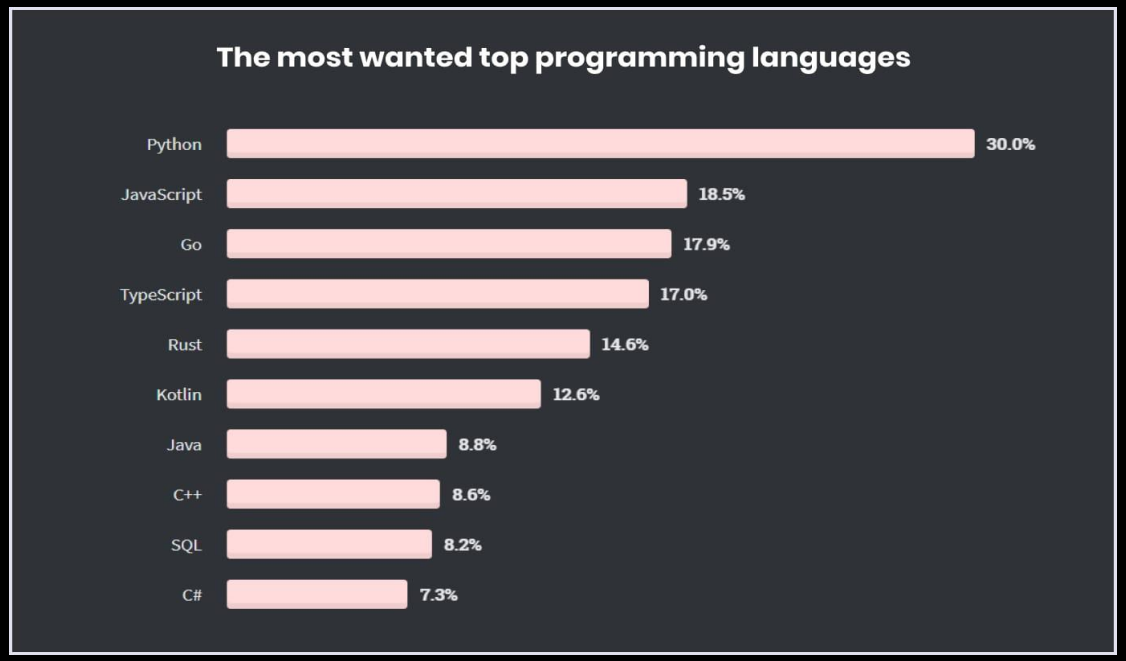
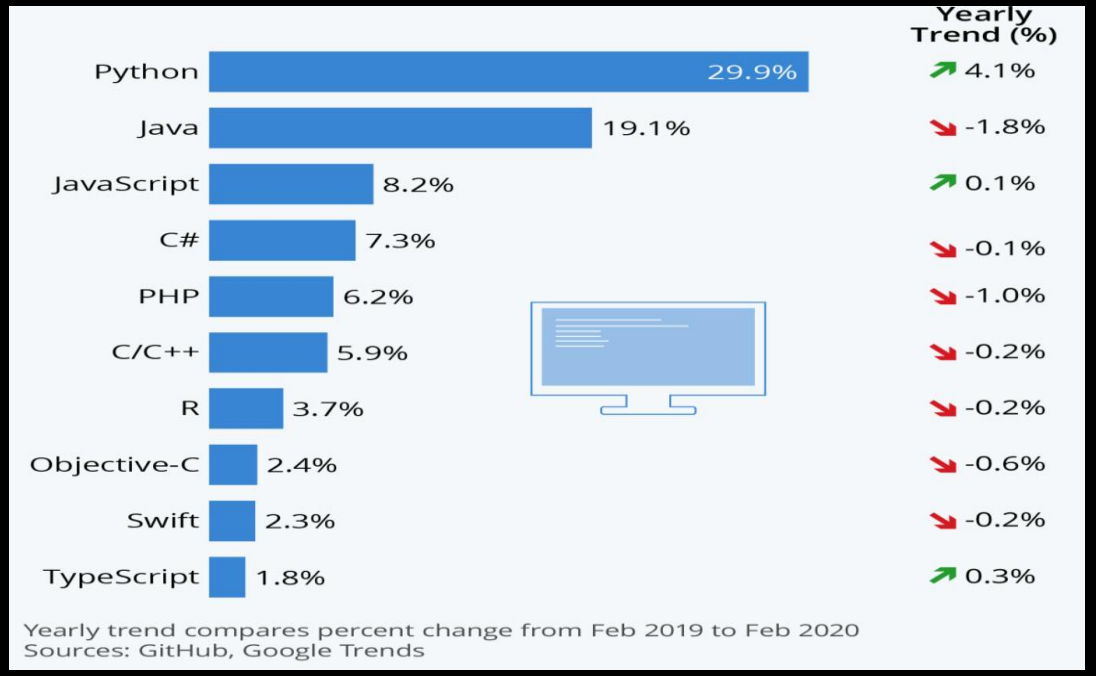
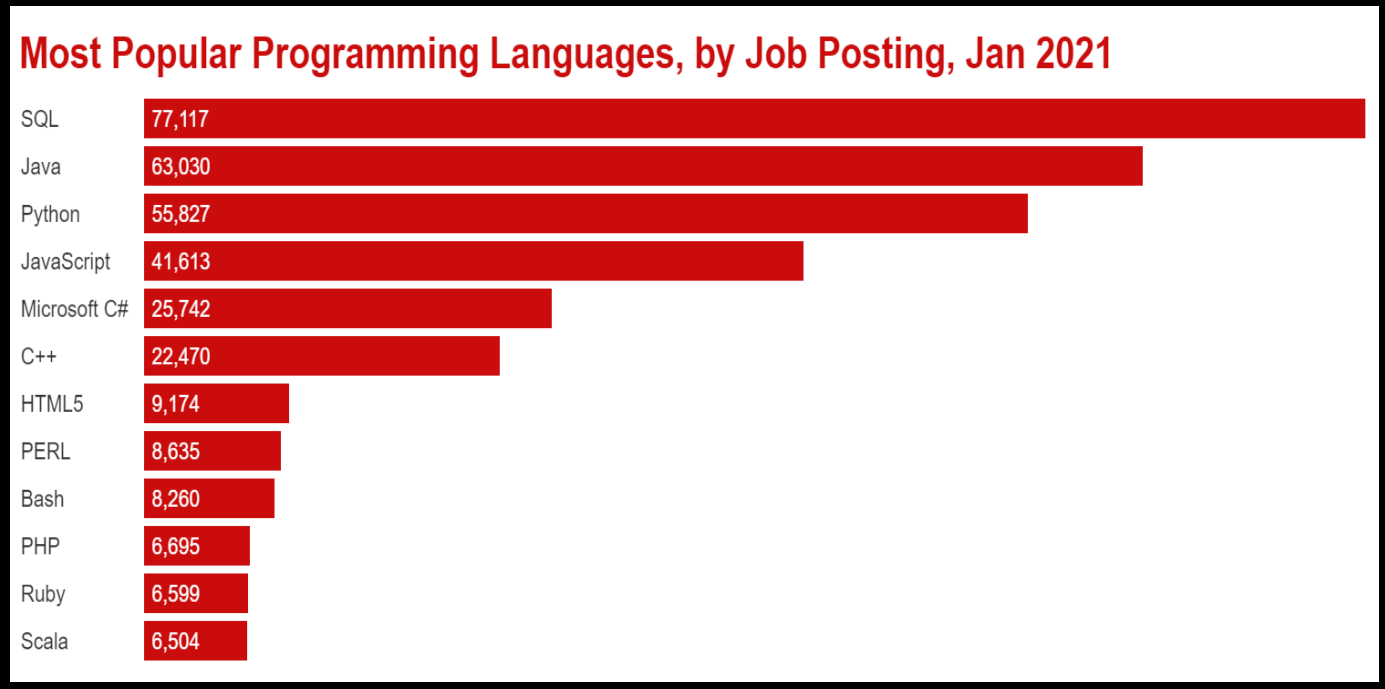
dMVX: Secure and Efficient Multi-Variant Execution in a Distributed Setting

