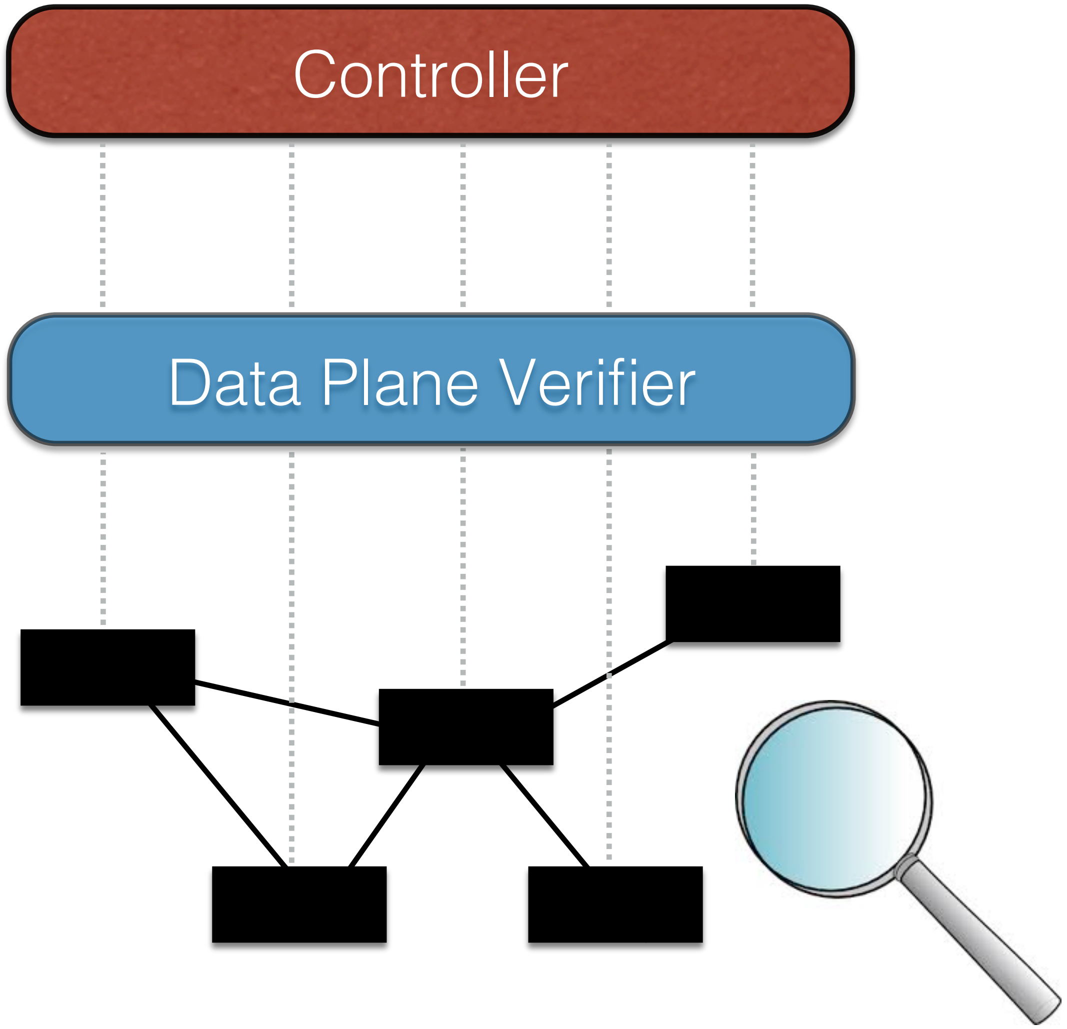
An Assertion Language for Debugging SDN Applications

Ryan Beckett
with
X. Kelvin Zou, Shuyuan Zhang,
Sharad Malik, Jennifer Rexford, David Walker

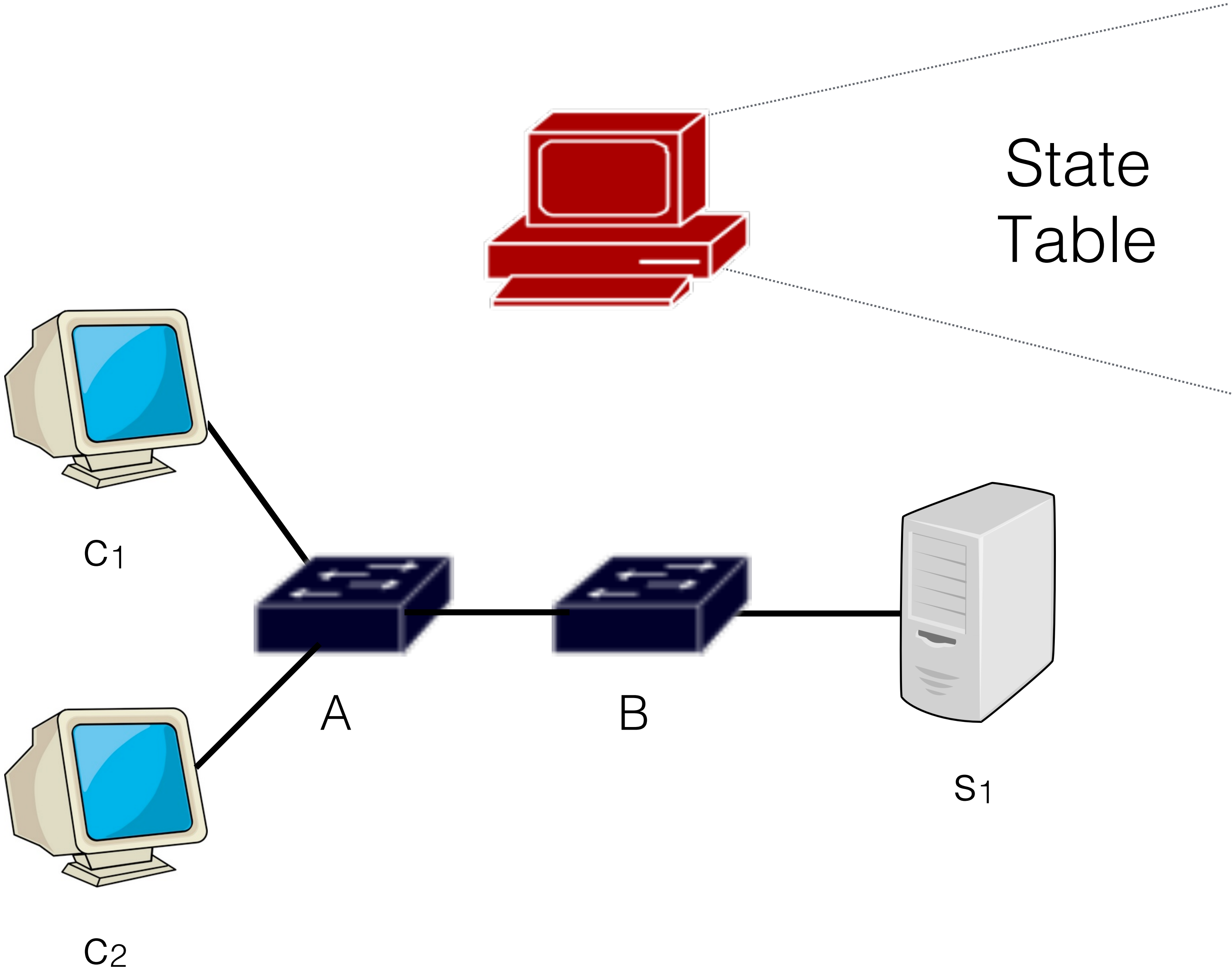
Princeton University

Data Plane Verification

- Find common misconfigurations
- Operate in real time
- Check fixed network properties
- Can report false positives during transitions



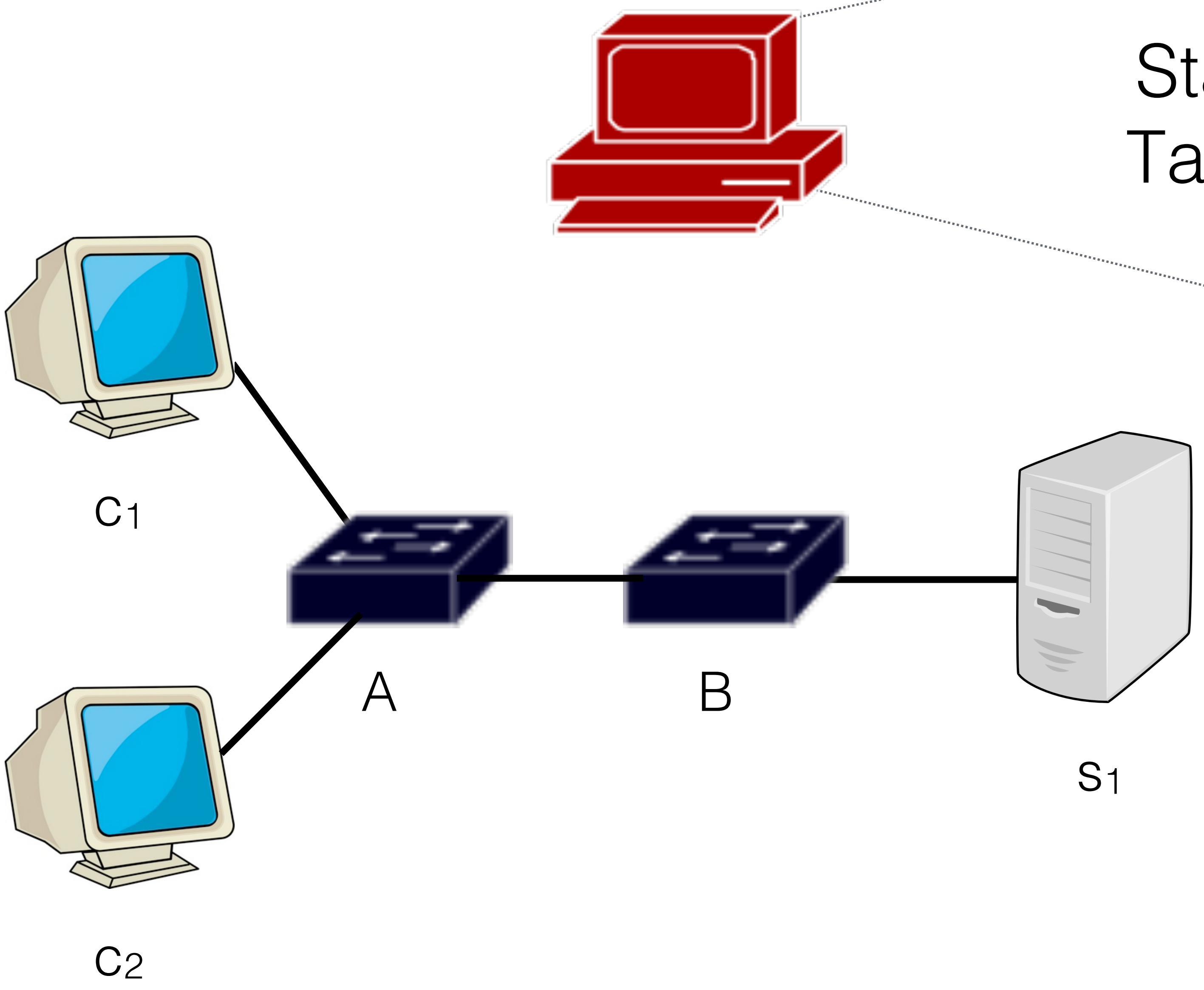
Stateful Firewall



State Table

client	server

Stateful Firewall



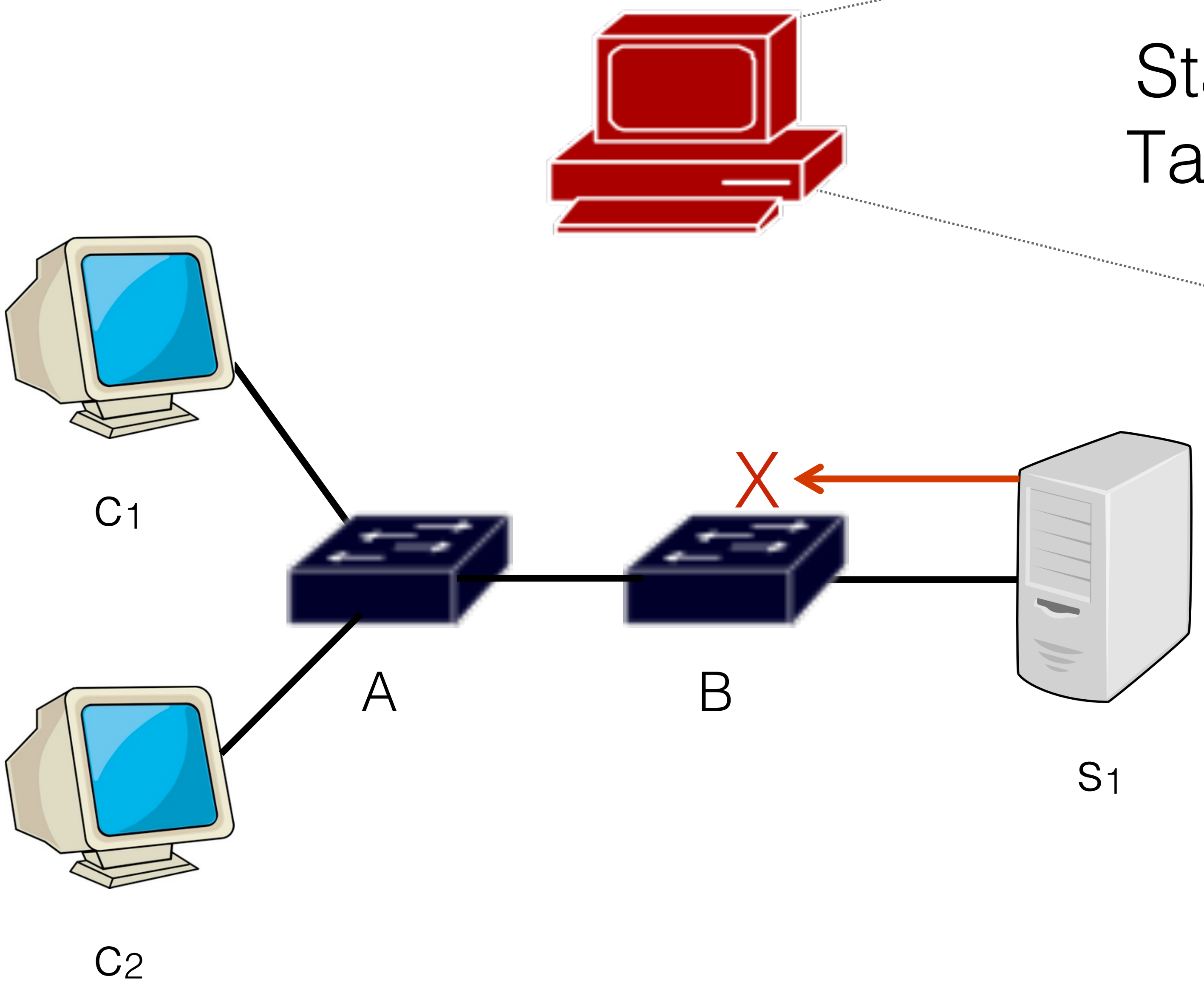
State Table

client	server

<u>Firewall Property:</u>

"A server can only reach a client if the client has communicated with the server"

