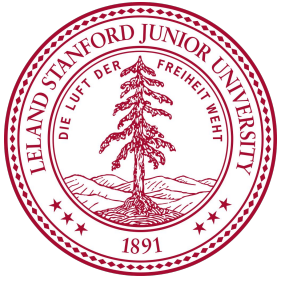
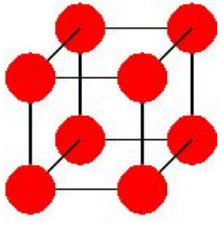
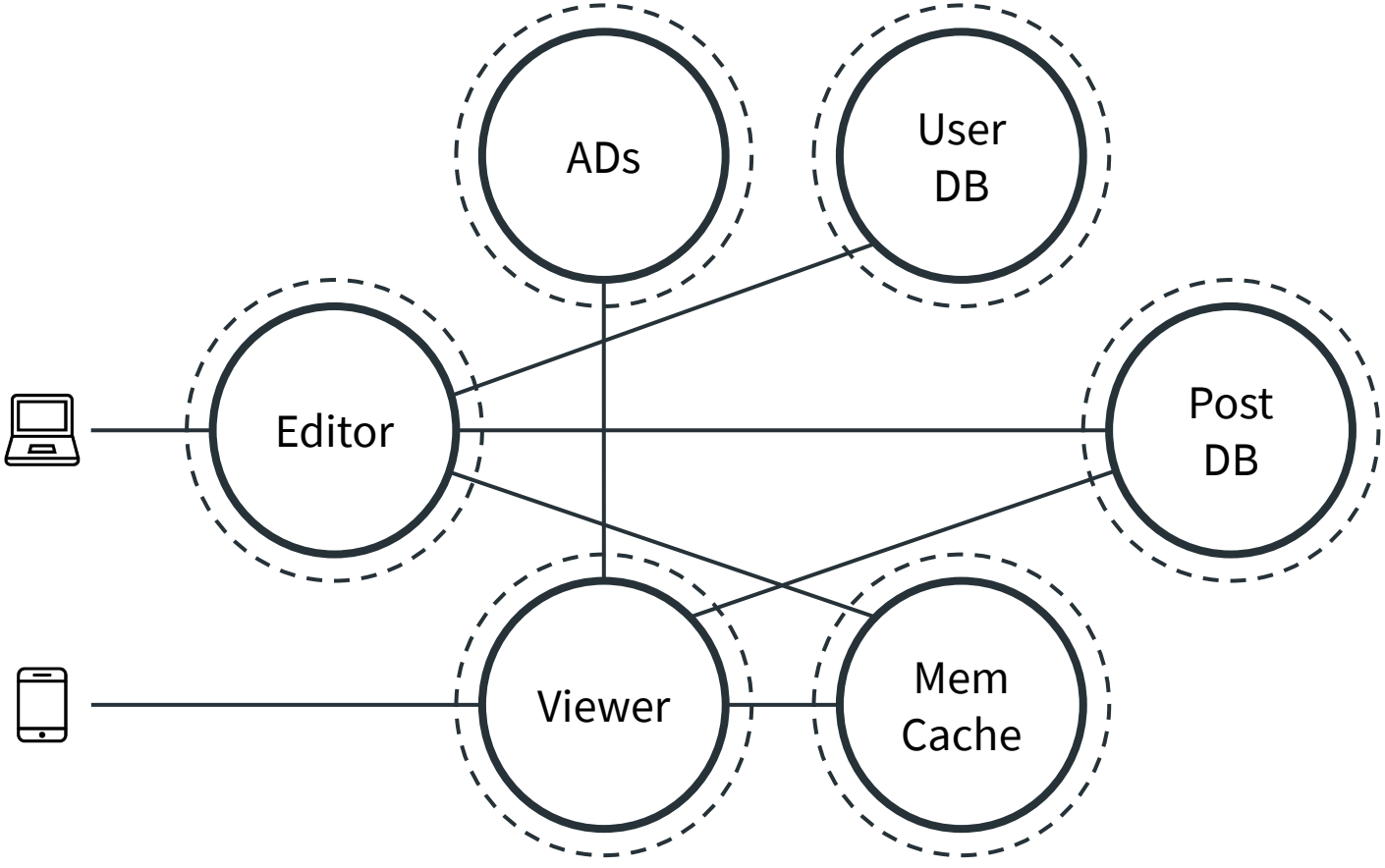
High Performance Service Oriented Computing

Nic McDonald
Stanford CVA Research Lab

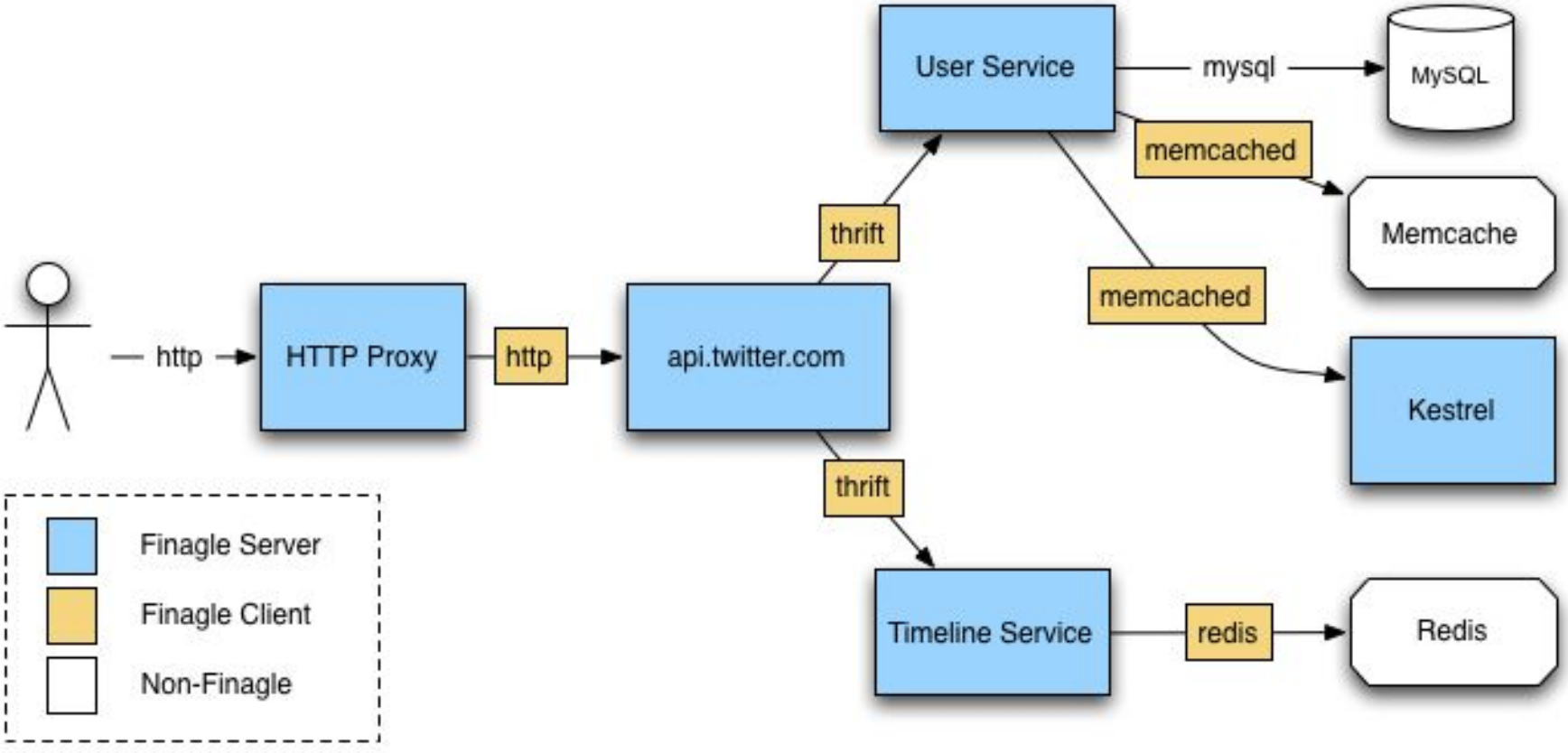




Motivation: Service Oriented Computing



Motivation: Twitter



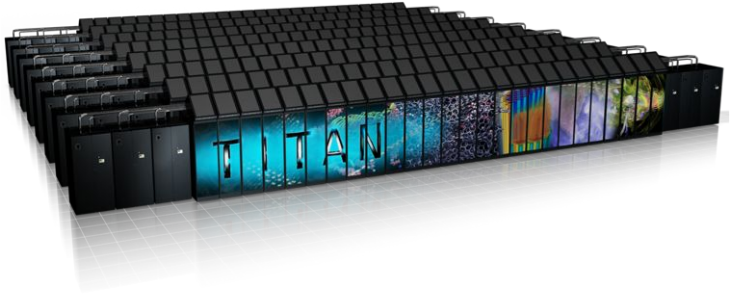
Motivation: Hailo

Motivation: High Performance Interconnects

“Lower latency will simplify application development, increase web application scalability, and enable new kinds of data-intensive applications that are not possible today.”

Rumble, Ongaro, Stutsman, Rosenblum, and Ousterhout.
“It’s time for low latency” in HotOS, vol. 13, 2011, pp. 11–11.

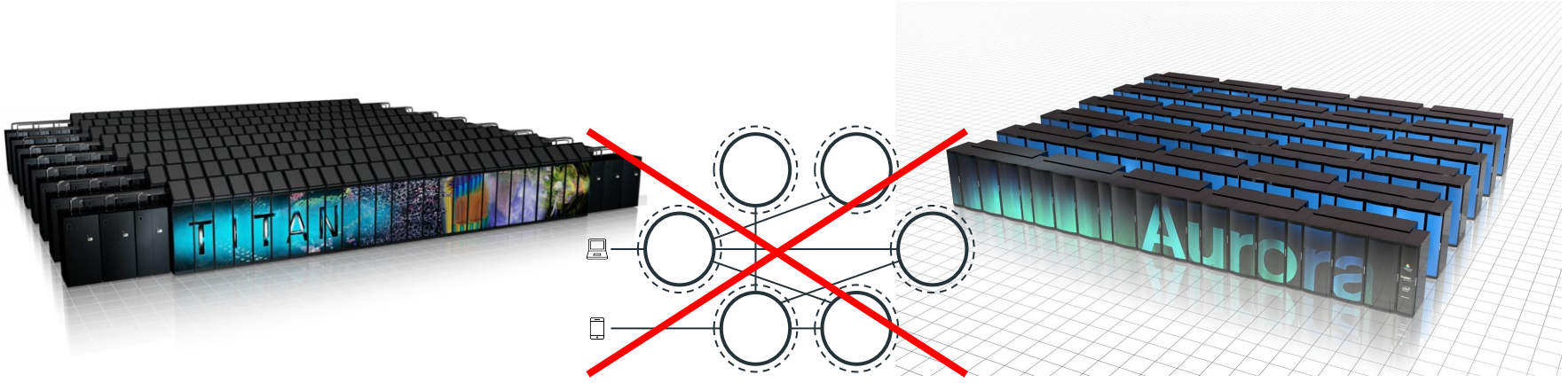
Motivation: High Performance Interconnects



"Lower latency will simplify application development, increase web application scalability, and enable new kinds of data-intensive applications that are not possible today."

Rumble, Ongaro, Stutsman, Rosenblum, and Ousterhout.
"It's time for low latency" in HotOS, vol. 13, 2011, pp. 11–11.

Motivation: High Performance Interconnects

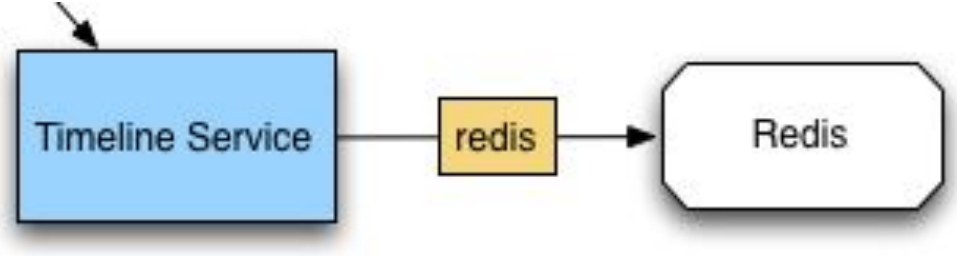


"Lower latency will simplify application development, increase web application scalability, and enable new kinds of data-intensive applications that are not possible today."

Rumble, Ongaro, Stutsman, Rosenblum, and Ousterhout.
"It's time for low latency" in HotOS, vol. 13, 2011, pp. 11–11.

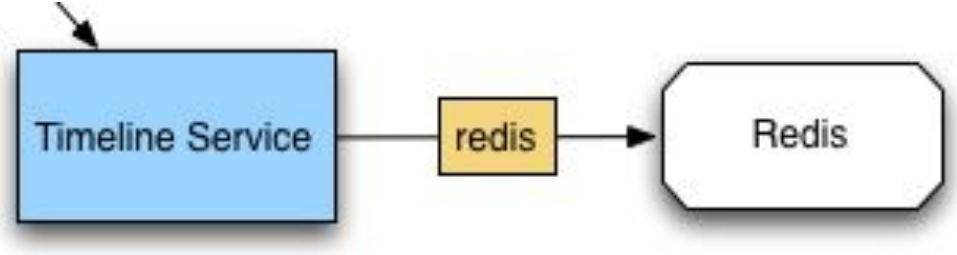
Motivation: Access Control Lists (N-ACL)

Protocol	Source		Destination	
	Address	Port	Address	Port
TCP	192.168.1.3	54321	10.0.2.10	123
TCP	192.168.1.3	43215	10.0.2.10	456
TCP	192.168.1.3	43215	10.0.3.10	456

