ReAct: Reflection Attack Mitigation For Asymmetric Routing

David Hay*, **Mary Hogan**^, Shir Landau Feibish+

*Hebrew University, ^Oberlin College, +University of Haifa

OBERLIN

COLLEGE & CONSERVATORY

DDoS attacks cause significant service disruptions

Keymous+ Hacker Group Claims 700+ DDoS Attacks Around The Globe

By **Tushar Subhra Dutta** July 3, 2025

Mastodon says its flagship server was hit by a DDoS attack

Zack Whittaker, Sarah Perez

9:58 AM PDT · April 20, 2026

DDoS attacks are outpacing networks as attack sizes rise by almost 70%

DNS and TCP-based attacks also jump as threat actors get more sophisticated

June 23, 2025 By: Ben Wodecki Have your say

Massive 7.3 Tbps DDoS Attack Delivers 37.4 TB in 45 Seconds, Targeting Hosting Provider

Ravie Lakshmanan Jun 20, 2025

Cyber Attack / Botnet

Bluesky blames DDoS attack for server outages

A couple of servers were down on Thursday morning.

Kris Holt
Contributing Reporter

Add Engadget on Google

Updated Fri, April 17, 2026 at 12:47 PM EDT

Hyper-volumetric DDoS attacks skyrocket: Cloudflare's 2025 Q2 DDoS threat report

2025-07-15

Omer Yoachimik

Jorge Pacheco

Amplified Reflection DDoS (AR-DDoS)

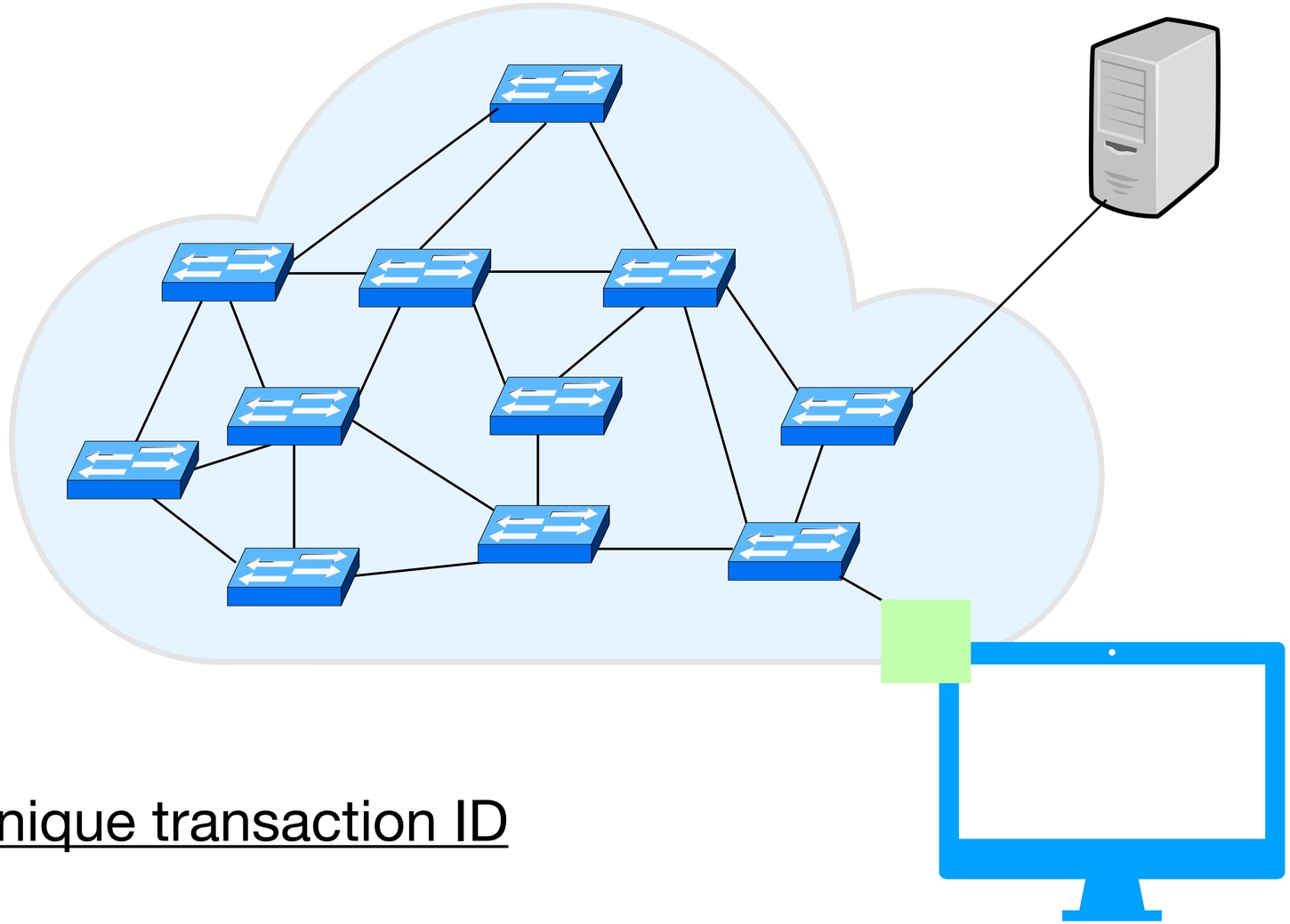
AR-DDoS attacks turn servers into *reflectors* that *amplify* the amount of traffic sent by the attackers

Attacks typically work on transaction-based protocols in which servers respond to client requests over UDP (e.g., DNS, NTP, etc.)

Existing in-network defenses use transaction IDs in packet headers to match legitimate requests and responses

DNS Requests

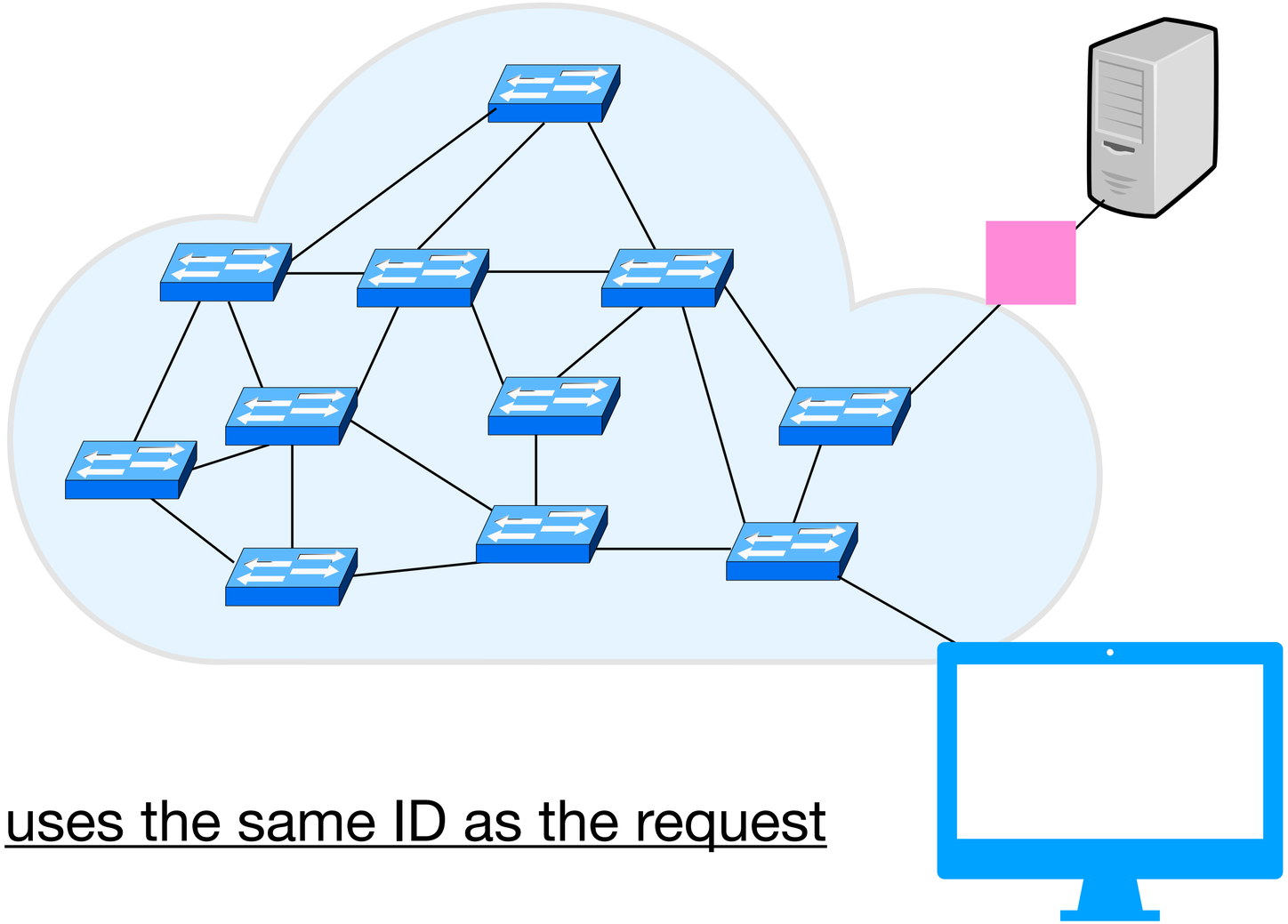
```
+--+--+--+--+--+--+--+--+--+--+
|           ID = 5             |
+--+--+--+--+--+--+--+--+--+--+
|                 …            |
+--+--+--+--+--+--+--+--+--+--+
|                 …            |
+--+--+--+--+--+--+--+--+--+--+
|                 …            |
 +-+--+--+--+--+--+--+--+--+--+
```



DNS request header includes a unique transaction ID

DNS Responses

```
+--+--+--+--+--+--+--+--+--+--+--+--+
|              ID = 5               |
+--+--+--+--+--+--+--+--+--+--+--+--+
|                   …               |
+--+--+--+--+--+--+--+--+--+--+--+--+
|                   …               |
+--+--+--+--+--+--+--+--+--+--+--+--+
|                   …               |
 +-+--+--+--+--+--+--+--+--+--+--+--+
```

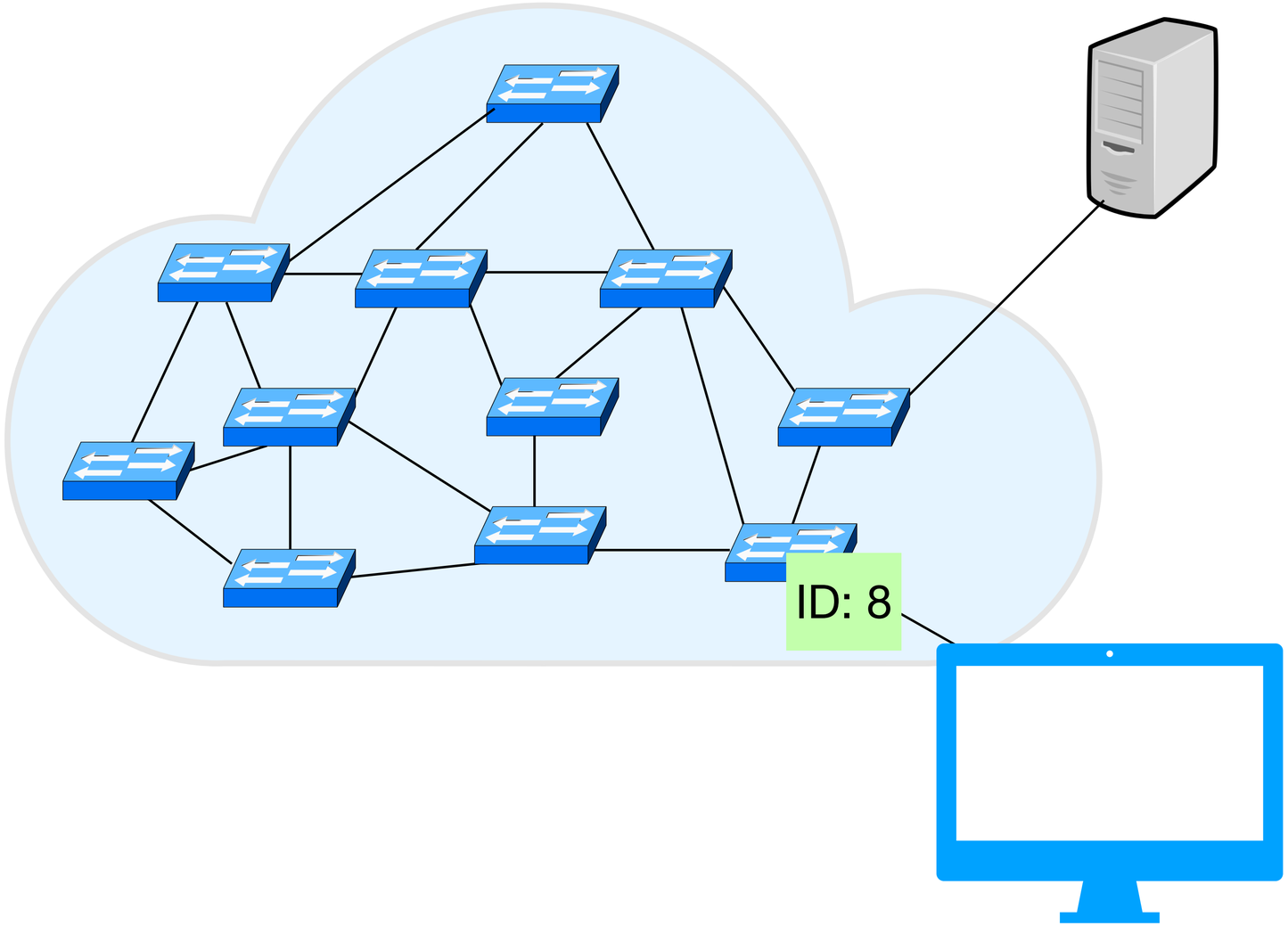


DNS response header includes a uses the same ID as the request

Strawman AR-DDoS Defense

We can track the number of times we've seen an ID

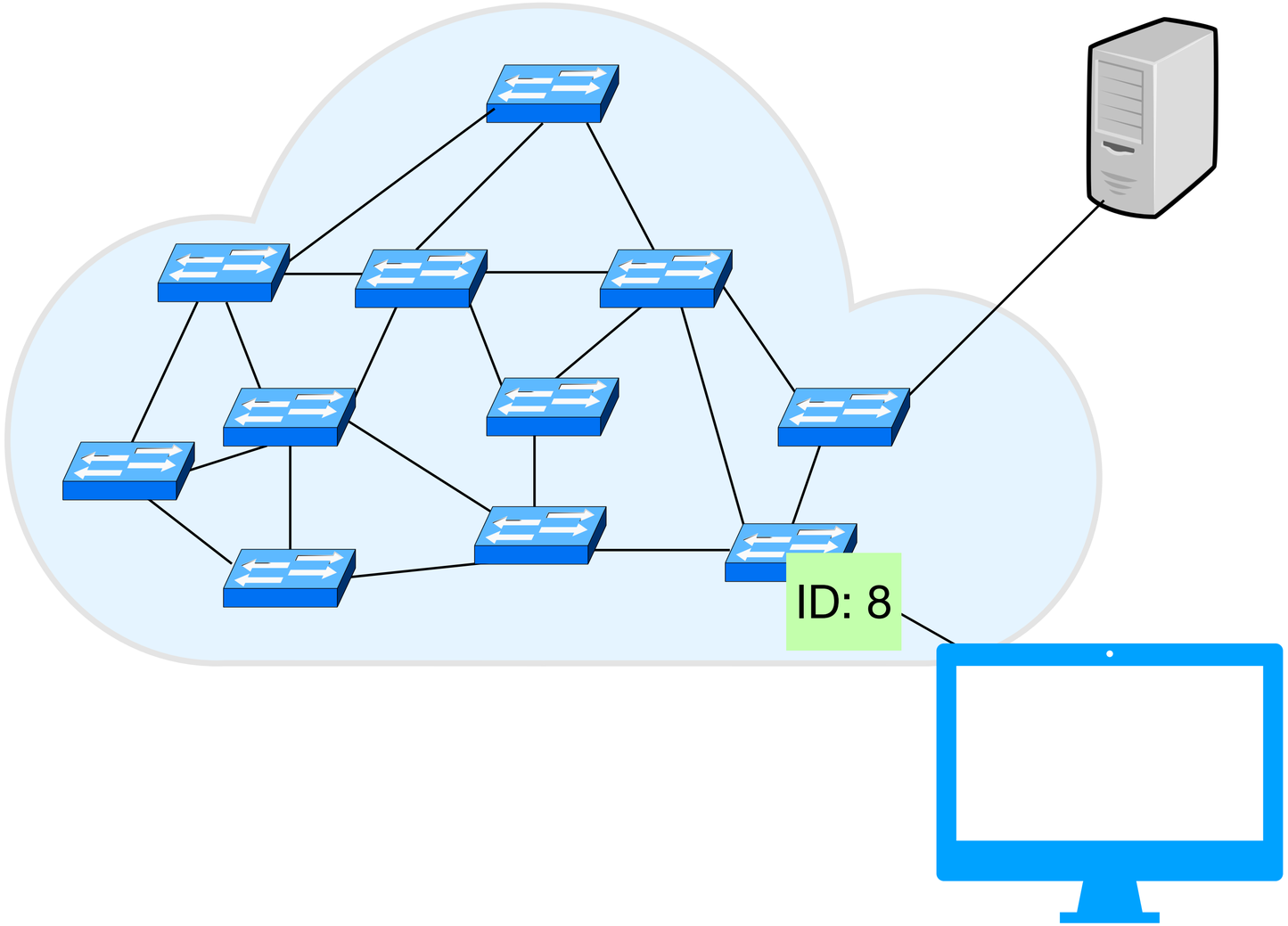
ID	Count



Strawman AR-DDoS Defense

We can track the number of times we've seen an ID

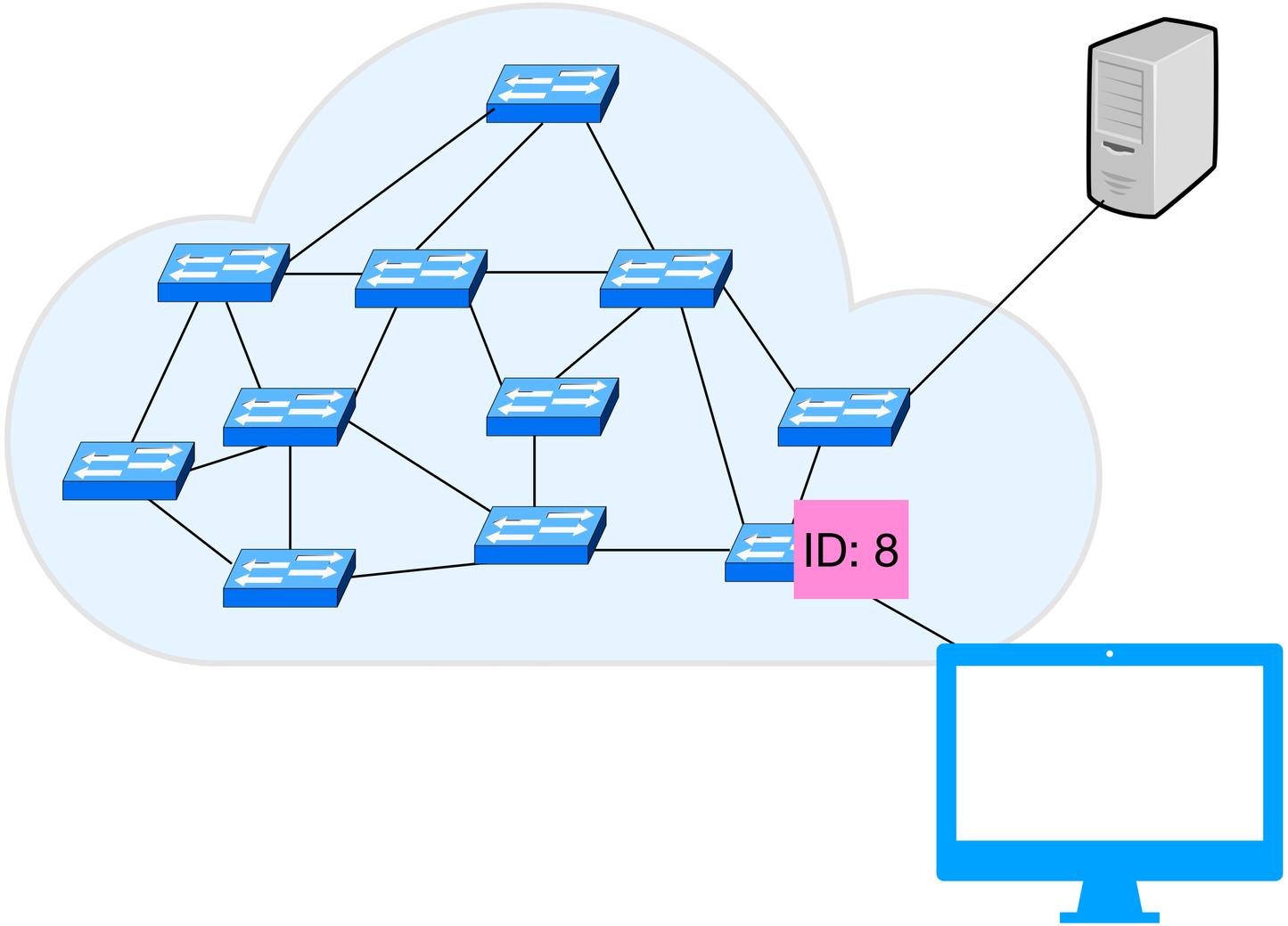
ID	Count
8	1



Strawman AR-DDoS Defense

If we've seen a request with the ID in the response, we classify it as legitimate

