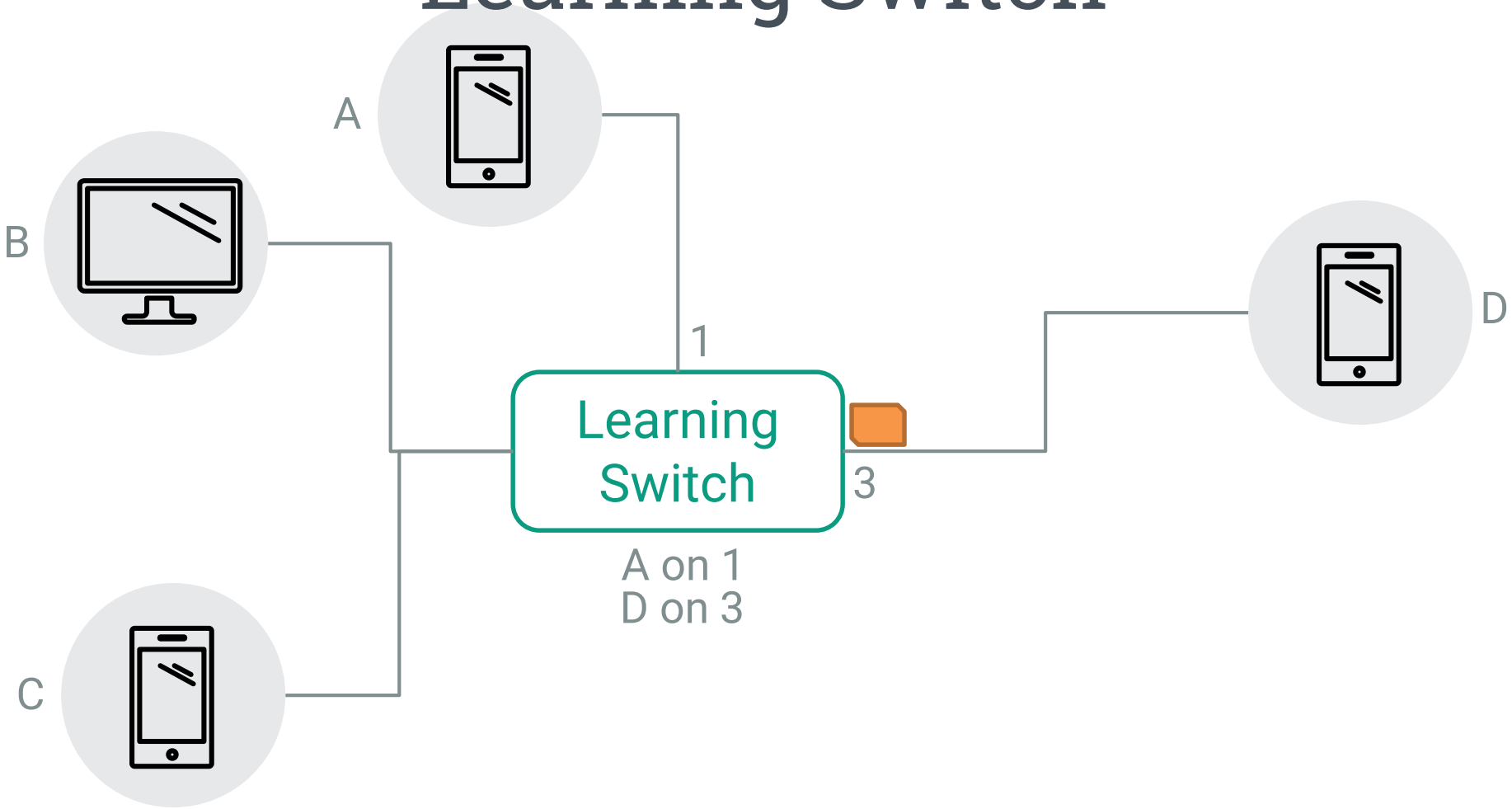
Progressing Middlebox – Learning Switch



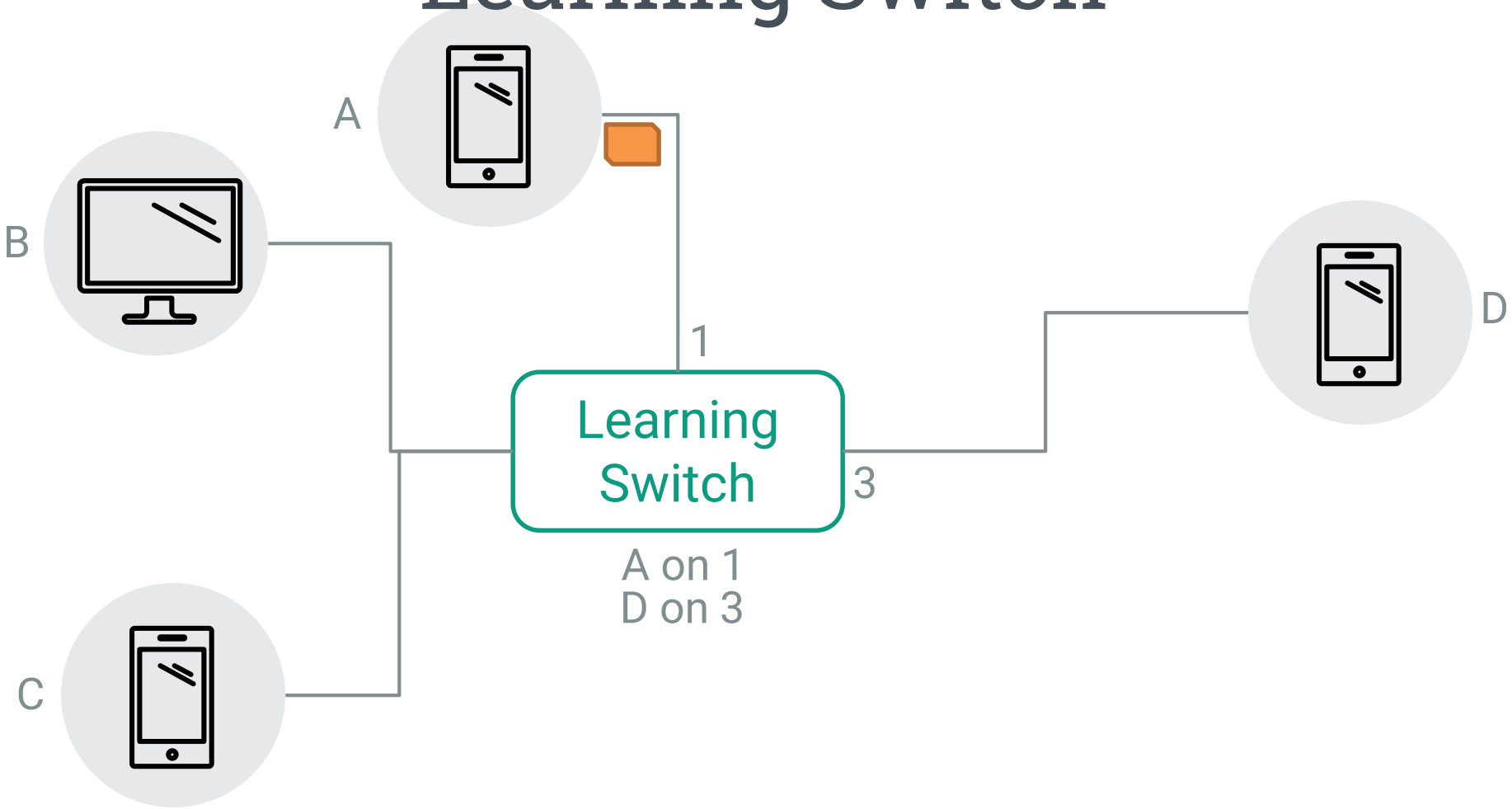
Progressing Middlebox – Learning Switch



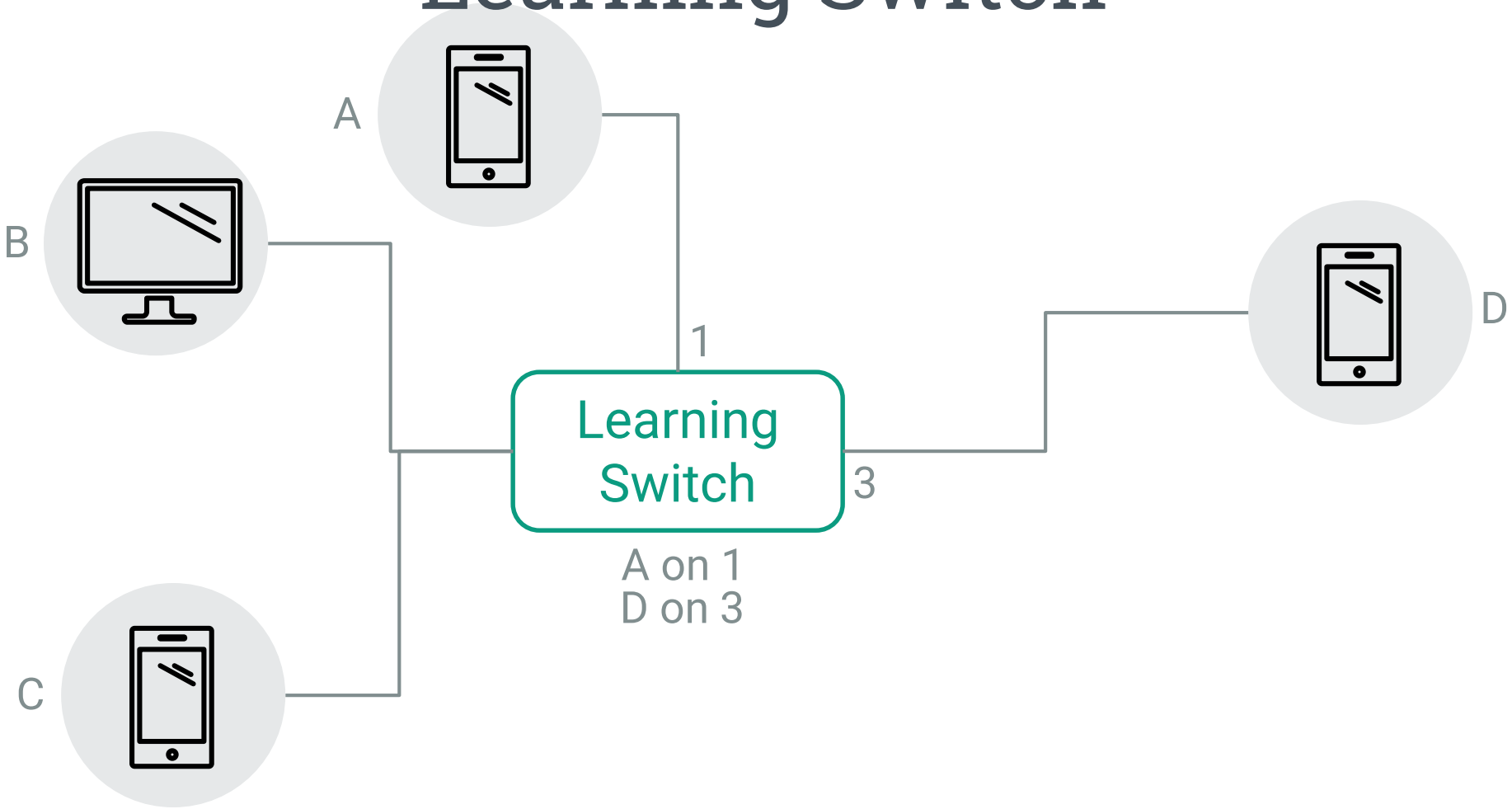
