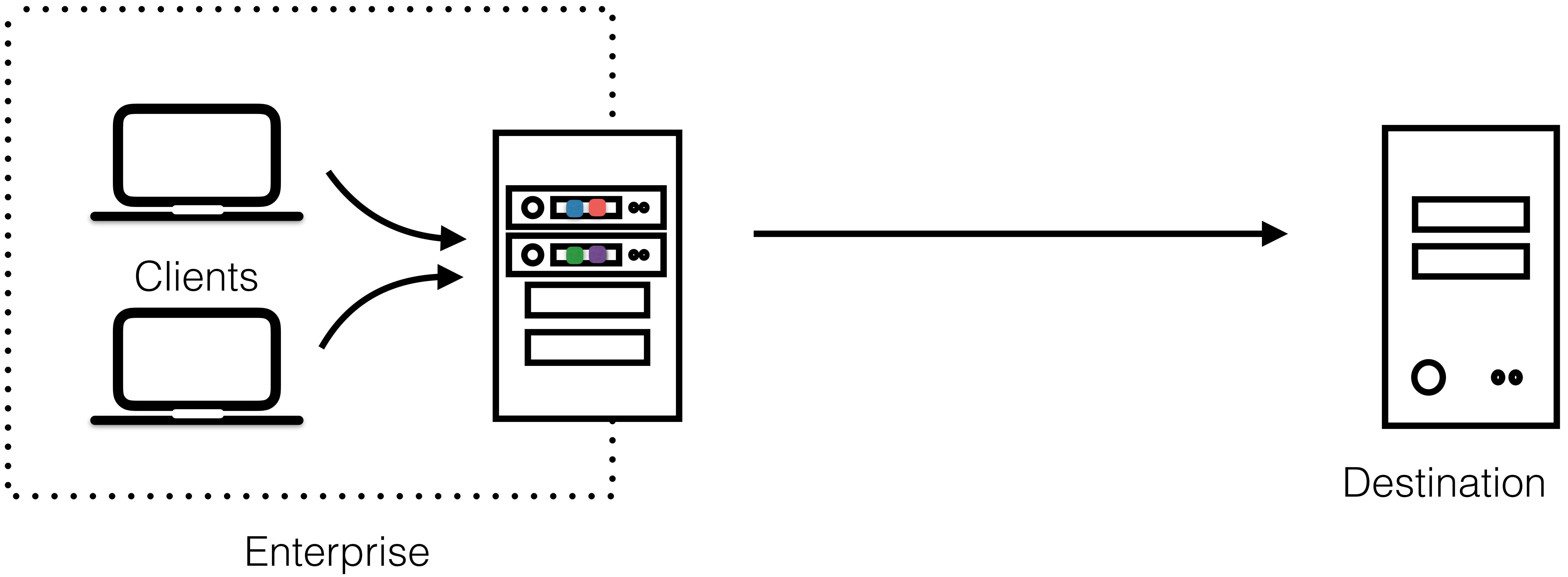
SafeBricks: Shielding Network Functions in the Cloud

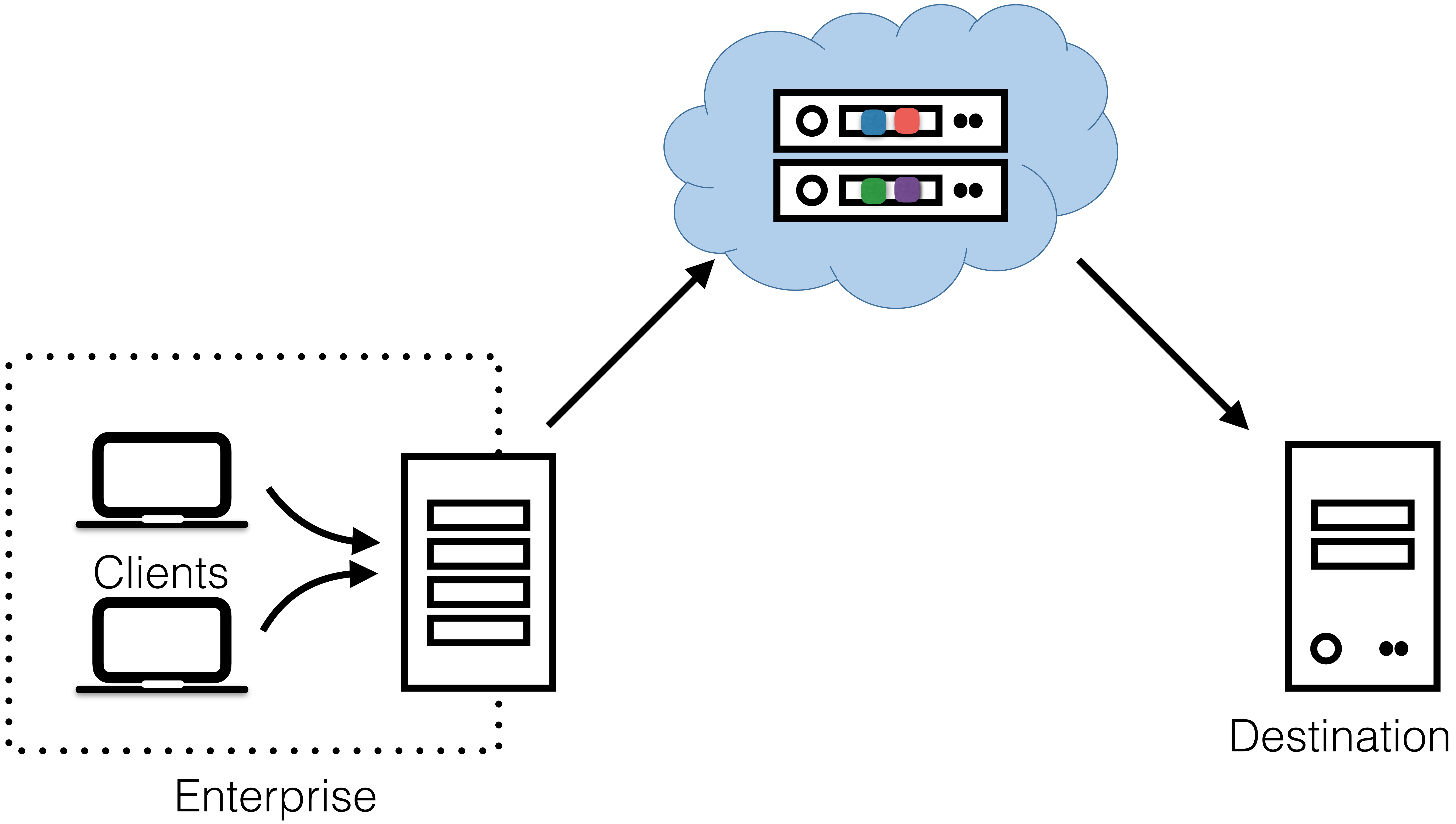
Rishabh Poddar, Chang Lan,
Raluca Ada Popa, Sylvia Ratnasamy