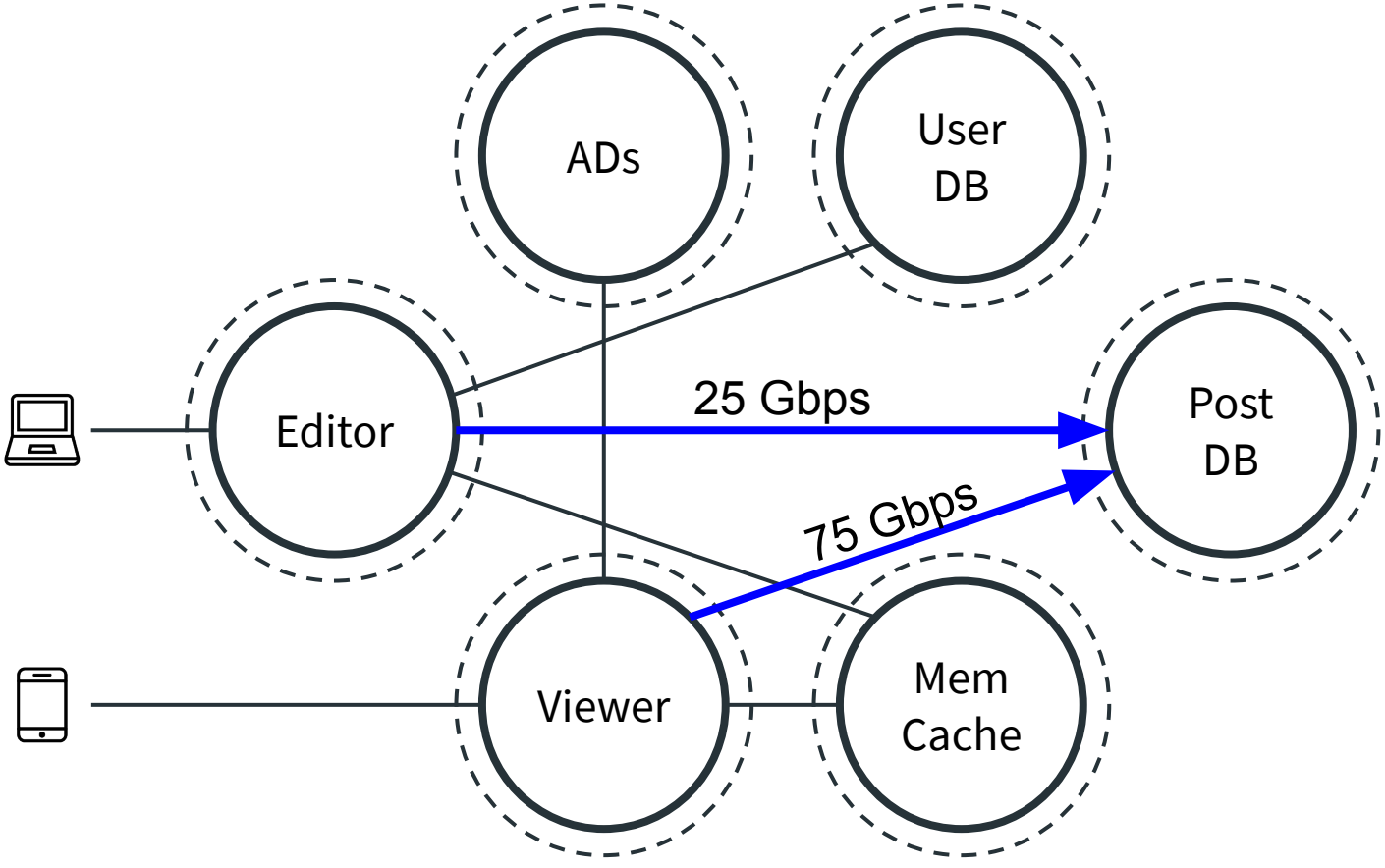


Motivation: Access Control Lists (S-ACL)

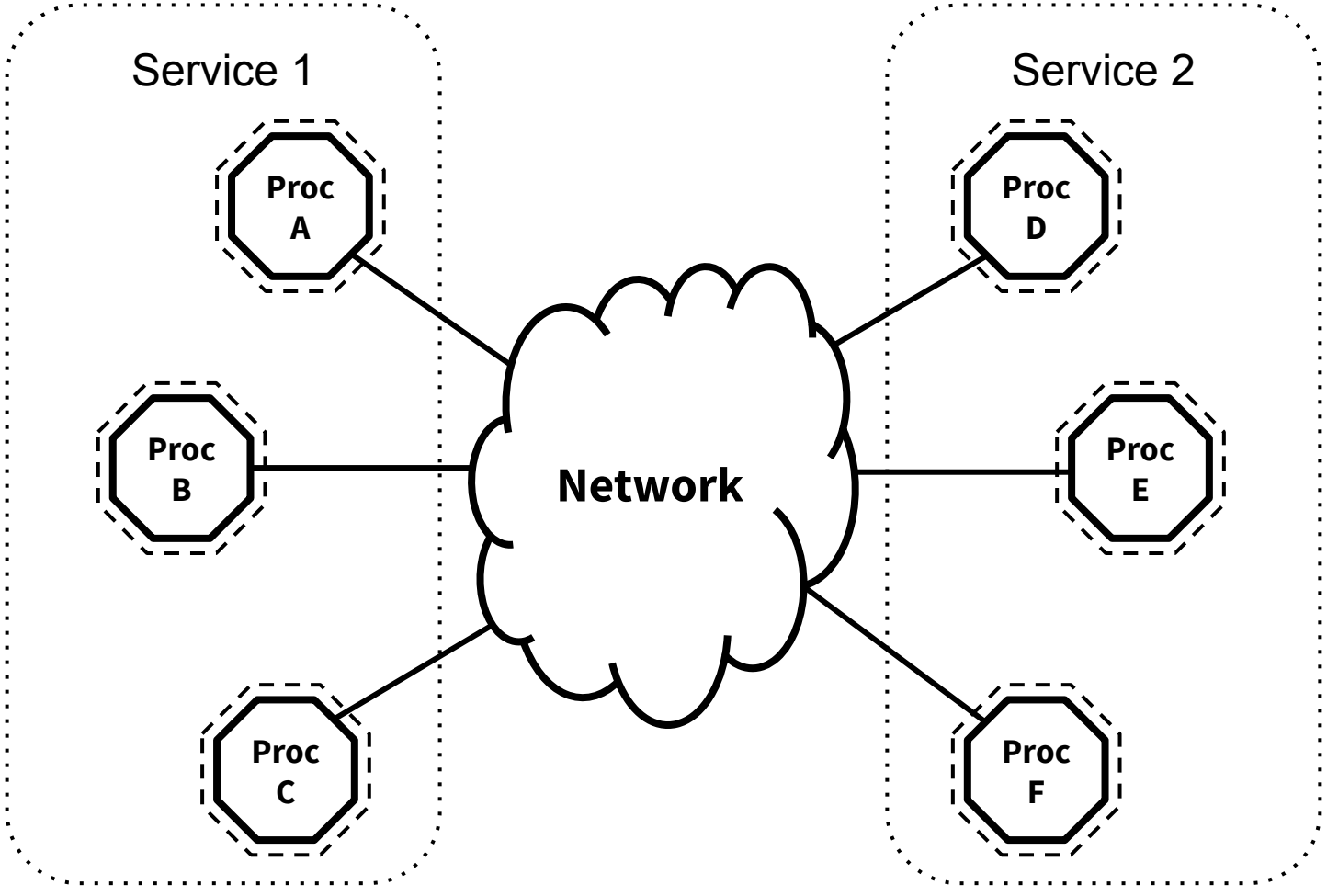
Source	Destination		
Service	**Service**	**Processes**	**Domains**
Timeline Service	Redis	6, 17, 32	Get, Set



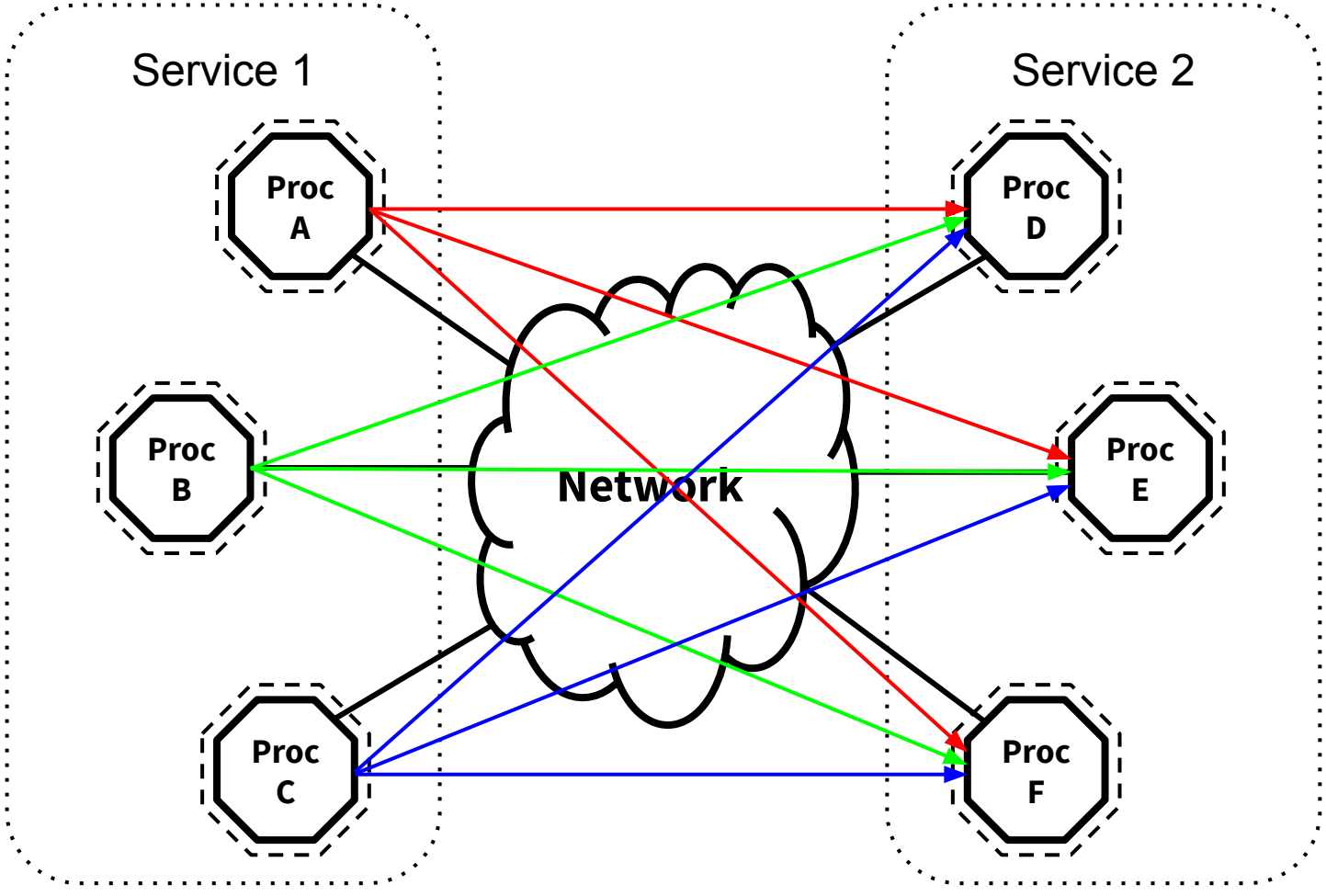
Motivation: Service Oriented Rate Control

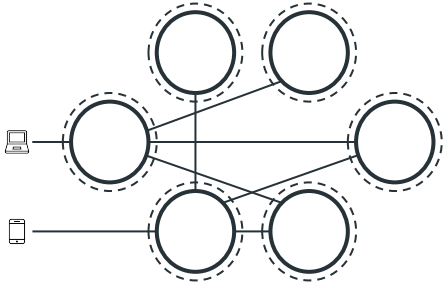


Motivation: Service Oriented Rate Control



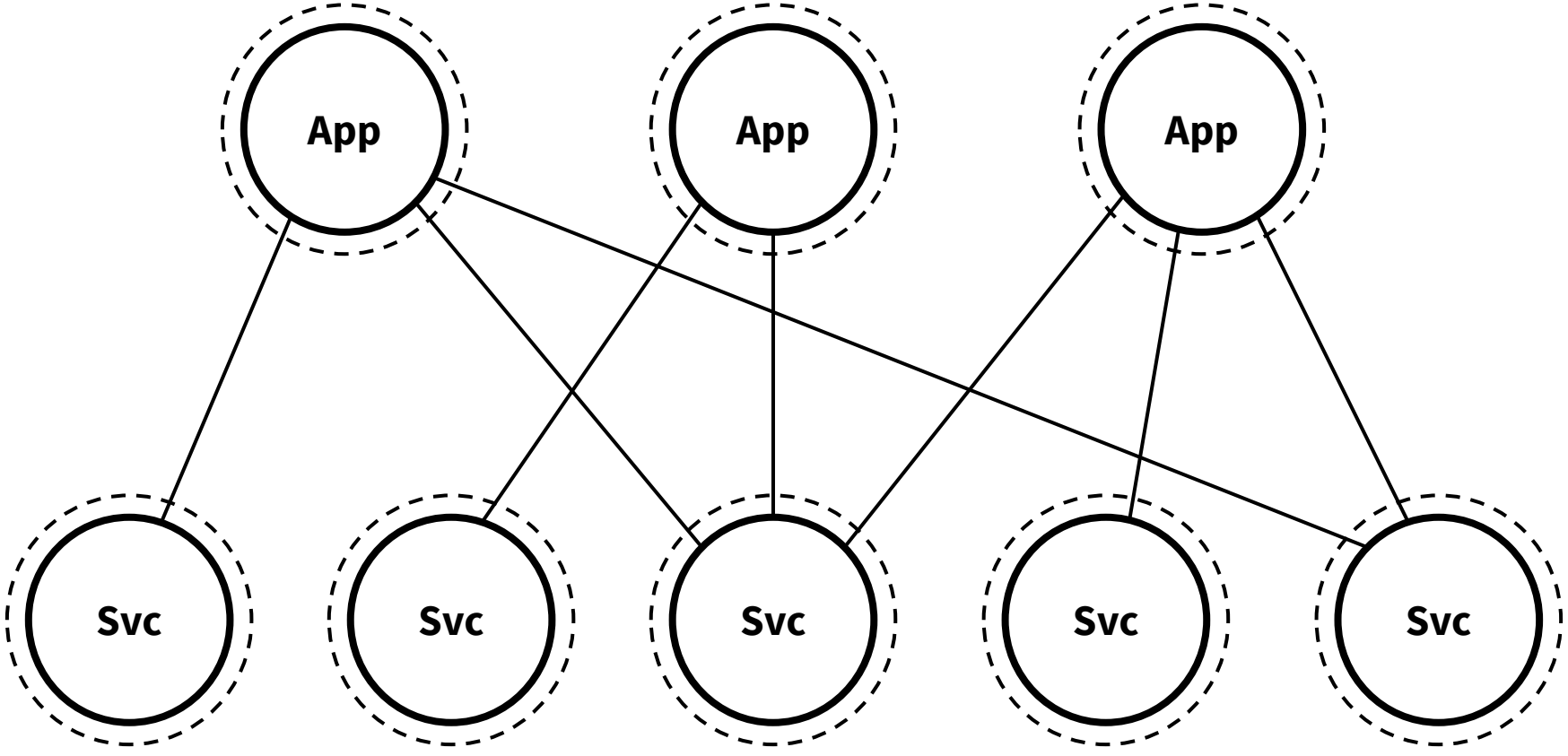
Motivation: Service Oriented Rate Control





