dFence: Transparent Network-based Denial of Service Mitigation

Ajay Mahimkar, Jasraj Dange,
Vitaly Shmatikov, Harrick Vin, Yin Zhang

University of Texas at Austin
mahimkar@cs.utexas.edu

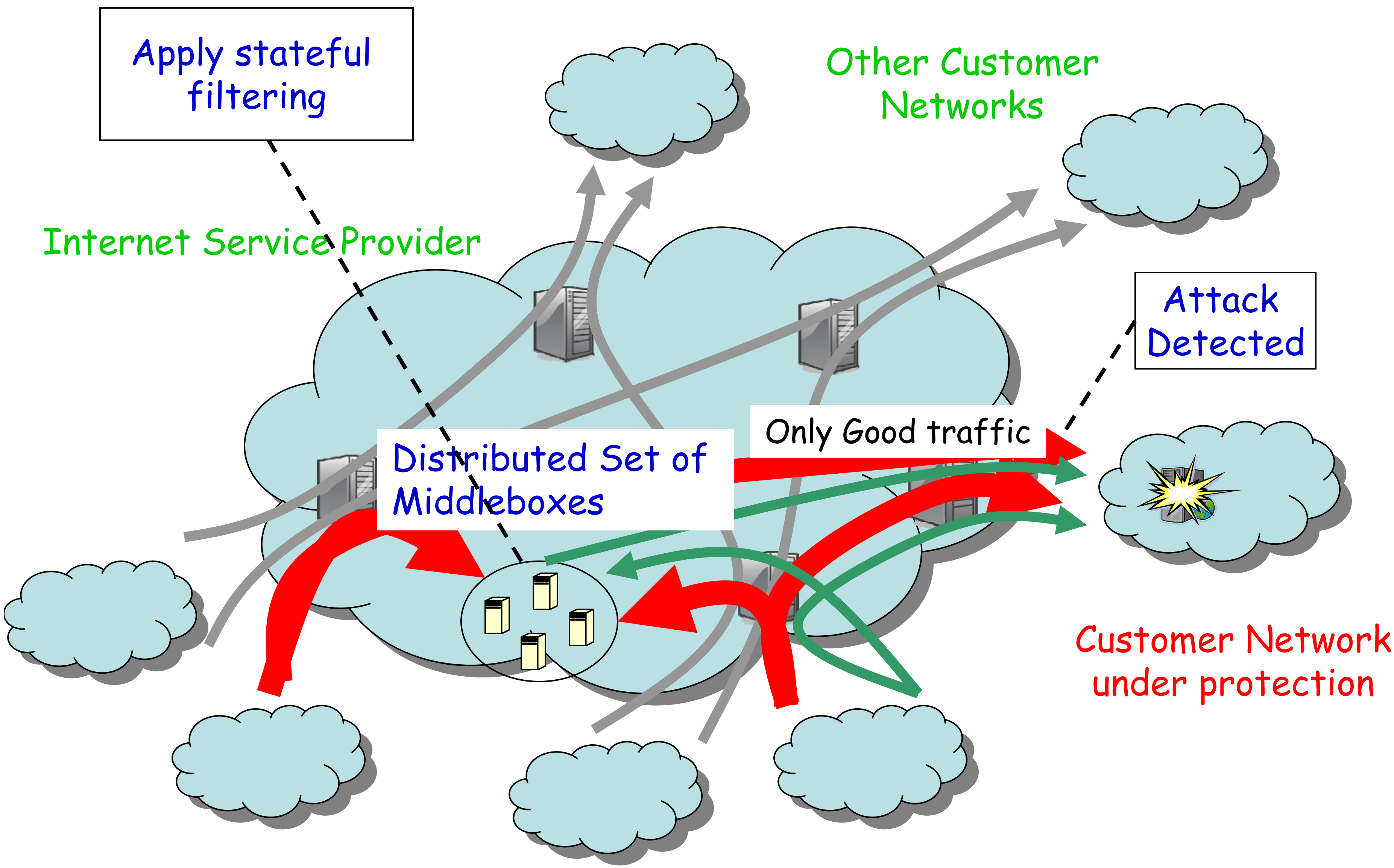
The Problem

- Denial of Service (DoS) attacks
  - A significant threat to Internet reliability & availability
  - Many forms – SYN flood, Data flood, NAPTHA, HTTP request flood, Botnet
- Lots of research and commercial products
  - Speak-up, SIFF, Kill-botz, TVA, Pushback, Cisco Guard, Arbor, …
- Yet, lots of attacks still out there
  - Feb 6. 2007 DDoS attack on 6 of 13 root DNS servers
  - Domain registrar GoDaddy.com was DDoSed (March 2007)

