Towards Software-Defined Middlebox Networking

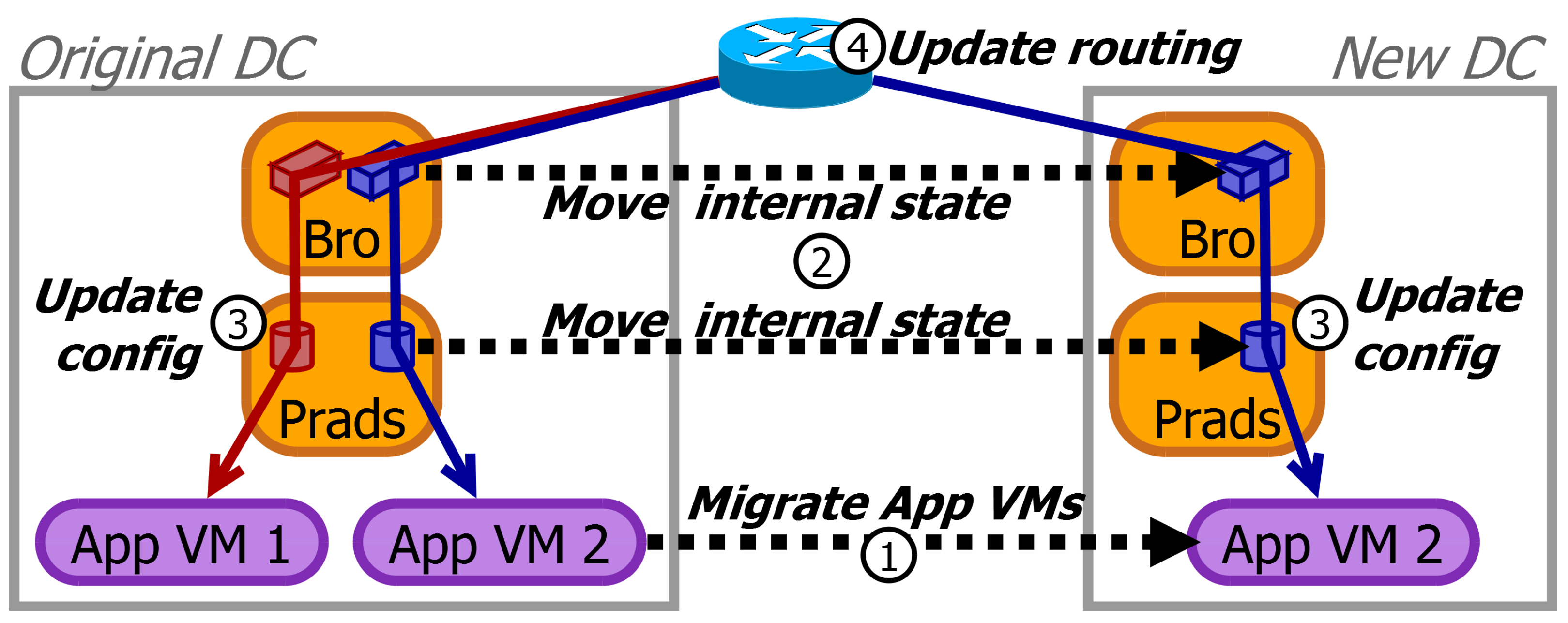
Aaron Gember, Robert Grandl, Junaid Khalid, and Shan-Hsiang Shen