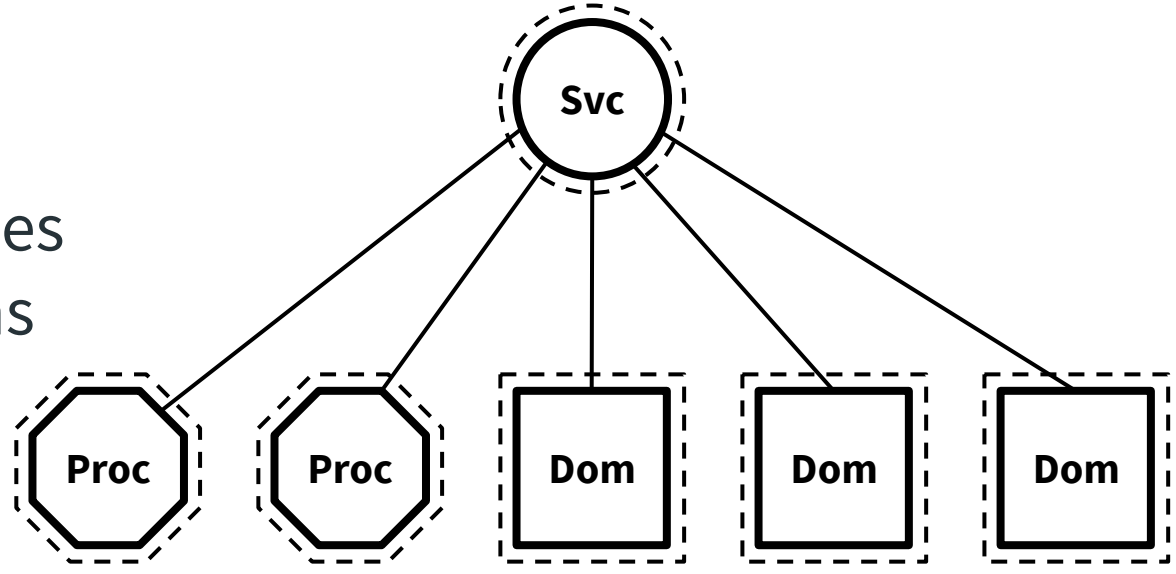
Service Oriented Application Model

Sikker: Service Oriented Application Model



Sikker: Service Oriented Application Model

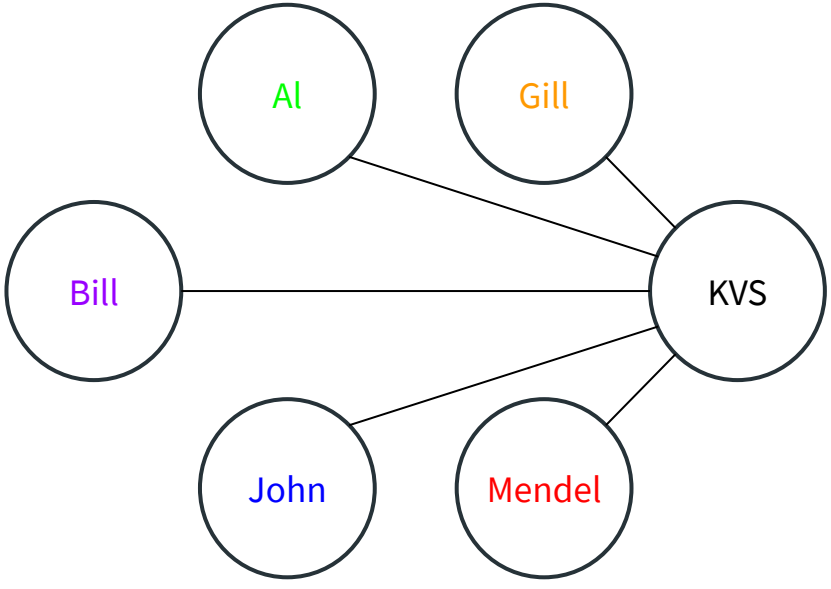
- Applications
  - Services
    - Processes
    - Domains



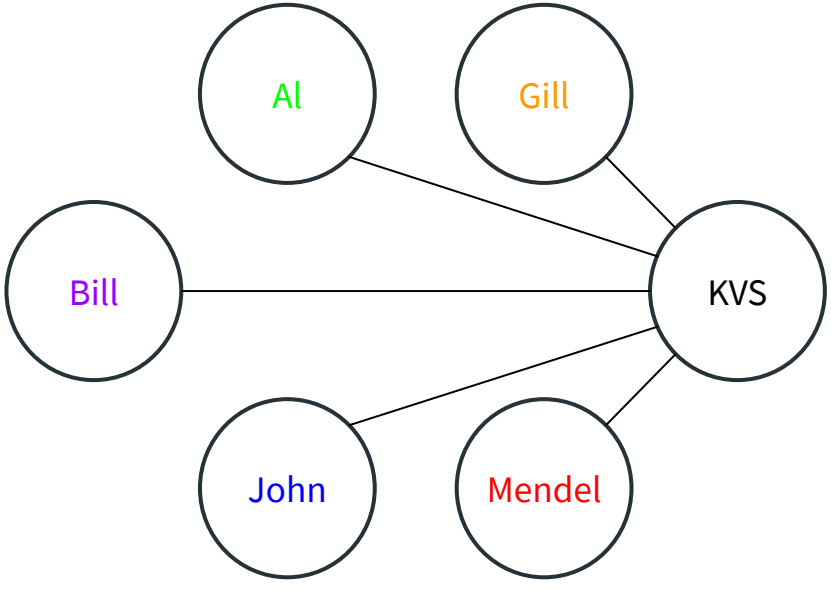
Process = execution unit (e.g., process, container, VM)

Domain = service-specific permission domain

Sikker: Service Oriented Application Model



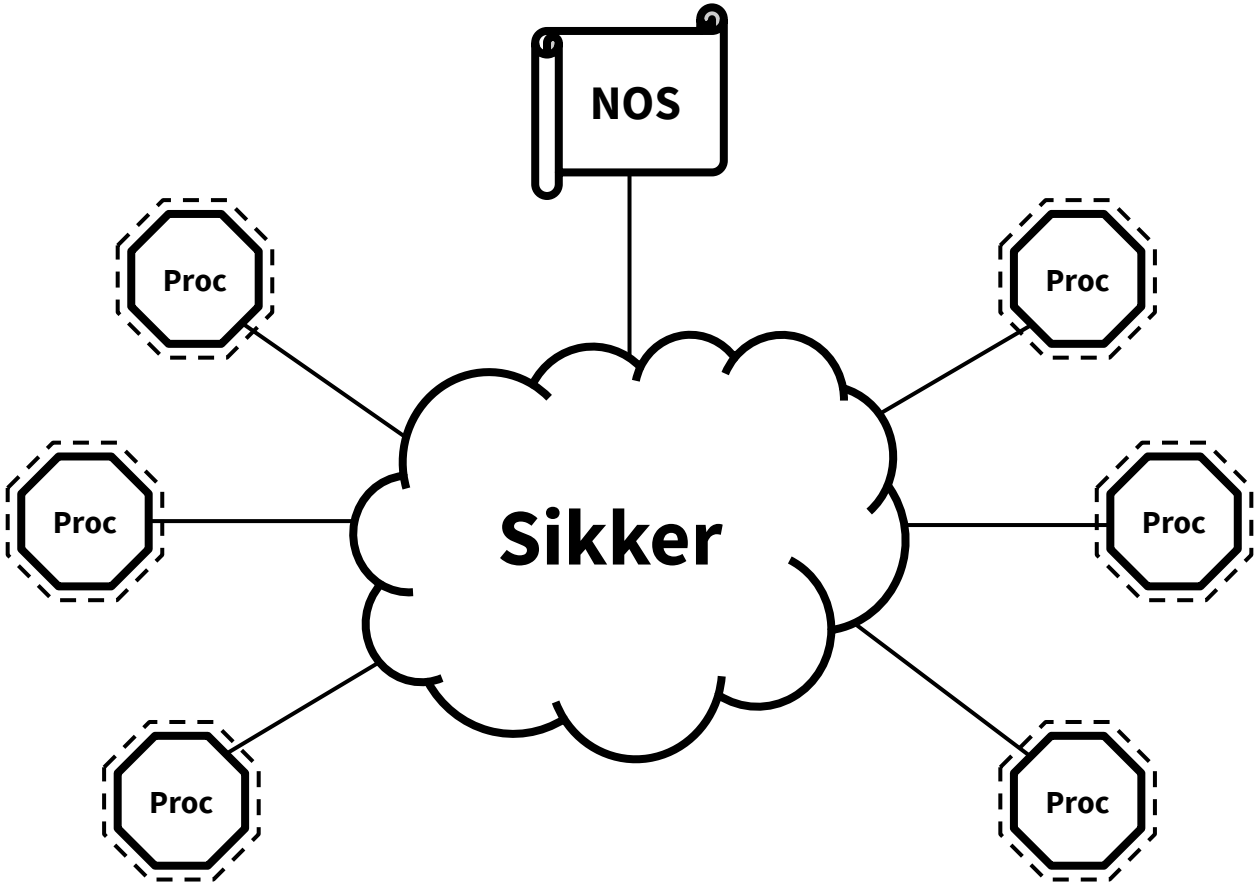
Sikker: Service Oriented Application Model



API Commands: Get, Set, Delete

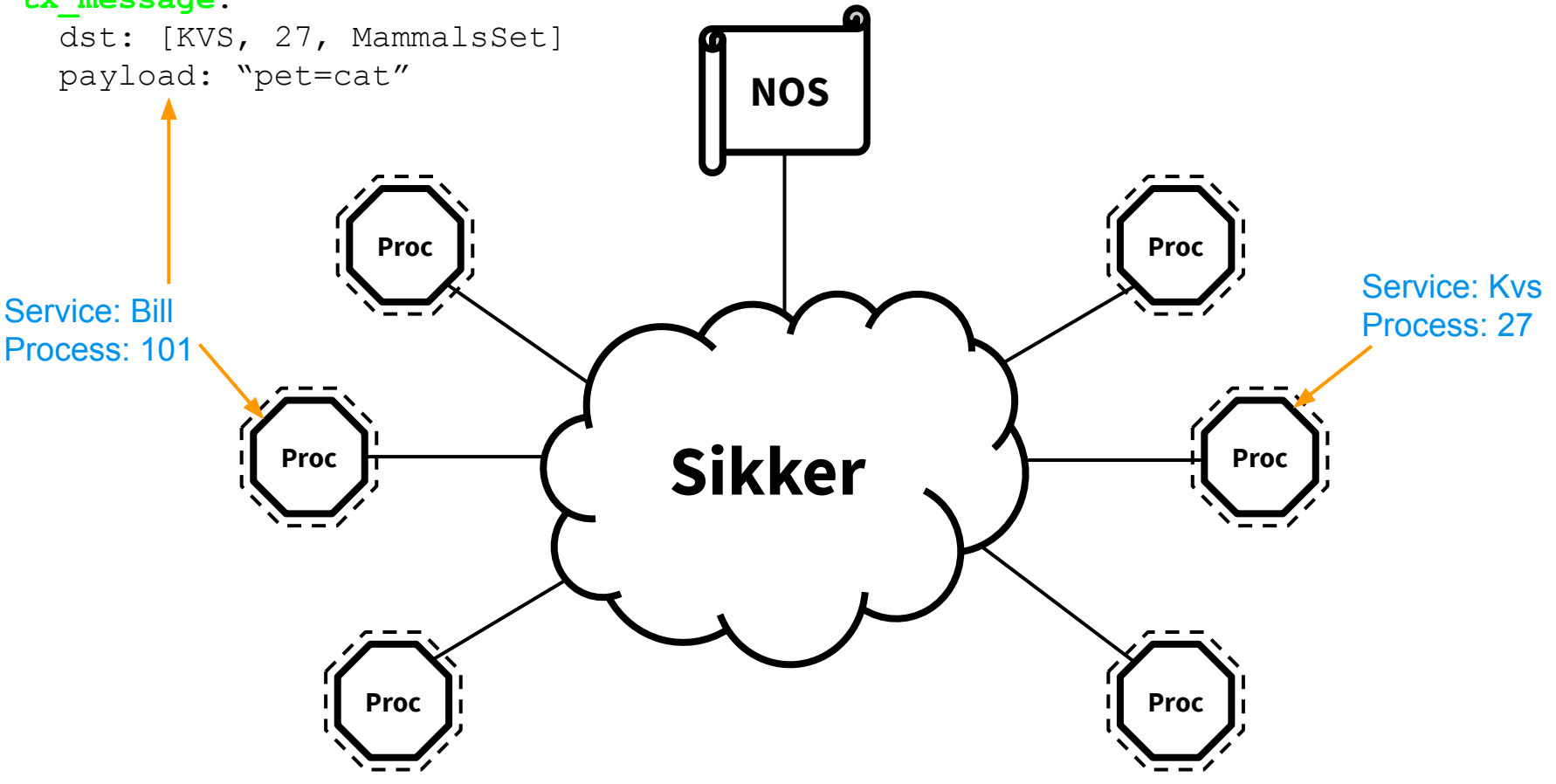
Table	Key	Value
Mammals	pet	dog
Engineering	department	Electrical
Locations	school	California
Companies	internship	Google
Sports	best	wakeboarding
Engineering	tool	oscilloscope
Sports	boring	baseball
Locations	born	Utah
Mammals	fastest	cheetah
Companies	career	HPE

Sikker: Service Oriented Application Model



Sikker: Service Oriented Application Model

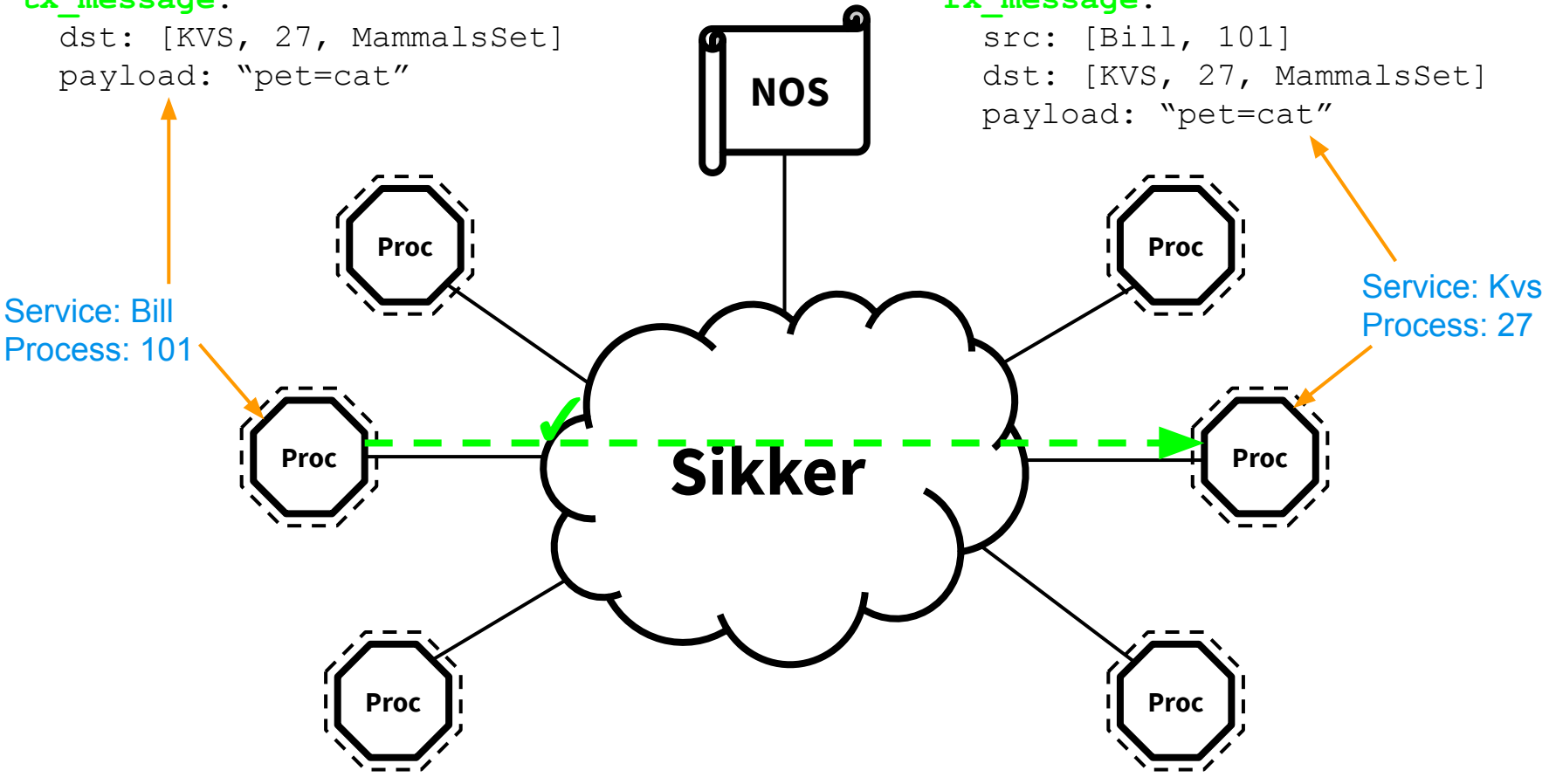
```
tx_message:
  dst: [KVS, 27, MammalsSet]
  payload: "pet=cat"
```