Stateful Firewall

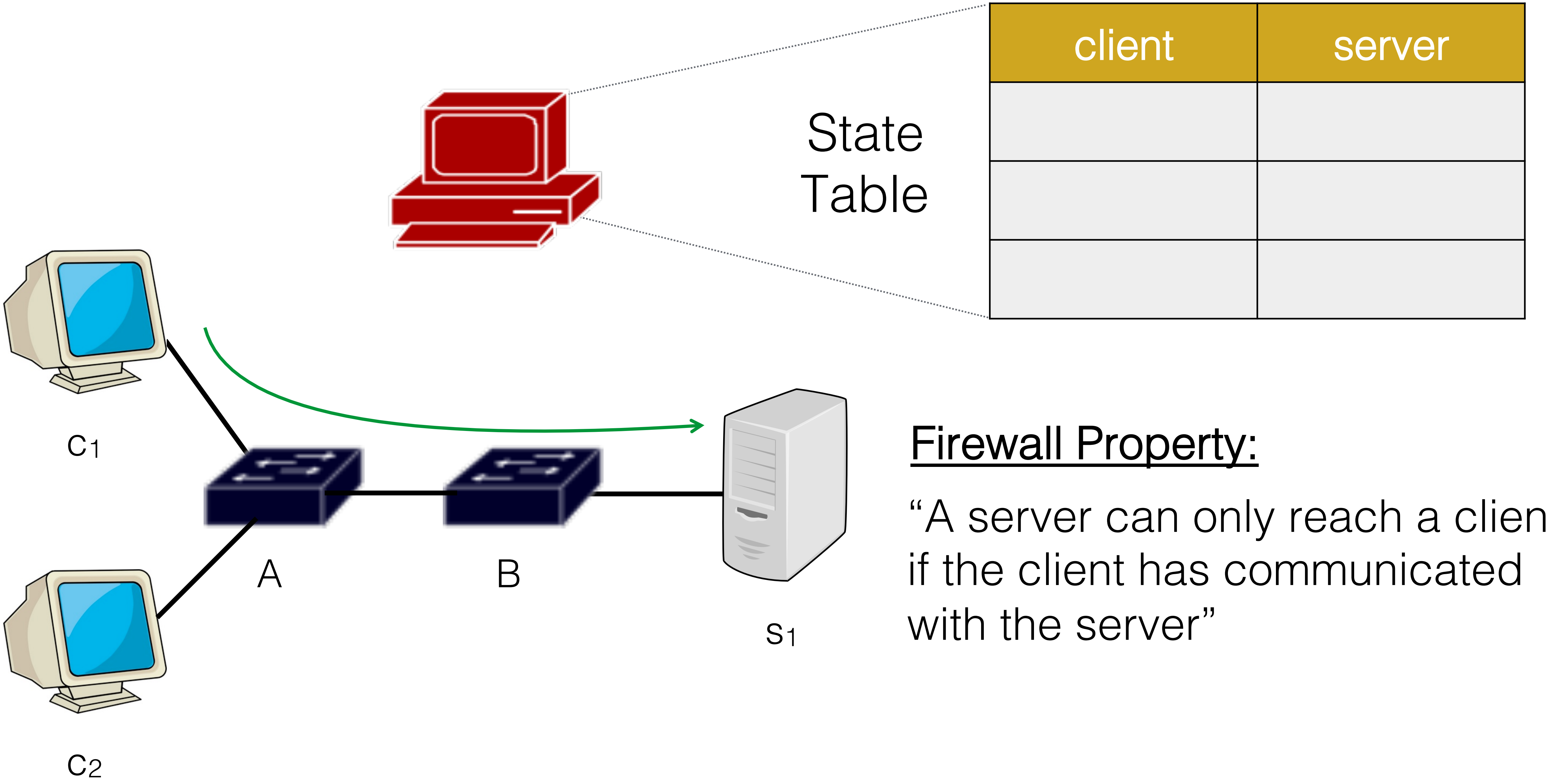


client	server

Firewall Property:

"A server can only reach a client if the client has communicated with the server"

Stateful Firewall

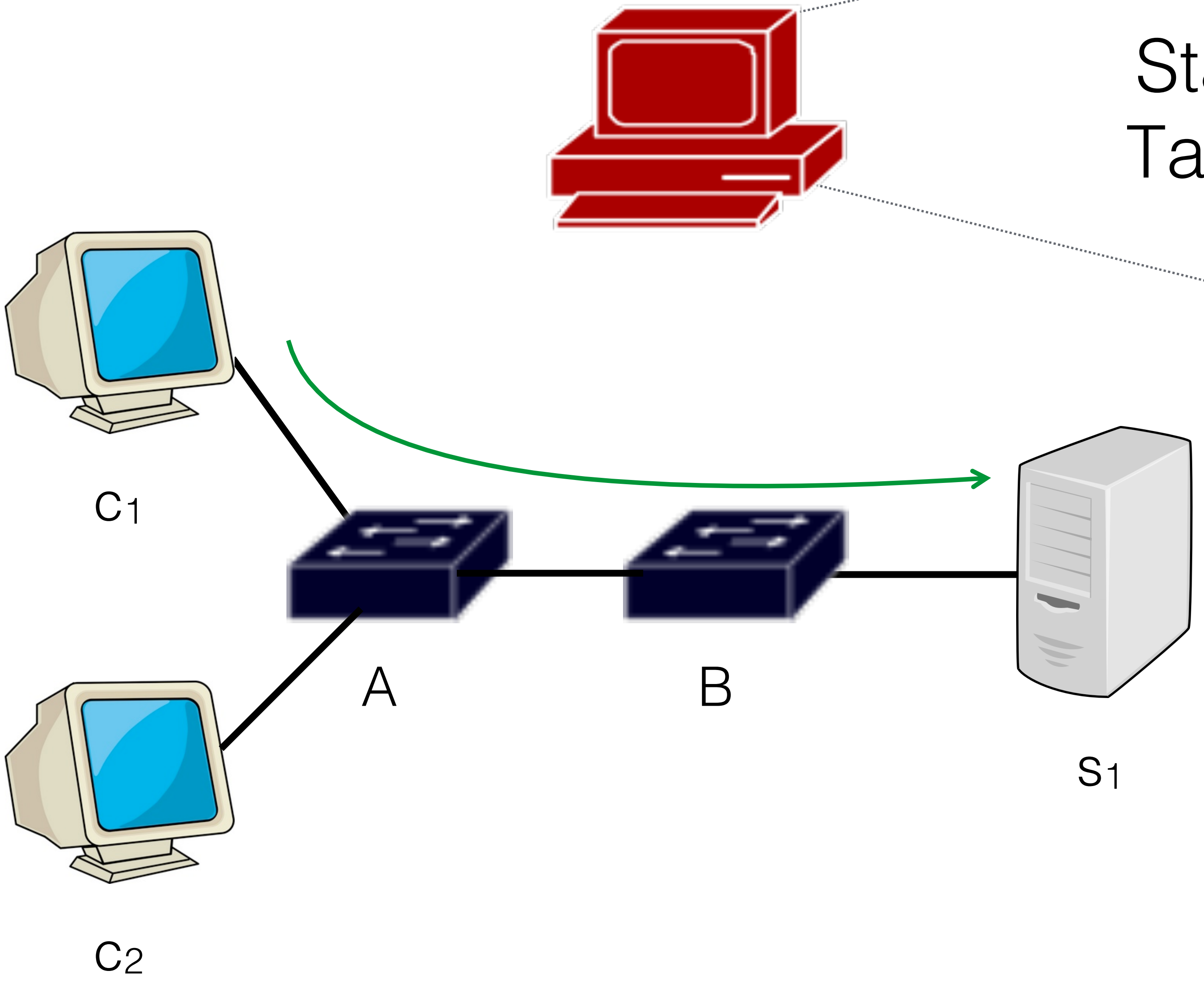


client	server

<u>Firewall Property:</u>

"A server can only reach a client if the client has communicated with the server"

Stateful Firewall



State Table

client	server
c_1	s_1

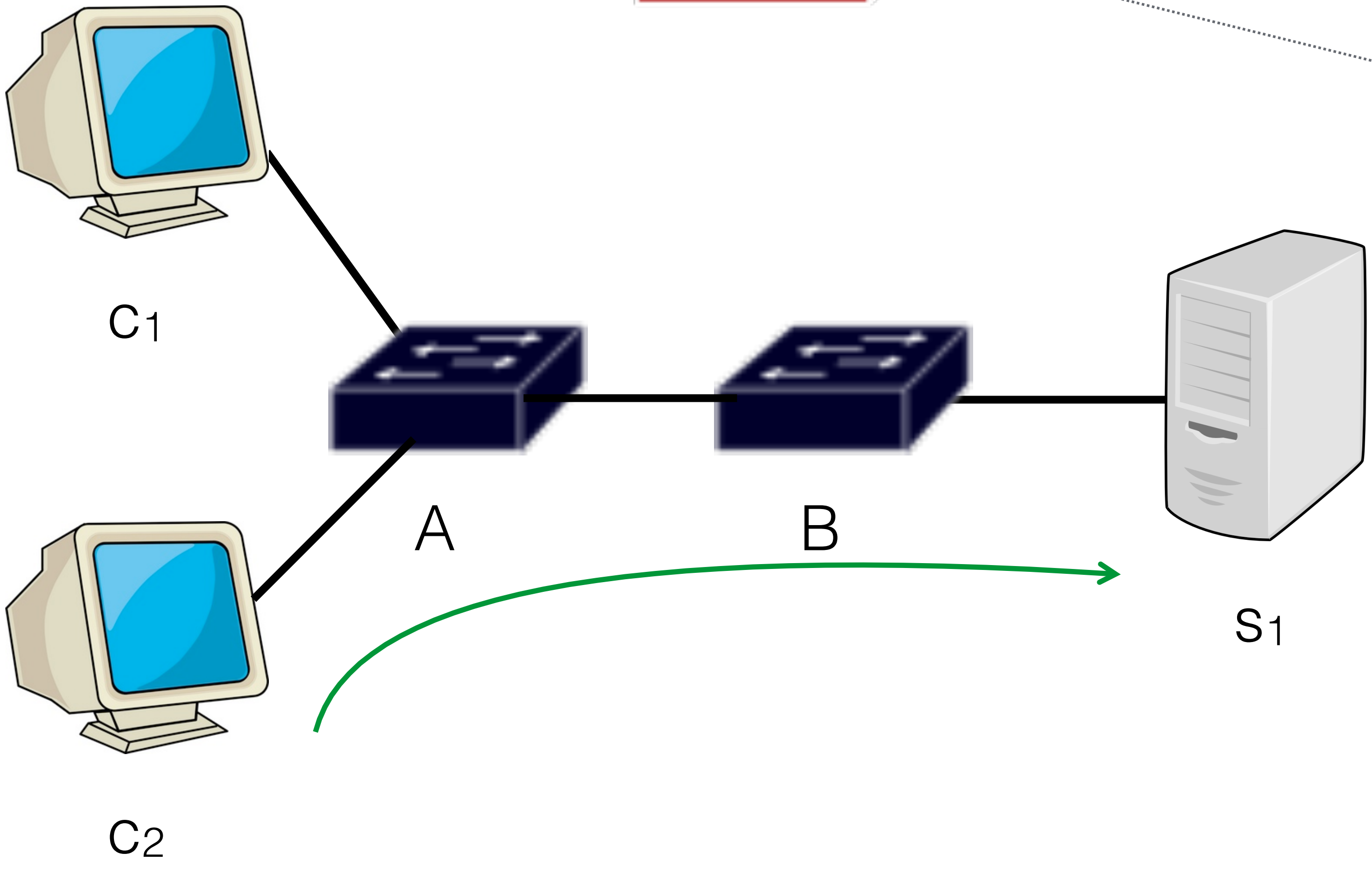
Firewall Property:

reachable(s_1, c_1)