dFence Principles

- Transparency
  - No software modifications to end-hosts or routers
- In-Network defense
  - Filter attack traffic before it gets close to server
- Shared on-demand infrastructure
  - Multiplex defense resources to protect multiple customers
  - No performance penalty during peace time
- Stateful mitigation
  - Necessary for effective defenses against a broad range of DoS attacks

dFence Overview



Challenges

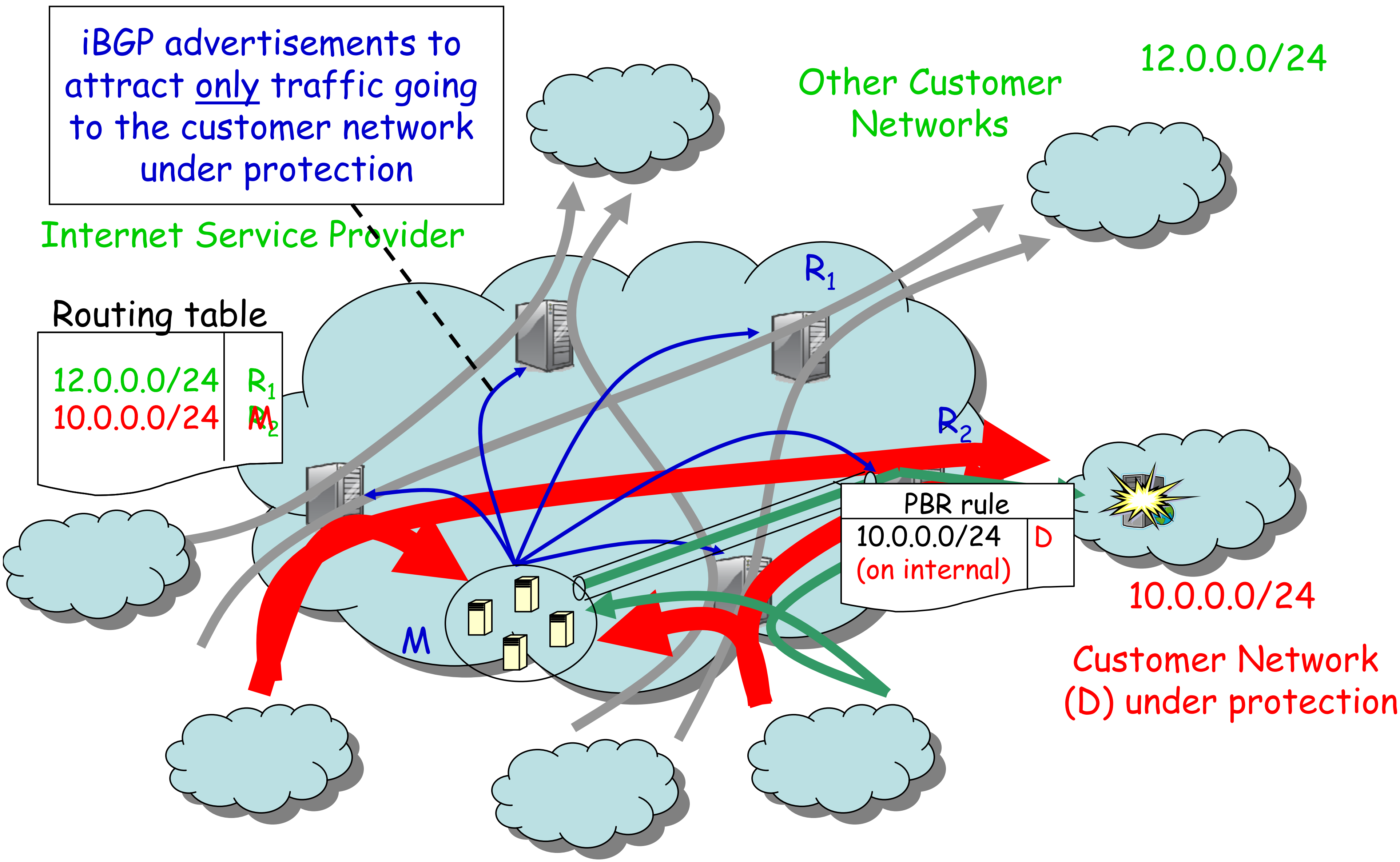
- Bidirectional Traffic Interception
- Attack Mitigation Functionality
- Dynamic State Management
- Robustness to route changes, failures and DoS attacks on middleboxes

Outline

- **Bidirectional Traffic Interception**
- Attack Mitigation Functionality
- Dynamic State Management