Alexios Voulimeneas, Dokyung Song, Per Larsen, Michael Franz, Stijn Volckaert


The most wanted top programming languages
Python 30.0%
JavaScript 18.5%
Go 17.9%
TypeScript 17.0%
Rust 14.6%
Kotlin 12.6%
Java 8.8%
C++ 8.6%
SQL 8.2%
C# 7.3%


Yearly Trend (%)
Python 29.9% 4.1%
Java 19.1% -1.8%
JavaScript 8.2% 0.1%
C# 7.3% -0.1%
PHP 6.2% -1.0%
C/C++ 5.9% -0.2%
R 3.7% -0.2%
Objective-C 2.4% -0.6%
Swift 2.3% -0.2%
TypeScript 1.8% 0.3%
Yearly trend compares percent change from Feb 2019 to Feb 2020
Sources: GitHub, Google Trends


Most Popular Programming Languages, by Job Posting, Jan 2021
SQL 77,117
Java 63,030
Python 55,827
JavaScript 41,613
Microsoft C# 25,742
C++ 22,470
HTML5 9,174
PERL 8,635
Bash 8,260
PHP 6,695
Ruby 6,599
Scala 6,504

Chromium Project Finds 70% of Its Serious Security Bugs Are Memory Safety Problems

slashdot.org 2020-05-24

Microsoft: 70 percent of all security bugs are memory safety issues

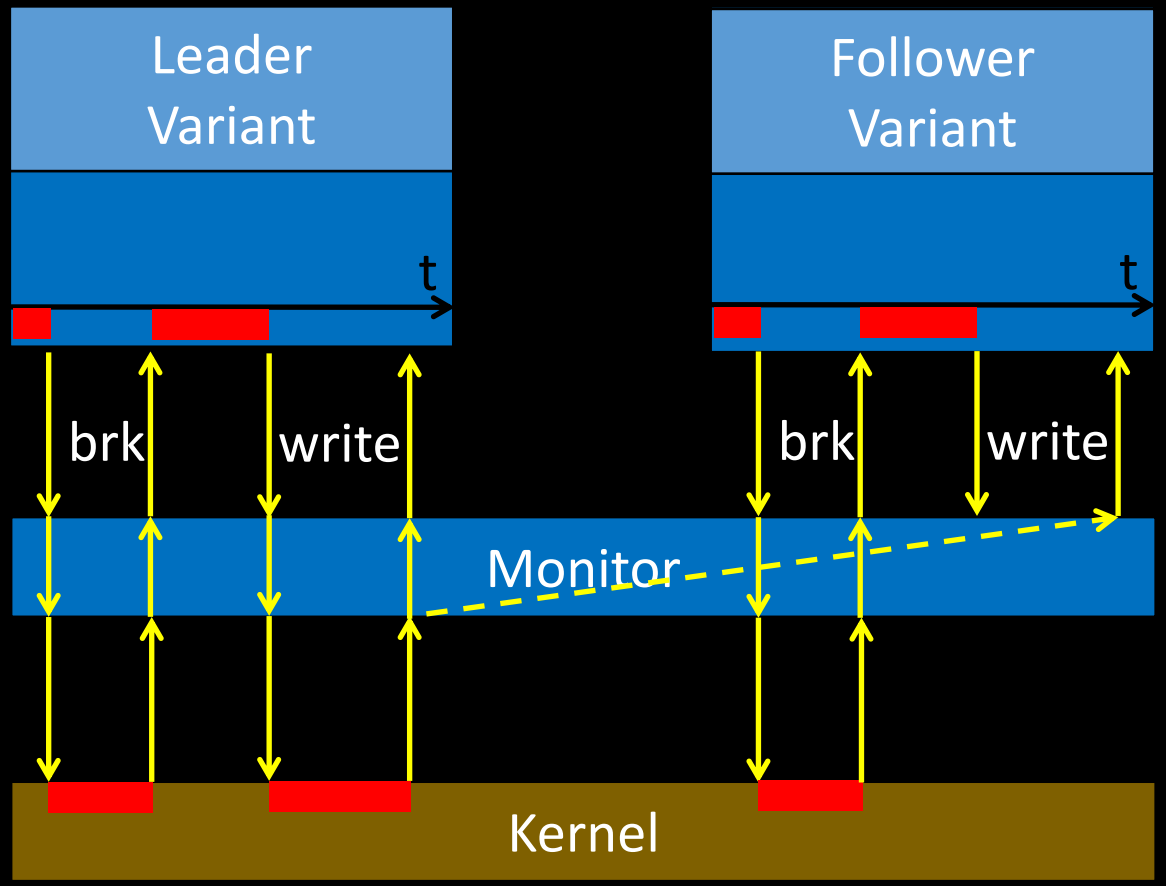
Percentage of memory safety issues has been hovering at 70 percent for the past 12 years.

By Catalin Cimpanu for Zero Day | February 11, 2019 -- 15:48 GMT (07:48 PST) | Topic: Security

Solutions

- Memory-Safe Programming Languages (e.g. Rust)
- Mitigations:
  - Integrity Enforcement (e.g. CFI)
  - Software Diversity (e.g. ASLR)
- Multi-Variant eXecution (MVX)

Multi-Variant eXecution (MVX)



In a nutshell:

- Run multiple diversified program variants in lockstep on identical inputs
- Suspend them at every system call
- Compare system call numbers/arguments
- Replicate I/O results

MVX Systems Security (1)

- ✓Protection against attacks that rely on knowledge of absolute addresses
- ✓Protection against attacks that attempt to acquire knowledge through information leakage

Bruschi et al. “Diversified process replicae for defeating memory error exploits.” In IPCCC, 2007.
Volckaert et al. “Cloning your gadgets: Complete ROP attack immunity with multi-variant execution.” In TDSC, 2012.
Lu et al. “Stopping memory disclosures via diversification and replicated execution.” In TDSC, 2018.