UC Berkeley

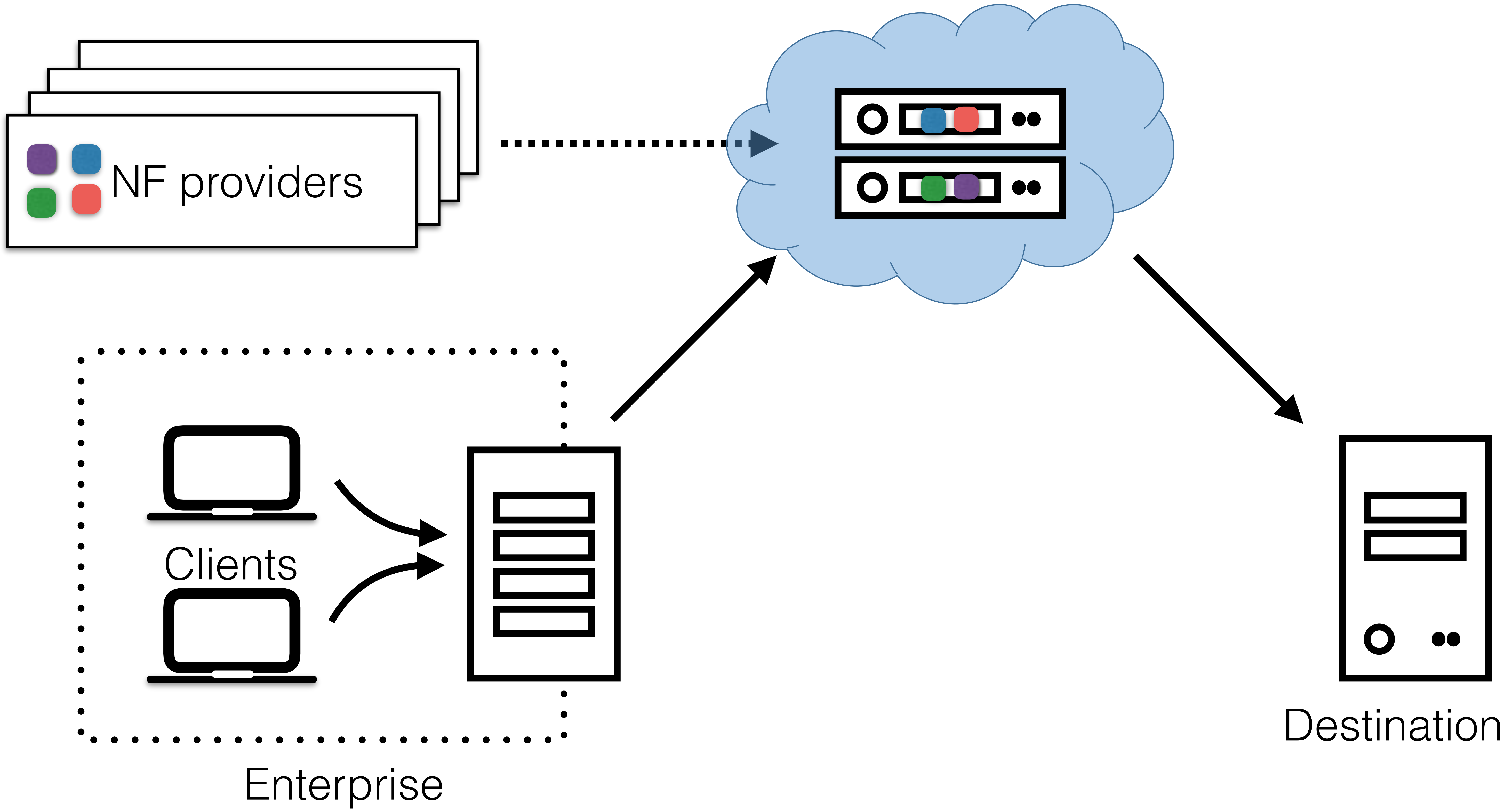
Network Functions (NFs) in the cloud



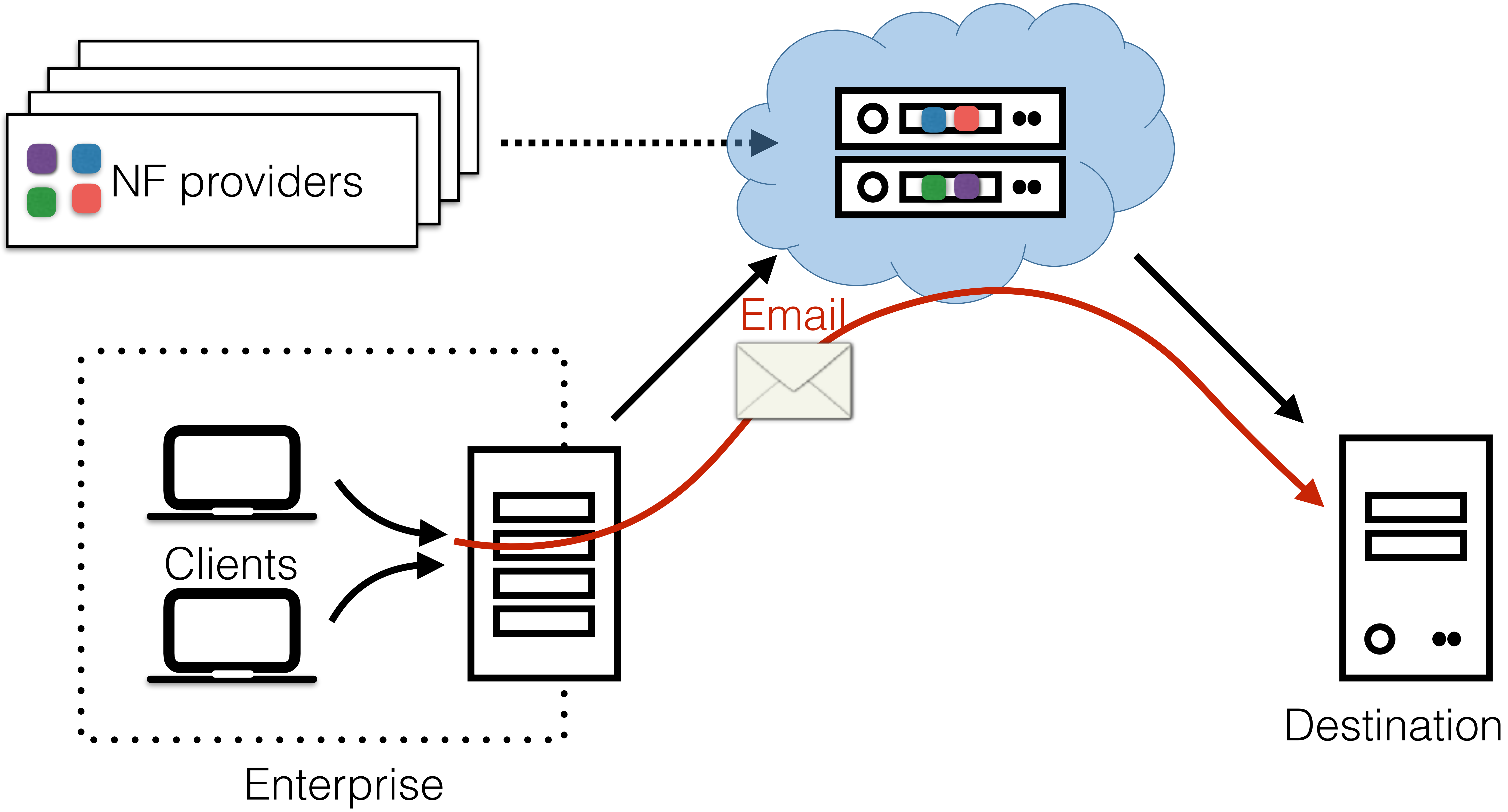
Network Functions (NFs) in the cloud



Network Functions (NFs) in the cloud

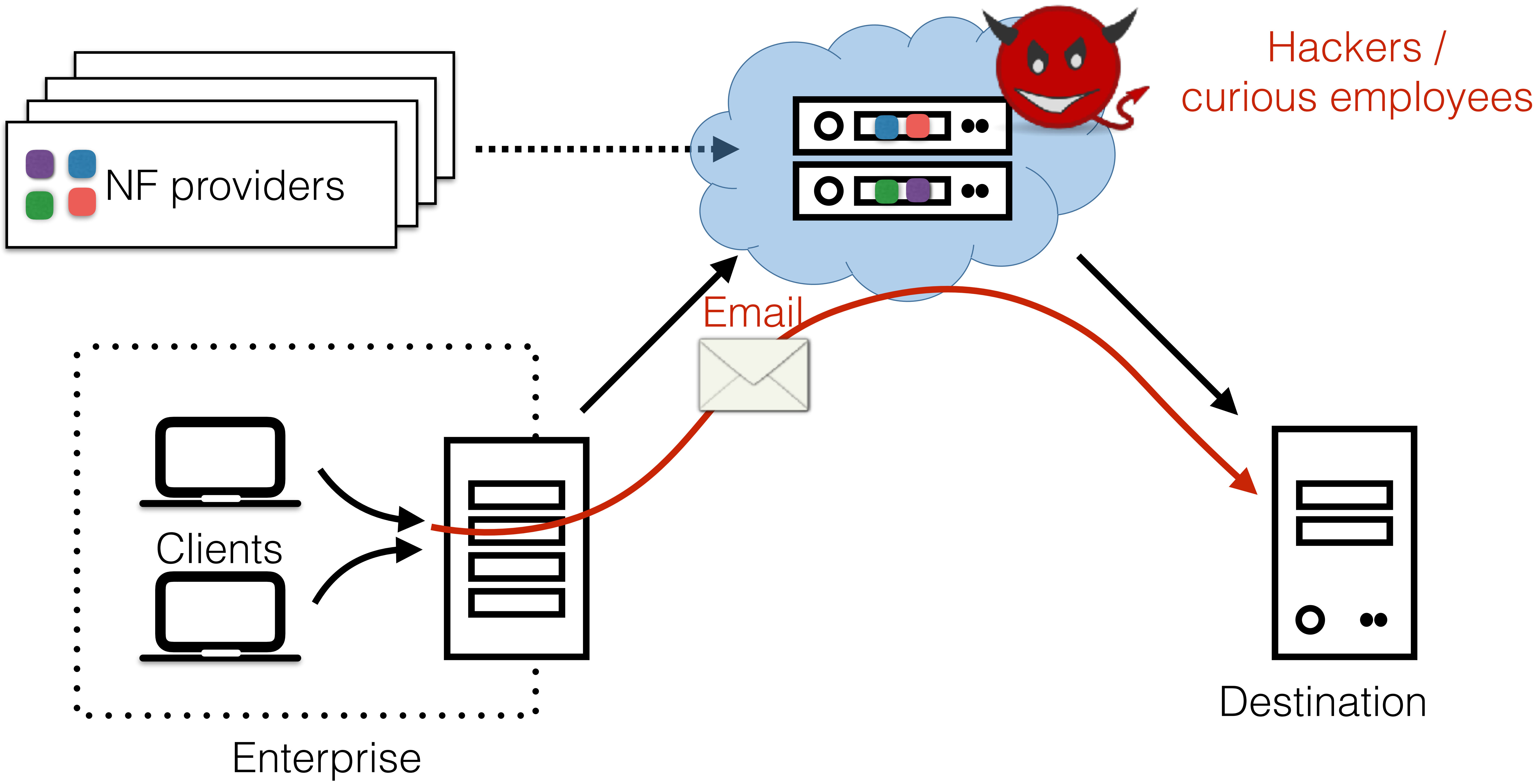


Problem: Security



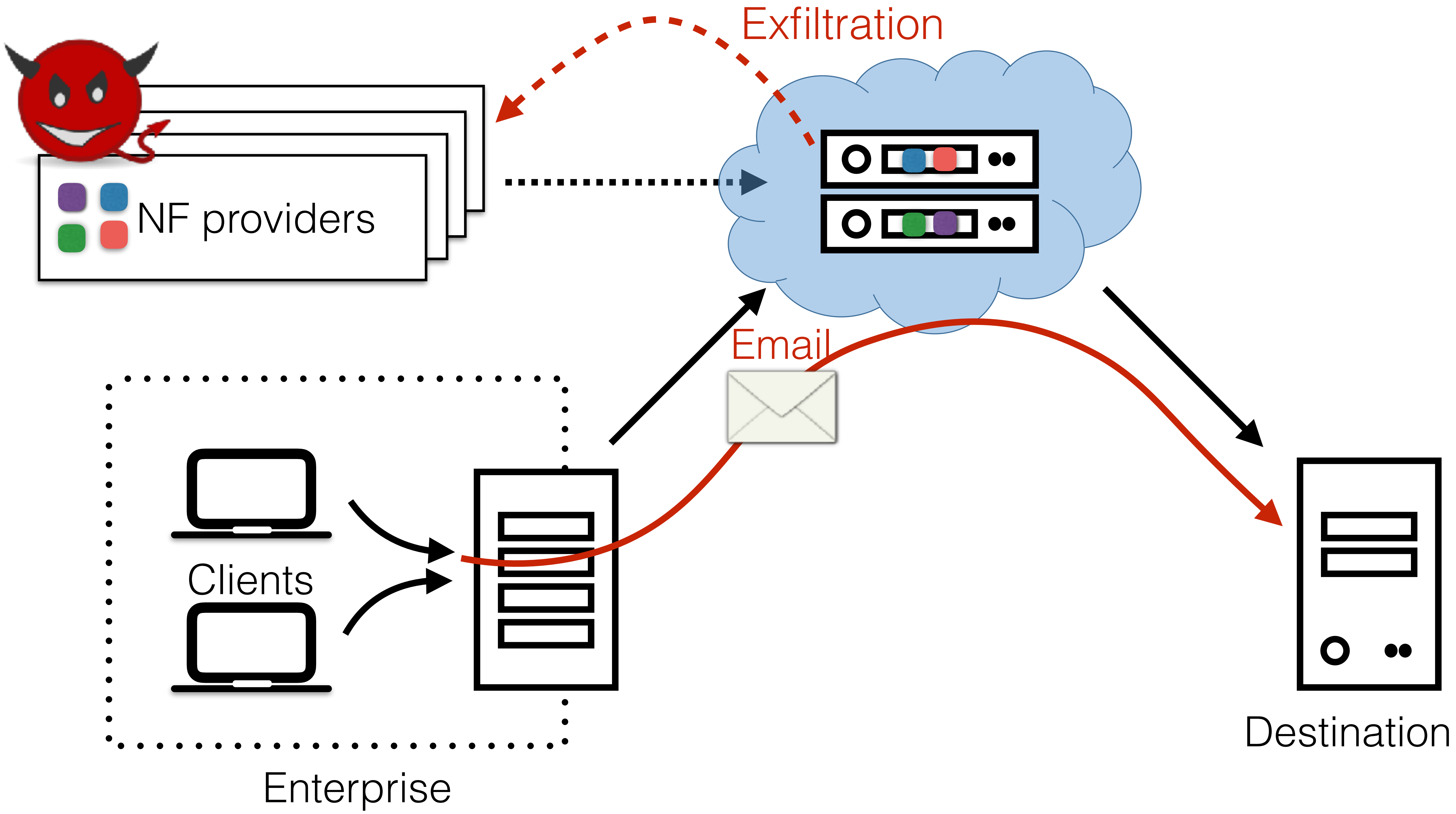
Problem: Security

1 Need to protect **traffic** from the **cloud provider**



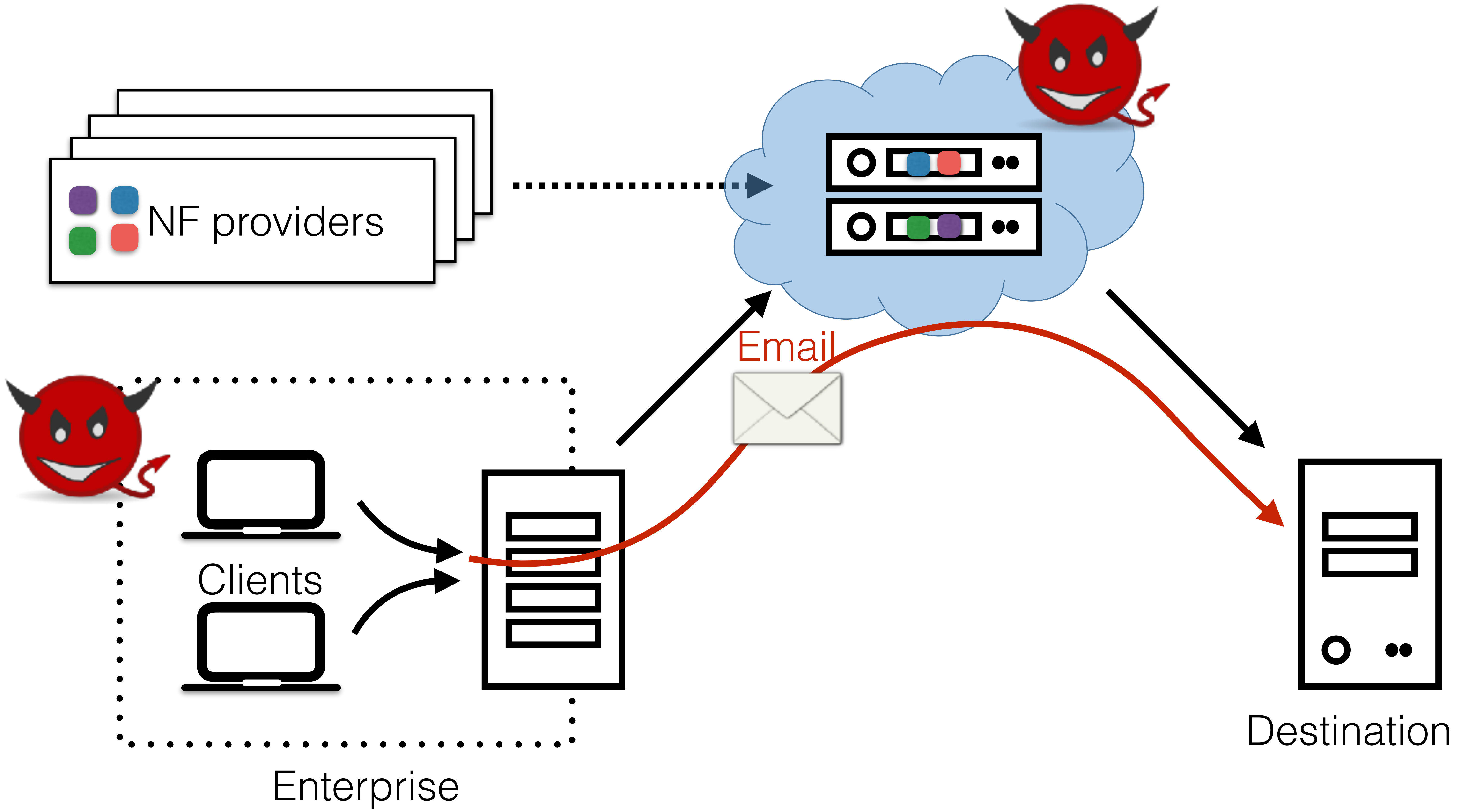
Problem: Security

2 Need to protect **traffic** from the **NF providers**

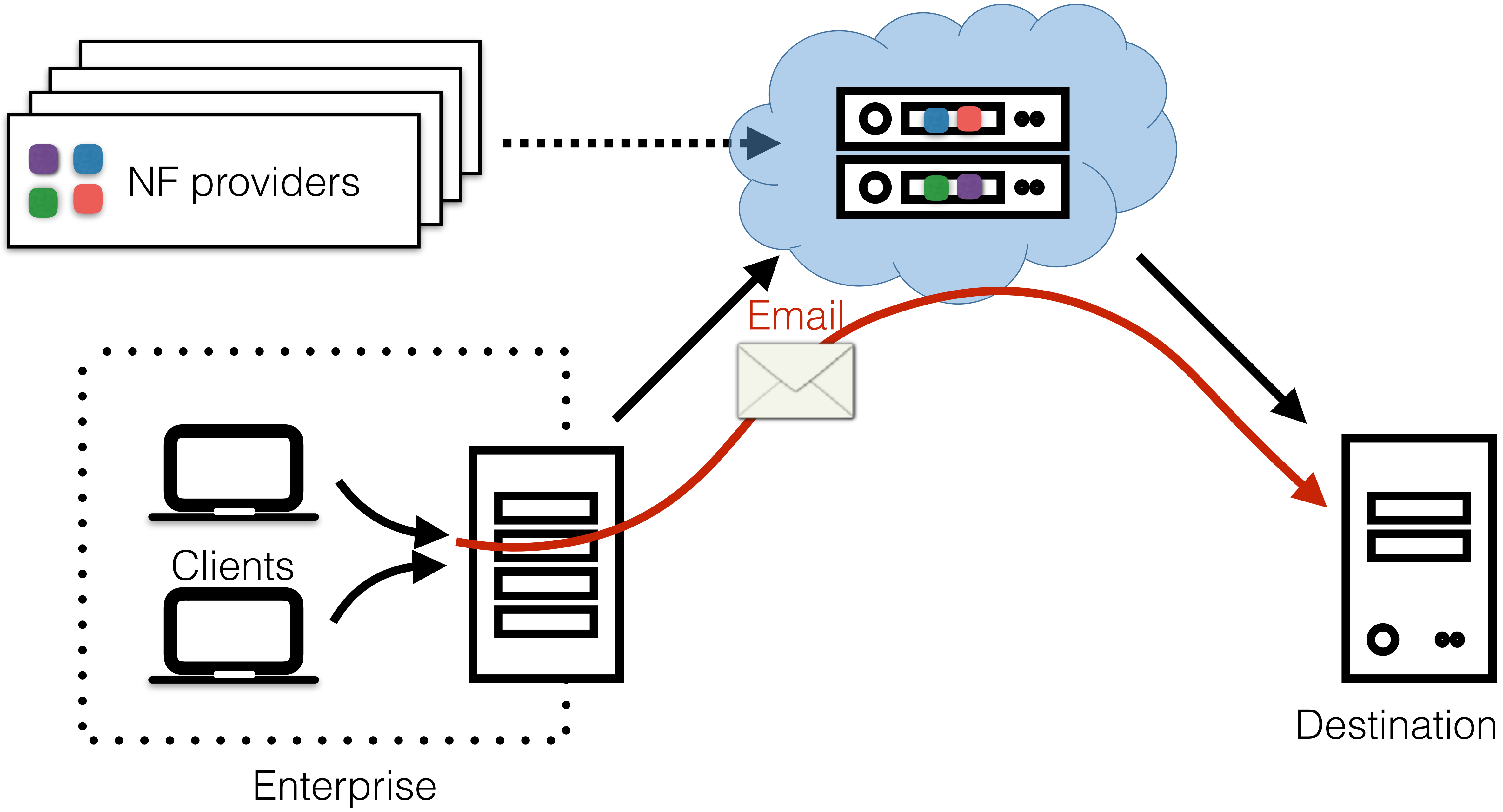


Problem: Security

3 Need to protect **NF code and rulesets** from client enterprise and cloud

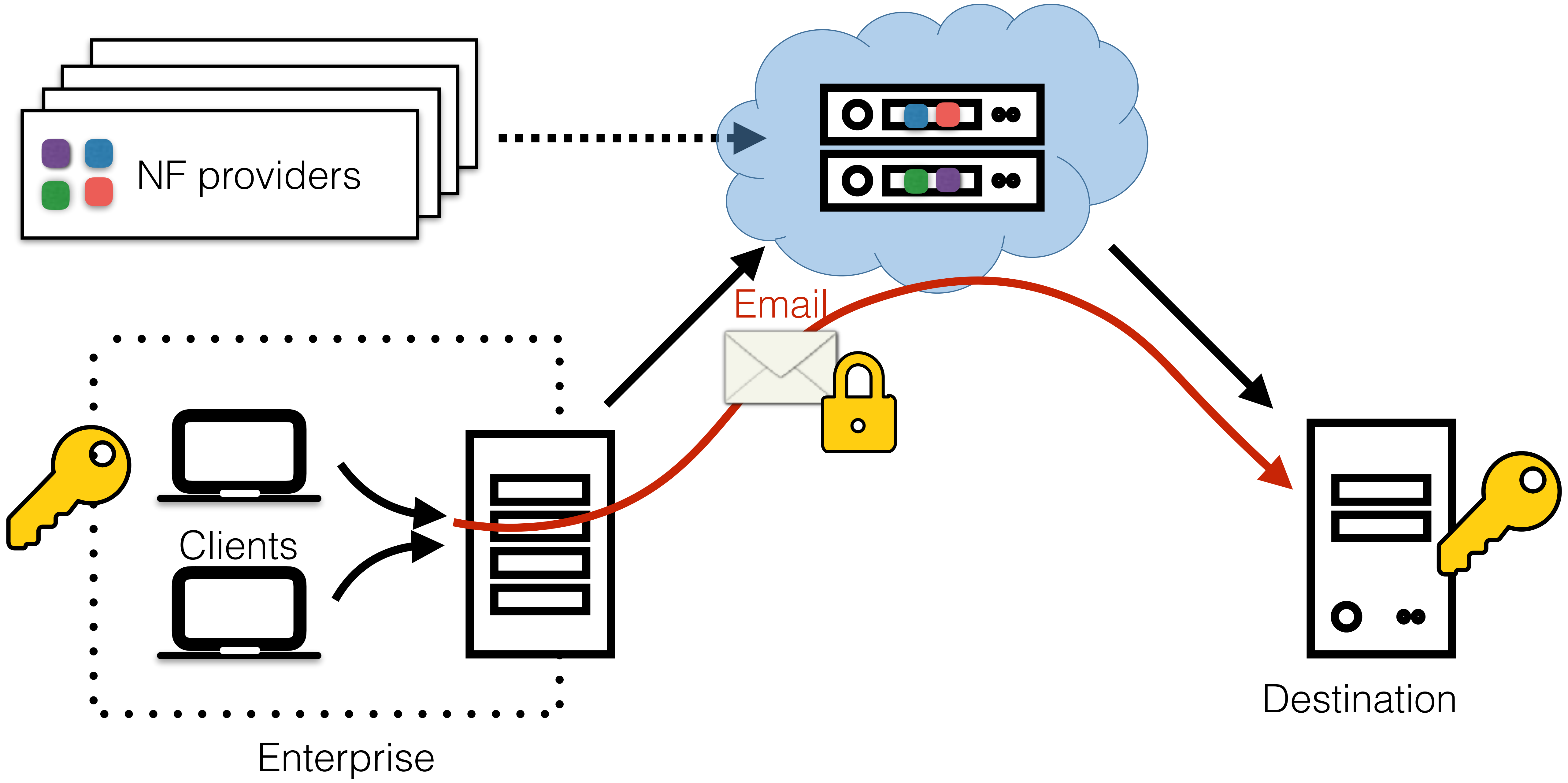


Cryptographic solutions do not suffice



Cryptographic solutions do not suffice

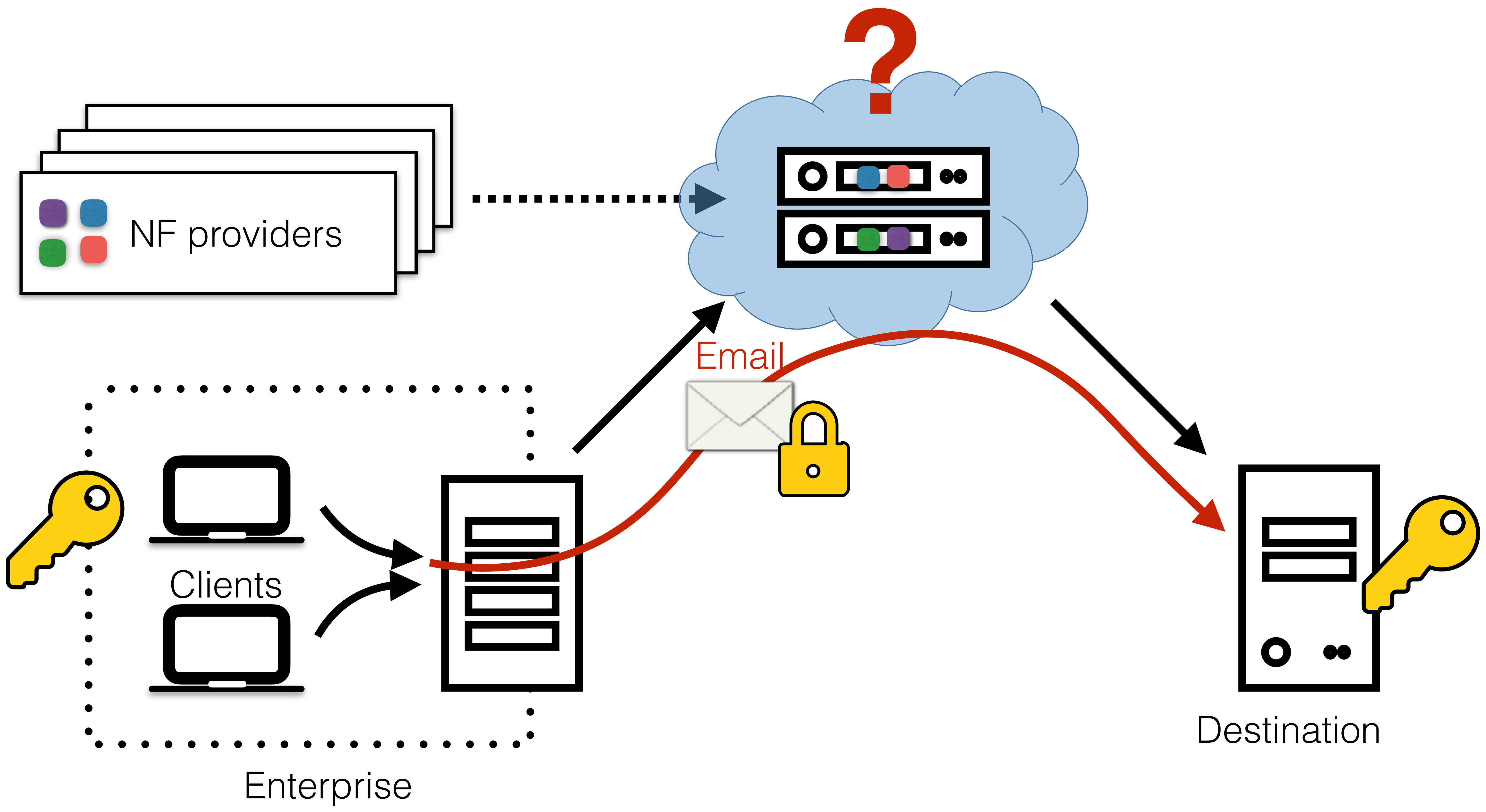
1 **Standard encryption**: e.g. end-to-end TLS



Cryptographic solutions do not suffice

1 **Standard encryption**: e.g. end-to-end TLS

✗ Functionality: Doesn't allow any computation on encrypted payload

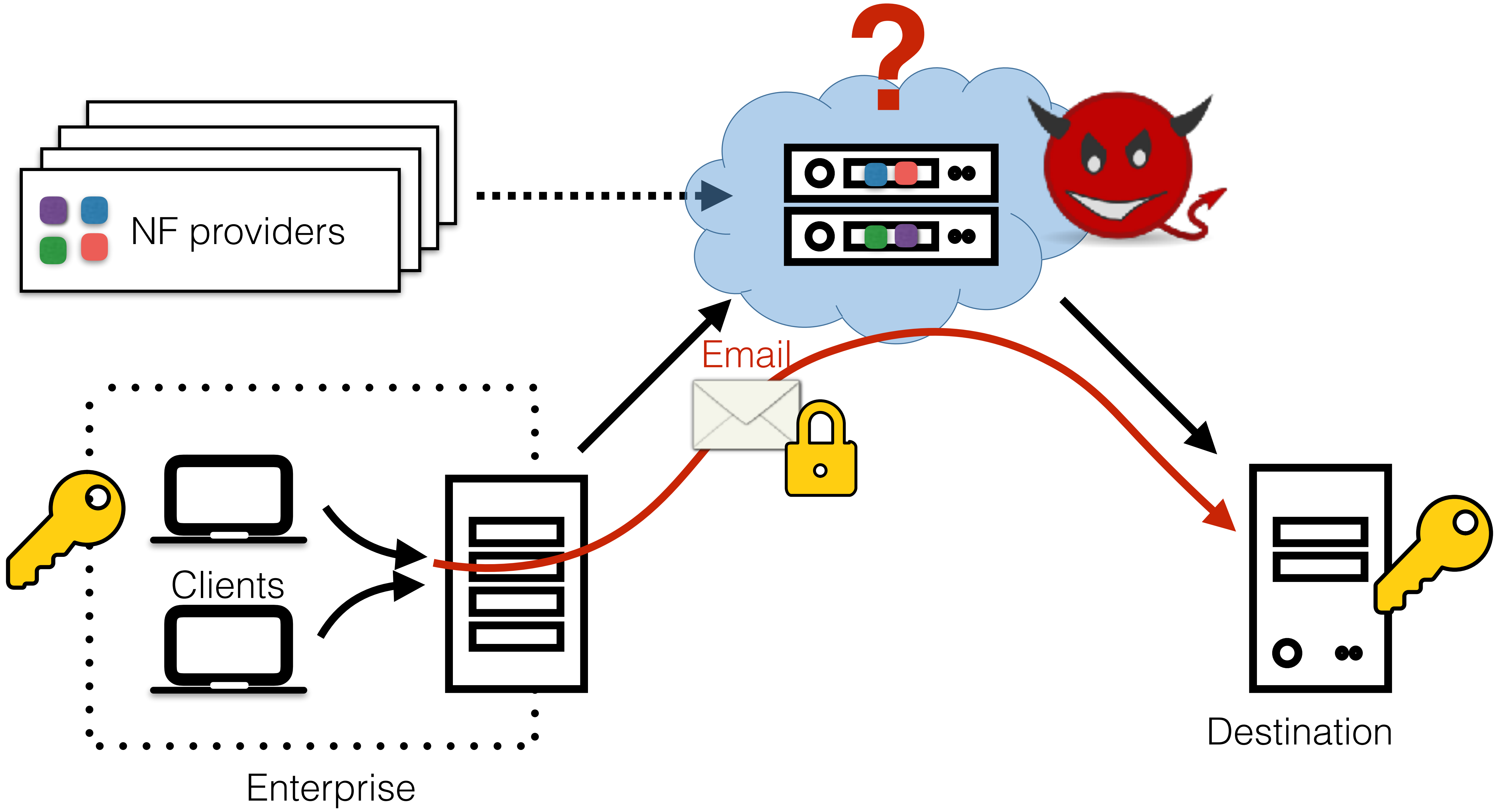


Cryptographic solutions do not suffice

1 **Standard encryption**: e.g. end-to-end TLS

✗ Functionality: Doesn't allow any computation on encrypted payload

✗ Security: Unencrypted fields (e.g. IP headers) still leak information



Cryptographic solutions do not suffice

2 **Specialized encryption**: e.g. BlindBox, Embark

[Sherry et al. (SIGCOMM’15)] [Lan et al. (NSDI’16)]

Cryptographic solutions do not suffice

2 **Specialized encryption**: e.g. BlindBox, Embark

✗ Too limited in functionality!

✓ Header-based comparisons

✓ Keyword matching

✗ Regular expressions

✗ Cross-flow analysis

✗ Statistical computations

How to achieve **full functionality** and

our security goals simultaneously?

SafeBricks

1. Protects **traffic** from the **cloud provider**
2. Protects **traffic** from the **NF providers**
3. Protects **NF source code and rulesets** from client enterprise and cloud

SafeBricks

Hardware enclaves + language-based isolation

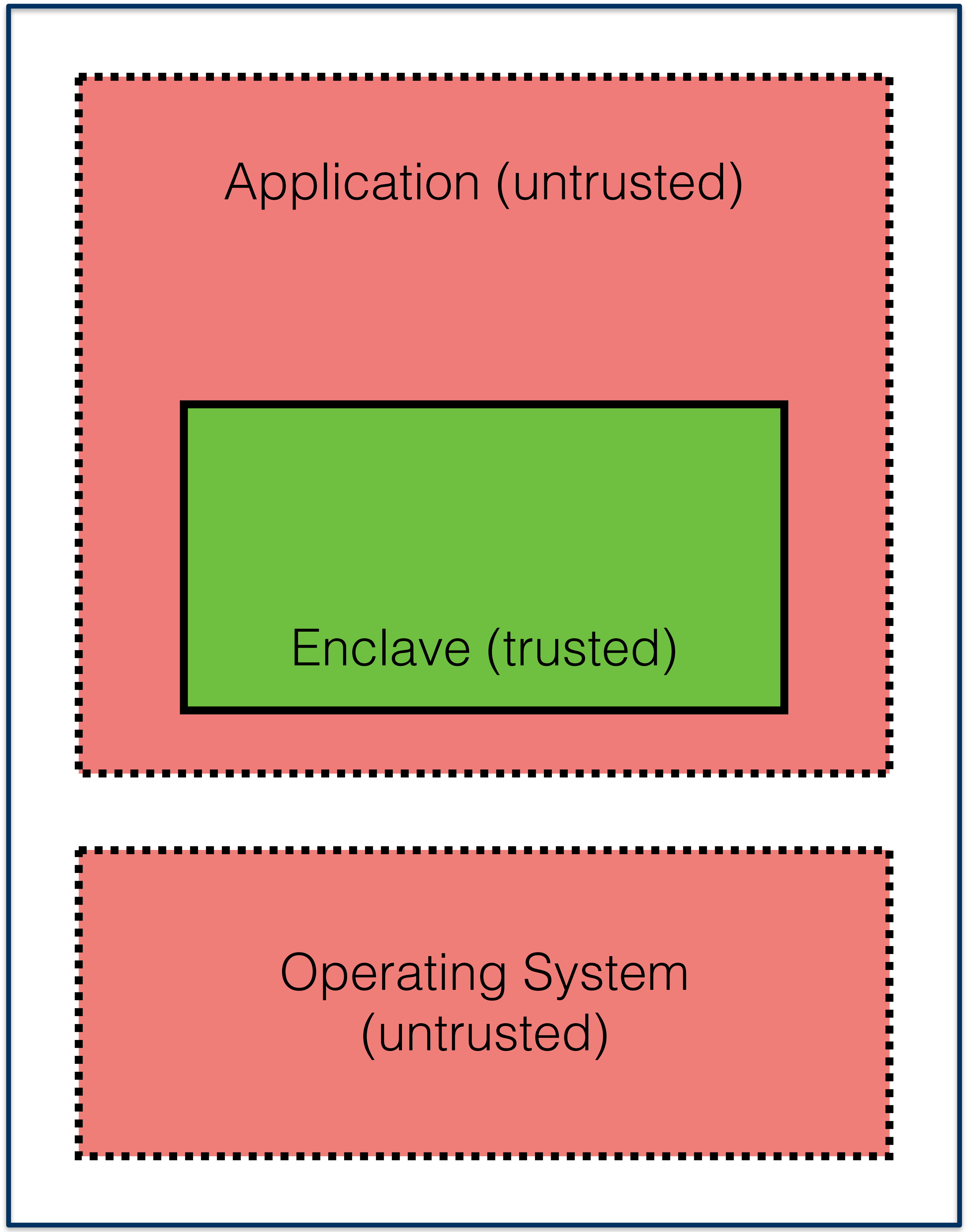
1. Protects **traffic** from the **cloud provider**

2. Protects **traffic** from the **NF providers**

3. Protects **NF source code and rulesets** from client enterprise and cloud

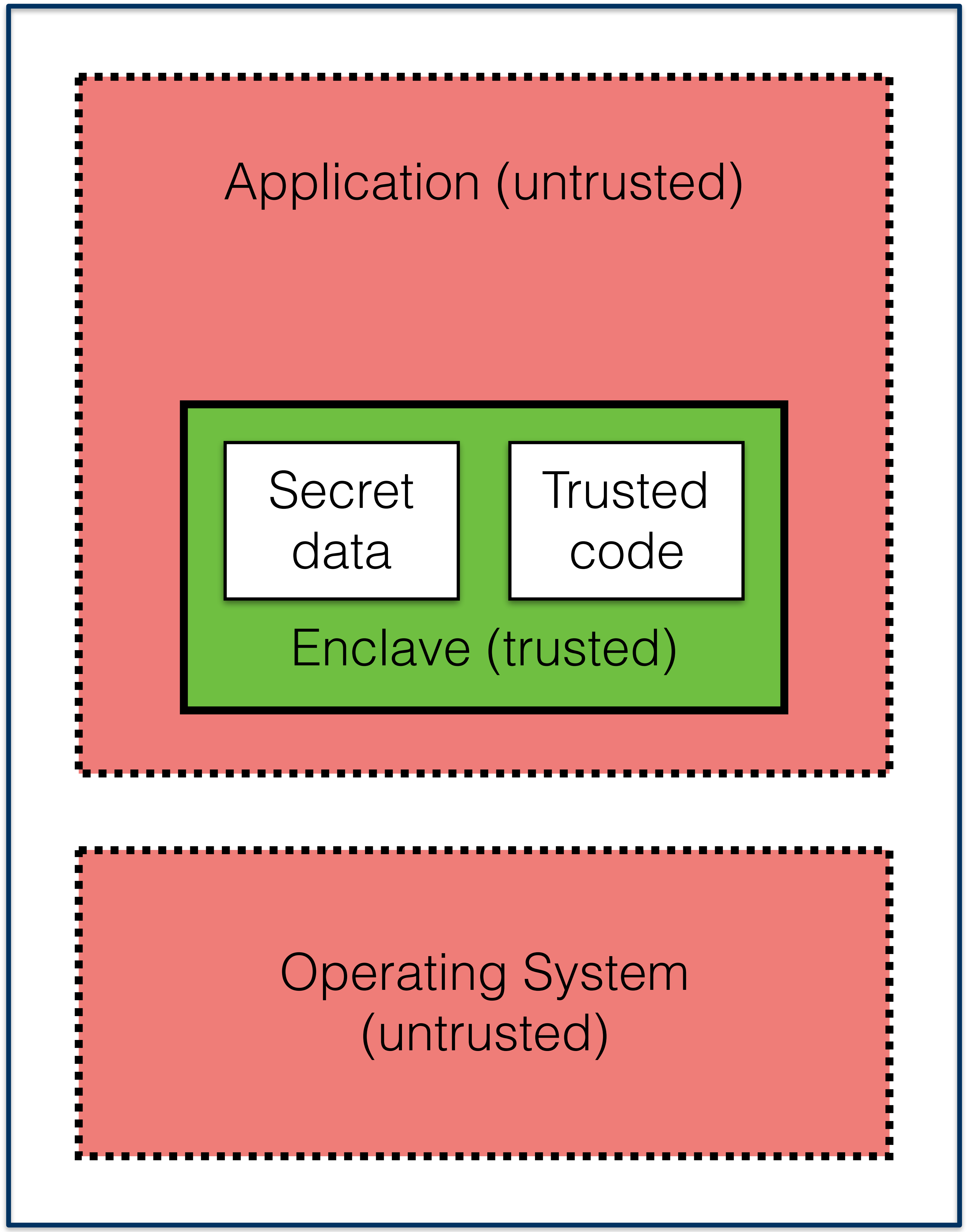
Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (**enclaves**) protected by hardware



Background: Hardware enclaves (e.g. Intel SGX)

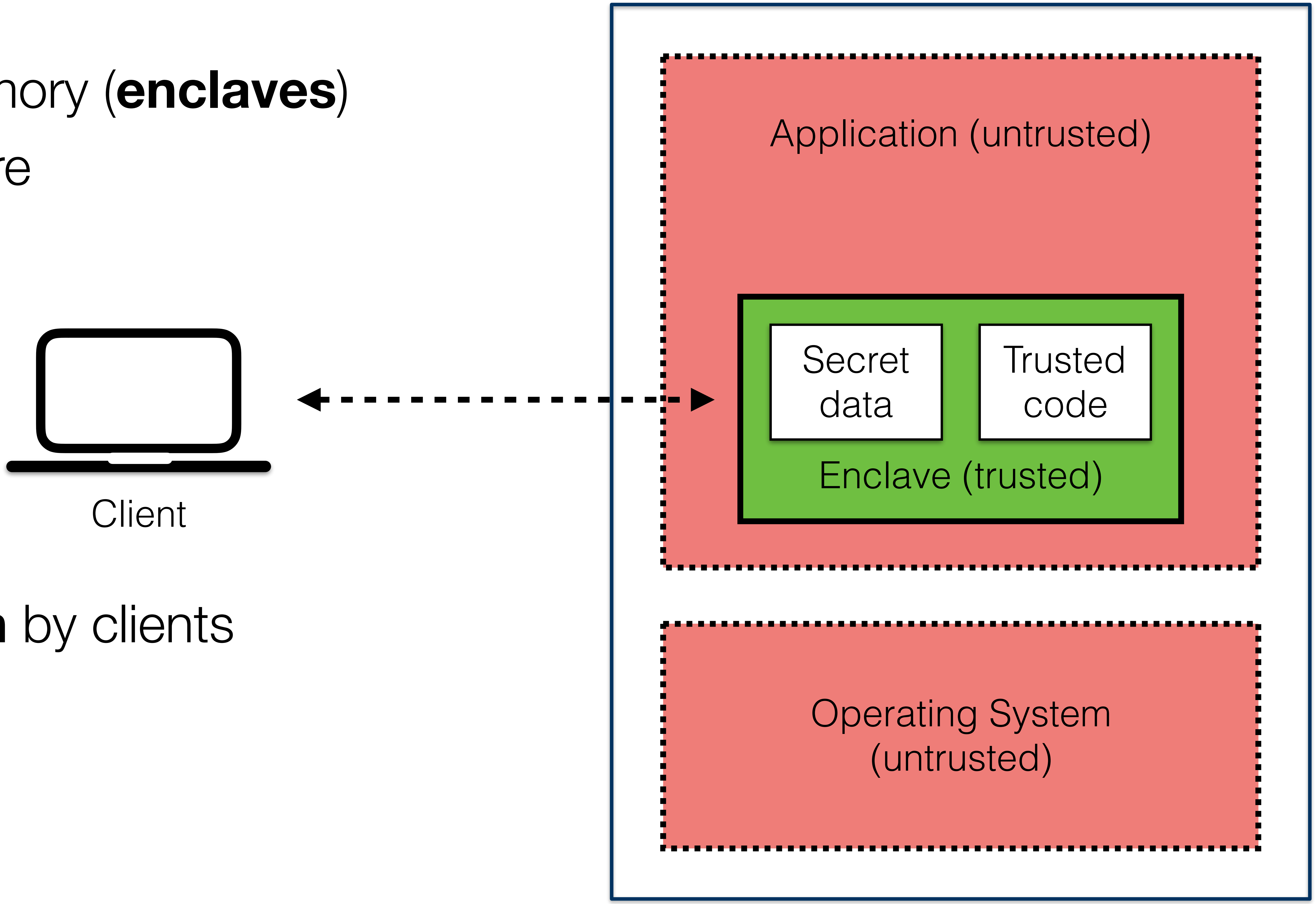
- Secure region of memory (**enclaves**) protected by hardware



Background: Hardware enclaves (e.g. Intel SGX)

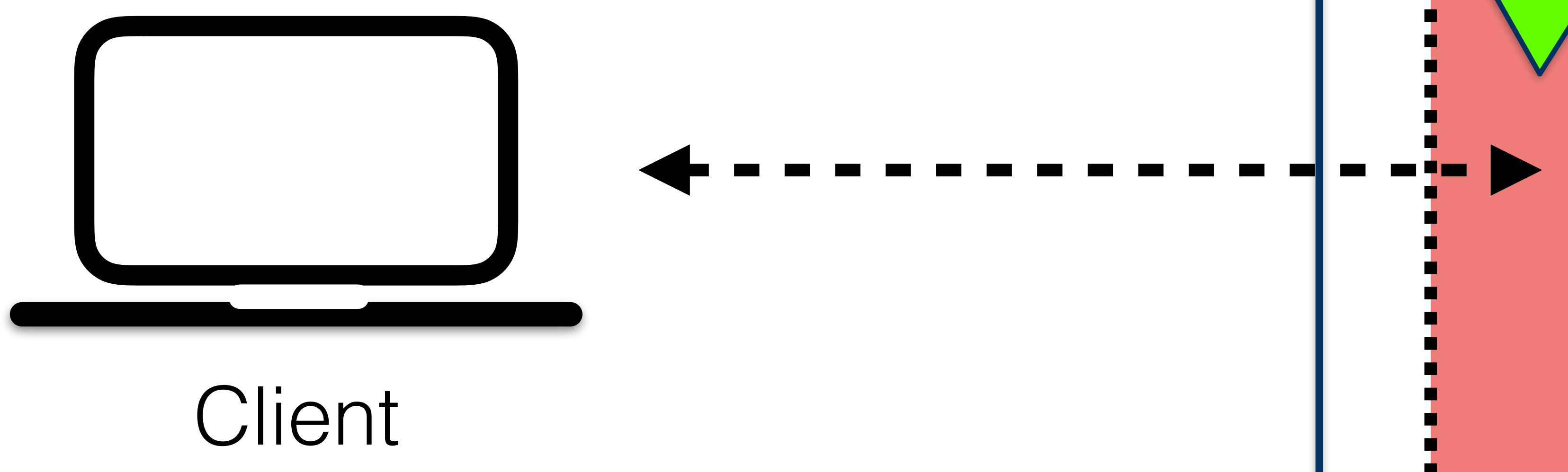
- Secure region of memory (**enclaves**) protected by hardware

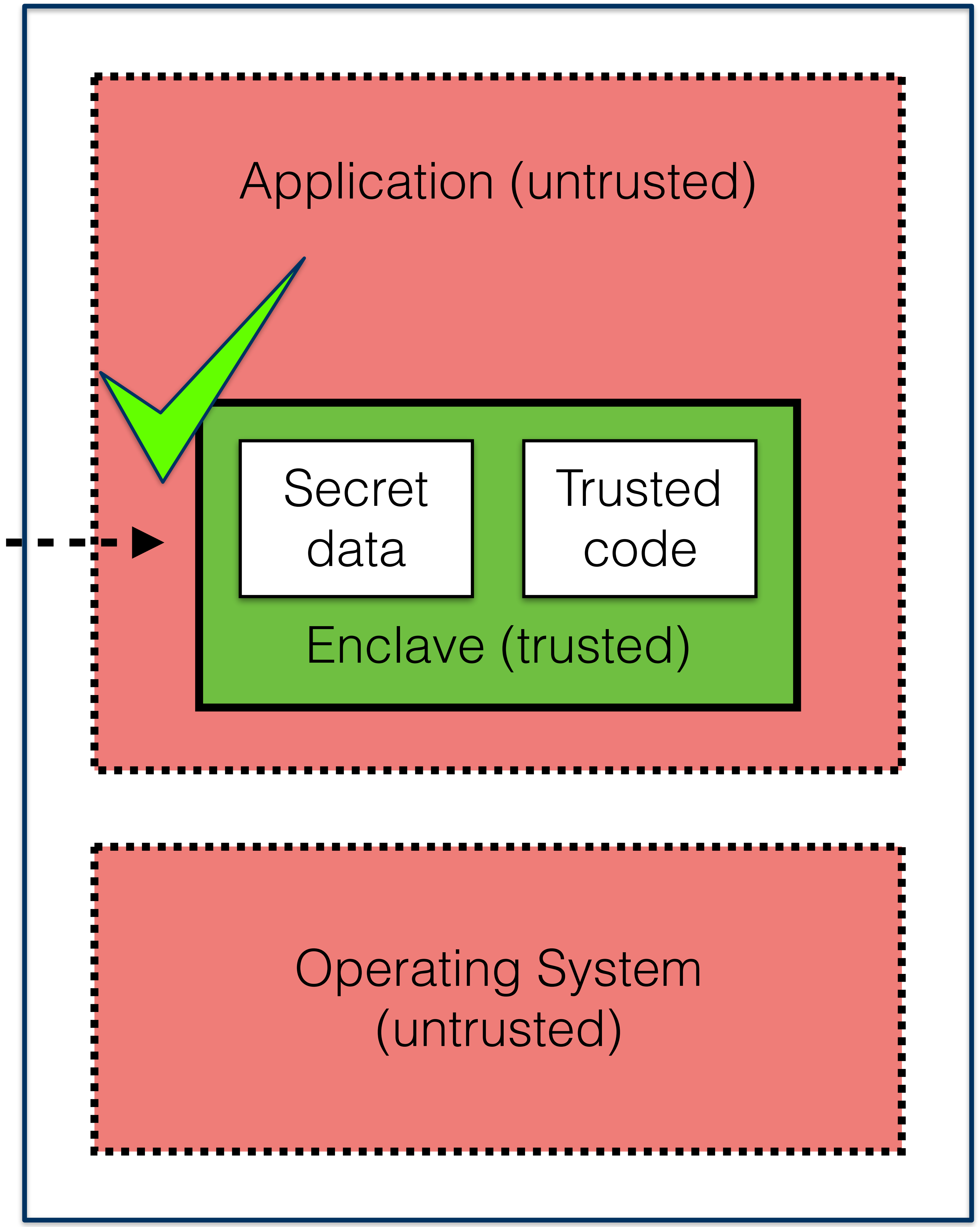
- **Remote attestation** by clients



Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (**enclaves**) protected by hardware