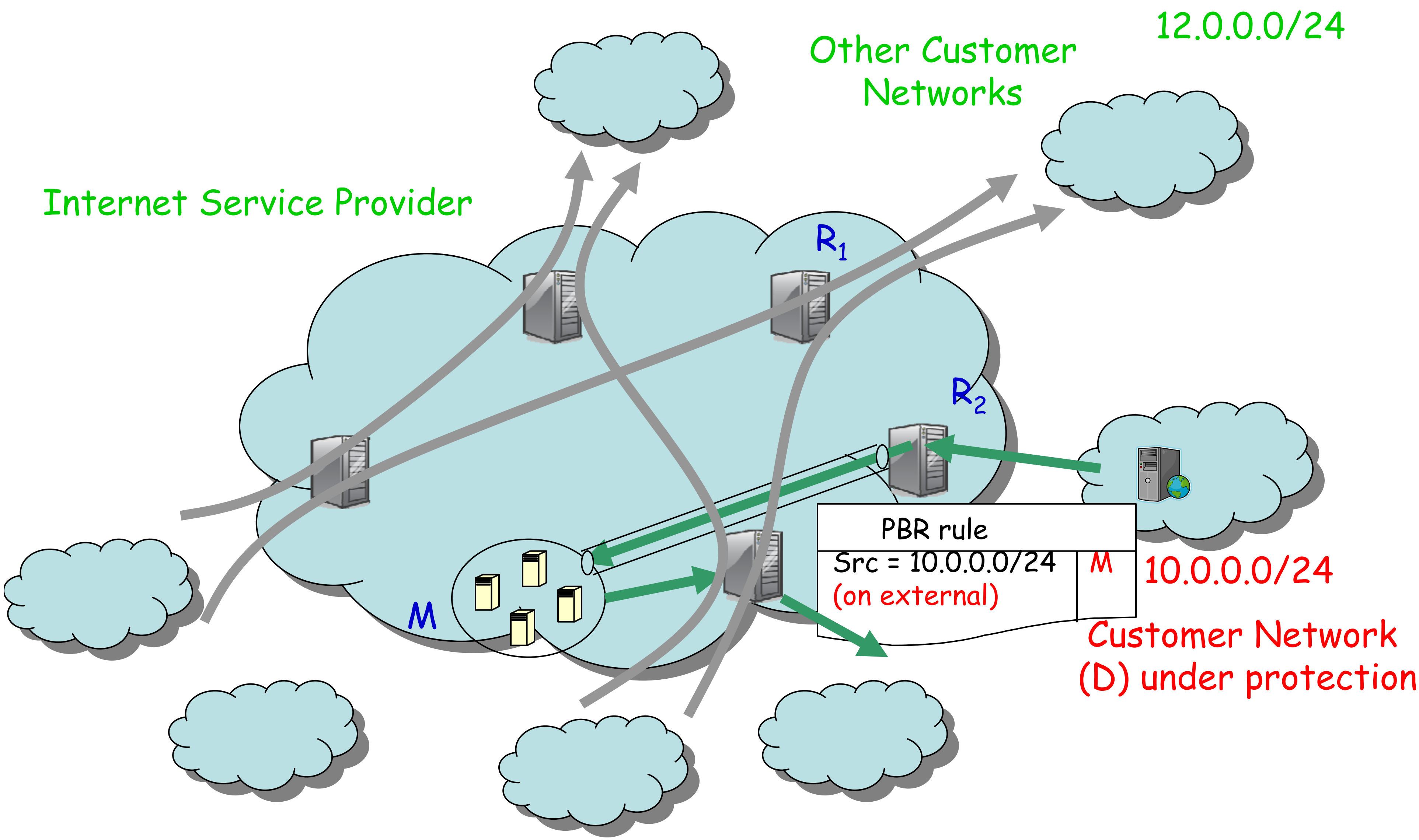
Inbound Traffic Interception

Inbound Traffic Interception



Outbound Traffic Interception

Outbound Traffic Interception



Outline

- Bidirectional Traffic Interception
- **Attack Mitigation Functionality**
- Dynamic State Management