Stateful Firewall

State Table

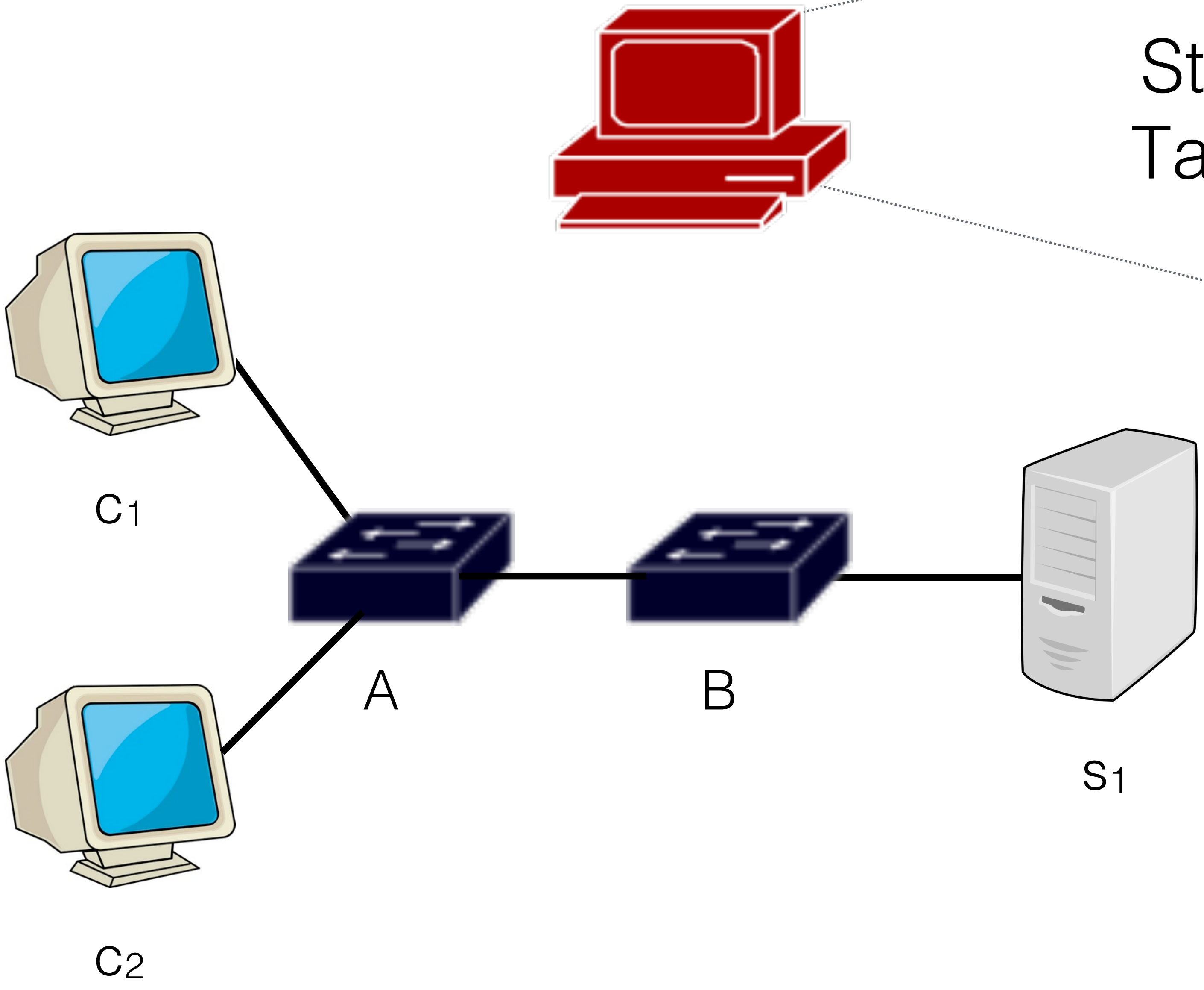
client	server
c_1	s_1
c_2	s_1



Firewall Property:

$\text{reachable}(s_1, c_1) \wedge$
$\text{reachable}(s_1, c_2)$

Stateful Firewall



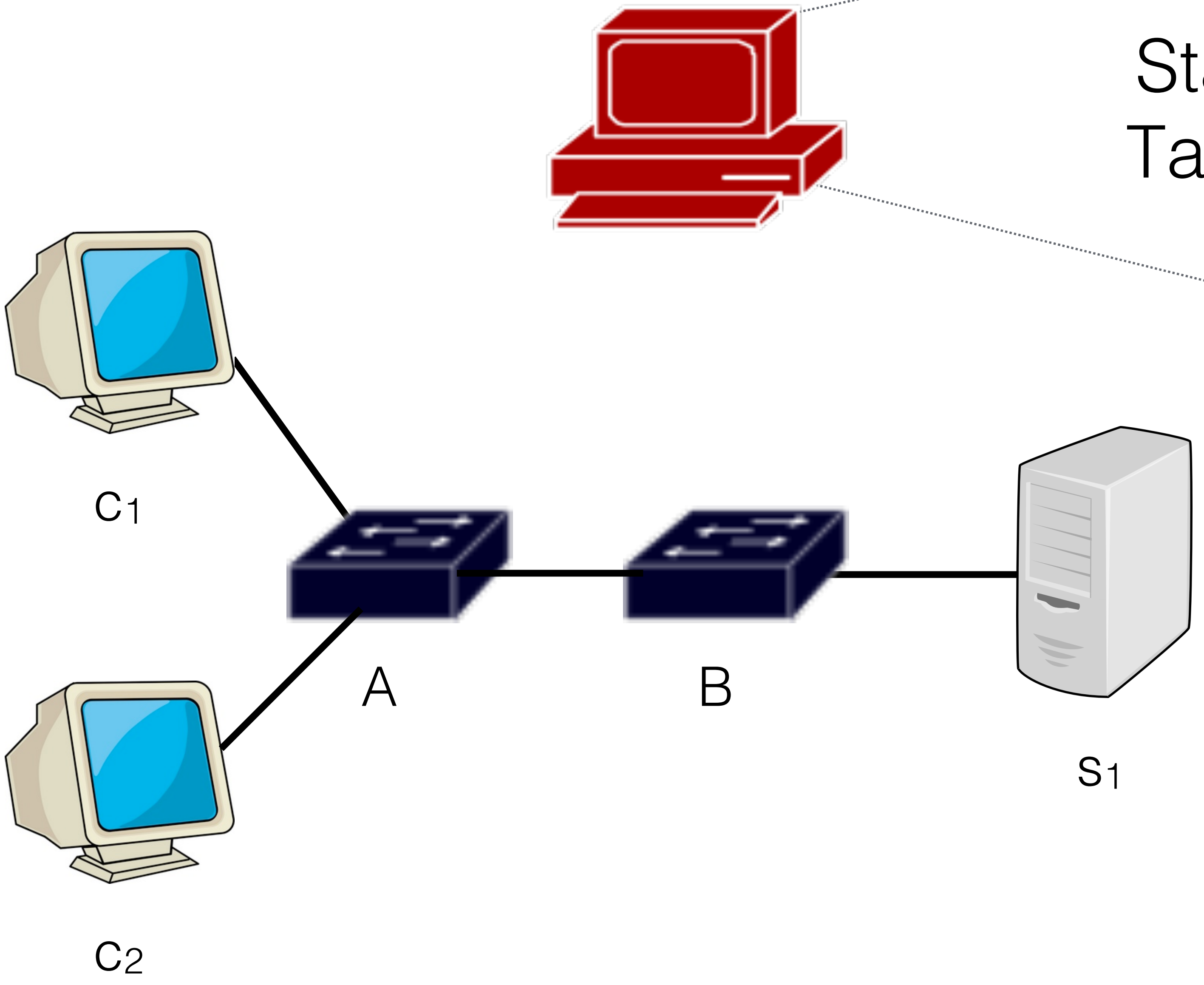
State Table

client	server
c_1	s_1
c_2	s_1

Firewall Property:

$\forall(c,s)\in\text{table}, \text{reachable}(s, c)$

Stateful Firewall



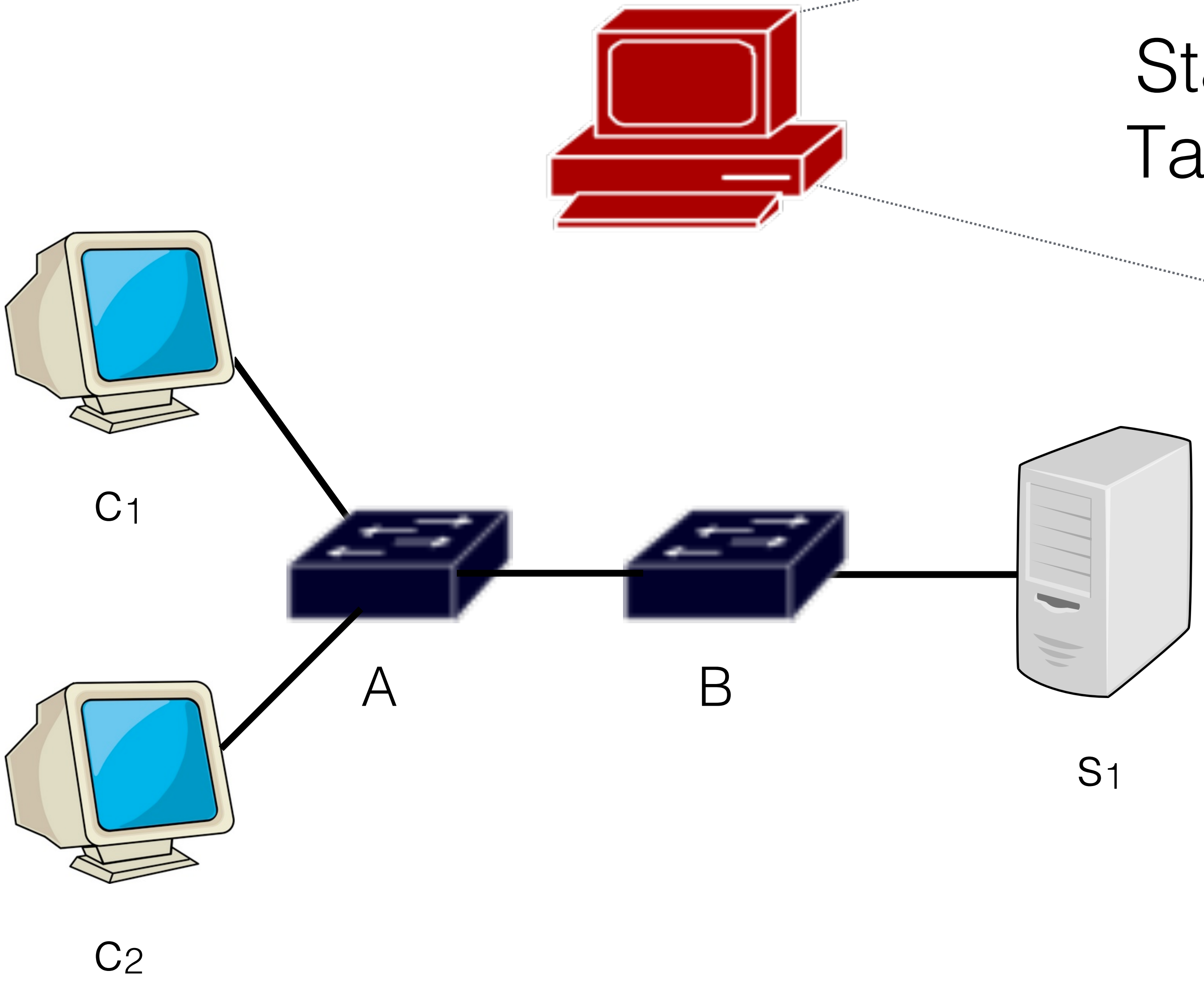
State Table

client	server
c_1	s_1
c_2	s_1

Firewall Property:

$\forall c \in \text{clients}, \forall s \in \text{servers},$
$\quad \text{reachable}(s, c) \leftrightarrow (c,s) \in \text{table}$

Stateful Firewall



State Table

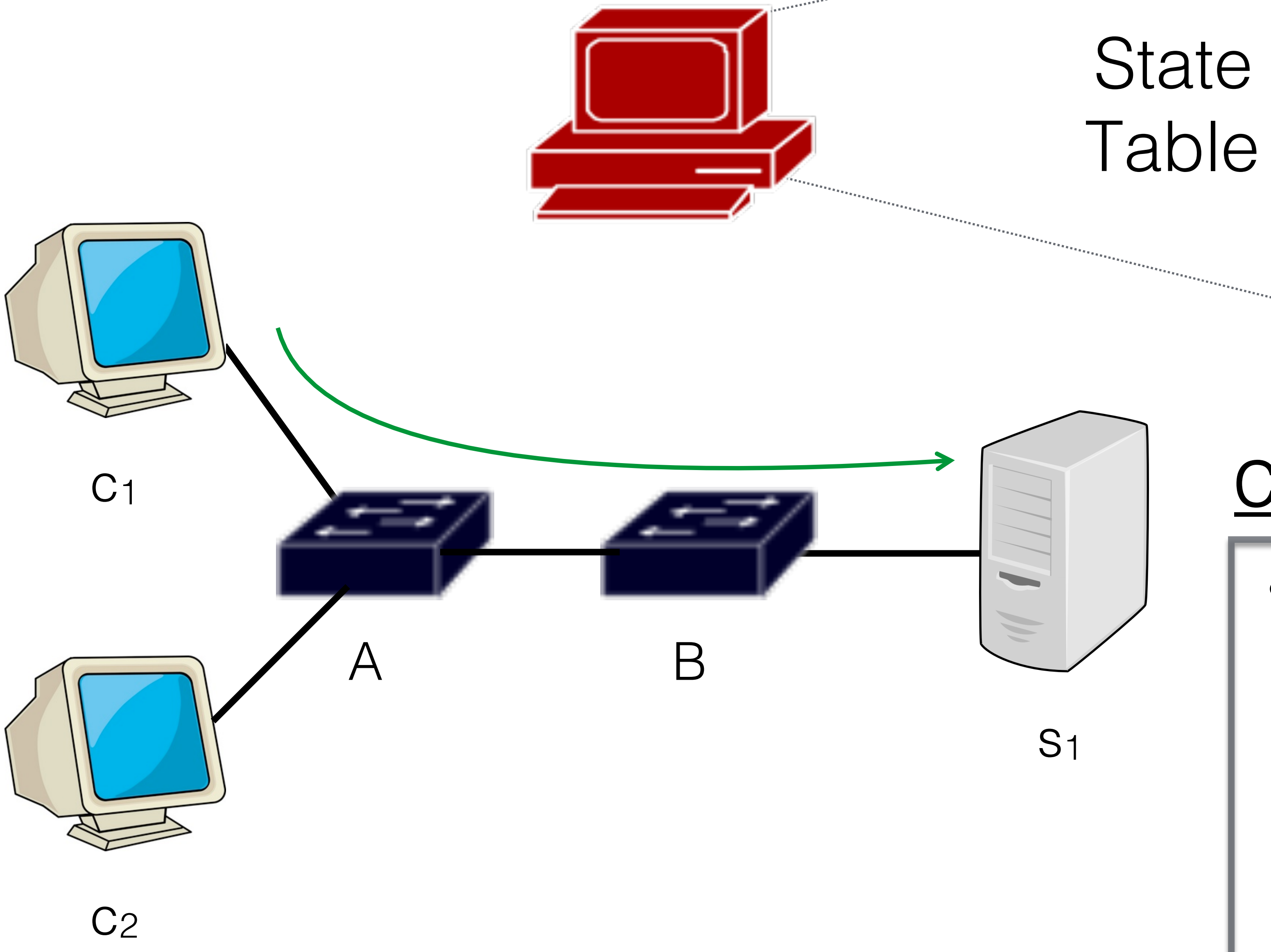
client	server
c_1	s_1
c_2	s_1

Controller Code:

```
assert_continuously(f)

def packet_in(event):
  pkt = event.parsed
  if pkt.typ != eth.IP_TYP:
    return
  …
```