Controlling Middleboxes

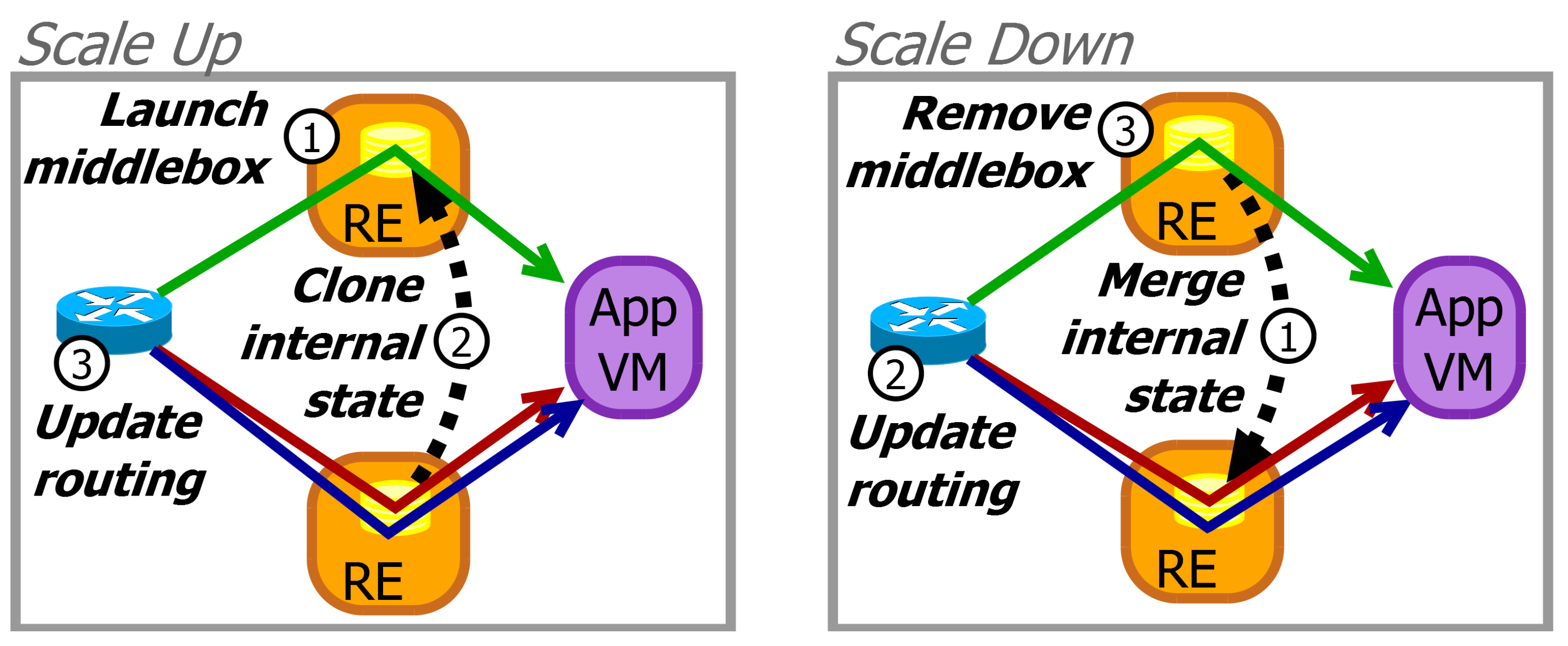
In recent years, software middleboxes have become an essential part of many enterprise data centers and cloud deployments to improve the security, availability and performance of the network. However, existing techniques to manage middleboxes—e.g., virtual machine snapshots, joint control of MB configuration and network routing [1], and application level libraries [2]—are clumsy and limited in their applicability. We propose a software-defined middlebox networking(SDMBN) framework that simplifies management and engenders rich, new applications.