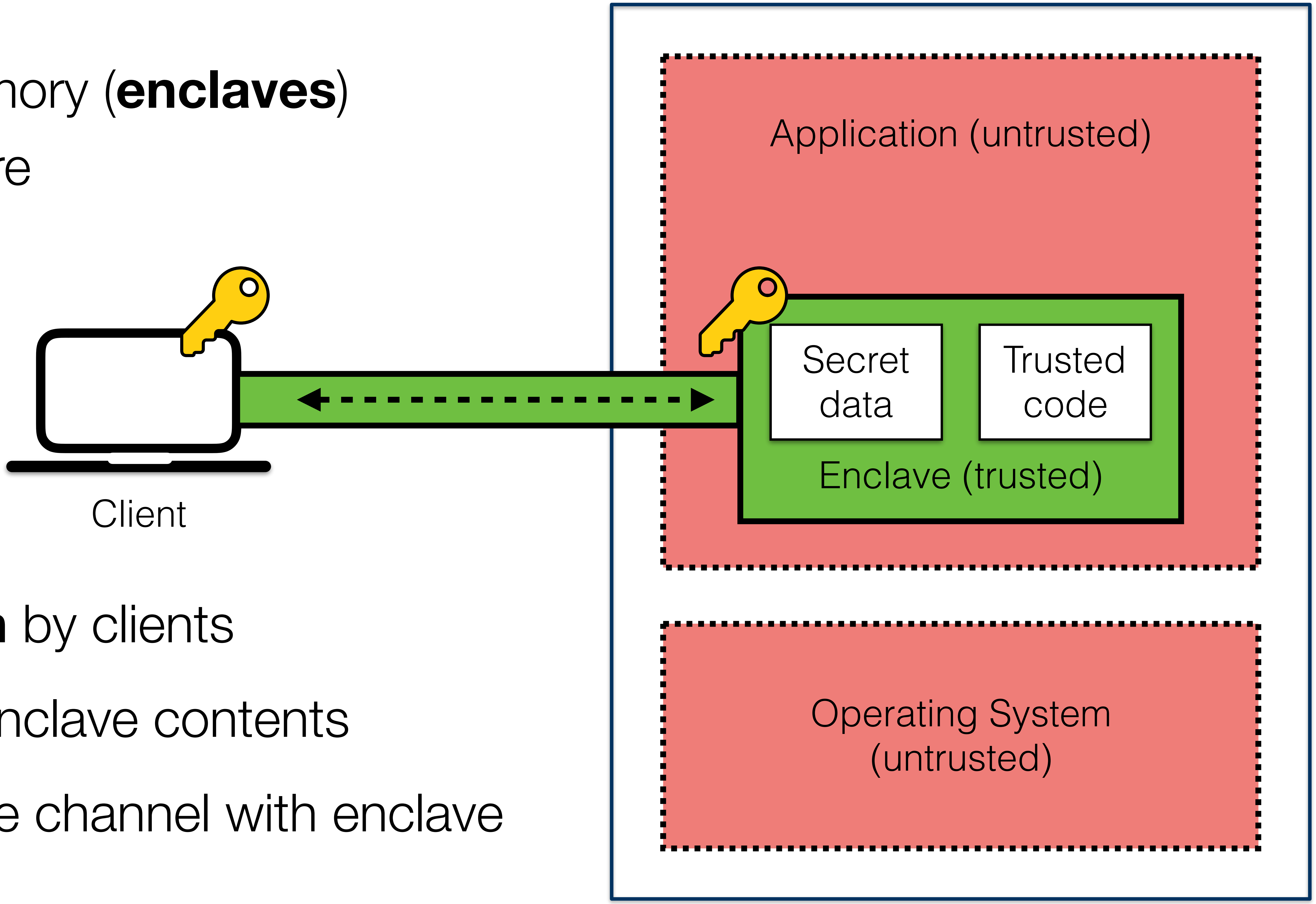




- **Remote attestation** by clients
  - Remotely verify enclave contents

Background: Hardware enclaves (e.g. Intel SGX)

- Secure region of memory (**enclaves**) protected by hardware

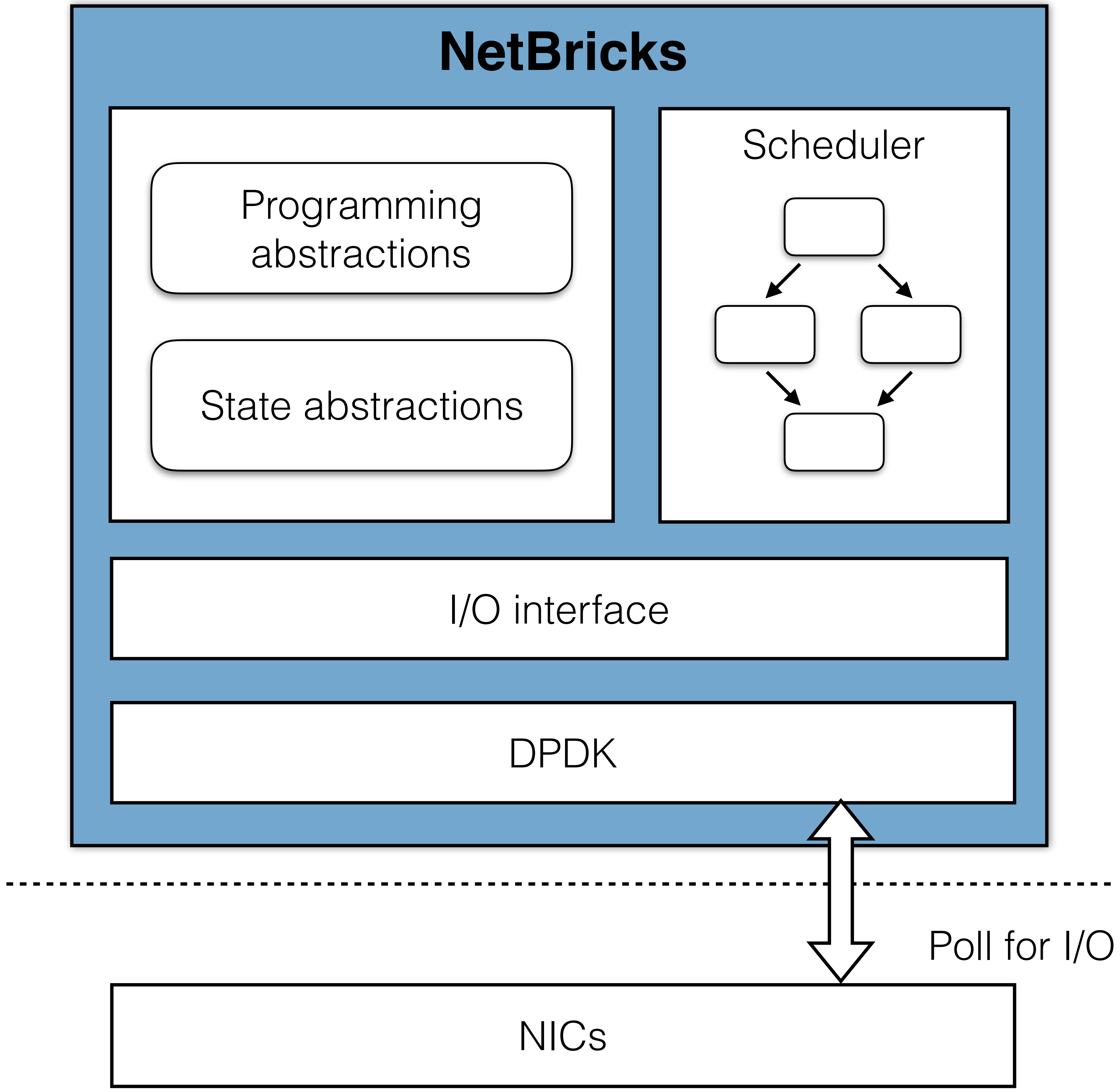


- **Remote attestation** by clients
  - Remotely verify enclave contents
  - Establish a secure channel with enclave

Background: NetBricks

[Panda et al. (OSDI'16)]

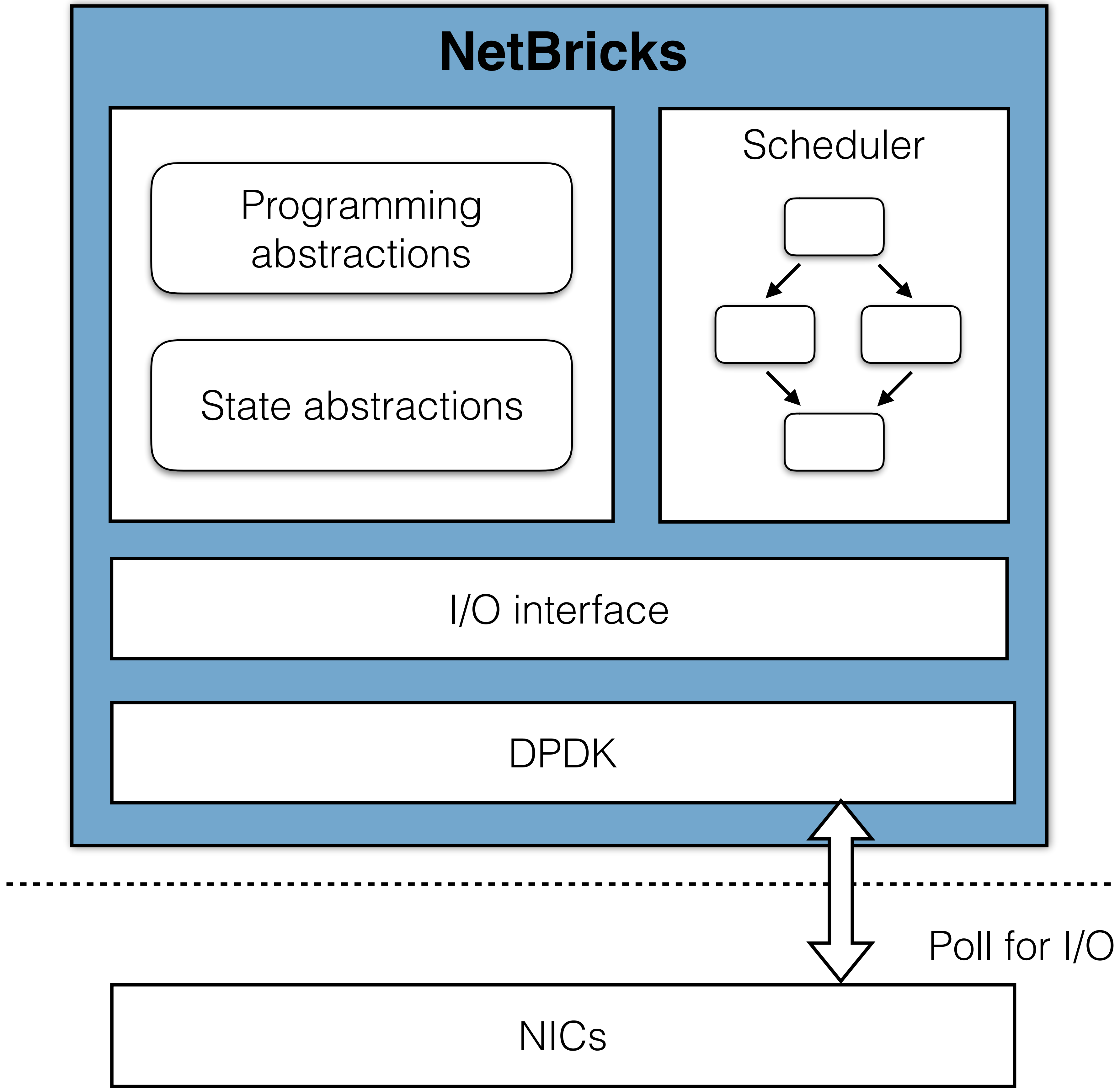
- Framework for developing **arbitrary NFs**



Background: NetBricks

[Panda et al. (OSDI'16)]

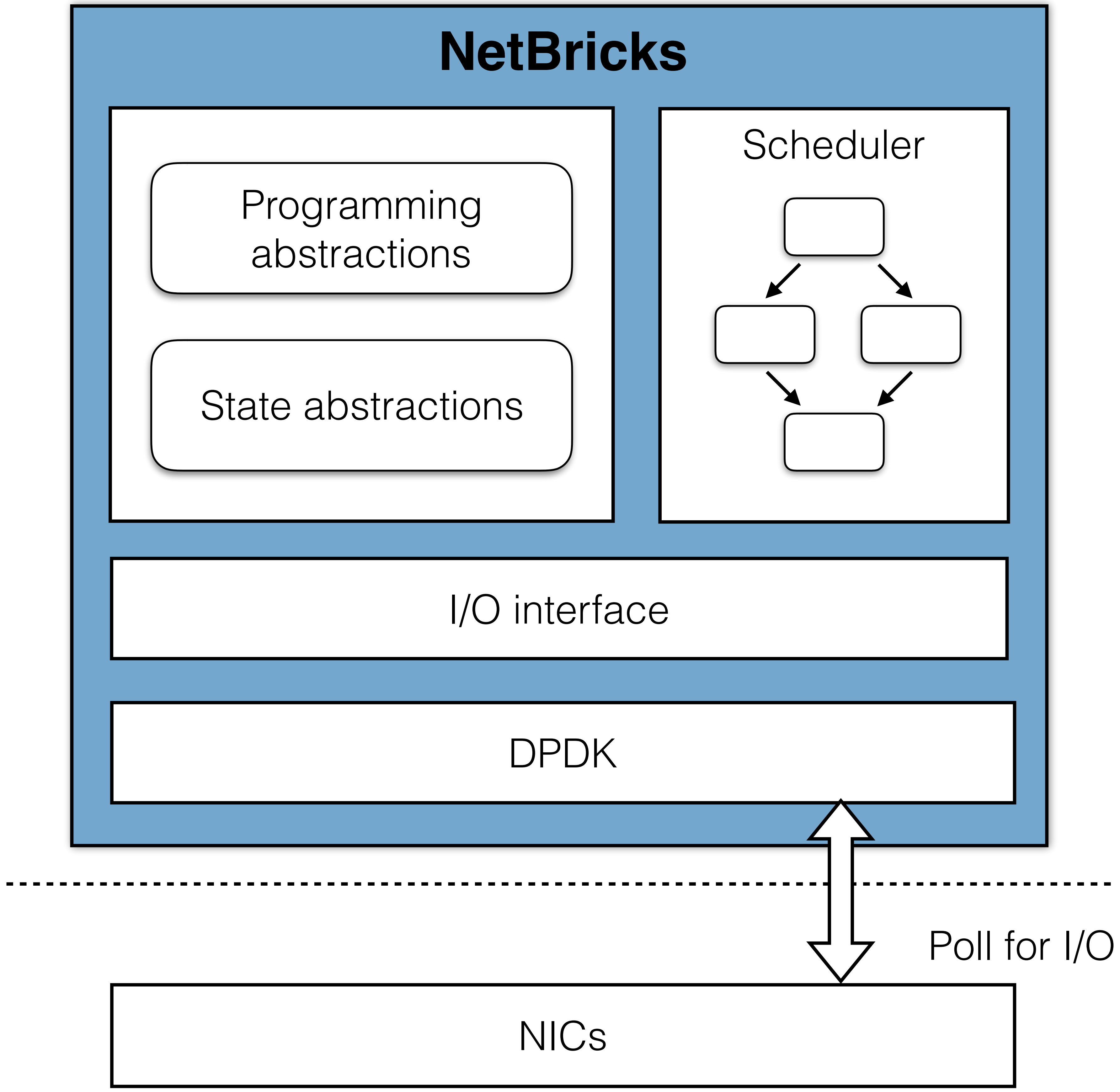
- Framework for developing **arbitrary NFs**
  - MapReduce like programming abstractions (operators) for packet processing



Background: NetBricks

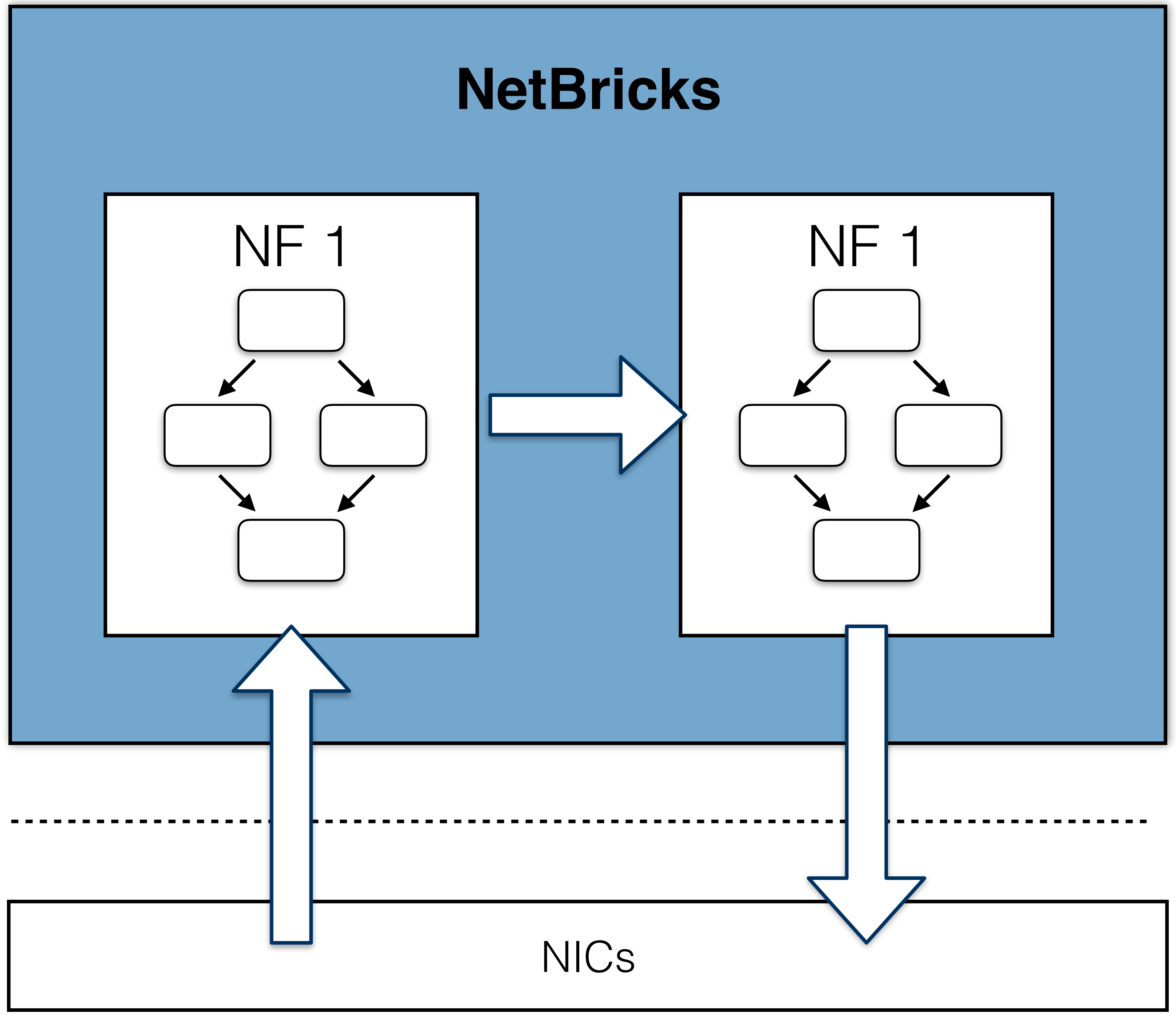
[Panda et al. (OSDI’16)]

- Framework for developing **arbitrary NFs**
  - MapReduce like programming abstractions (operators) for packet processing
  - NFs represented as a **directed graph** with operators as nodes



Background: NetBricks

- Written in **Rust**
  - Fast, safe, zero-copy semantics
  - **Isolates NFs** deployed in a chain while running them in the same address **space**



SafeBricks

1. Protects **traffic** from the **cloud provider**

2. Protects **traffic** from the **NF providers**

3. Protects **NF source code and rulesets** from client enterprise and cloud

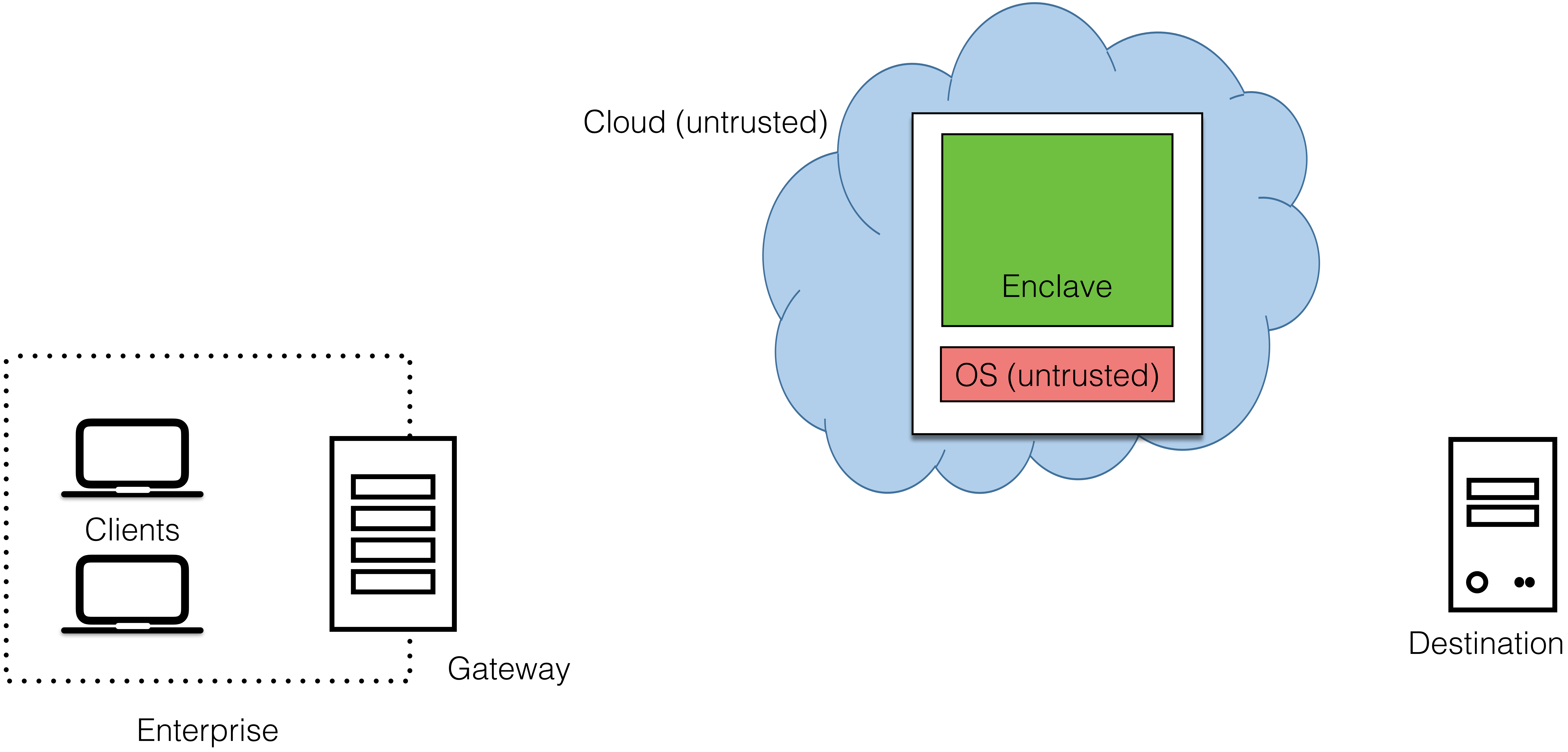
SafeBricks

1 Protects **traffic** from the **cloud provider**

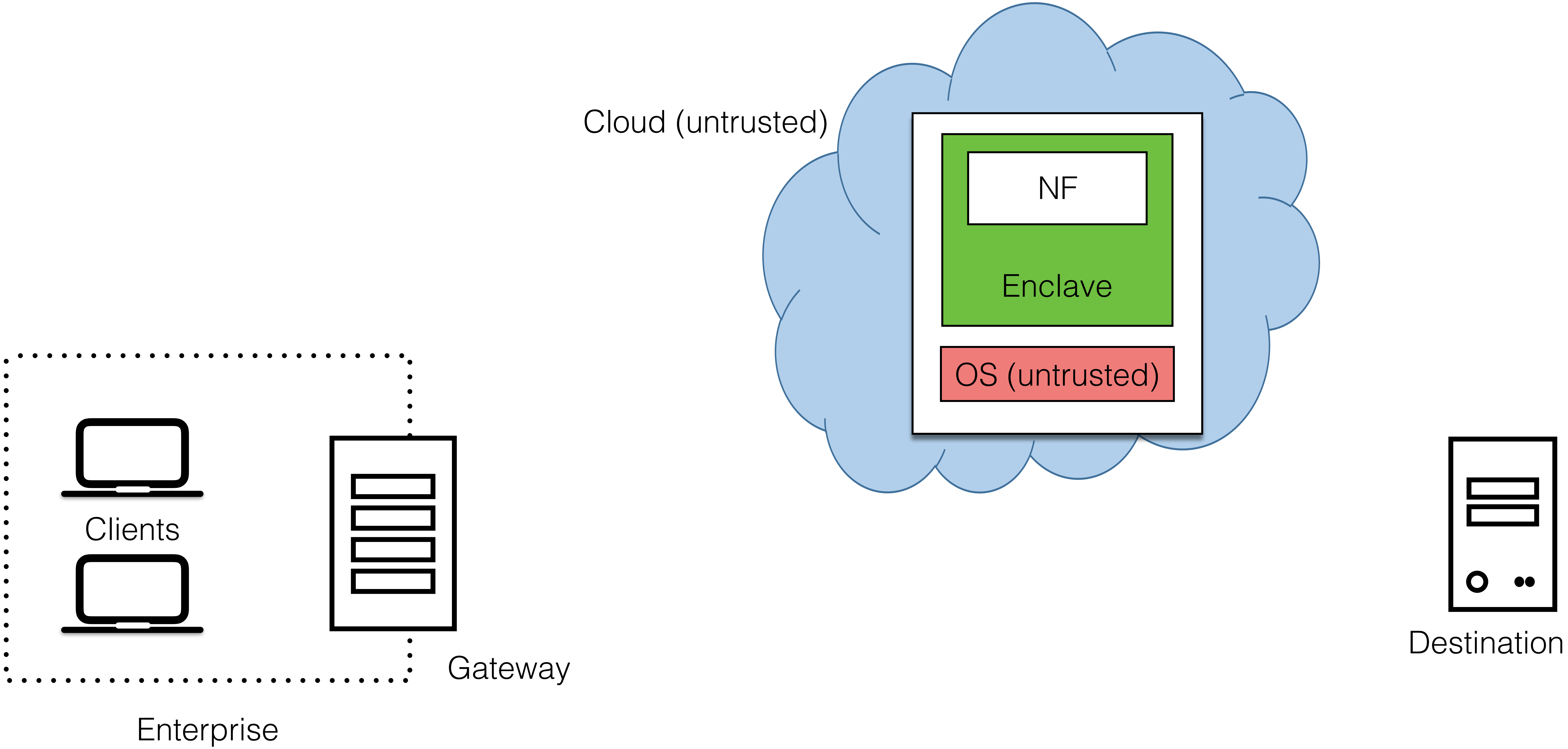
2 Protects **traffic** from the **NF providers**