Attack Mitigation at Middlebox

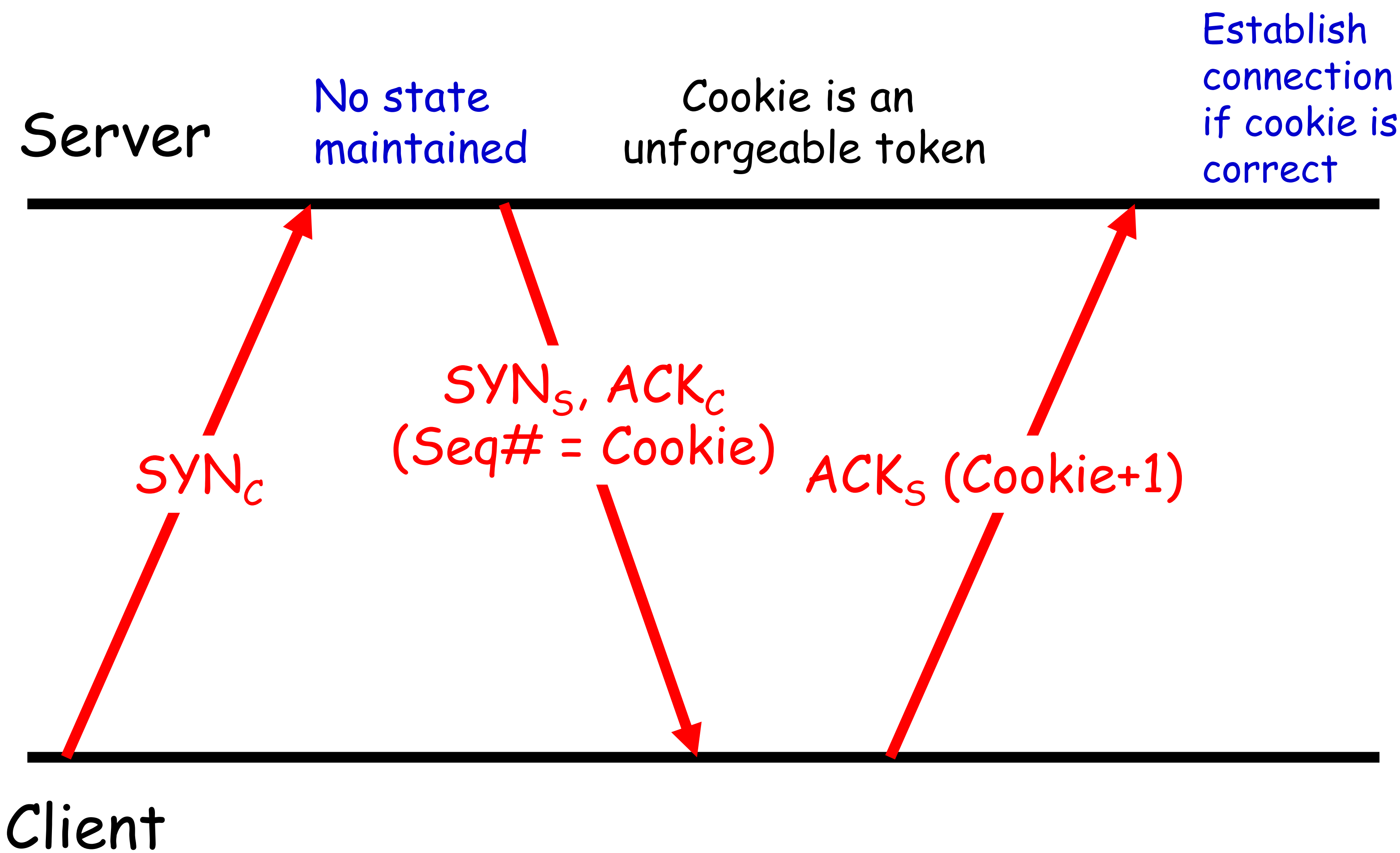
- Stateful policies are a good match for TCP-based attacks
- Careful creation of minimal state for connections

Attack Classification	Attack Examples	State Requirement
Spoofed	Spoofed SYN Spoofed TCP data Reflector attacks	Zero
Un-spoofed mis-behaving	NAPTHA Un-spoofed data flood	Temporary
Un-spoofed well-behaving	Normal traffic	Life-time of connection

An Example Policy

- Mitigating Spoofed Attacks
  - SYN flood: exhaust server resources by flooding it with bogus SYN requests
  - Network-based SYN cookie generation
  - Advantages over server-side
    - Transparency
    - Multiplexing

SYN Cookie [D. Bernstein]