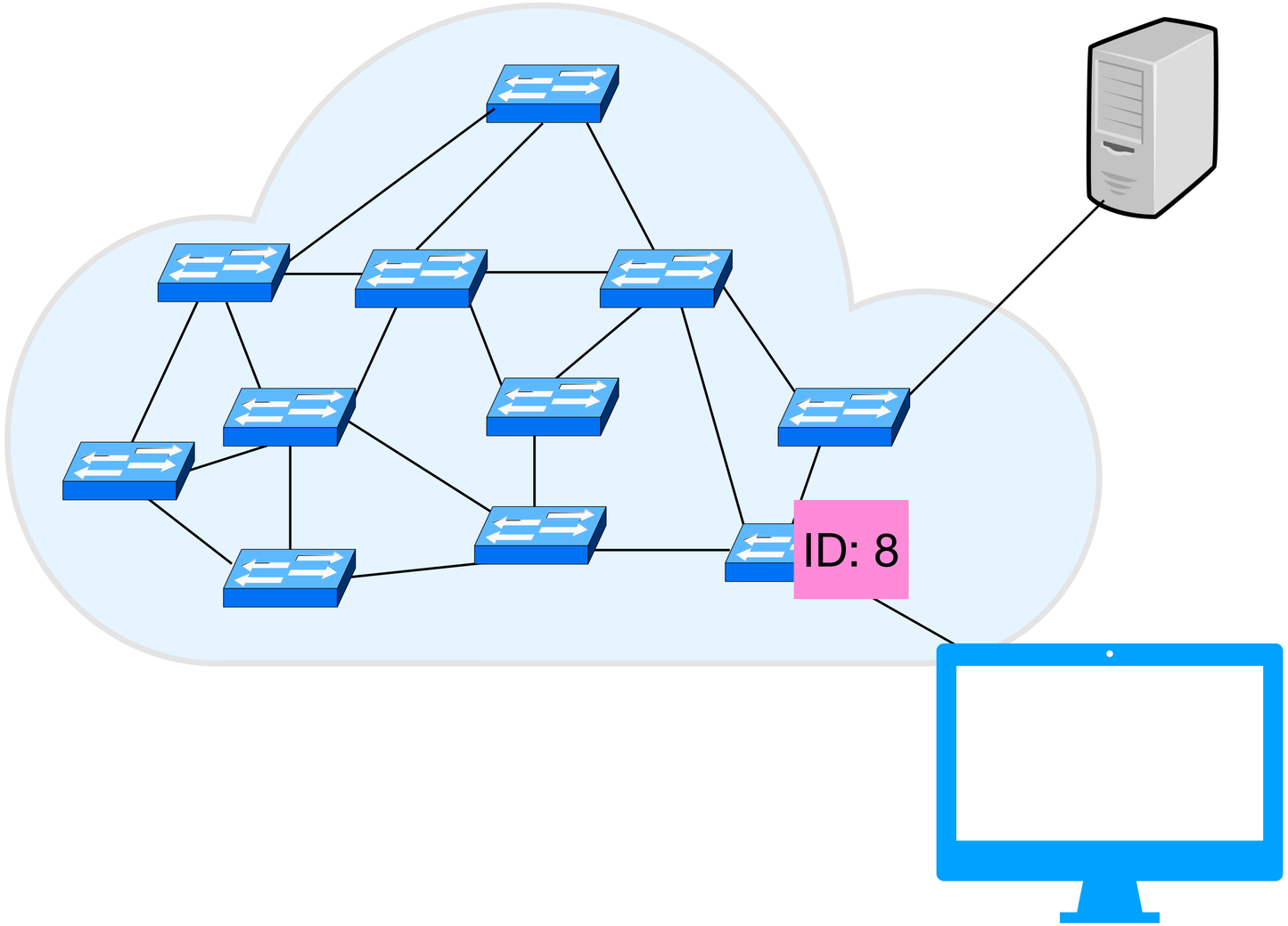
ID	Count
8	1



Strawman AR-DDoS Defense

If we've seen a request with the ID in the response, we classify it as legitimate

ID	Count
8	0

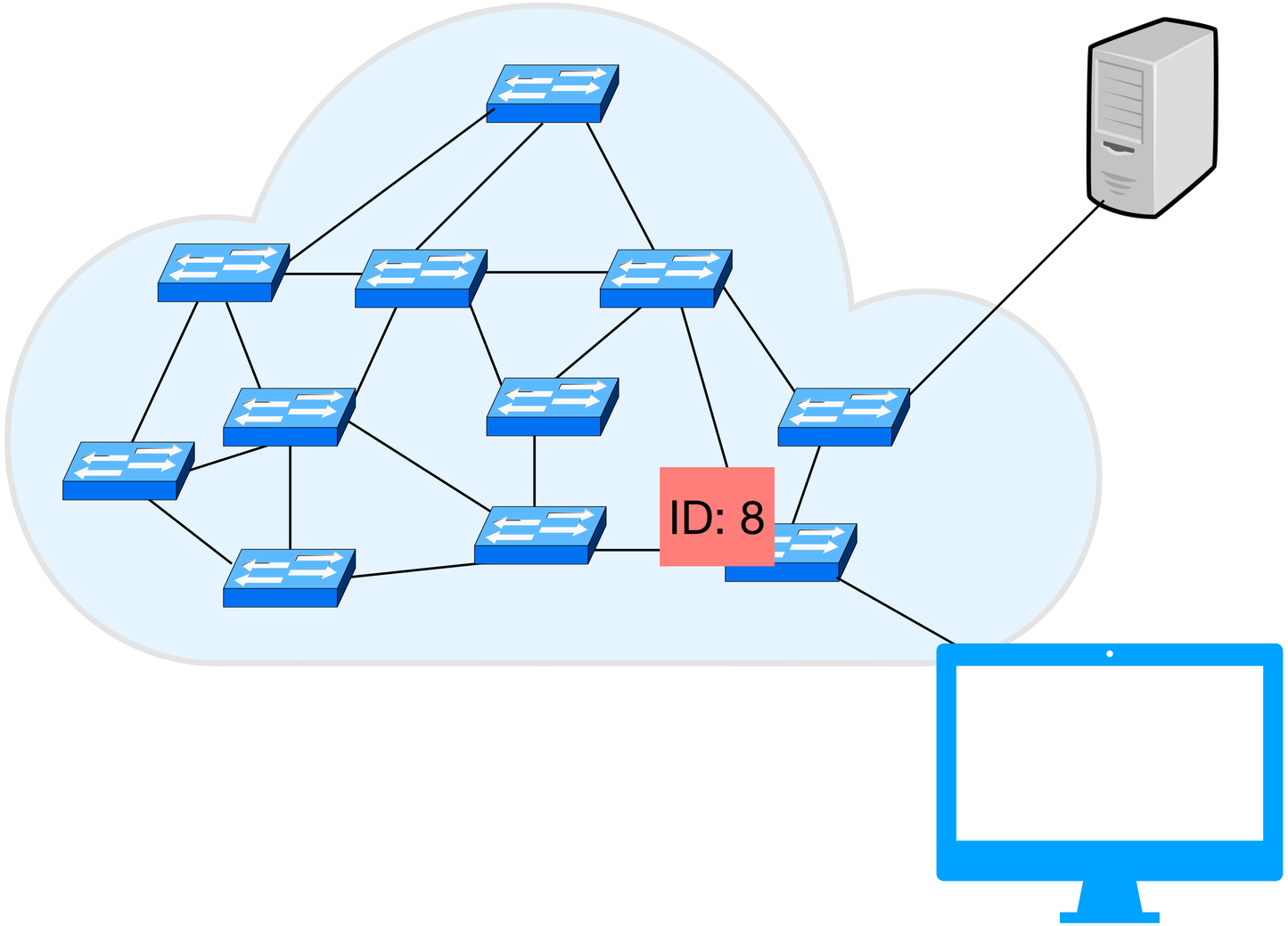


We decrement the count for each legitimate response, to keep the count up to date

Strawman AR-DDoS Defense

If the count is zero for an ID, the response is dropped

ID	Count
8	0



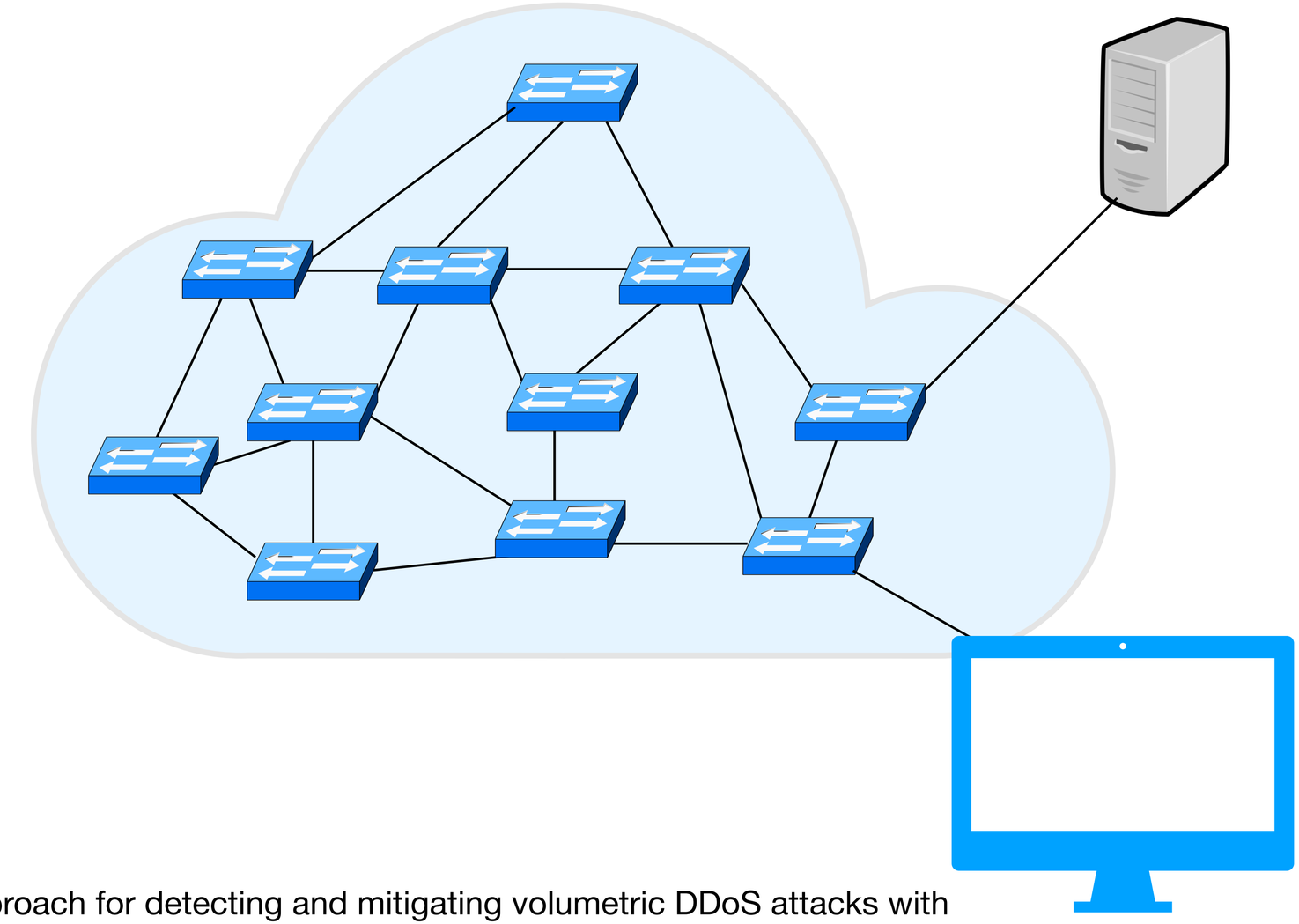
Challenges of AR-DDoS Defenses

In-network defenses running on programmable hardware must grapple with significant resource and computational limitations

Existing AR-DDoS Defenses

Bloom Filters keep track of previously-seen transaction IDs, without storing the IDs

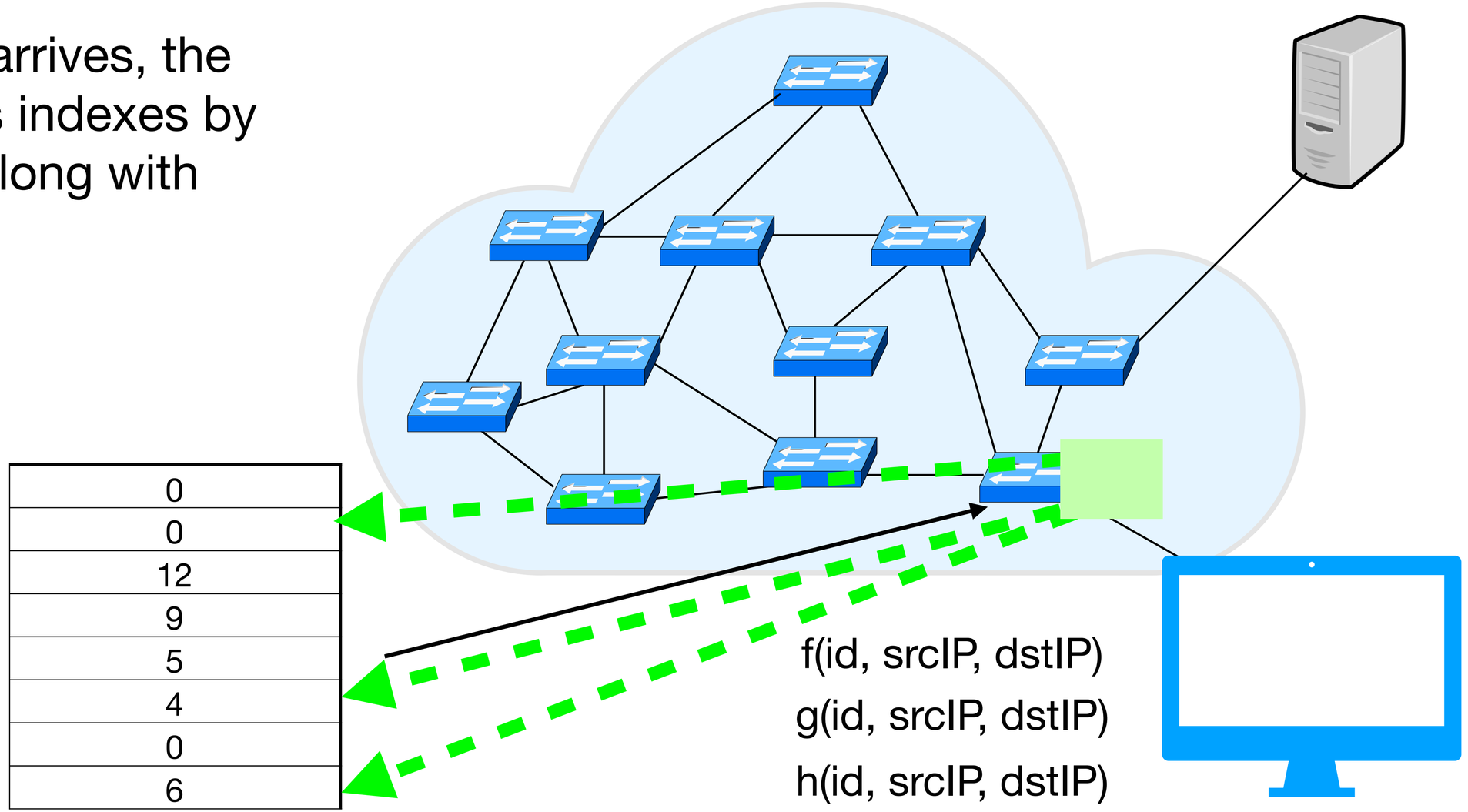
Bloom Filters are a compact, probabilistic data structure to accurately measure set membership



Z. Liu, et al. “Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches.” USENIX Security Symposium, 2021,

Existing AR-DDoS Defenses

When a request arrives, the switch computes indexes by hashing the ID, along with src and dst IPs

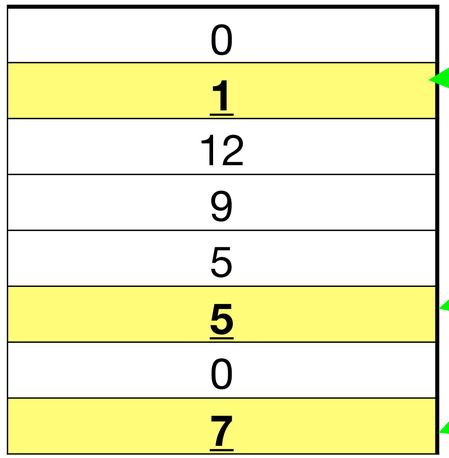


Counting Bloom Filter

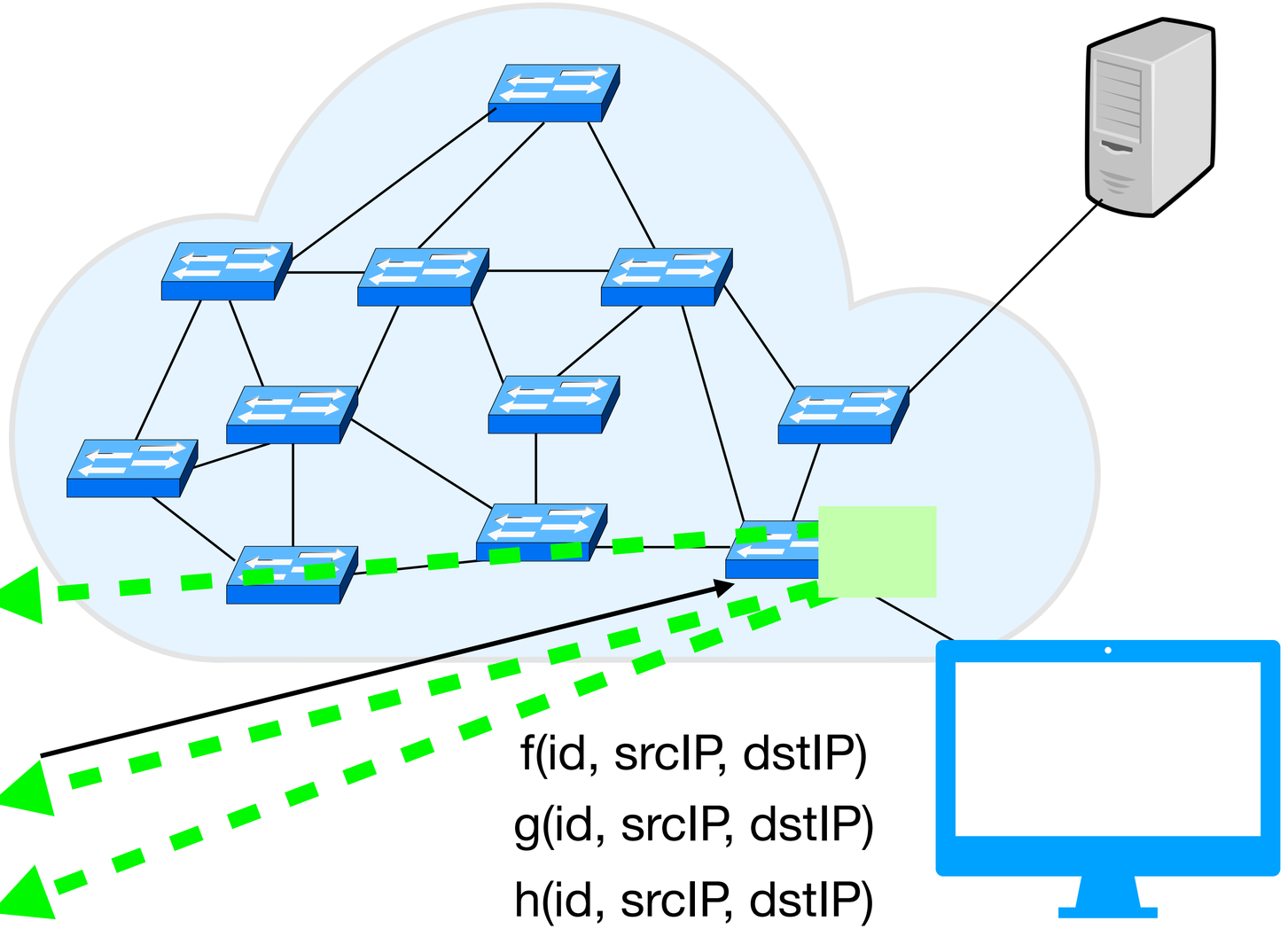
Existing AR-DDoS Defenses

When a request arrives, the switch computes indexes by hashing the ID, along with src and dst IPs

Counts at each index are incremented



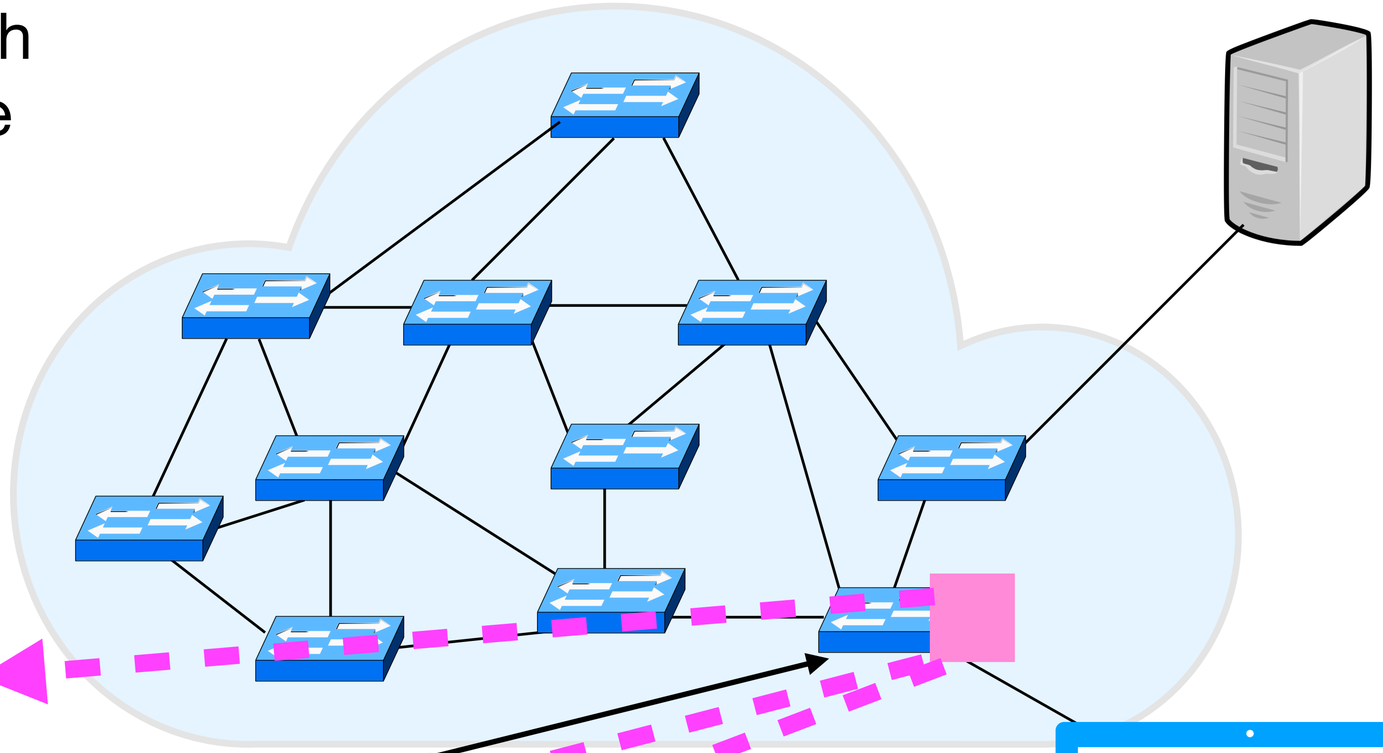
Counting Bloom Filter



Existing AR-DDoS Defenses

When a response arrives, the switch checks the filter using a hash of the ID and IPs in the header

If there was a (non-attack) request with the same ID, the values in the filter will be > 0