3 Protects **NF source code and rulesets** from client enterprise and cloud

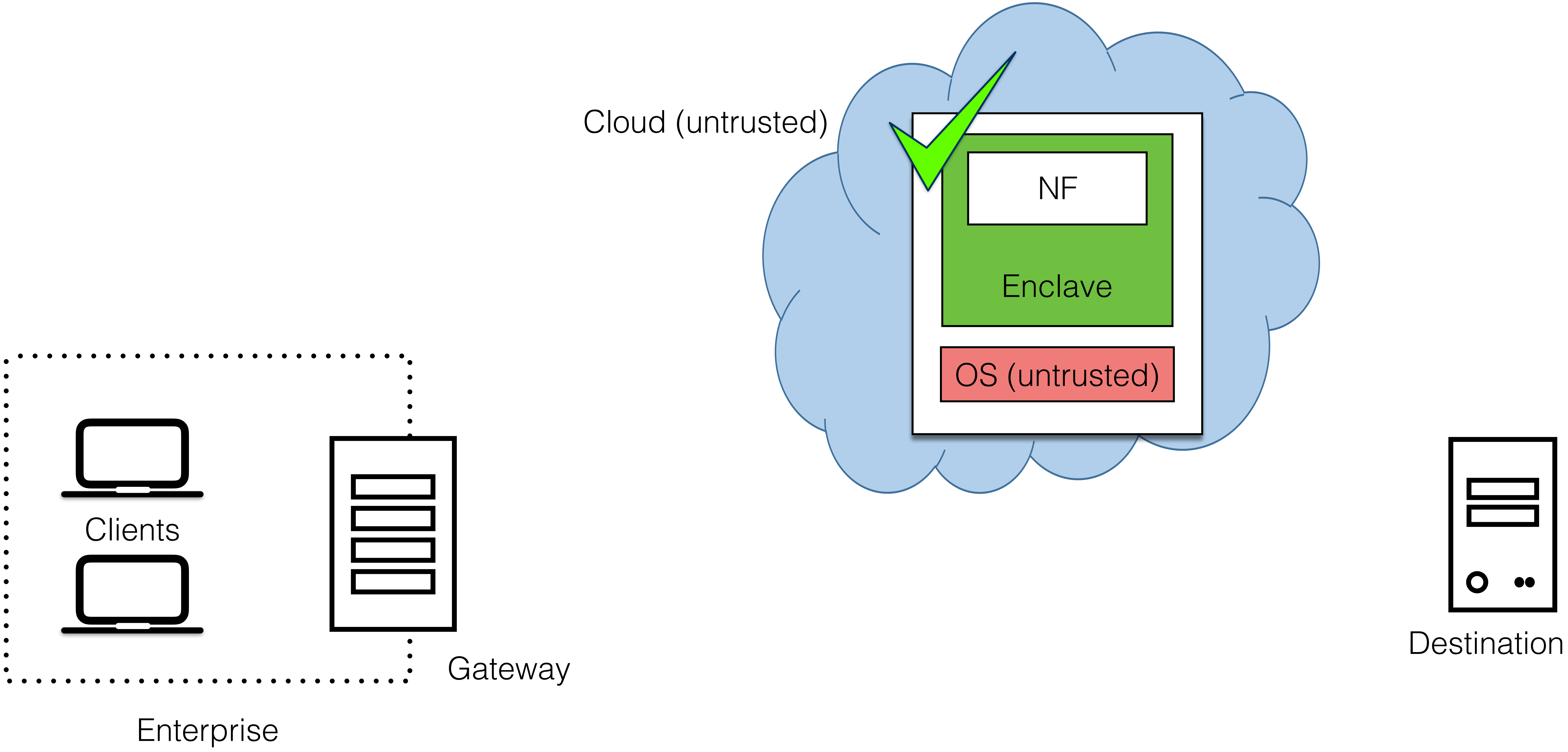
Outsourcing NFs using hardware enclaves



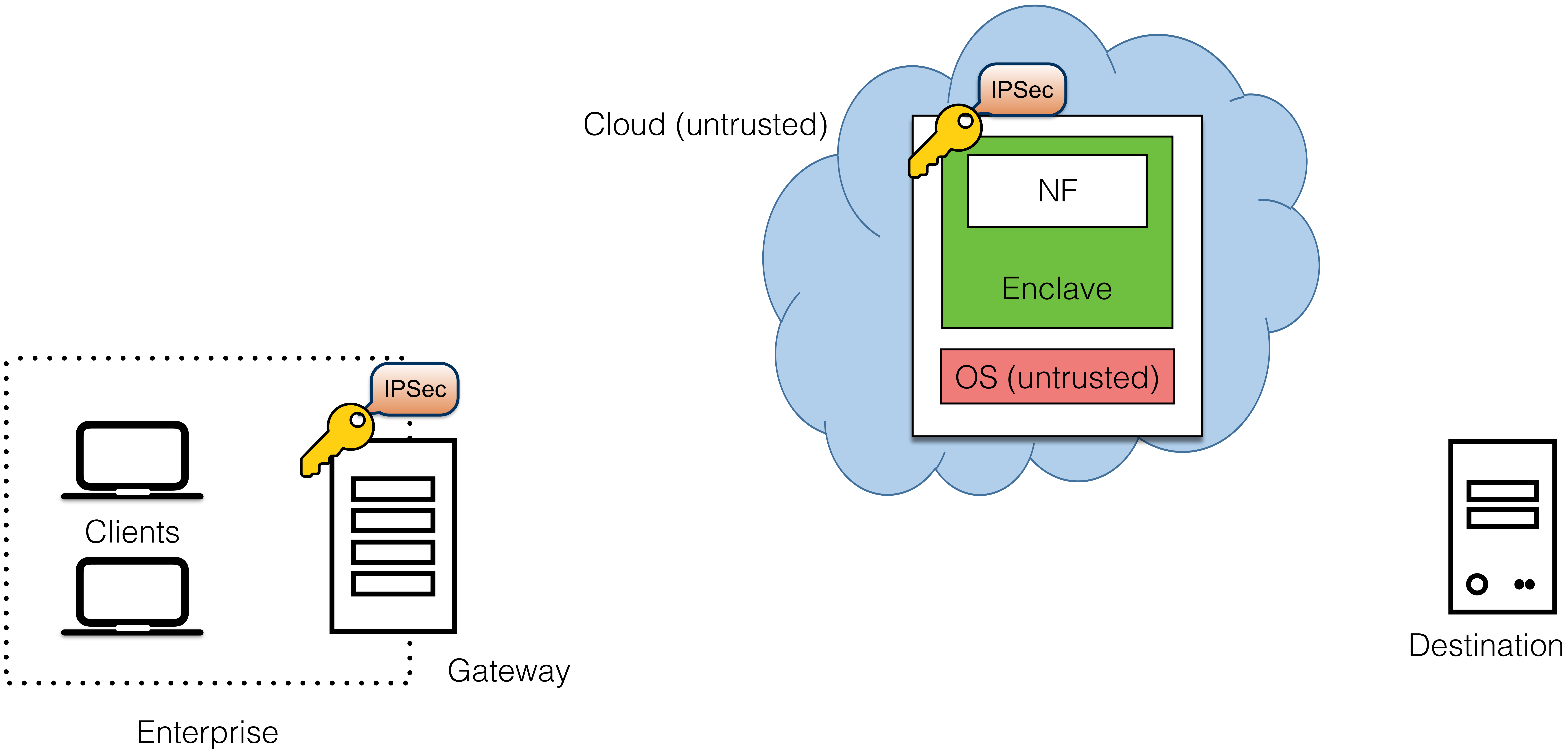
Outsourcing NFs using hardware enclaves



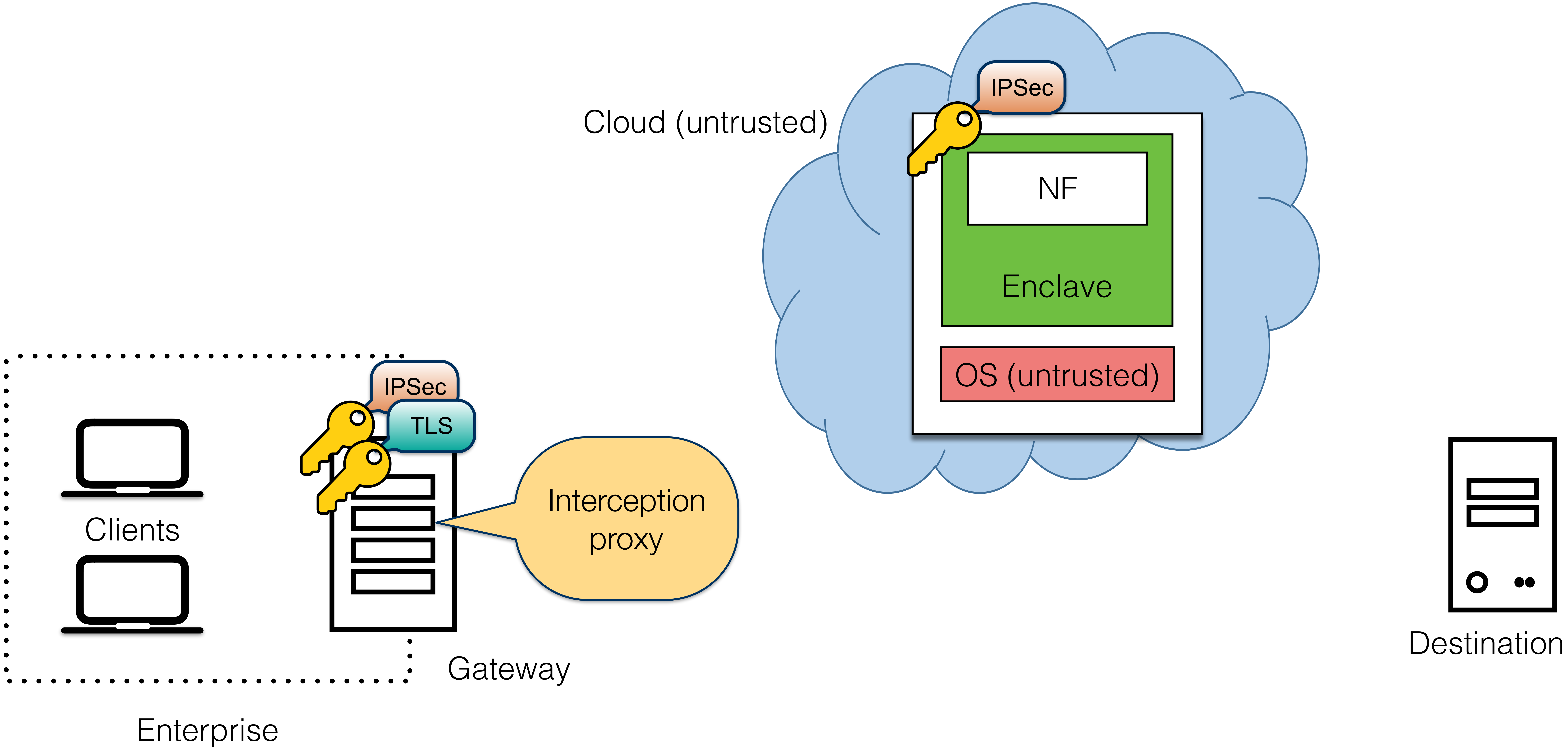
Outsourcing NFs using hardware enclaves



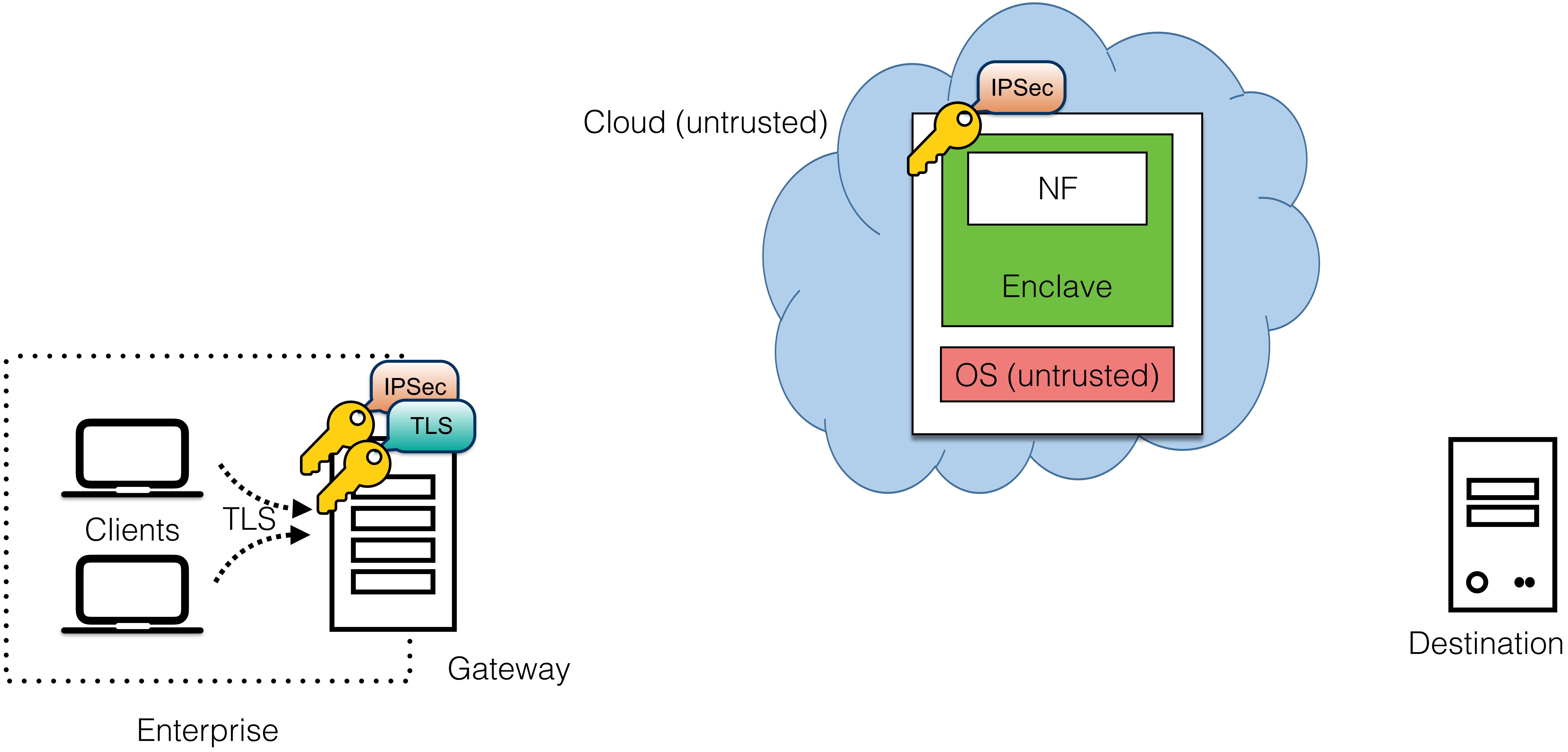
Outsourcing NFs using hardware enclaves



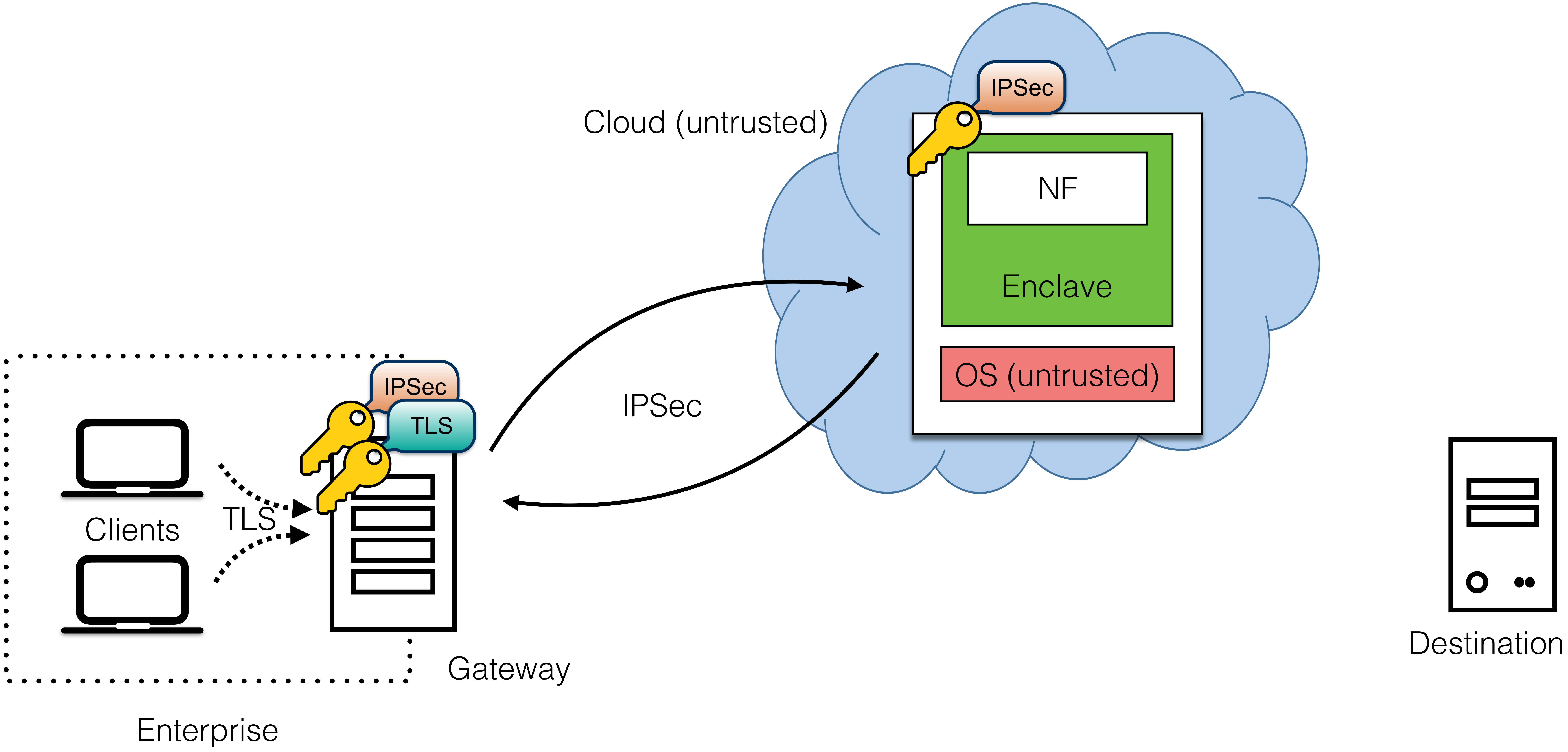
Outsourcing NFs using hardware enclaves



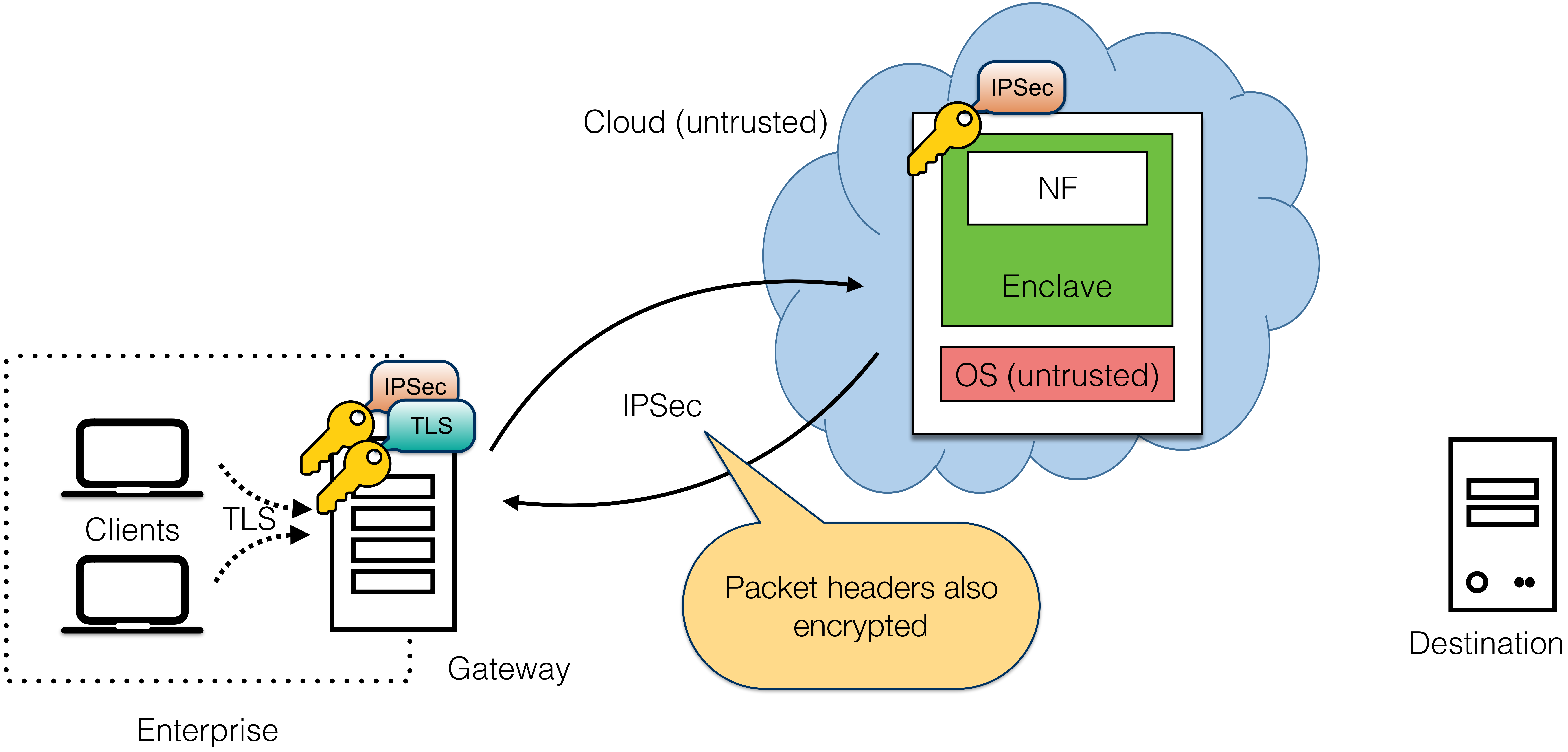
Outsourcing NFs using hardware enclaves



Outsourcing NFs using hardware enclaves

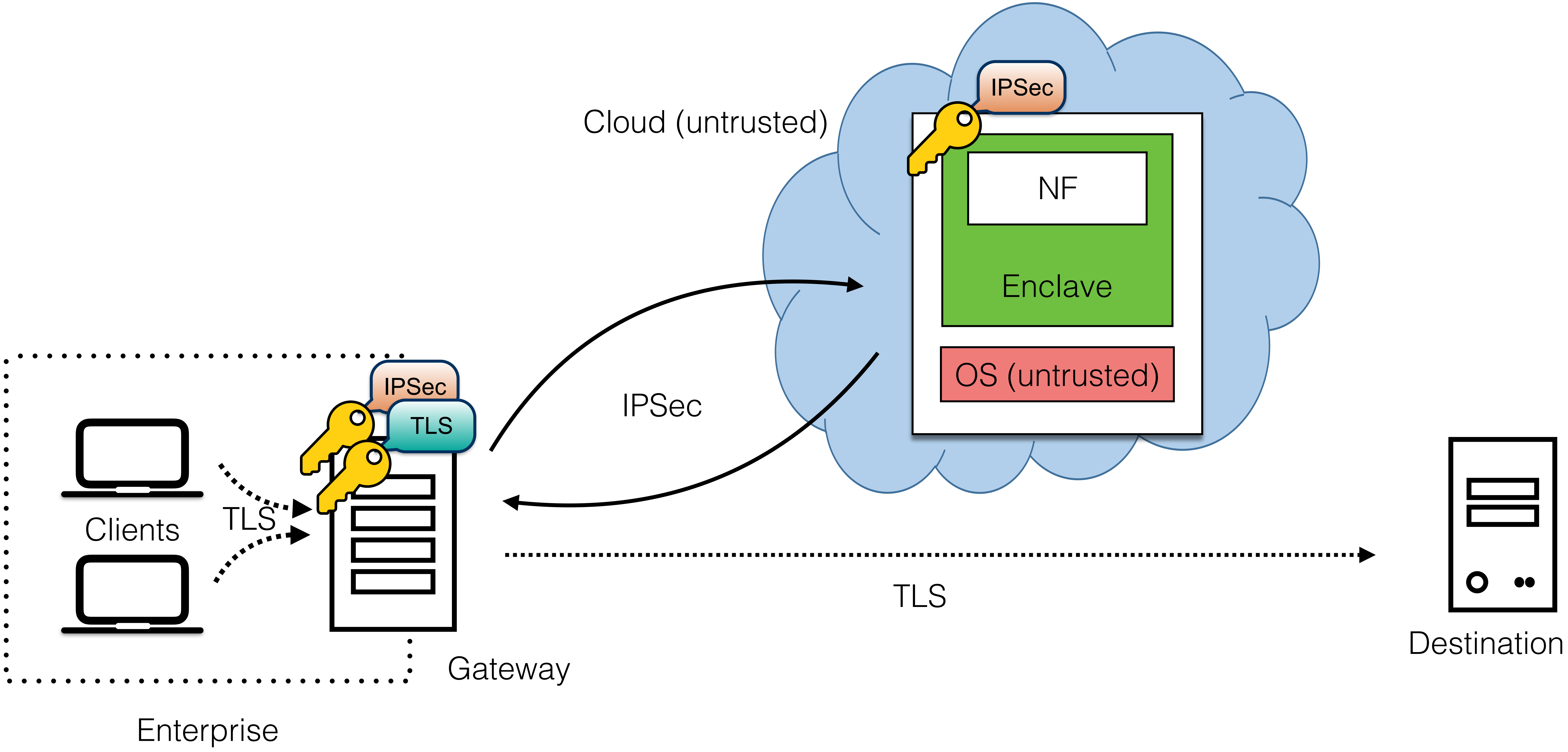


Outsourcing NFs using hardware enclaves

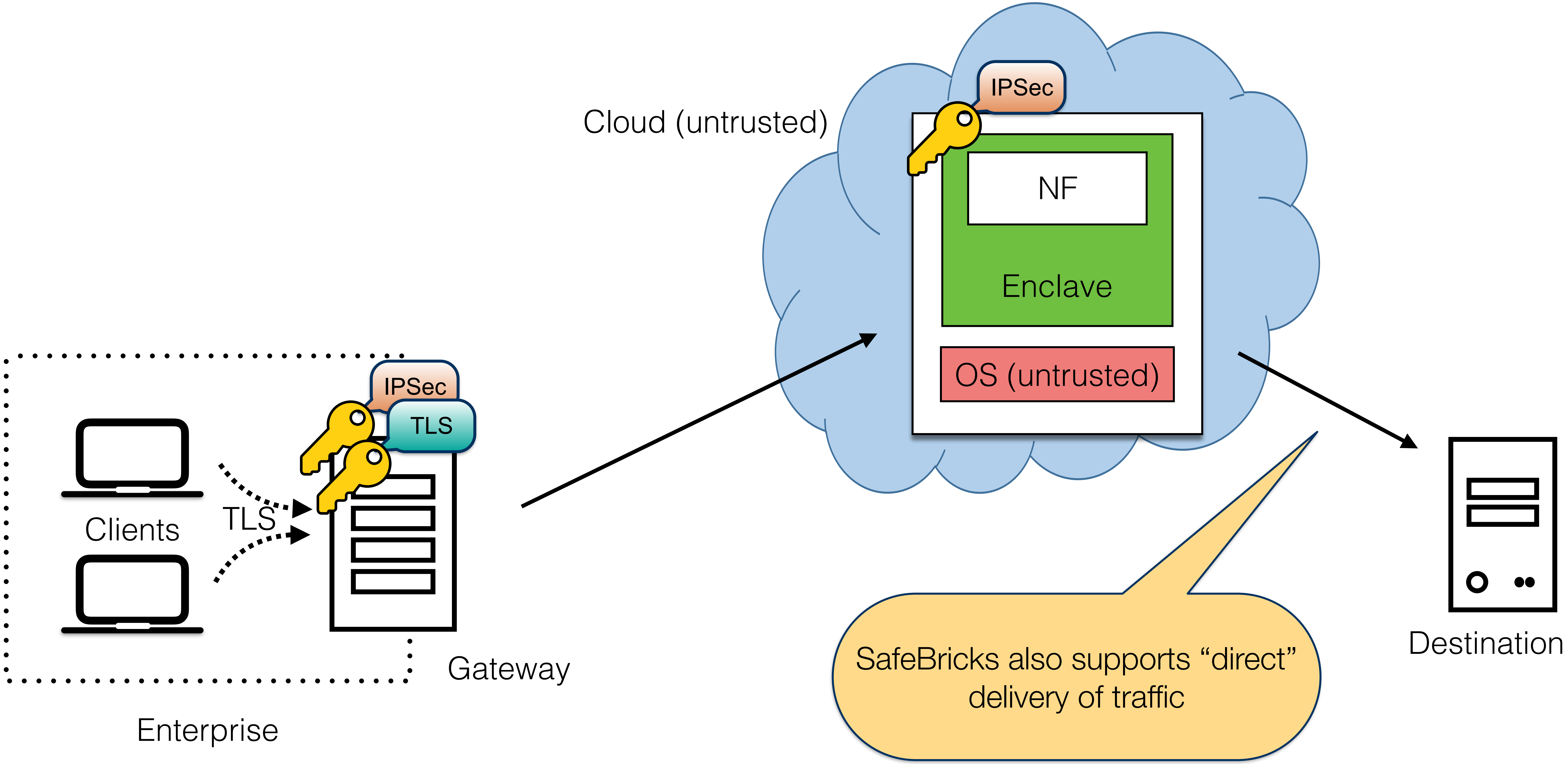


Packet headers also encrypted

Outsourcing NFs using hardware enclaves

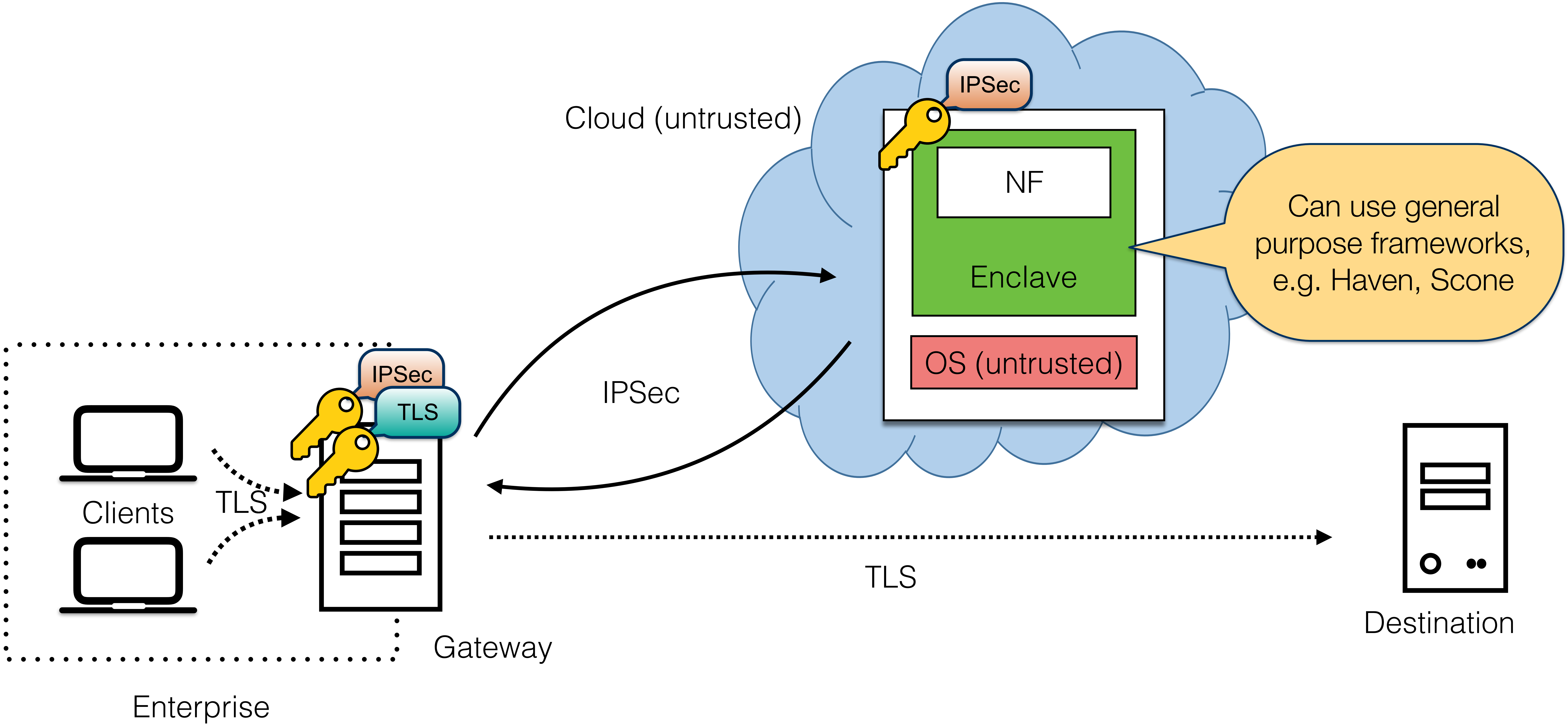


Outsourcing NFs using hardware enclaves



SafeBricks also supports “direct” delivery of traffic

Outsourcing NFs using hardware enclaves



Can use general purpose frameworks, e.g. Haven, Scone

Challenges

1 **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

3. **Illegal enclave instructions** — SGX does not support system calls or instructions that may lead to a `VMEXIT`