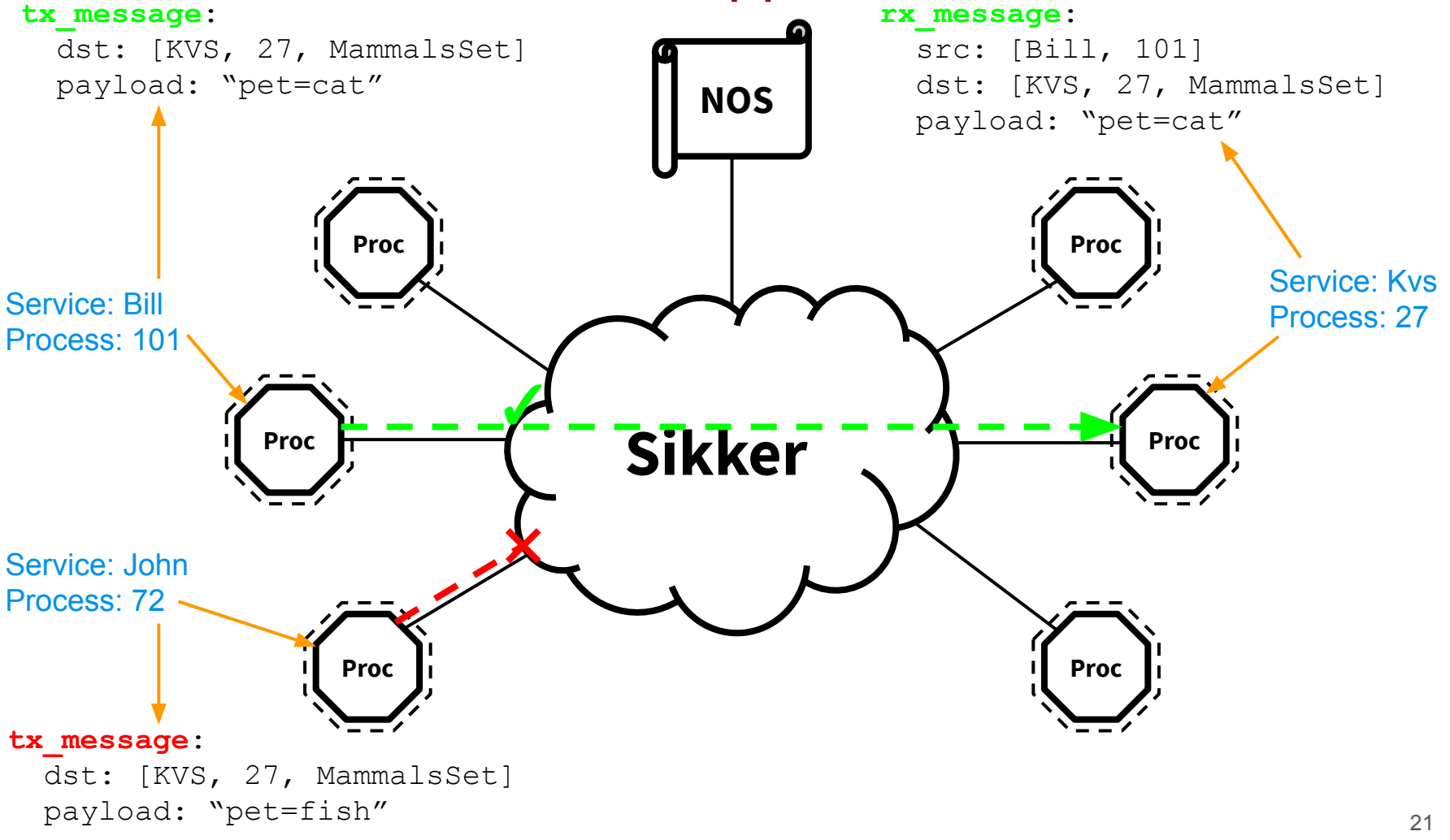
Sikker: Service Oriented Application Model

```
tx_message:
  dst: [KVS, 27, MammalsSet]
  payload: "pet=cat"
```

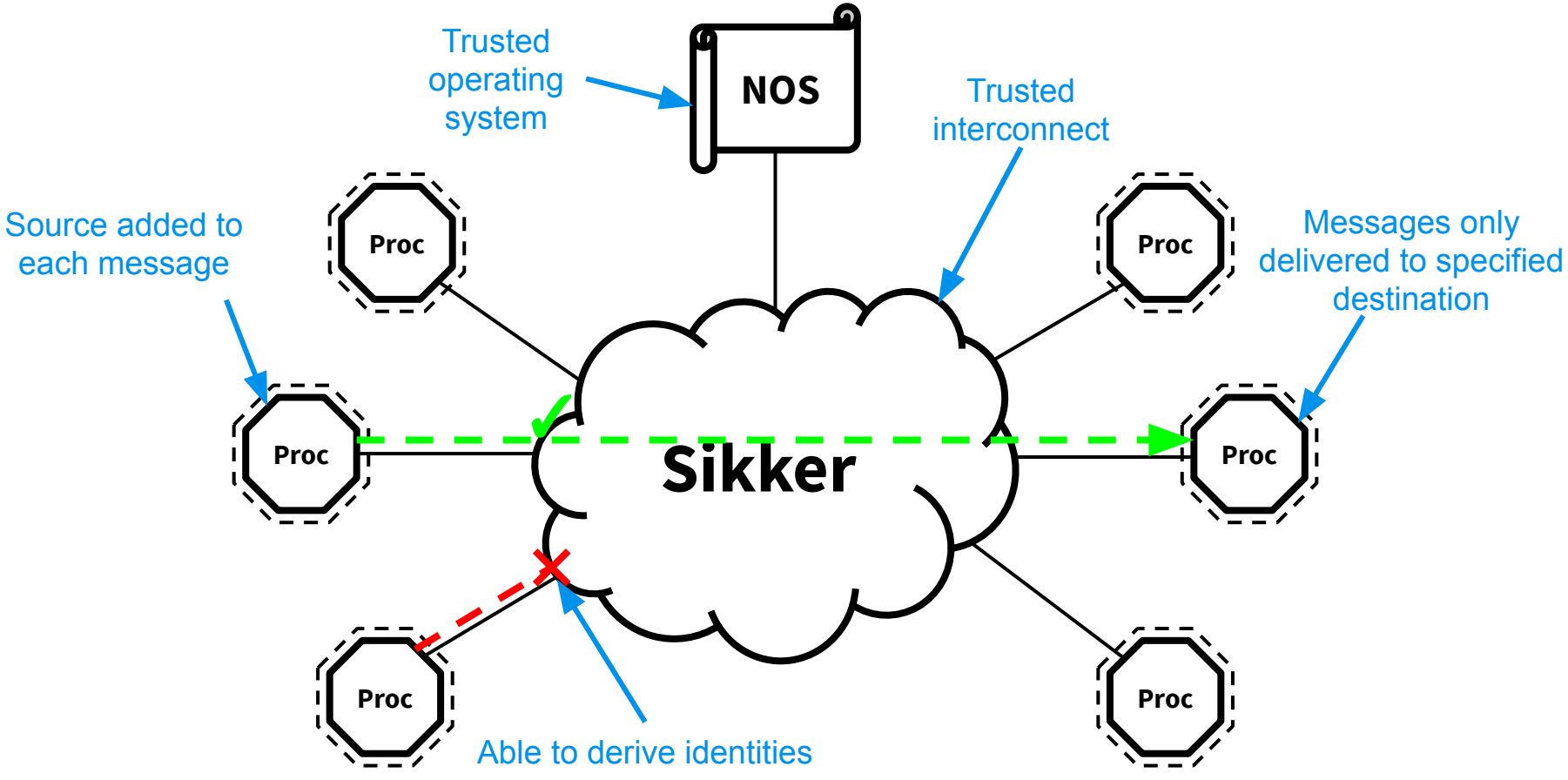
```
rx_message:
  src: [Bill, 101]
  dst: [KVS, 27, MammalsSet]
  payload: "pet=cat"
```



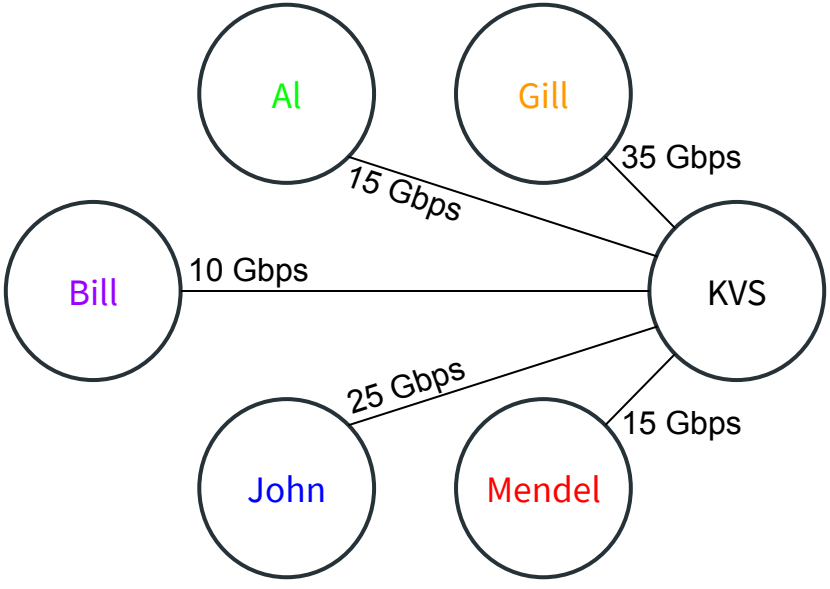
Sikker: Service Oriented Application Model



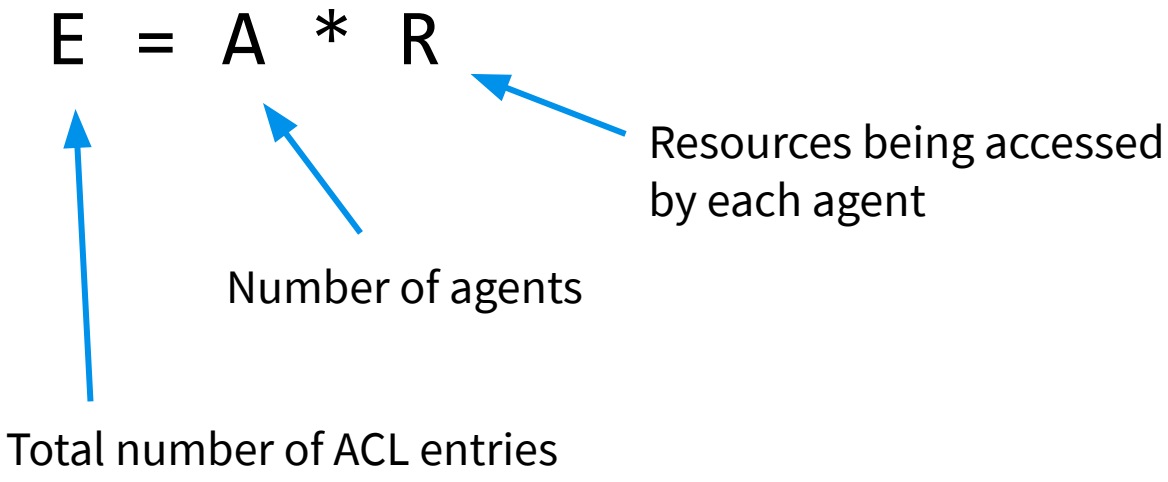
Sikker: Service Oriented Application Model



Sikker: Service Oriented Rate Control



Sikker: Scalability



Sikker: Scalability

$$E = A * R$$

$$N_{NACL} = s_t p_s * s_a p_a d_a$$

$$N_{SACL} = s_t * s_a (p_a + d_a)$$

LEGEND:

s_t= Total Services

p_s= Processes per Service

s_a= Accessible Services

p_a= Accessible Processes

d_a= Accessible Domains

p_h= Processes per host

Sikker: Scalability

$$E = A * R$$

$$N_{NACL} = s_t p_s * s_a p_a d_a$$

$$N_{SACL} = s_t * s_a (p_a + d_a)$$

$$H_{NACL} = p_h * s_a p_a d_a$$

$$H_{SACL} = p_h * s_a (p_a + d_a)$$

LEGEND:

s_t= Total Services

p_s= Processes per Service

s_a= Accessible Services

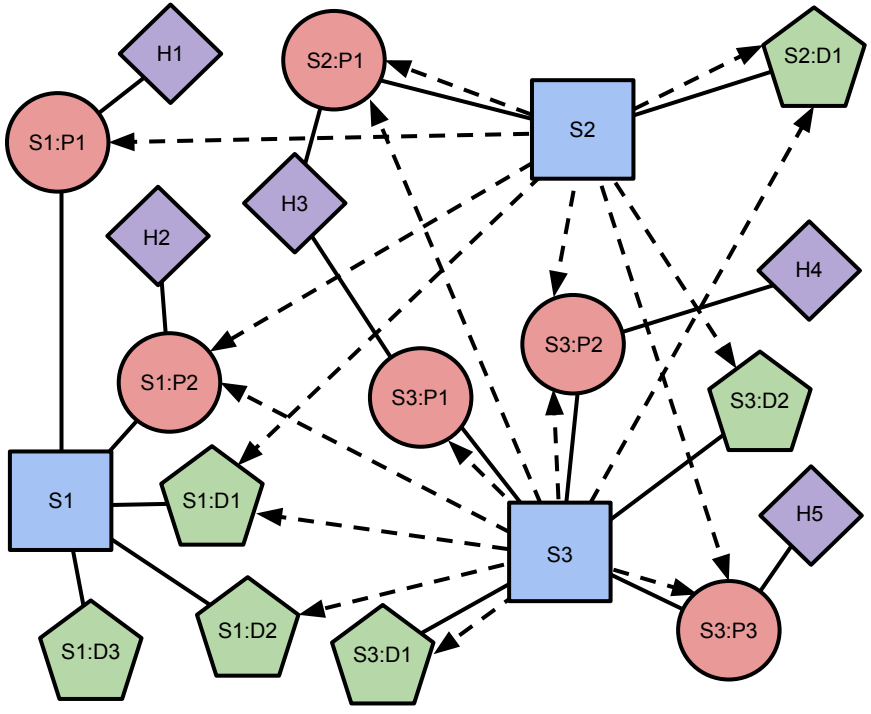
p_a= Accessible Processes

d_a= Accessible Domains

p_h= Processes per host

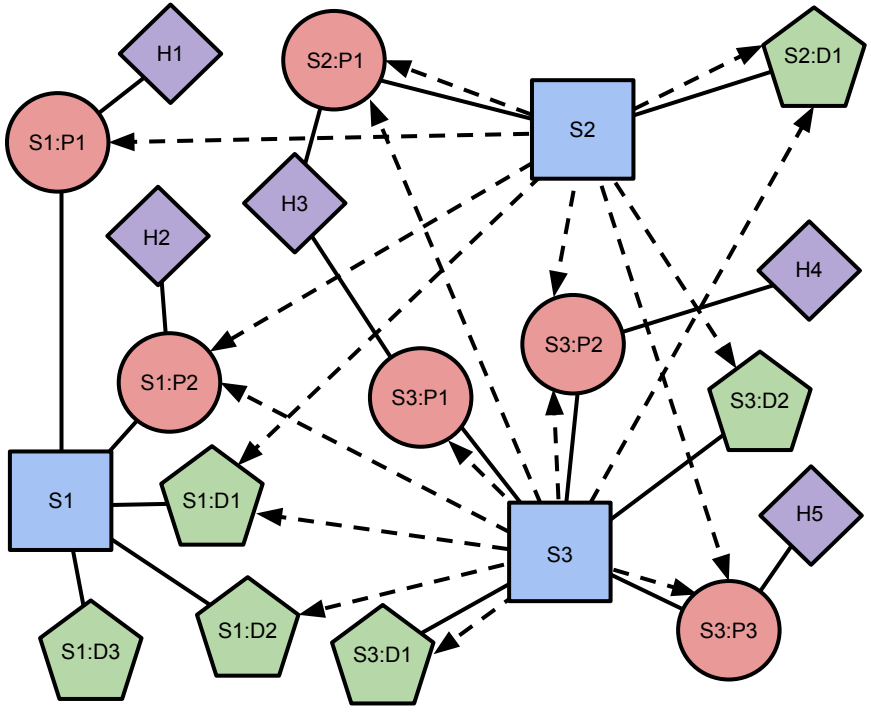
Sikker: Connectivity Model

Processes per NMU (per host)	16
Processes per Service	512
Domains per Service	256
Service coverage	20%
Process coverage	65%
Domain coverage	25%