Not an attack

0
1
12
9
5
5
0
7

Counting Bloom Filter

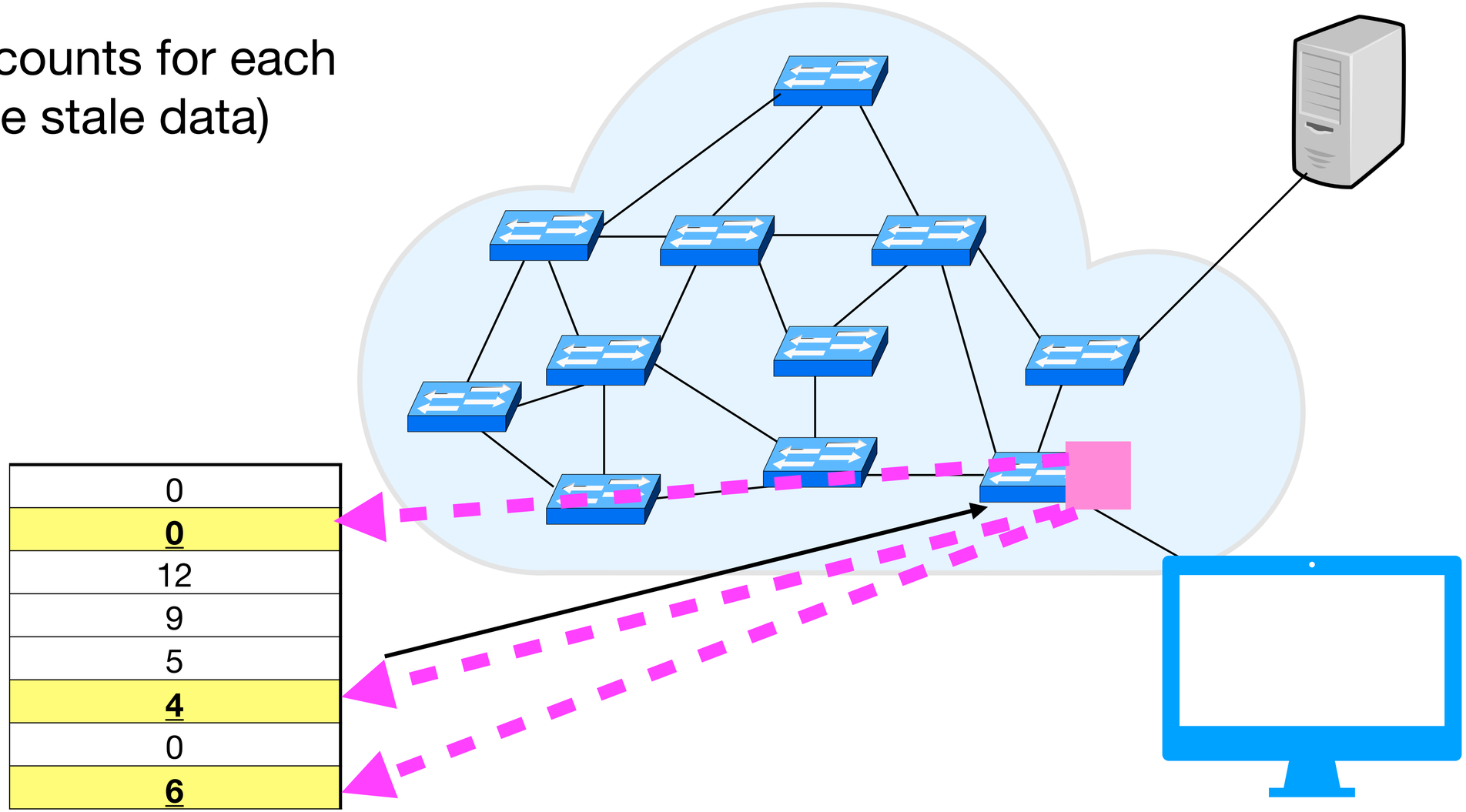
f(id, dstIP, srcIP)

g(id, dstIP, srcIP)

h(id, dstIP, srcIP)

Existing AR-DDoS Defenses

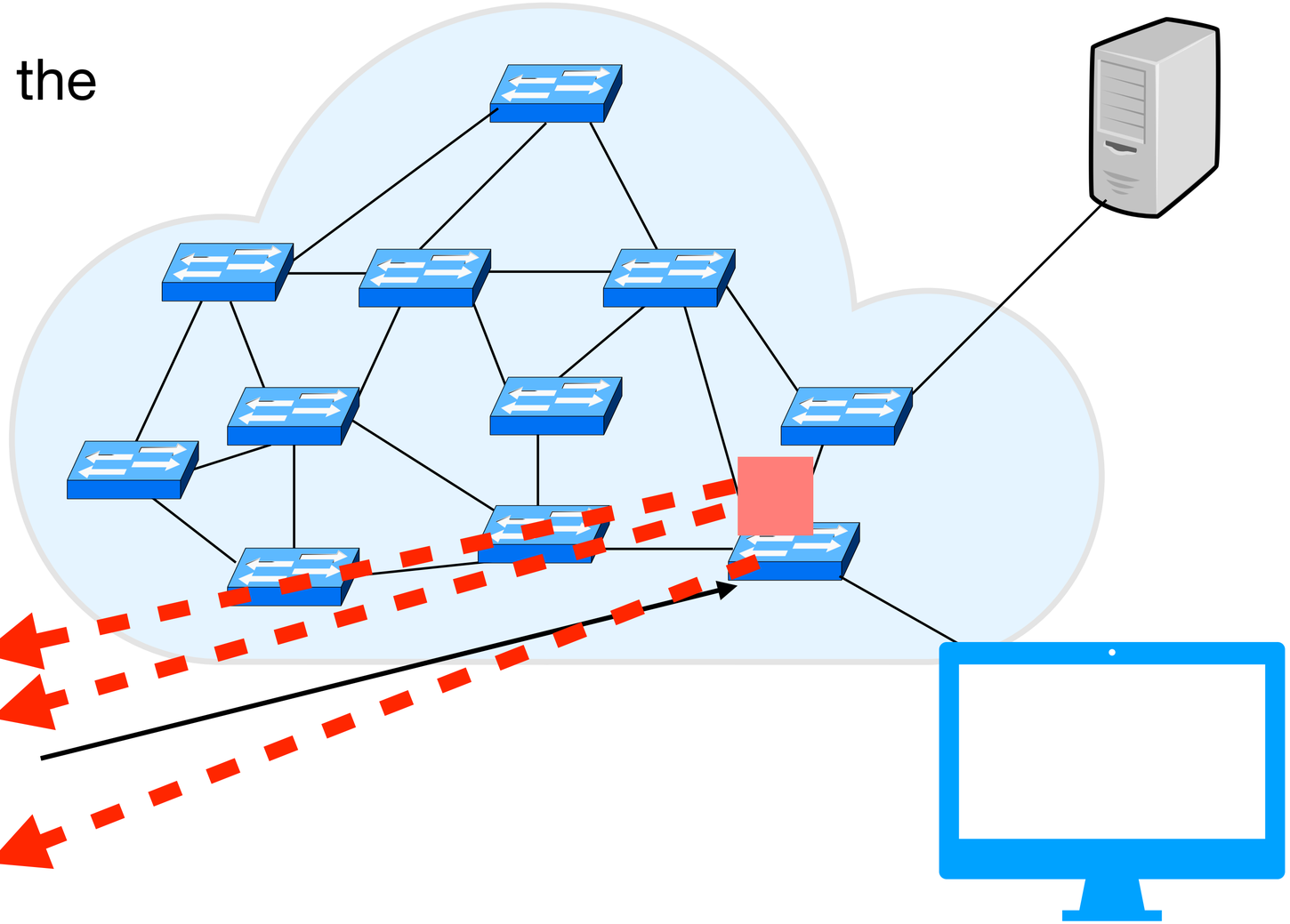
We decrement the counts for each response (to remove stale data)



Counting Bloom Filter

Existing AR-DDoS Defenses

If at least one of the counts is 0, the response is dropped



Attack!

0
0
12
9
5
4
0
6

Counting Bloom Filter

Challenges of AR-DDoS Defenses

In-network defenses running on programmable hardware must grapple with significant resource and computational limitations

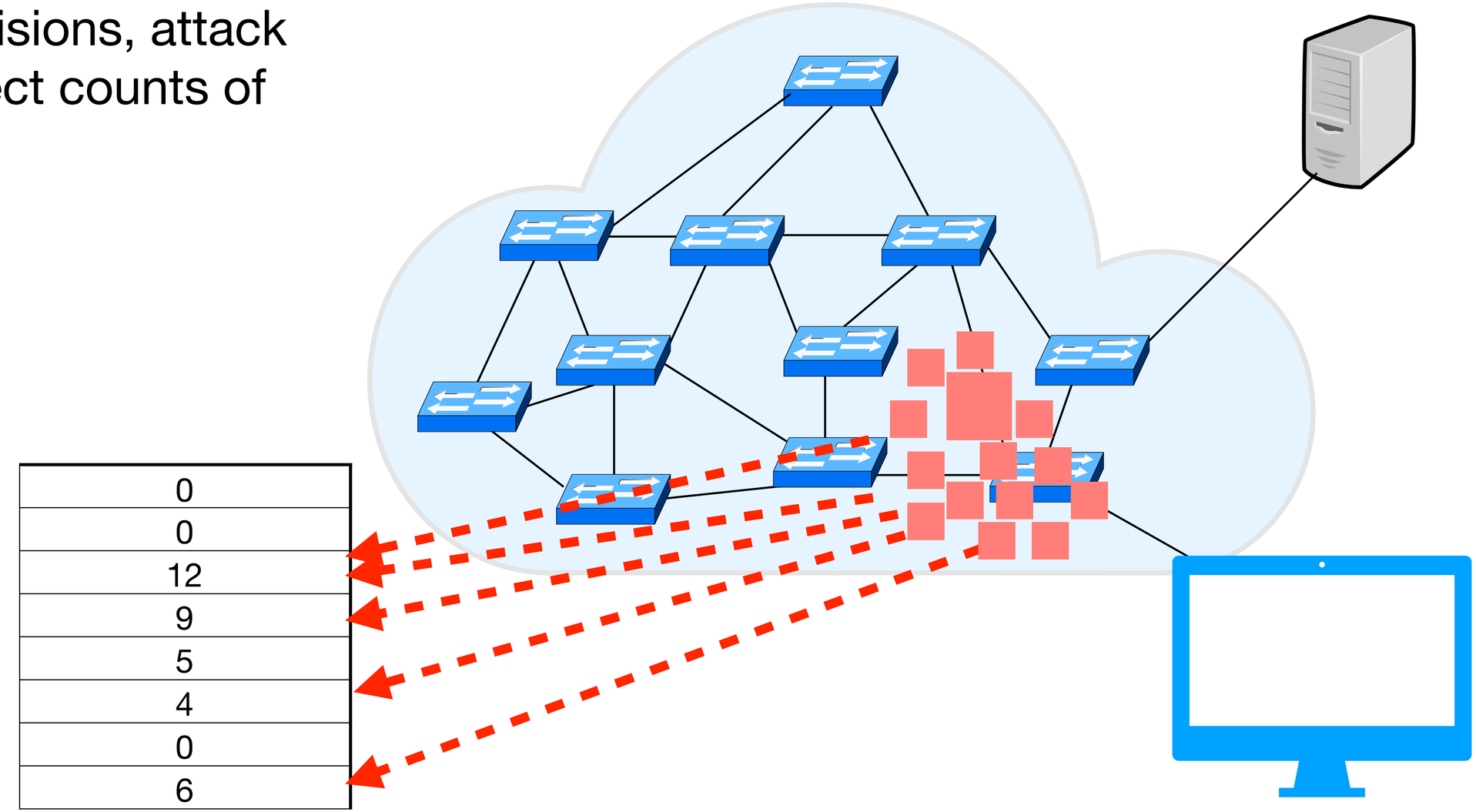
Challenges of AR-DDoS Defenses

In-network defenses running on programmable hardware must grapple with significant resource and computational limitations

Performance of defenses maintaining per-ID counts is sensitive to attack volume because *attack responses alter the saved state*

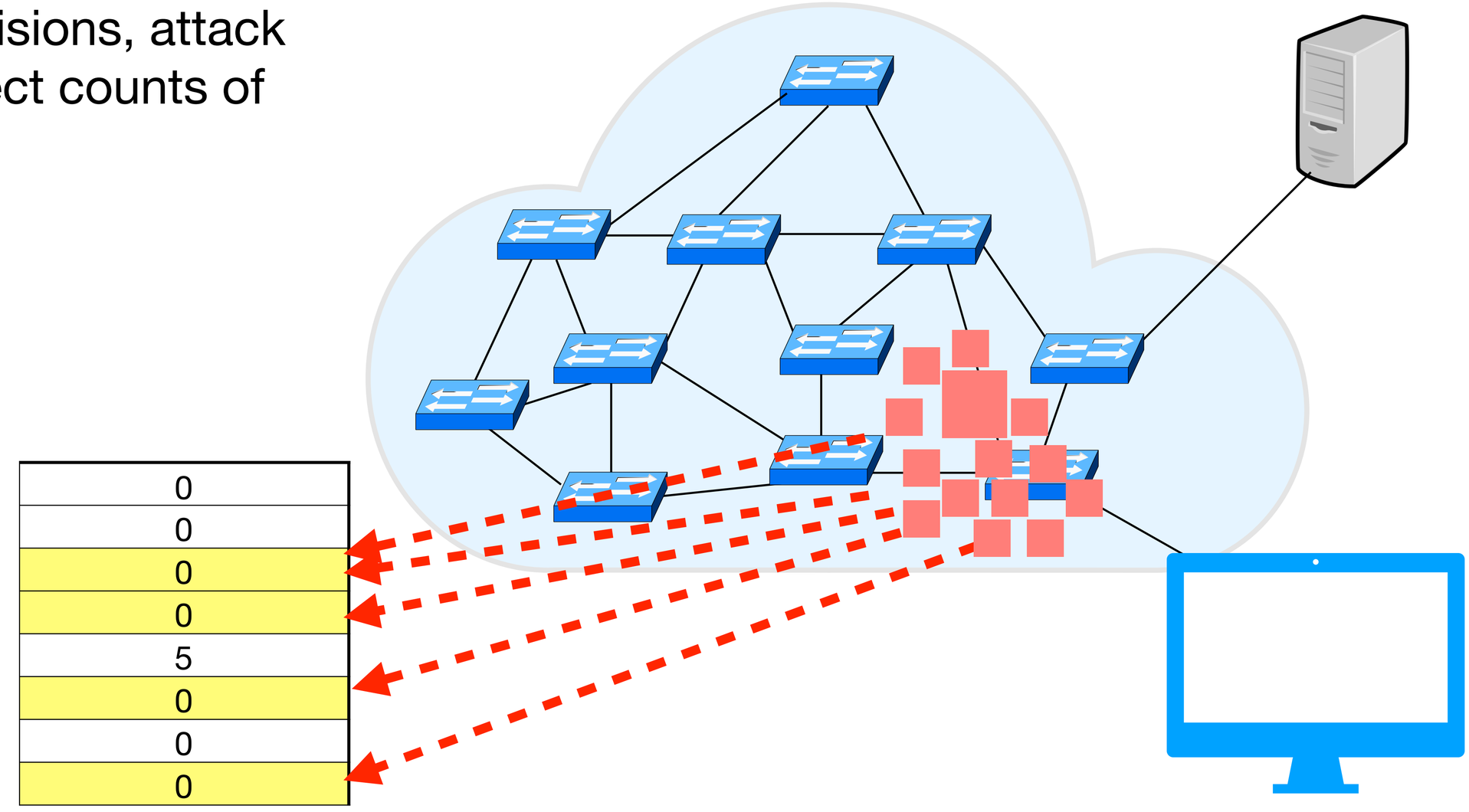
Attack traffic affects performance

Due to hash collisions, attack packets can affect counts of legitimate IDs



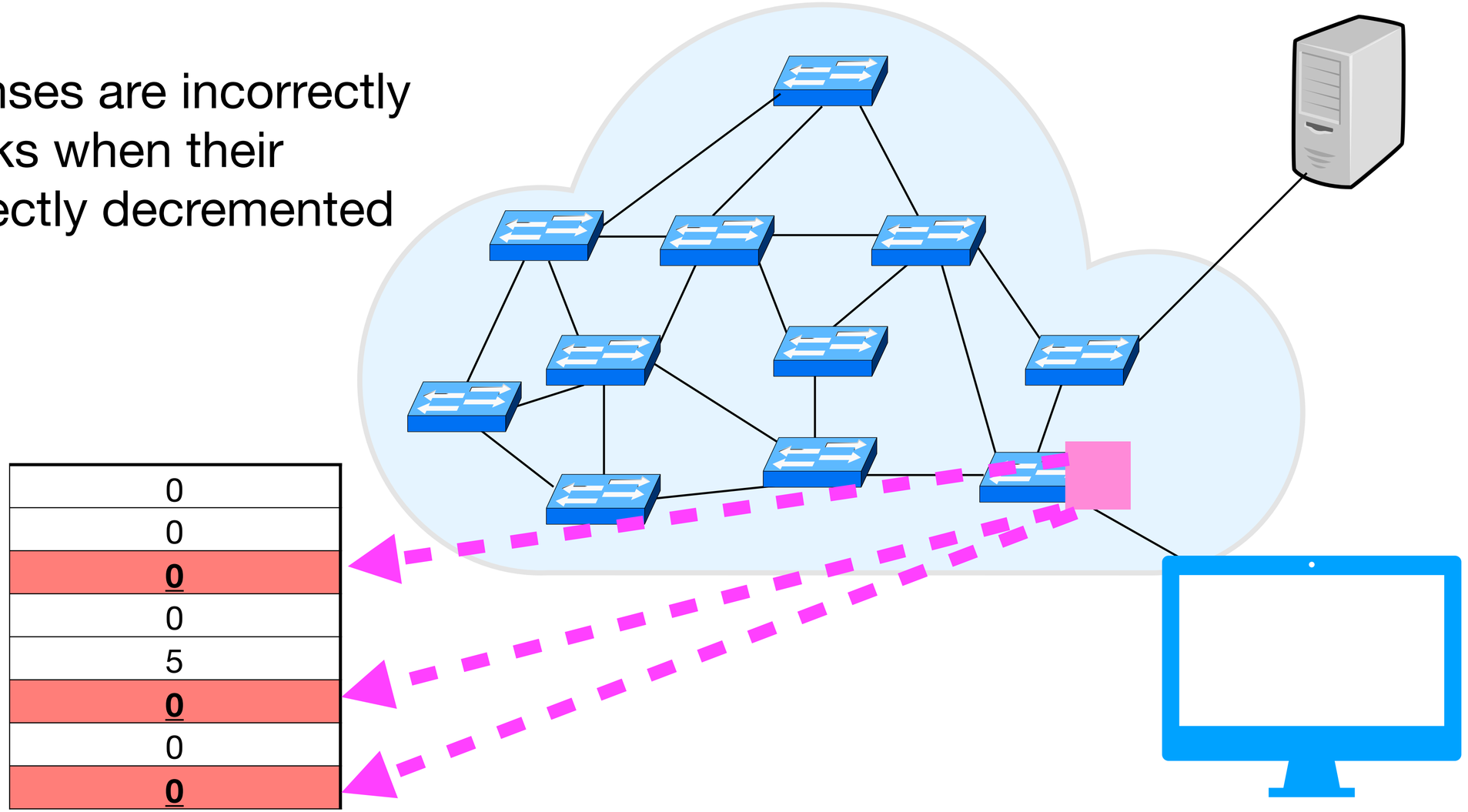
Attack traffic affects performance

Due to hash collisions, attack packets can affect counts of legitimate IDs



Attack traffic affects performance

Legitimate responses are incorrectly dropped as attacks when their counts are incorrectly decremented



Challenges of AR-DDoS Defenses

In-network defenses running on programmable hardware must grapple with significant resource and computational limitations

Performance of defenses maintaining per-ID counts is sensitive to attack volume because *attack responses alter the saved state*

Challenges of AR-DDoS Defenses

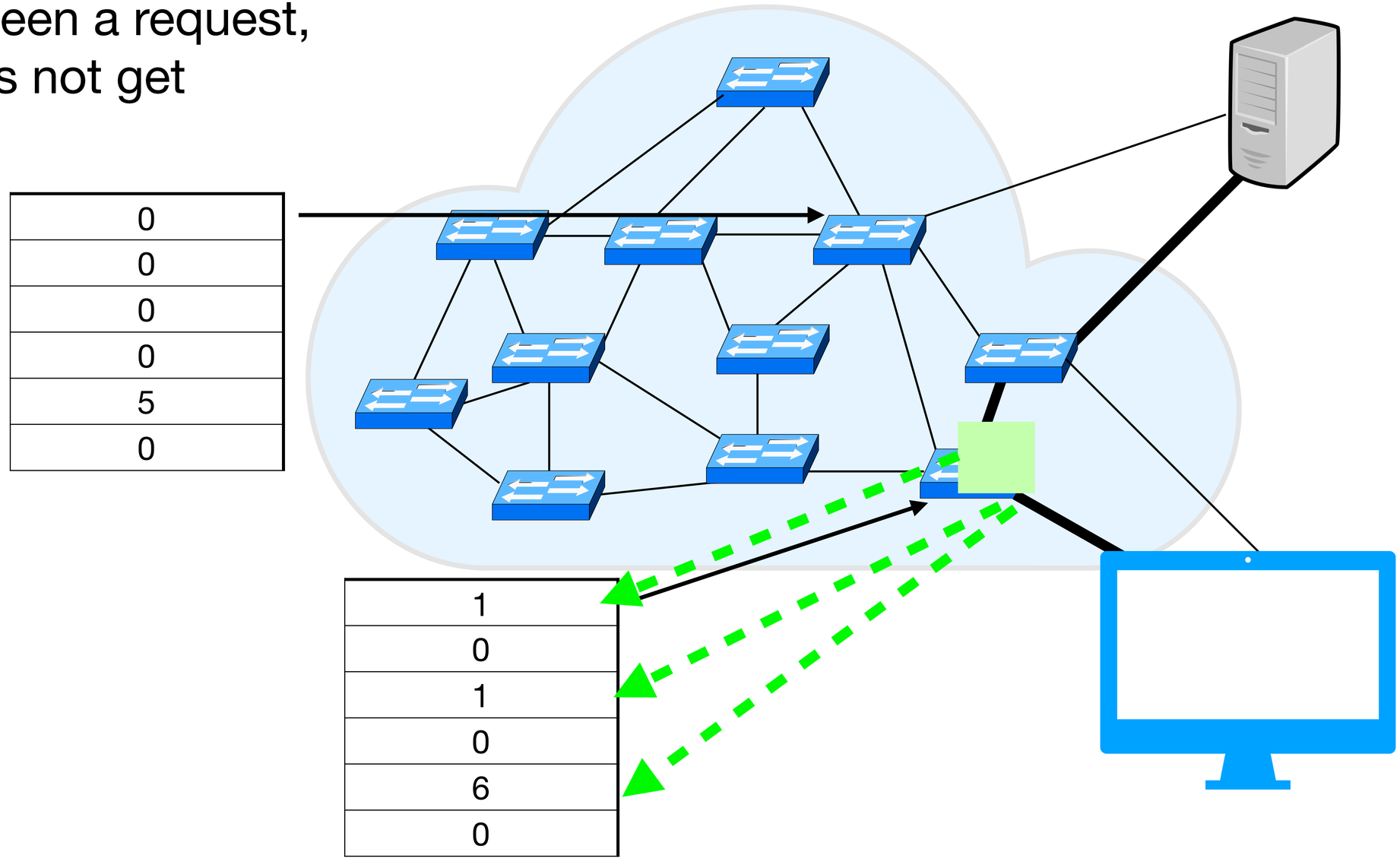
In-network defenses running on programmable hardware must grapple with significant resource and computational limitations