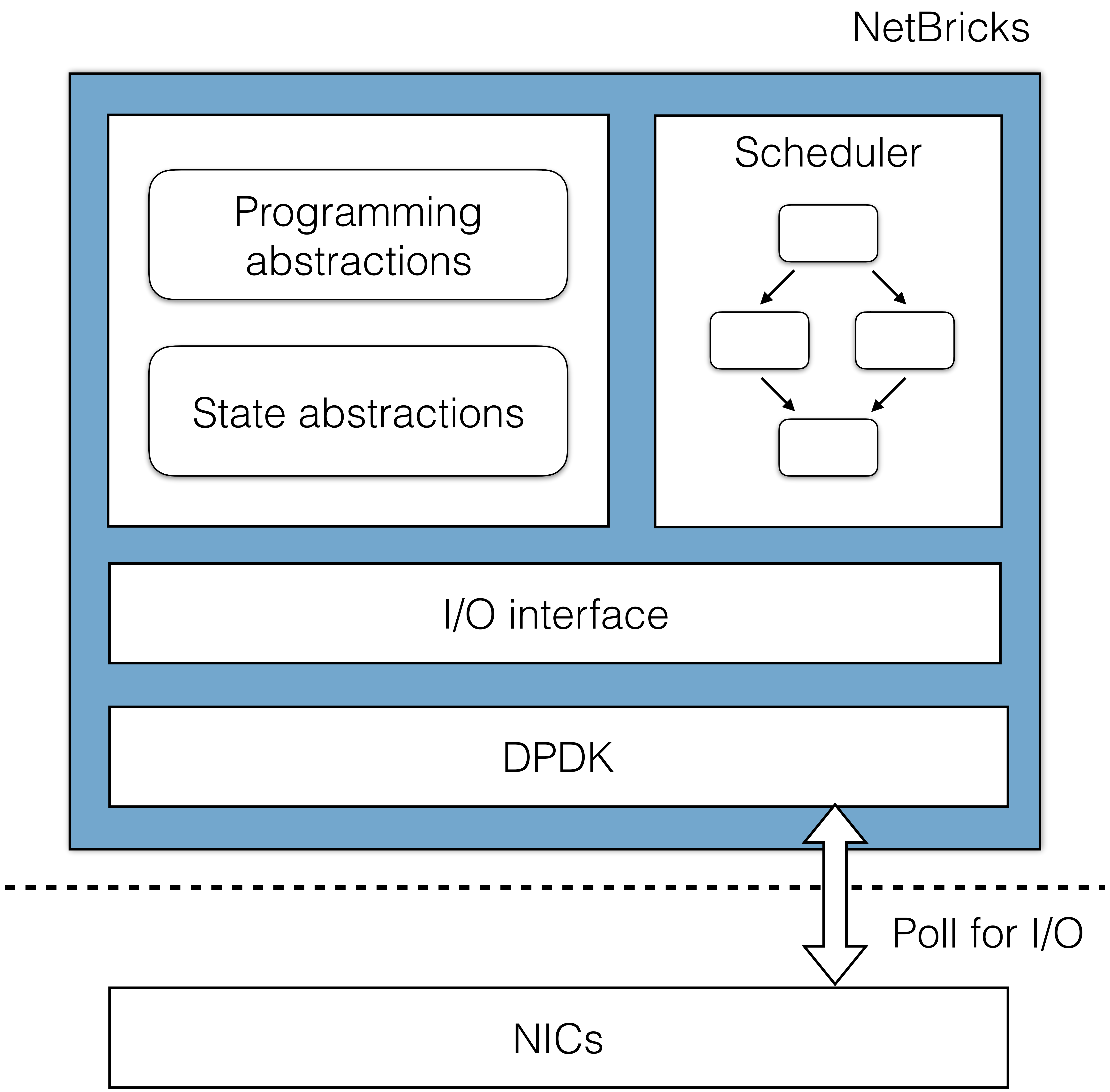
Challenges

1 **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

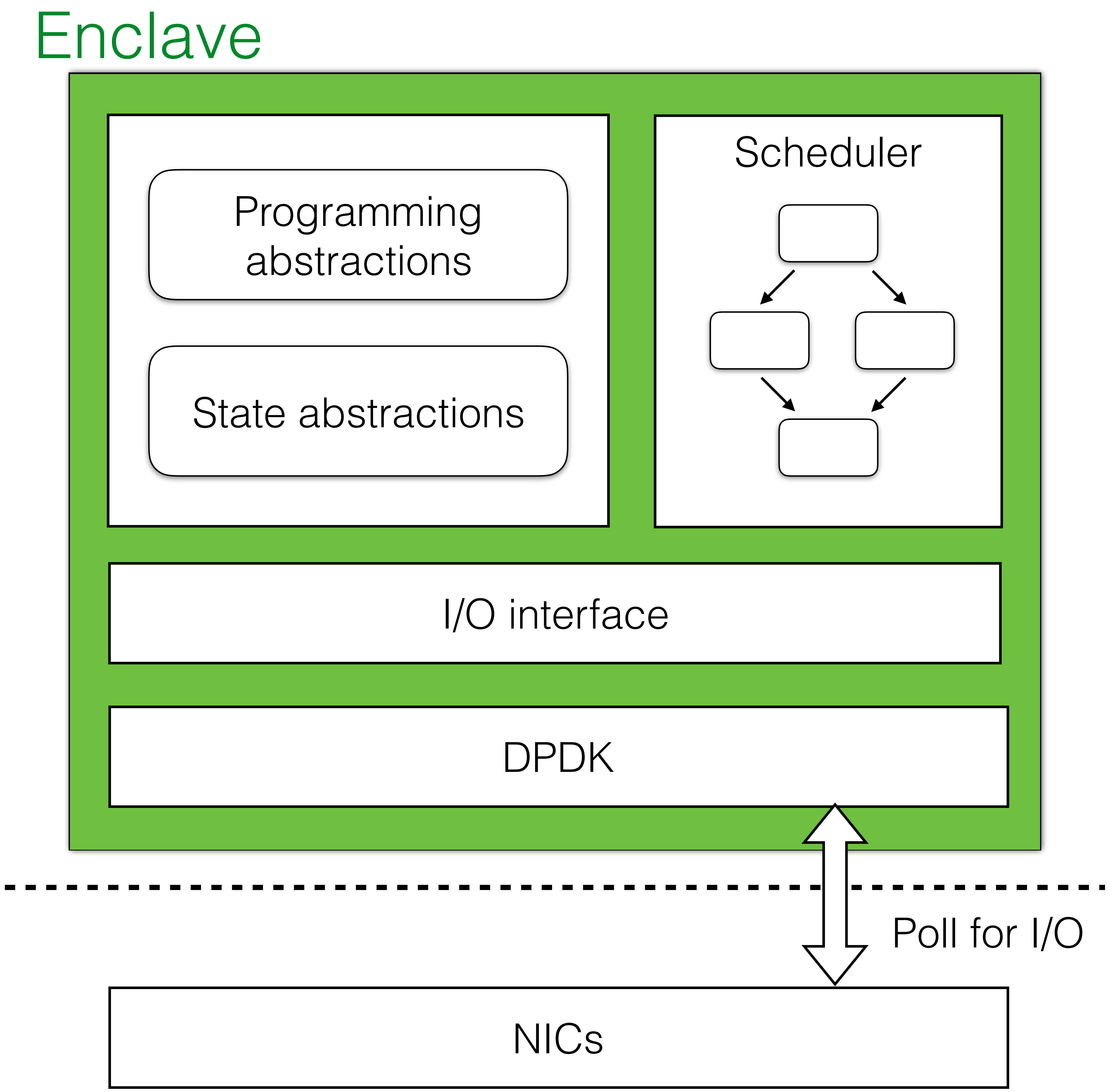
2 **High performance** — Transitioning into / out of enclaves is expensive!

3 **Illegal enclave instructions** — SGX does not support system calls or instructions that lead to a `VMEXIT`

1

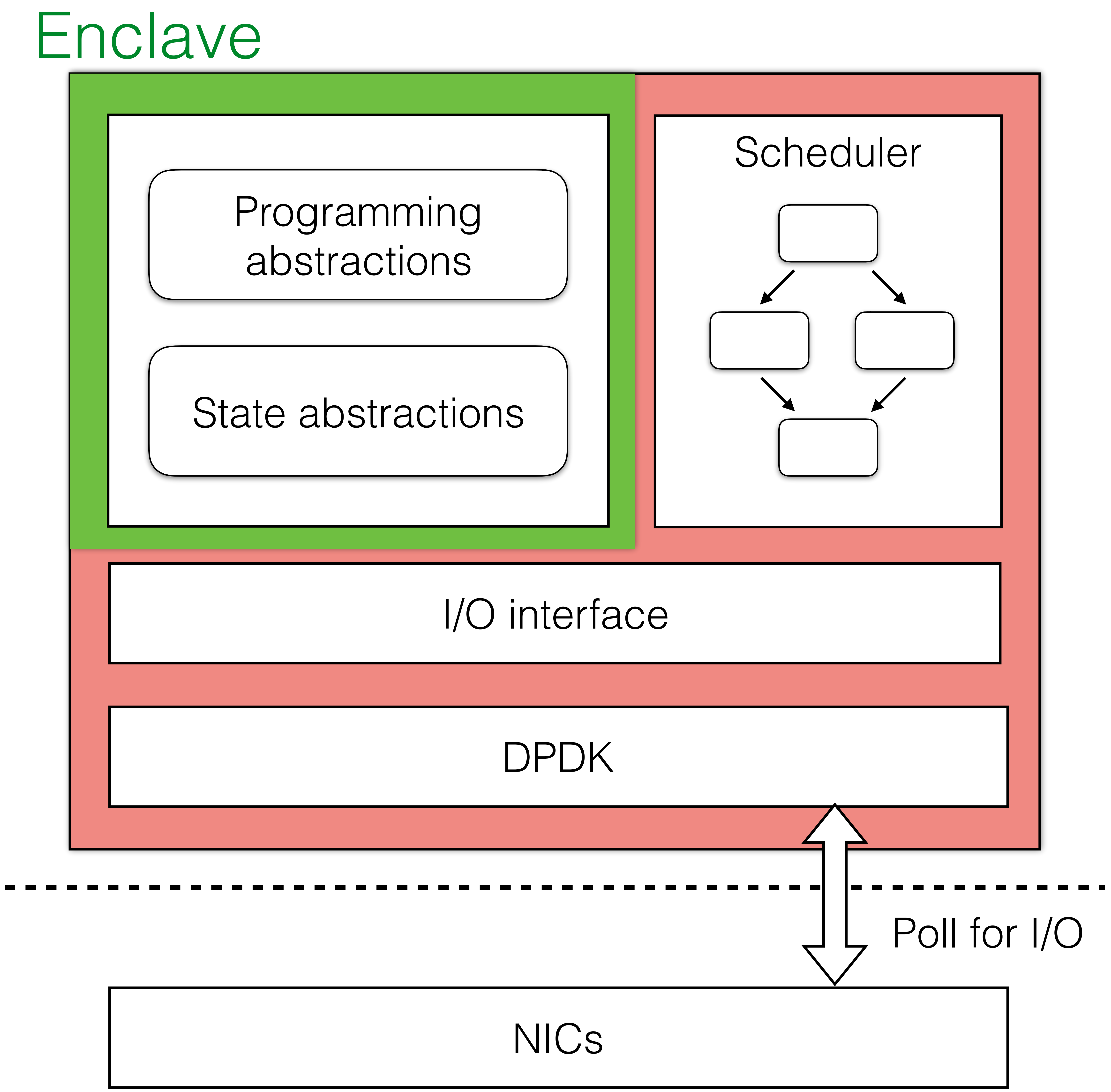

NetBricks
Programming abstractions
State abstractions
Scheduler
I/O interface
DPDK
Poll for I/O
NICs

1



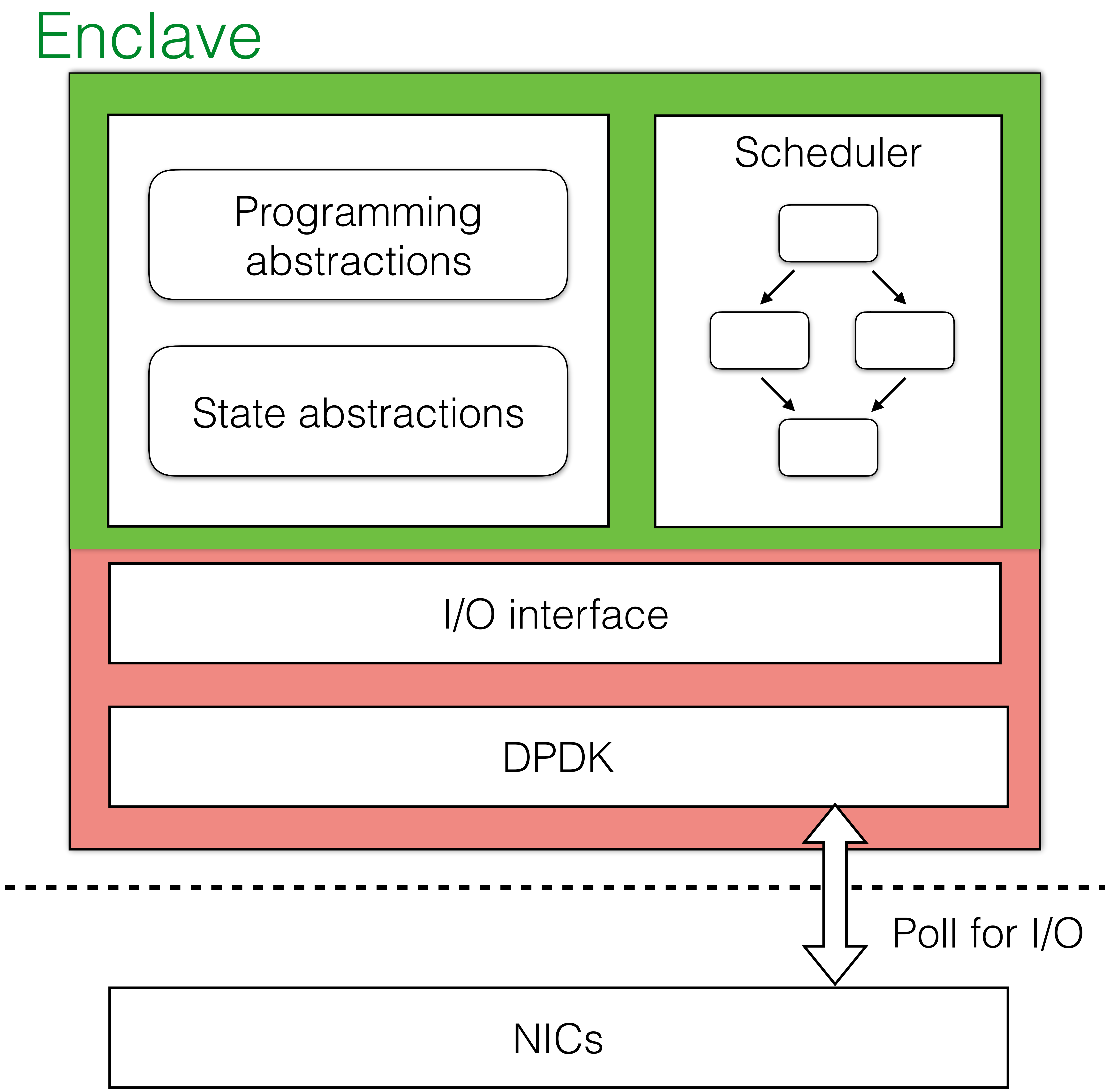
- **Maximal TCB:** NetBricks stack entirely within enclave

1

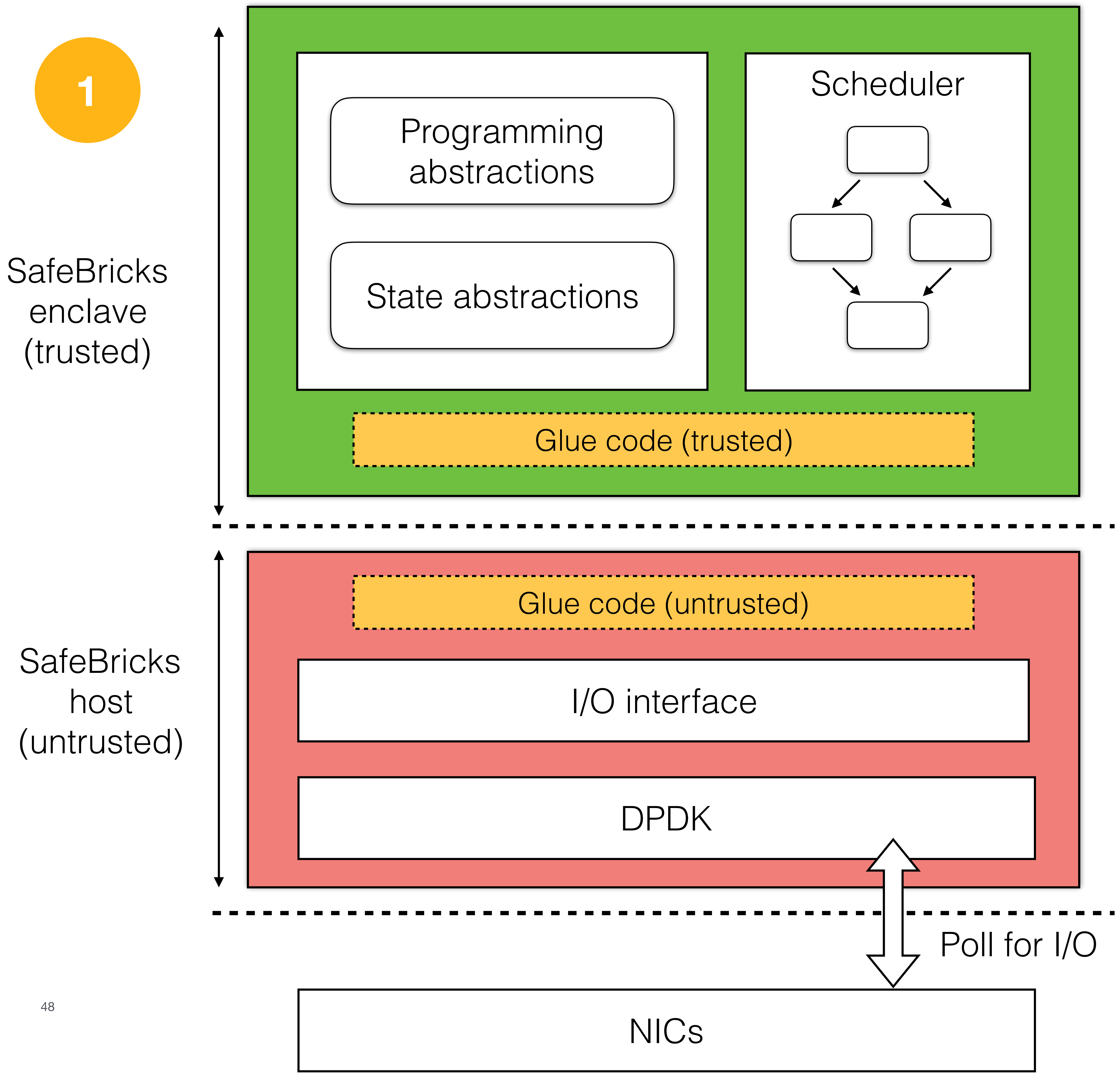


- **Minimal TCB:** Only security-critical components within enclave
- One enclave transition **per node per packet batch**

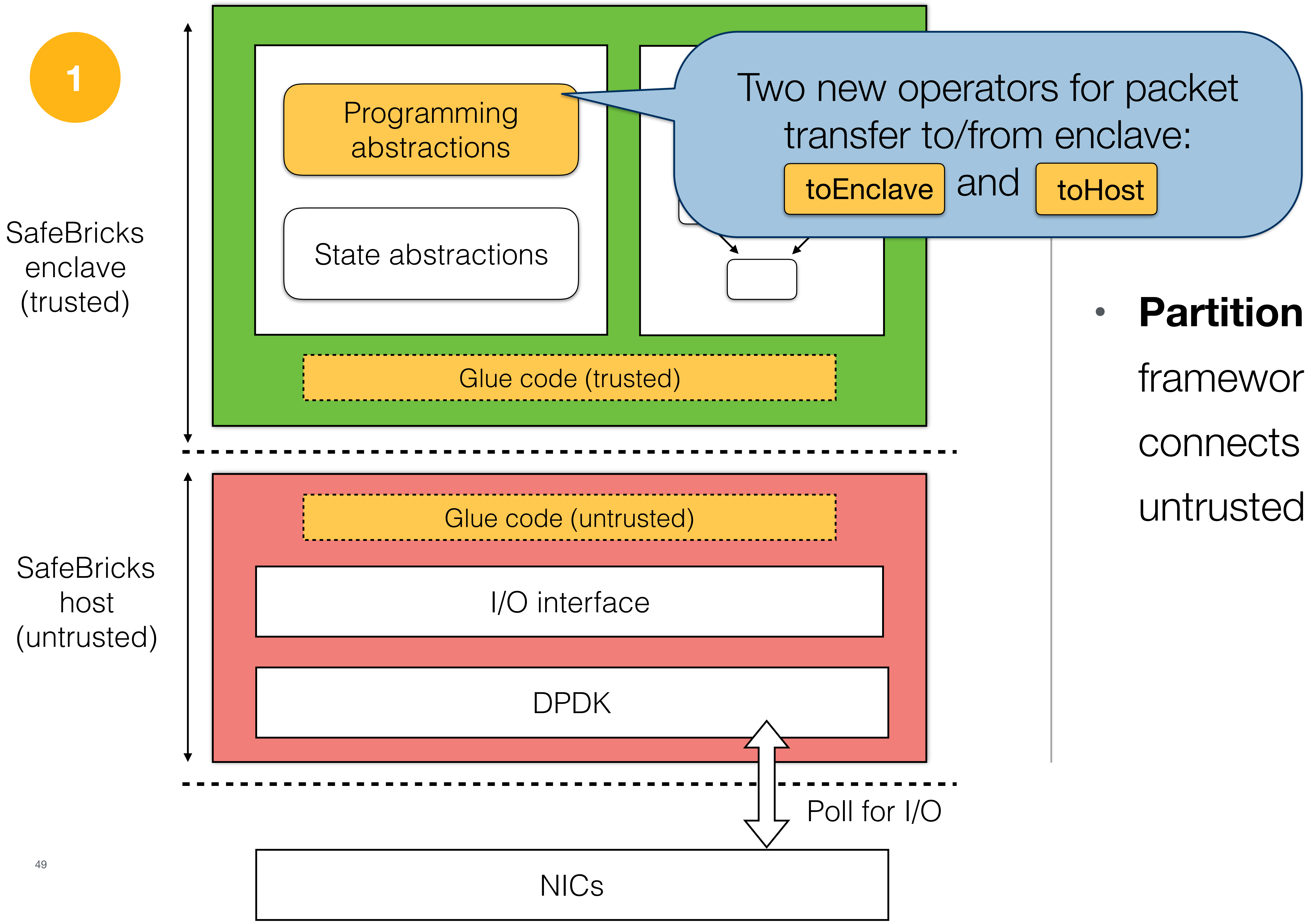
1



- **Intermediate** TCB
- One enclave **transition per packet batch**



- **Partitioned** NetBricks framework; glue code connects trusted and untrusted code



Two new operators for packet transfer to/from enclave: toEnclave and toHost

- **Partitioned** NetBricks framework; glue code connects trusted and untrusted code

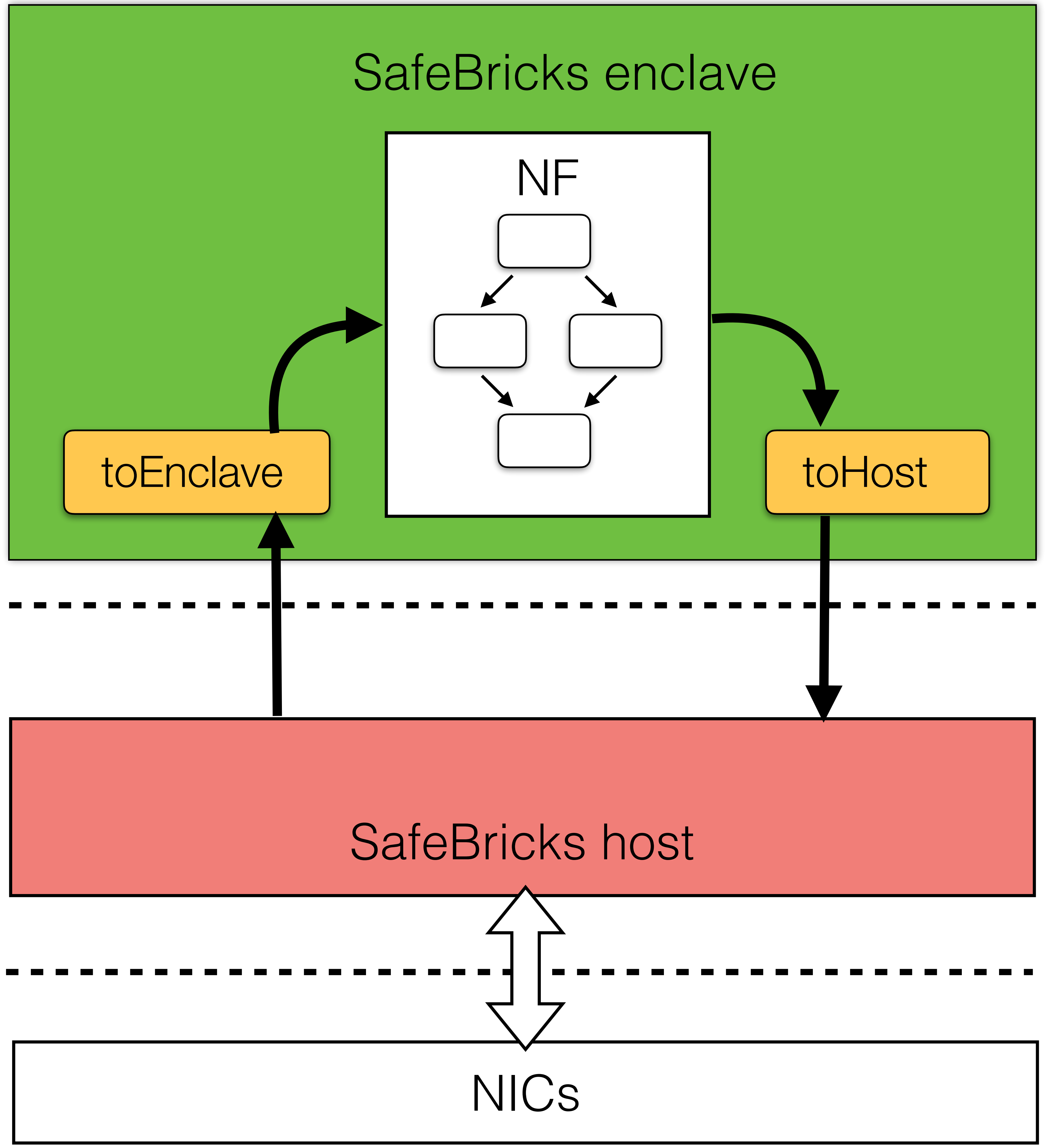
Challenges

1. **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2. **High performance** — Transitioning into / out of enclaves is expensive!