Motivating Scenarios

Live migration between data centers



Middlebox scaling and load balancing

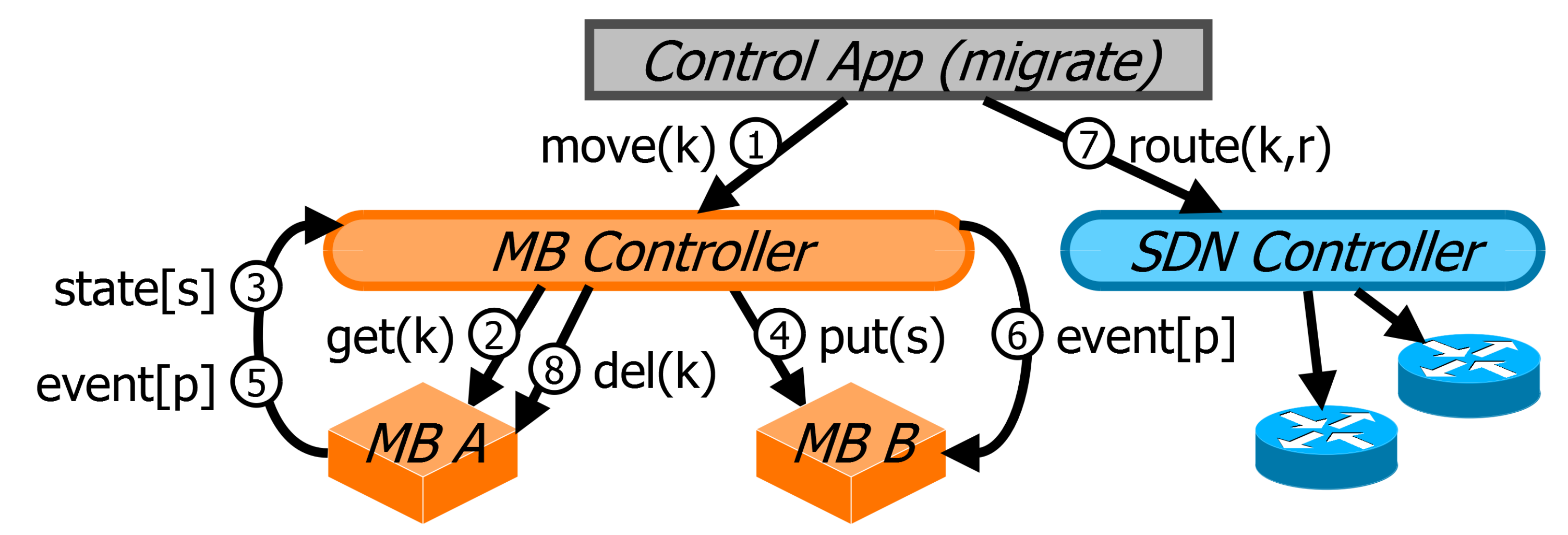


Middlebox State Taxonomy

Role	Definition	
Configuration	Defines and tunes middlebox behavior	*Shared only Middlebox reads*
Supporting	Guides middlebox decisions and actions based on past traffic	*Per-flow & shared Middlebox reads & writes*
Reporting	Quantify observations and decisions	*Per-flow & shared Middlebox writes*

Our taxonomy highlights commonalities that can be leveraged to design control interfaces

SDMBN Architecture



1) High-level operation to move state
2 & 3) Controller issues a get request and receives the state
4) Insert the moved state
5 & 6) Reprocessing events to ensure atomic state change
7) Update the route
8) Remove moved state

Northbound API

Application Interface

- Simplifies control applications by hiding complex details of get/put/delete, events, etc.
- Enables independent middlebox evolution

```
moveInternal(<Src>,<Dst>,<HdrFieldList>)
cloneSupport(<Src>,<Dst>)
mergeInternal(<Src>,<Dst>)
```

Implemented live migration and scaling control applications on top of northbound API

Modified Bro, PRADS, and SmartRE to support southbound API

Southbound API

State Interface

- Desire to conceal state structure and protect its integrity
- Need to move, clone, and merge state at fine granularity