Progressing Middlebox – Learning Switch



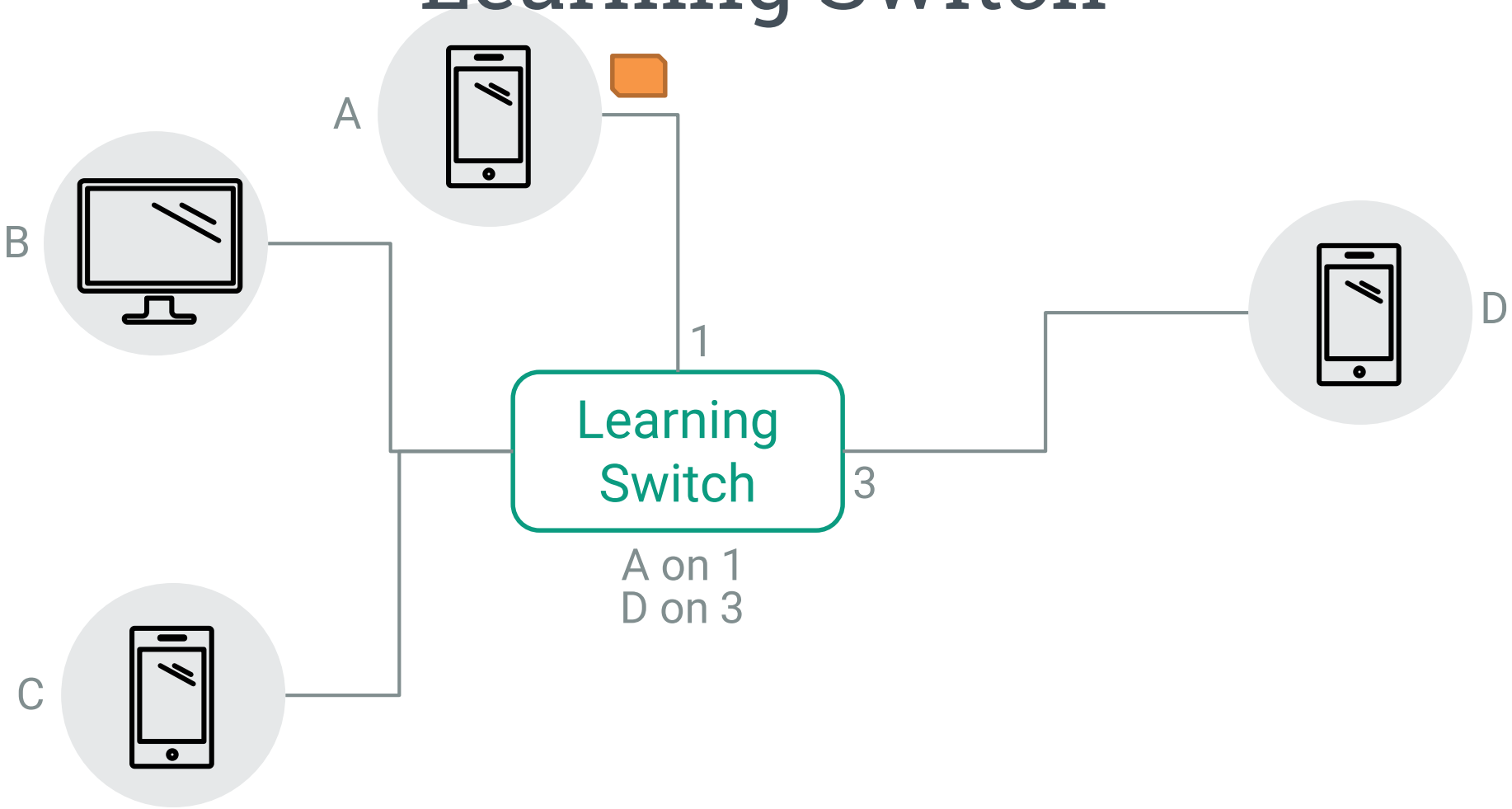
Progressing Middlebox – Learning Switch



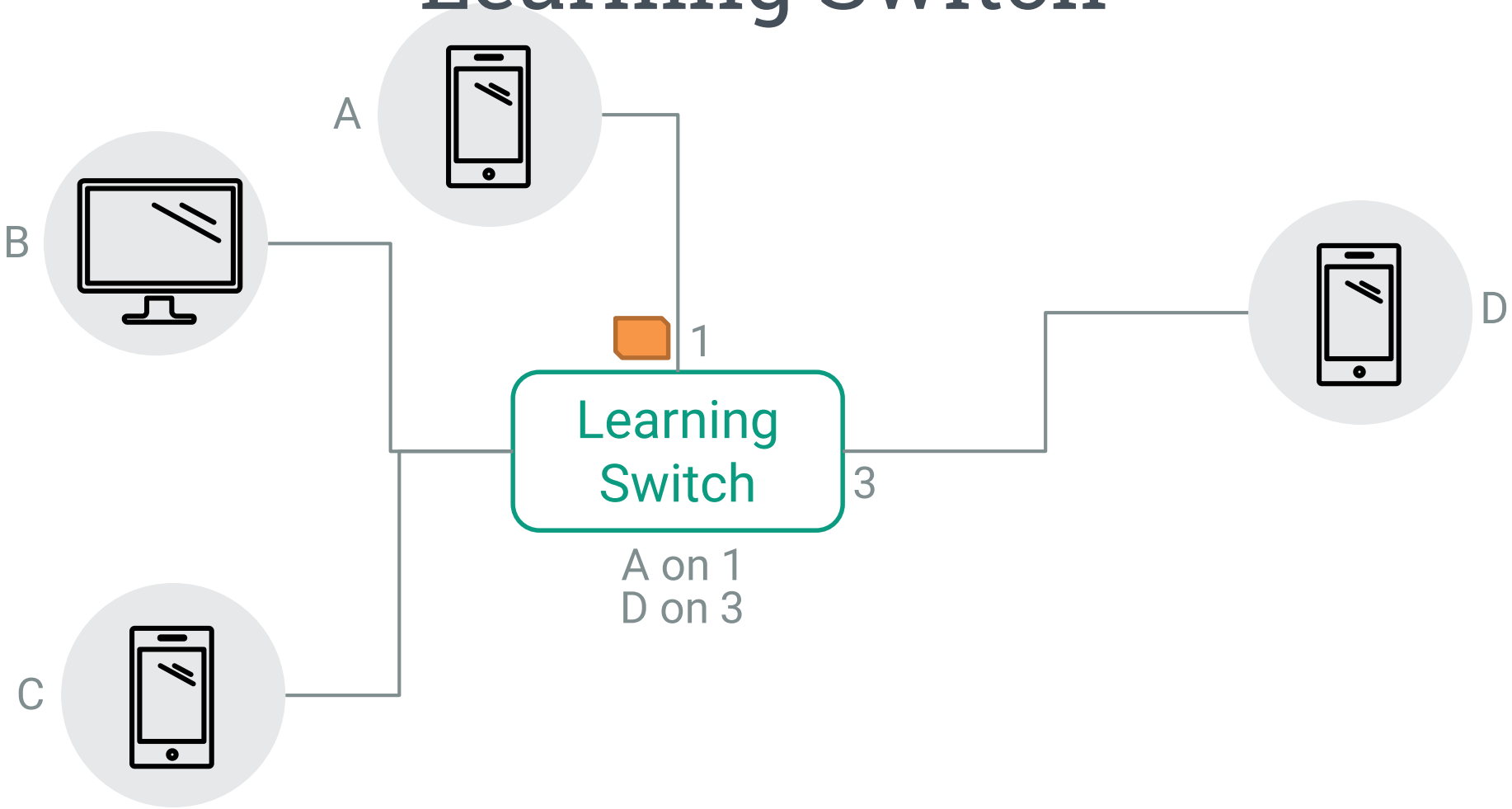
Progressing Middlebox – Learning Switch



