State Synchronization for Fast Failover of Stateful Firewall VNF

Nicholas Gray, Claas Lorenz, Alexander Müssig, Steffen Gebert,
Thomas Zinner, Phuoc Tran-Gia

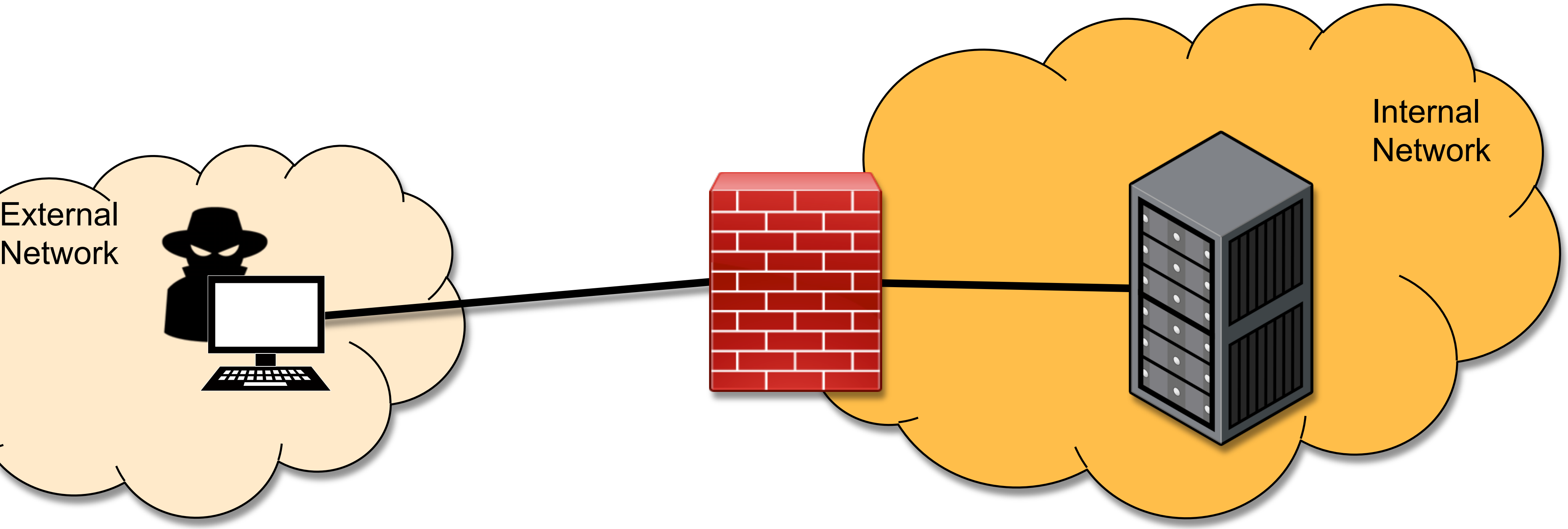
comnet.informatik.uni-wuerzburg.de

arDiNe
ine-project.org

GEFÖRDERT VOM
Bundesministe
für Bildung
und Forschung

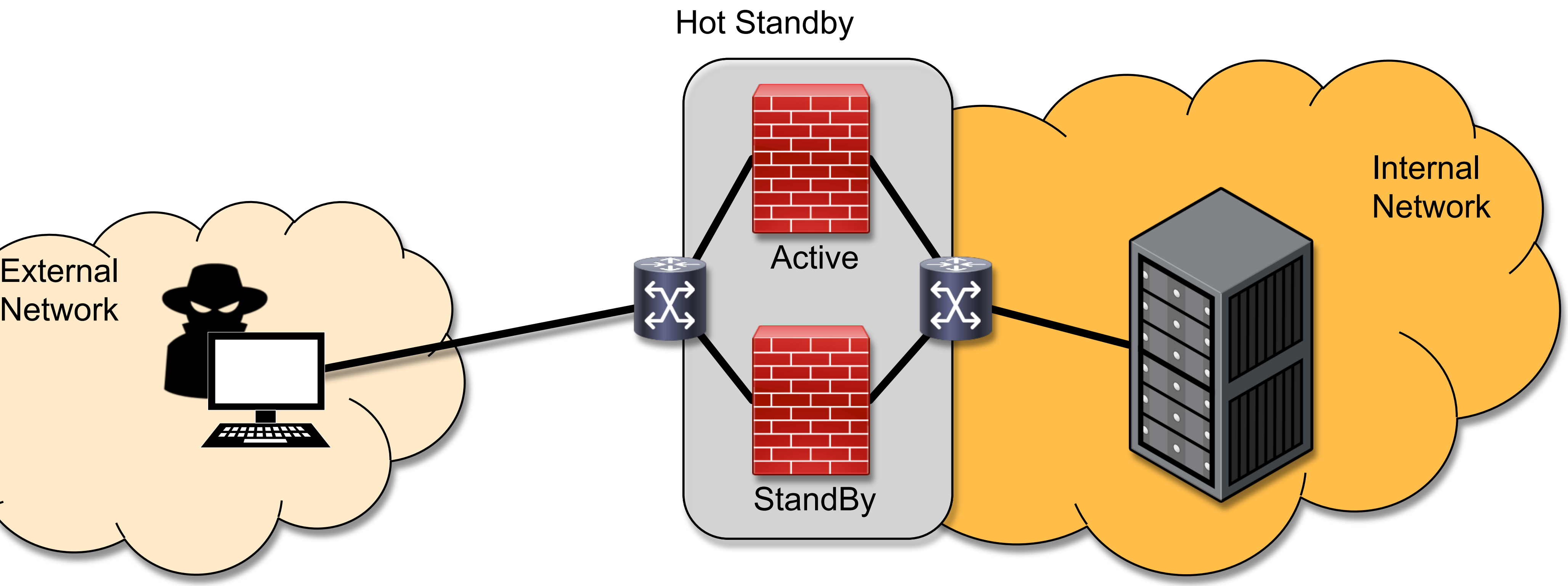
Motivation

- **Traditional deployment**: Single hardware device separates networks



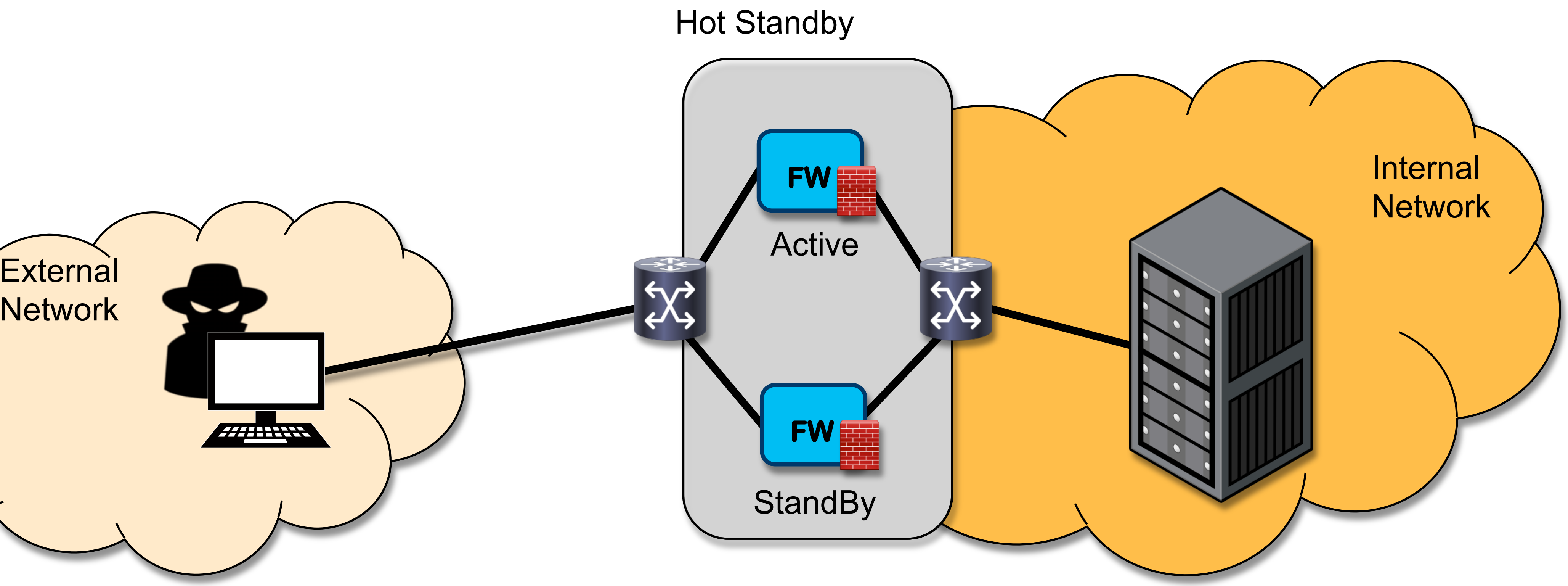
Motivation

- **Traditional deployment**: Single hardware device separates networks
- **Resilience**: Secondary instance as hot standby