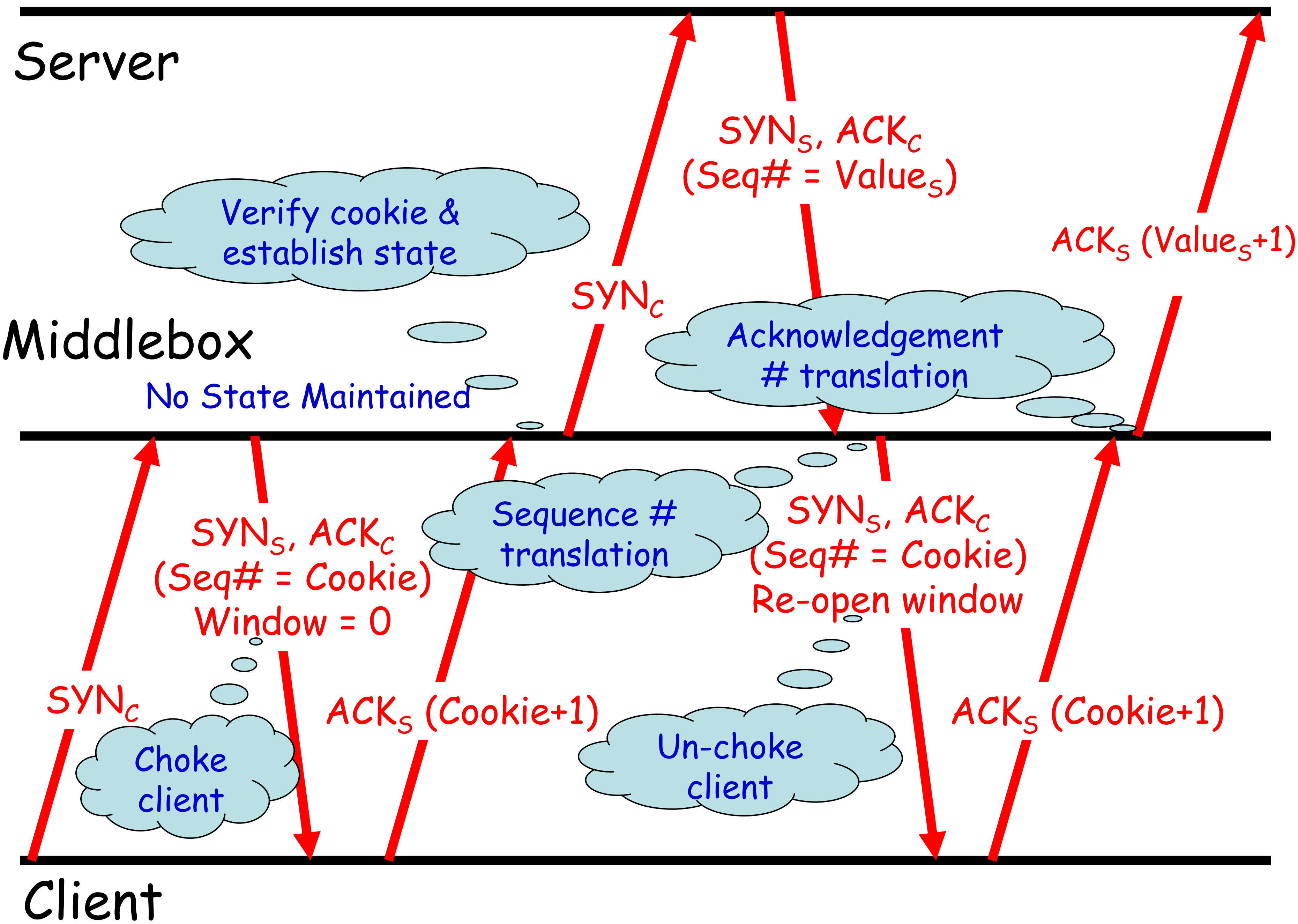


Network-based SYN Cookie

- Challenges
  - How to handle mismatch in sequence number generated by middlebox and server
  - How does middlebox handle data received from clients before its handshake with server is complete

What does <u>not</u> work

- Full TCP splicing with address / port / sequence / acknowledgement number translations
  - Increases state requirement at middlebox
  - Adds more processing burden
- Buffer data packets till handshake with server is complete
  - Opens door to another DoS attack
- Drop data packets till handshake with server is complete
  - Client enters TCP time-out and suffers 3 second delay


Server
Middlebox
Client
Verify cookie & establish state
No State Maintained
SYN_C
SYN_S, ACK_C (Seq# = $Value_S$)
ACK_S ($Value_S$+1)
Acknowledgement # translation
SYN_C
SYN_S, ACK_C (Seq# = Cookie) Window = 0
Choke client
ACK_S (Cookie+1)
Sequence # translation
SYN_S, ACK_C (Seq# = Cookie) Re-open window
Un-choke client
ACK_S (Cookie+1)

Outline

- Bidirectional Traffic Interception
- Attack Mitigation Functionality
- **Dynamic State Management**