Performance of defenses maintaining per-ID counts is sensitive to attack volume because *responses alter the saved state*

State-of-the-art defenses block legitimate traffic when routing is asymmetric

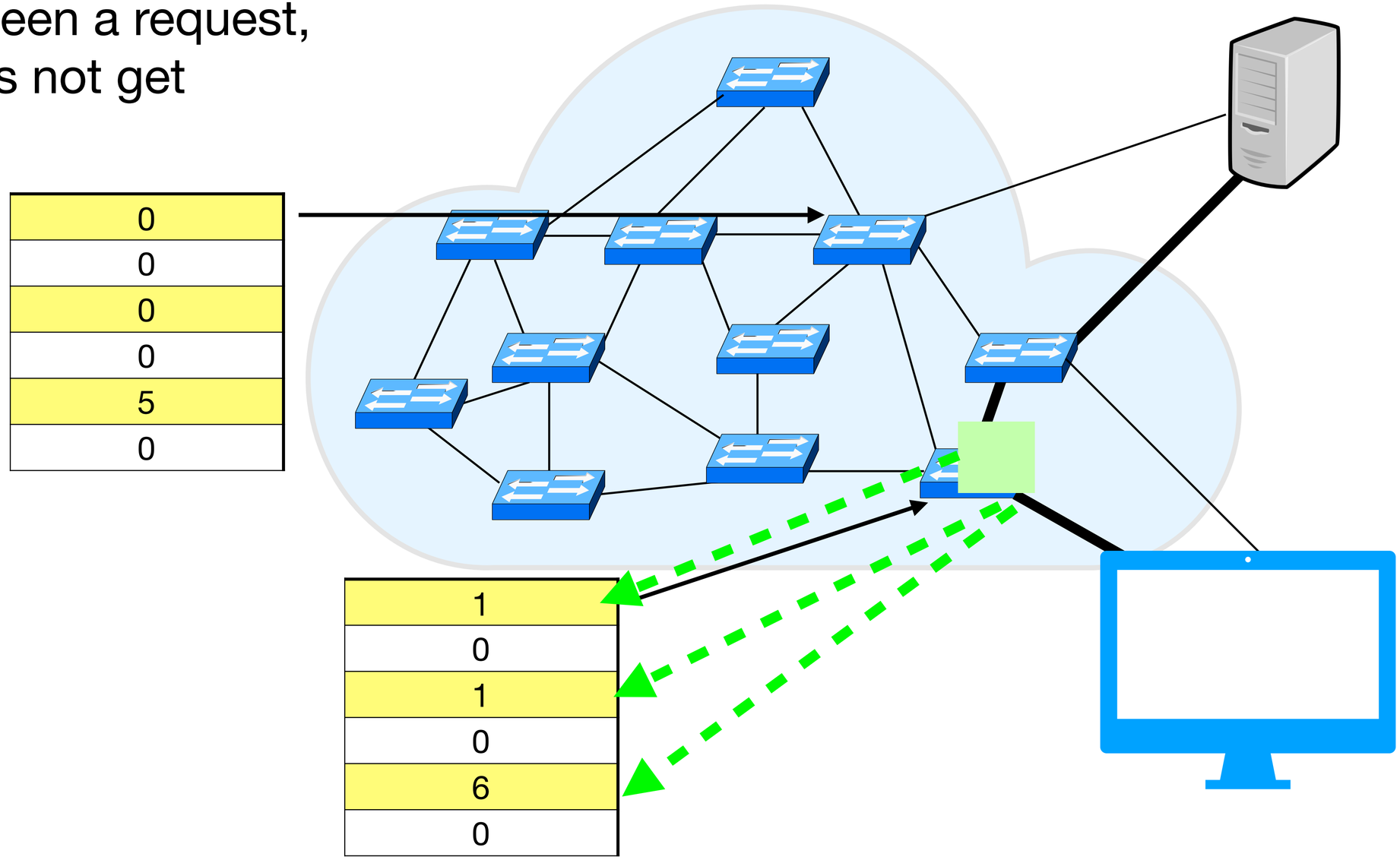
Asymmetric traffic is blocked

If the switch hasn’t seen a request, the Bloom Filter does not get incremented



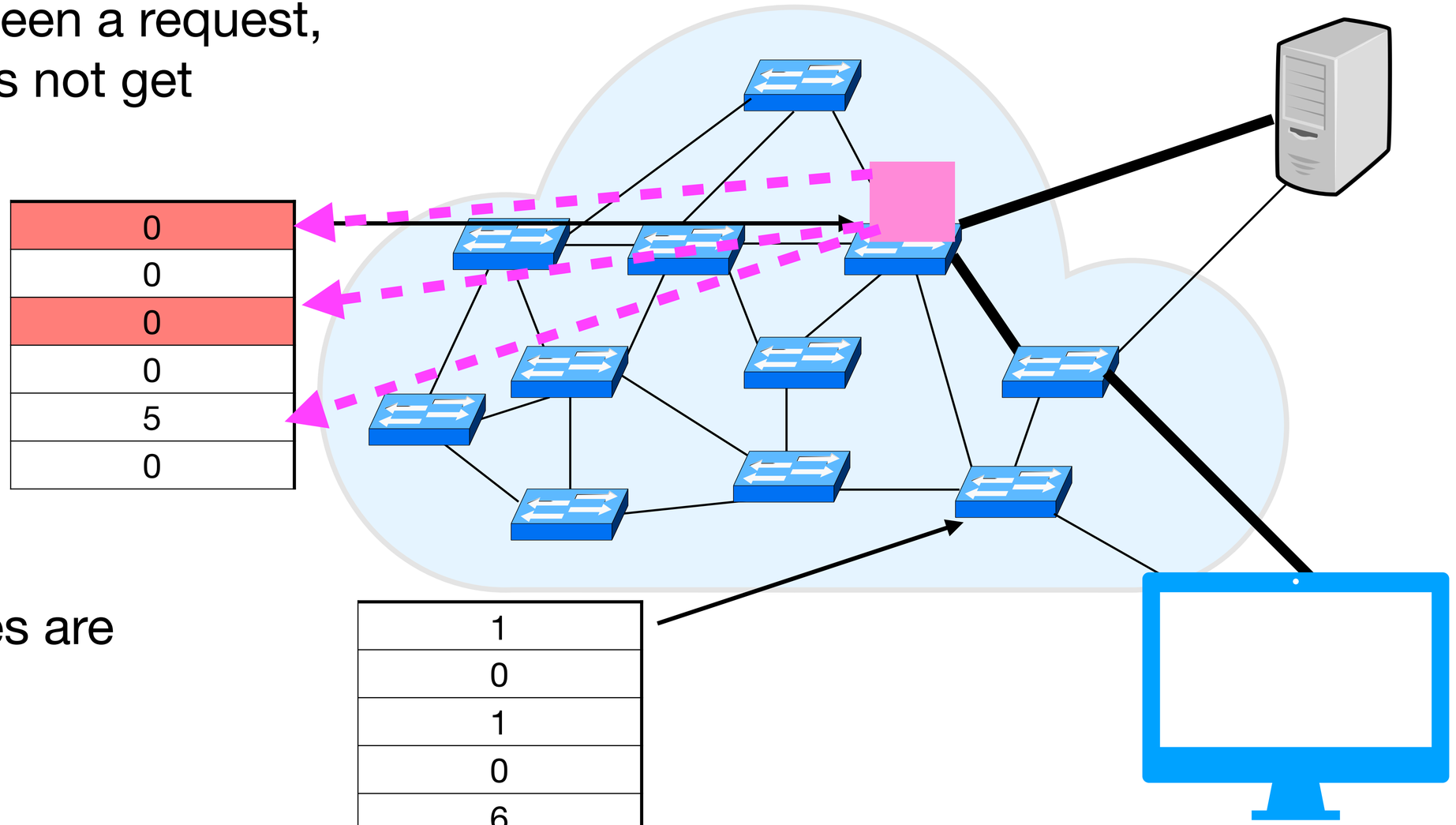
Asymmetric traffic is blocked

If the switch hasn’t seen a request, the Bloom Filter does not get incremented



Asymmetric traffic is blocked

If the switch hasn’t seen a request, the Bloom Filter does not get incremented



Legitimate responses are treated as attacks

Defenses block legitimate traffic

Even if this only happens occasionally, this is a significant roadblock to getting AR-DDoS defenses adopted in practice

Network operators and ISPs need a solution that blocks most attack traffic while still allowing legitimate traffic with minimal disruption

ReAct

Overview

Implementation

Evaluation

Summary

ReAct

Overview

Implementation

Evaluation

Summary

ReAct: a practical DDoS solution

ReAct is an attack mitigation strategy that works for both symmetric and asymmetric traffic

The key insights are:

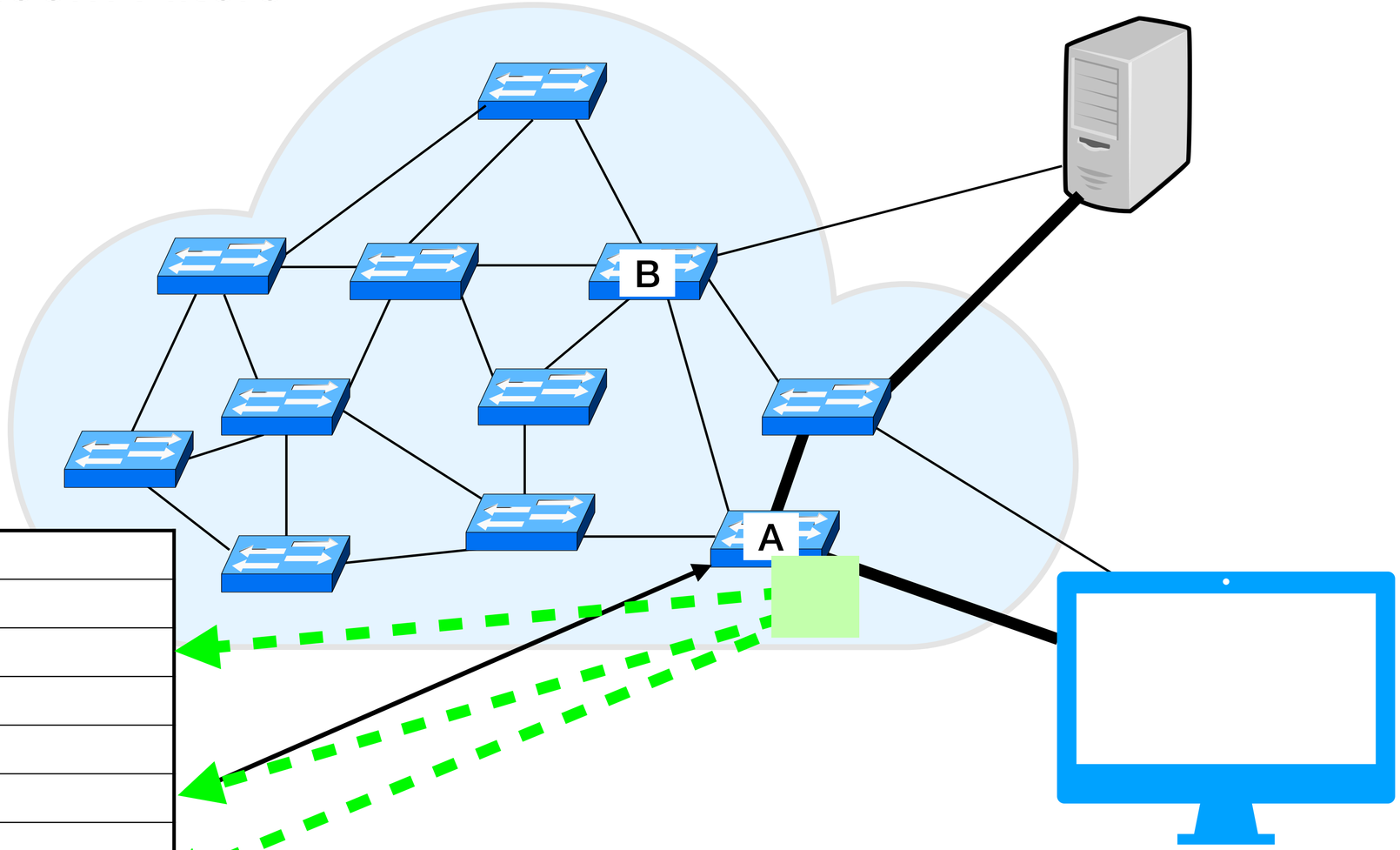
- When a switch sees a request, it will send it to the switch that will see the matching response
- Switches dynamically learn where responses are processed
- Only requests can alter the state of the Bloom Filters

ReAct implements an efficient sharing mechanism to reduce communication overhead

ReAct

ReAct uses non-counting Bloom Filters

If a switch has seen a request ID at least once, the value in the filter is 1

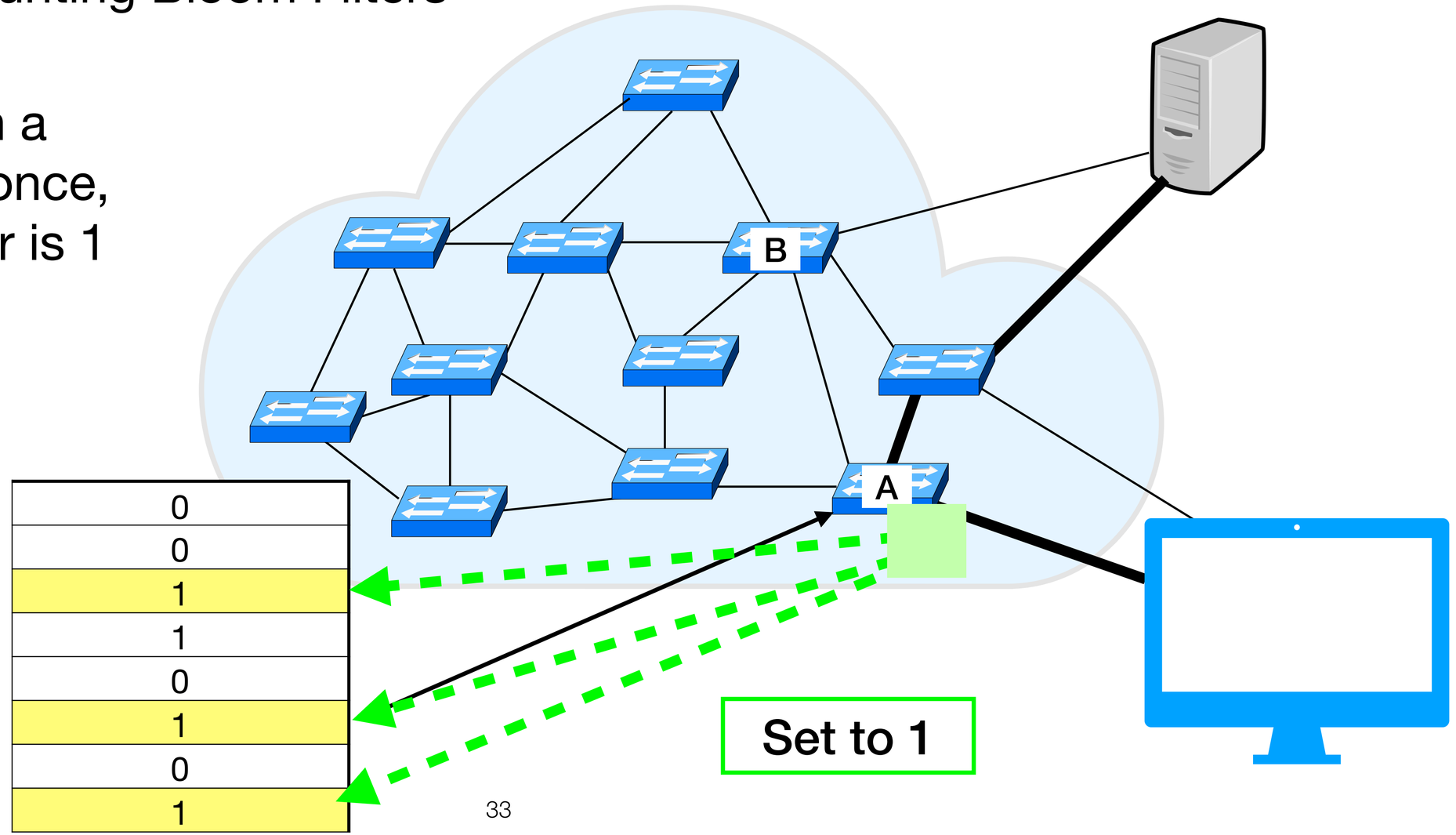


Bloom Filter

ReAct

ReAct uses non-counting Bloom Filters

If a switch has seen a request ID at least once, the value in the filter is 1

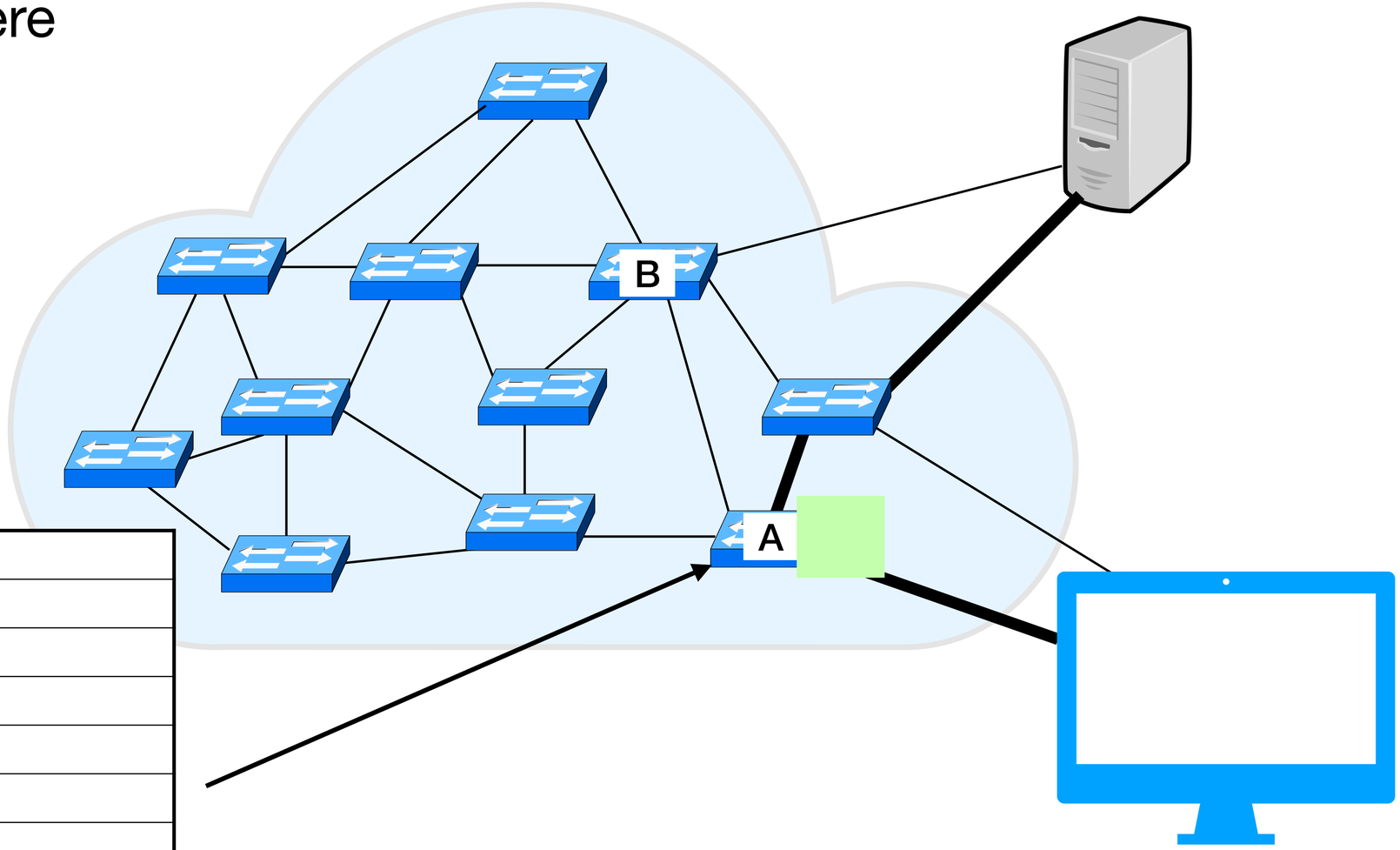


ReAct

Switches keep track of where responses are processed

Src IP Prefix	Response switch
10.168.0.0/16	B

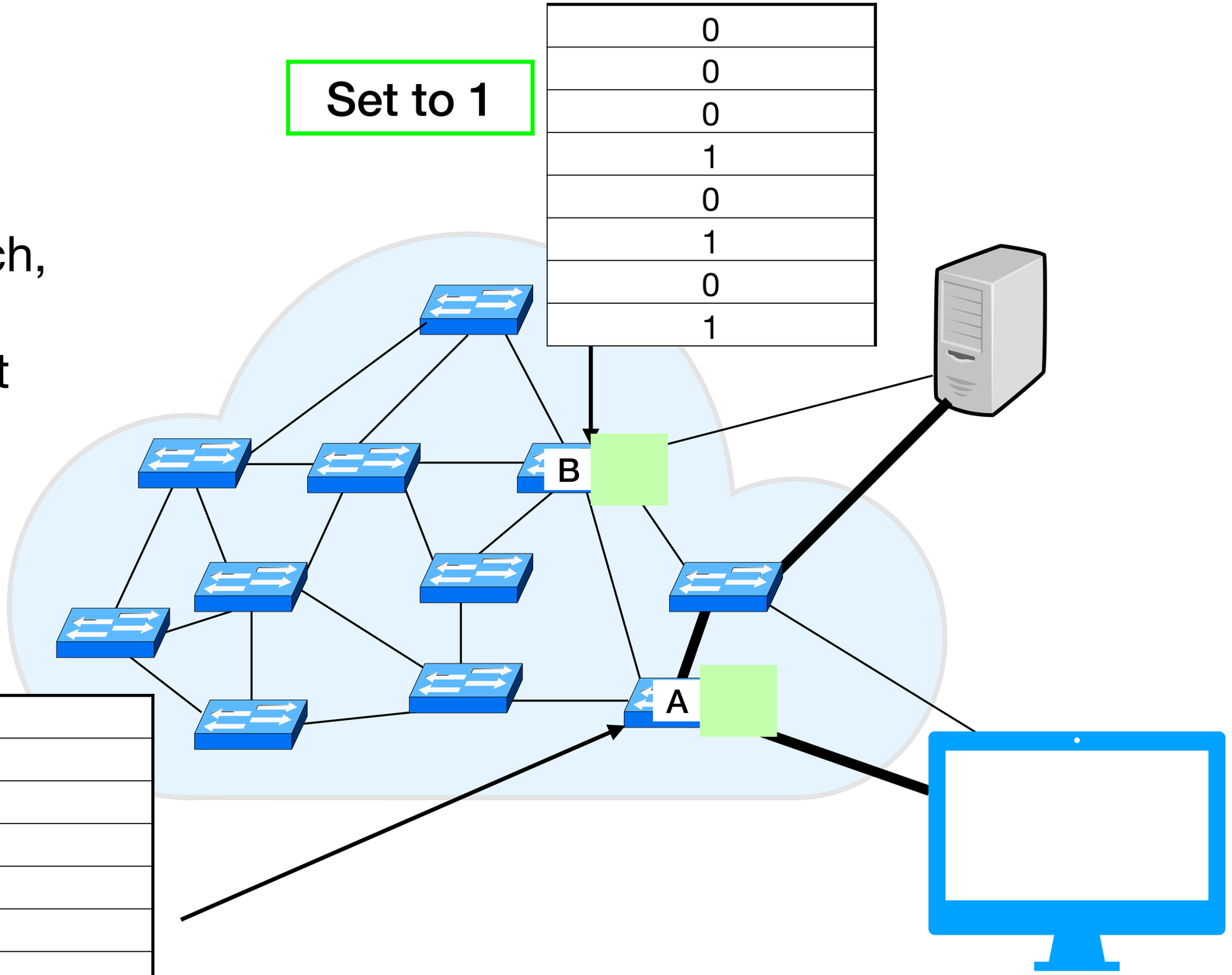
0
0
1
1
0
1
0
1



ReAct

If the response will be handled by a different switch, ReAct duplicates and forwards the request to that switch