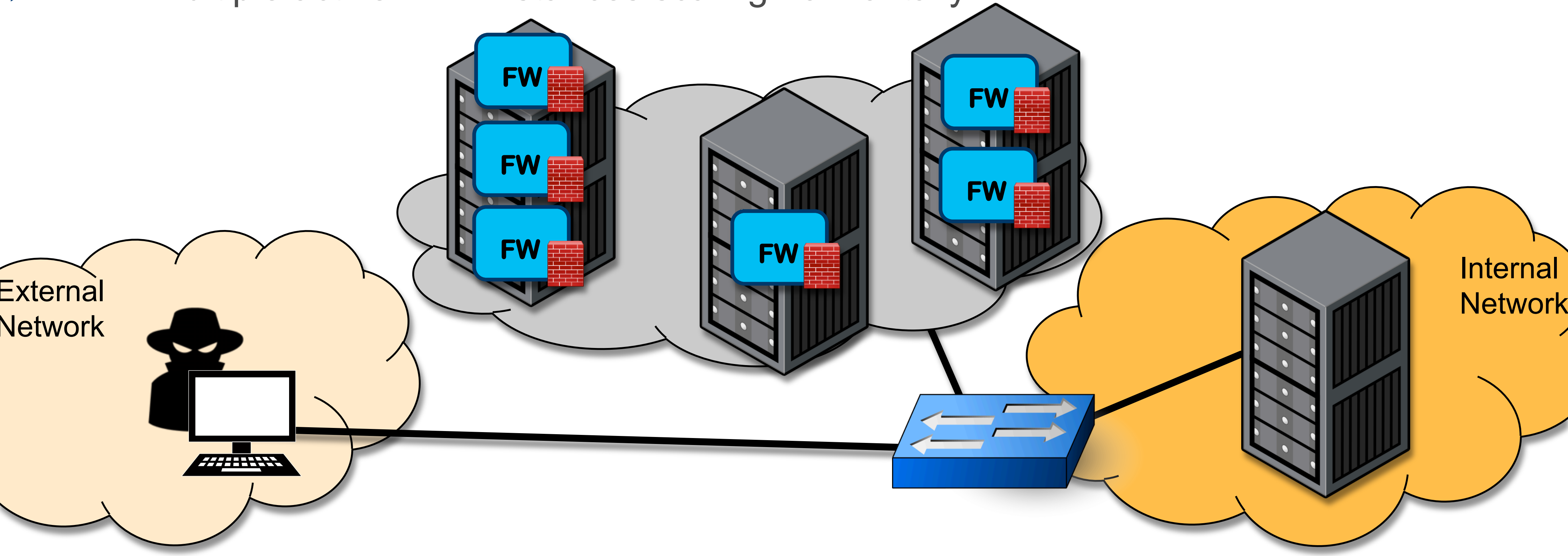
Motivation

- **Traditional deployment**: Single hardware device separates networks
- **Resilience**: Secondary instance as hot standby
- **Softwarization:** Firewall software running in virtual machine



Motivation

- **Traditional deployment**: Single hardware device separates networks
- **Resilience**: Secondary instance as hot standby
- **Softwarization:** Firewall software running in virtual machine
- **NFV**: Multiple active VNF instances scaling horizontally



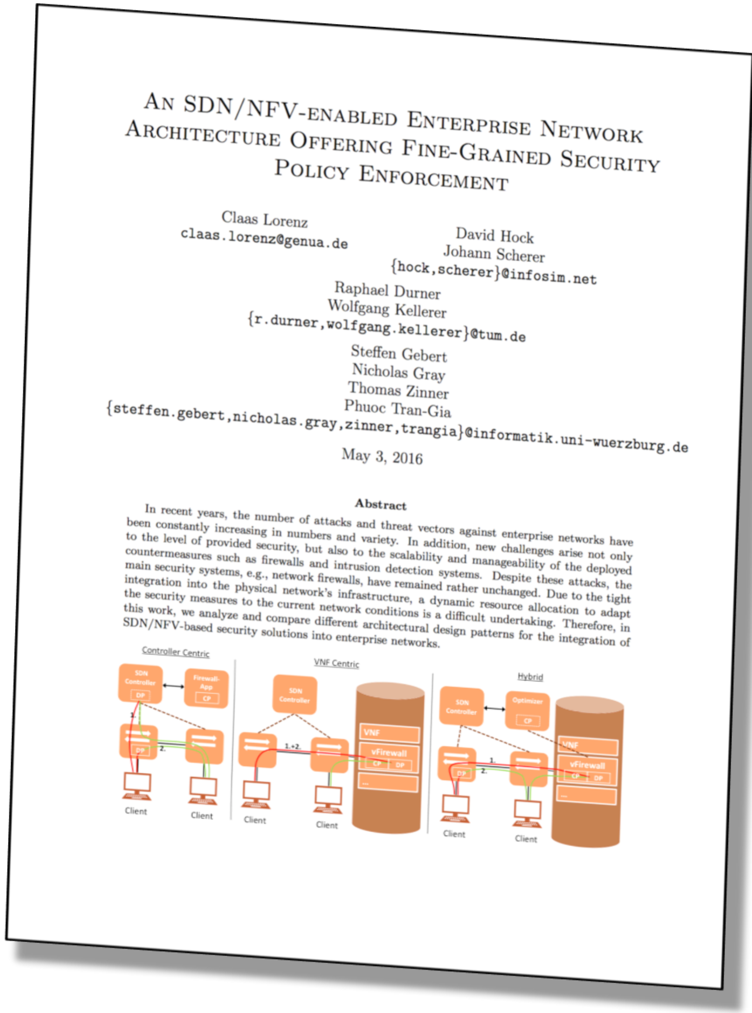
Motivation

- First approaches described by the research community

- Demands:
  - Horizontal scalability: Scale in and scale out
  - Resilience: Failover mechanisms

- Open questions:
  - Detailed design specifications
  - Suitable synchronization mechanisms
  - Performance evaluations

→ No existing firewall VNF meets these requirements



An SDN/NFV-enabled Enterprise Network Architecture Offering Fine-Grained Security Policy Enforcement

Claas Lorenz
claas.lorenz@genua.de

David Hock
Johann Scherer
{hock,scherer}@infosim.net

Raphael Durner
Wolfgang Kellerer
{r.durner,wolfgang.kellerer}@tum.de

Steffen Gebert
Nicholas Gray
Thomas Zinner
Phuoc Tran-Gia
{steffen.gebert,nicholas.gray,zinner,trangia}@informatik.uni-wuerzburg.de

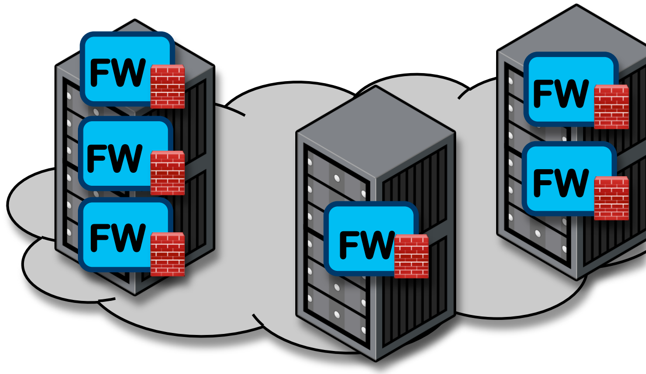
May 3, 2016

Abstract

In recent years, the number of attacks and threat vectors against enterprise networks have been constantly increasing in numbers and variety. In addition, new challenges arise not only to the level of provided security, but also to the scalability and manageability of the deployed countermeasures such as firewalls and intrusion detection systems. Despite these attacks, the main security systems, e.g., network firewalls, have remained rather unchanged. Due to the tight integration into the physical network's infrastructure, a dynamic resource allocation to adapt the security measures to the current network conditions is a difficult undertaking. Therefore, in this work, we analyze and compare different architectural design patterns for the integration of SDN/NFV-based security solutions into enterprise networks.

Agenda

- Motivation
- Background
- Concept of a cloud-based firewall
  - Failover
  - Scale in/out
  - Synchronization
- Implementation
  - VNF firewall module
  - Test bed configuration
- Evaluation



Firewalling

Packet Filter
OSI-Layer 3 (L4)

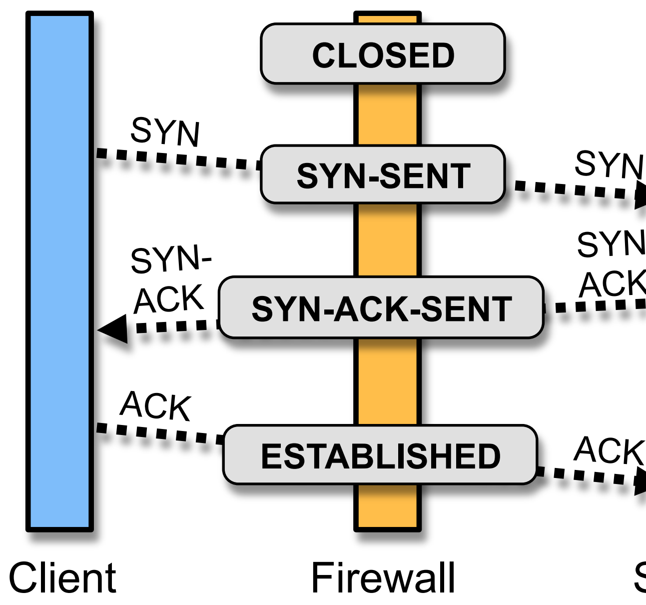
: → 132.187.15.12:80 ✓
: → *:* (implicite) ✗