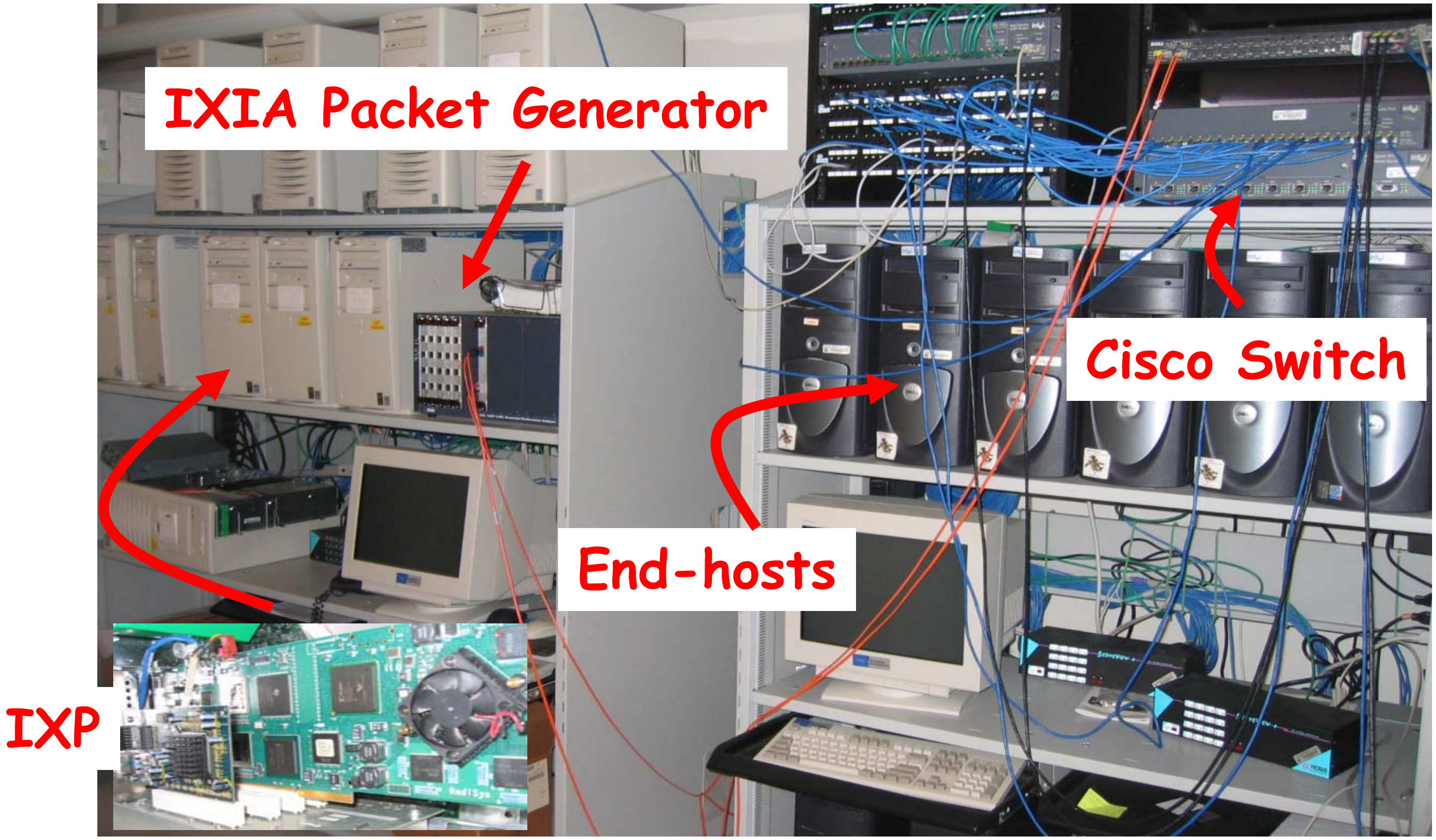
Dynamic State Management

- Middlebox introduction
  - How to capture state for ongoing connections ?
  - Naïve solution: terminate all ongoing connections and let clients start anew (not transparent!!)
  - Our solution
    - Add grace period to transparently bootstrap state for ongoing connection
  - During bootstrap
    - SYN cookies for new connection request
    - Data packets (good or bad) are forwarded to the server
    - State established for data packets for which ACK is seen