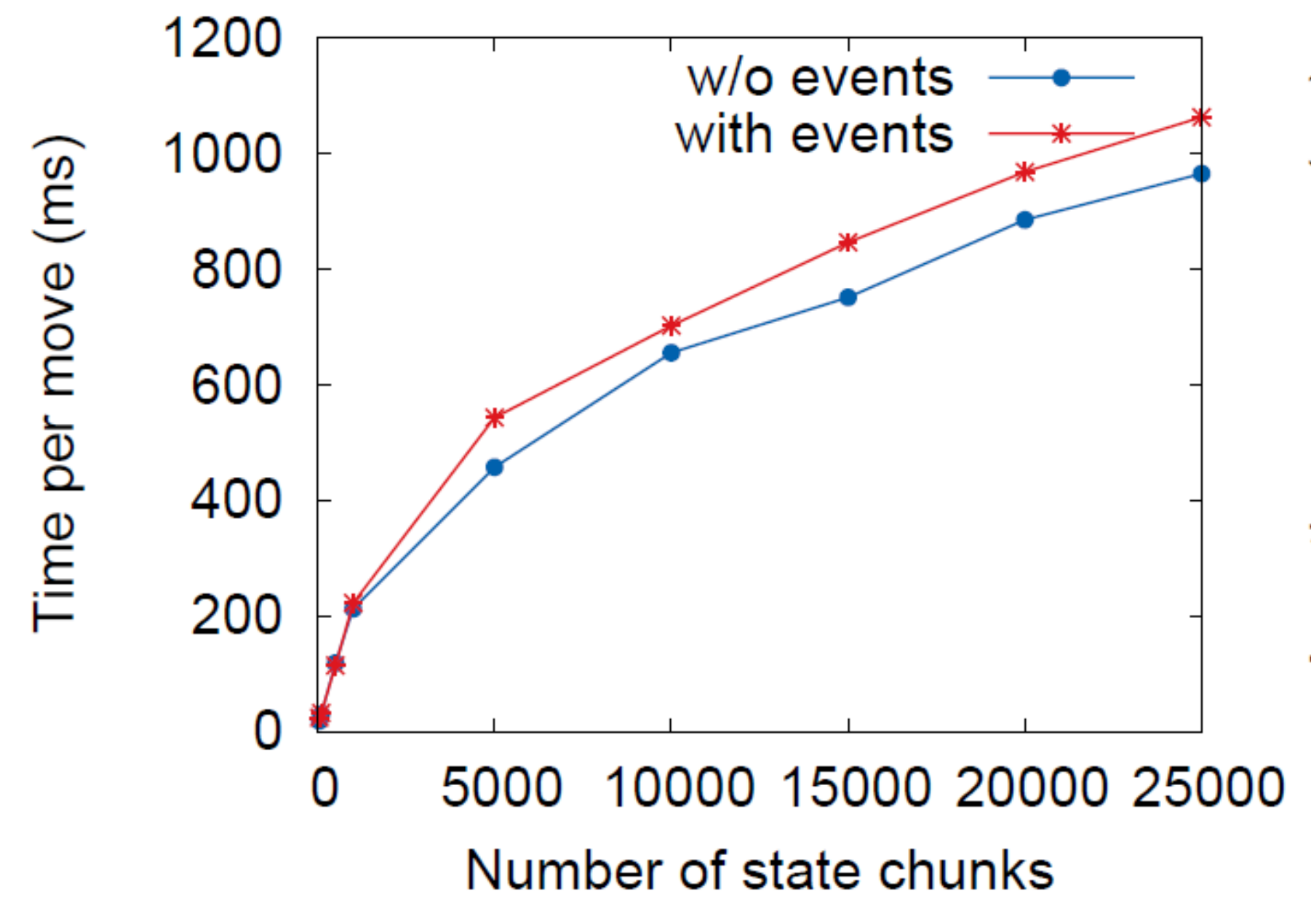
```
getSupport (<HeaderFieldList>)
putSupport ([<HeaderFieldList>:<EncryptedChunk>])
delSupport (<HeaderFieldList>)
```