Stateful Firewall
OSI-Layer 4

:, TCP-SYN
→ 132.187.15.12:80 ✓
:, TCP-ACK
132.187.15.12:80 ✗

Application Layer Firewall
OSI-Layer 7

:, HTTP GET /index.html ✓
:, HTTP GET /evil.html ✗



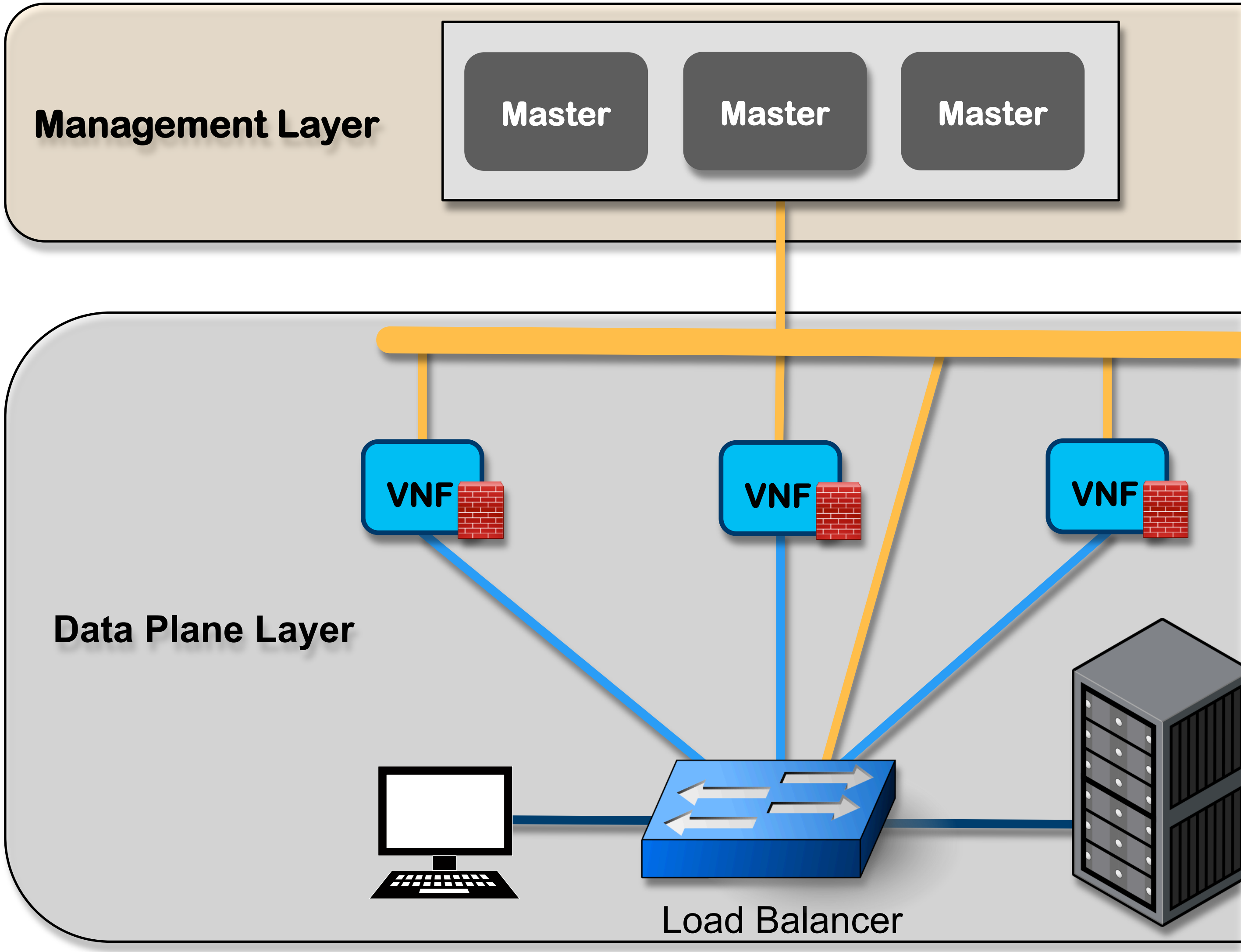
Concept

Management Layer

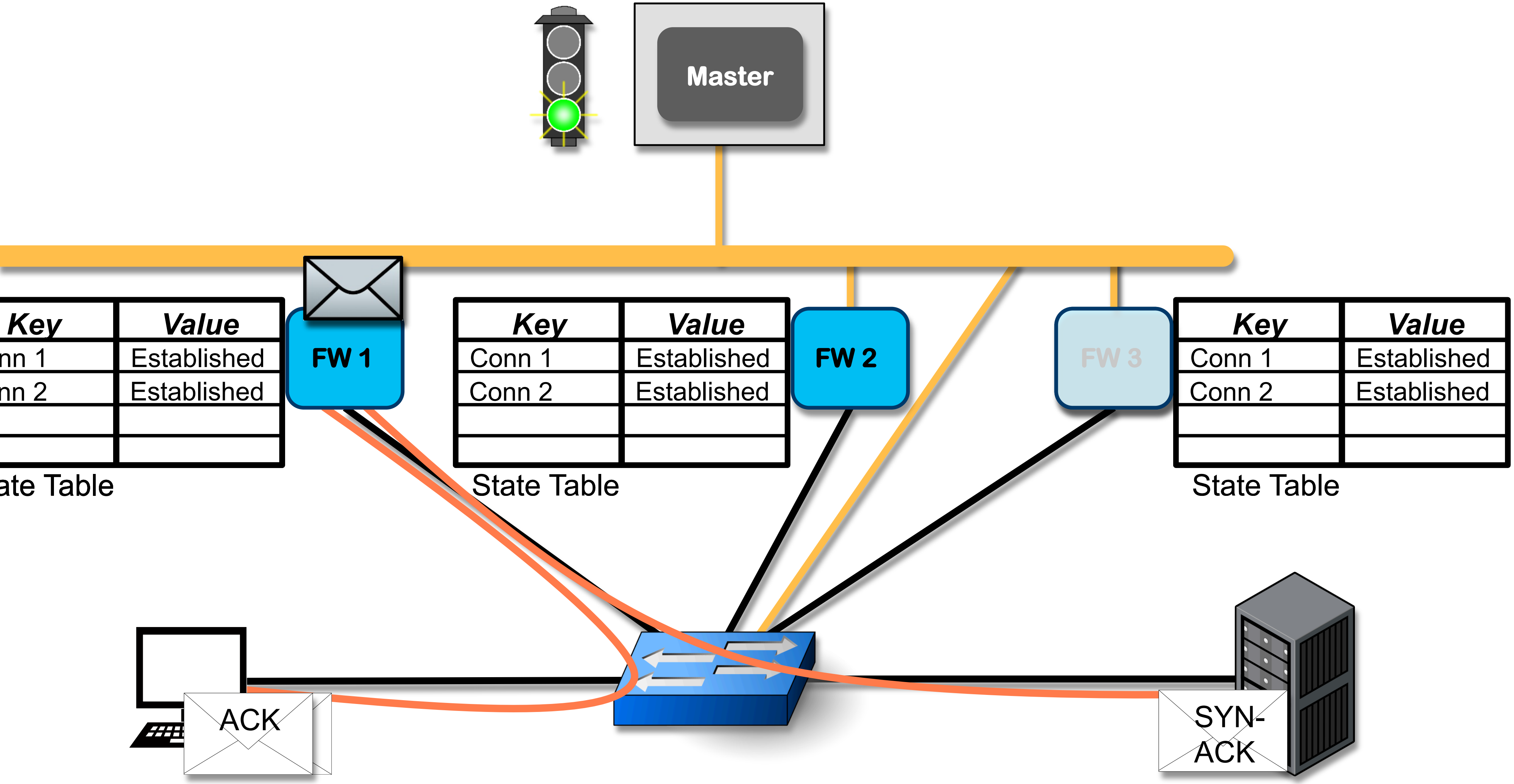
- Coordination
- Cluster health

Data Plane Layer

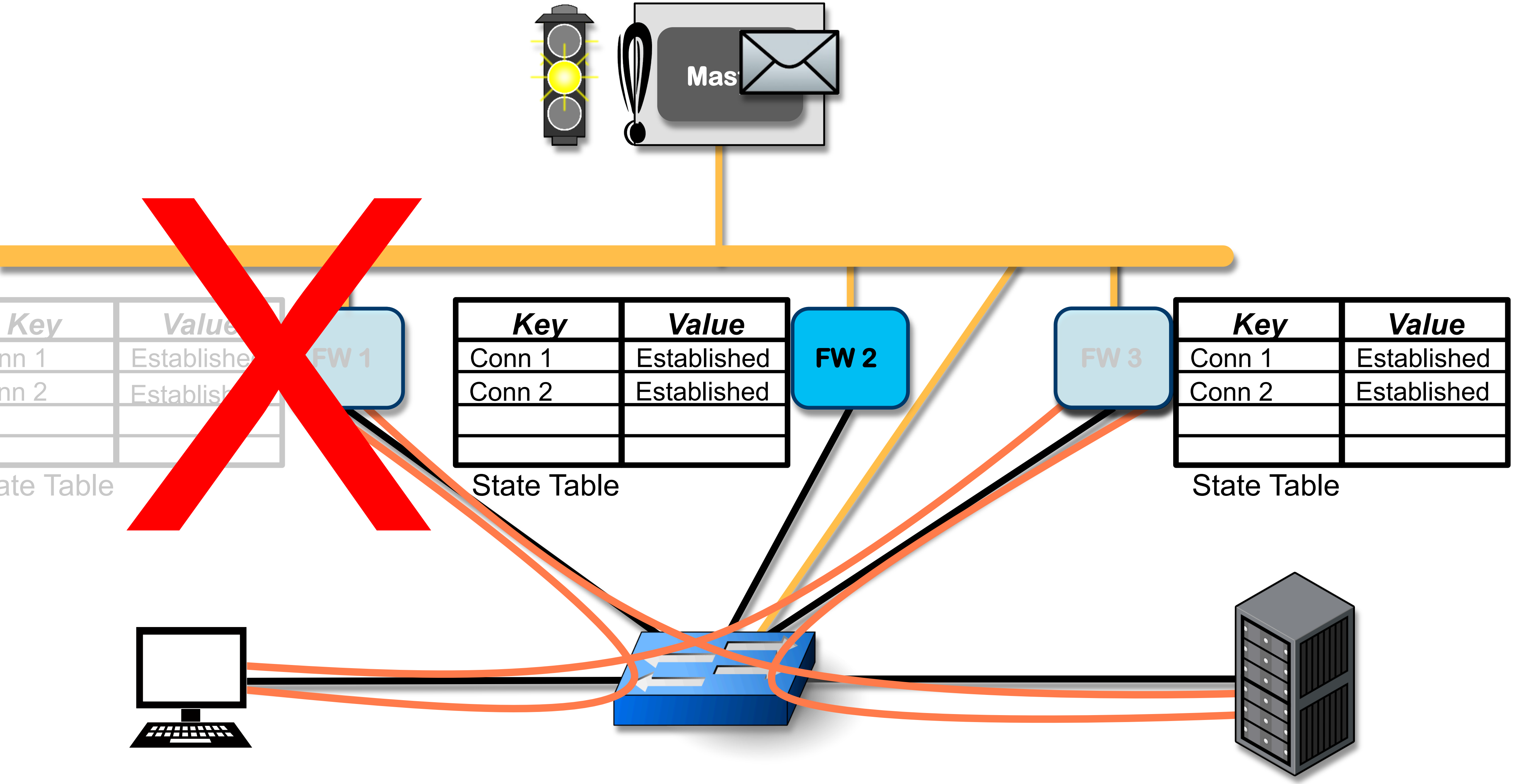
- Load balancing
- Traffic redirection
- Packet inspection