Dynamic State Management

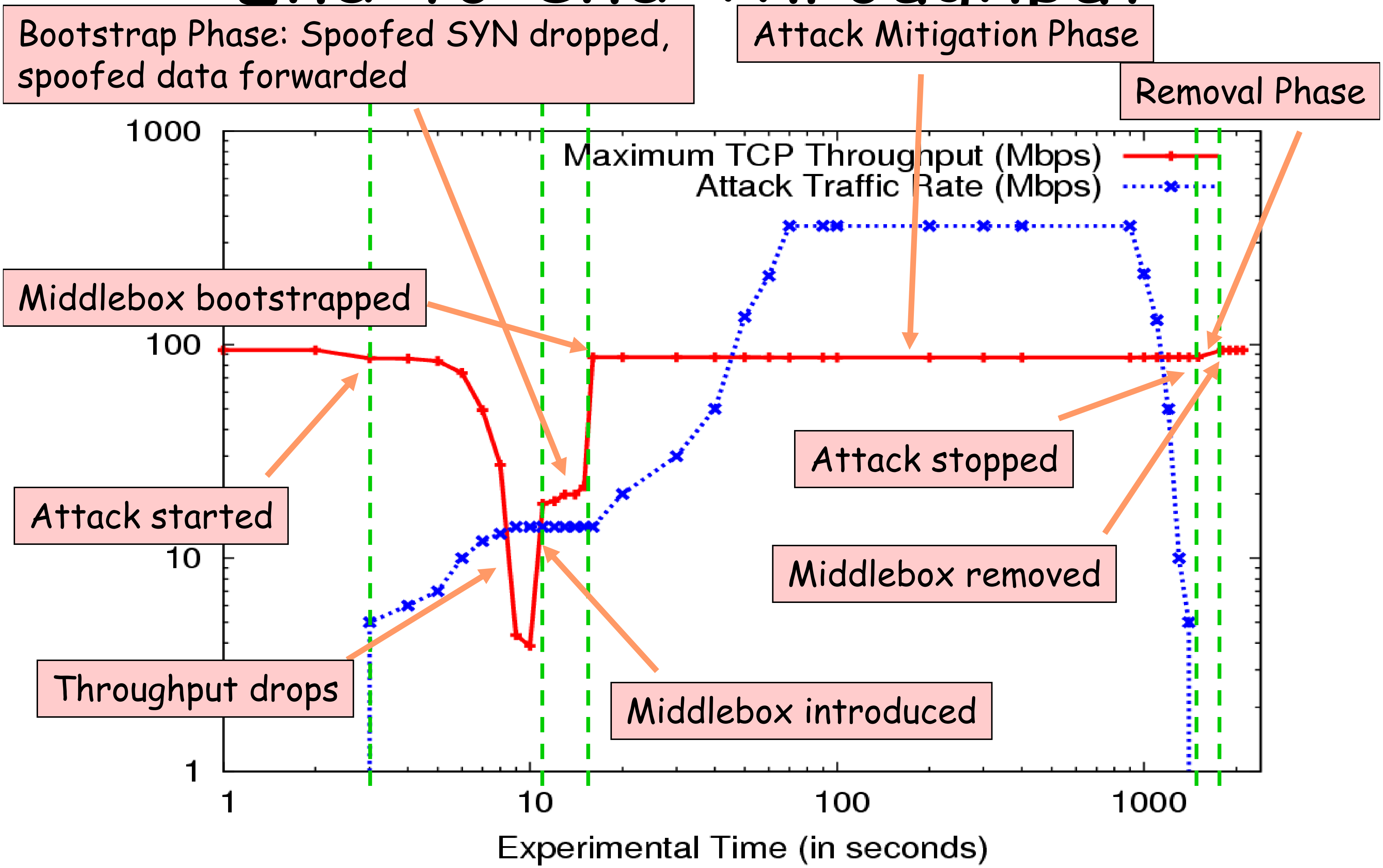
- Middlebox removal
  - What about active connections established via middlebox ?
  - Naïve solution: terminate all and remove middlebox from the data path (not transparent!!)
  - Our solution
    - Add grace period during which the connections established via middlebox undergo sequence and acknowledgement numbers translation
    - New connection requests are forwarded to the server (no SYN cookies)
    - No state established for new connections during the removal phase

Experimental Setup



- XORP for Traffic Interception
- Intel IXP Network Processor for attack mitigation policies
- IXIA for attack workload, iperf/httperf for legitimate traffic

End-to-end Throughput



Conclusion

- dFence DoS mitigation system
  - Transparent solution
  - In-network defense
  - Shared on-demand infrastructure
  - Stateful mitigation
- Can be viewed as providing group insurance service
- General platform to deploy other network security services such as malware filtering

Thank You !