Sikker: Connectivity Model

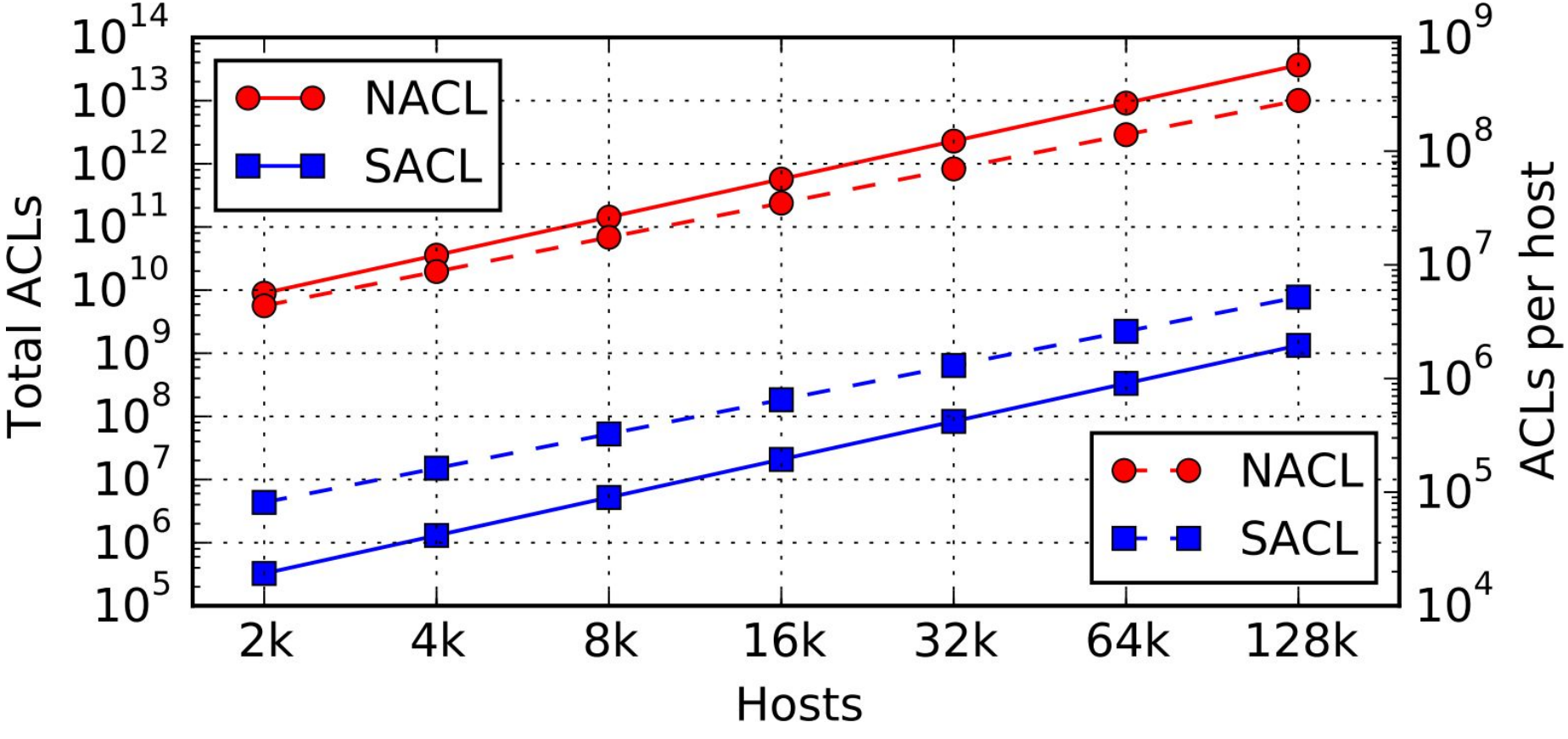
Processes per NMU (per host)	16
Processes per Service	512
Domains per Service	256
Service coverage	20%
Process coverage	65%
Domain coverage	25%



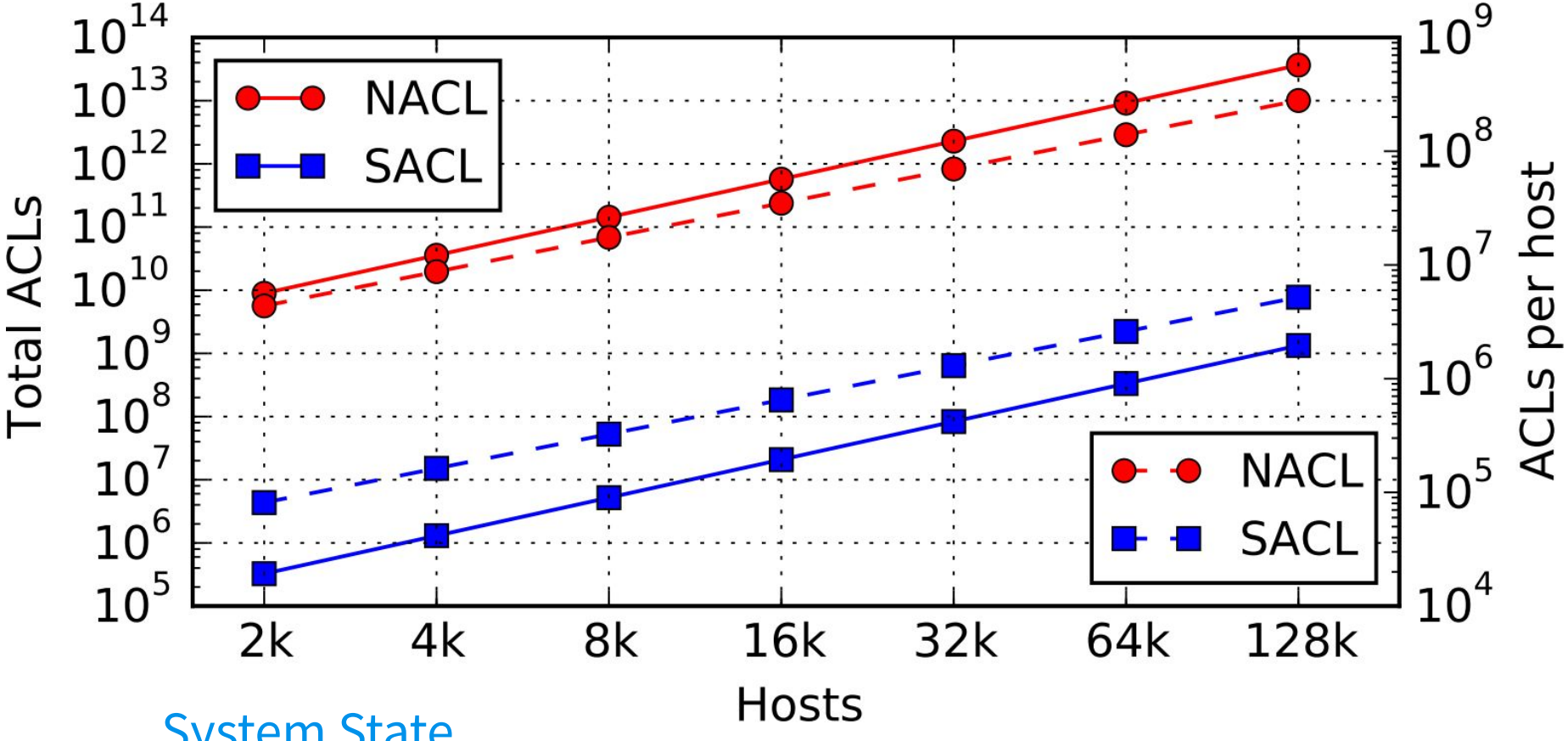
If there are 2^{17} (i.e. 131,072) hosts then:

# of total Processes	131,072 * 16 =	2,097,152
# of Services	131,072 / 512 =	4,096
Service connections	4,096 * 0.20 =	819
Processes per connection	512 * 0.65 =	333
Domains per connection	256 * 0.25 =	64