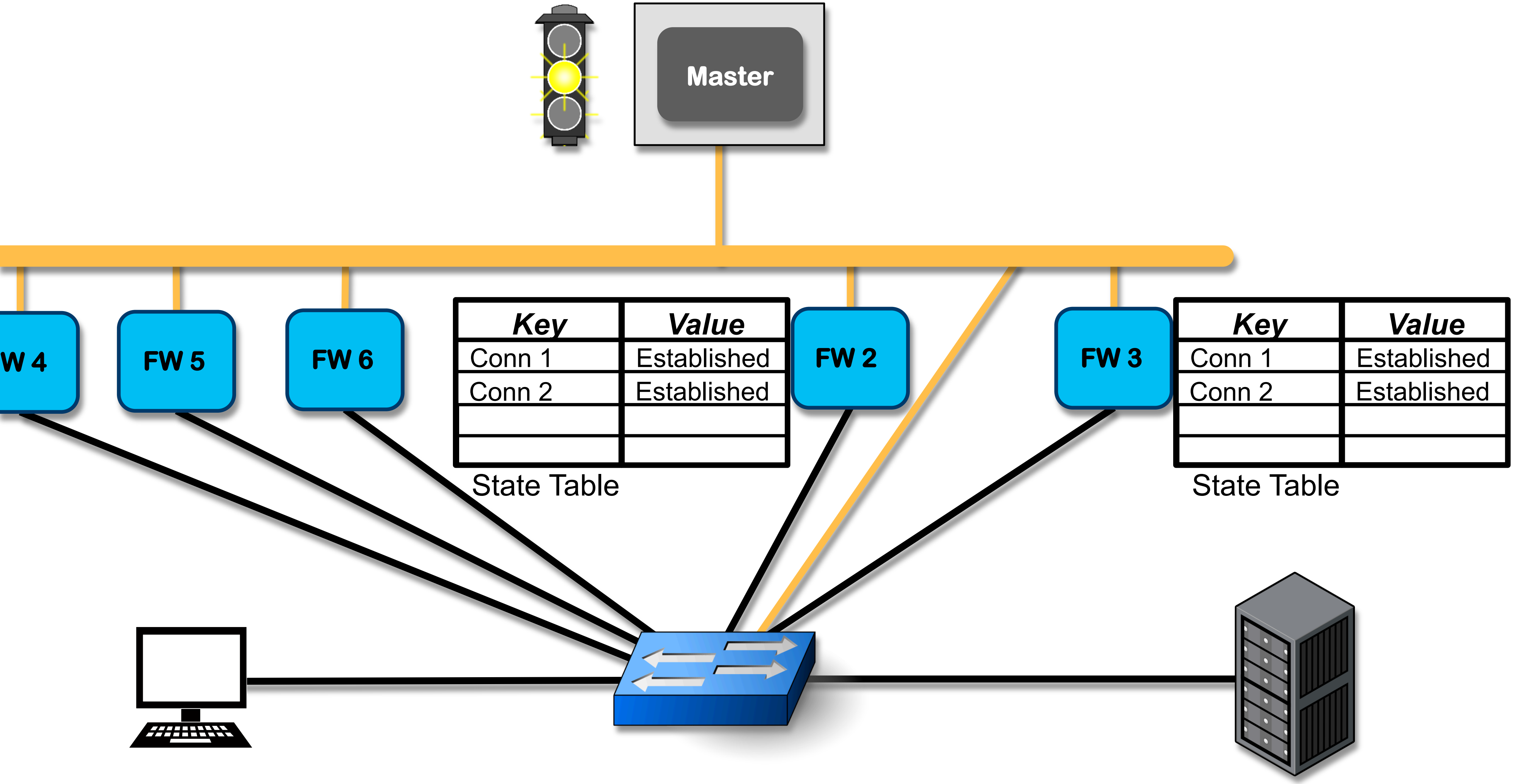
Example: Traffic Flow



Example: Fail over



Example: Scale Out



Implementation

- **Erlang** is a functional programming language by Ericsson:
  - Provides high availability
  - Specialized for multithreading

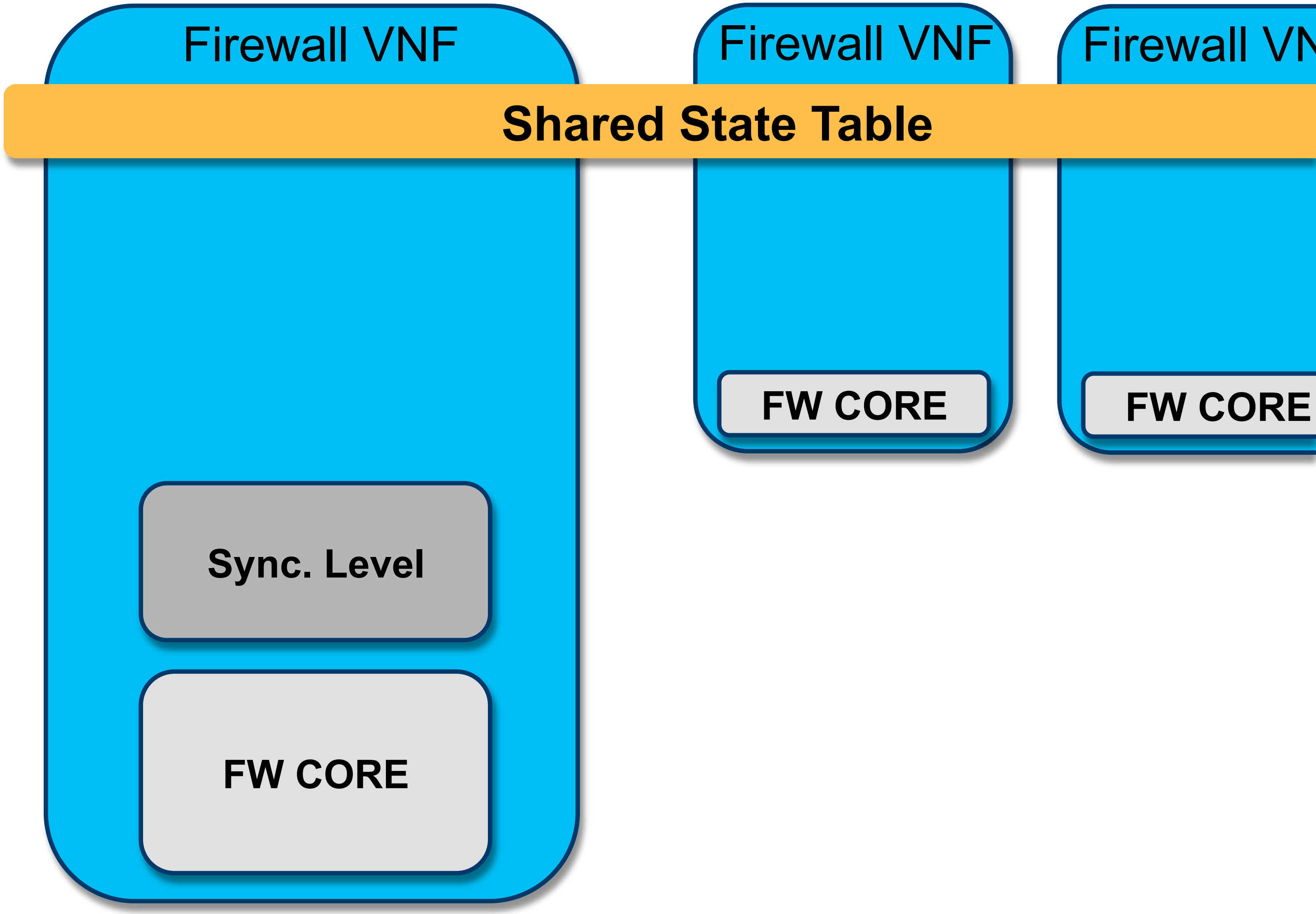
- Prototypical implementation:
  - Stateful firewall: Every state is logged and packets are inspected
  - Cluster size expands dynamically

- Parameter configuration:
  - Synchronization level
  - Data access
  - Synchronization strategy