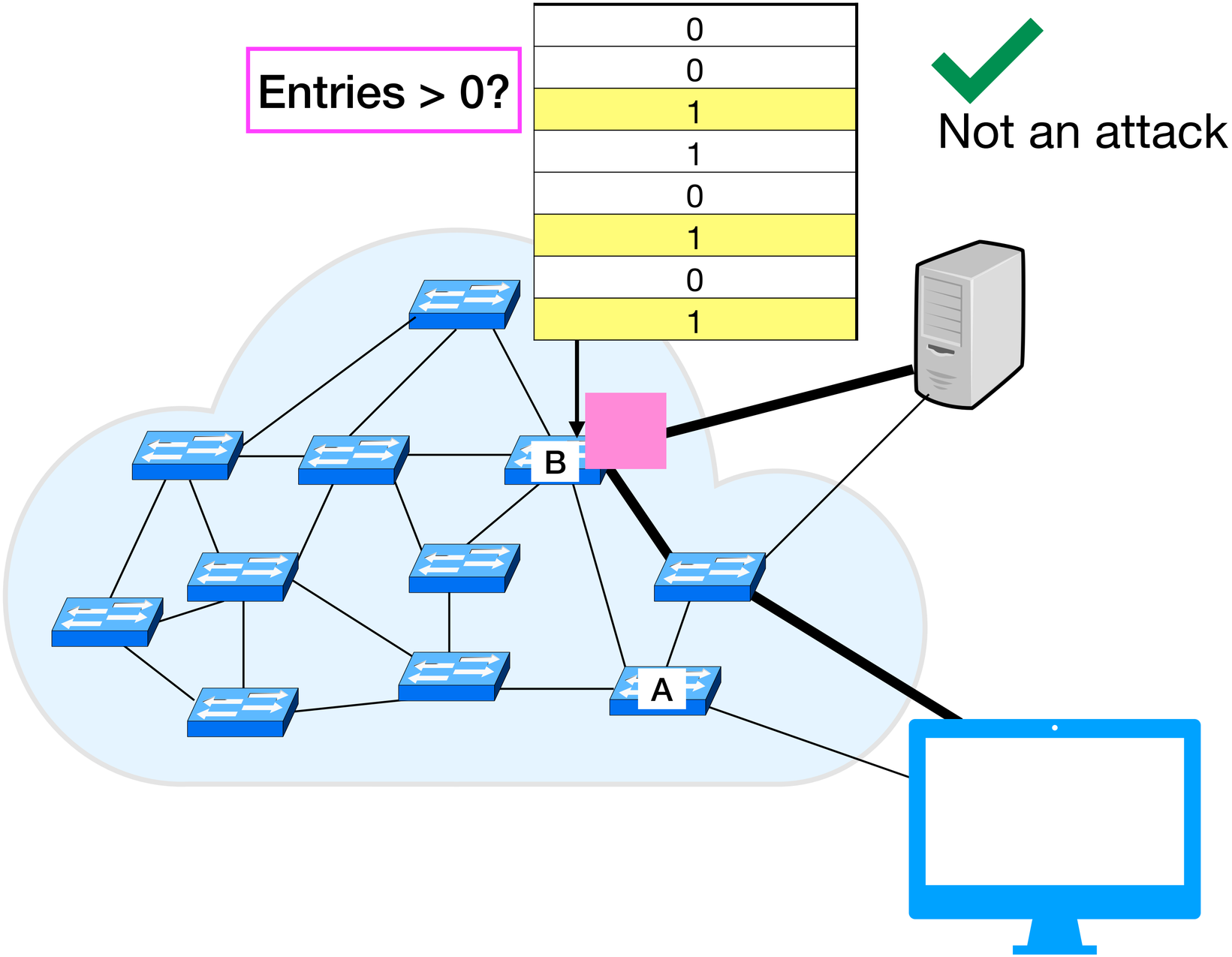
Src IP Prefix	Response switch
10.168.0.0/16	B



ReAct

Switches query the filter upon receiving a response

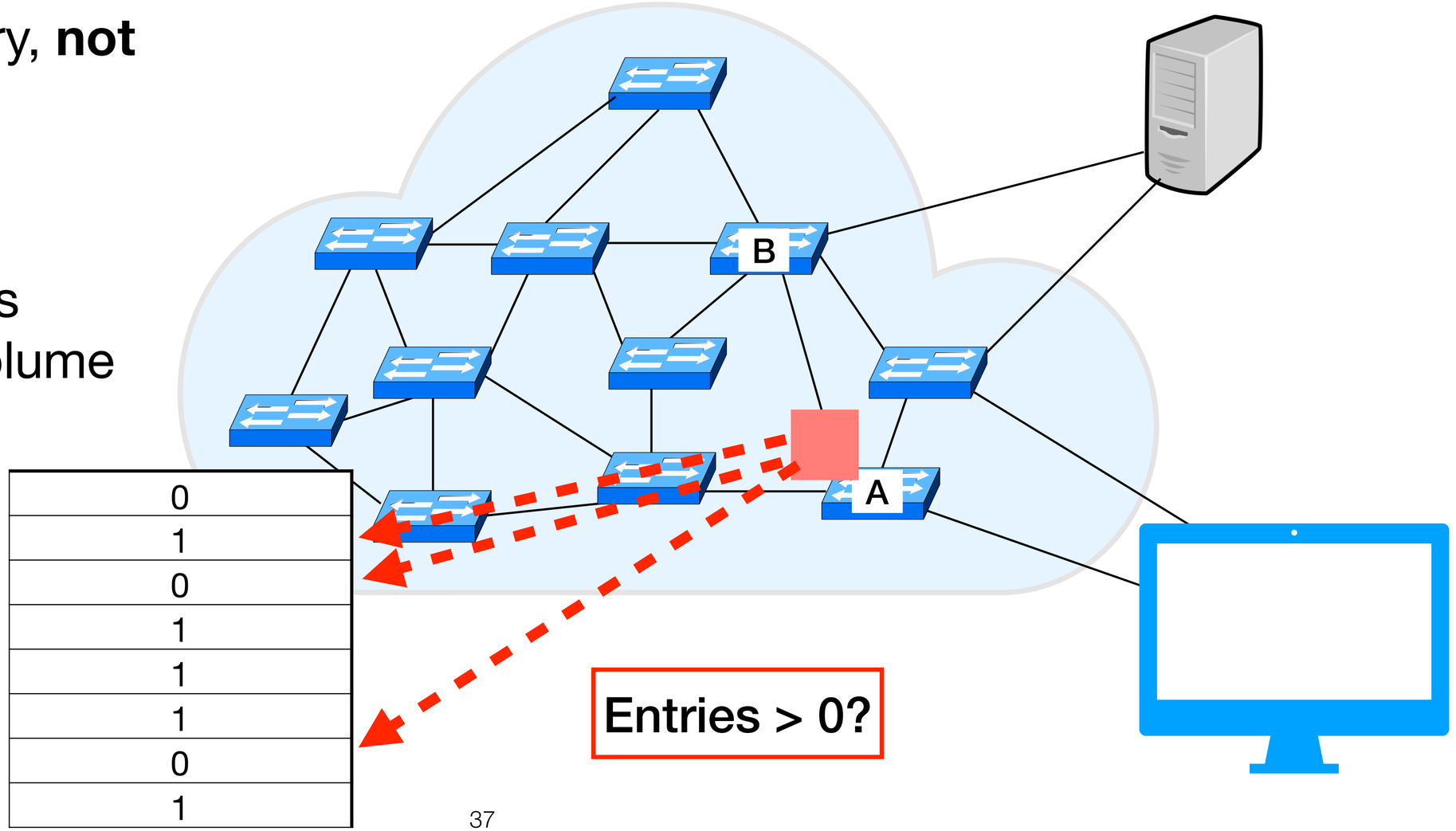
The switch processing the response has up-to-date measurements



ReAct - Attack

Responses only query, **not alter**, the filter

ReAct performance is agnostic to attack volume



ReAct

Overview

Implementation

Evaluation

Summary

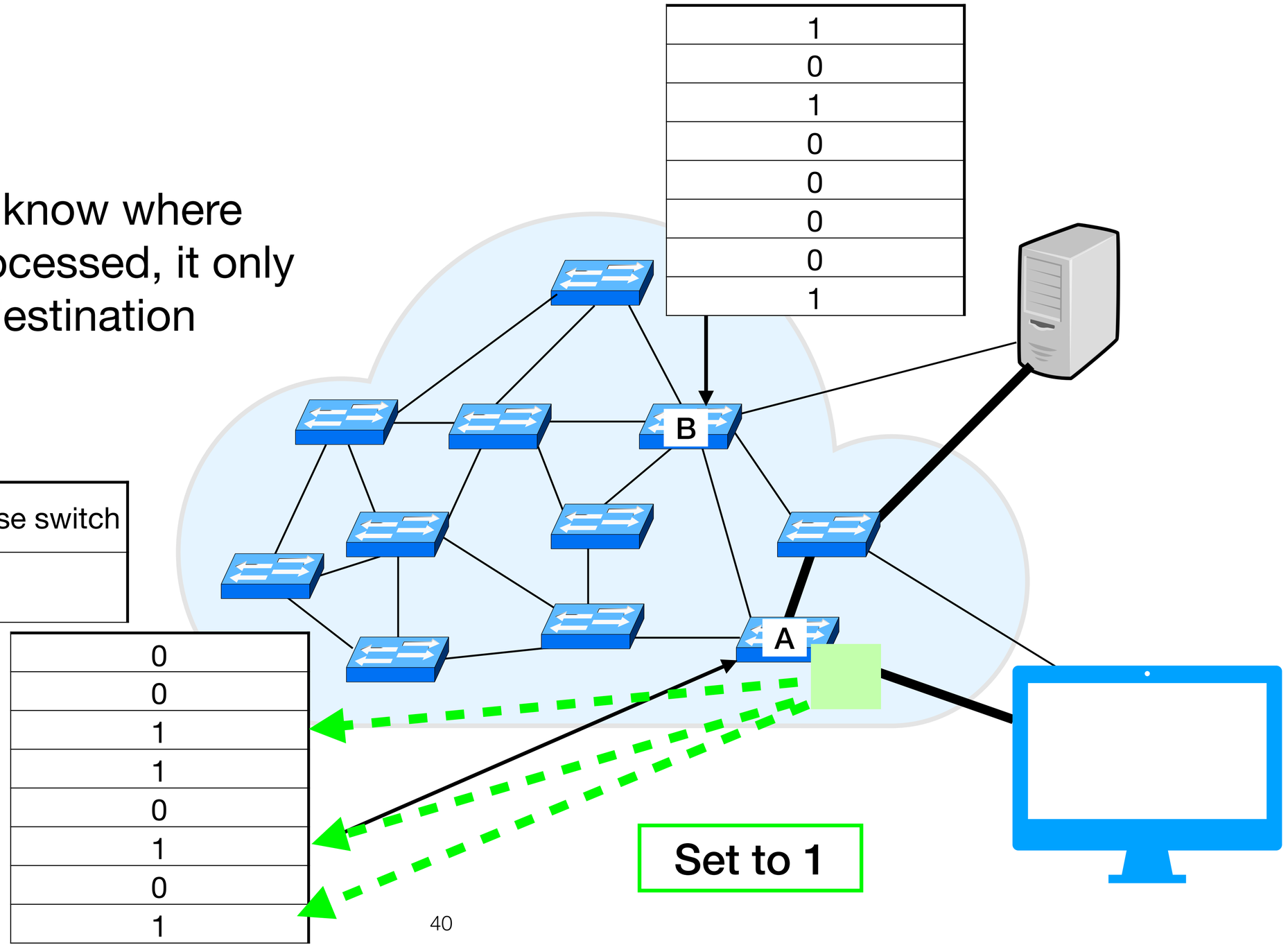
How ReAct is implemented

Broadcast mechanism for sharing requests if we don’t know where responses are processed ahead of time

ReAct

If a switch doesn’t know where the response is processed, it only forwards it to the destination