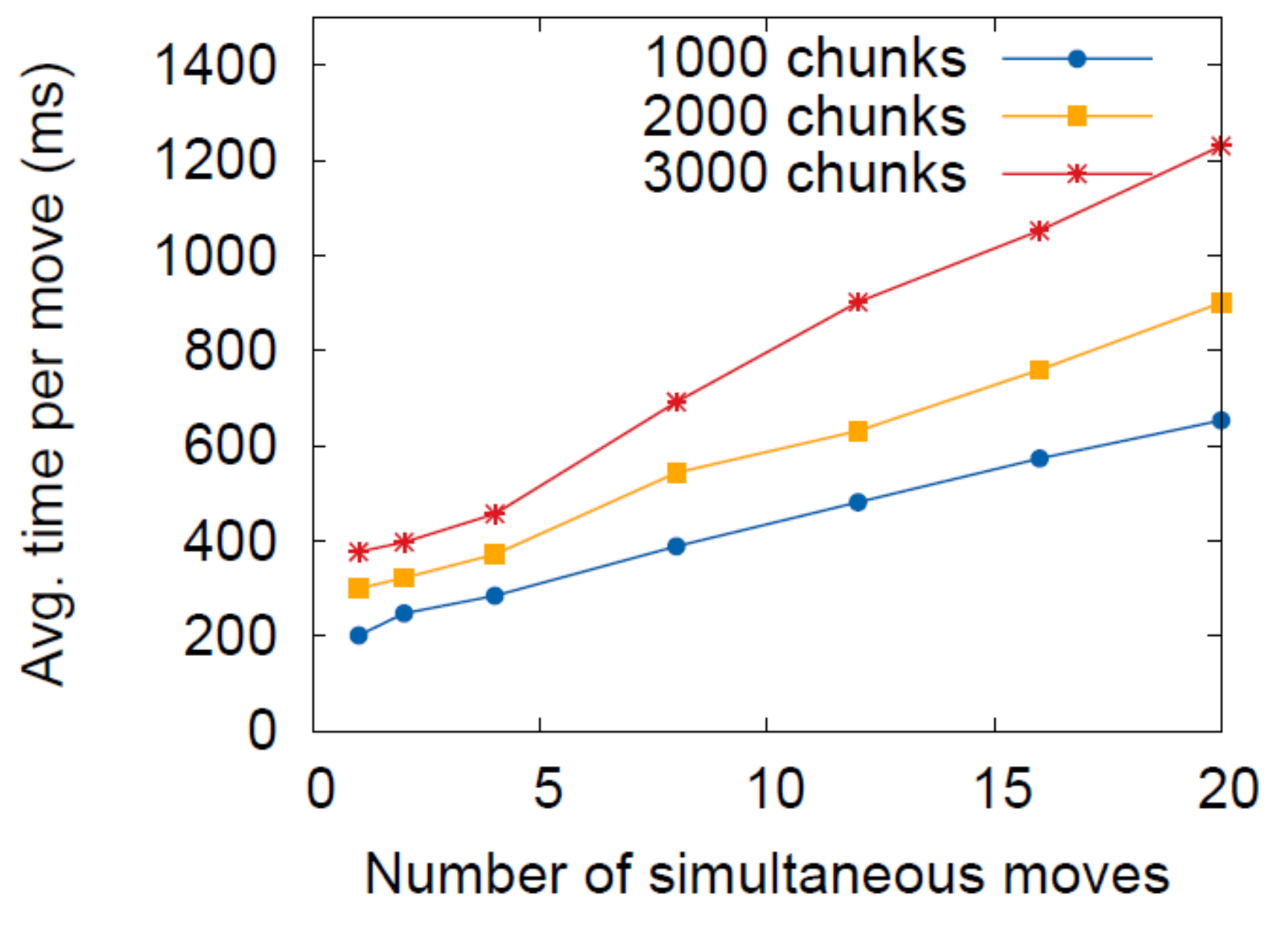
State Events

- Need to ensure state changes (e.g. move) are atomic
- Type of events : Packet re-process, Packet re-direct