Firewall Modules

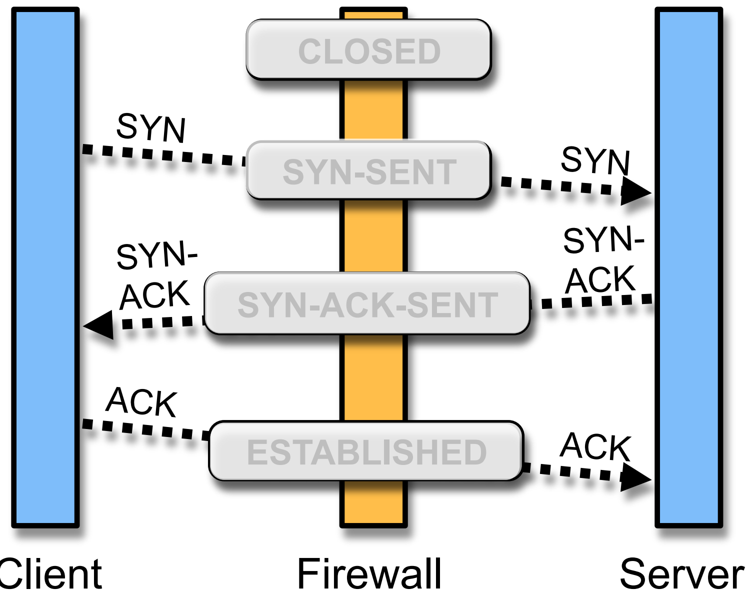
CP states to
ynchronize

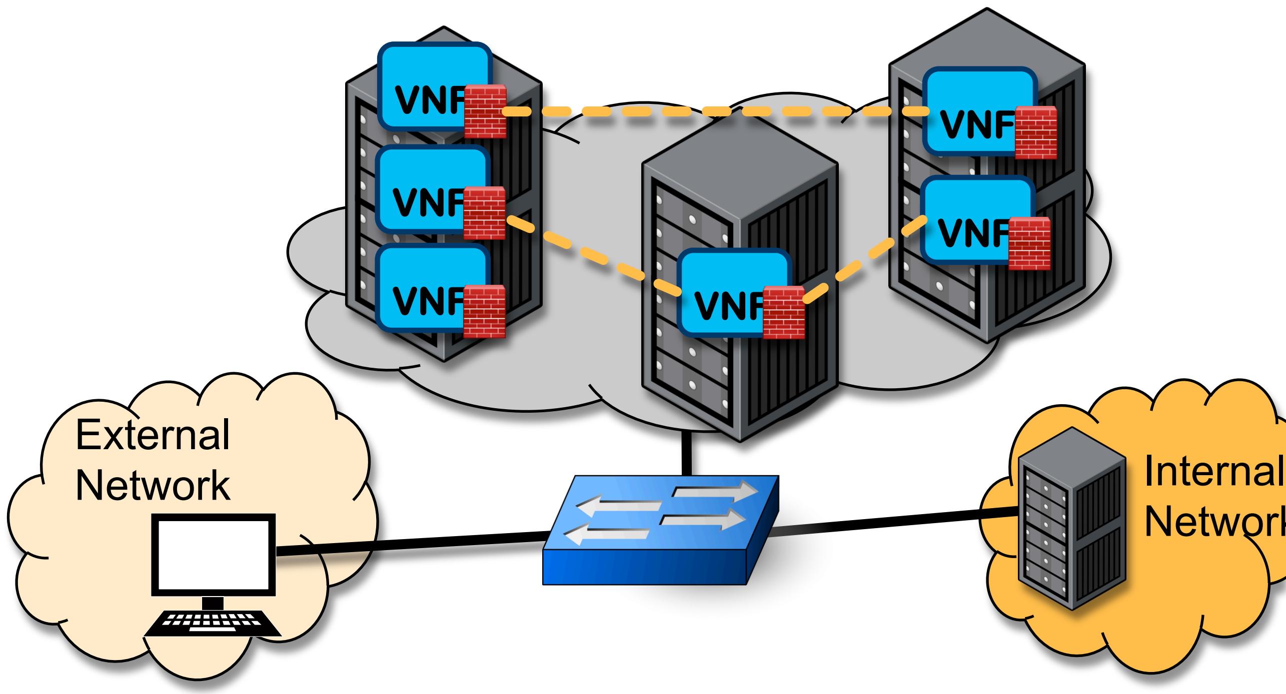
tateful packet filtering



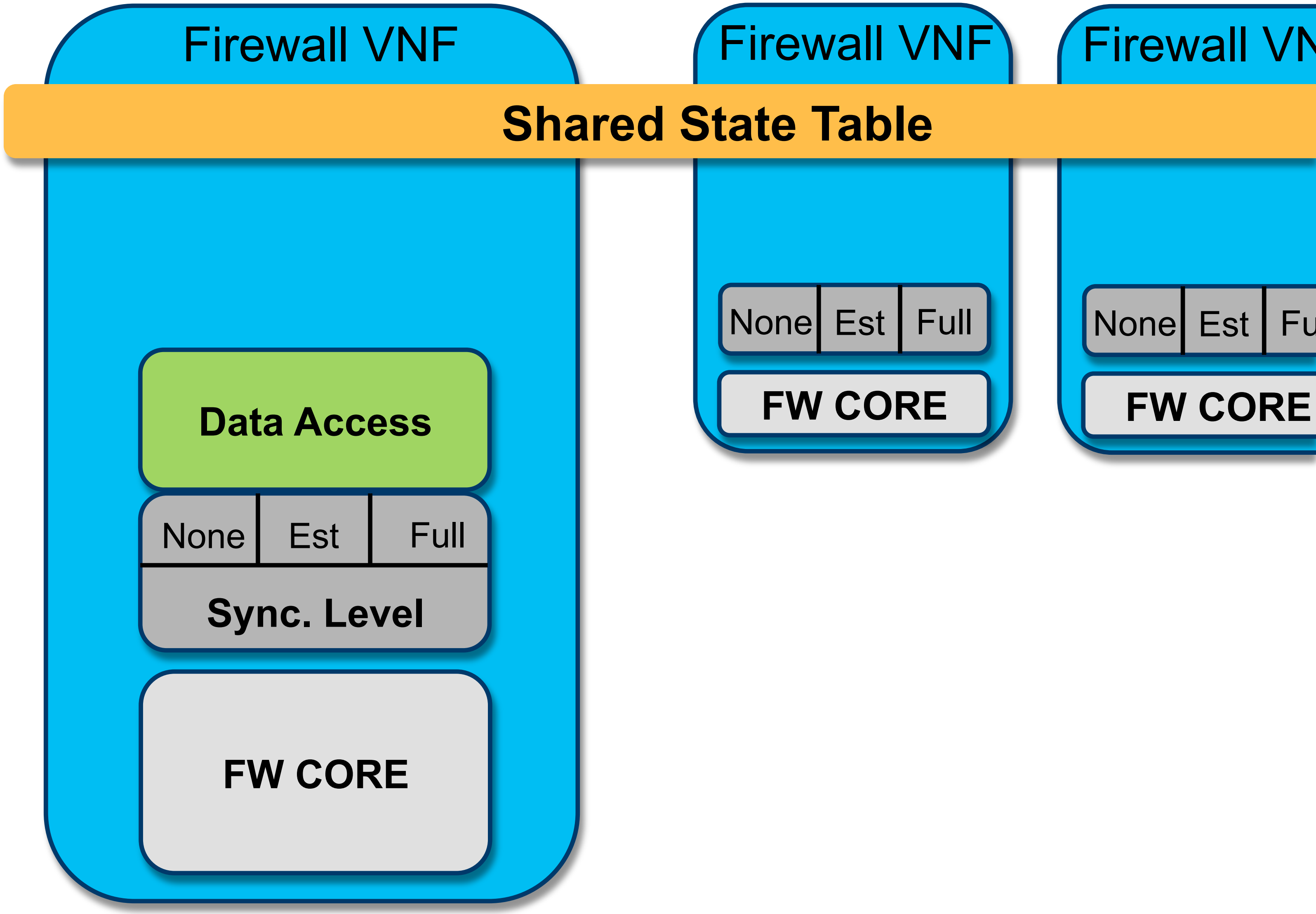
Synchronization Levels

- Propagating levels for TCP states:
  - *NONE*: No changes are propagated
  - *ESTABLISHED*: Only essential state changes *Established* and *Closed* are propagated
  - *FULL*: All changes are propagated to the network





Firewall Modules



/rite mode to shared state

CP states to
ynchronize

tateful packet filtering

Database Write

Clean transaction context

Dirty transaction context

+ Maintains data consistency:
  Only one process can update a record
- Low performance:
  Requires locking and unlocking the database

+ High performance:
  Directly update the record
- No data consistency guarantee:
  Ignore side effects of concurrent acces