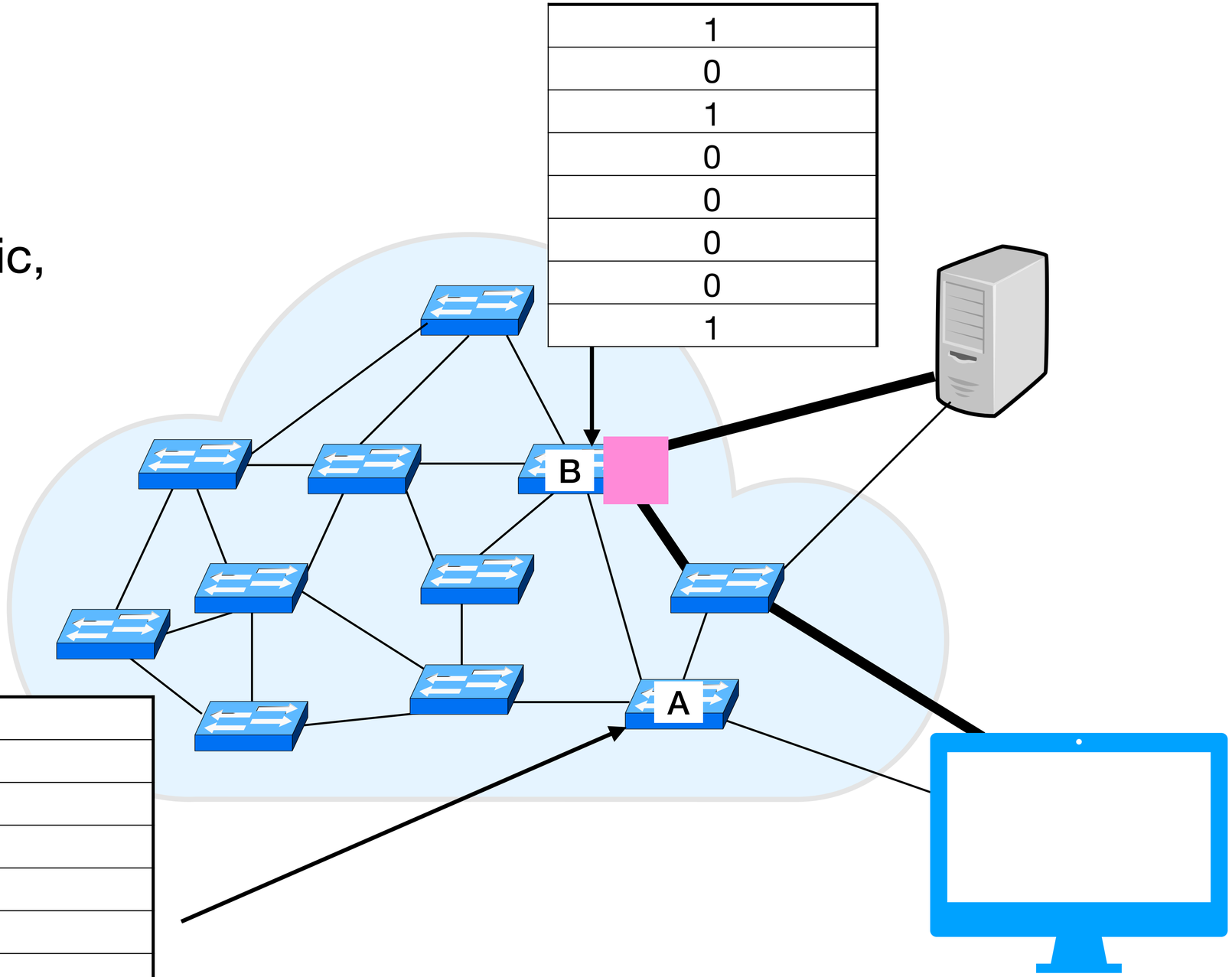
Src IP Prefix	Response switch



ReAct

If the response is asymmetric,
it will initially be dropped

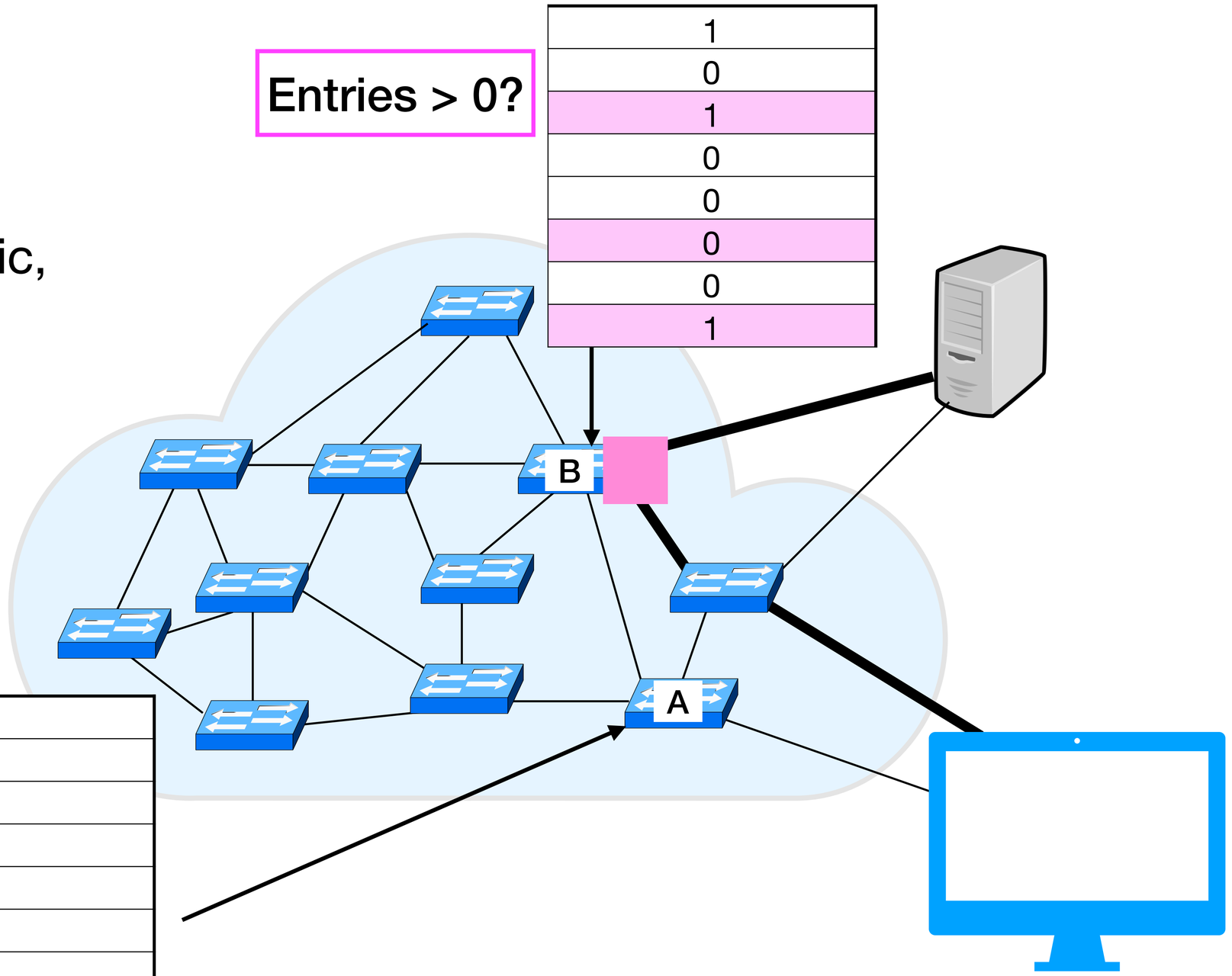
Src IP Prefix	Response switch



ReAct

If the response is asymmetric,
it will initially be dropped

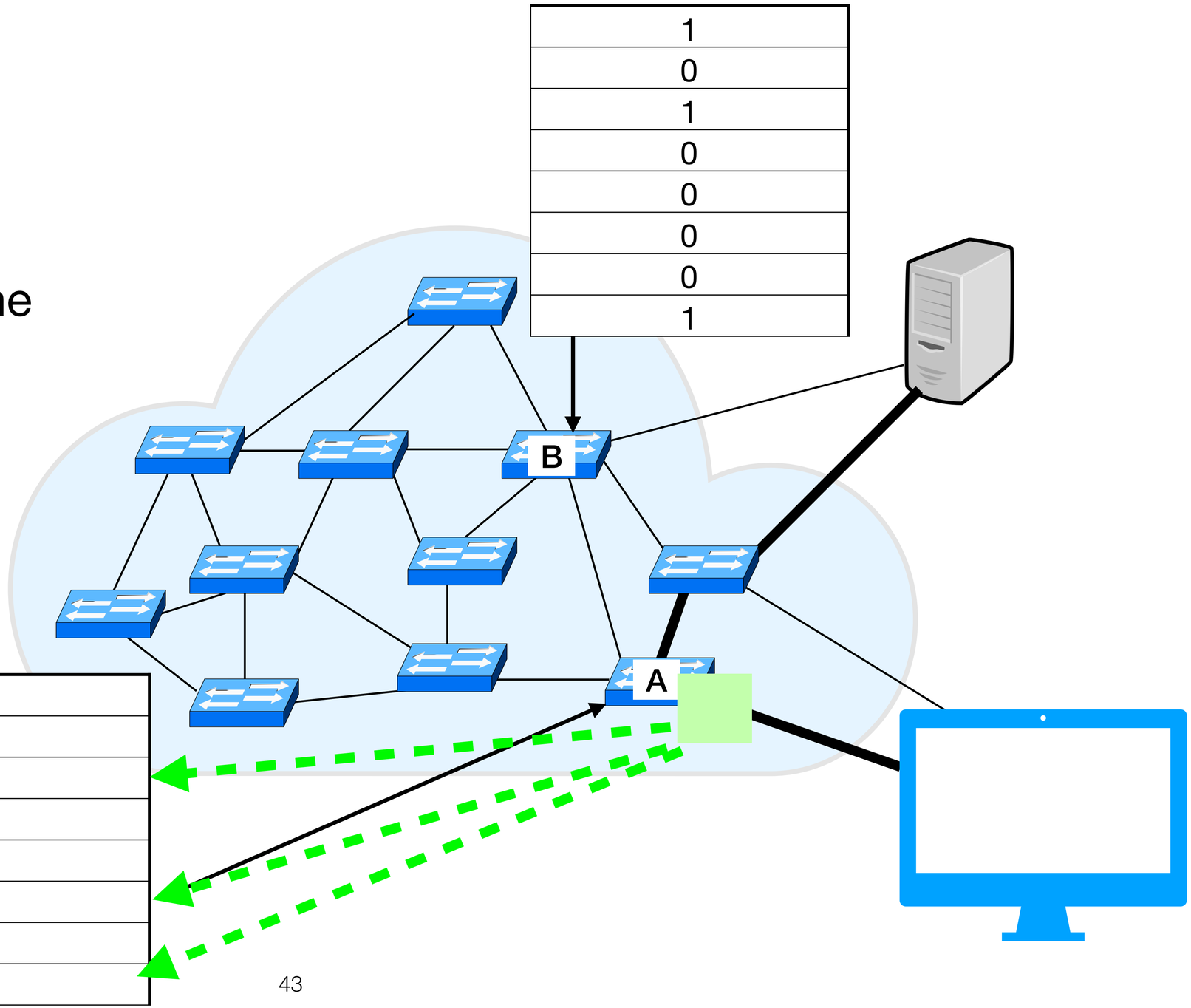
Src IP Prefix	Response switch



ReAct

After a timeout, the requester will retransmit the original request

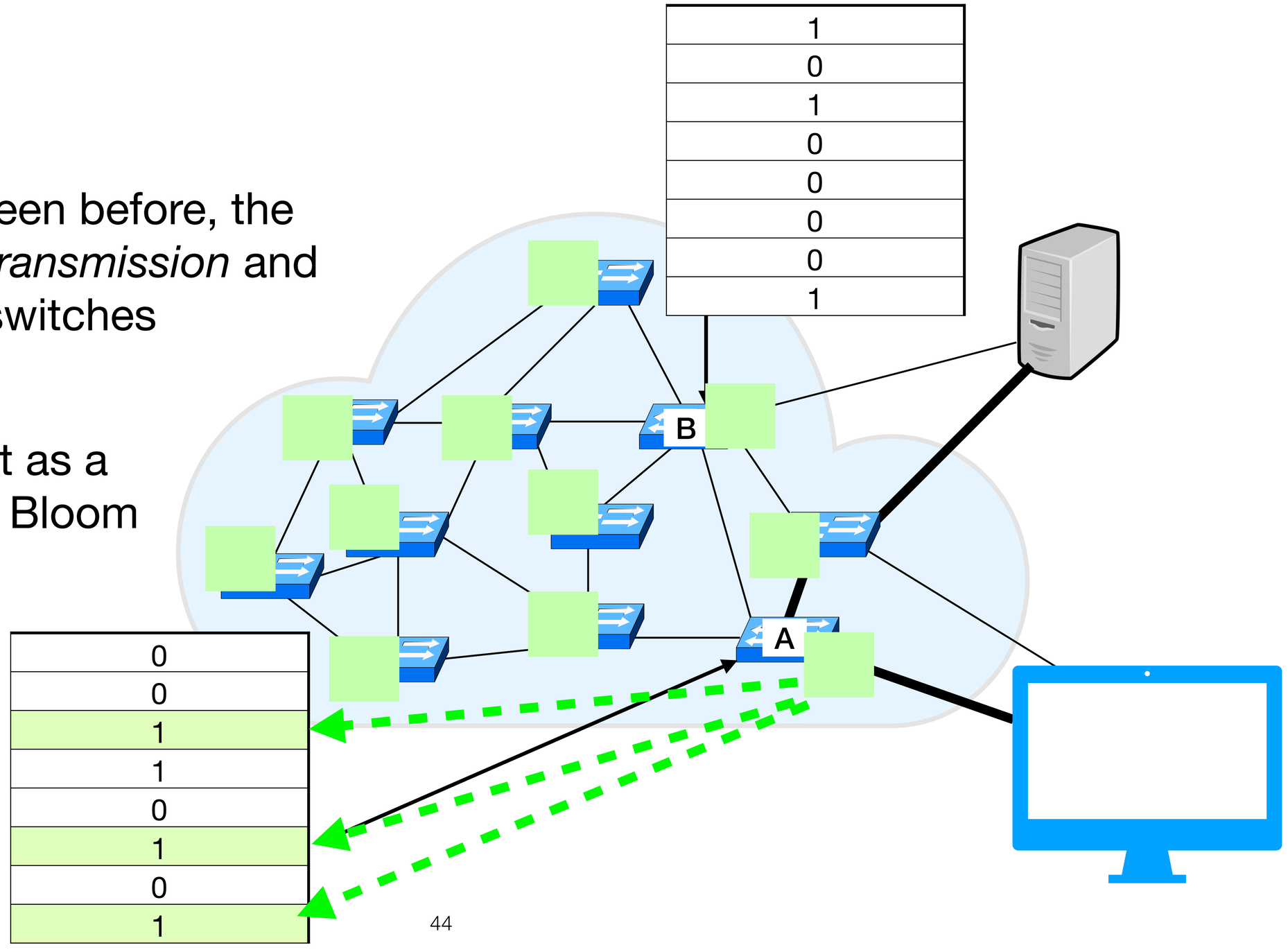
Src IP Prefix	Response switch



ReAct

If the request was seen before, the switch detects a *retransmission* and *broadcasts* it to all switches

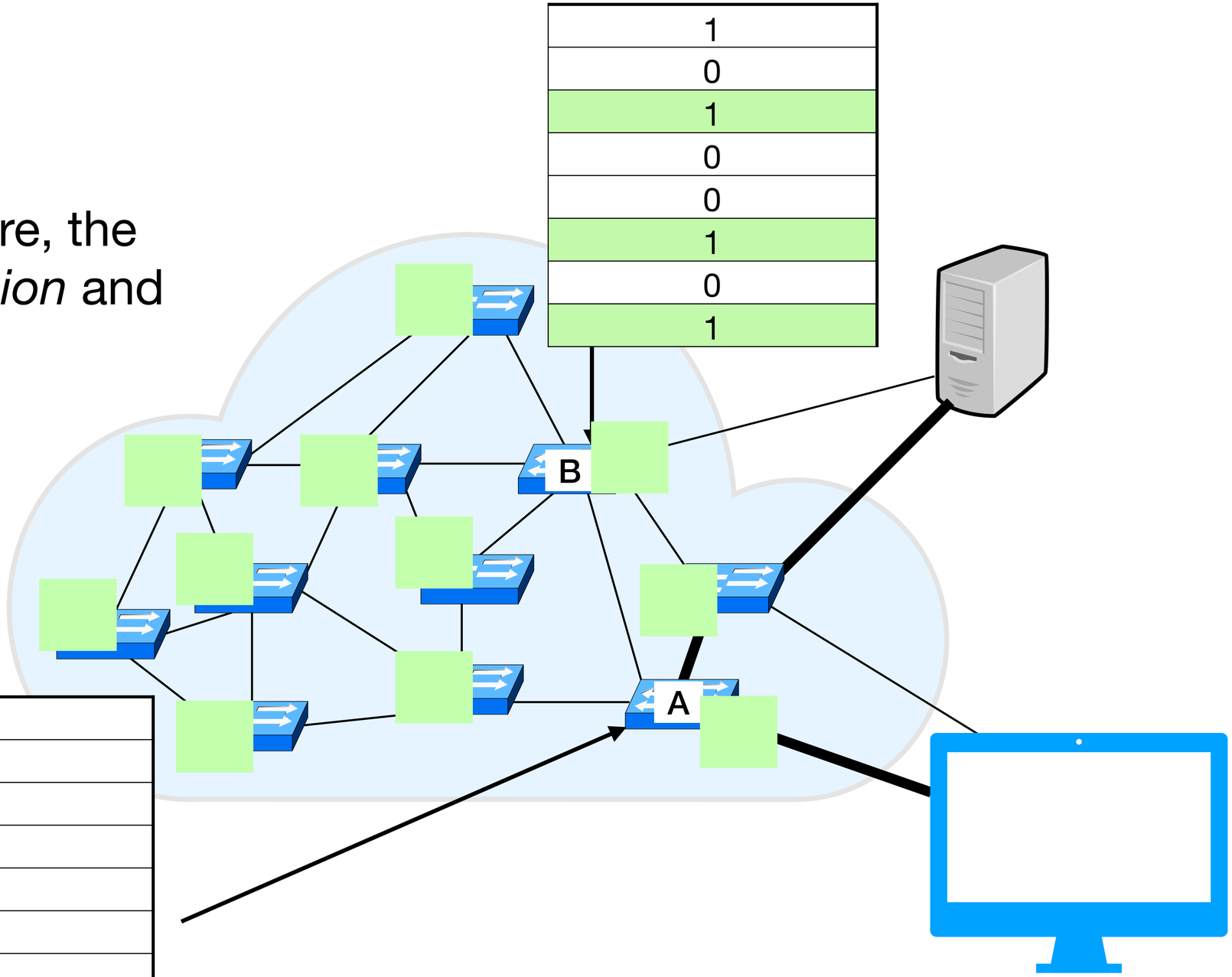
We identify a request as a retransmission if the Bloom filter values are all 1



ReAct

If the request was seen before, the switch detects a *retransmission* and *broadcasts* it to all switches

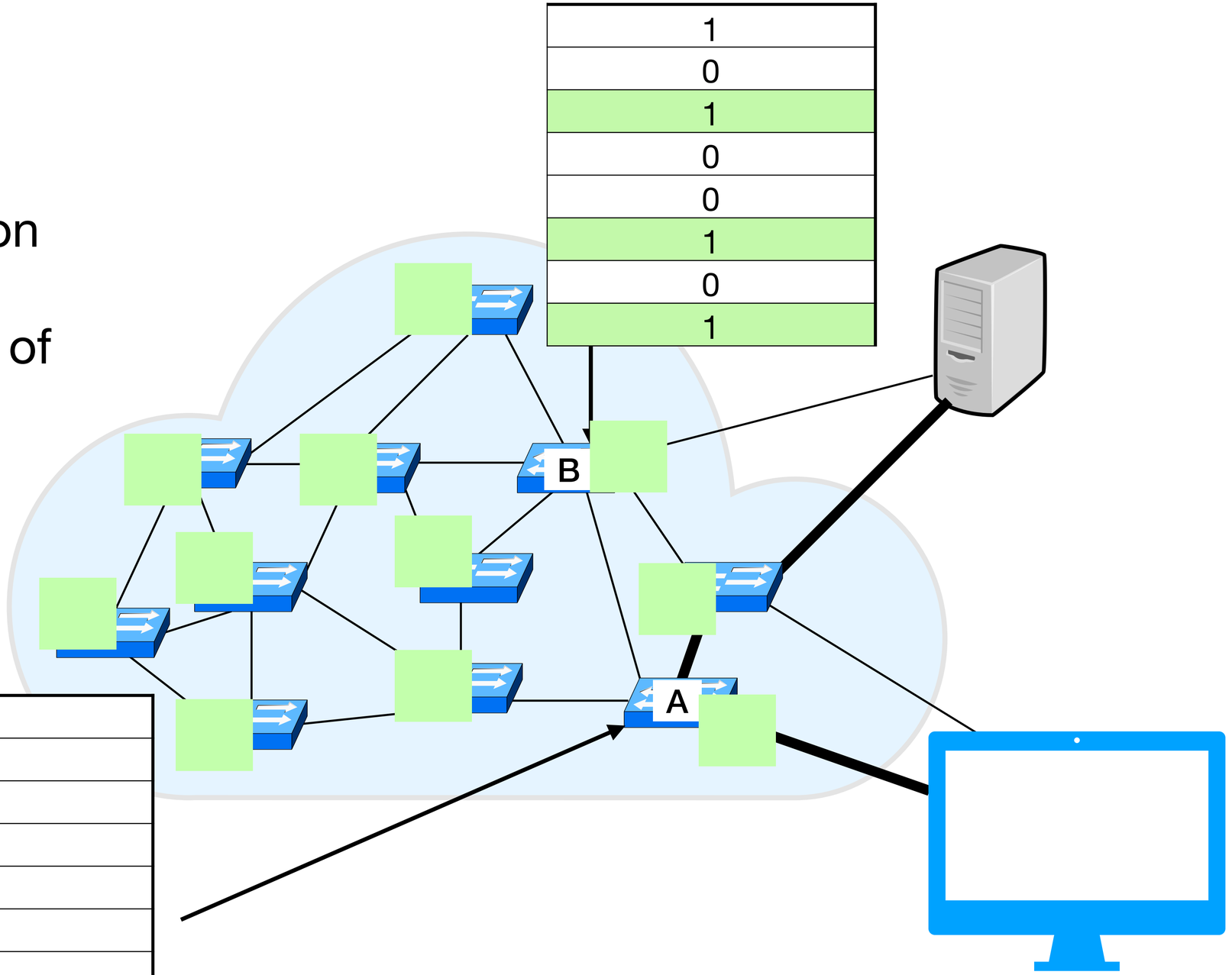
Src IP Prefix	Response switch



ReAct

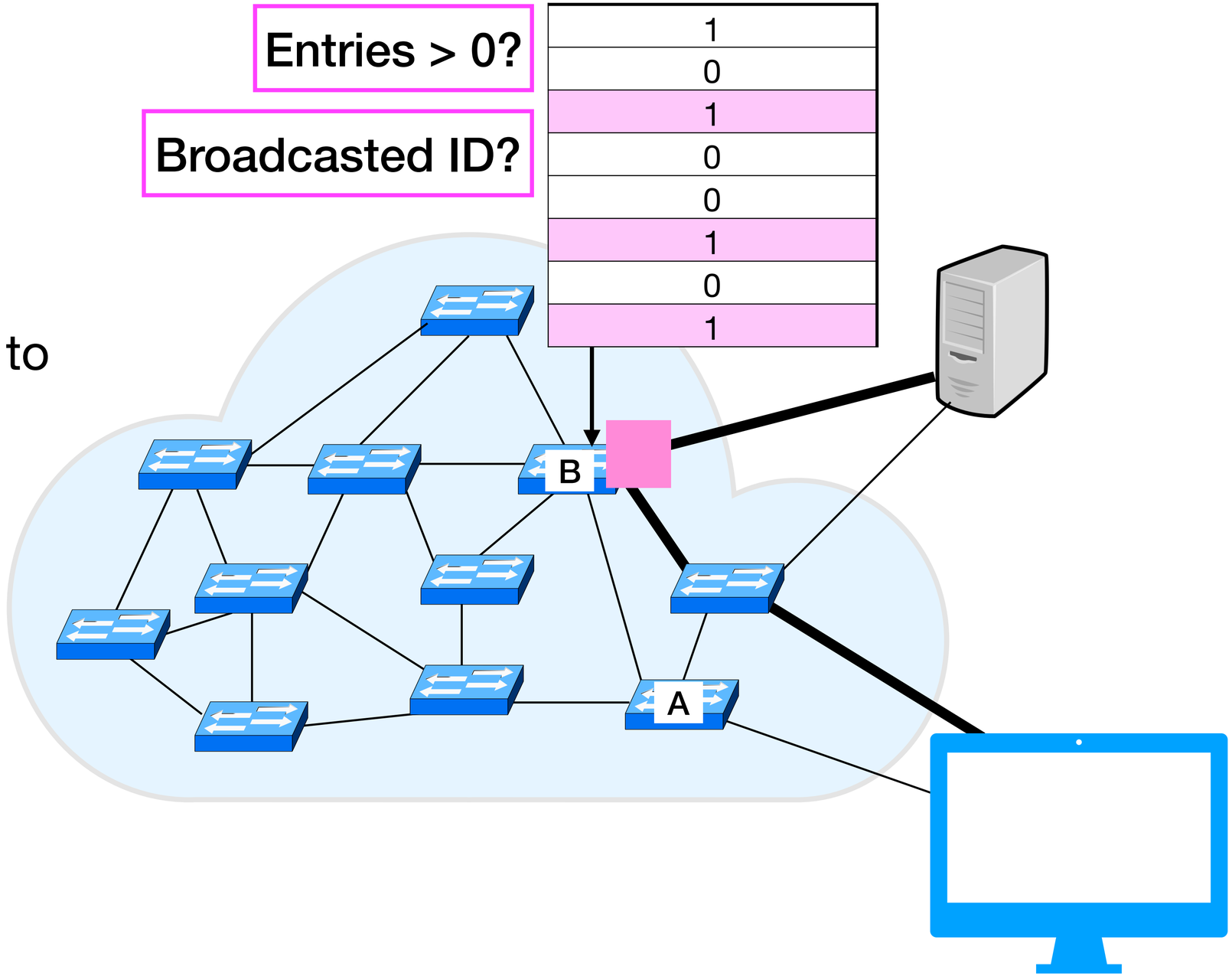
ReAct only broadcasts upon retransmission to avoid unnecessary broadcasting of symmetric traffic

Src IP Prefix	Response switch



ReAct

When a switch receives a response to a broadcasted request, it triggers an update to the broadcasting switch

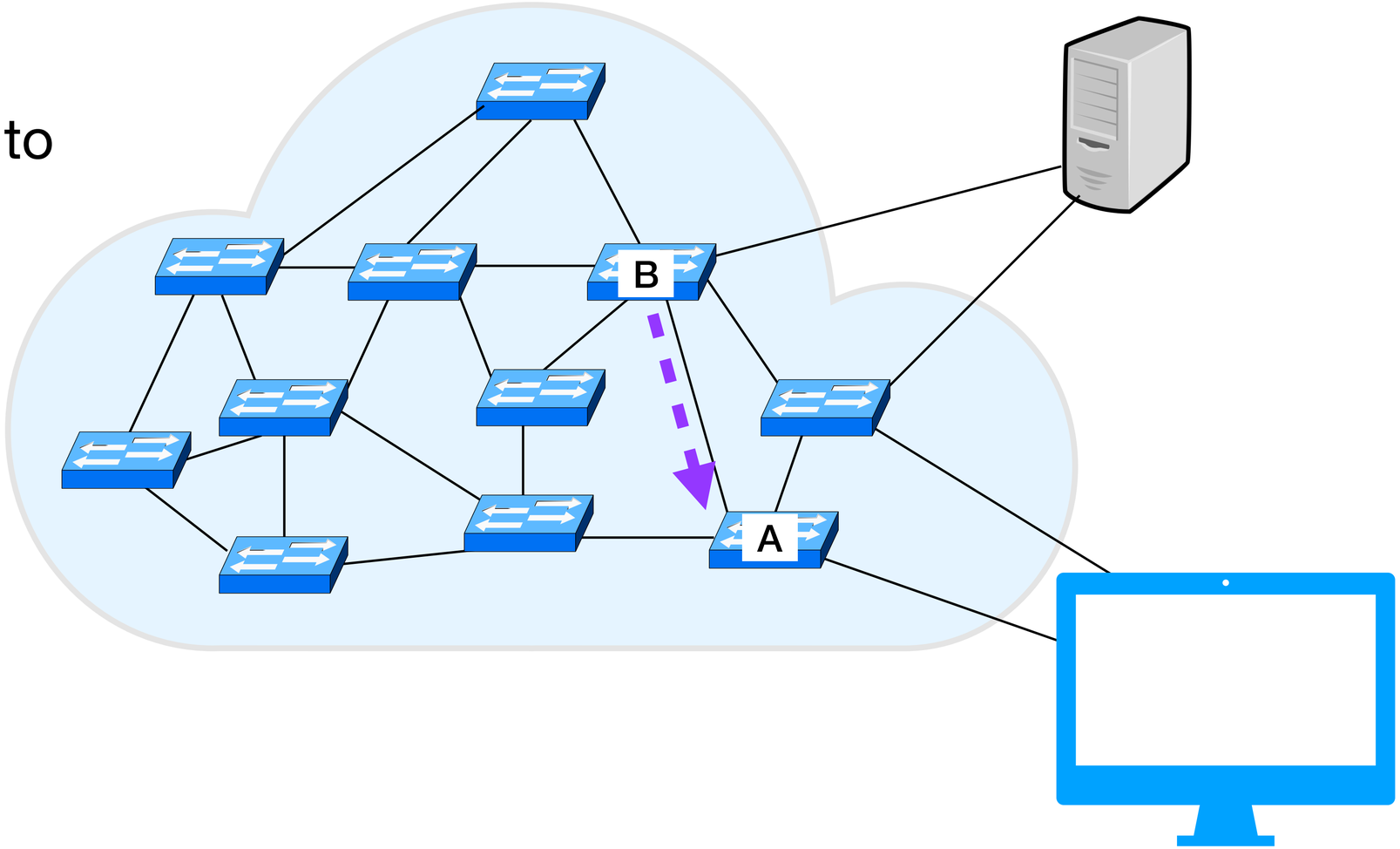


ReAct

When a switch receives a response to a broadcasted request, it triggers an update to the broadcasting switch