Sikker: Scalability



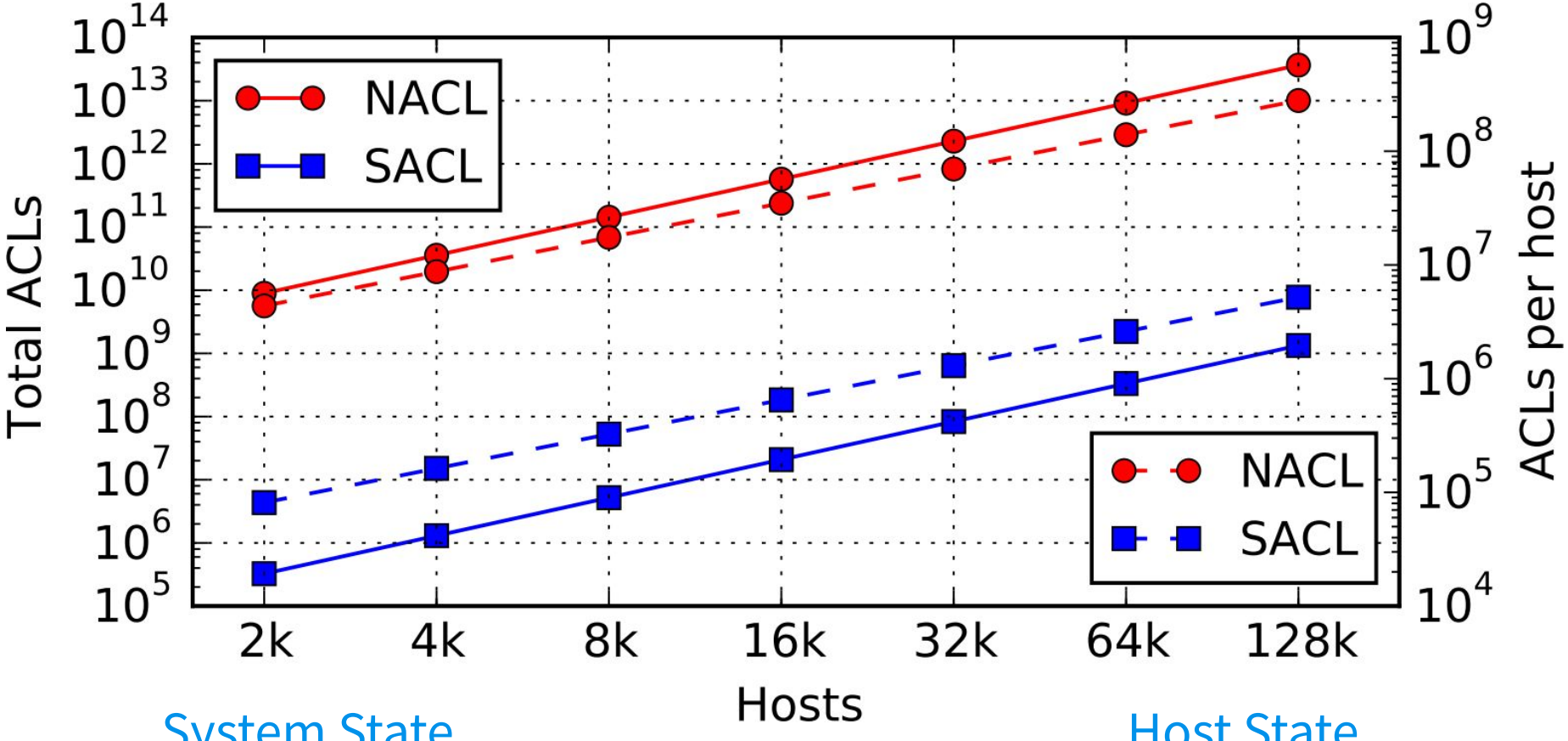
Sikker: Scalability



System State

146 TB vs. 5.33 GB

Sikker: Scalability



System State

146 TB vs. 5.33 GB

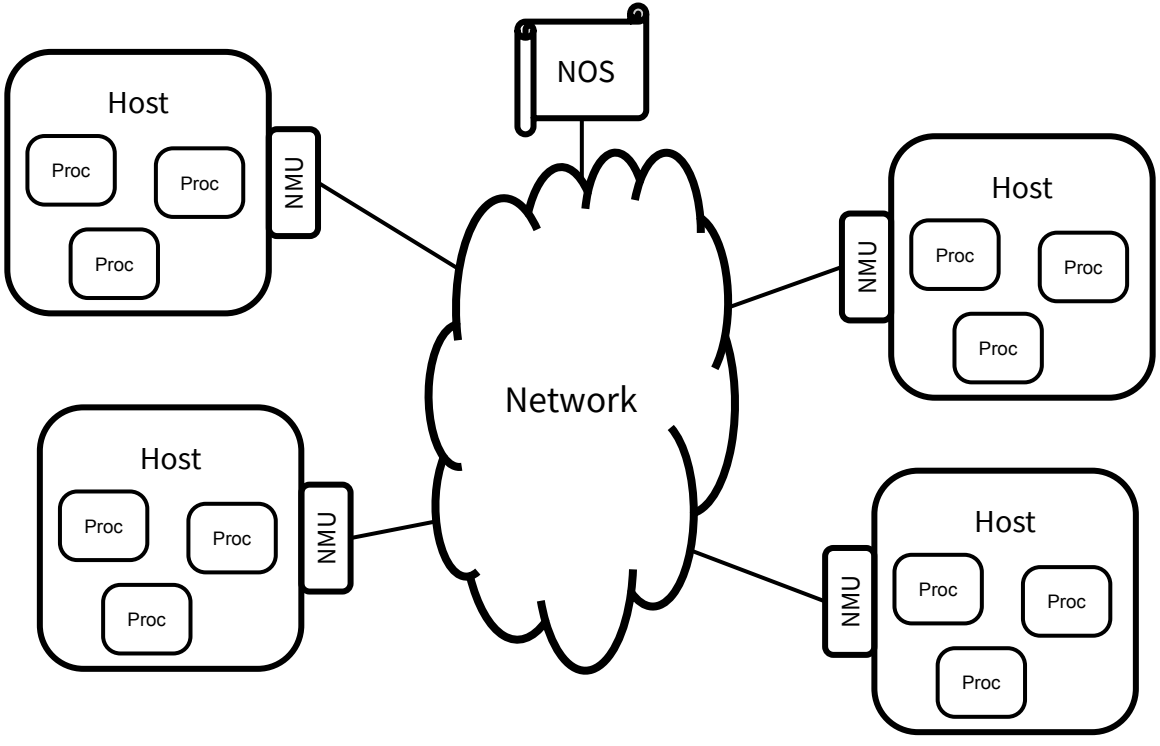
Host State

1.12 GB vs. 20.8 MB

High Performance Access Control

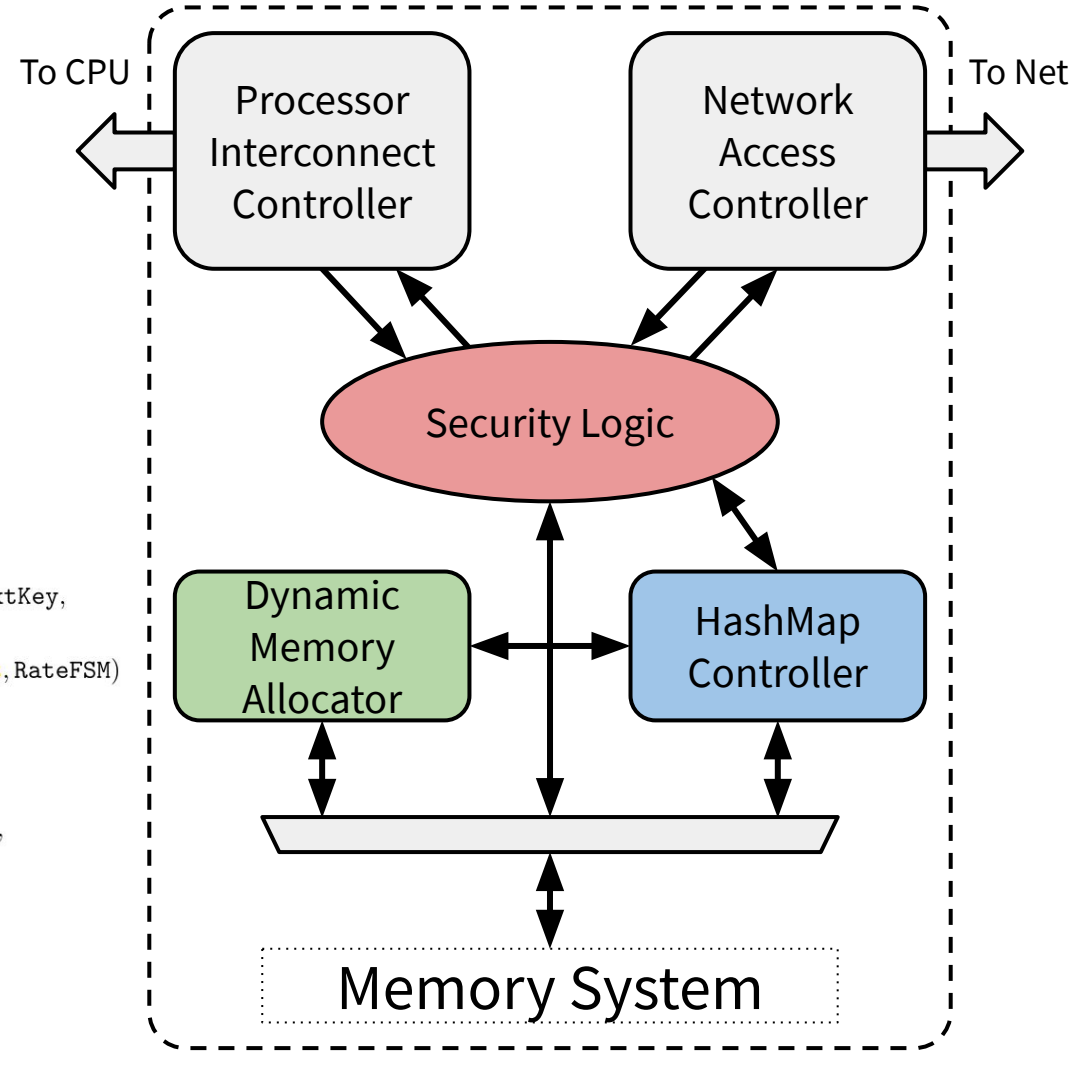
NMU: Architecture

The workhorse of Sikker is a new network interface controller (NIC) architecture called, the Network Management Unit (NMU)



NMU: Architecture

The NMU architecture is a data structure accelerator specifically for managing nested hashmaps.



$\texttt{IndexMap}: (\texttt{LocalService}, \texttt{LocalProcess}) \rightarrow \texttt{LocalIndex}$

$\texttt{InfoMap}: \texttt{LocalIndex} \rightarrow (\texttt{LocalService}, \texttt{LocalProcess}, \texttt{OtpNextKey}, \texttt{PermissionMap}, \texttt{OtpMap}, \texttt{PeerSet})$

- $\texttt{PermissionMap}: \texttt{RemoteService} \rightarrow (\texttt{ProcessMap}, \texttt{DomainSet}, \texttt{RateFSM})$
  - $\texttt{ProcessMap}: \texttt{RemoteProcess} \rightarrow \texttt{Address}$
  - $\texttt{DomainSet}: \texttt{RemoteDomain}$
- $\texttt{OtpMap}: \texttt{OtpKey} \rightarrow (\texttt{RequesterService}, \texttt{RequesterProcess}, \texttt{RecipientService}, \texttt{RecipientProcess}, \texttt{RecipientDomain}, \texttt{RecipientAddress})$
- $\texttt{PeerSet}: \texttt{PeerAddress}$

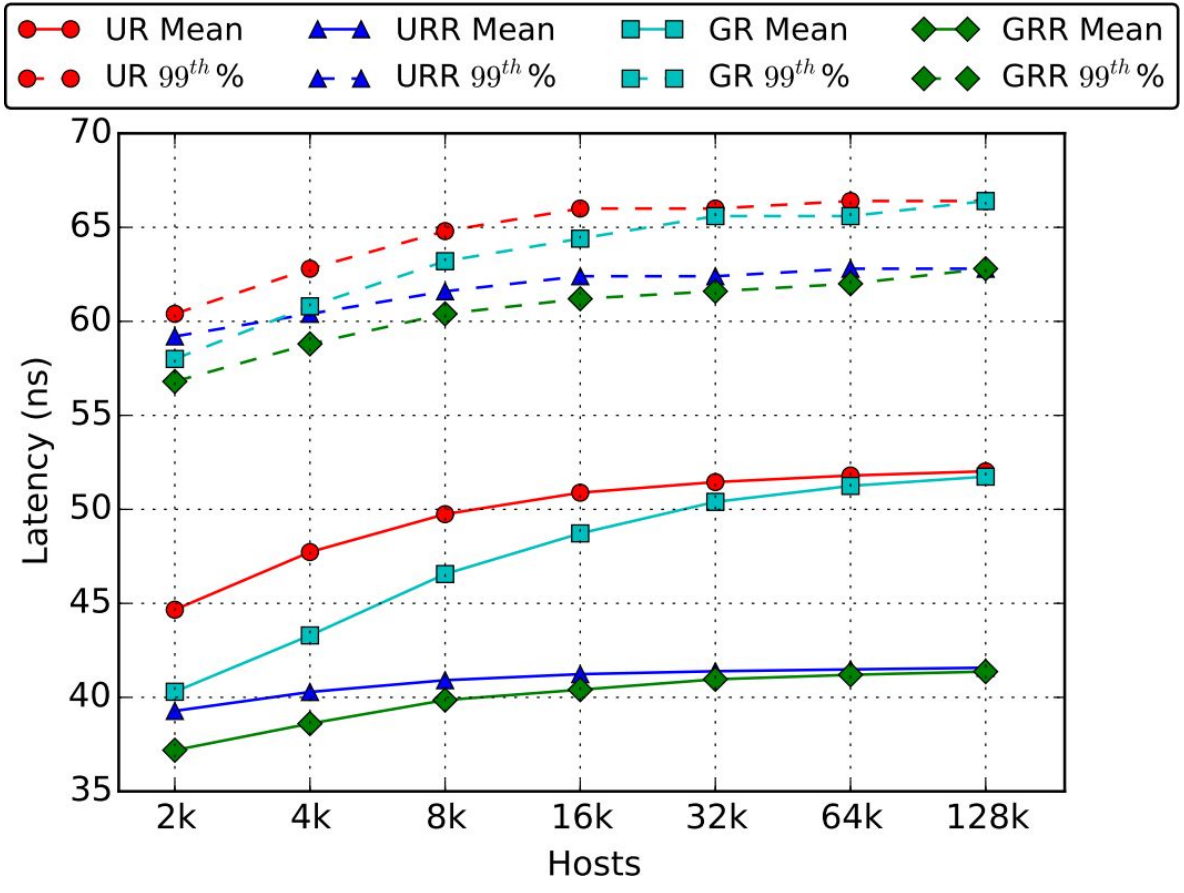
NMU: Latency Results

Memory System:

- L1 cache: 8-way 32 kiB
- L2 cache: 16-way 4 MiB
- DRAM: ~100MiB

32nm process technology

DDR3-1600 technology



NMU: Bandwidth Results

A single NMU logic engine:

	UR	GRR
Permission checks per second	19.23 Mcps	24.39 Mcps
Bandwidth (850 byte packets)*	130.77 Gbps	165.85 Gbps
Bandwidth (200 byte packets)*	30.77 Gbps	39.02 Gbps

8 logic engines @ 90%:

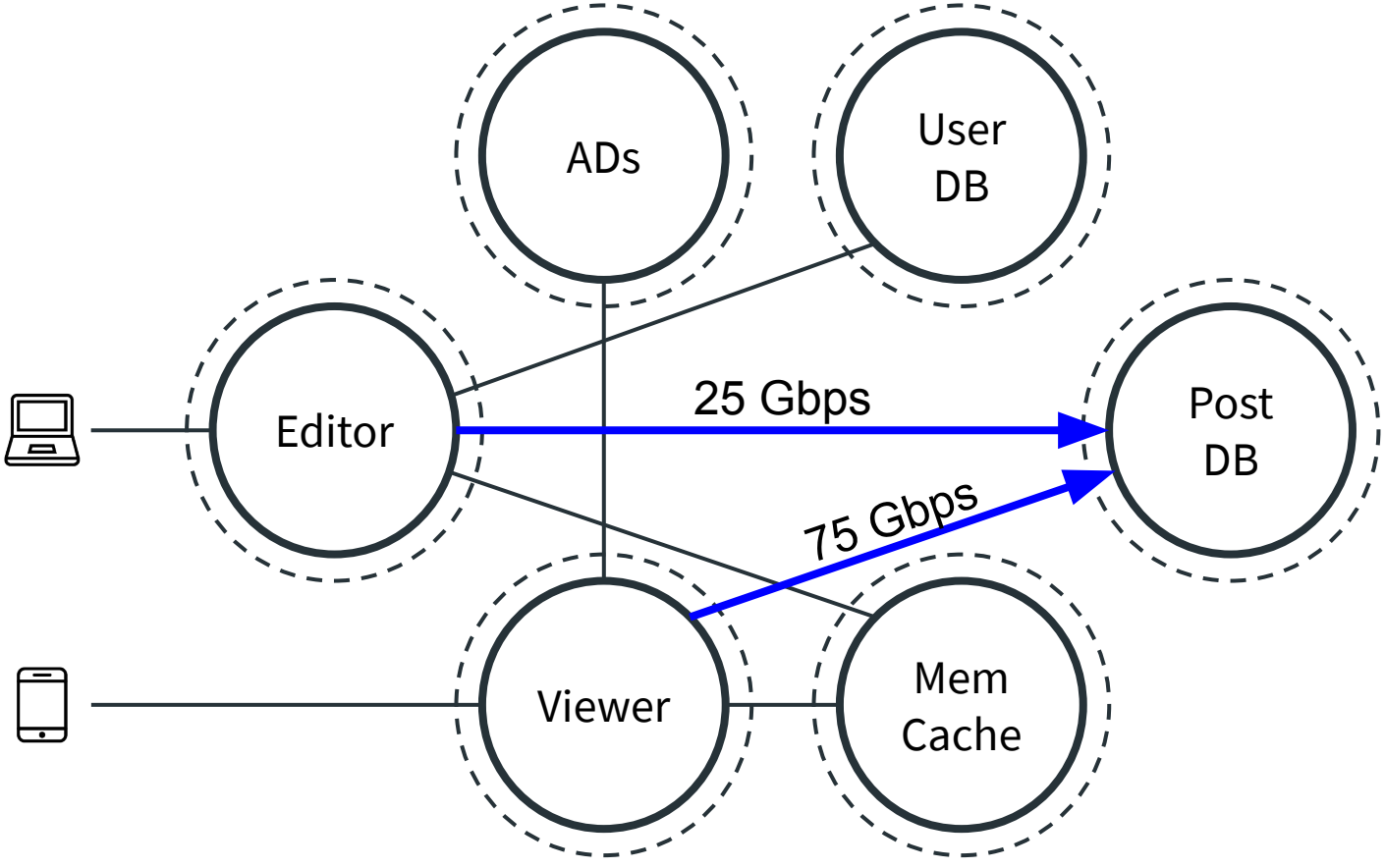
138-176 Mcps (over 1 Tbps on 850 byte packets)

*average packet size in a data center is 850 bytes (Microsoft's claim)

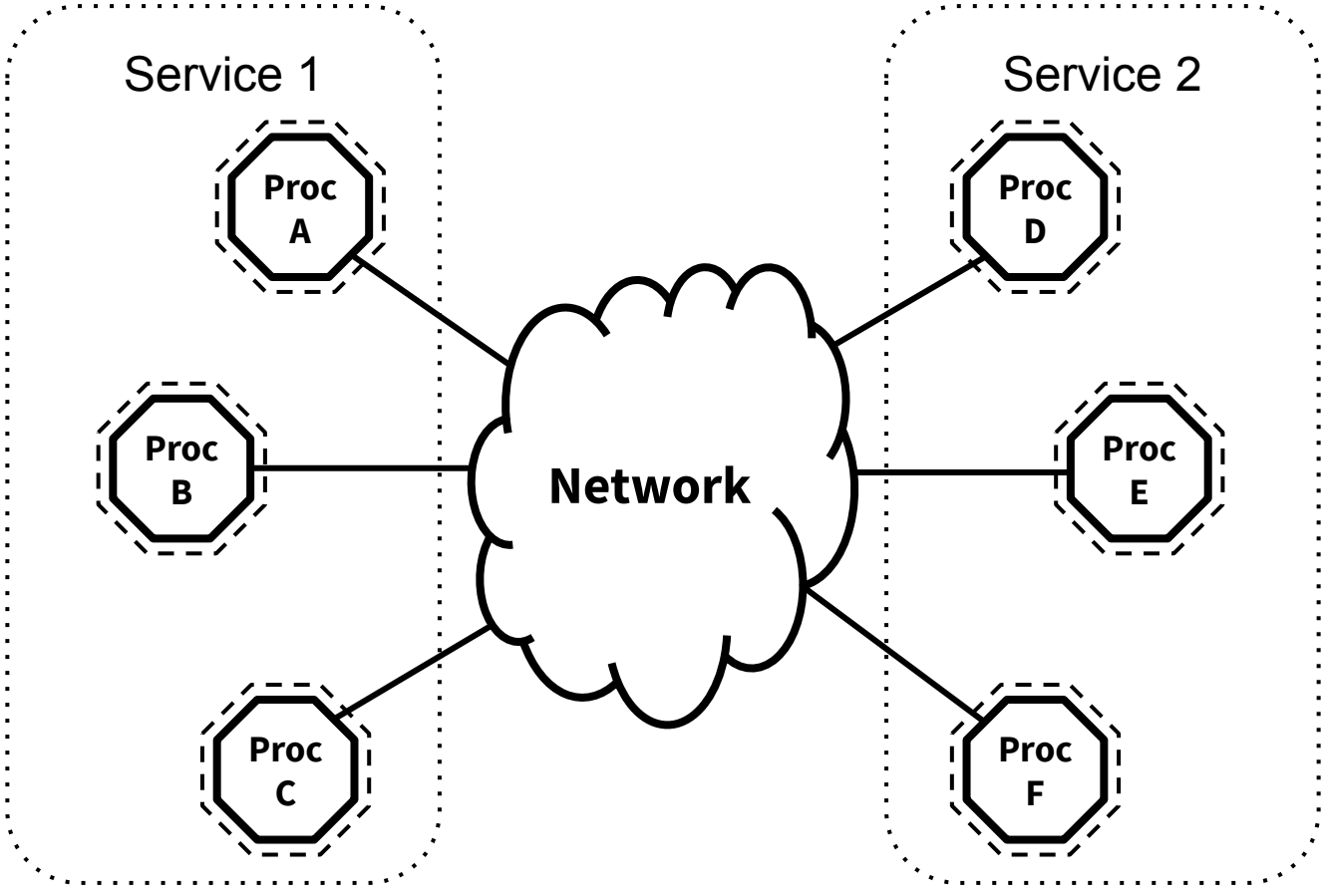
*average packet size in a data center is 200 bytes (Facebook study)