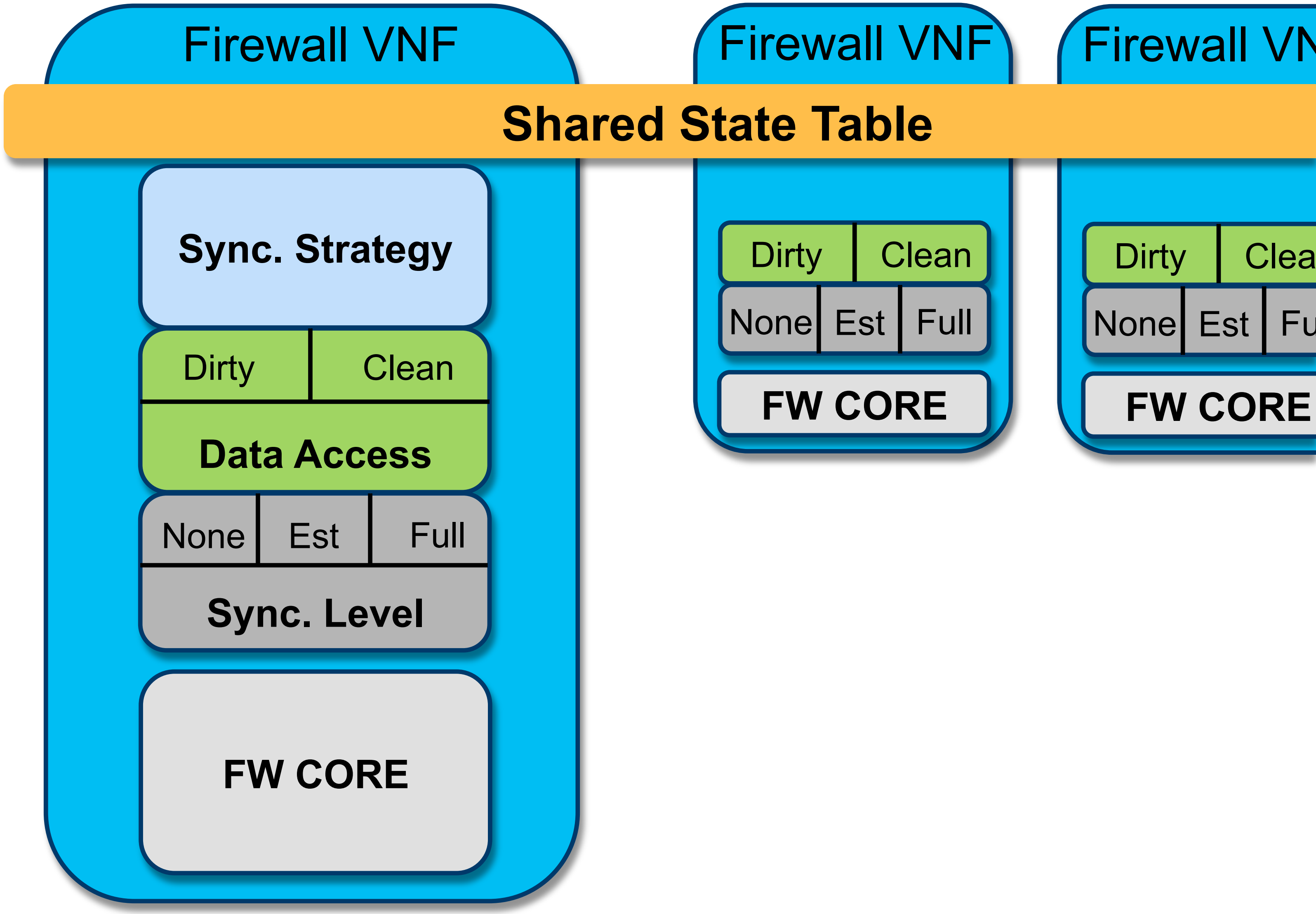
Firewall Modules

Vrite confirmation
mong the cluster

Vrite mode
o shared state

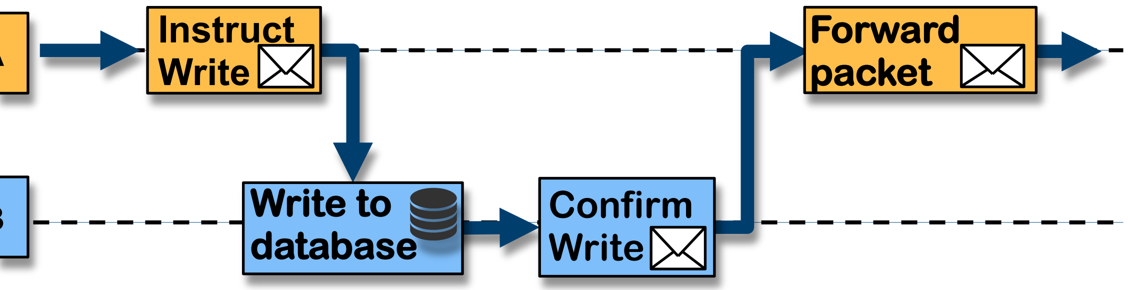
CP states to
ynchronize

tateful packet filtering



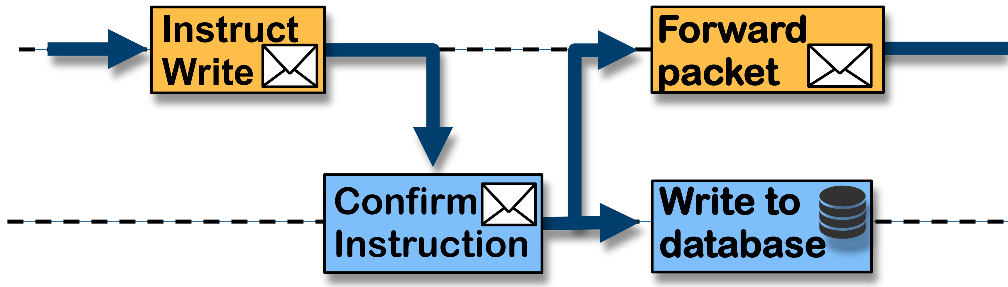
Confirmation Strategies

Synchronized confirmation strategy



+ Maintains data consistency:
  Throughout the entire cluster
- Low performance:
  Wait for all nodes to confirm write

Asynchronized confirmation strategy



+ High performance:
  Concurrent write and packet forwarding
- No data consistency:
  A successful write cannot be ensured