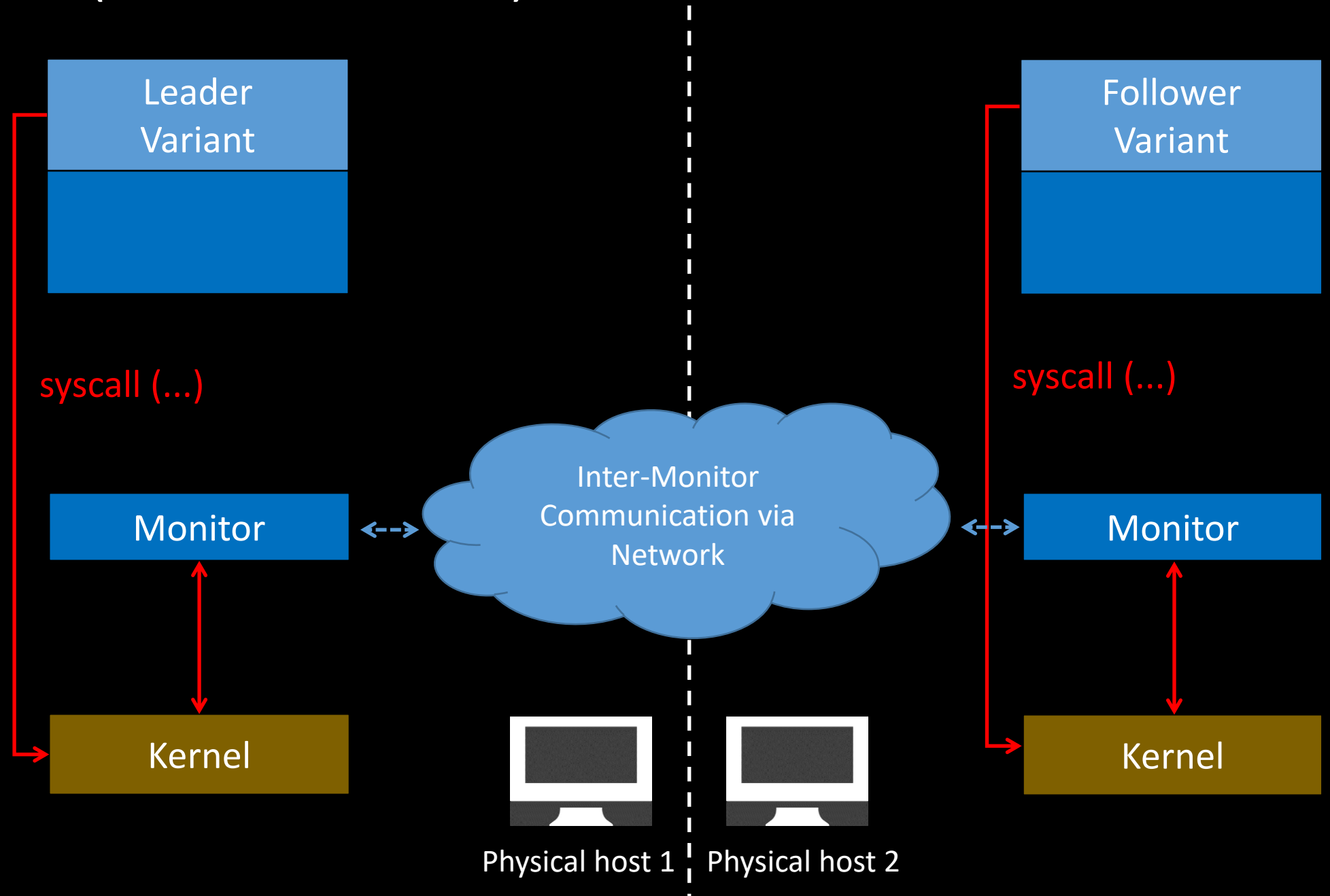
MVX Systems Security (2)

- ✘ Vulnerable to attacks that use relative memory locations
- ✘ Data-only attacks are still possible

Hu et al. “Data-oriented programming: On the expressiveness of non-control data attacks.” In S&P, 2016.
Göktas et al. “Position-independent code reuse: On the eectiveness of ASLR in the absence of information disclosure.” In EuroS&P, 2018.

Observation: Diversity is limited to what a single platform can offer.

DMON (DIMVA 2020)



Distributed Heterogeneous N-Variant Execution

- Variants run on different physical machines
- Leverage ISA and ABI heterogeneity to increase diversity

Additional Diversity

ISA-Heterogeneity

- Machine instructions
- Endianness
- Register set
- Pointer width
- Available system calls

