Design and Implementation of a Framework for Software-Defined Middlebox Networking

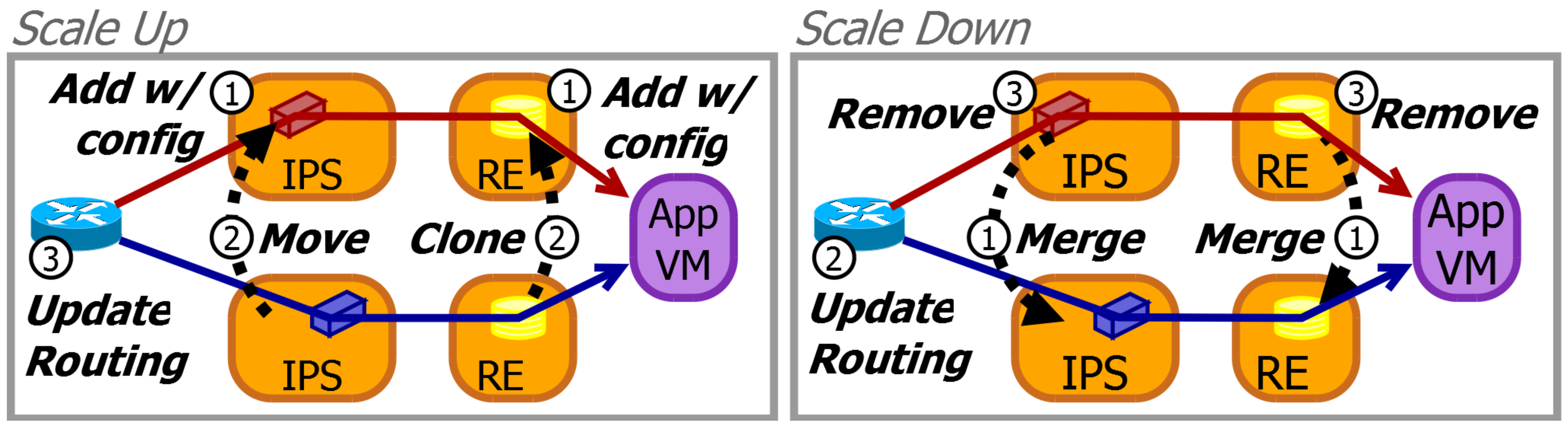
Aaron Gember, Robert Grandl, Junaid Khalid, Aditya Akella

{agember,rgrandl,junaid,akella}@cs.wisc.edu

Motivation

Virtual middleboxes are becoming increasingly attractive because of the flexibility and agility they enable. Several frameworks (e.g., Stratos, SIMPLE) have been developed for managing the composition and provisioning of virtual middlebox.

However, control over how middleboxes examine and modify network traffic is limited: policies and parameters are manipulated using narrow, middlebox-specific interfaces, while internal algorithms and state are completely inaccessible and unmodifiable. A lack of fine-grained control over middleboxes and their state precludes correct and well performing implementation of control scenarios that involve re-allocating live flows across middleboxes: e.g., horizontal scaling.



Potential Solutions

Configuration Control

Standardized configuration protocols (e.g., SIMCO, SNMP) – only provides control over externally-created state, not middlebox-created state

Control over middlebox configuration and routing [2] – enables an optimal configuration of middleboxes and the network, but the new configuration cannot fully take affect until all existing flows have finished

Internal State Control

Virtual machine snapshot – clones more state than necessary, possibly leading to incorrect middlebox behavior; does not support merging

Vendor-provided controller – vendors can transfer state between middleboxes based on detailed knowledge of middlebox internals, but the controller's state decisions may conflict with network-wide objectives