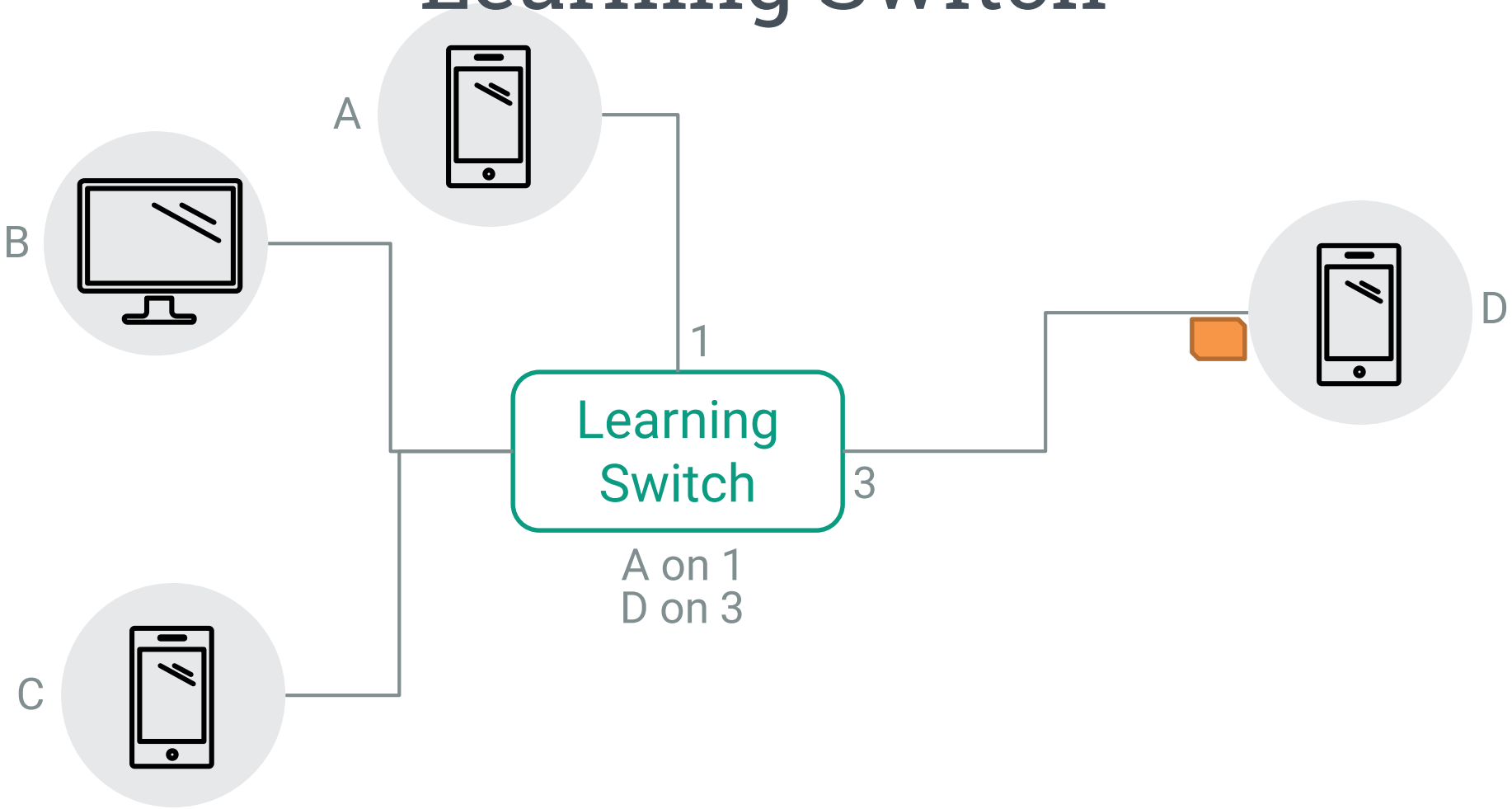
Progressing Middlebox – Learning Switch



Progressing Middlebox – Learning Switch



Progressing Middlebox – Learning Switch



Progressing

- Forwarding behaviour ‘progresses’ over time
- The transducer state graph is a DAG
- Syntactic restriction – no removals from relations
  - Monotonic middlebox state

Progressing

Theorem:

Safety verification of Progressing networks is **coNP-Complete**

Complexity Result Summary

Class	Unordered	FIFO
Stateless	PTIME	PTIME
Increasing	PTIME	PTIME
Progressing	coNP-Complete	?
Arbitrary	EXPSPACE	Undecidable

Summary

- Classify middleboxes according to the forwarding behaviour
  - Dependence on history
- Tight complexity Results for the different classes
- Compact symbolic representation preserves complexity results
  - Exponential saving in common cases