Stateful Firewall

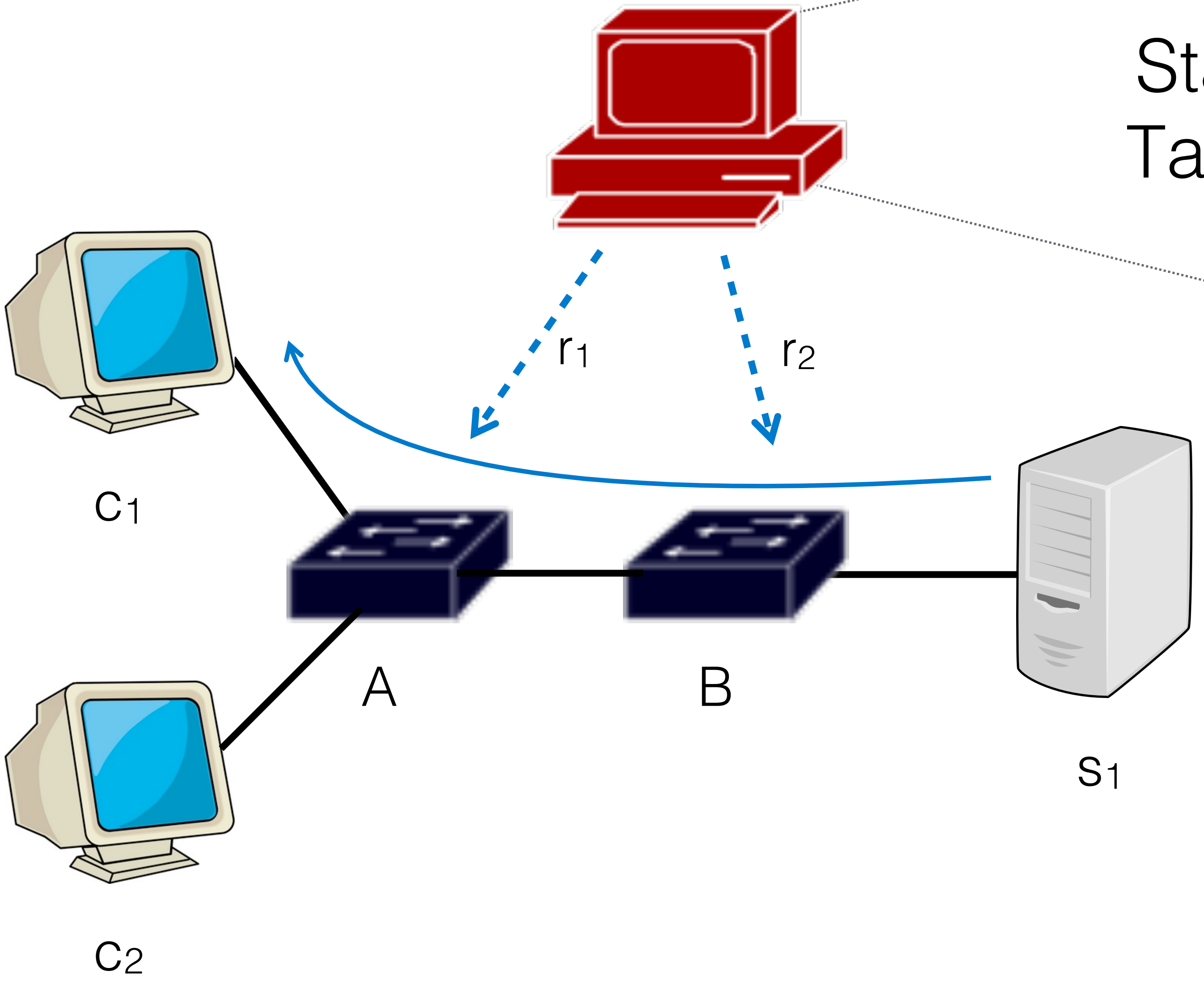


client	server

Controller Code:

```
assert_continuously(f)
```

Stateful Firewall



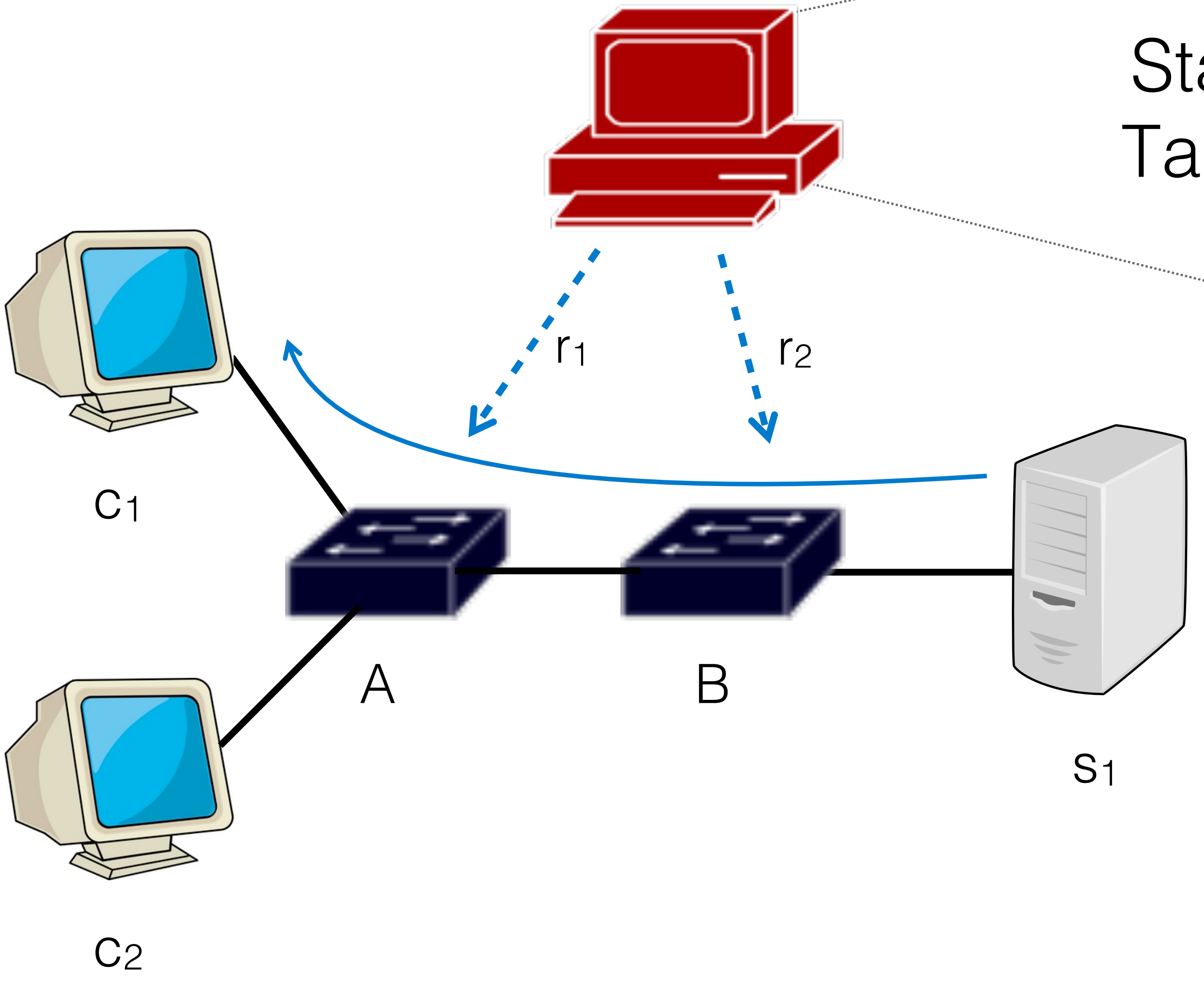
State Table

client	server
c_1	s_1

Controller Code:

```
assert_continuously(f)
```

Stateful Firewall



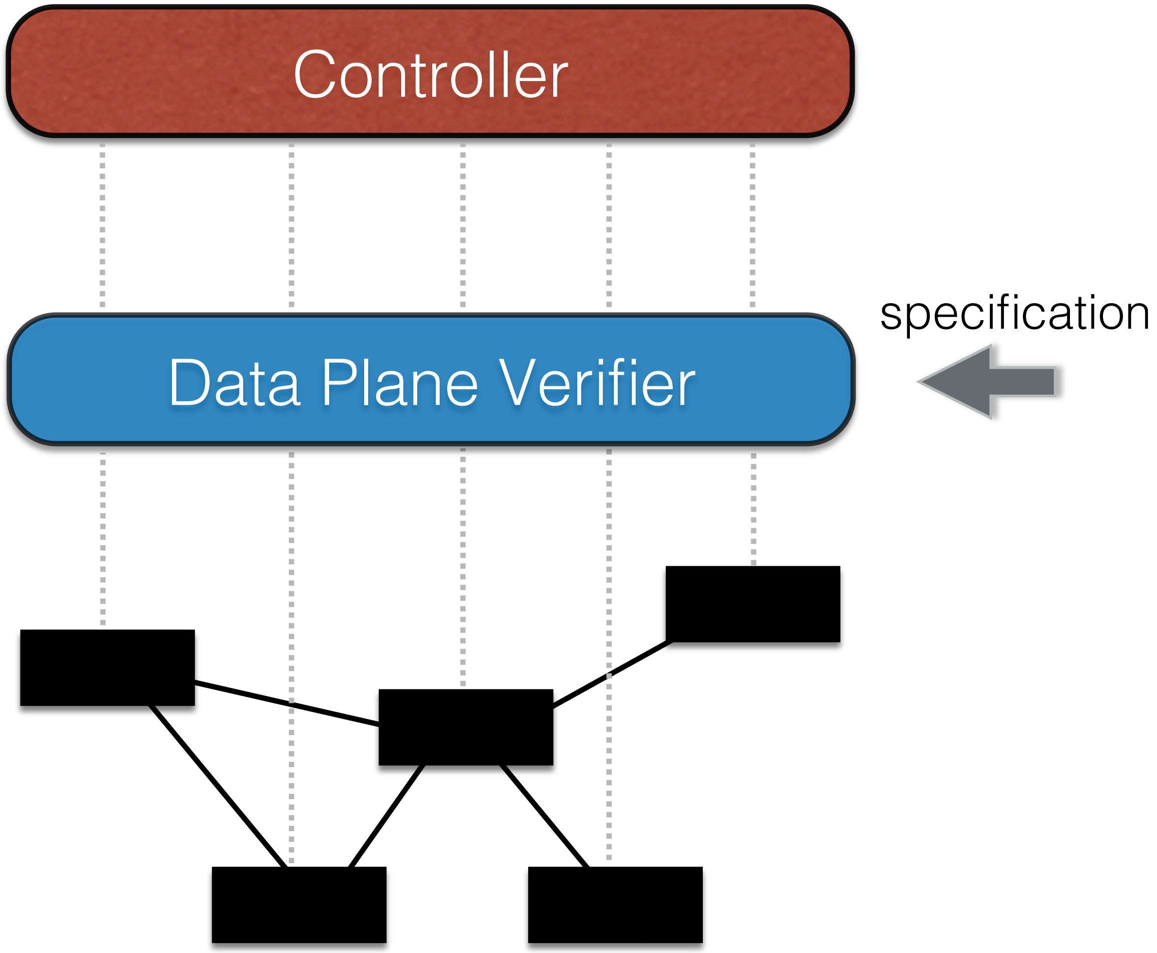
State Table

client	server
c_1	s_1

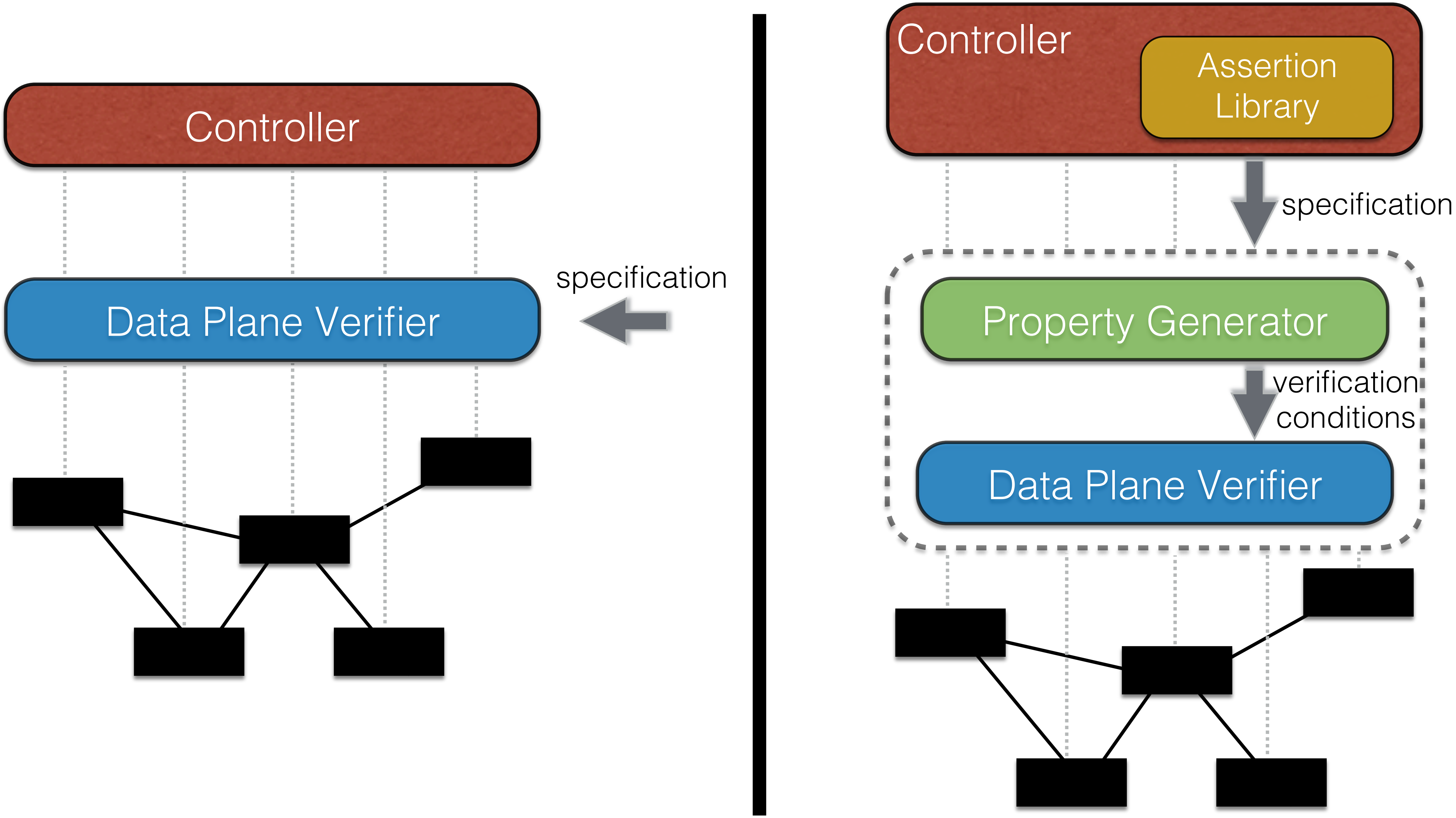
Controller Code:

```
assert_continuously(f)
        ⇣
stop(f)
install(r1)
install(r2)
assert_continuously(f)
```