Src IP Prefix	Response switch

Record prefix and switch ID

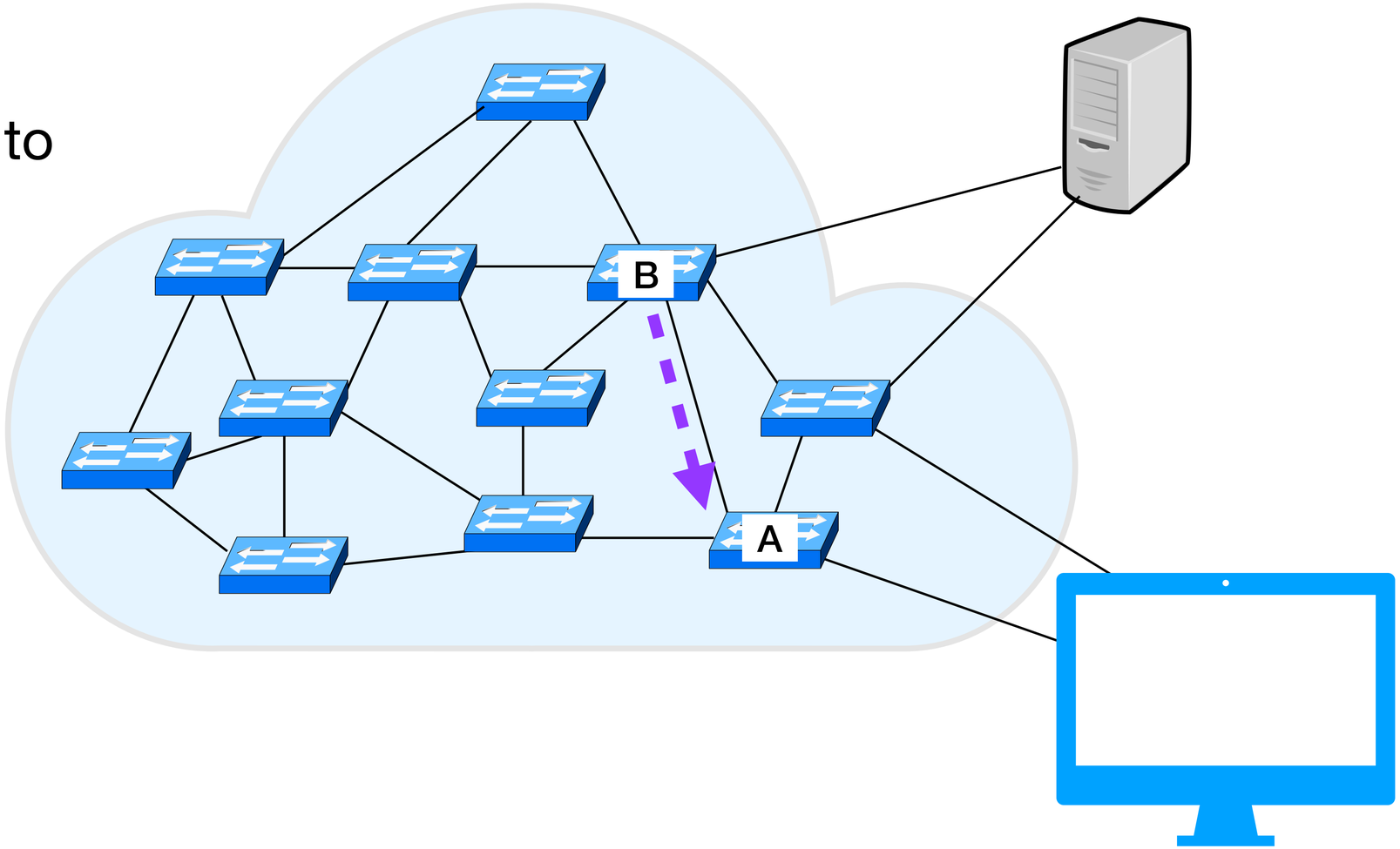


ReAct

When a switch receives a response to a broadcasted request, it triggers an update to the broadcasting switch

Src IP Prefix	Response switch
10.168.0.0/16	B

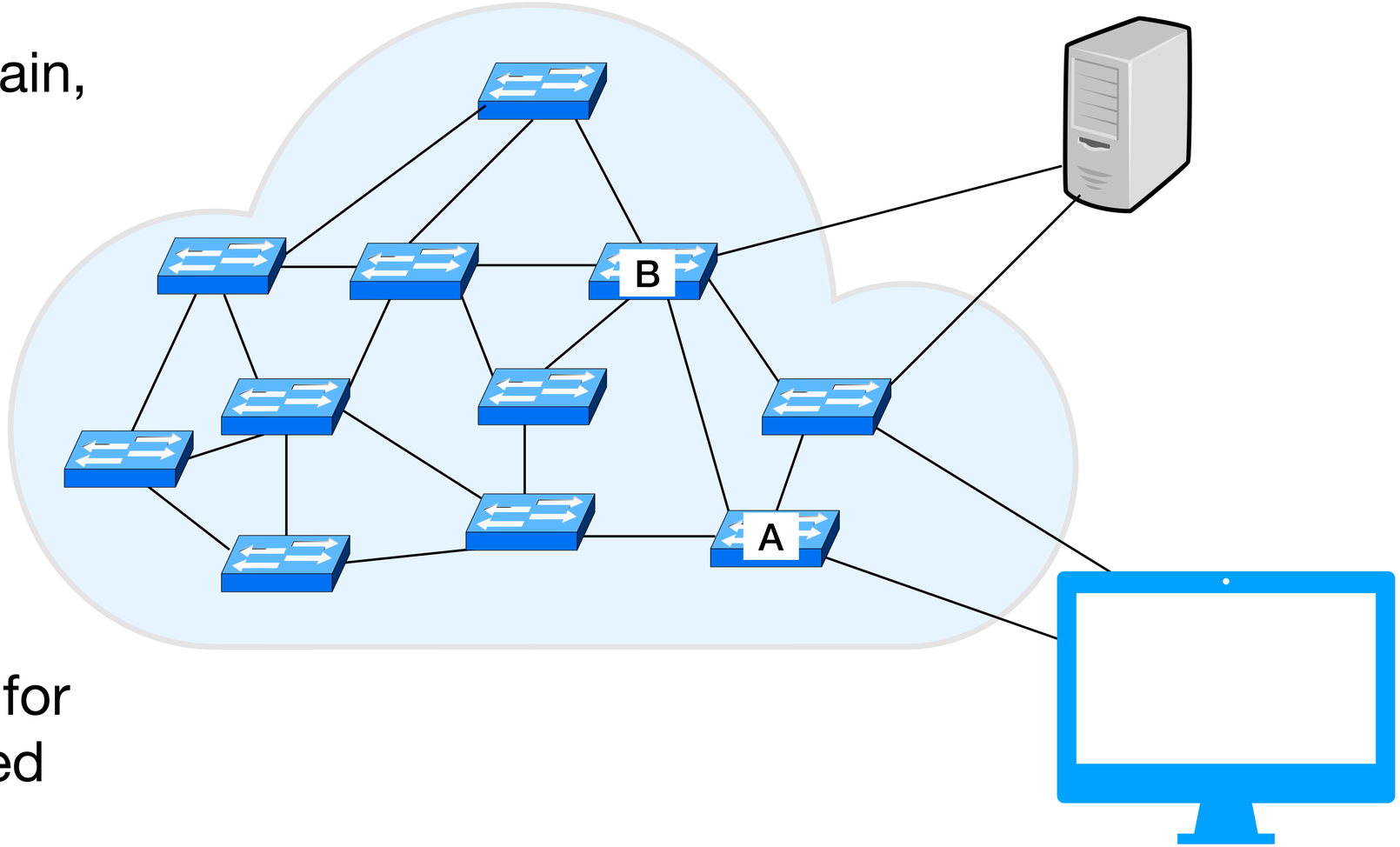
Record prefix and switch ID



ReAct

When a request from the broadcasted prefix arrives again, ReAct does not need to broadcast

Src IP Prefix	Response switch
10.168.0.0/16	B



Future legitimate responses for that prefix will not be dropped

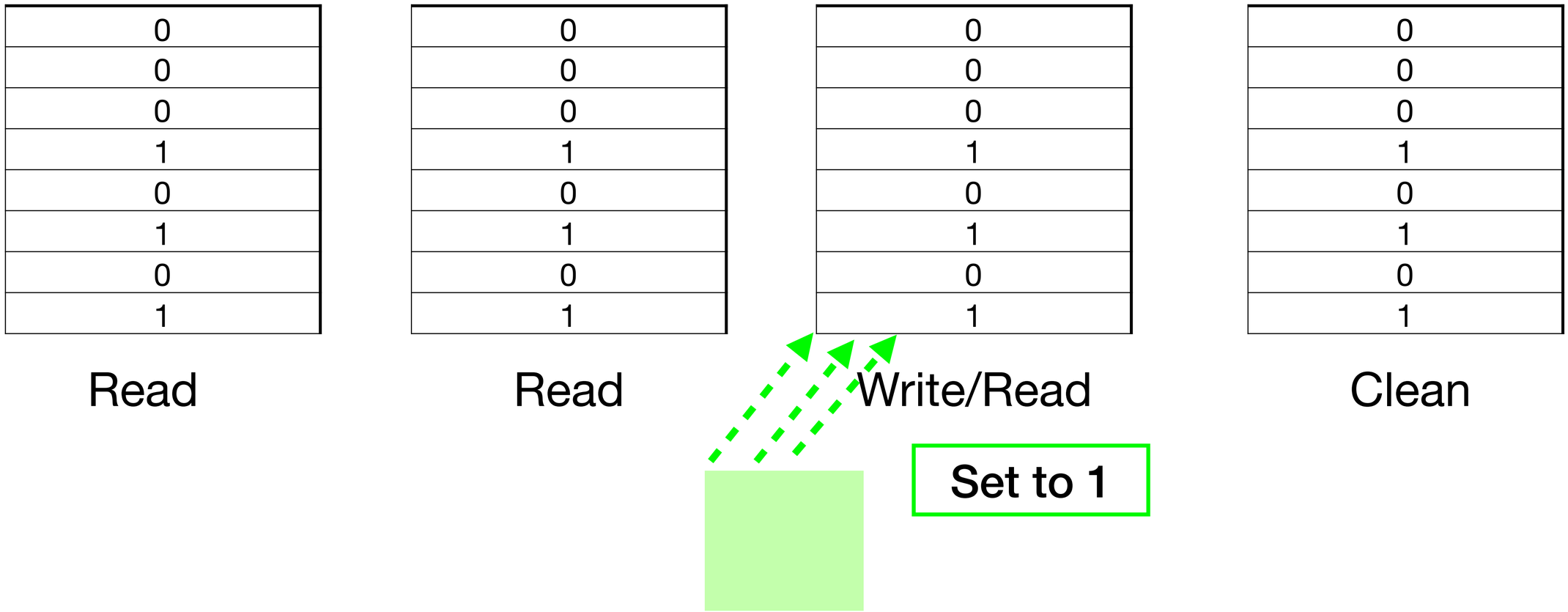
How ReAct is implemented

Broadcast mechanism for sharing requests if we don’t know where responses are processed ahead of time

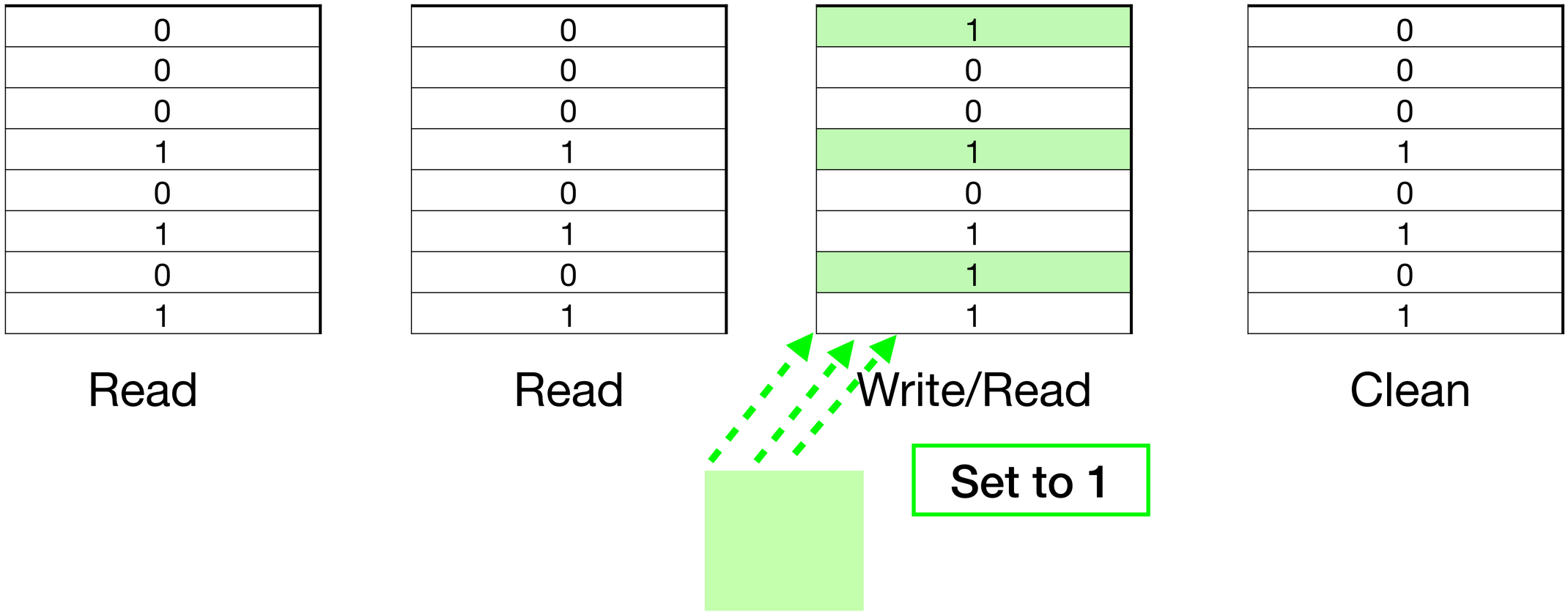
Rotating sliding window of Bloom Filters that are periodically cleaned of stale data to maintain high accuracy rates

Responses don’t update state, so we need an alternative mechanism to remove out-of-date information