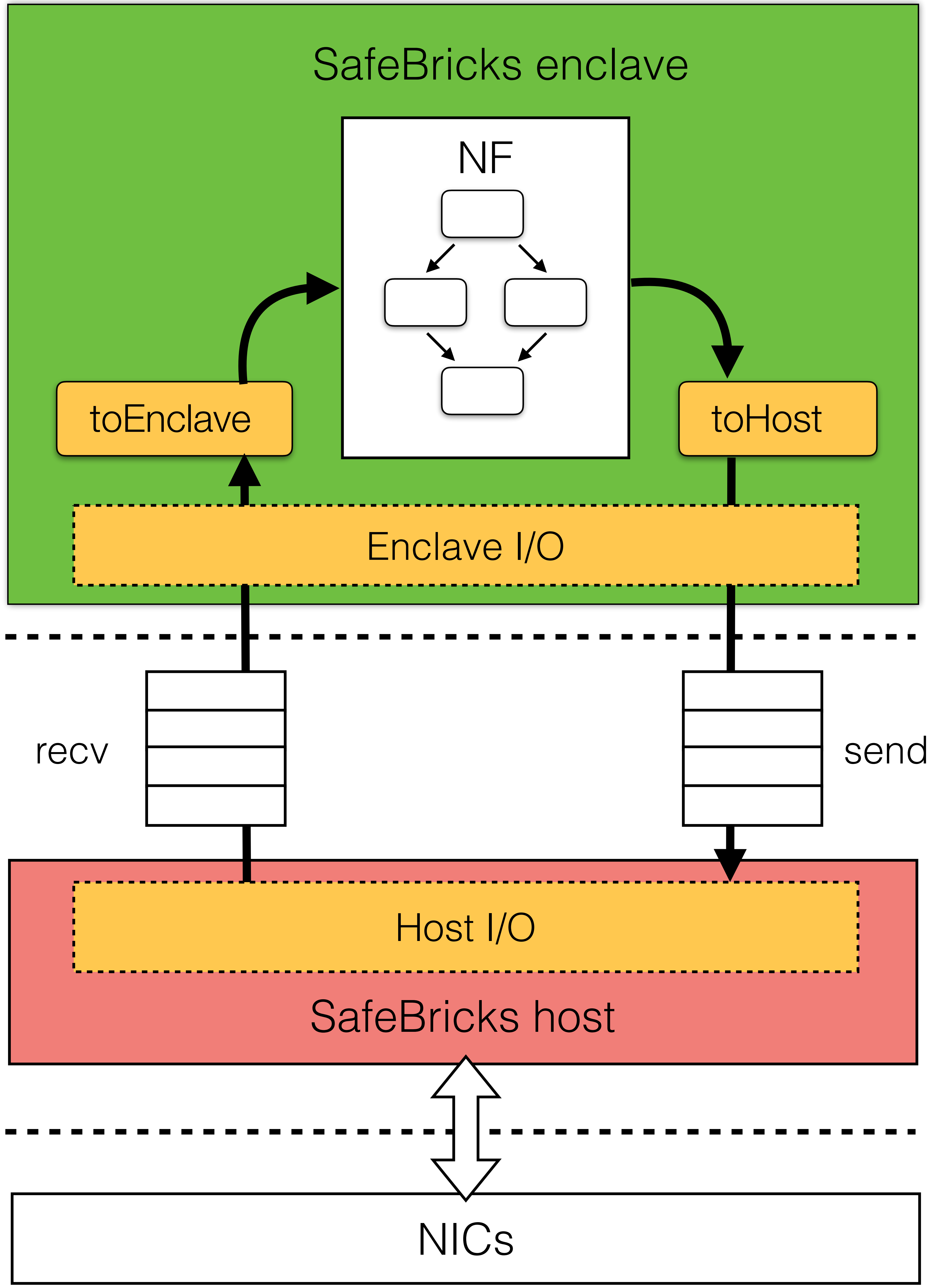
3. **Illegal enclave instructions** — SGX does not support system calls or instructions that lead to a `VMEXIT`

2



- One enclave transition per packet batch

2



- **Shared queues** in non-enclave heap
- Separate enclave and host threads
- Access queues without exiting enclave — **zero enclave transitions**

Challenges

1 **Small trusted computing base (TCB)** — enclave should contain minimal amount of code

2 **High performance** — Transitioning into / out of enclaves is expensive!

3 **Illegal enclave instructions** — SGX does not support system calls or instructions that lead to a `VMEXIT`

3

Observation: NFs in general do not require support for system calls / instructions that lead to VMEXITs

3

Observation: NFs in general do not require support for system calls / instructions that lead to VMEXITs, except:

- Logging
- Timestamps (using `rdtsc`)

3

Observation: NFs in general do not require support for system calls / instructions that lead to VMEXITs, except:

- Logging
- Timestamps (using `rdtsc`)

SafeBricks designs **custom solutions** for these operations without enclave transitions

SafeBricks

1 Protects **traffic** from the **cloud provider**

2 Protects **traffic** from the **NF providers**

3 Protects **NF source code and rulesets** from client enterprise and cloud

Problem: Malicious NFs within enclaves

⚠ **Malicious NFs** inside the enclave can exfiltrate or tamper with packets!

Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!

Observation: NFs typically need access **only to specific packet fields**

- E.g. Firewall needs read-only access to TCP/IP headers
- E.g. NAT needs both read-write access to headers but not to packet payload

Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!

Observation: NFs typically need access **only to specific packet fields**

- E.g. Firewall needs read-only access to TCP/IP headers
- E.g. NAT needs both read-write access to headers and payload

IP addresses; TCP ports; HTTP payload

Problem: Malicious NFs within enclaves

Malicious NFs inside the enclave can exfiltrate or tamper with packets!

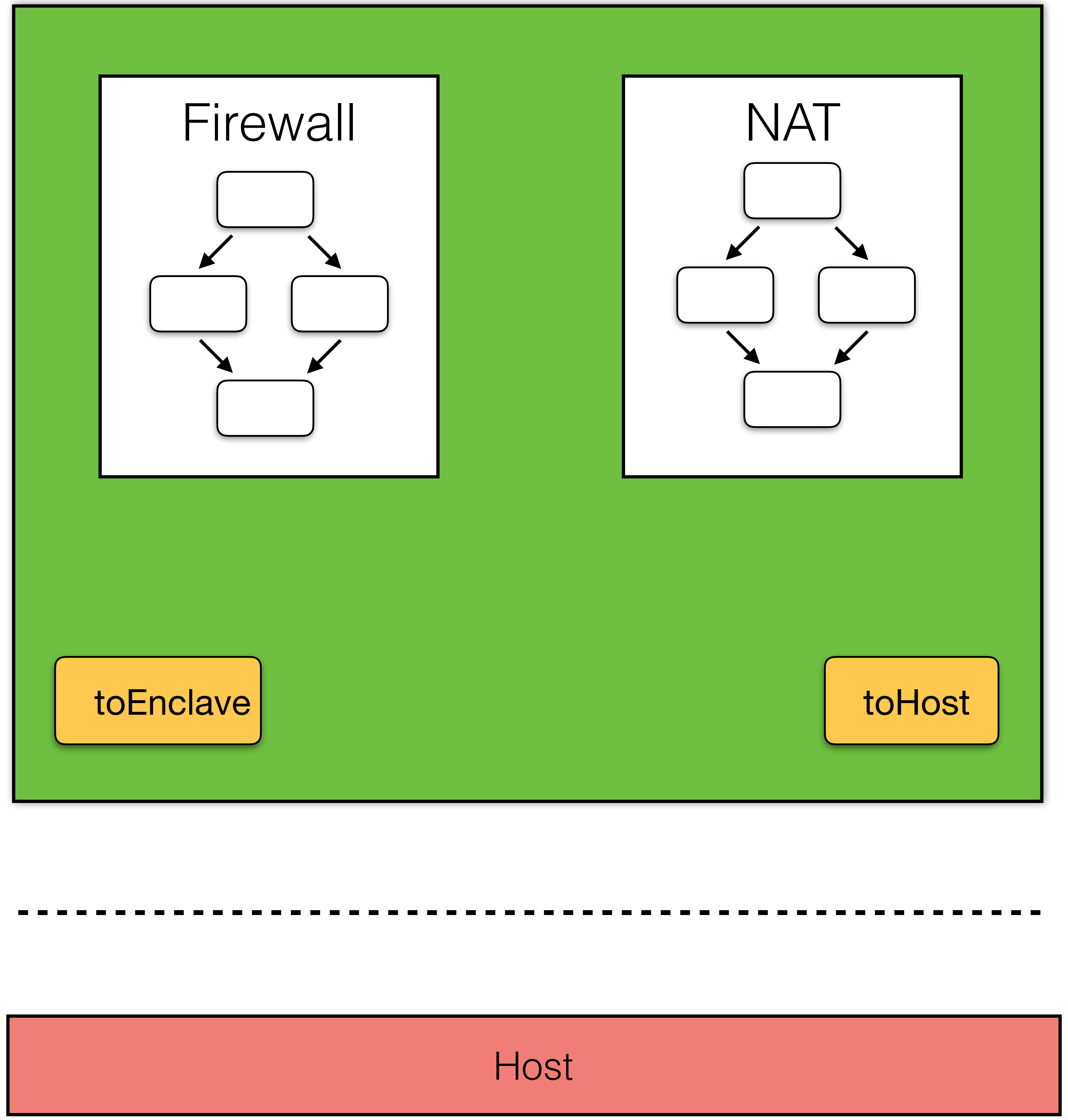
Observation: NFs typically need access **only to specific packet fields**

- E.g. Firewall needs read-only access to TCP/IP headers
- E.g. NAT needs both read-write access to headers but not to packet payload

SafeBricks enforces **least privilege** across NFs within the enclave

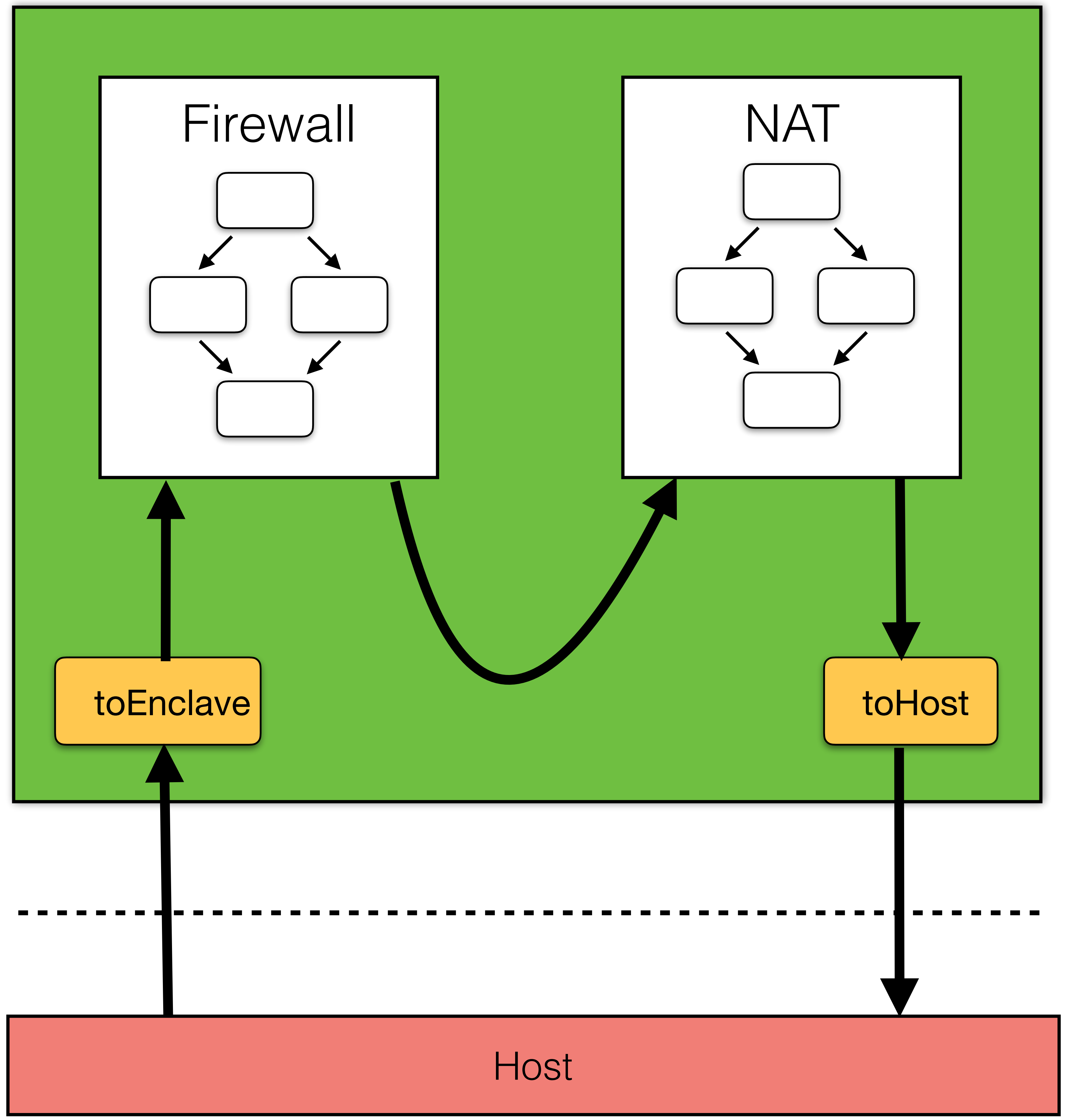
Least privilege enforcement

Run NFs within the **same** enclave



Least privilege enforcement

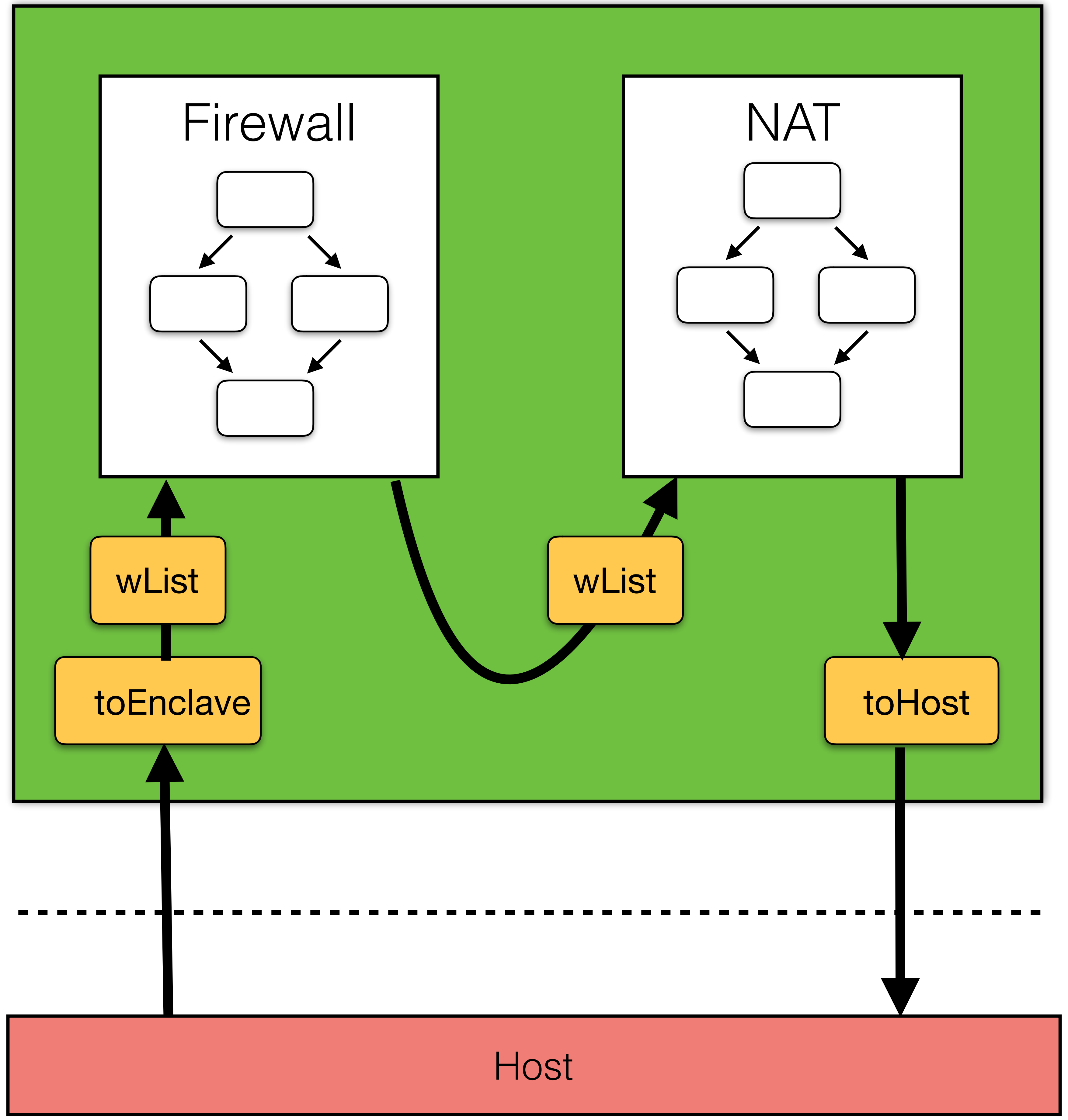
Run NFs within the **same** enclave



Least privilege enforcement

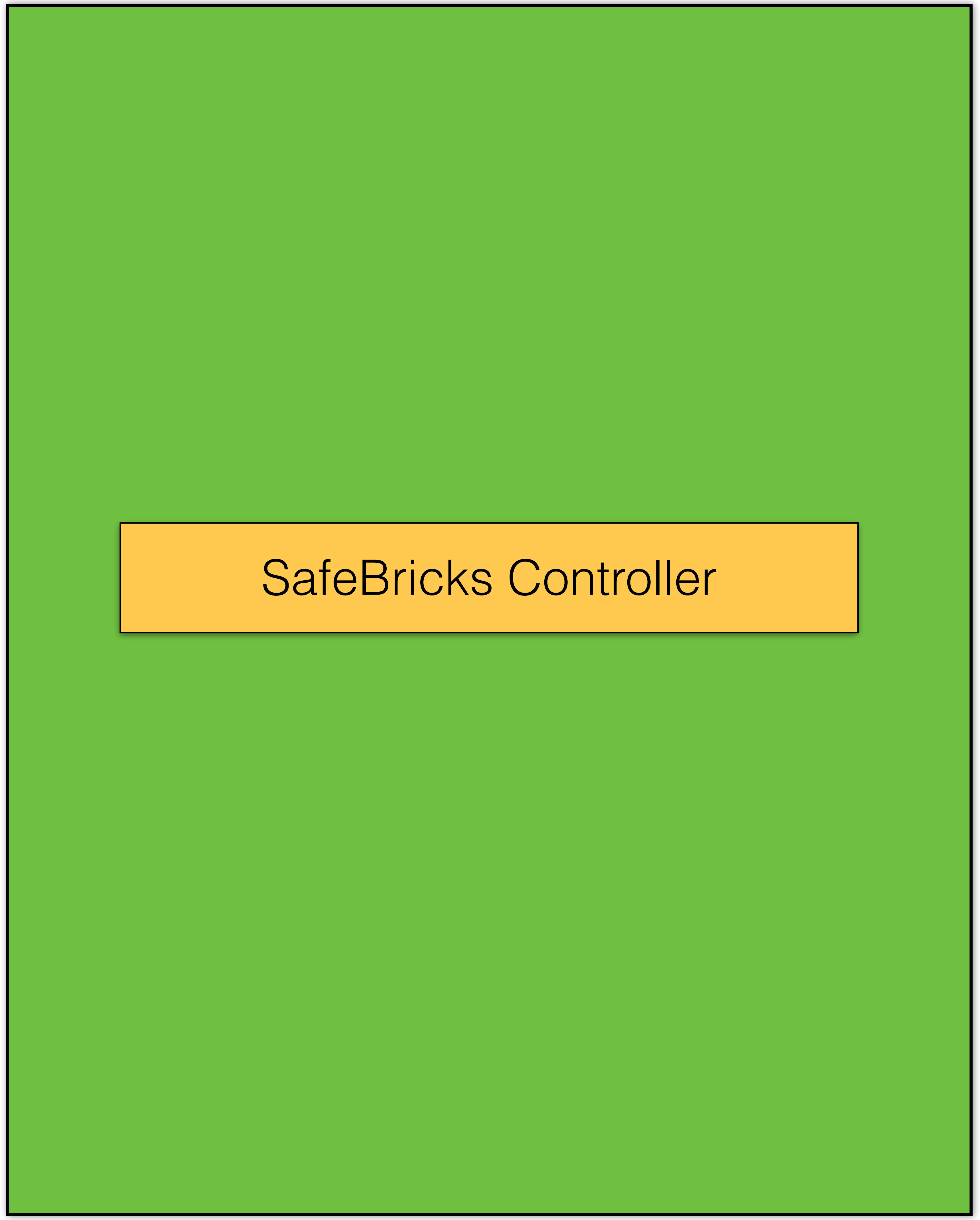
Run NFs within the **same** enclave

- Stitch NFs together interspersed with an operator (wList) that embeds a **vector of permissions** in packets — two bits per packet field



Least privilege enforcement

Enforce permissions by **mediating** access to packets using Rust's **ownership model**

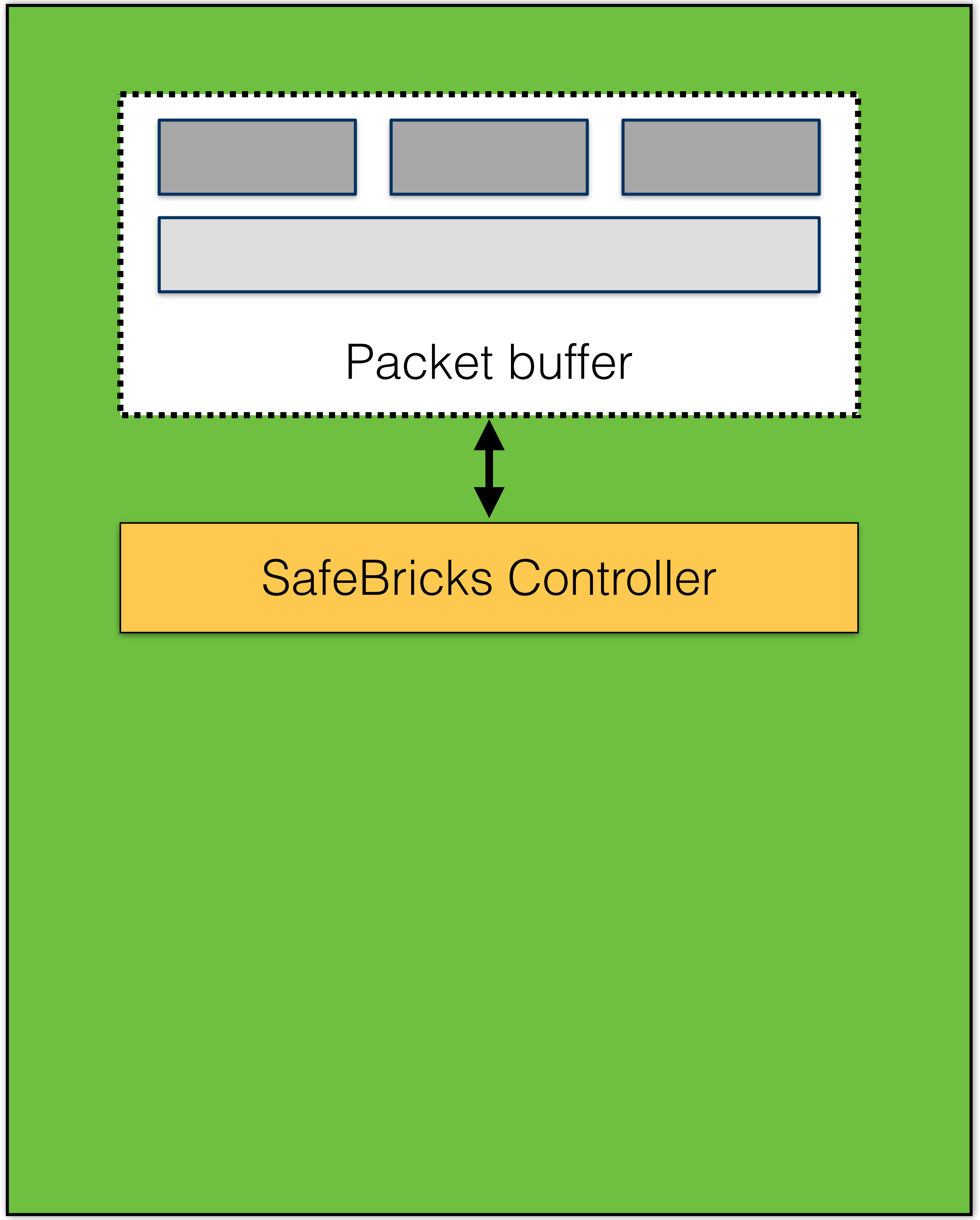


SafeBricks Controller

Least privilege enforcement

Enforce permissions by **mediating** access to packets using Rust's **ownership model**

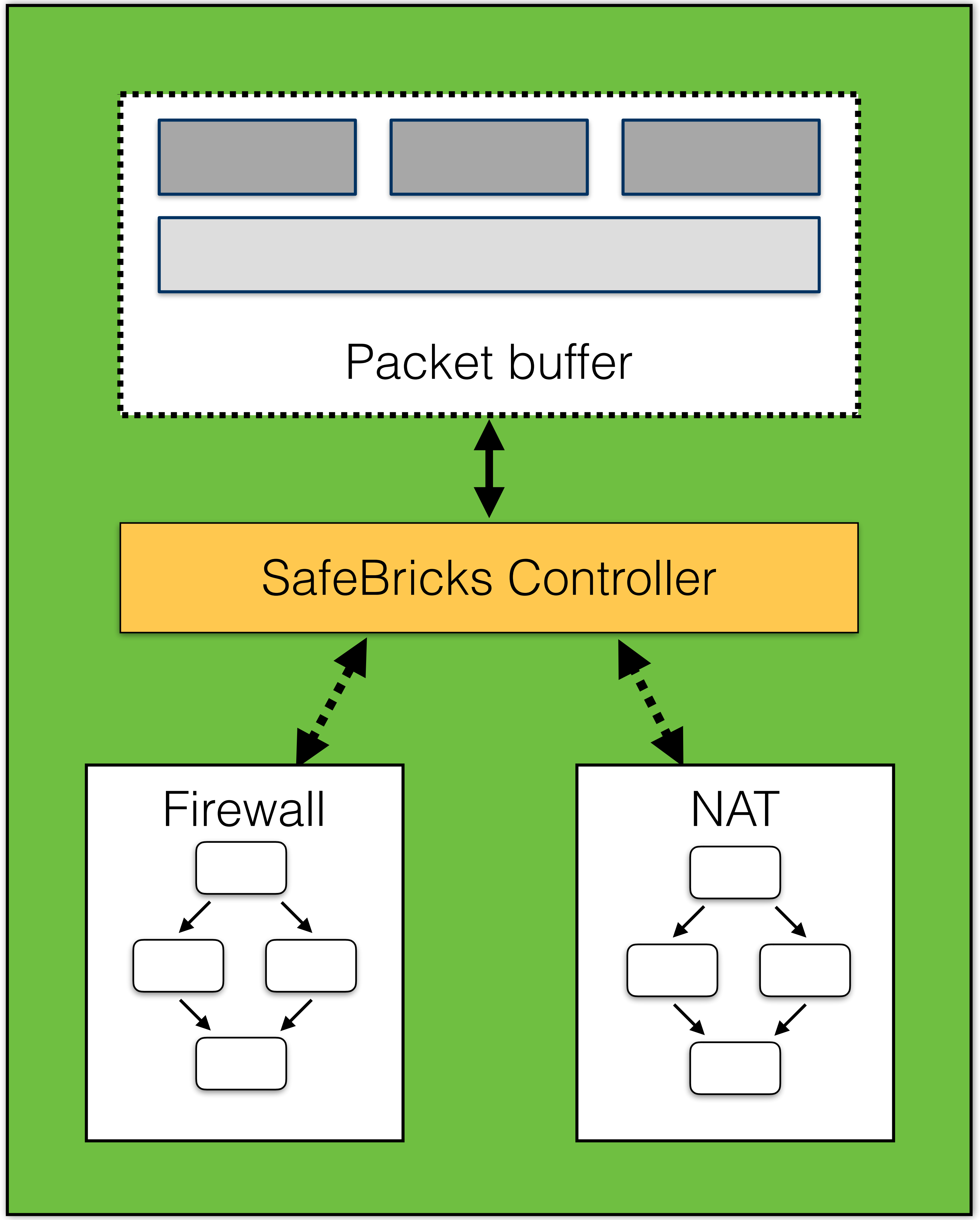
- Controller module holds ownership of packet buffers