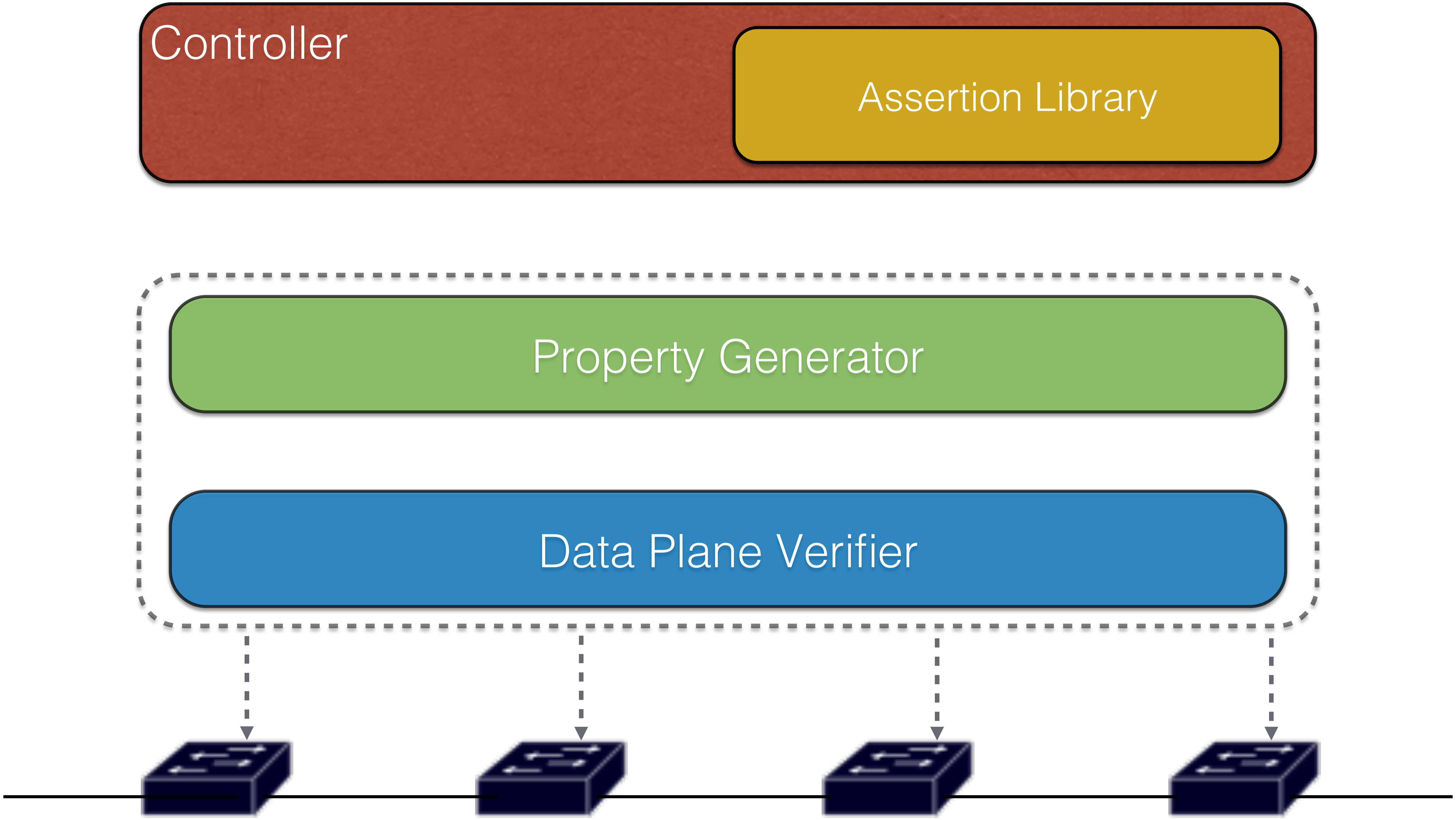
Design Overview



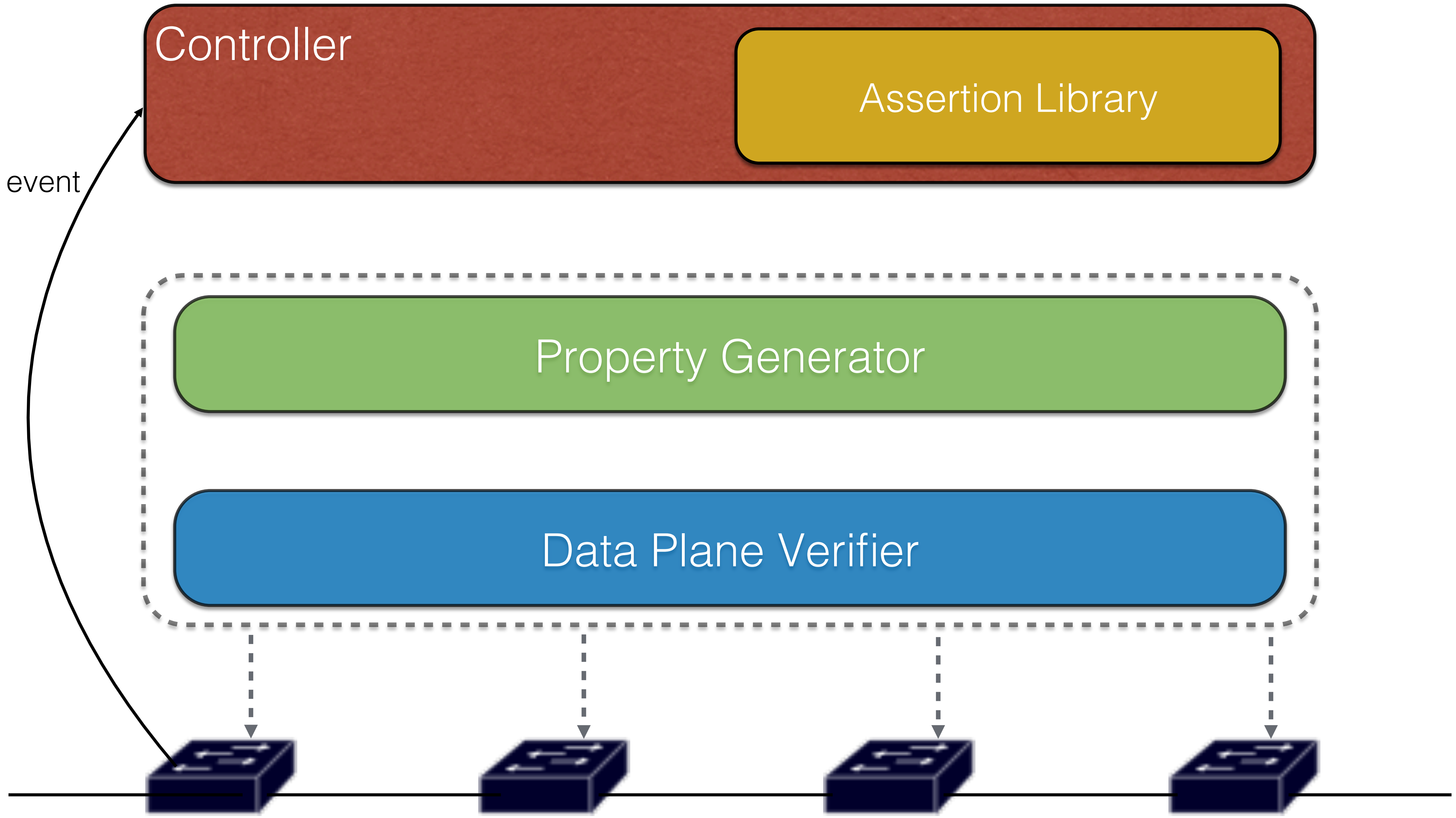
Design Overview



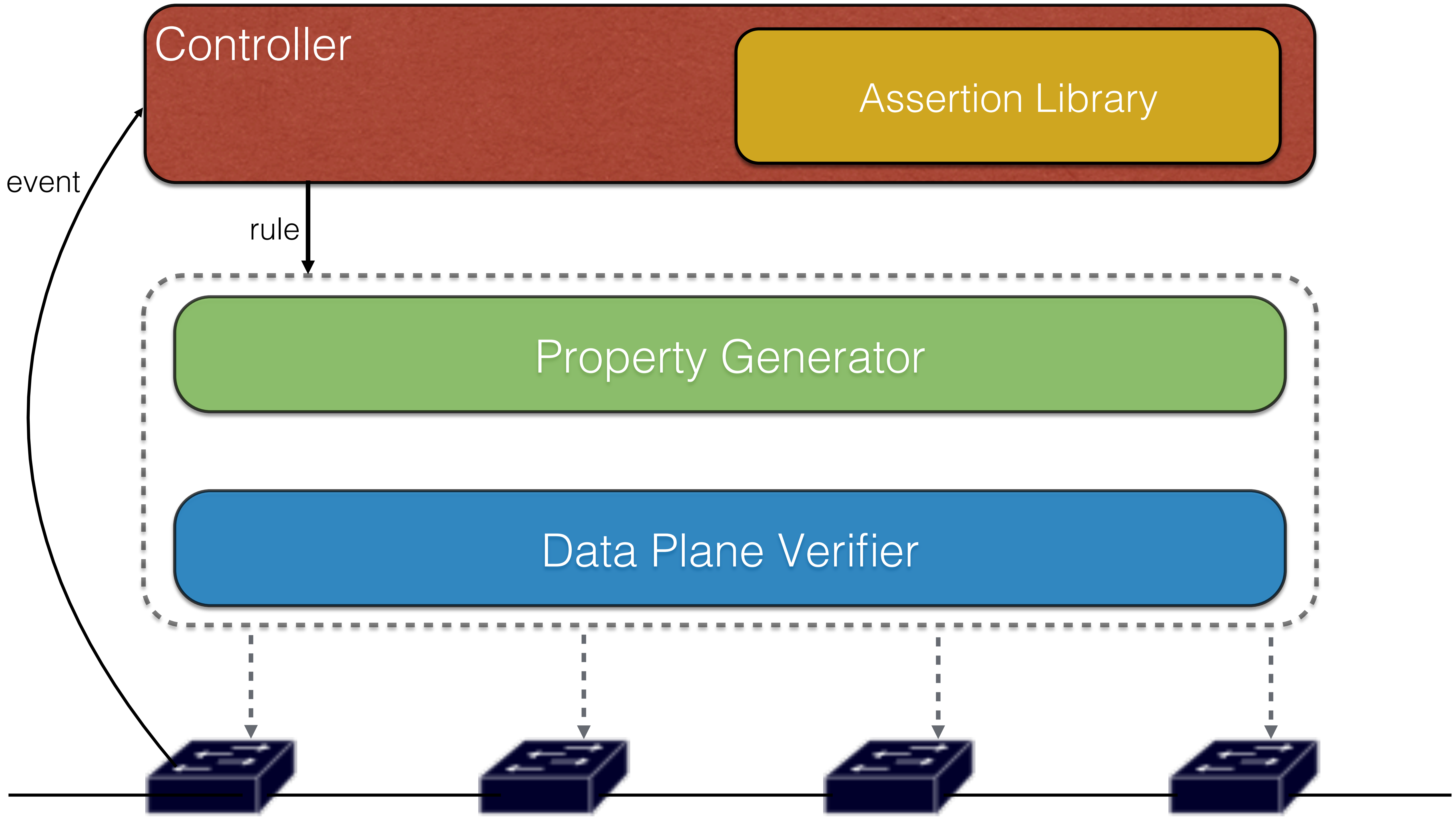
Design Overview



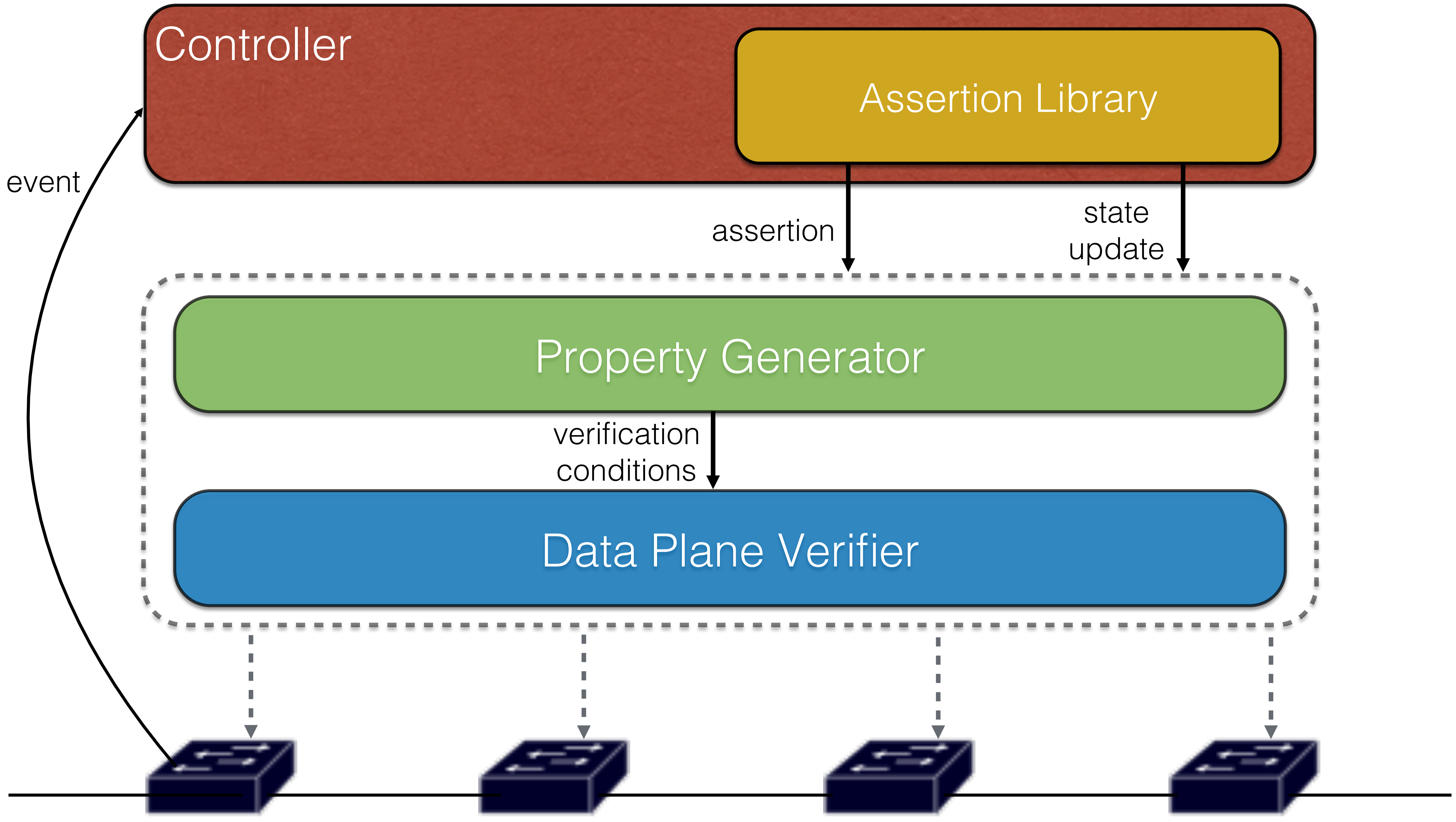
Design Overview



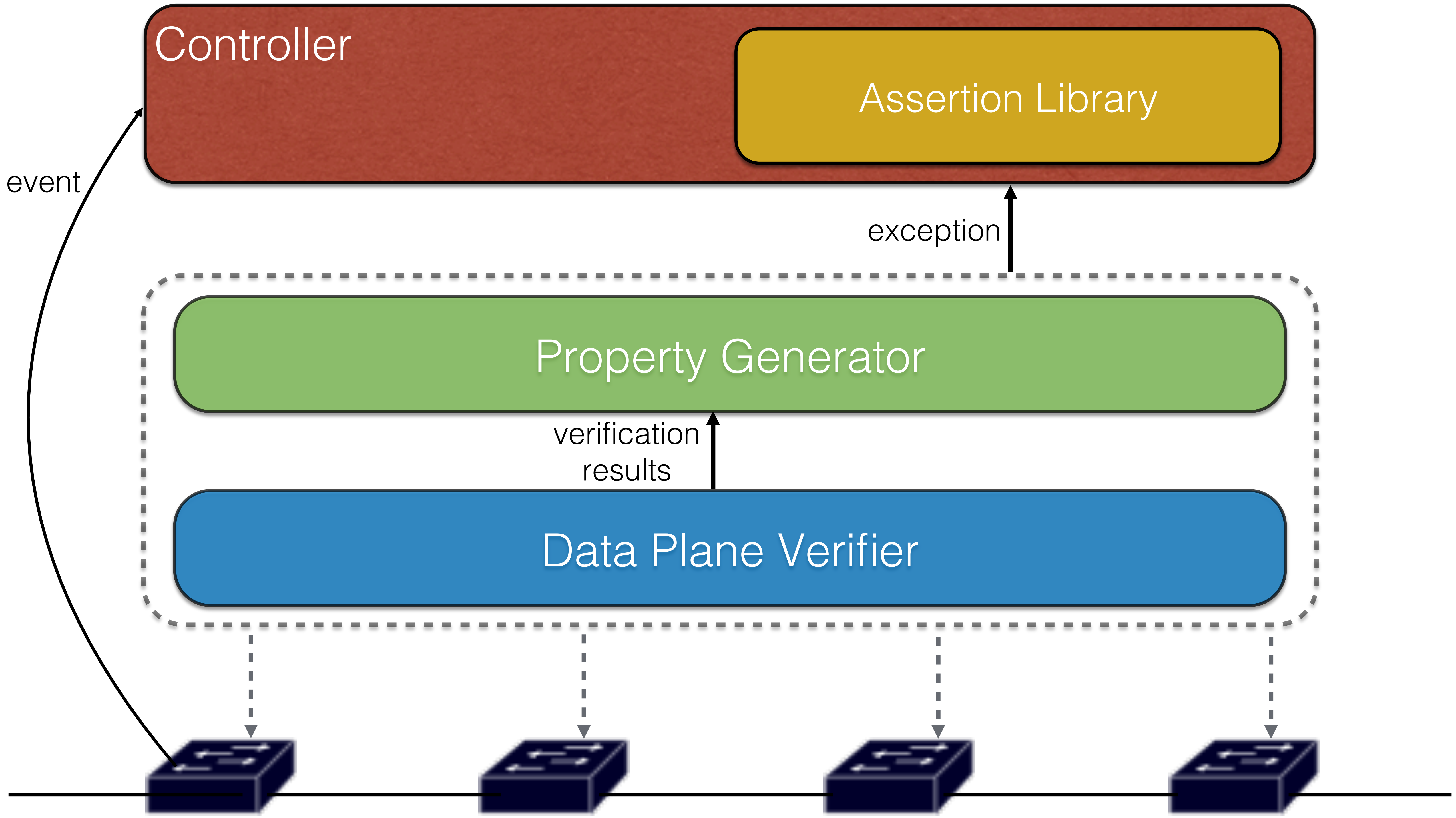
Design Overview



Design Overview



Design Overview



Incremental Verification

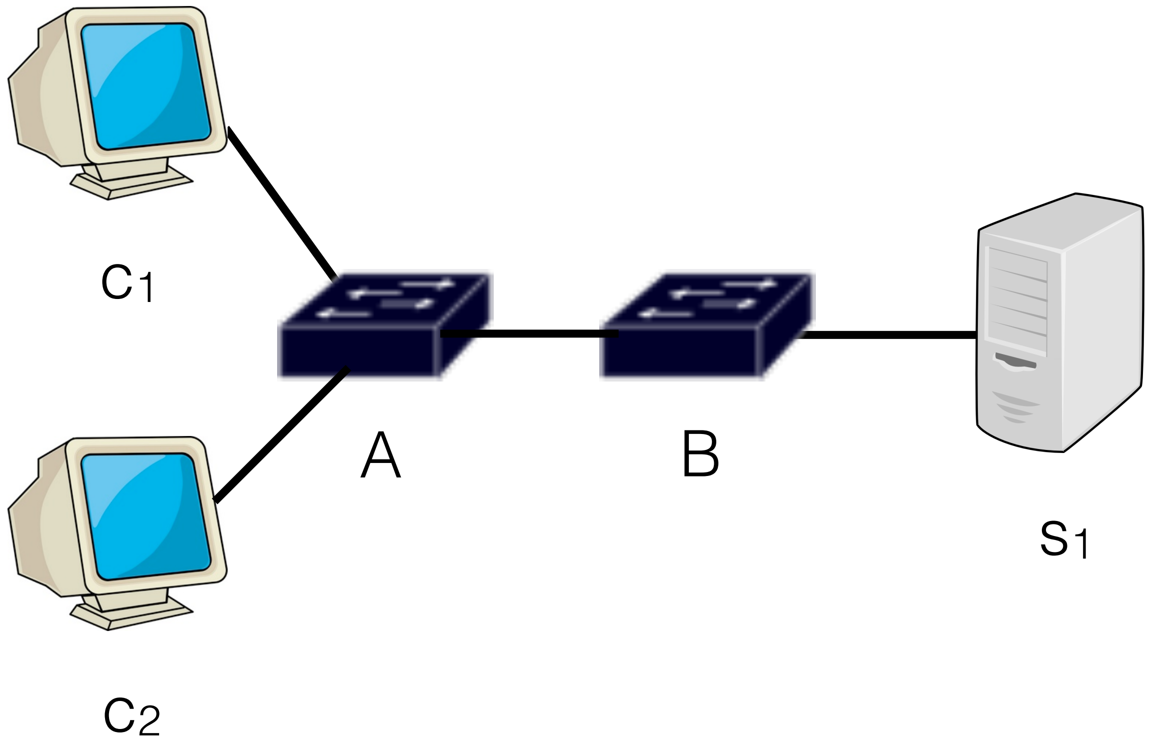
- Change in data plane (existing tools handle this)