ABI-Heterogeneity

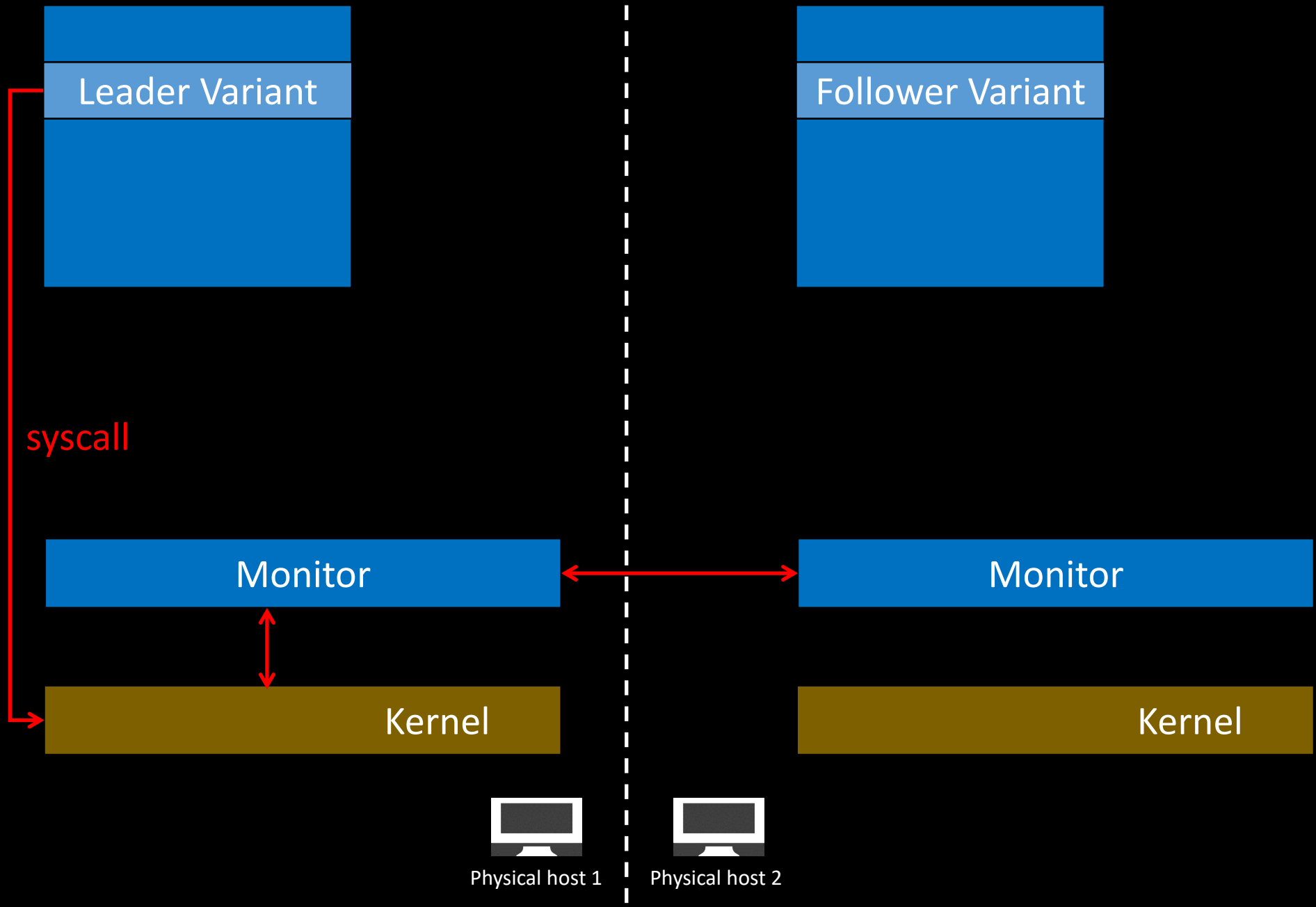
- Size of primitive data types
- Structs layout
  - Packing
  - Alignment
  - Padding
- Constants
  - System call numbers
  - Flags and modes
- Calling conventions

Performance (?)

System Call Interception

Monitoring and Replication

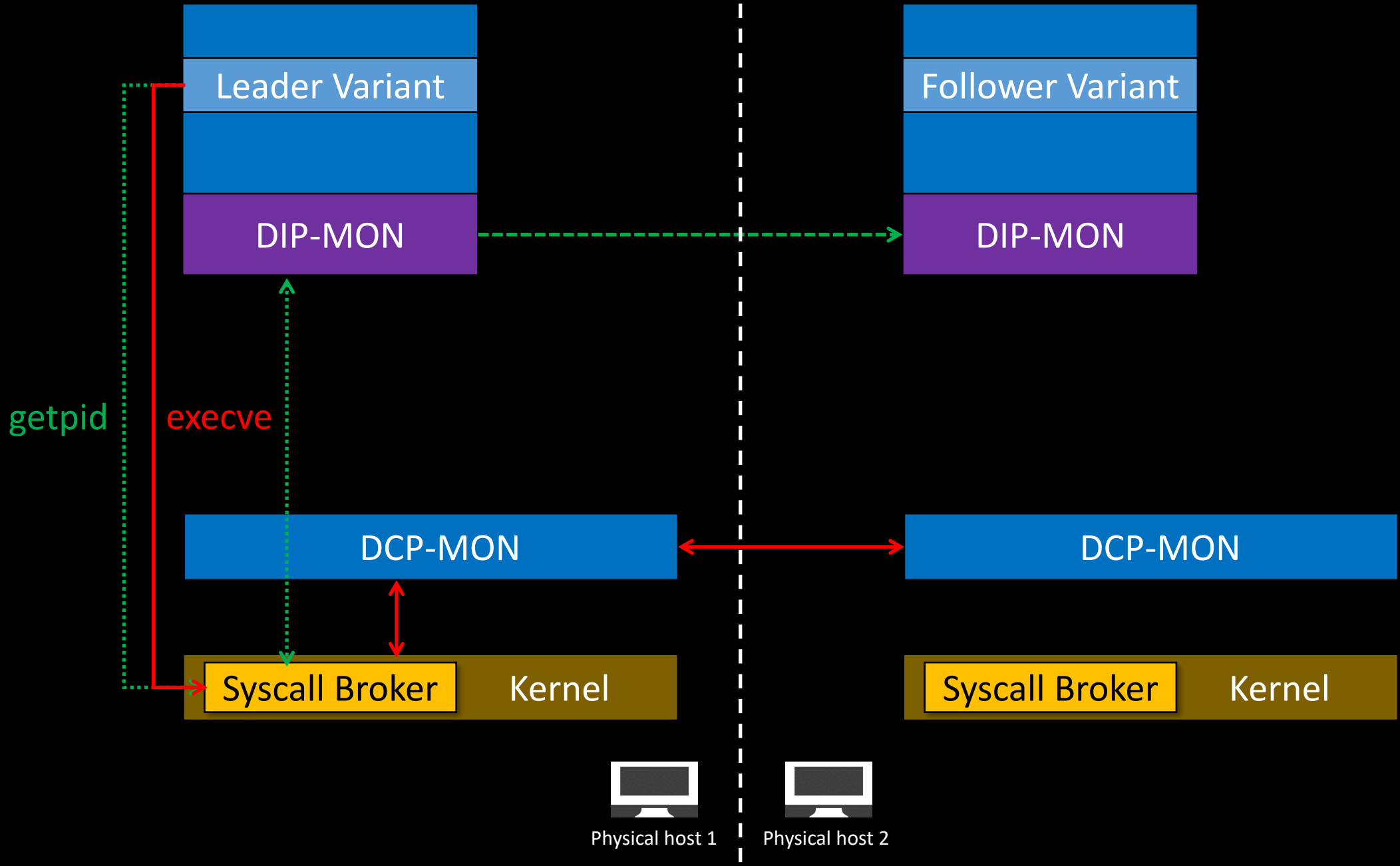
DMON (DIMVA 2020)