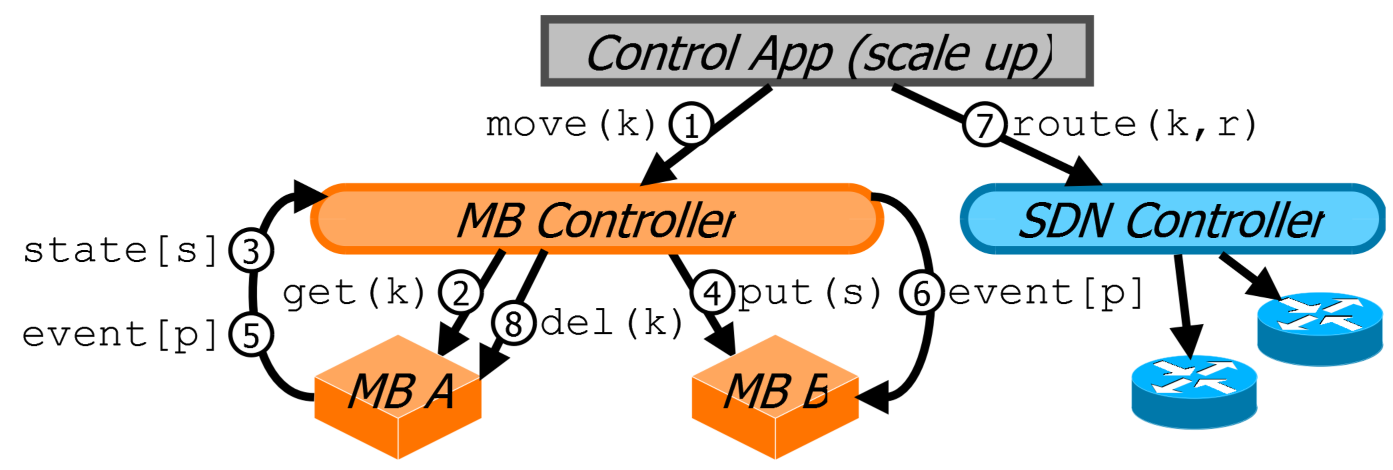
Application-level library [1] – middleboxes call the library to allocate, free, and access state, and a controller calls the library to import/export state; limited support for state that is shared across flows

Middlebox State Taxonomy

Role	Definition	IPS Examples	Partitioning	Operations
Configuration	Defines and tunes middlebox behavior	Rules, alert level	Shared only	Middlebox reads
Supporting	Guides middlebox decisions and actions based on past traffic	Connection records	Per-flow & shared	Middlebox reads & writes
Reporting	Quantify observations and decisions	Packet counters, alert logs	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) Controller issues a get request
3) Send the requested state
4) Insert the state
5) Issue reprocessing event to ensure atomic state change
6) Reprocess packet to update state
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

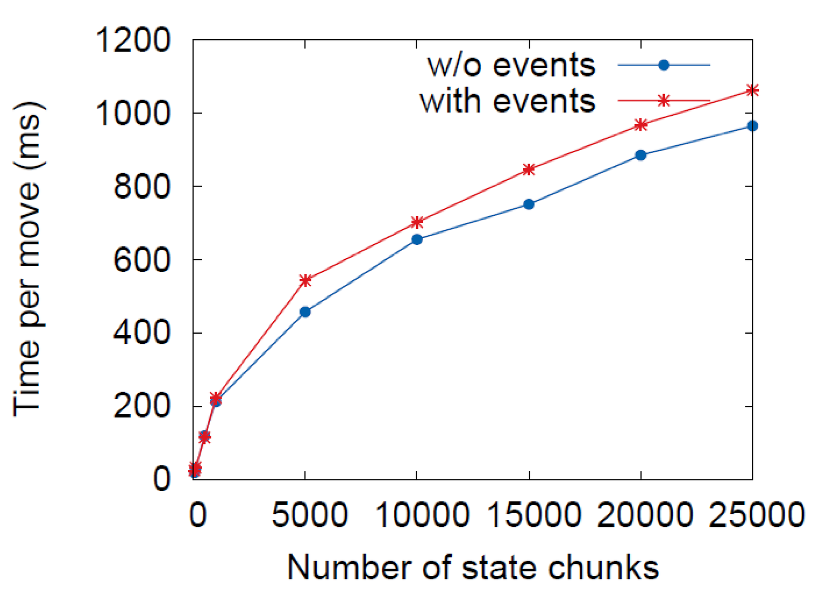
- Desire to conceal state structure and protect its integrity
- Need to move, clone, and merge state at fine granularity

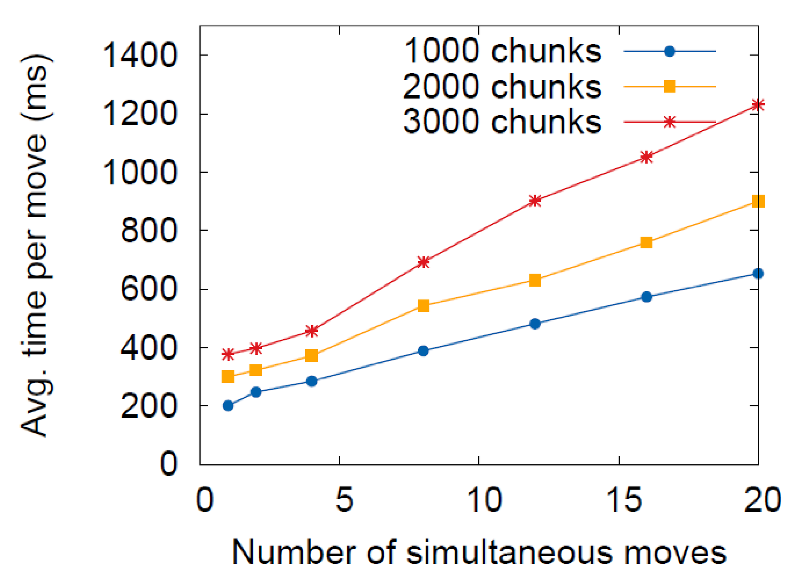
```
getSupport (<HeaderFieldList>)
putSupport ([<HeaderFieldList>:<EncryptedChunk>])
delSupport (<HeaderFieldList>)
```