Evaluation

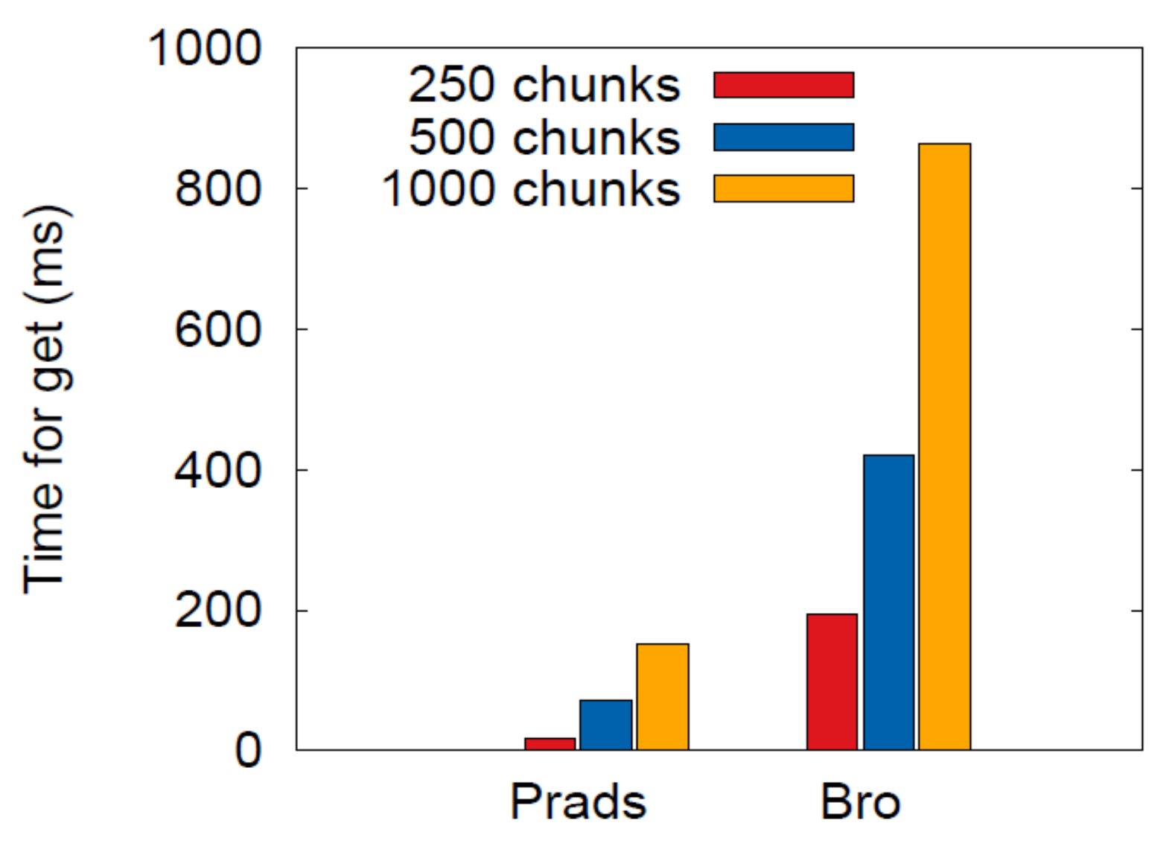
Controller performance





Controller handles operations efficiently and is scalable

Middlebox performance



Middlebox	Normal operation	During get
Bro	6.93ms	7.06ms
Smart RE	0.781ms	0.790ms

Average per-packet processing latency

Middleboxes maintain performance during operations and implement operations efficiently

References
[1] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield. Split/merge: System support for elastic execution in virtual middleboxes. In NSDI, 2013.
[2] V. Sekar, R. Krishnaswamy, A. Gupta, and M. K. Reiter. Network-wide deployment of intrusion detection and prevention systems. In CoNEXT, 2010.