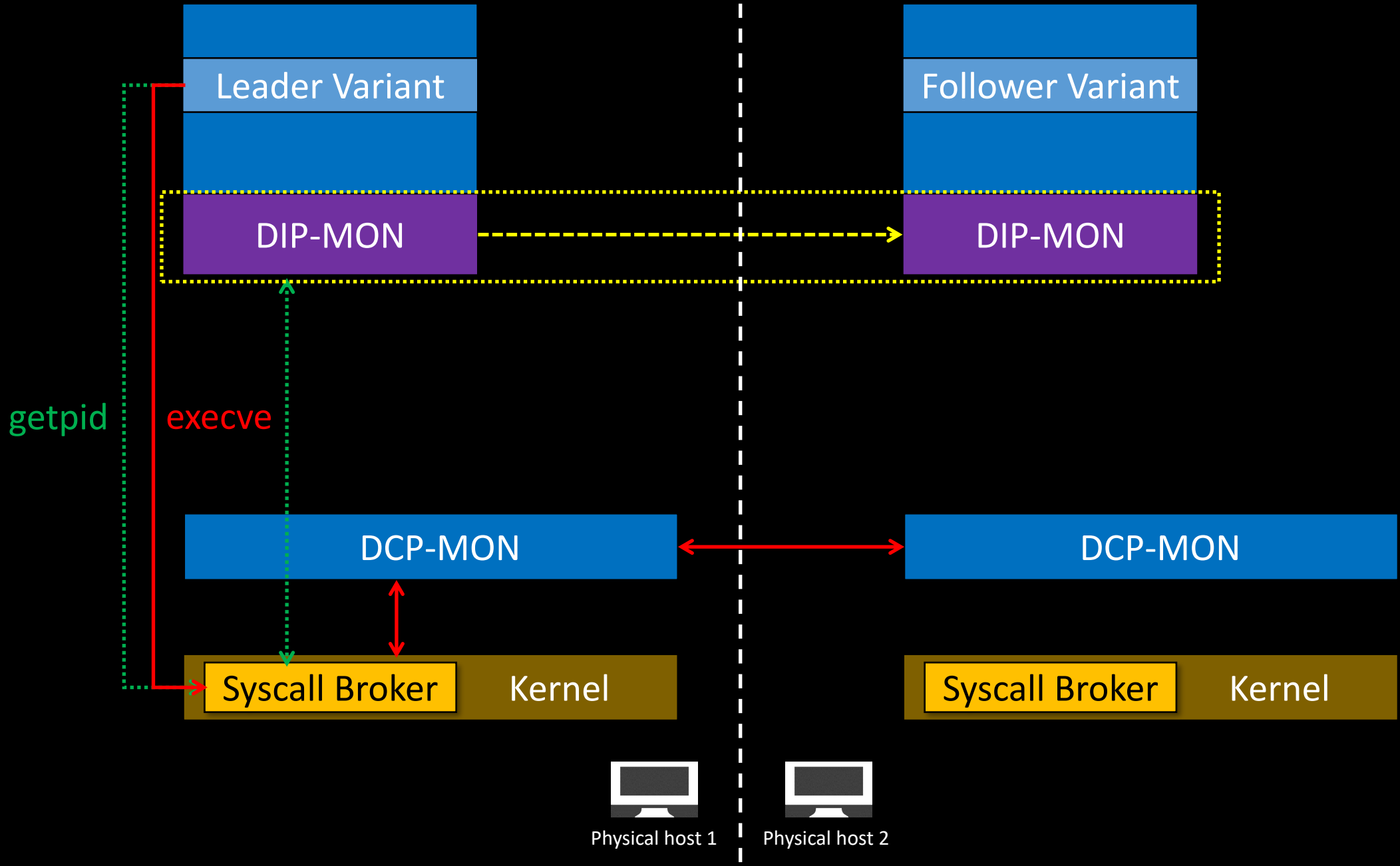
ReMon (ATC 2016)

- Hybrid MVX design
  - Cross-process monitor (CP-MON)
  - In-process monitor (IP-MON)
- Classification of system calls
- CP-MON handles security-sensitive system calls (e.g. execve)
- IP-MON handles non-sensitive system calls (e.g., getpid)