Backup Slides

Flow Pinning

- Why Pinning ?
  - Ensure both directions of flow go through the same middlebox
  - Ensure that the same middlebox handles the flow even when there are route changes / failures

- Pin the flow to a home middlebox
  - Home middlebox = $hash_1$ (src IP, src port) EXOR $hash_2$ (dest IP, dest port)
  - Symmetric

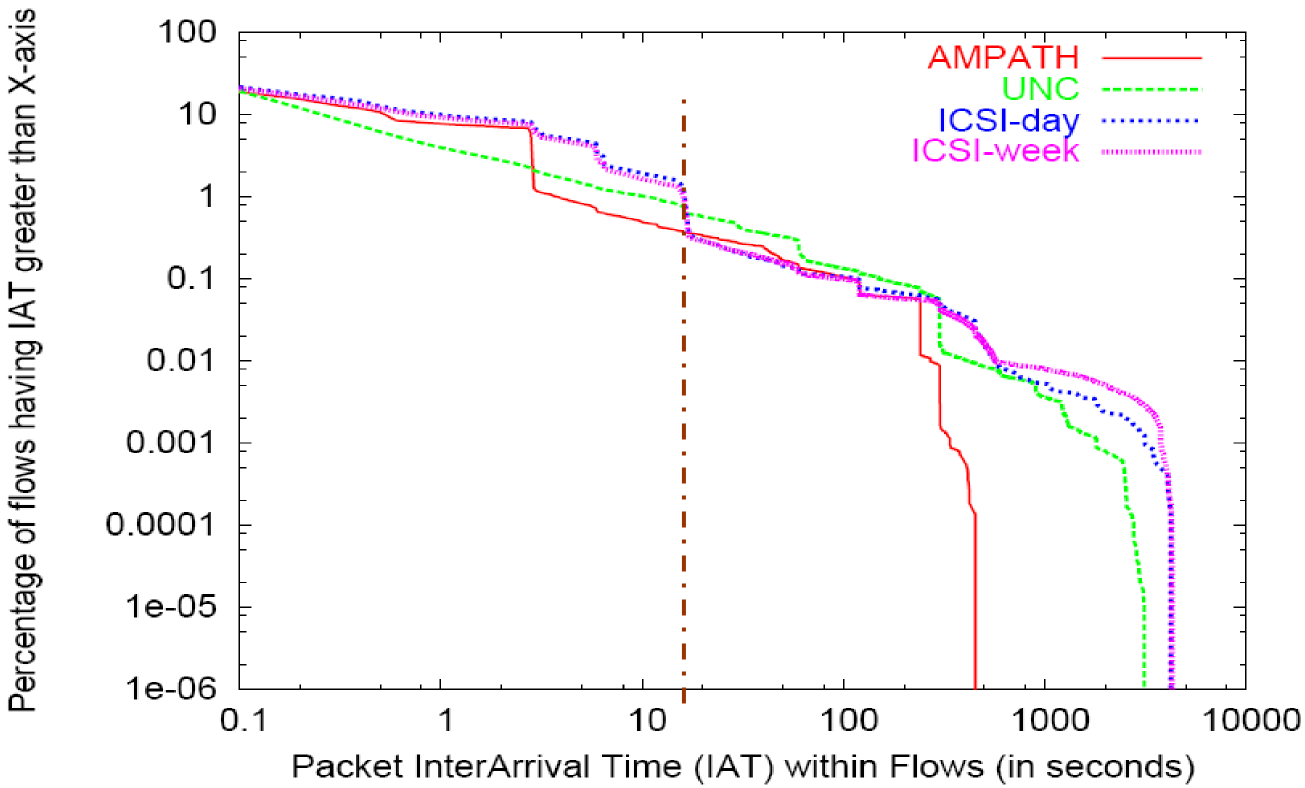
Bootstrap Interval T_b

Too high

- Severe damage during bootstrap phase

Too low

- Ongoing connections may get terminated



Trace analysis shows that majority of connections has packet IATs of the order a few seconds

XORP BGP Policy

```
policy-statement next-hop-selection {
    term 1 {
            to { network4: 10.0.0.0/24 }
            then { localpref: 300 }
    }
}
…
protocols {
    bgp {
            …
            import "next-hop-selection"
            export "next-hop-selection"
    }
}
```

Middlebox Attacks & Defenses

- Exhausting the connection state
  - Defense: Limit number of connections from any single host
    - Middlebox only maintains state for un-spoofed well-behaved sources
- Adaptive traffic variation attack
  - ON/OFF attack pattern
  - Defense: Avoid rapid introduction & removal of middleboxes
    - Randomize the removal phase time interval
- Werewolf attack
  - Behave legitimate at first, get established in middlebox state and then bombard with attack traffic
  - Defense: Periodic measurement of traffic sending rates & source prefix white-listing

End-to-end latency