Least privilege enforcement

Enforce permissions by **mediating** access to packets using Rust's **ownership model**

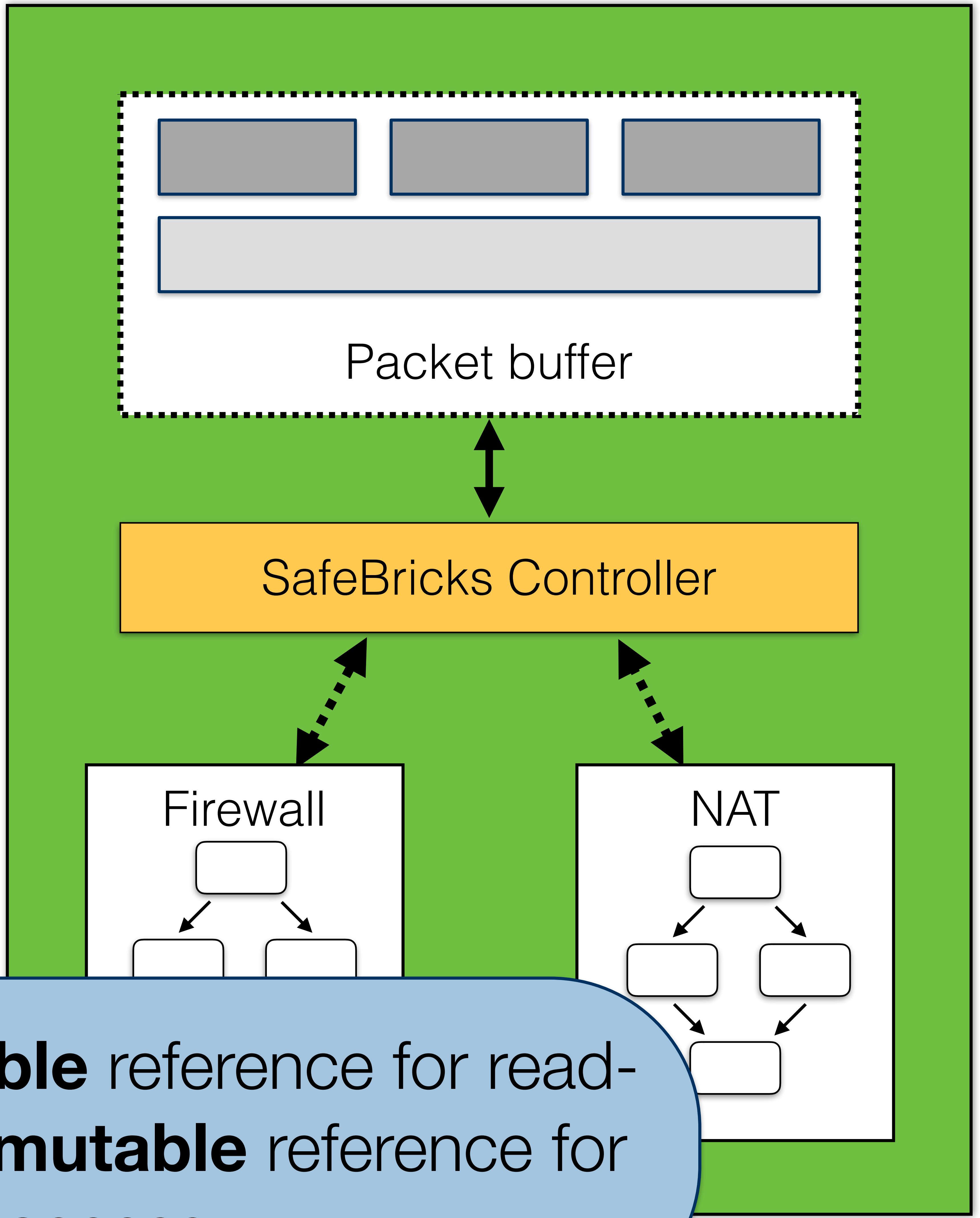
- Controller module holds ownership of packet buffers
- NFs **borrow references** to packet fields from the Controller, which checks permissions vector in packet



Least privilege enforcement

Enforce permissions by **mediating** access to packets using Rust's **ownership model**

- Controller module holds ownership of packet buffers
- NFs **borrow references** to packet fields from the Controller, which checks permissions vector in packet



Returns an **immutable** reference for read-only access, and a **mutable** reference for write access

Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a **compiler that prohibits *unsafe* operations!**

Assumption: Trusted compilation of NFs

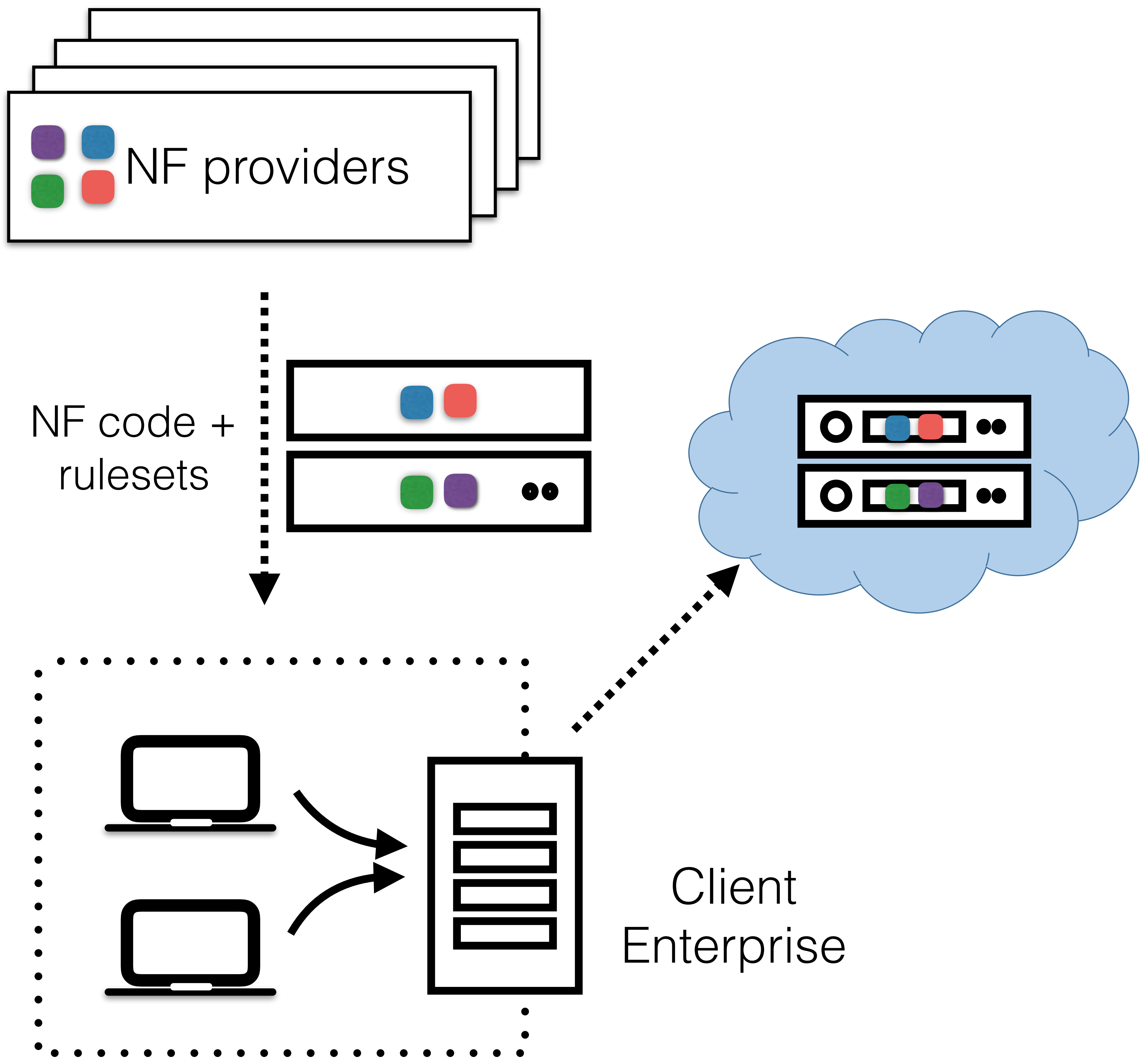
Least privilege guarantees only hold if NFs are built using a **compiler that prohibits *unsafe* operations!**

E.g. Check array bounds, no pointer arithmetic, no unsafe type casts

Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a **compiler that prohibits *unsafe* operations!**

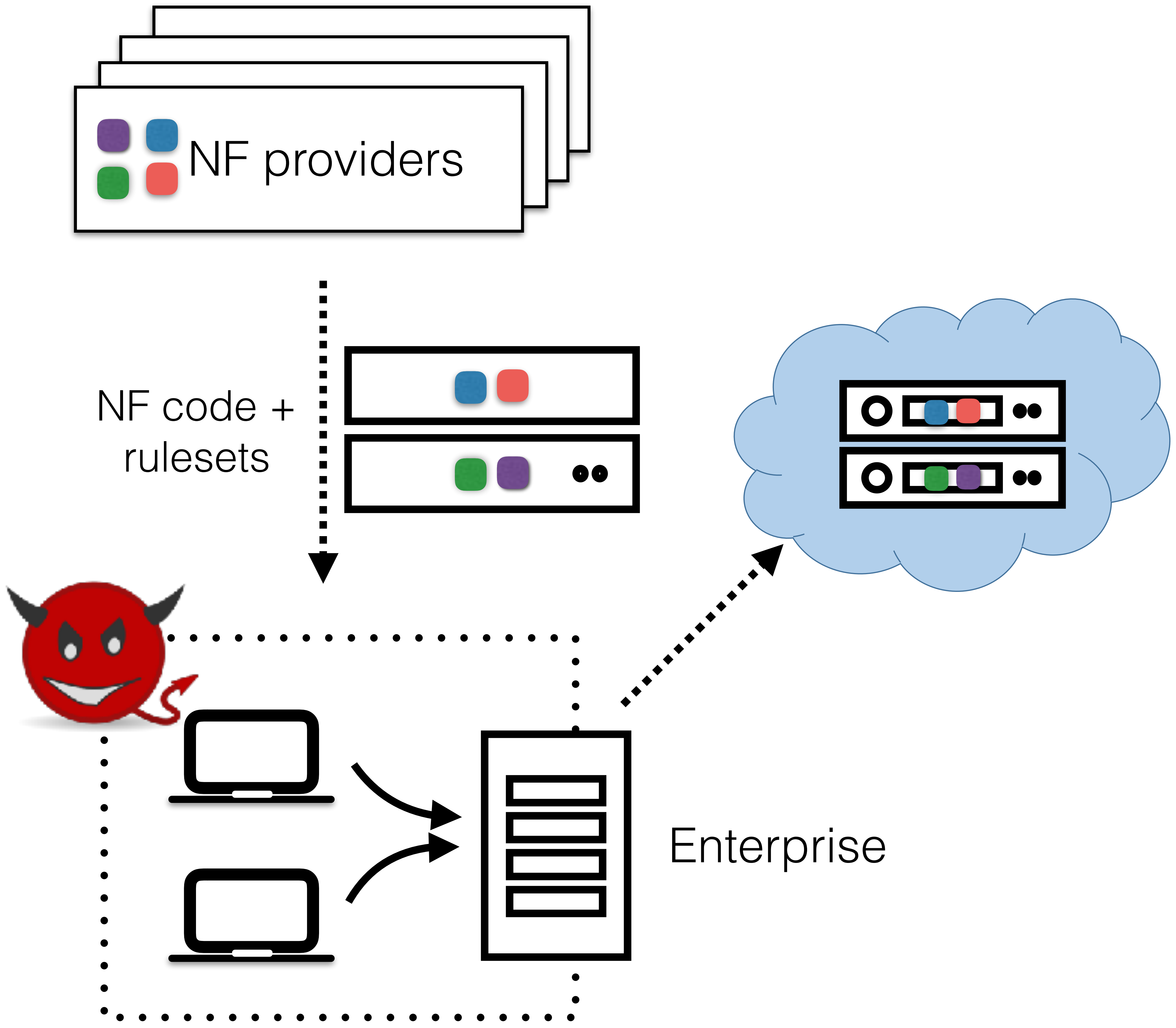
- Possible solution: Client obtains NF source codes from providers and assembles them locally



Assumption: Trusted compilation of NFs

Least privilege guarantees only hold if NFs are built using a **compiler that prohibits *unsafe* operations!**

- Possible solution: Client obtains NF source codes from providers and assembles them locally
- **Problem:** This violates the confidentiality of NF source code!



SafeBricks

1. Protects **traffic** from the **cloud provider**

2. Protects **traffic** from the **NF providers**

3. Protects **NF source code and rulesets** from client enterprise and cloud

Assembling NFs

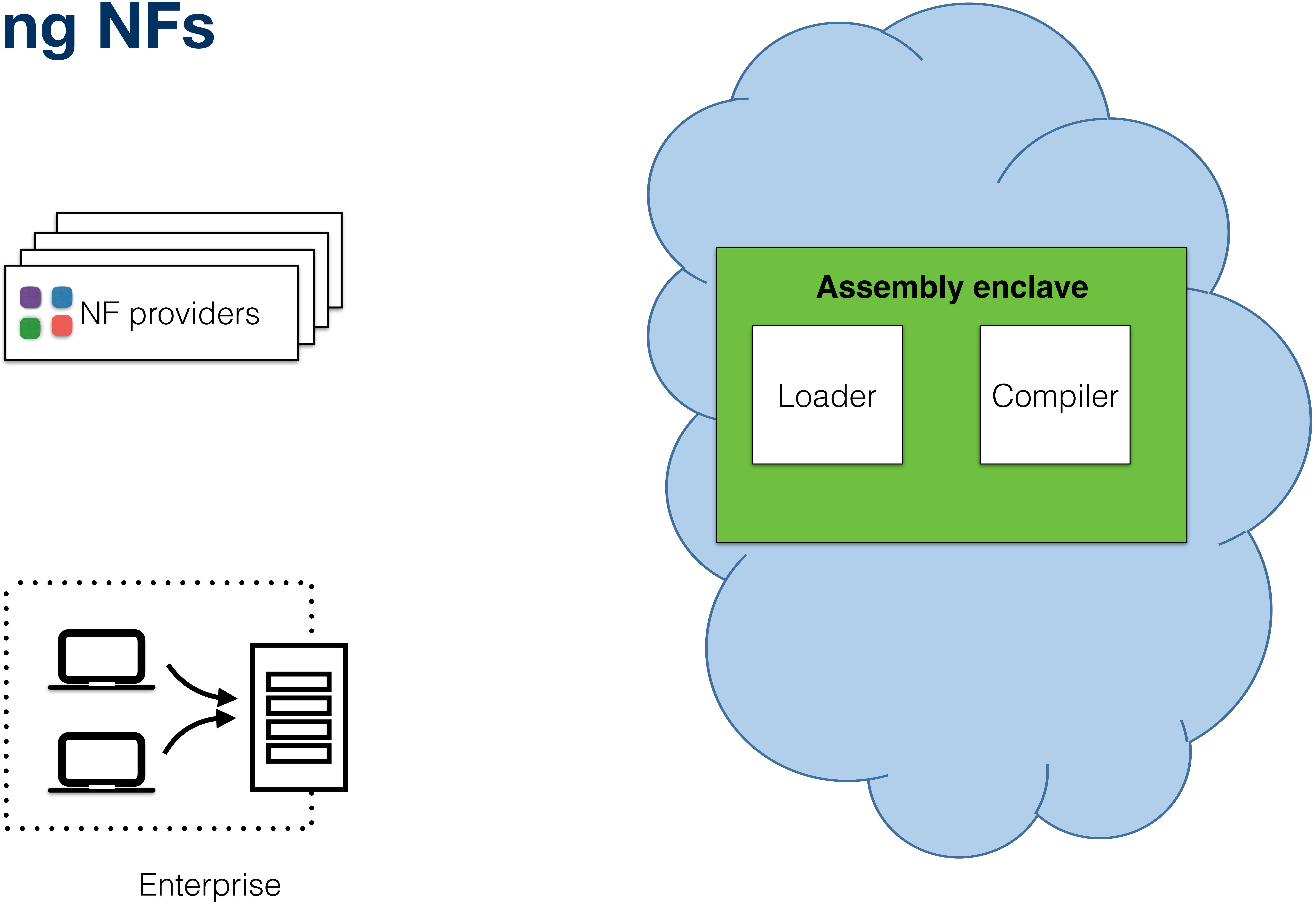
- **Key idea**: Build NFs within a special “meta”-enclave in the cloud using an **agreed upon compiler**

Assembling NFs

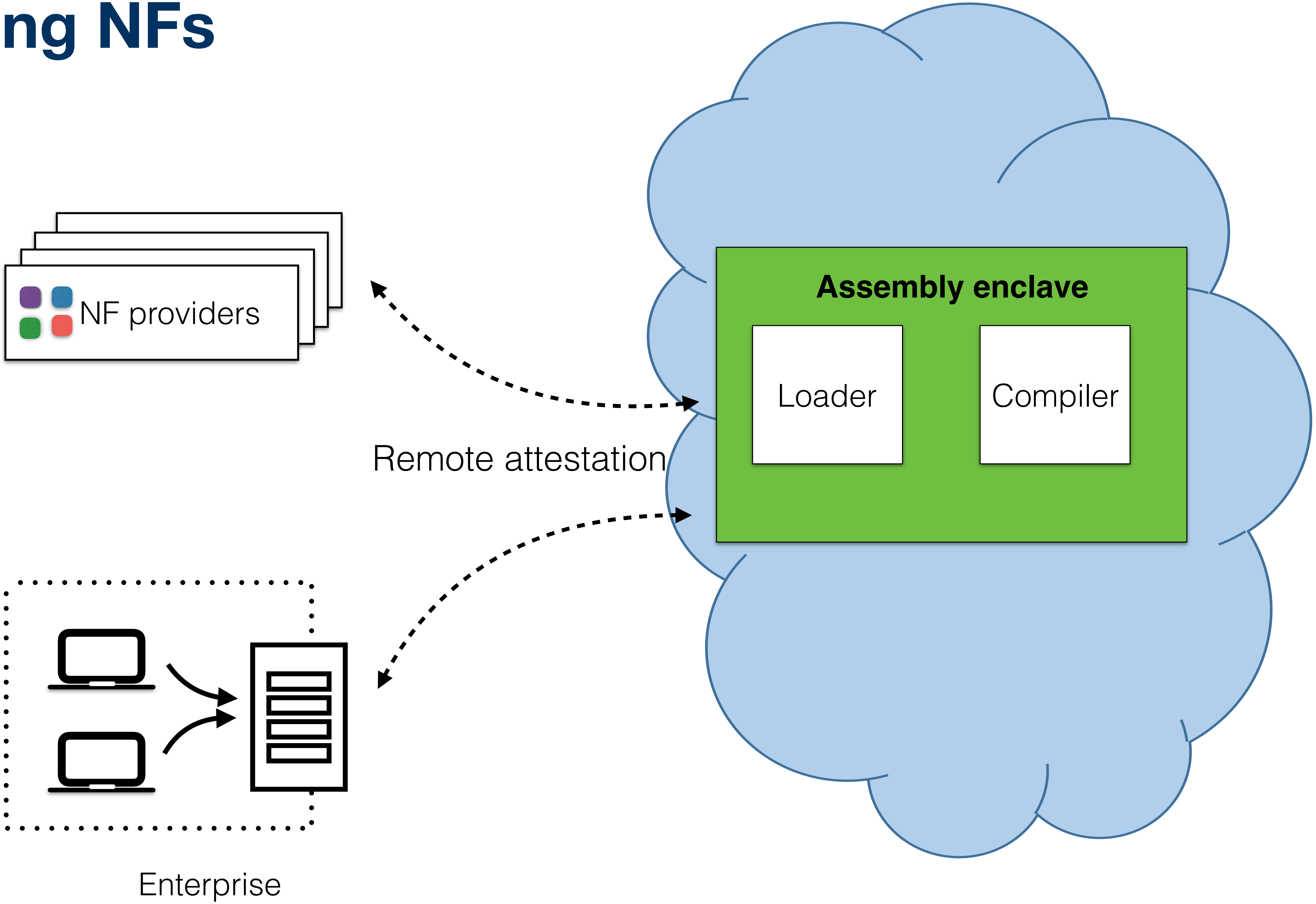
- **Key idea**: Build NFs within a special “meta”-enclave in the cloud using an **agreed upon compiler**

- Both client and NF providers can verify the agreed upon compiler using **remote attestation**