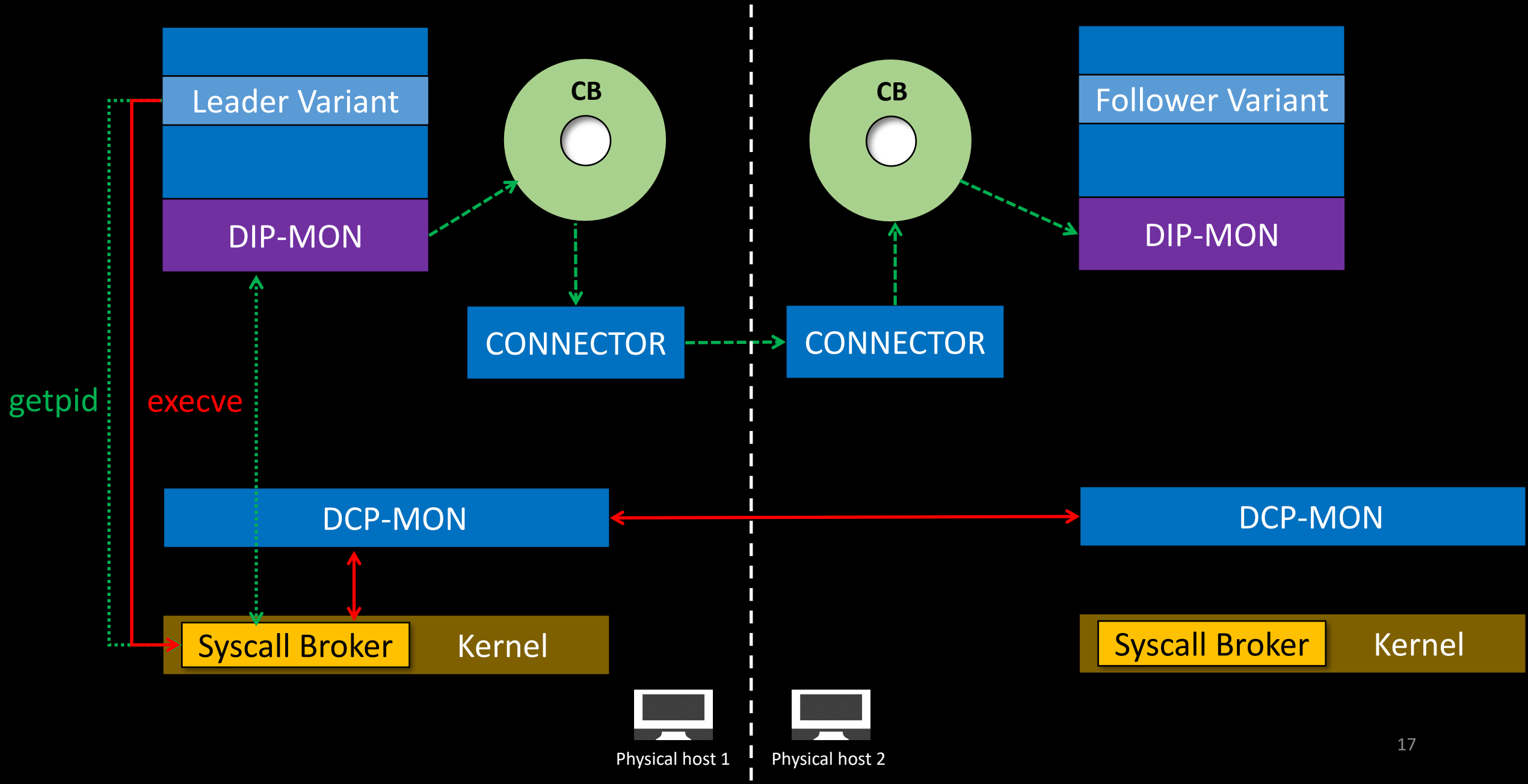
Distributed Hybrid Design



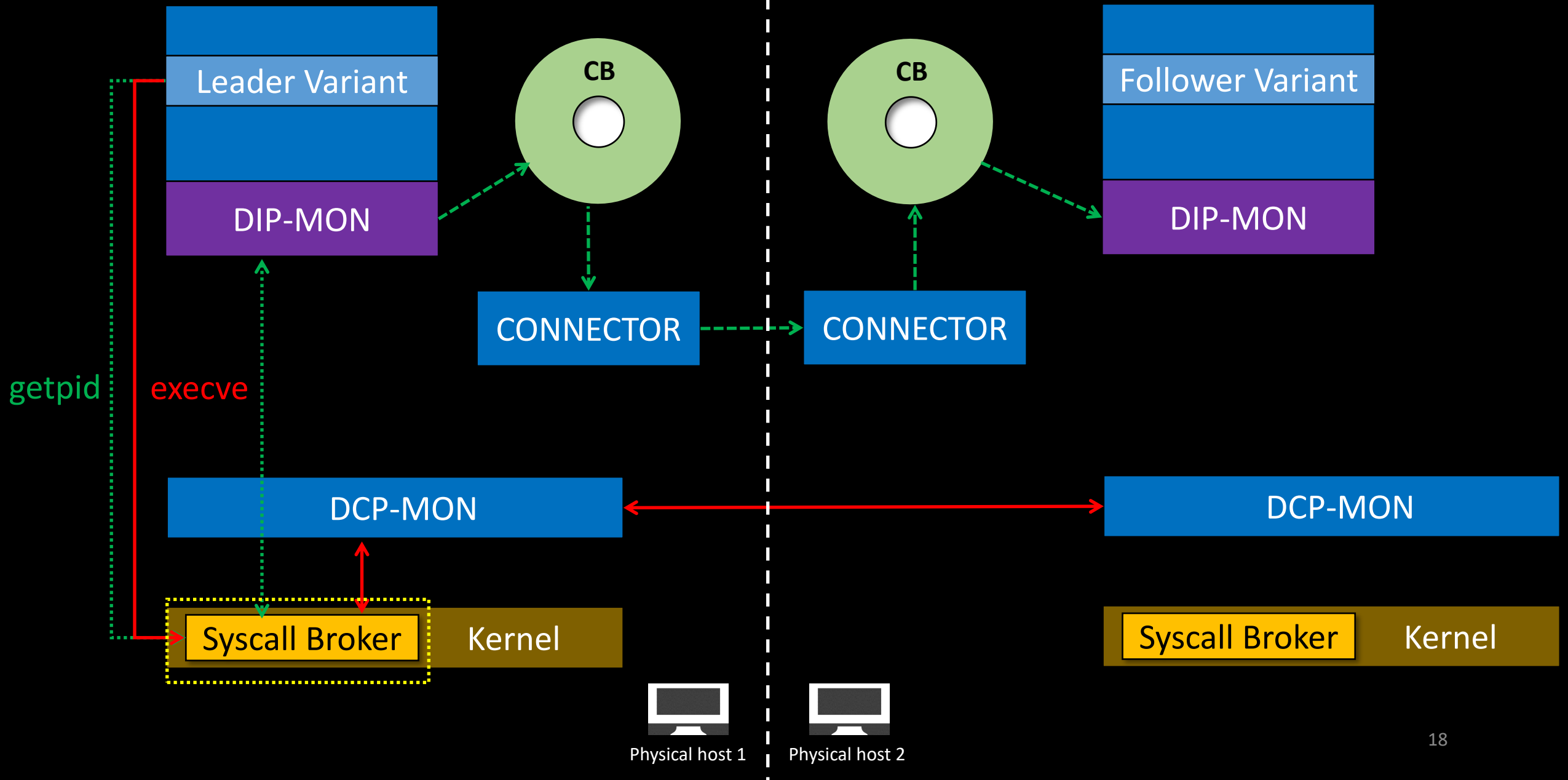
Distributed Hybrid Design



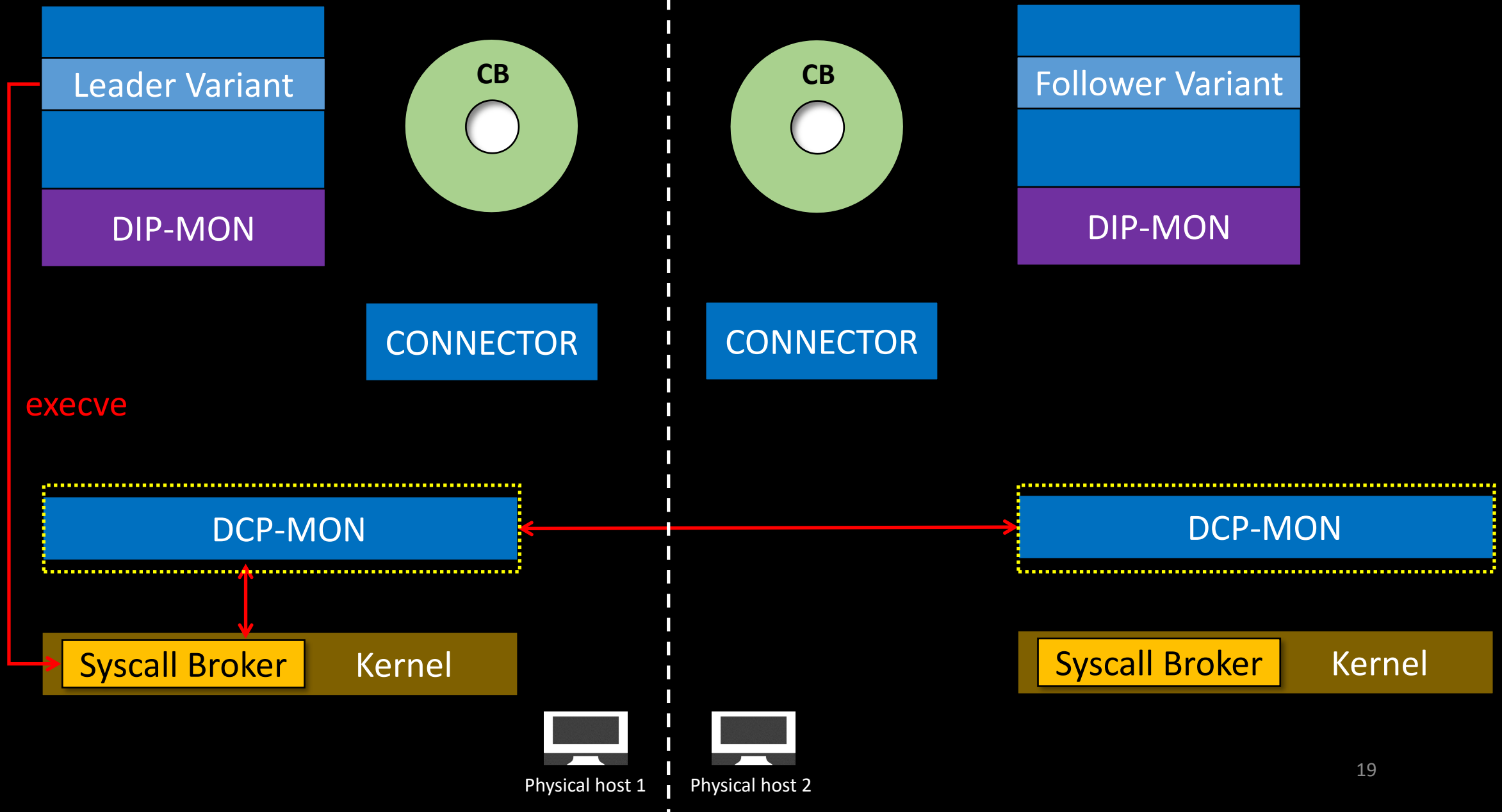
dMVX Design


Leader Variant
DIP-MON
CB
CB
Follower Variant
DIP-MON
CONNECTOR
CONNECTOR
getpid
execve
DCP-MON
DCP-MON
Syscall Broker
Kernel
Syscall Broker
Kernel
Physical host 1
Physical host 2