- Change in assertion property

Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
    - Incrementally generate new verification conditions

Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
    - Incrementally generate new verification conditions

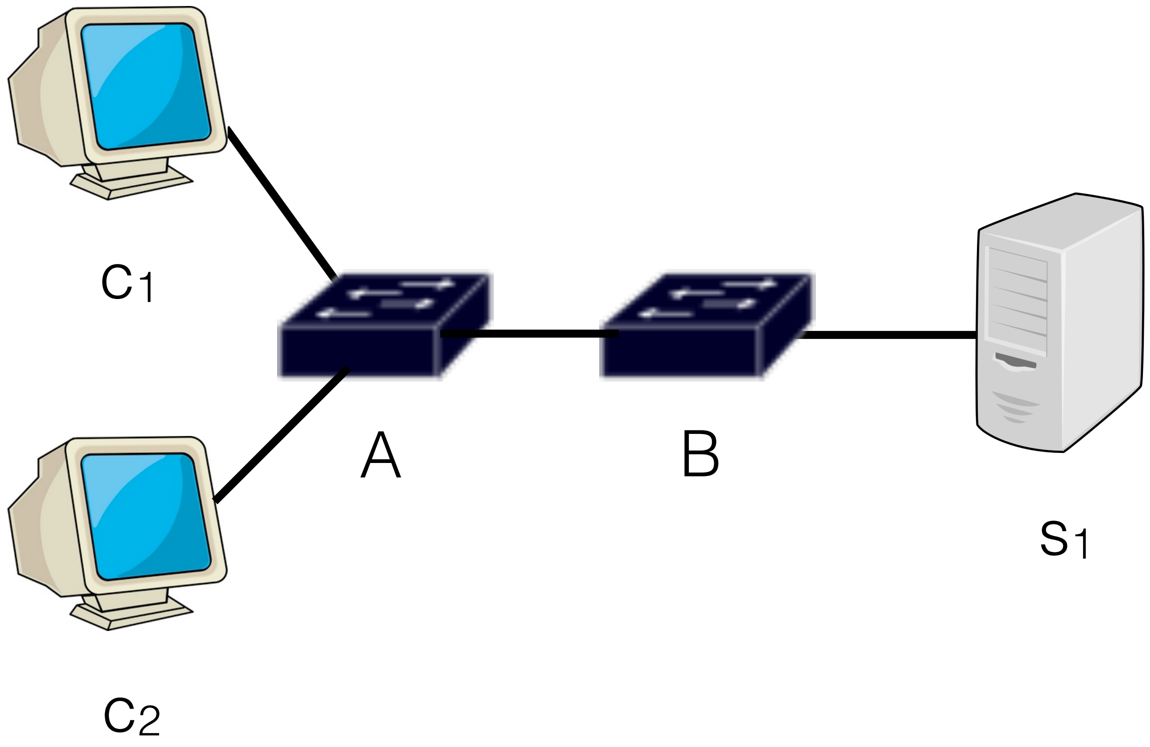


Firewall Property:

reachable(s_1, c_1)

Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
    - Incrementally generate new verification conditions