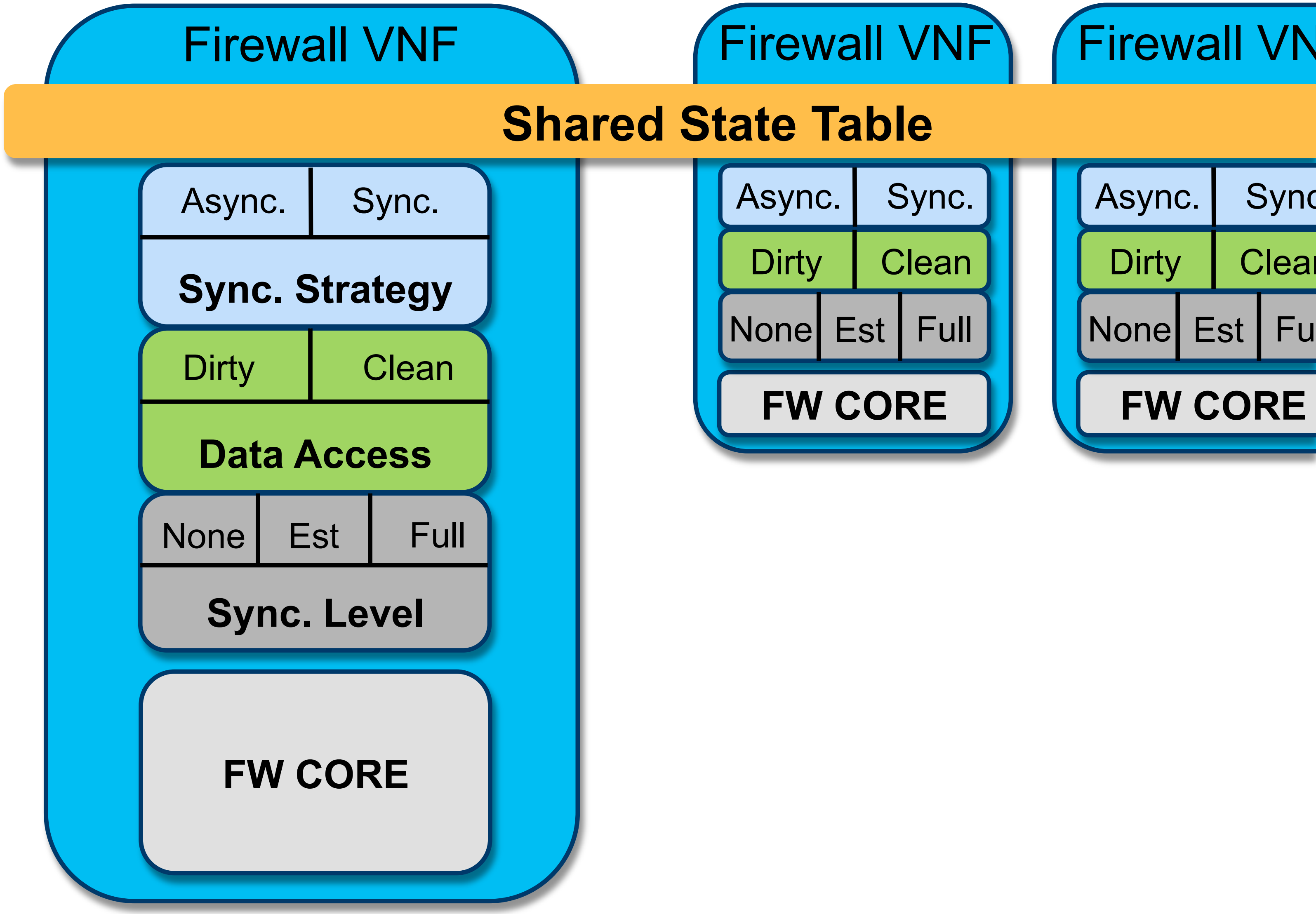
Firewall Modules

Vrite confirmation
mong the cluster

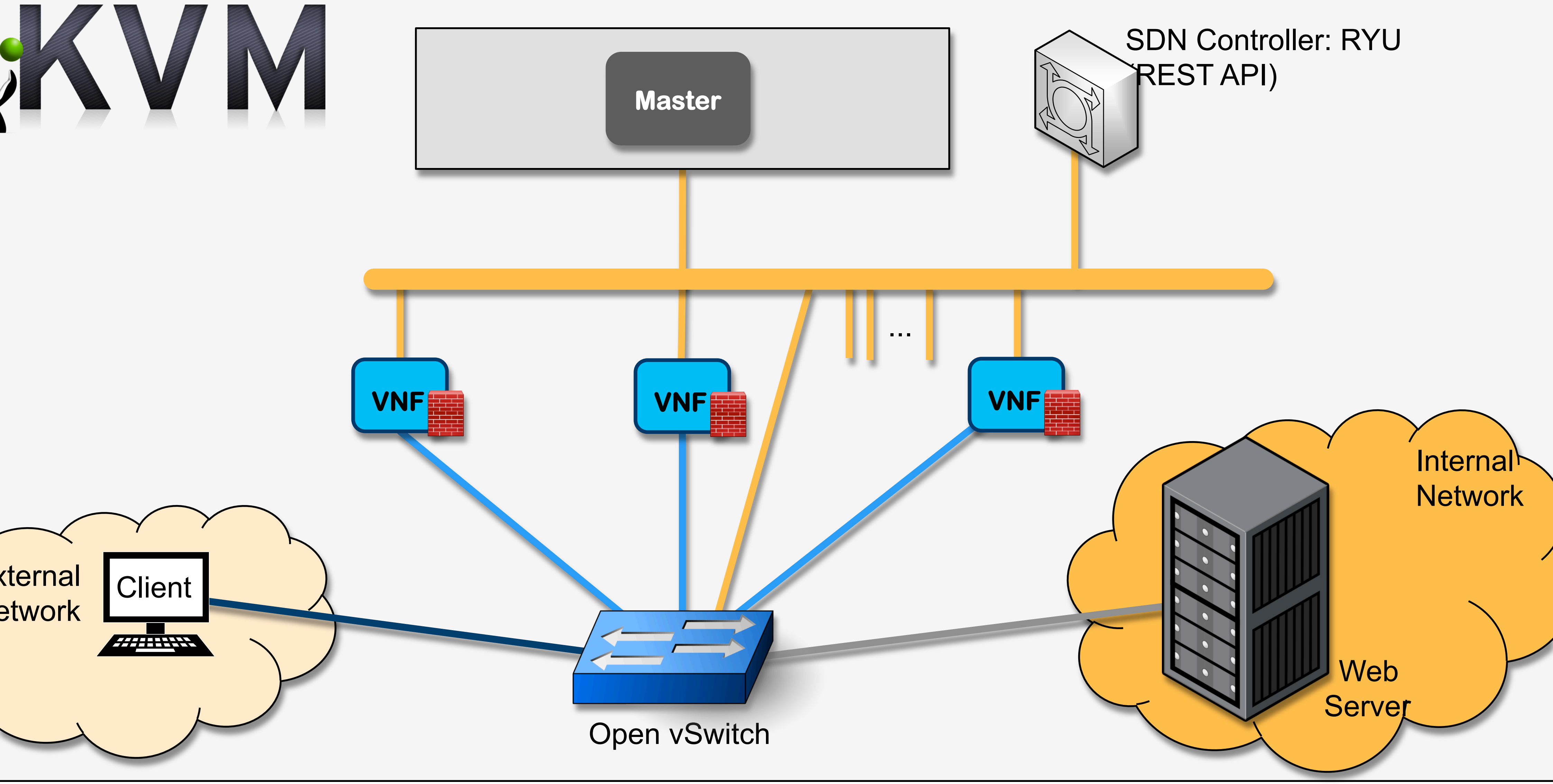
Vrite mode to shared state

CP states to
ynchronize

tateful packet filtering

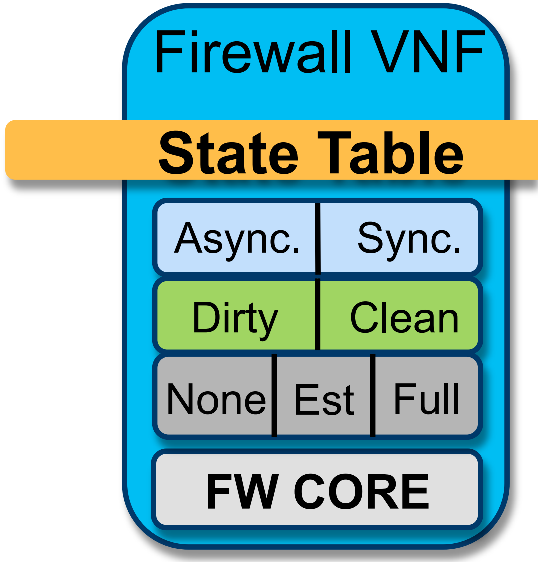


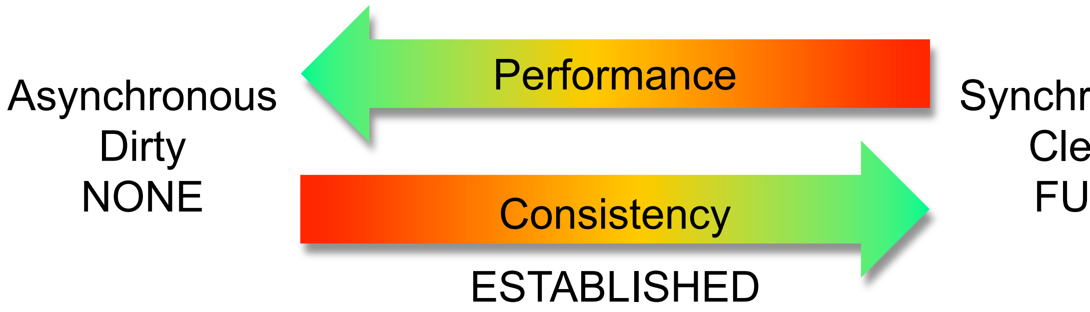
Test Bed Configuration



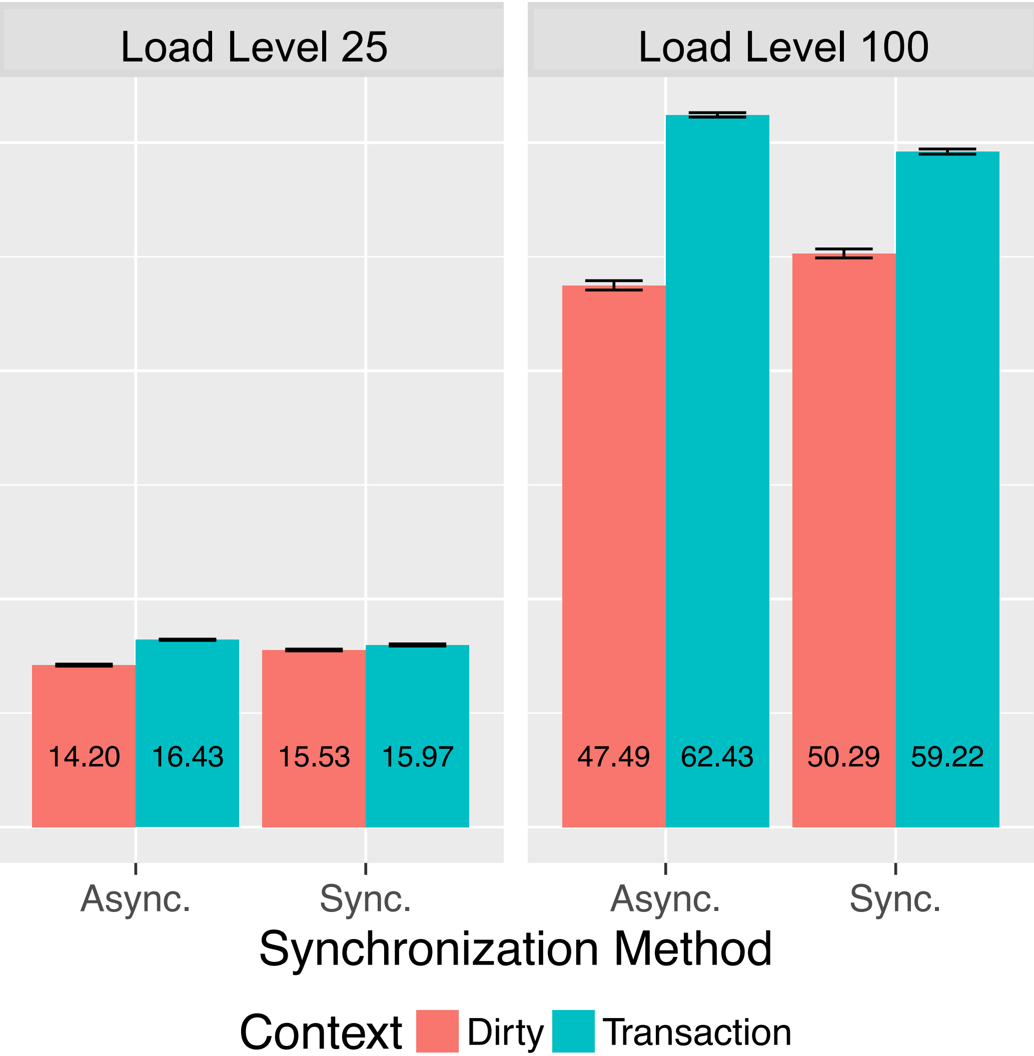
Methodology

- Test bed setup:
  - One Monitoring node
  - One active firewall node
  - One backup firewall node
- Scenario:
  - Downloading *index.html* (1 Byte) from web server
  - Different load levels of 25 and 100 concurrent connections
  - 10 runs with 10,000 downloads each
- Parameter configuration:
  - Synchronization level
  - Data access
  - Synchronization strategy
- Objective: TCP connection setup times



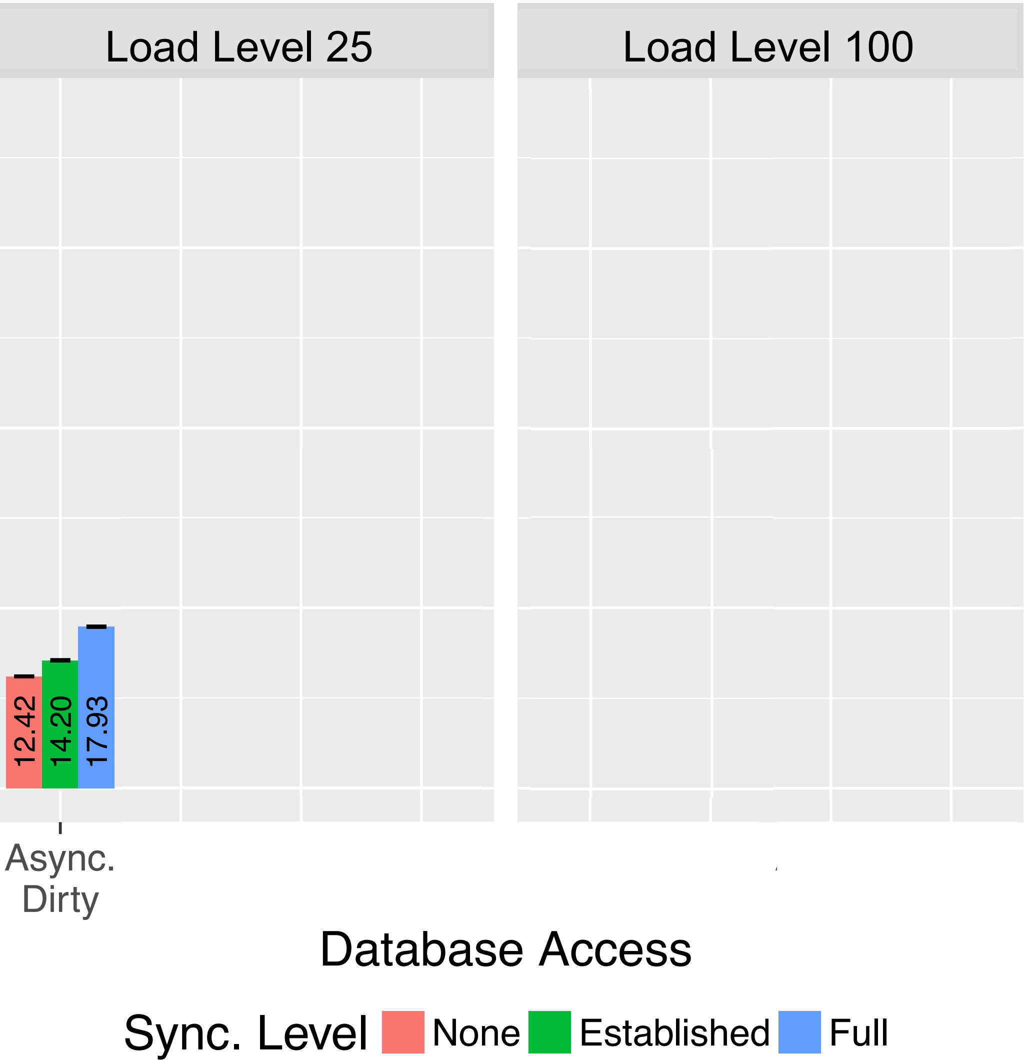


Database Access Strategies



- Load level 25:
  Minor difference up to 2ms
- Load level 100:
  Significant difference up to 15ms
- Increased impact for higher concurrency leve
- Dirty context significantly faster than transact
- Contrary to all expectations, synchronous transactions faster than asynchronous transa