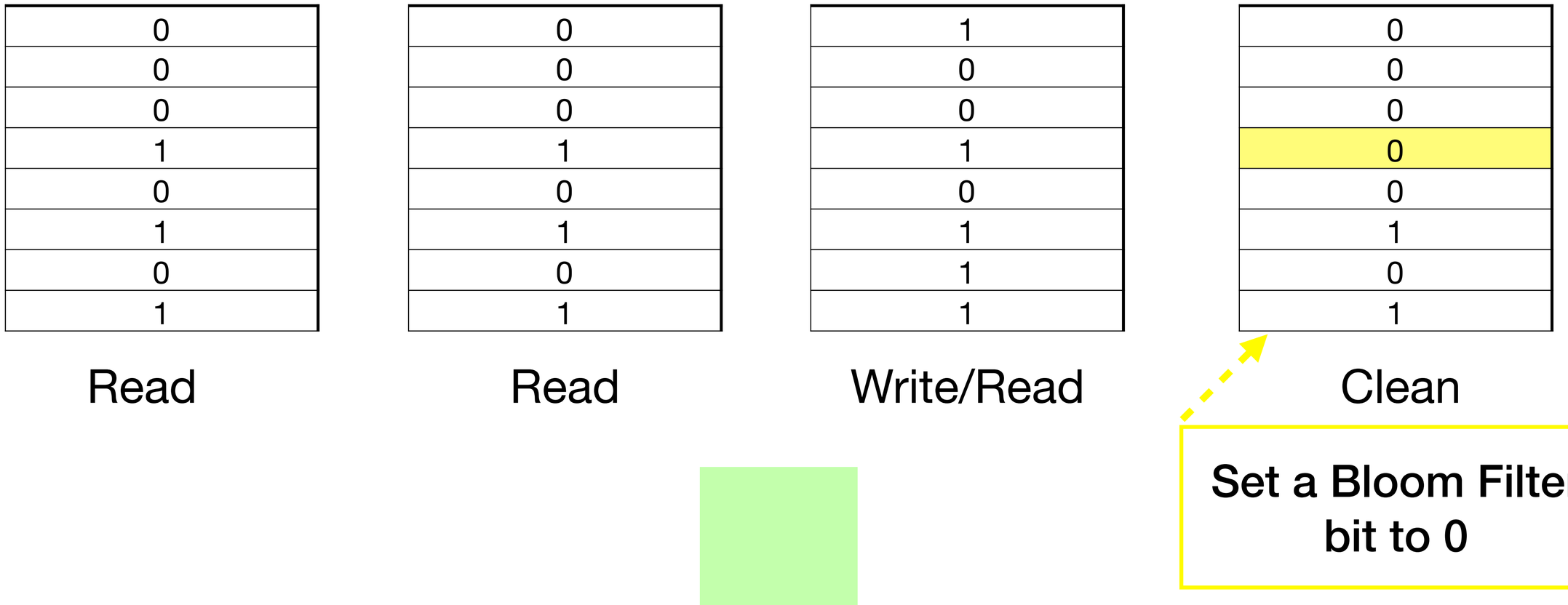
Sliding Window Boom Filters



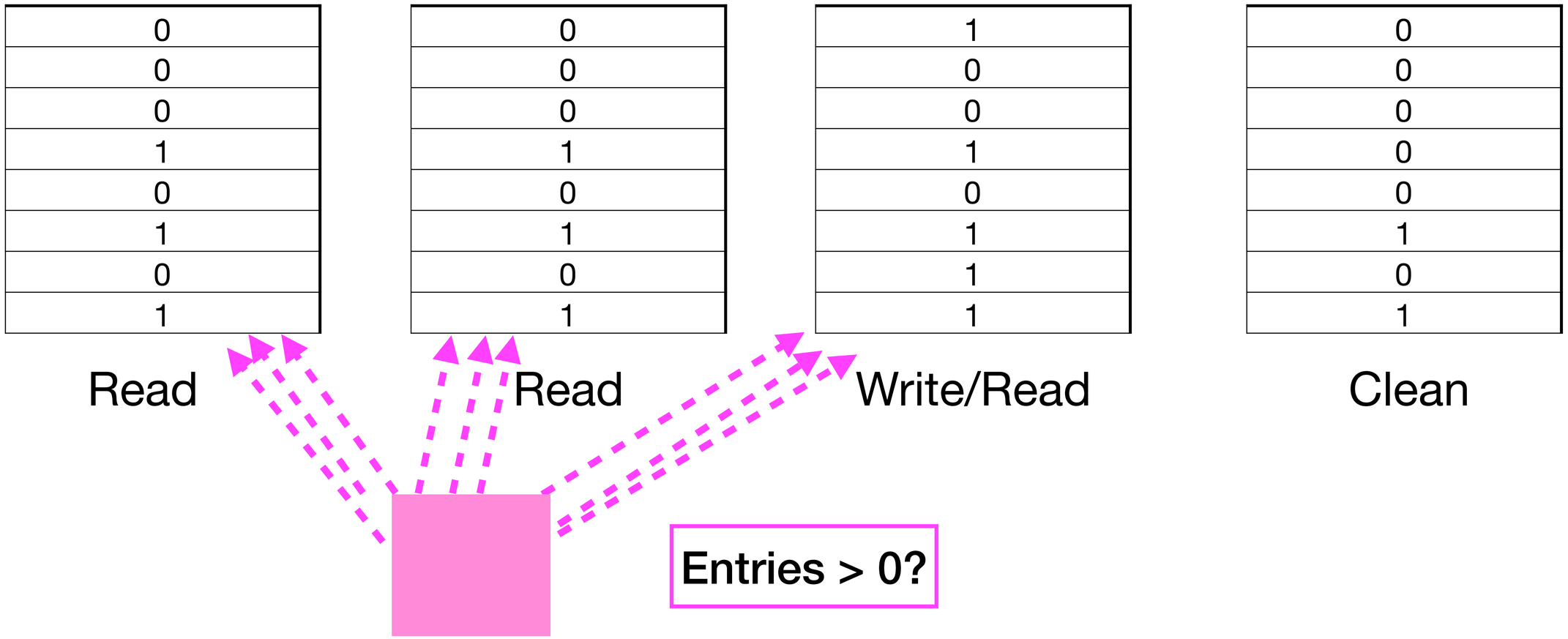
Sliding Window Boom Filters



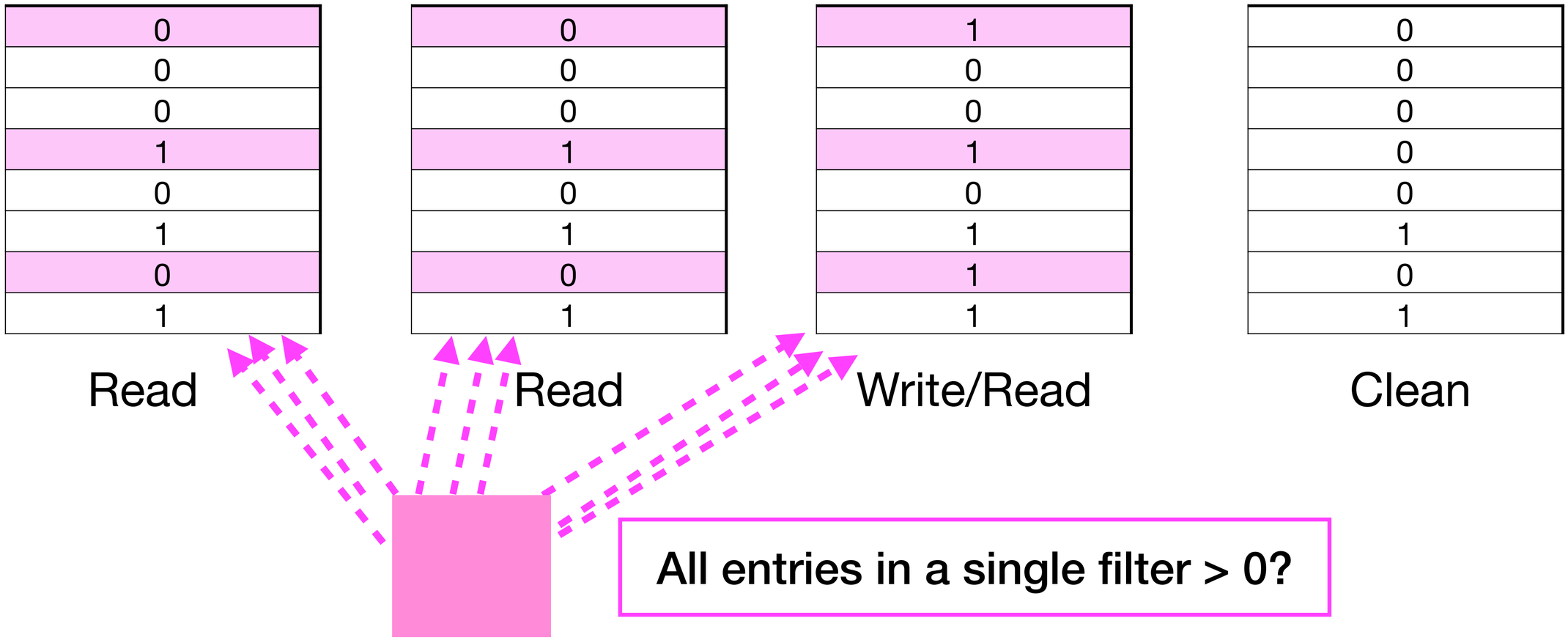
Sliding Window Boom Filters



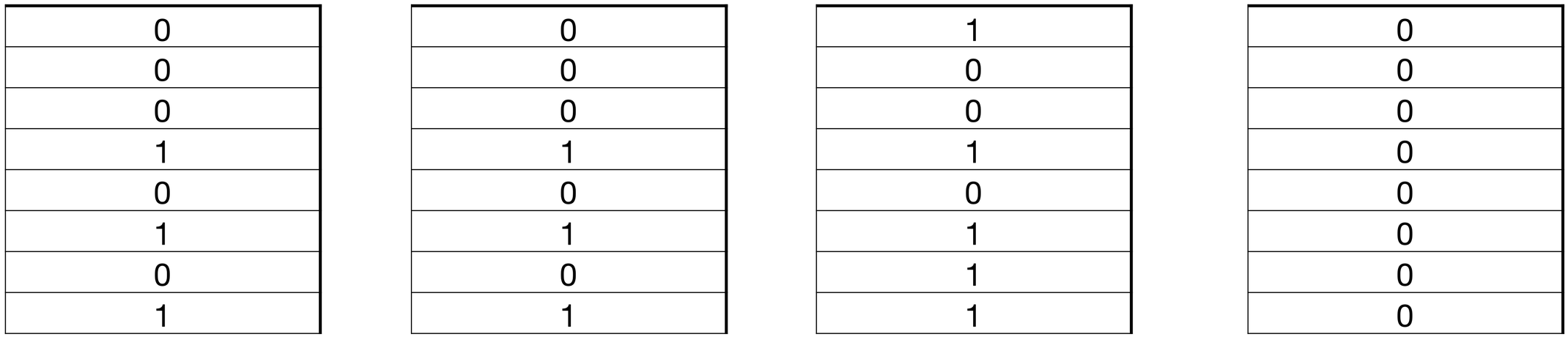
Sliding Window Boom Filters



Sliding Window Boom Filters



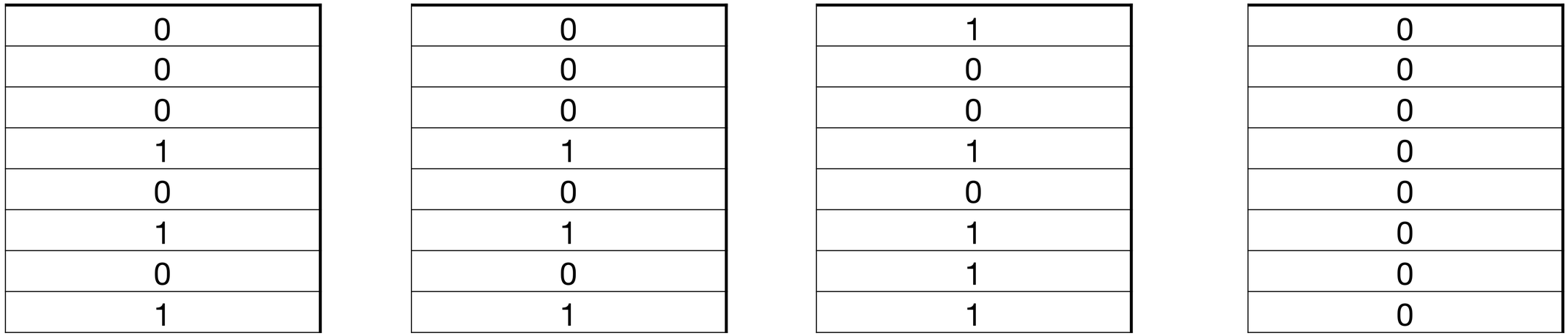
Sliding Window Boom Filters



After every α seconds, we rotate the Bloom filters (4 seconds in our simulations)

Sliding Window Boom Filters

Periodically rotating through the structures allows ReAct to remove stale data without utilizing responses



After every α seconds, we rotate the Bloom filters (4 seconds in our simulations)

ReAct

Overview

Implementation

Evaluation

Summary

Evaluating ReAct

(1) Does ReAct effectively block malicious DNS traffic?

(2) Does ReAct block legitimate traffic?

(3) Does ReAct cause high overhead?

(4) Is ReAct performance consistent under high attack load?

Evaluated on simulated programmable switches

Evaluated on an NVIDIA Bluefield-3

Evaluating ReAct

(1) Does ReAct effectively block malicious DNS traffic?

(2) Does ReAct block legitimate traffic?

(3) Does ReAct cause high overhead?

(4) Is ReAct performance consistent under high attack load?

Evaluated on simulated programmable switches

Evaluated on an NVIDIA Bluefield-3

Evaluating ReAct

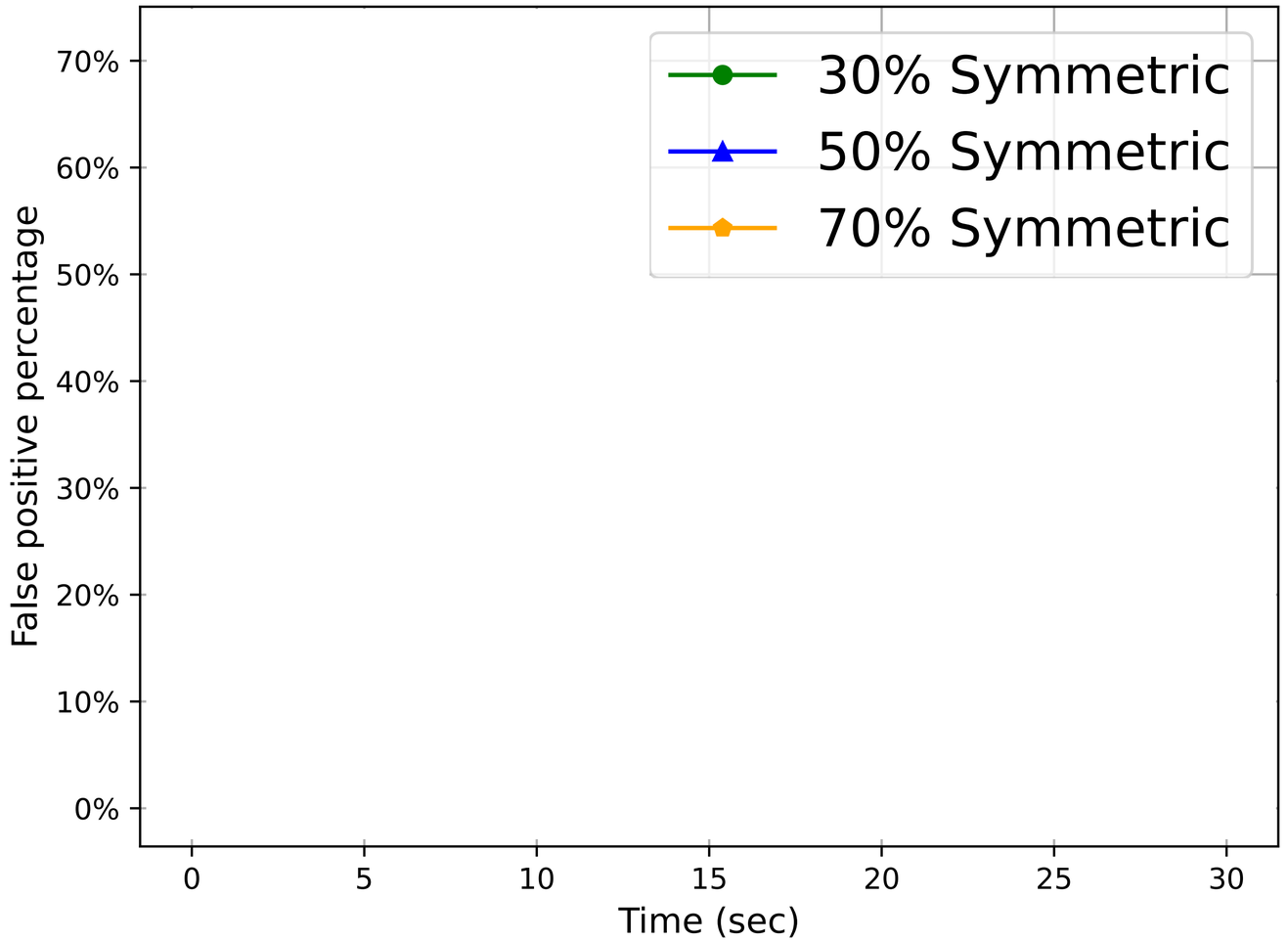
(1) Does ReAct effectively block malicious DNS traffic?

Yes! ReAct blocked over 97% of attack traffic on average in our simulations

Evaluated on simulated programmable switches

Does ReAct block legitimate traffic?

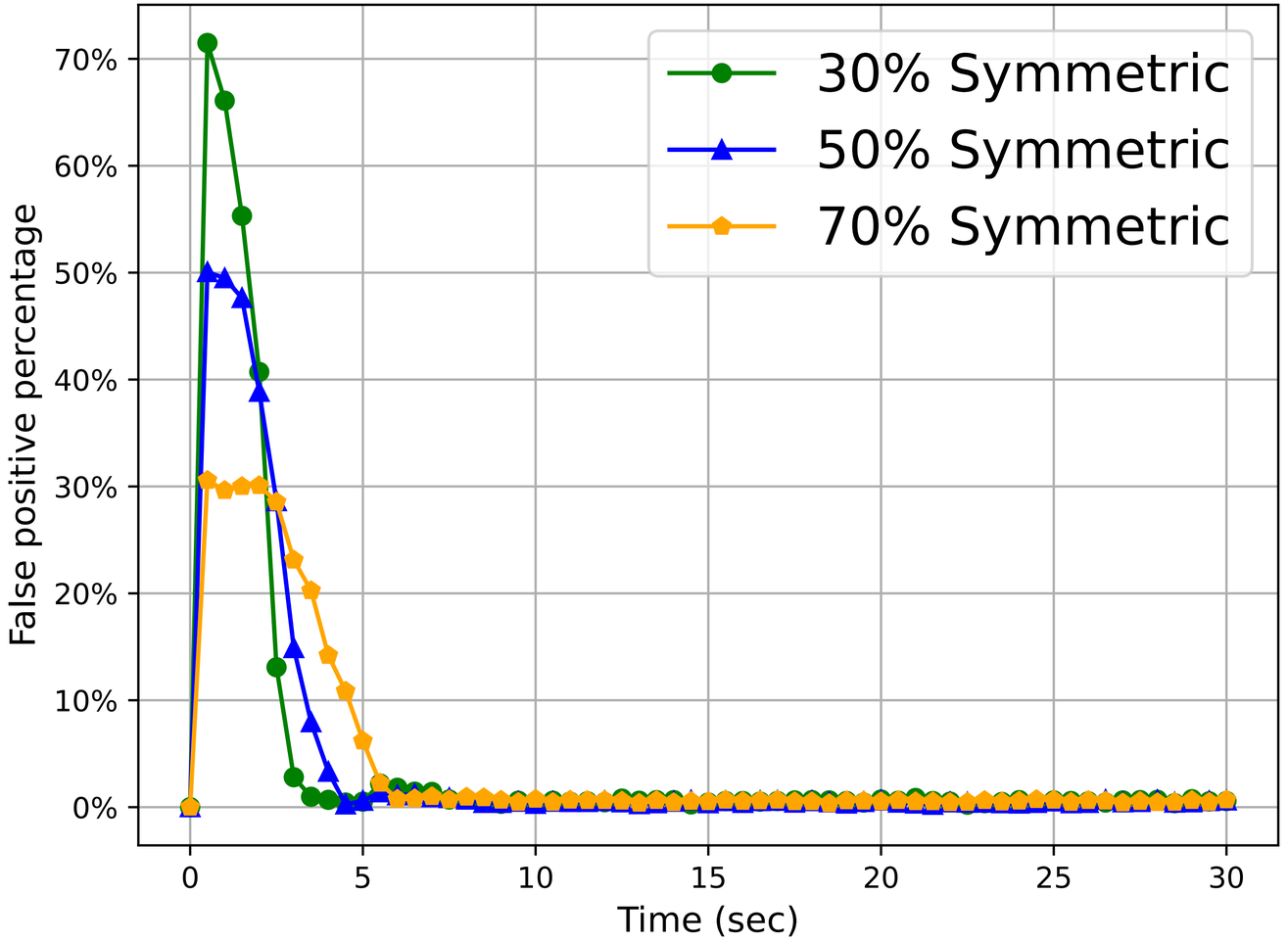
At the beginning, ReAct switches have to learn where responses are processed by broadcasting requests



Does ReAct block legitimate traffic?

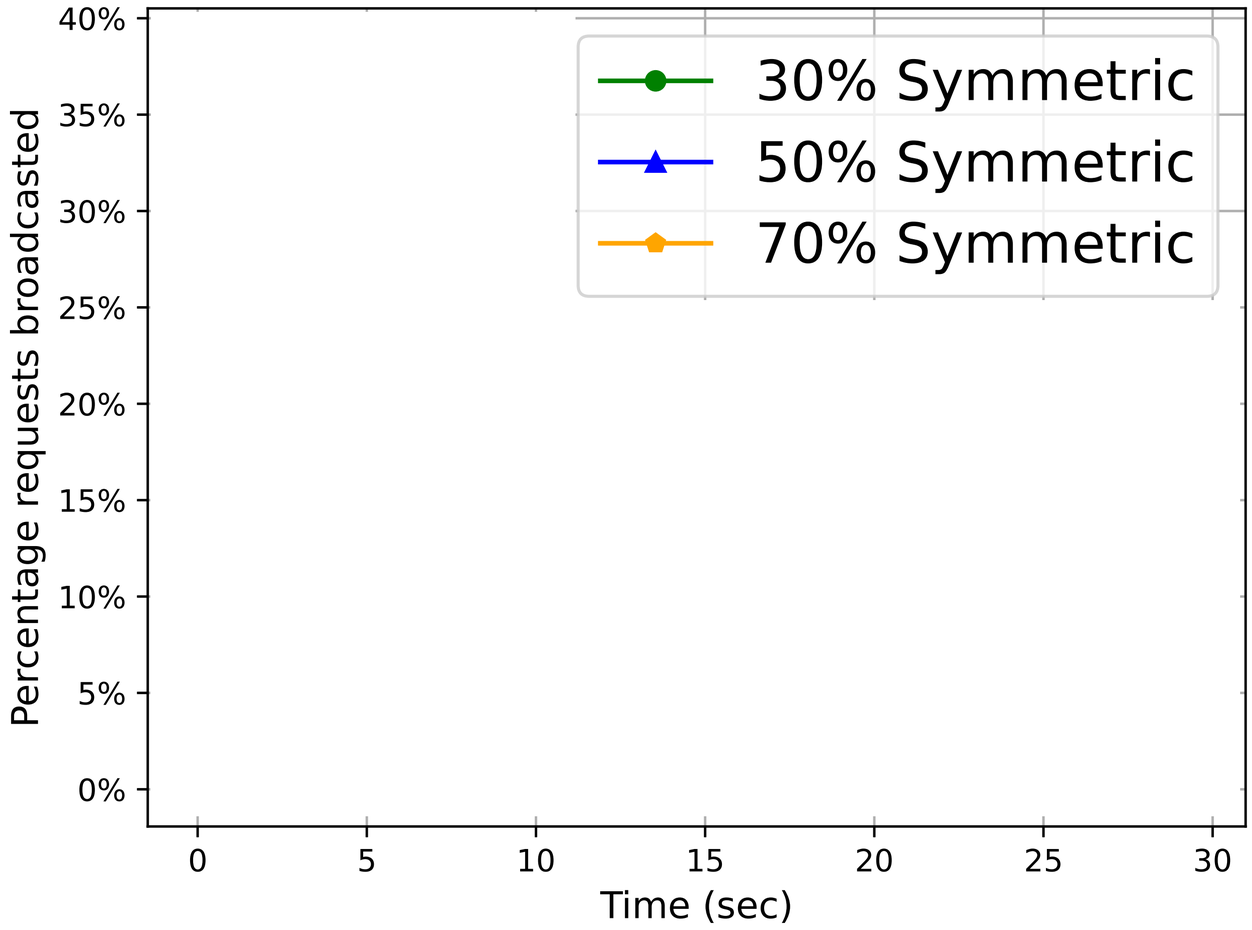
At the beginning, ReAct switches have to learn where responses are processed by broadcasting requests

Initial legitimate asymmetric responses are dropped, but are correctly forwarded after request broadcasting



Does ReAct cause excessive overhead?

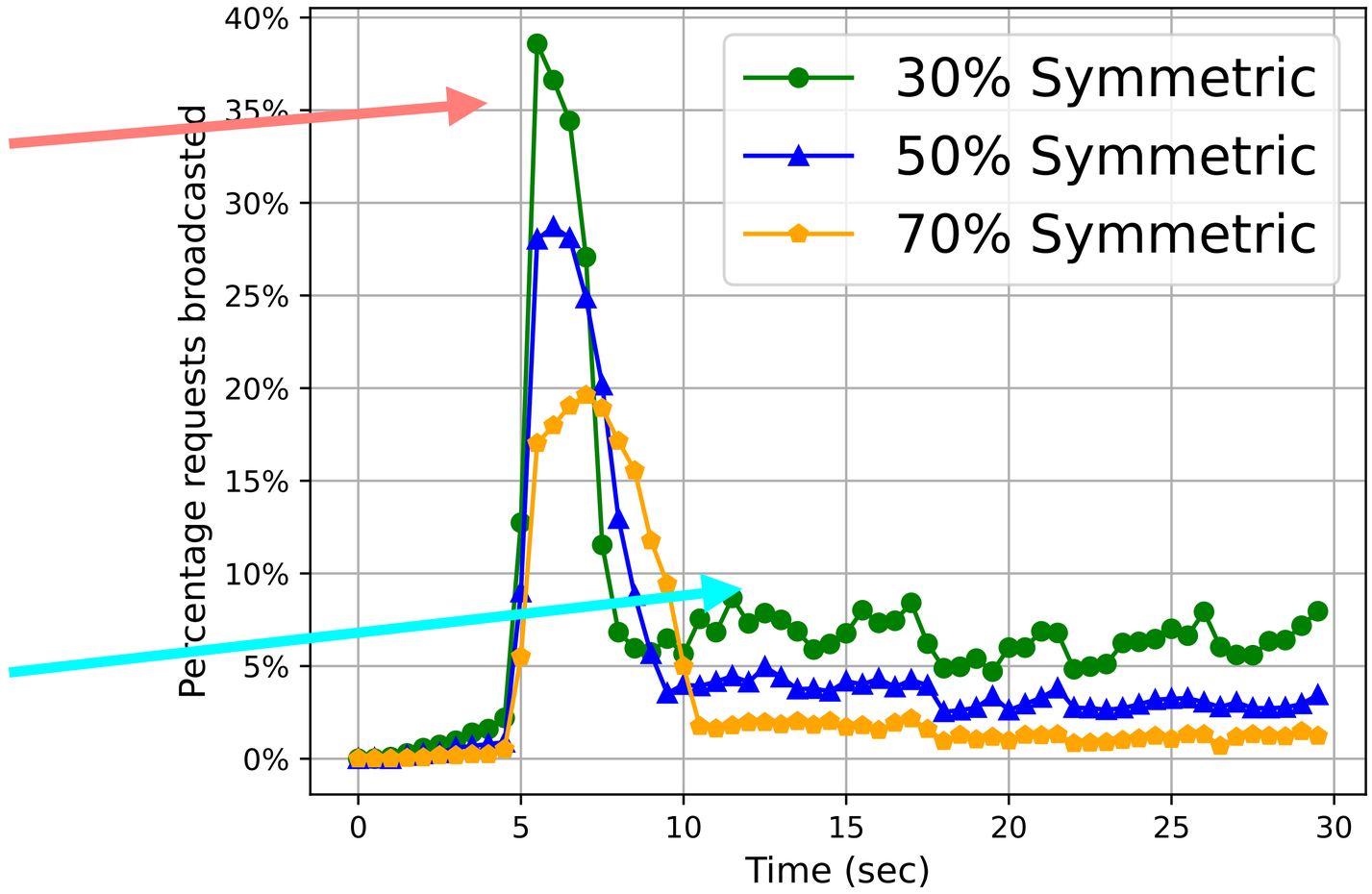
At the beginning, ReAct switches have to learn where responses are processed by broadcasting requests



Does ReAct cause excessive overhead?

While switches are learning, they broadcast up to 40% of requests in our simulations

Once they learn, they need to broadcast < 10% of requests, which are falsely identified as retransmissions due to hash collisions



ReAct

Overview

Implementation

Evaluation

Summary

ReAct: AR-DDoS Mitigation

Solutions that only target symmetric traffic are too risky to implement in real networks because they can cause service disruptions

ReAct is a practical solution for DDoS mitigation that reliably serves both symmetric and asymmetric legitimate traffic

ReAct successfully blocks over 97% of attack traffic on average, regardless of its symmetry

ReAct is open source! We have implementations for the Intel Tofino and NVIDIA Bluefield-3

https://github.com/davidhay1976/react_infocom2026

ReAct: Reflection Attack Mitigation For Asymmetric Routing

David Hay*, **Mary Hogan**^, Shir Landau Feibish+

*Hebrew University, ^Oberlin College, +University of Haifa

OBERLIN

COLLEGE & CONSERVATORY

Simulation Setup - Tofino

We simulate two Intel Tofino switches with the Lucid interpreter

A switch receives 7000 DNS requests per second, with random transaction IDs, source IPs, and destination IPs

An attacker injects 70,000 forged responses per second

Experiments run for 30 seconds, with 4 bloom filters of 2^{17} bits per switch

Simulation Setup - Bluefield-3

Client sends *r* DNS requests per second with random transaction IDs and source ports

Local BIND9 DNS server responds from its cache

Attacker injects *a* forged responses per second with random transaction IDs and destination ports

Experiments run for 1 minute on 14 ARM cores in the BF-3, each with two filters of sizes 2^{13} bits

Does ReAct stop attacks?

% of symmetric traffic	Average % of attack traffic stopped
30%	
50%	
70%	

Does ReAct stop attacks?

% of symmetric traffic	Average % of attack traffic stopped
30%	**97.5%**
50%	**97.5%**
70%	**97.9%**

ReAct effectively stops over 97% of attack traffic, regardless of traffic symmetry

Limitations + Future Work

We don’t always have a unique ID maintained across retransmissions

- If we don’t know where the response was processed, we ask all measuring devices if they’ve seen a response from that source

ReAct can add extra delay because of retransmissions, especially at the beginning, while learning where requests are processed

- But if we reduce the number of retransmissions, it introduces extra overhead by inducing extra coordination between measurement devices
- We need to carefully time this coordination to make sure it only happens when the network isn’t already busy
- The control plane can also aid in coordination, to reduce data plane overhead

ReAct can add extra delay because of retransmissions, especially at the beginning, while learning where requests are processed