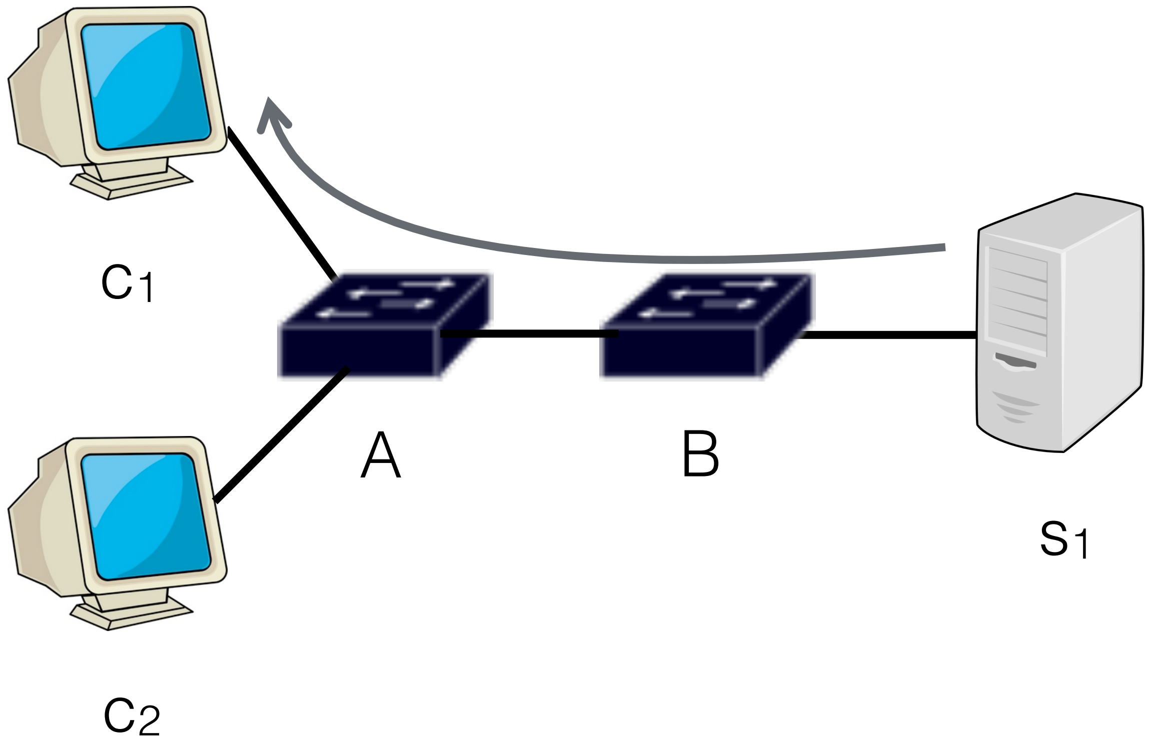
Firewall Property:

reachable(s_1, c_1) ∧

reachable(s_1, c_2)

Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
    - Incrementally generate new verification conditions
    - Precompute and cache intermediate results

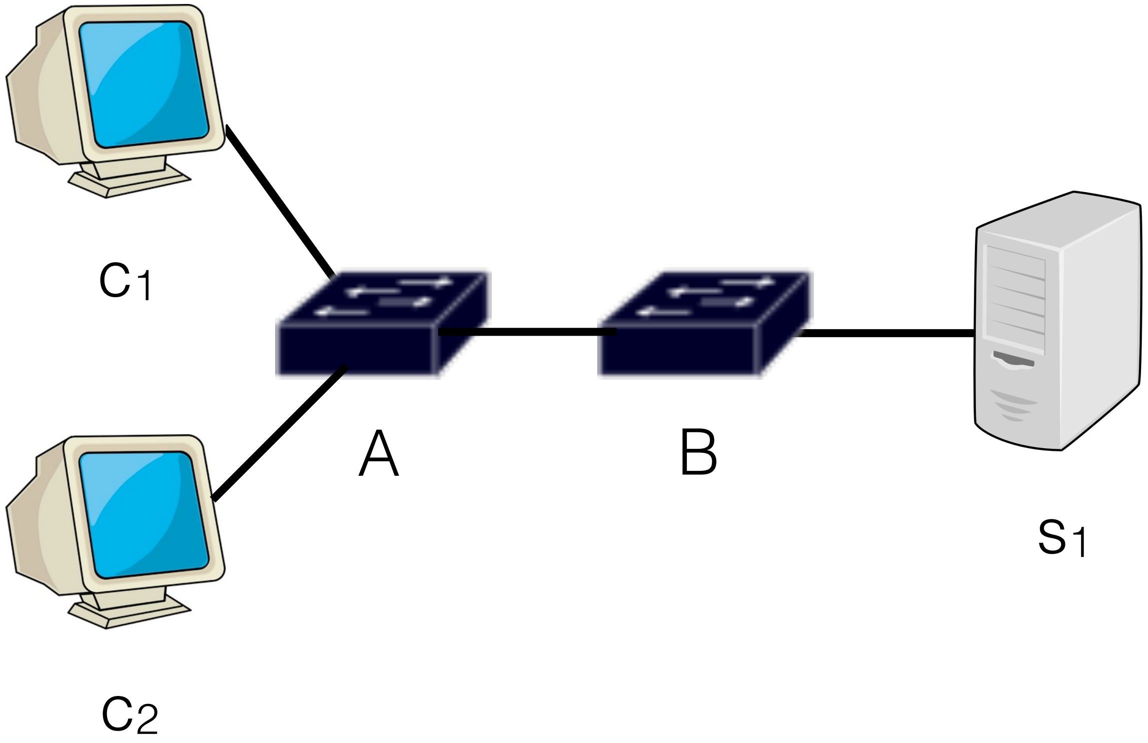


Firewall Property:

reachable(s_1, c_1)

Incremental Verification

- Change in data plane (existing tools handle this)
- Change in assertion property
  - Incrementally generate new verification conditions
  - Precompute and cache intermediate results



Firewall Property:

reachable(s_1, c_1) ∧

reachable(s_1, c_2)

Incremental Data Structure

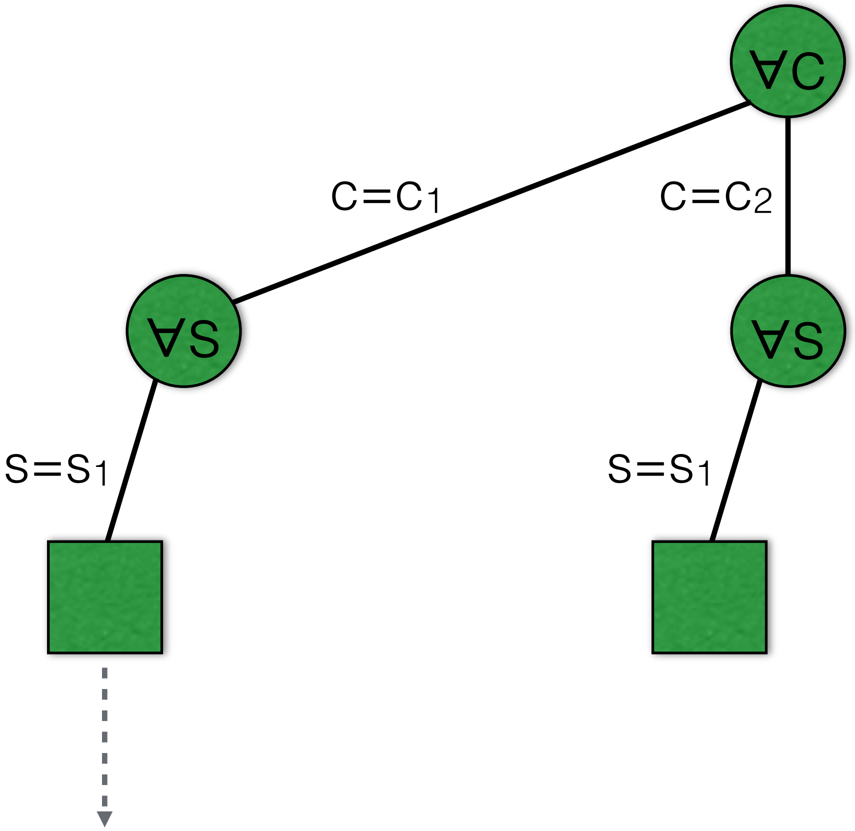
Firewall Property:

$\forall c \in \text{clients}, \forall s \in \text{servers},$

$\quad \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table}$

🟩 True

🟥 False



$\text{reachable}(s_1, c_1) \leftrightarrow (c_1,s_1) \in \text{table}$

Incremental Data Structure

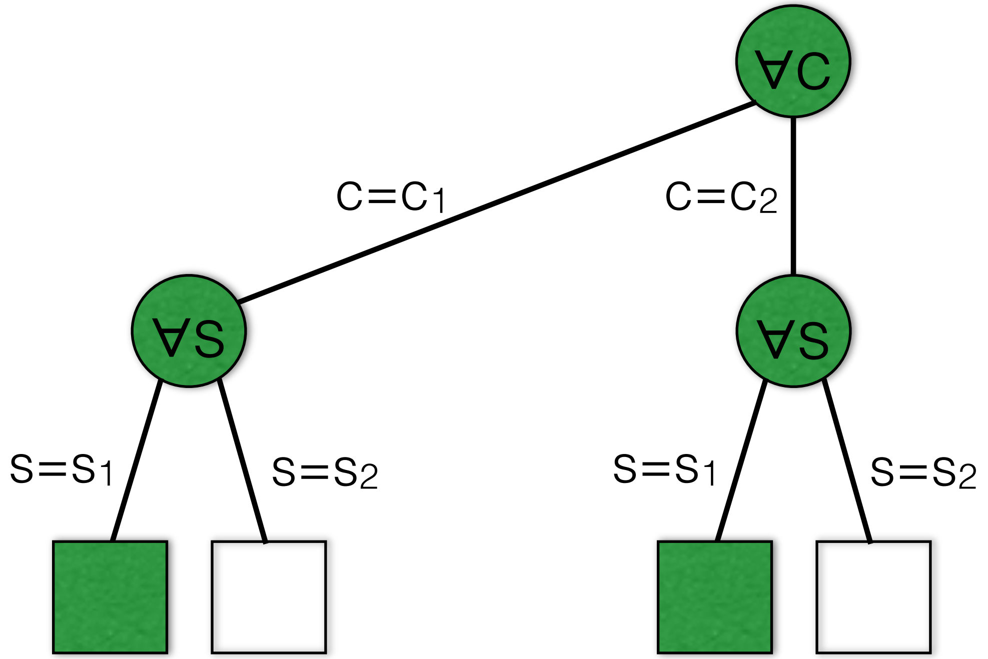
<u>Firewall Property:</u>

$\forall c \in \text{clients}, \forall s \in \text{servers},$

$\quad \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table}$

🟩 True

🟥 False

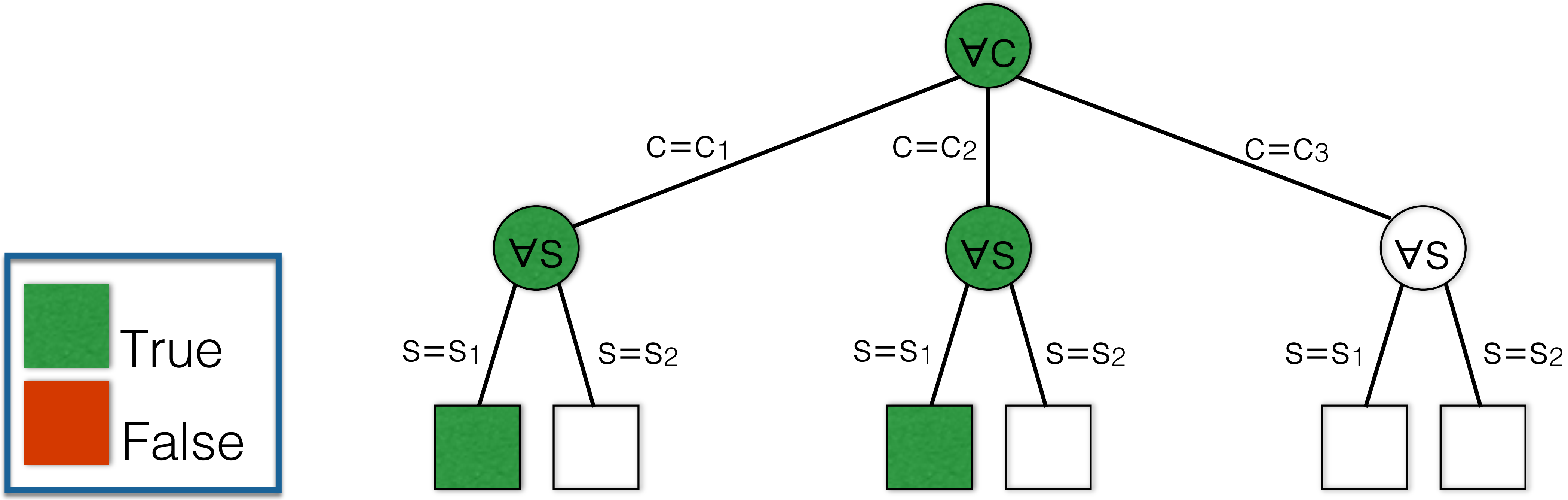


Incremental Data Structure

Firewall Property:

$\forall c \in \text{clients}, \forall s \in \text{servers},$

$\quad \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table}$

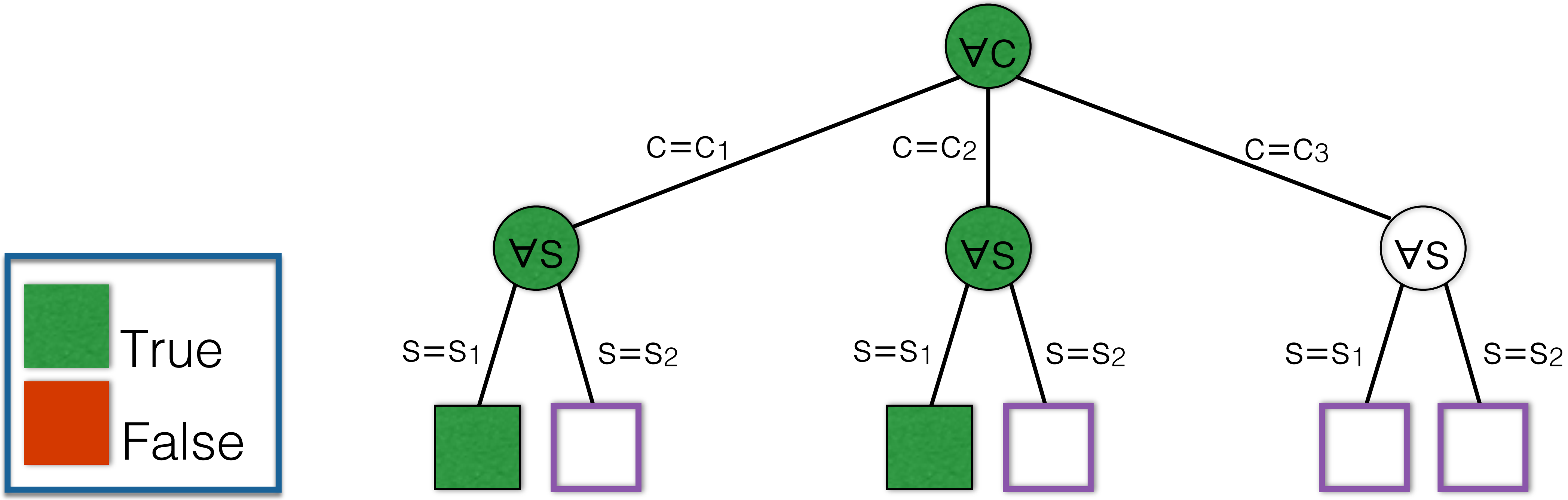


Incremental Data Structure

Firewall Property:

$\forall c \in \text{clients}, \forall s \in \text{servers},$

$\quad \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table}$

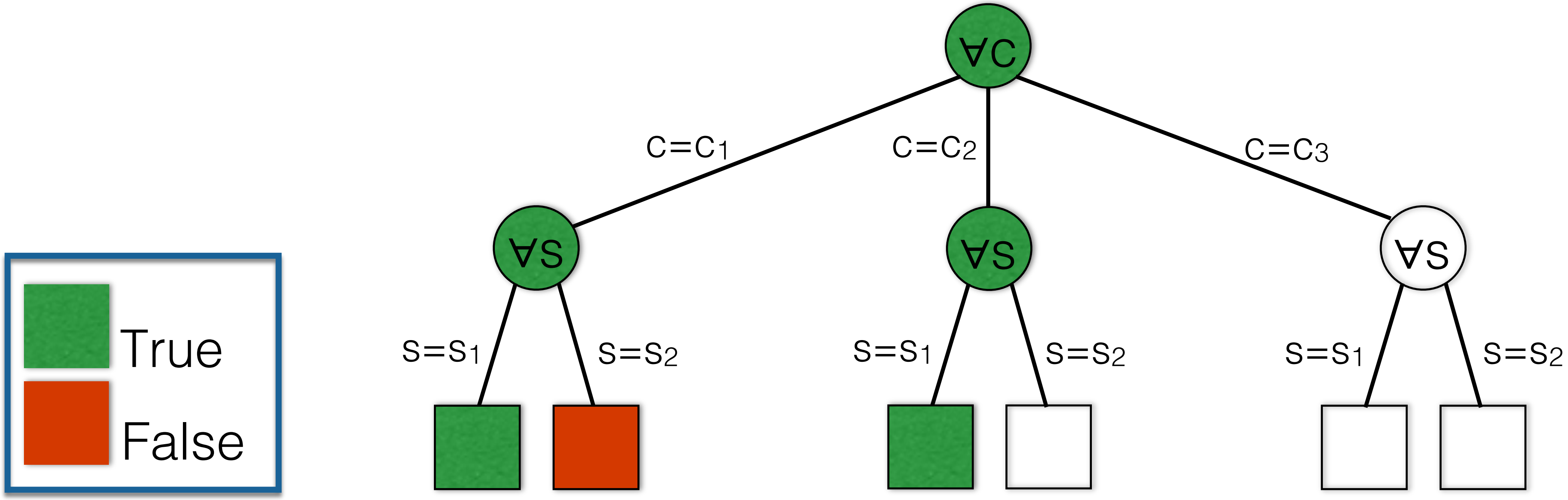


Query data-plane verifier

Incremental Data Structure

Firewall Property:

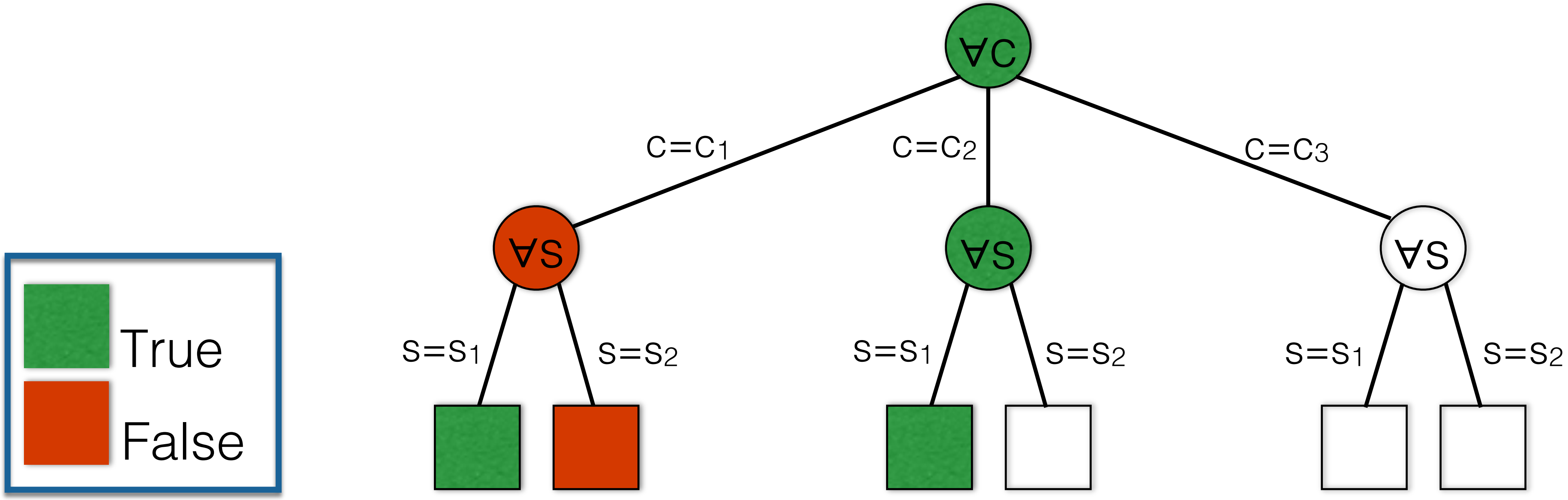
∀c∈clients, ∀s∈servers,
  reachable(s,c) ↔ (c,s)∈table



Incremental Data Structure

Firewall Property:

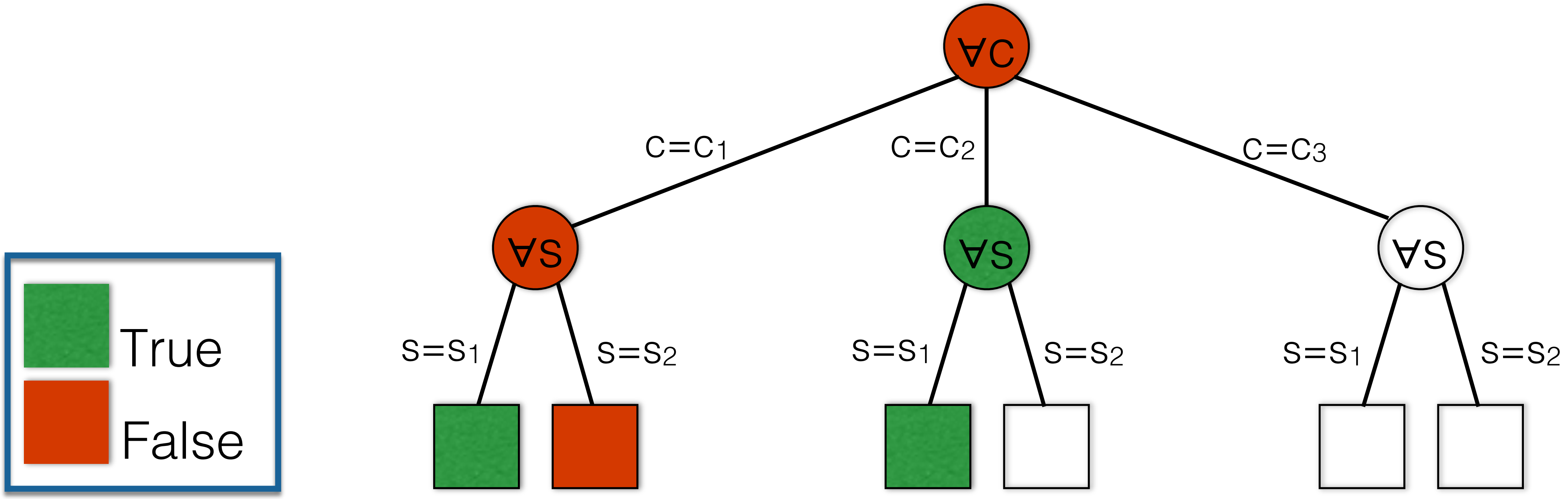
$\forall c \in \text{clients}, \forall s \in \text{servers},$
$\quad \text{reachable}(s,c) \leftrightarrow (c,s) \in \text{table}$



Incremental Data Structure

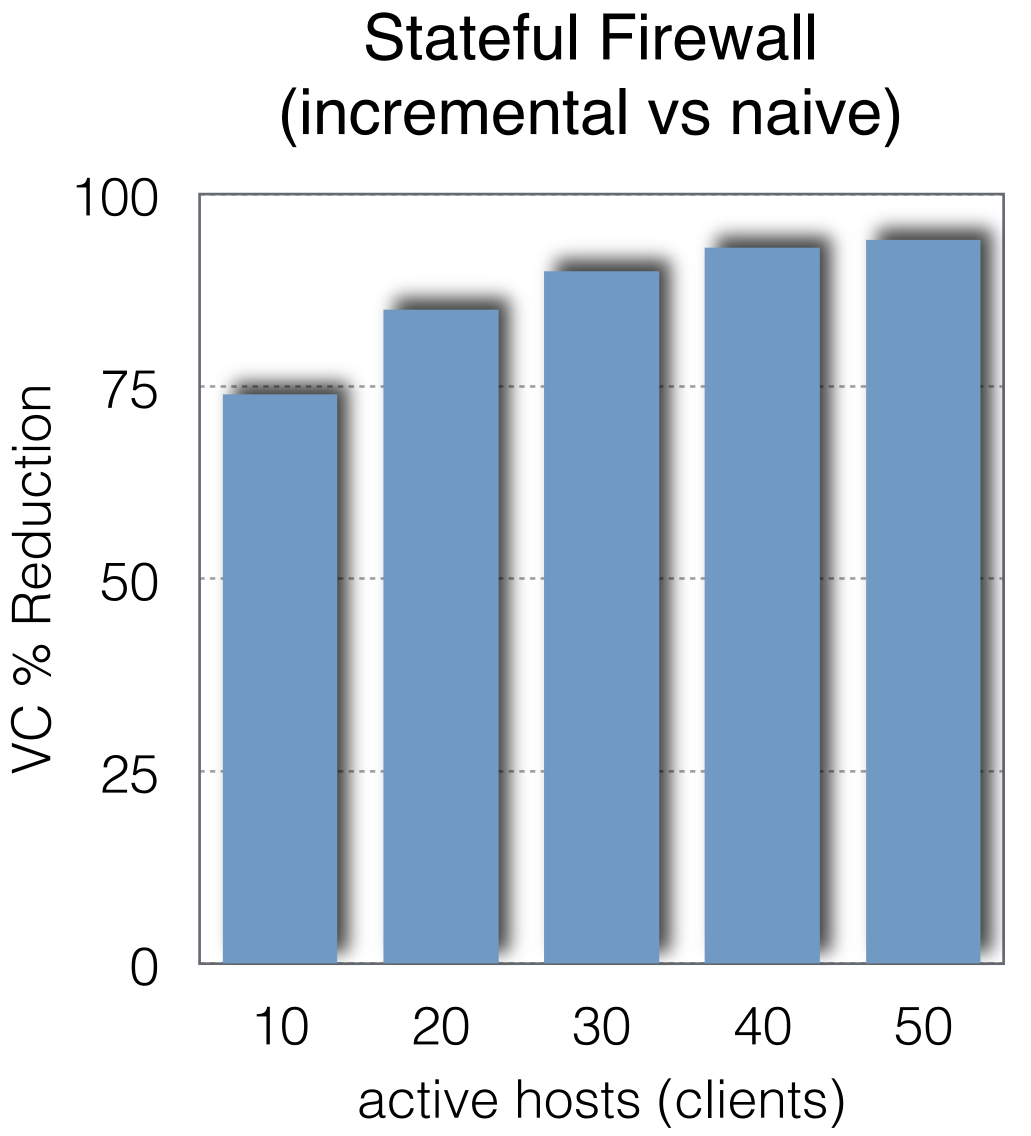
Firewall Property:

$\forall c \in clients, \forall s \in servers,$
$\quad reachable(s,c) \leftrightarrow (c,s) \in table$



Prototype Implementation

- Python assertion debugging library
- Support for Pyretic, Pox, Ryu
- Uses the VeriFlow verification tool
- Initial performance is promising



Conclusion

- Assertions to verify dynamic properties
- Programmatic control over verification timing
- Incremental algorithm to verify dynamic assertion properties
- Prototype with reasonable performance

Q&A