State Events

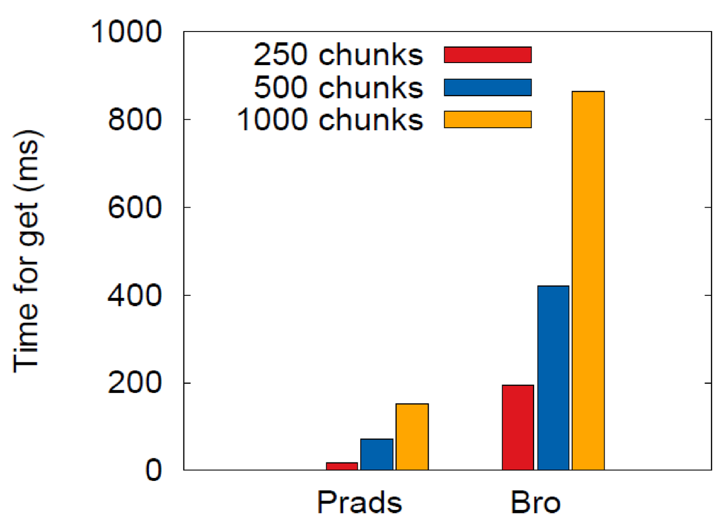
- Need to ensure state changes (e.g. move) are atomic
- Type of events: Packet re-process, Packet re-direct

Evaluation





Controller handles operations efficiently and is scalable



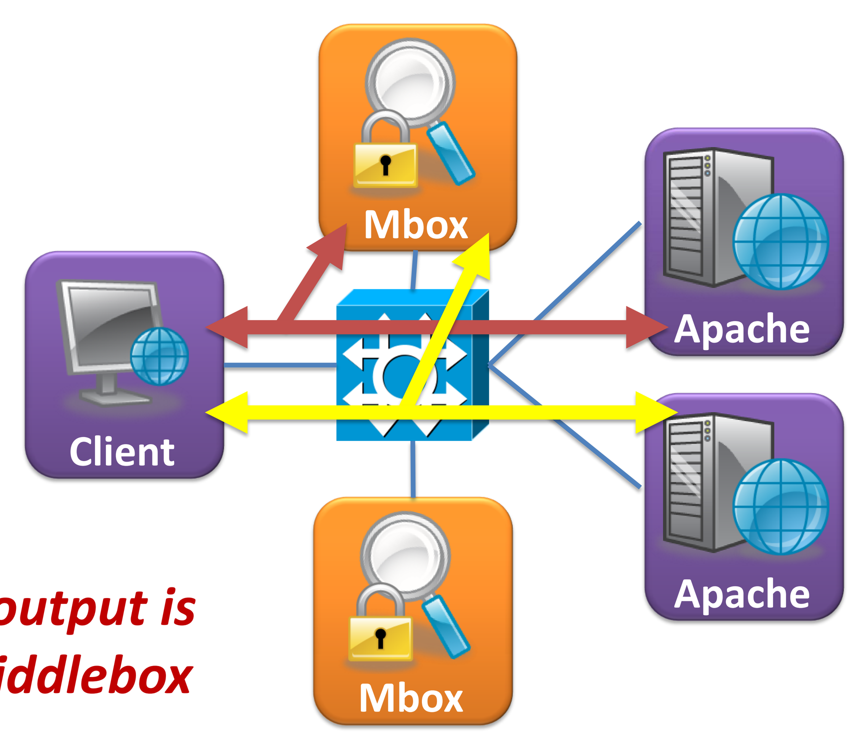
Middlebox	Without operation	During get operation
Bro	6.93ms	7.06ms
Smart RE	0.781ms	0.790ms

Average per-packet processing latency

Middleboxes maintain performance during operations and implement operations efficiently

Demonstration

1. Send copies of flows for both servers to the same middlebox
2. When network load increases, move state and flows for one server to a new middlebox
3. When load decreases, move state and flows back to the original middlebox



Observe that the middlebox's output is equivalent to using a single middlebox

Learn More

http://agember.com/go/OpenMB

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.