High Performance Rate Control

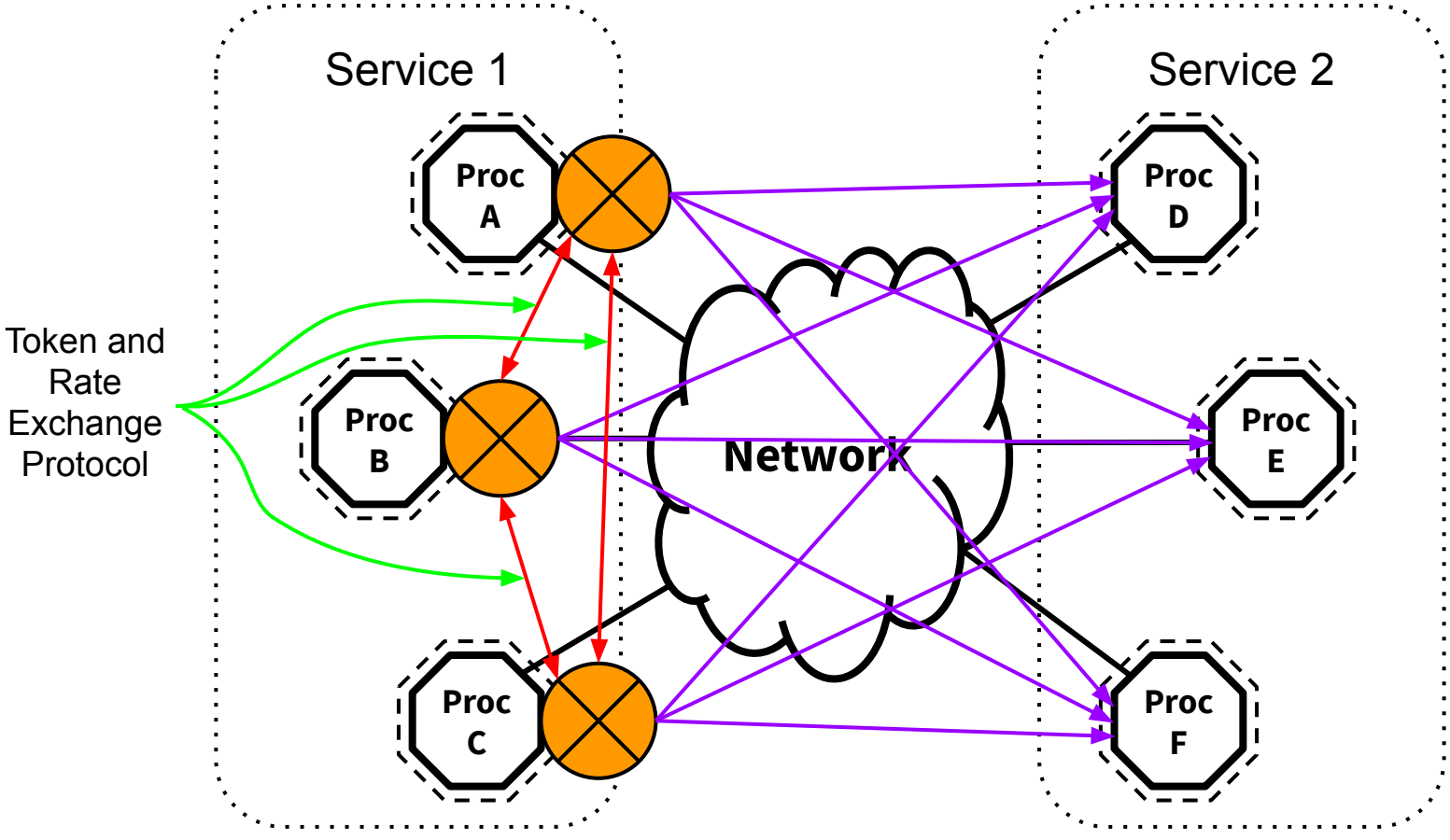
Rate Control



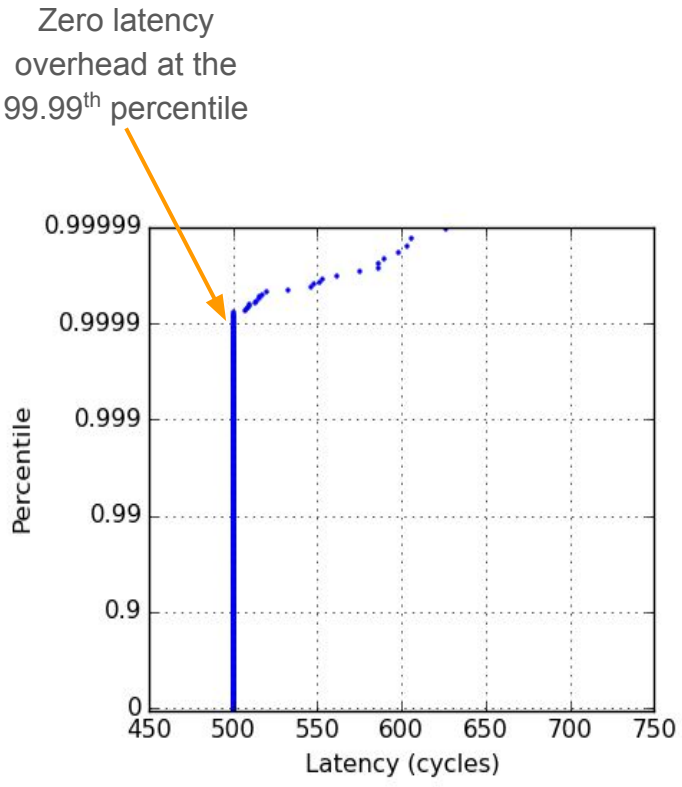
Rate Control: Sender Enforced - Token and Rate Exchange (SE-TRE)



Rate Control: Sender Enforced - Token and Rate Exchange (SE-TRE)

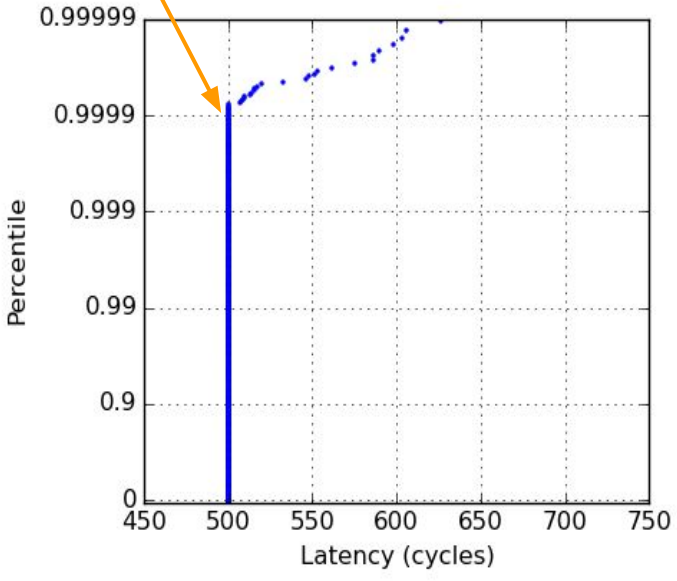


Rate Control: Results

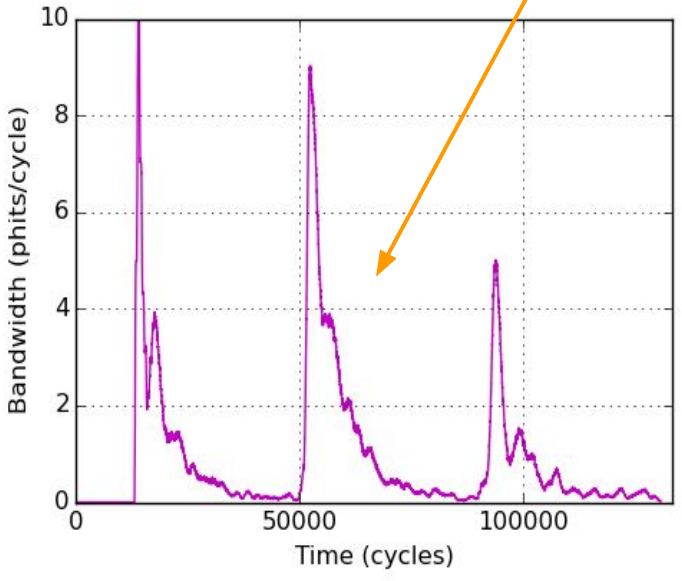


Rate Control: Results

Zero latency overhead at the 99.99th percentile



0.3% bandwidth overhead @ 25 kHz



Related Work

Related Work

Super Computing (e.g., Cray, IBM, Mellanox):

- Partitioning
- Security with no isolation (e.g., Infiniband keys)

Related Work

Super Computing (e.g., Cray, IBM, Mellanox):

- Partitioning
- Security with no isolation (e.g., Infiniband keys)

Cloud Computing (e.g., AWS, GCE, Azure):

- Partitioning (e.g., VLAN, VXLAN, NVGRE)
- Bridging (e.g., OpenStack Neutron, VMware NSX)
- Security with no isolation (e.g., SSL)

Related Work

Super Computing (e.g., Cray, IBM, Mellanox):

- Partitioning
- Security with no isolation (e.g., Infiniband keys)

Cloud Computing (e.g., AWS, GCE, Azure):

- Partitioning (e.g., VLAN, VXLAN, NVGRE)
- Bridging (e.g., OpenStack Neutron, VMware NSX)
- Security with no isolation (e.g., SSL)

Enterprise Computing (e.g., Facebook, State Farm, Chase):

- Security model similar to S.C., technology of C.C.

Related Work

Super Computing (e.g., Cray, IBM, Mellanox):

- Partitioning
- Security with no isolation (e.g., Infiniband keys)

Cloud Computing (e.g., AWS, GCE, Azure):

- Partitioning (e.g., VLAN, VXLAN, NVGRE)
- Bridging (e.g., OpenStack Neutron, VMware NSX)
- Security with no isolation (e.g., SSL)

Enterprise Computing (e.g., Facebook, State Farm, Chase):

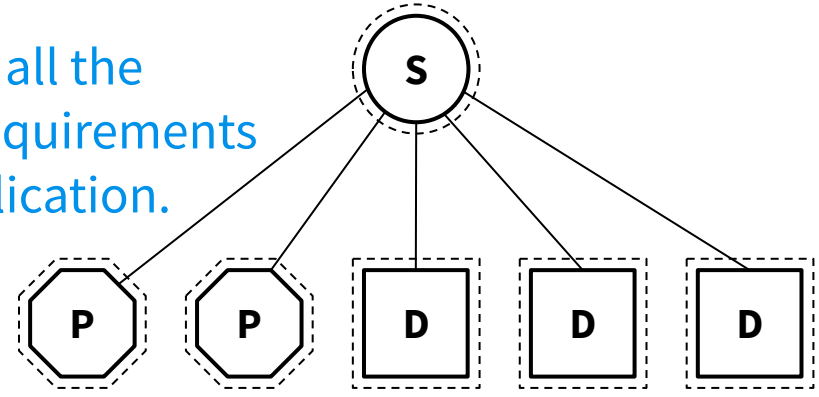
- Security model similar to S.C., technology of C.C.

No system meets the full security and isolation needs of the application

Conclusion

Conclusion

Captures all the interactivity requirements of the application.



System State

146 TB vs. 5.33 GB

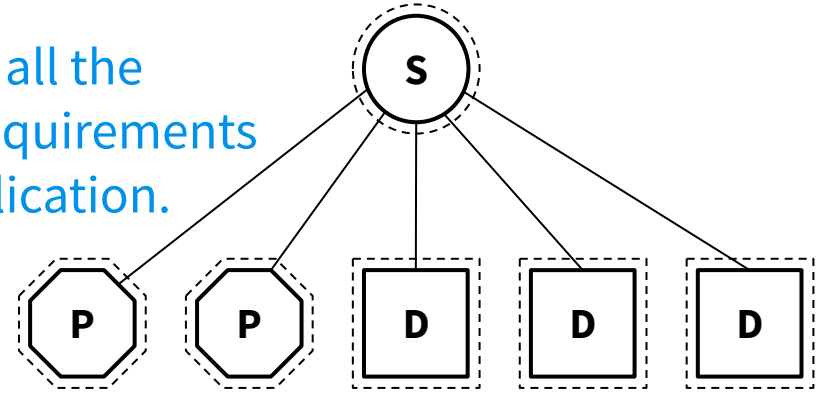
Host State

1.12 GB vs. 20.8 MB

SACLs

Conclusion

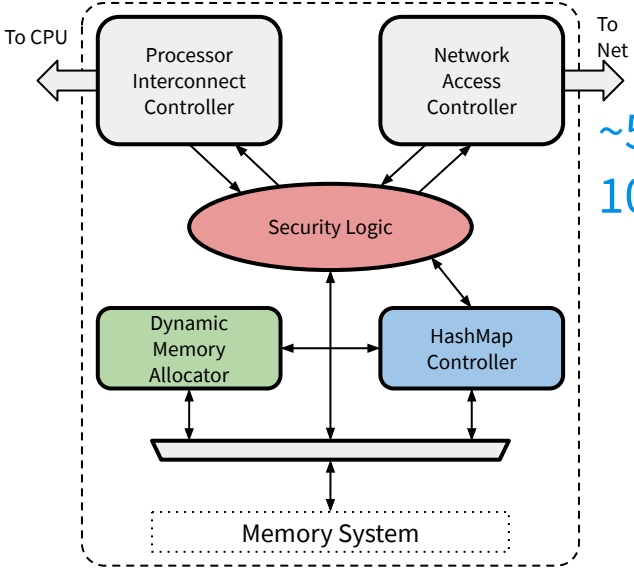
Captures all the interactivity requirements of the application.