Synchronization Level

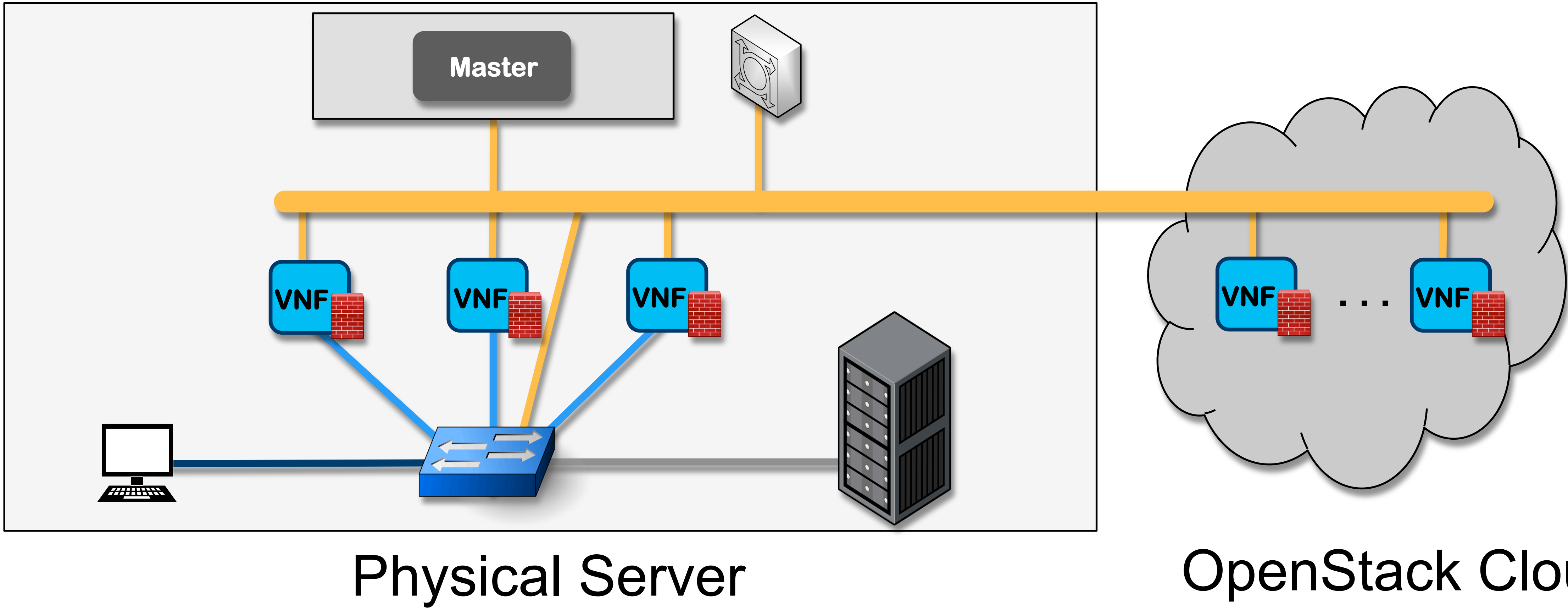


- Dirty context: More synchronization leads to higher connection setup times
- Transaction context: *ESTABLISHED* faster than *NONE*
  → More balanced database tables
- Synchronization levels show higher impact at higher concurrency levels
- Increased connection setup times for *FULL* synchronization

Cluster Sizes

nodes on physical KVM server

dditional nodes connected via *OpenStack*

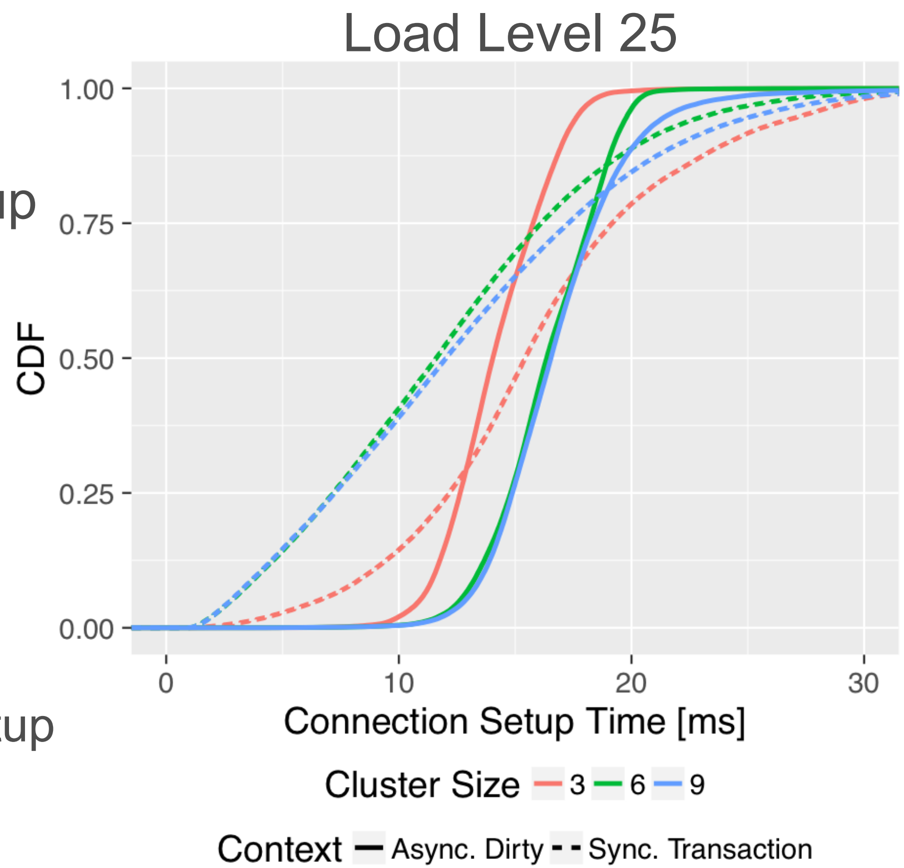


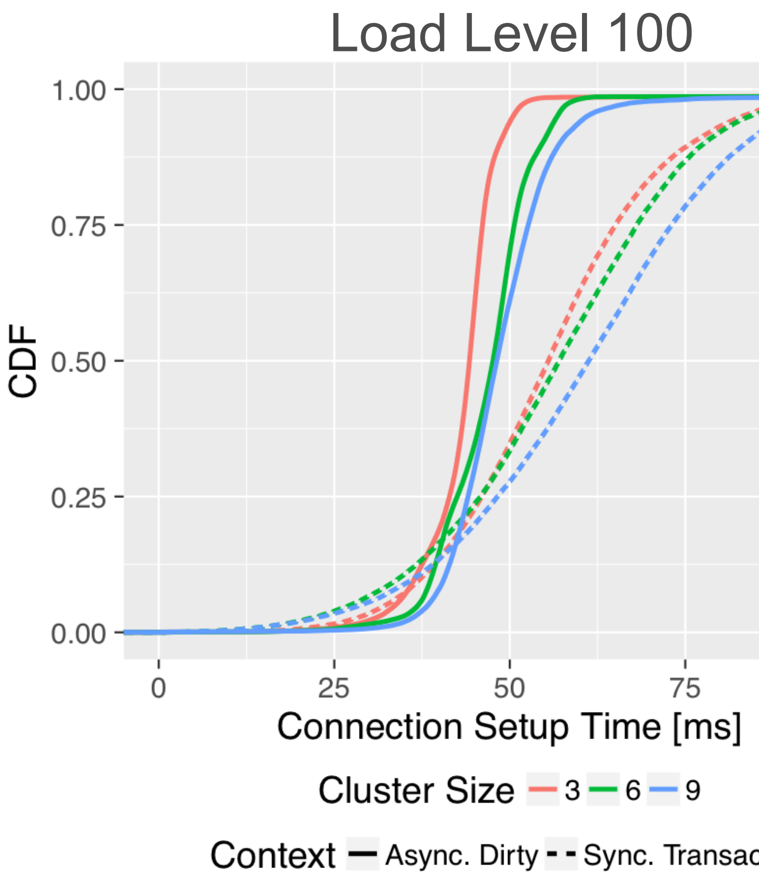
Cluster Sizes

nodes on physical KVM server
dditional nodes connected via *OpenStack*

irty context:
arger Cluster
→ Slower connection setup

ransaction context:
- Load level 25:
  Cluster size 3 with
  highest setup times
- Load level 100:
  Larger cluster
  → Slower connection setup





Conclusion and Outlook

- Concept of a stateful firewall VNF
  - Horizontal scalability
  - Failover
- Prototypical implementation of a stateful firewall VNF
  - Different database access strategies
  - Varying synchronization levels
- Test bed setup
  - Multiple VMs running firewall VNF
  - Connection to OpenStack cloud to increase cluster size
- Investigation of TCP connection setup times w.r.t. consistency and performance
  - Synchronizing all states leads to 19-26% slower connection setup times
  - 20% faster connection setup times when focusing on performance
  - Cluster sizes of 6 and 9 adds delay of 7% and 10% in comparison to a size of 3
- Future work: Alternative data stores