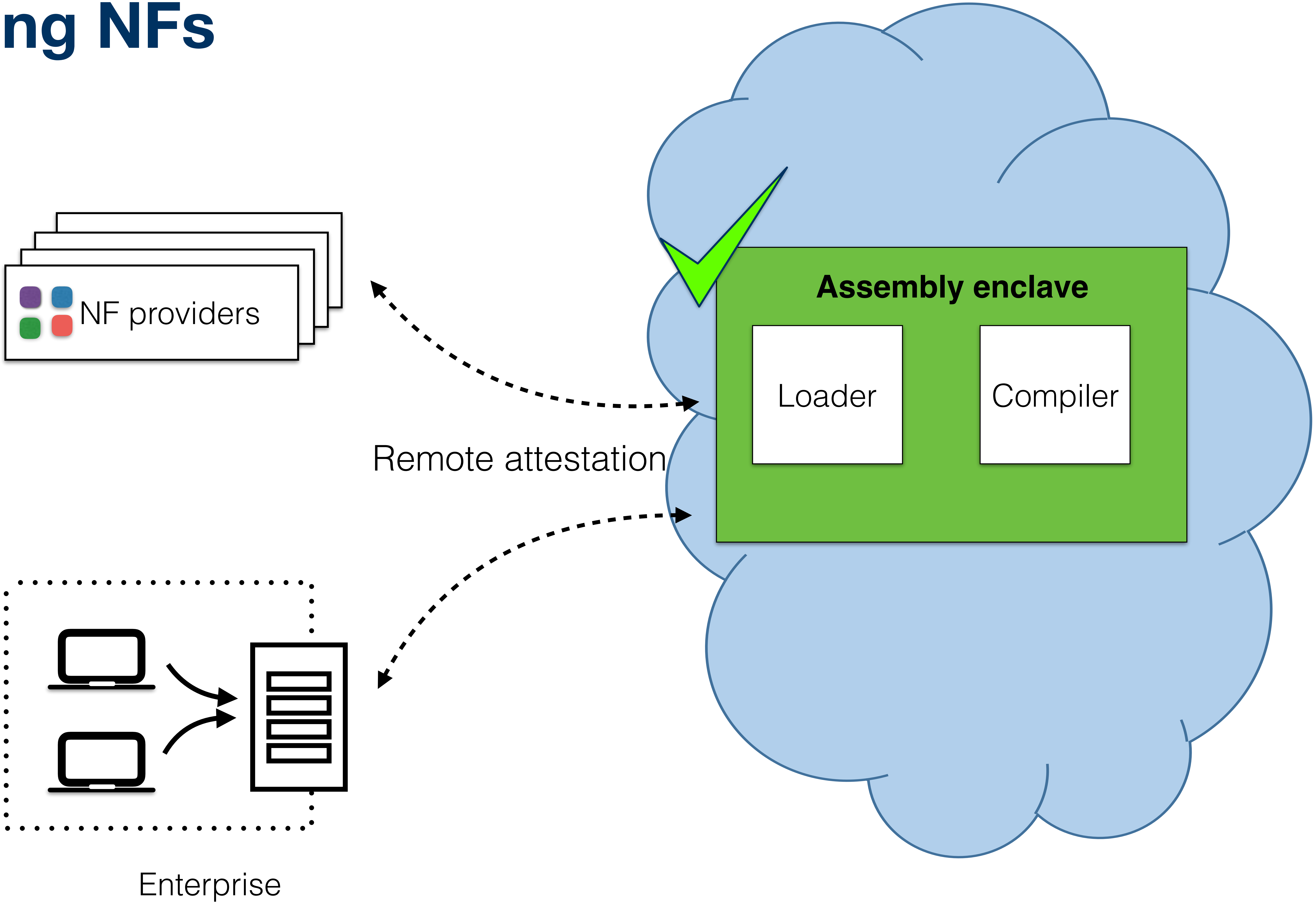
Assembling NFs



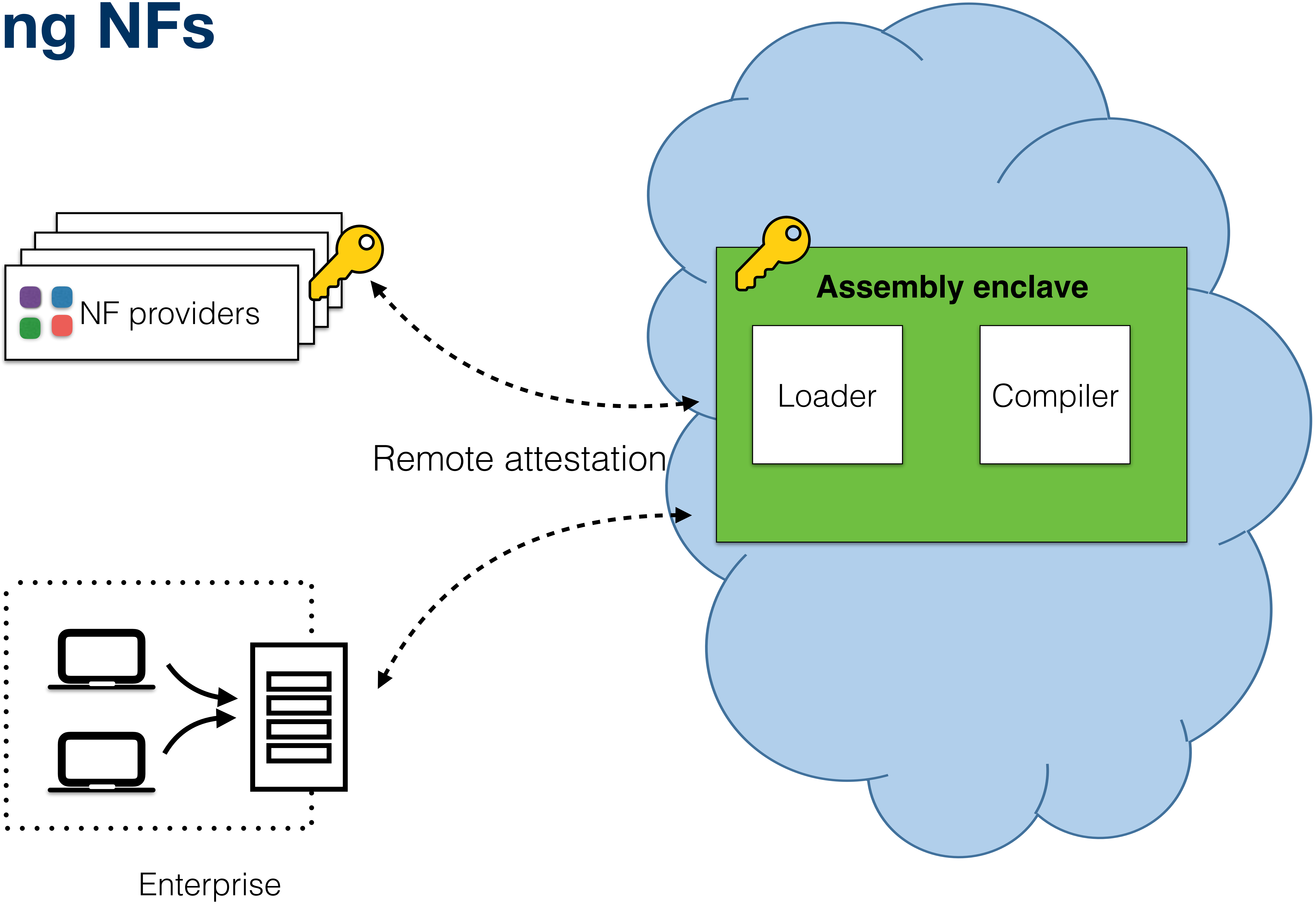
Assembling NFs



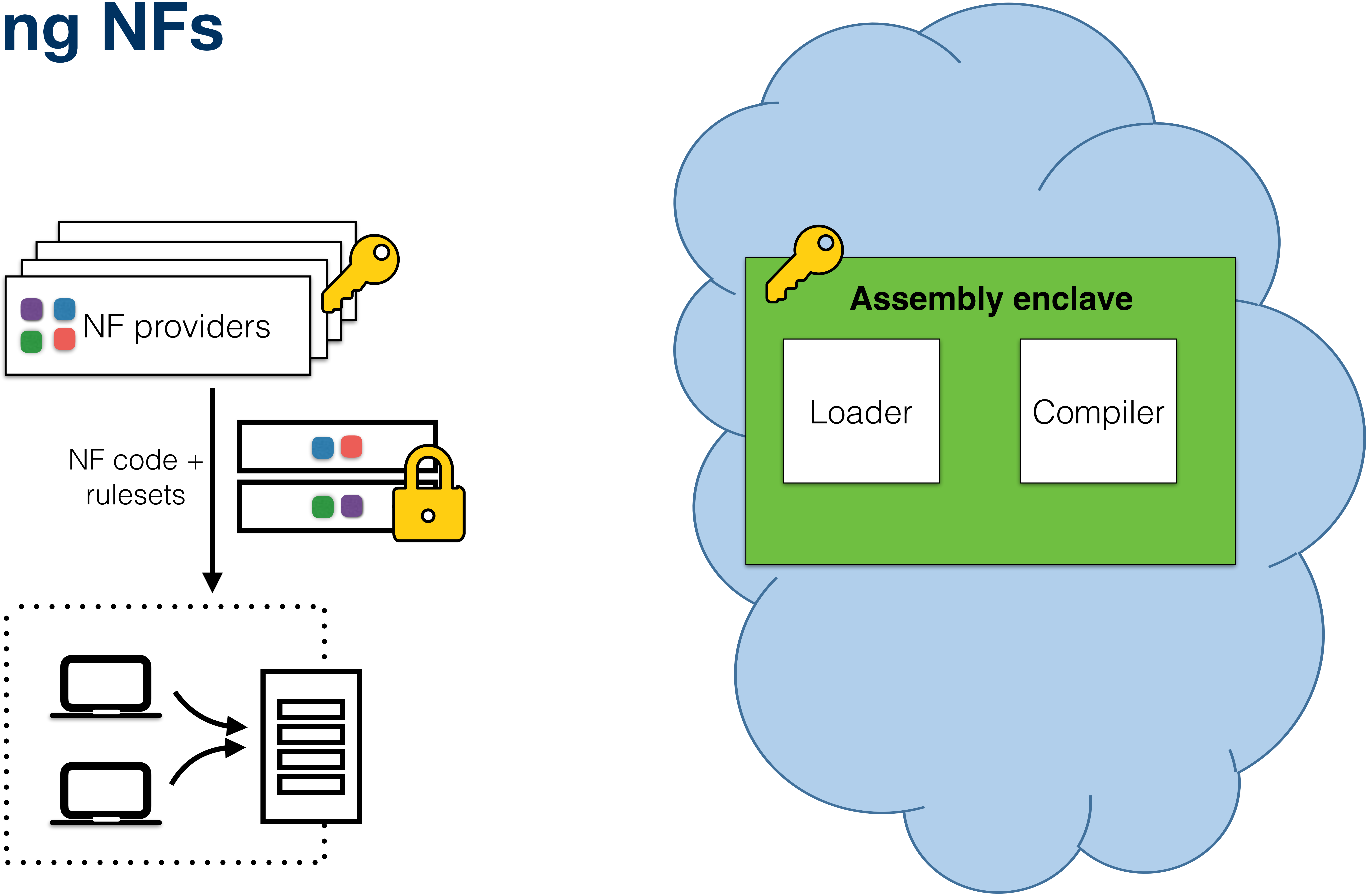
Assembling NFs



Assembling NFs

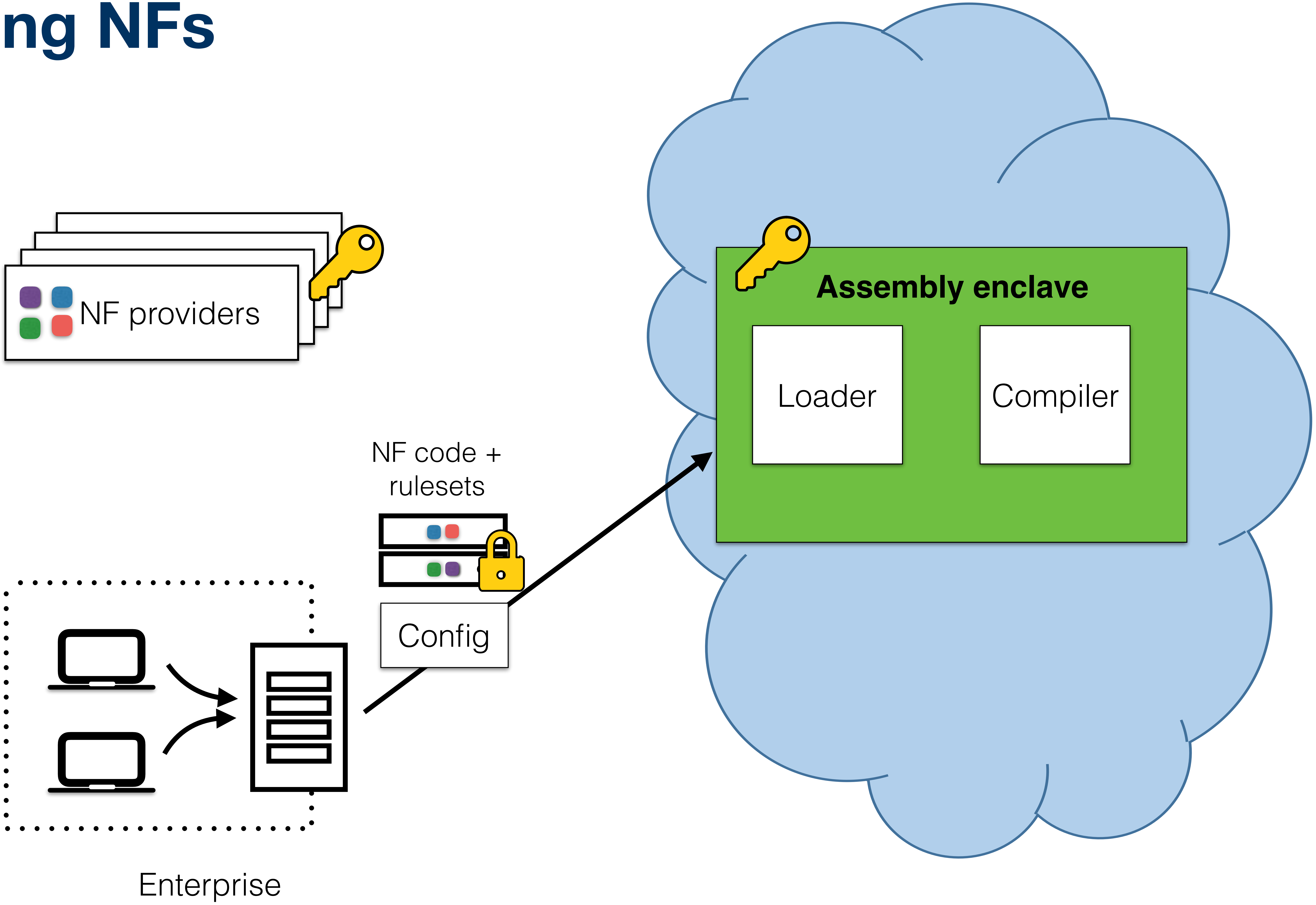


Assembling NFs

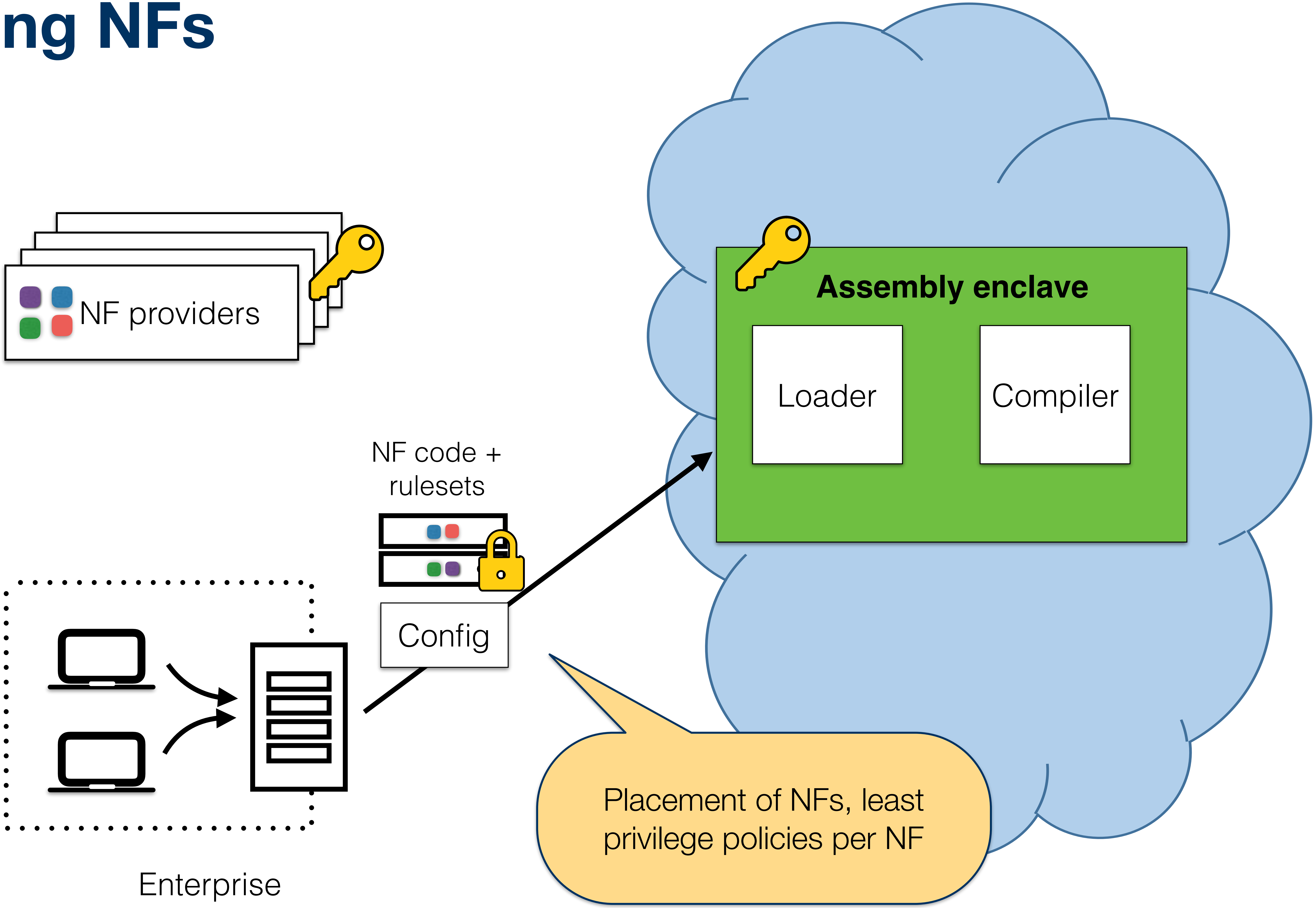


Enterprise

Assembling NFs

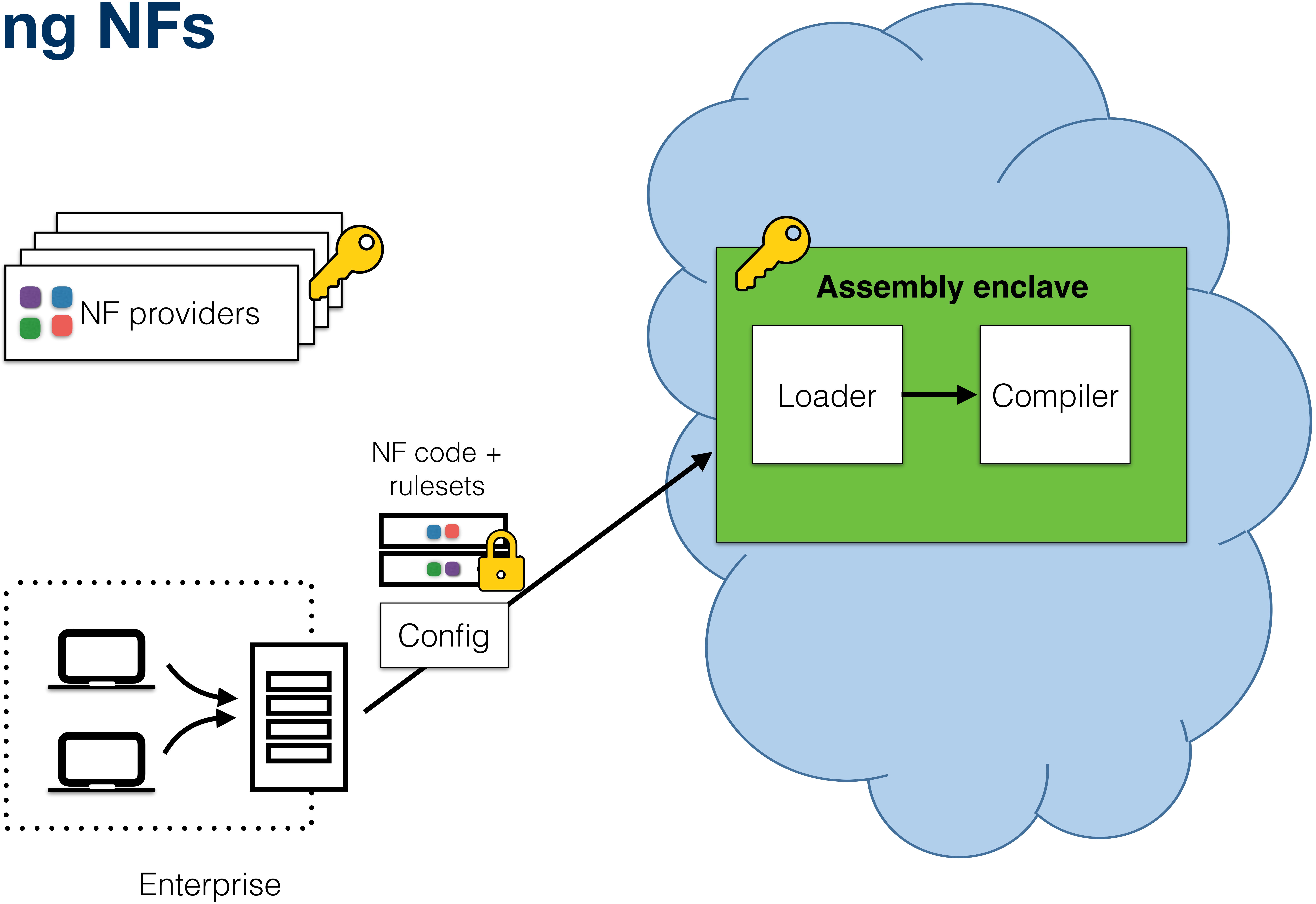


Assembling NFs

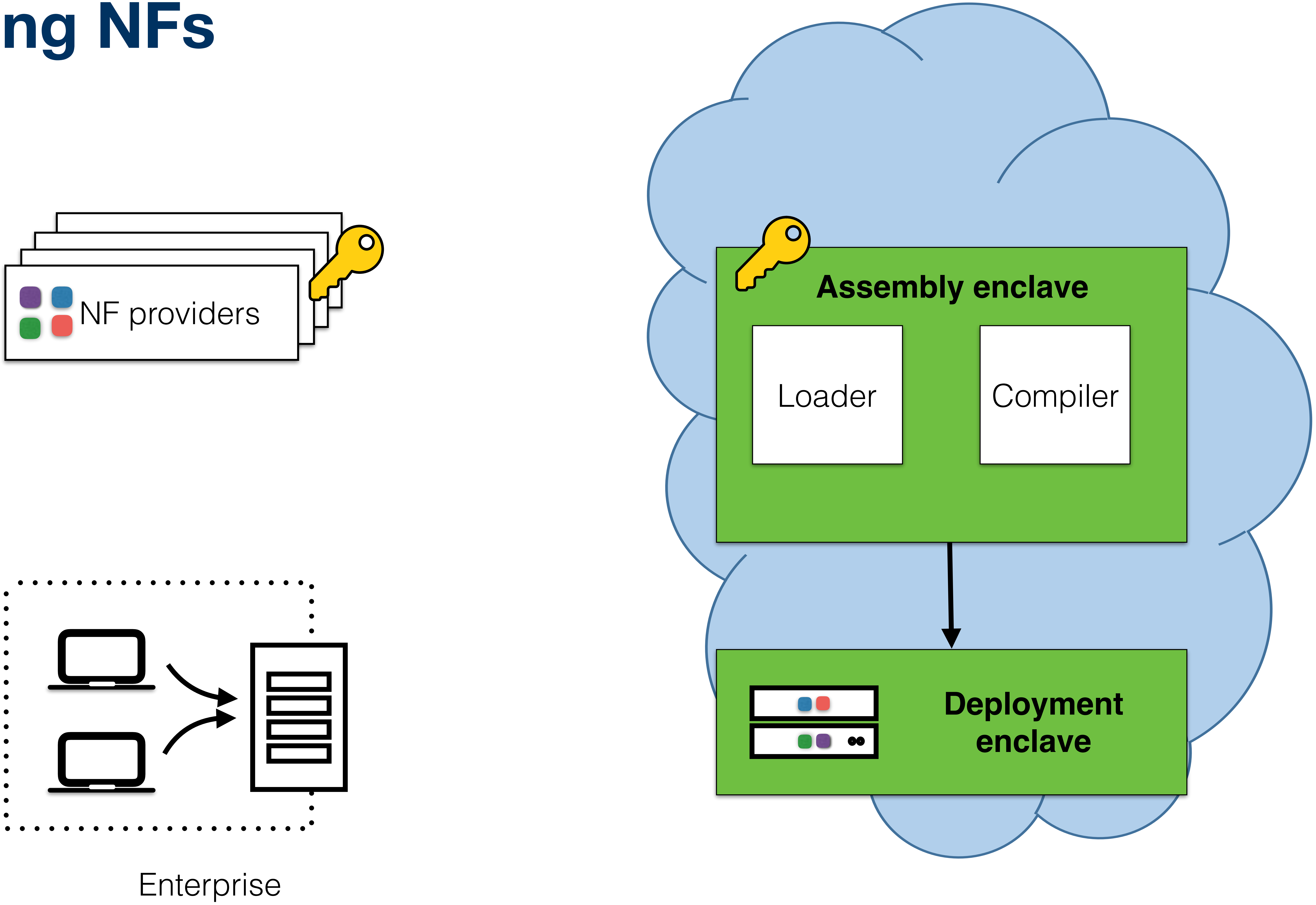


Placement of NFs, least privilege policies per NF

Assembling NFs



Assembling NFs



SafeBricks

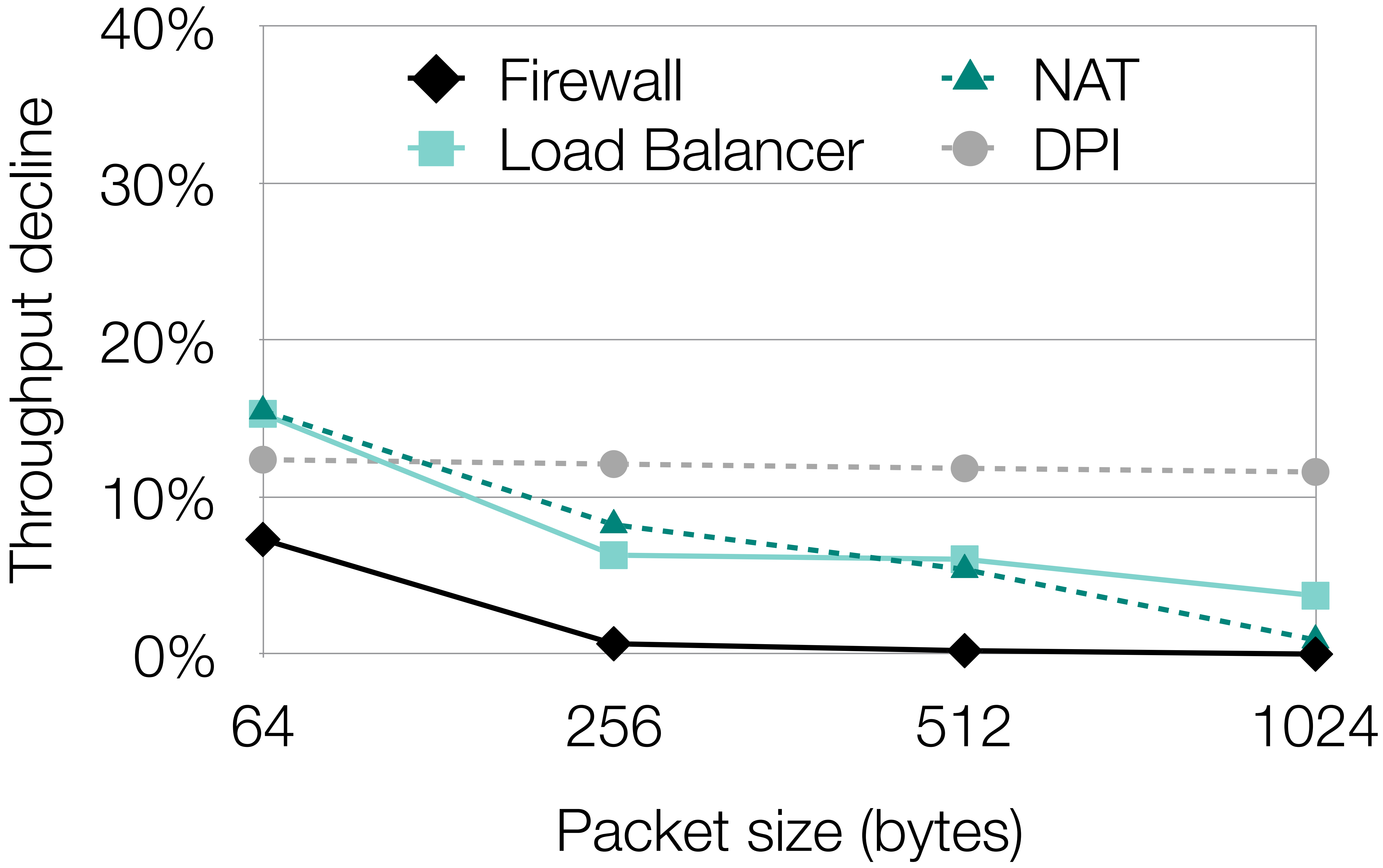
1. Protects **traffic** from the **cloud provider**

2. Protects **traffic** from the **NF providers**

3. Protects **NF source code and rulesets** from client enterprise and cloud

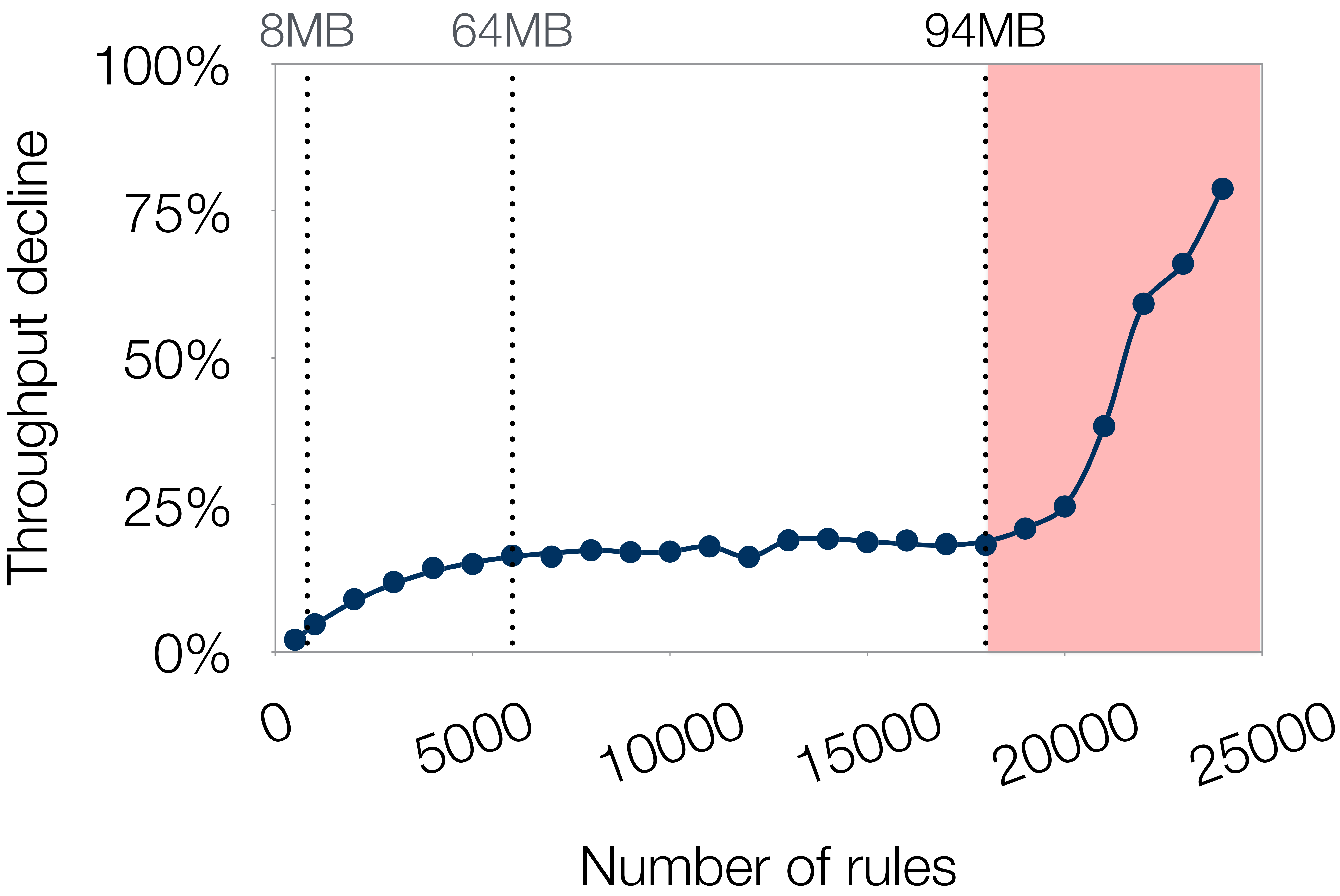
Performance

Throughput decline across NFs



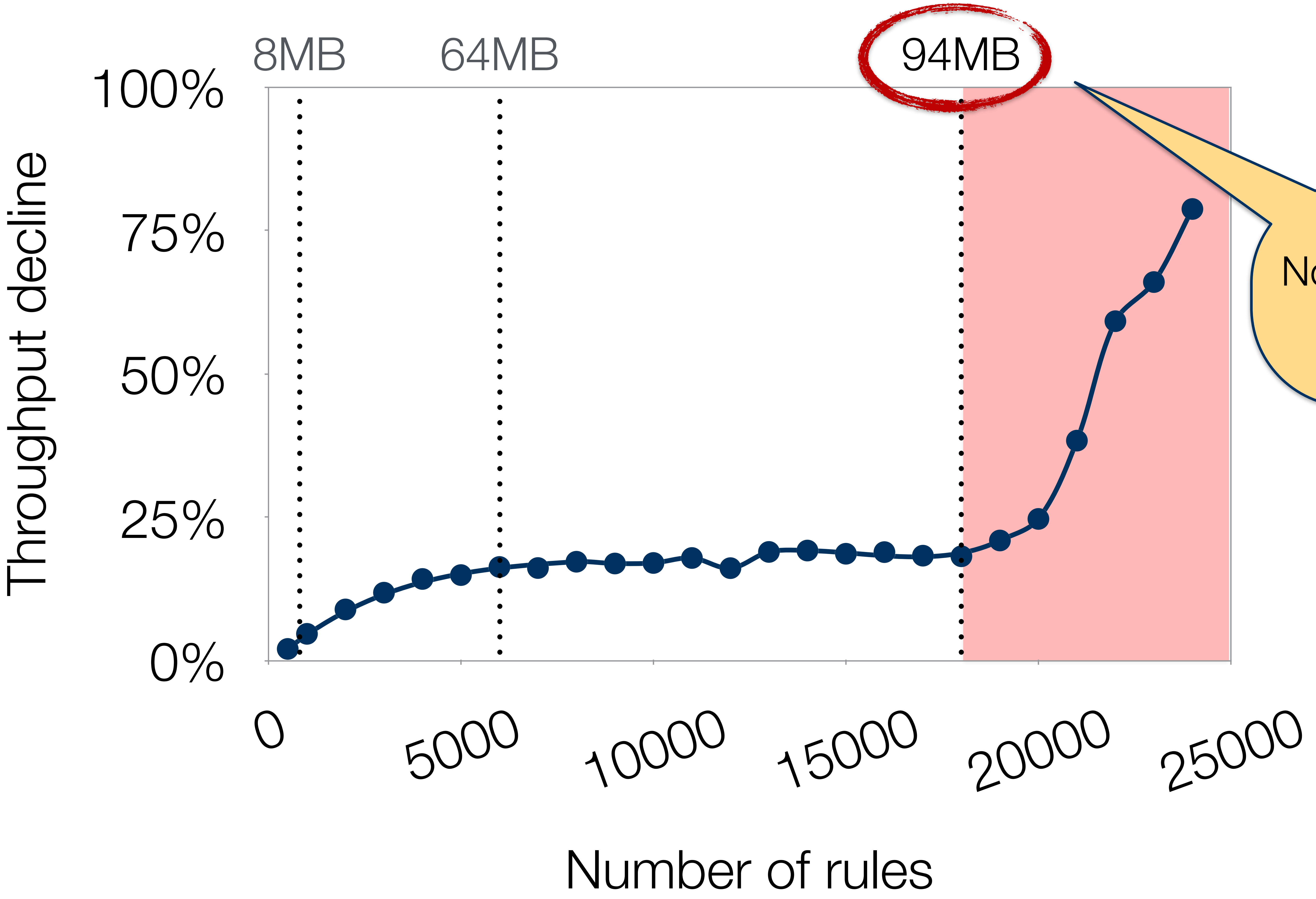
~**0–15% overhead** across applications for different packet sizes

DPI performance with increasing no. of rules



Overhead spikes when NF working set **exceeds enclave memory**

DPI performance with increasing no. of rules



Not a fundamental limitation

Overhead spikes when NF working set **exceeds enclave memory**

Summary

rishabhp@eecs.berkeley.edu

SafeBricks uses a combination of **hardware enclaves** and **language-based isolation** to:

- Protect client traffic from the cloud provider
- Enforce least privilege across NFs
- Protect the confidentiality of NF code and rulesets

Modest overhead across a range of applications