System State
146 TB vs. 5.33 GB

Host State
1.12 GB vs. 20.8 MB

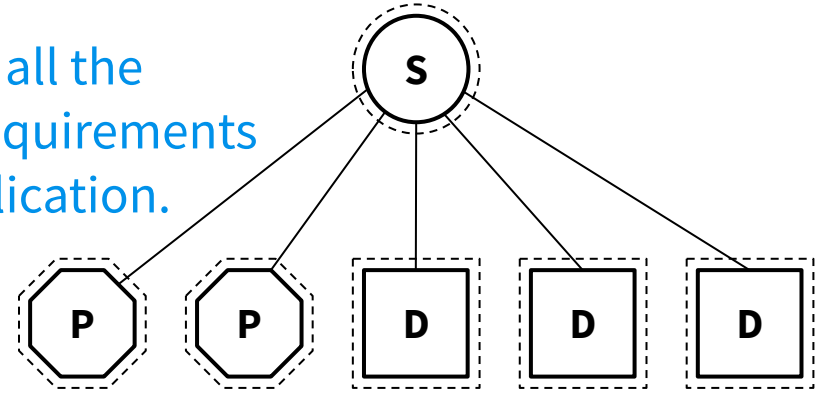
SACLs



~50ns
100s Gbps

Conclusion

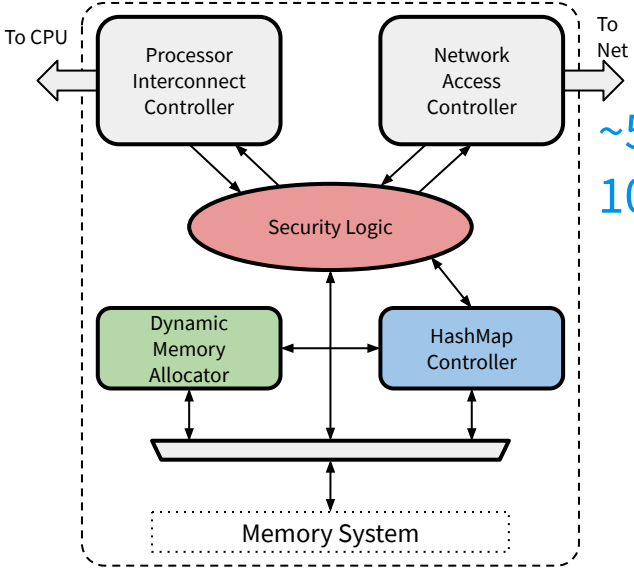
Captures all the interactivity requirements of the application.



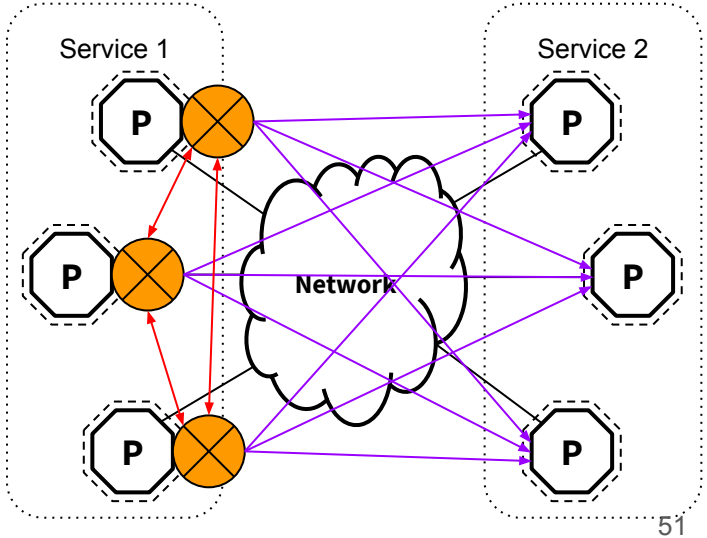
System State
146 TB vs. 5.33 GB

Host State
1.12 GB vs. 20.8 MB

SACLs



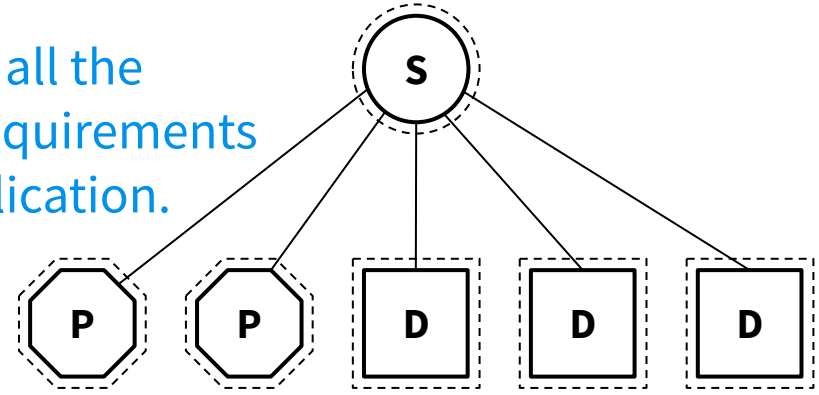
~50ns
100s Gbps



Zero @ 99.99th
0.3% bandwidth

Conclusion

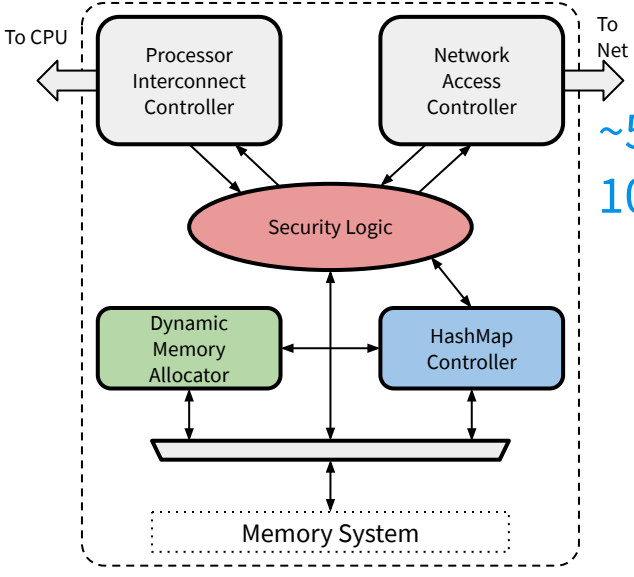
Captures all the interactivity requirements of the application.



System State
146 TB vs. 5.33 GB

Host State
1.12 GB vs. 20.8 MB

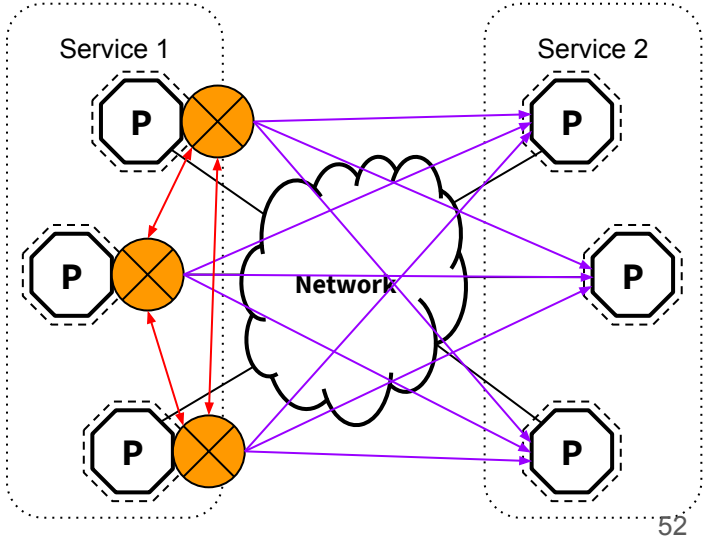
SACLs



~50ns
100s Gbps

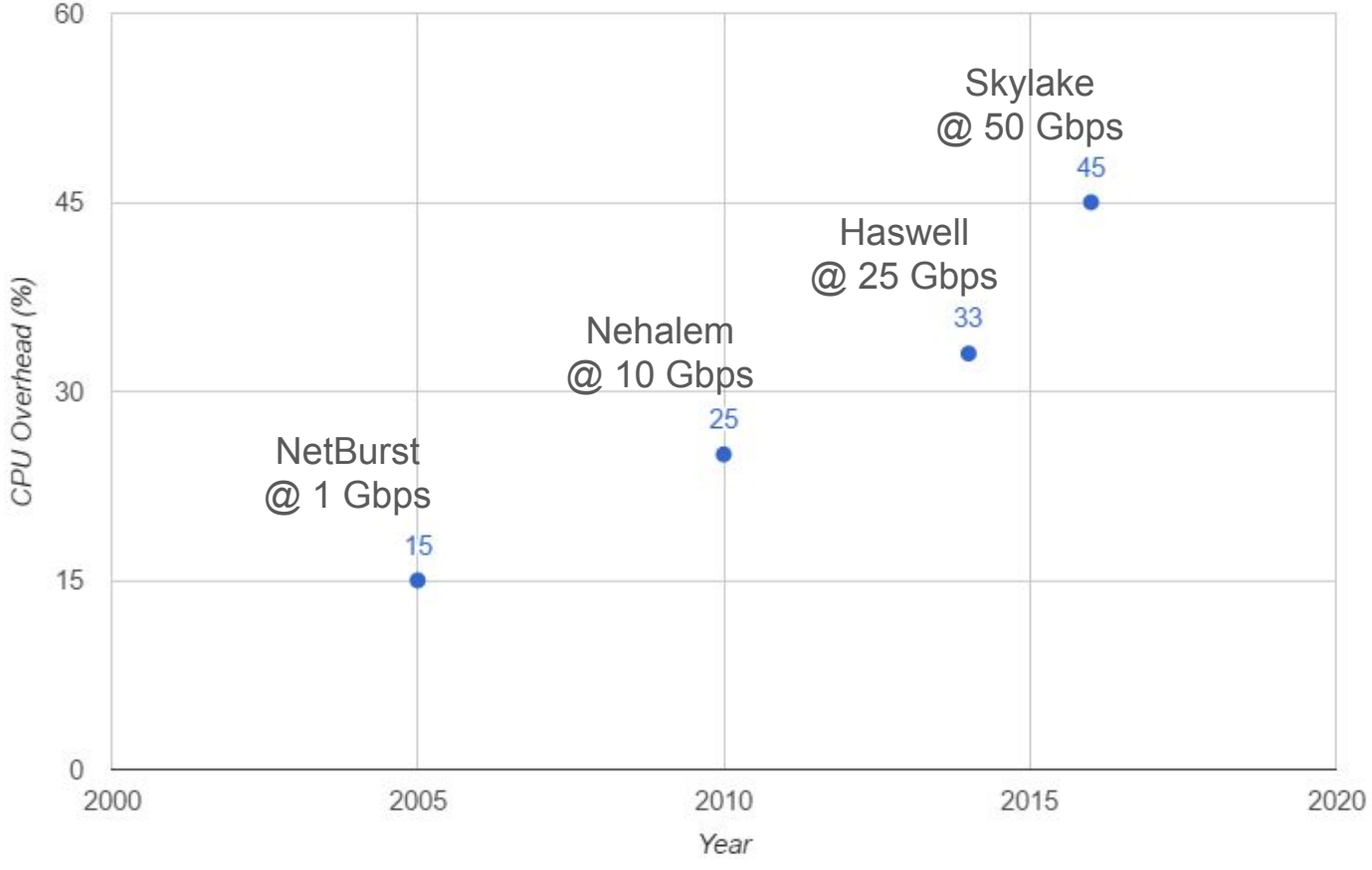
Zero CPU Overhead

Zero @ 99.99th
0.3% bandwidth



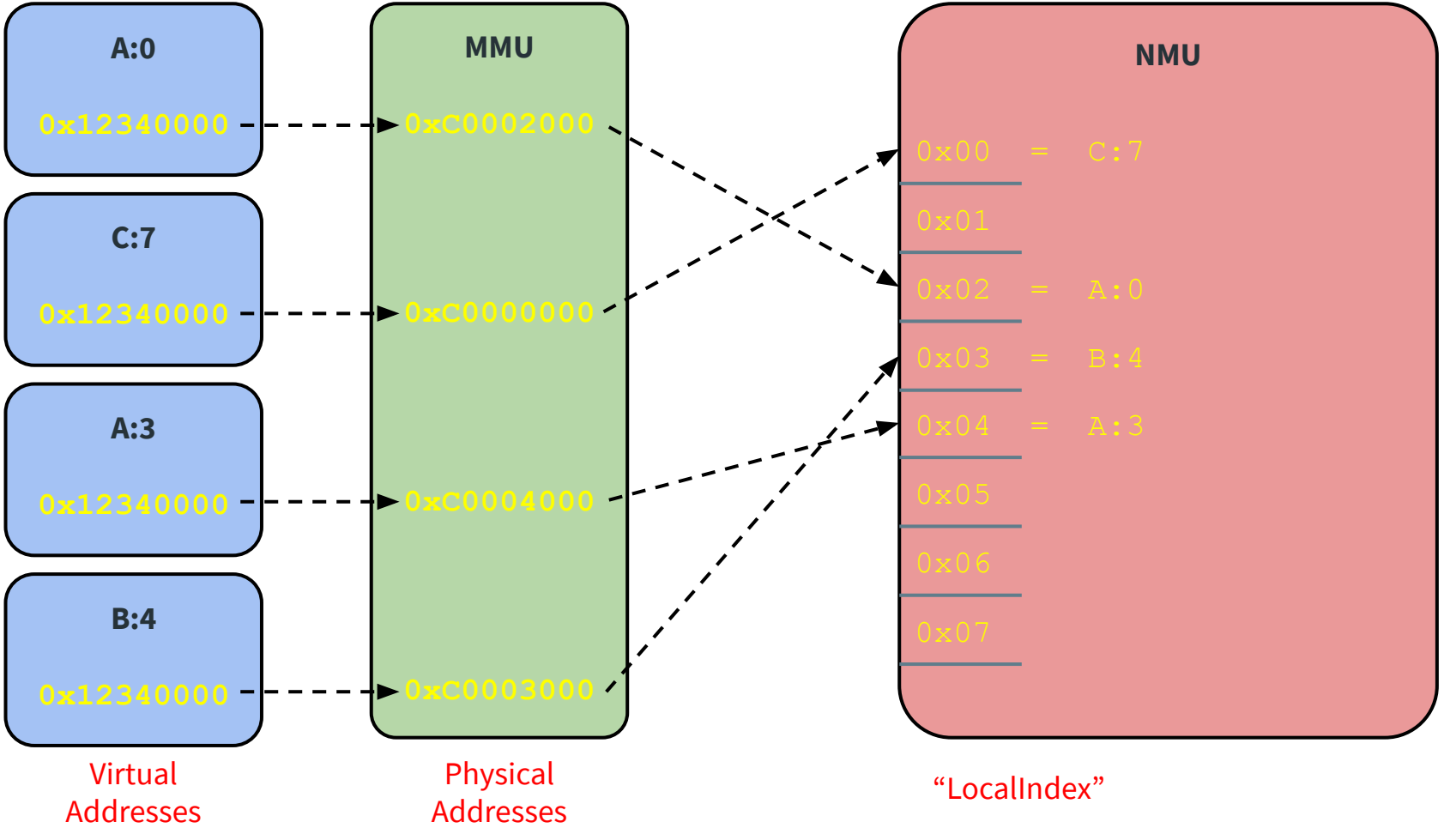
Thank You!

CPU Overheads in Cloud Computing



*study by Broadcom

NMU: Architecture



NMU: Security Analysis

The NMU completely implements the Sikker security and isolation policy. In the presence of an exploited host OS, the NMU provides increased security compared to modern systems.