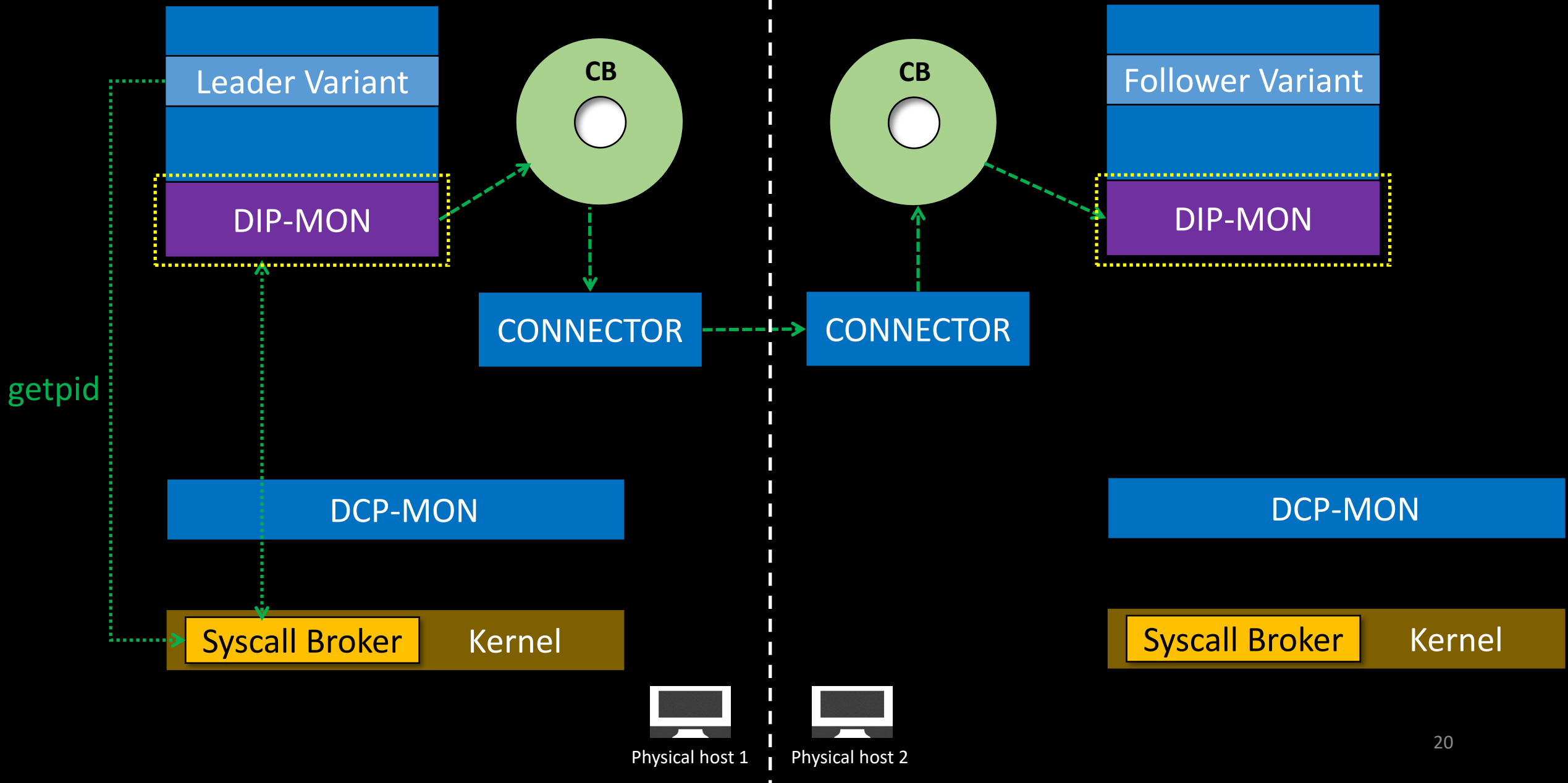
Core Components



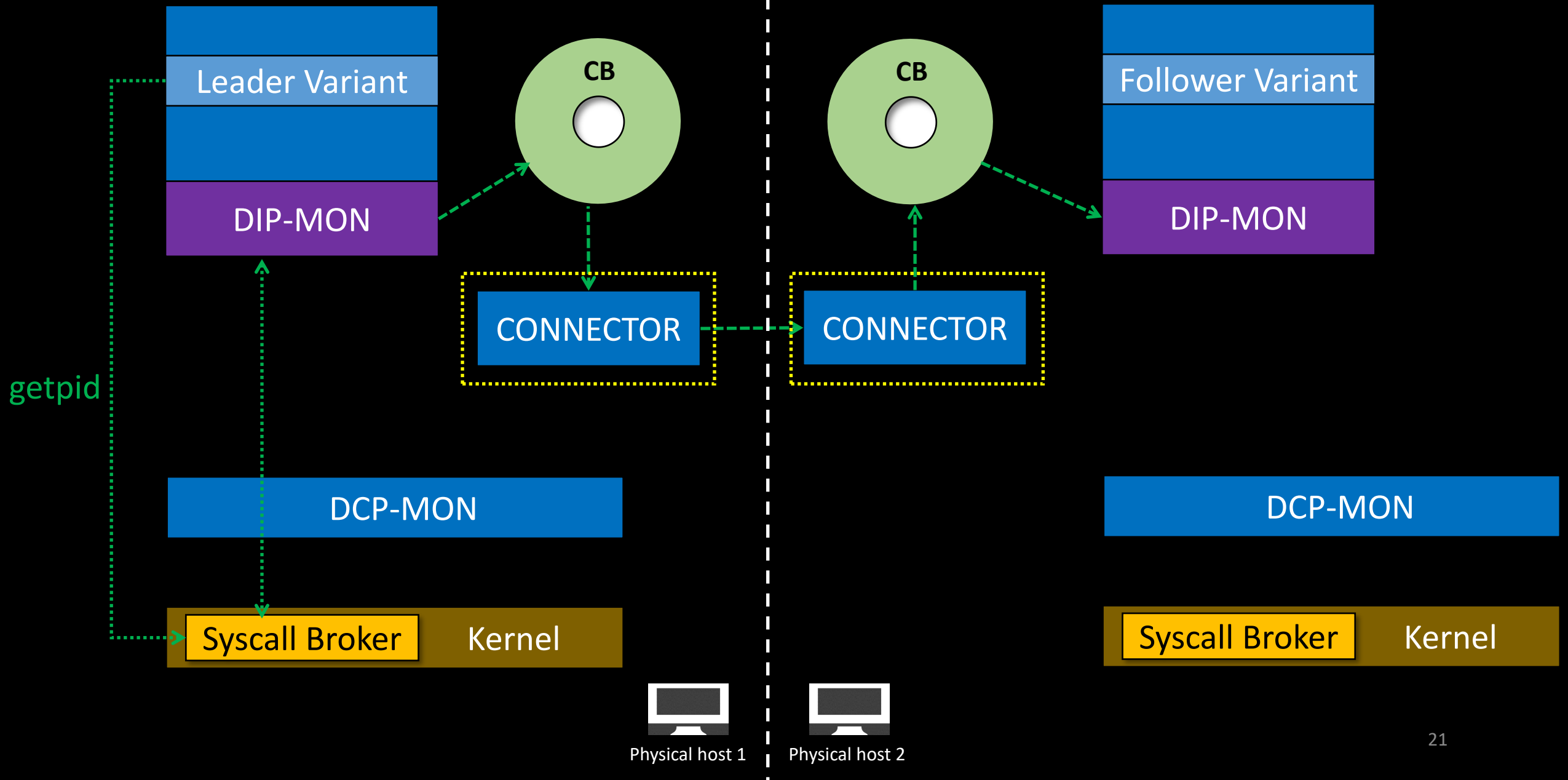
Core Components



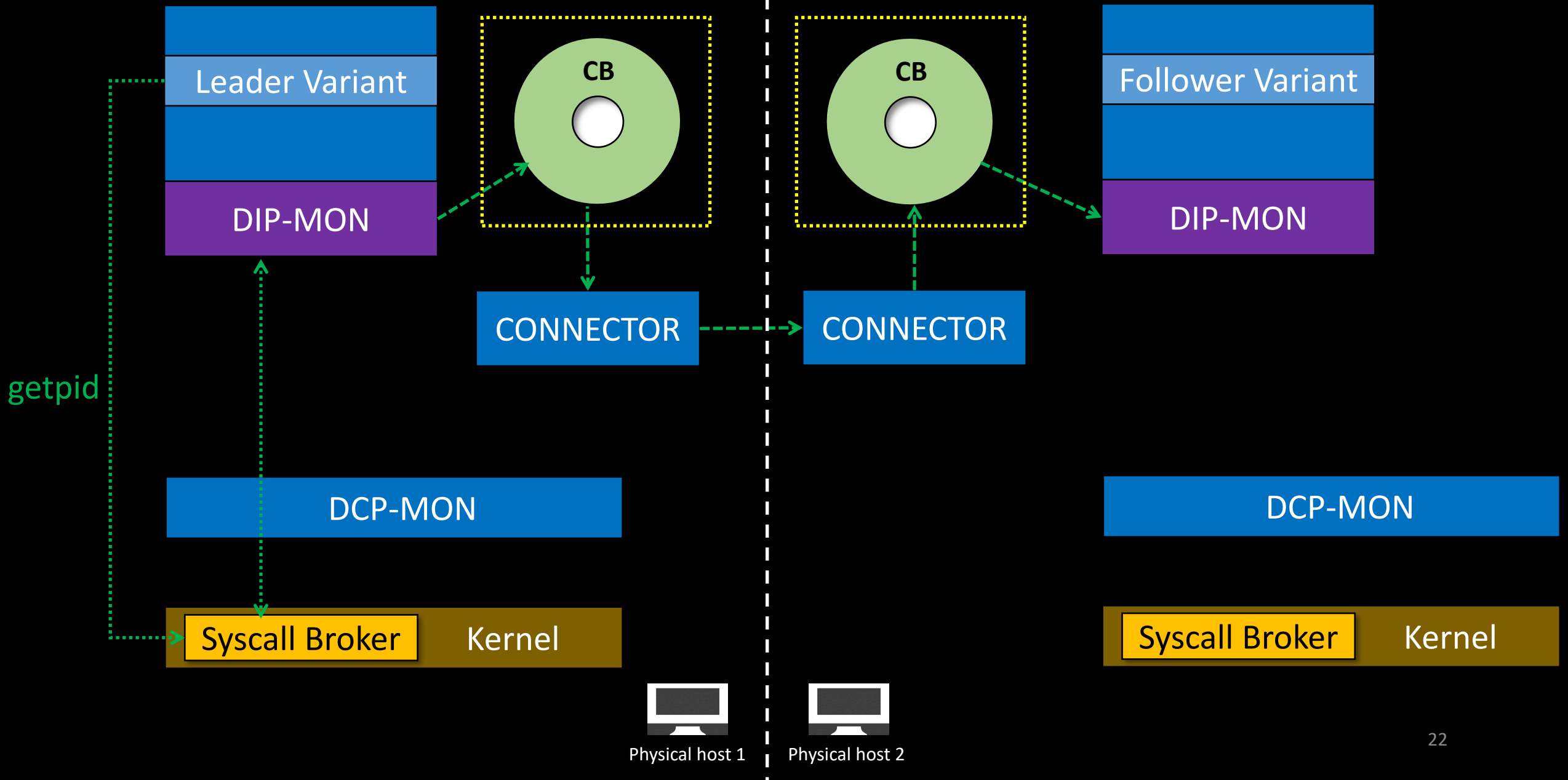
Core Components



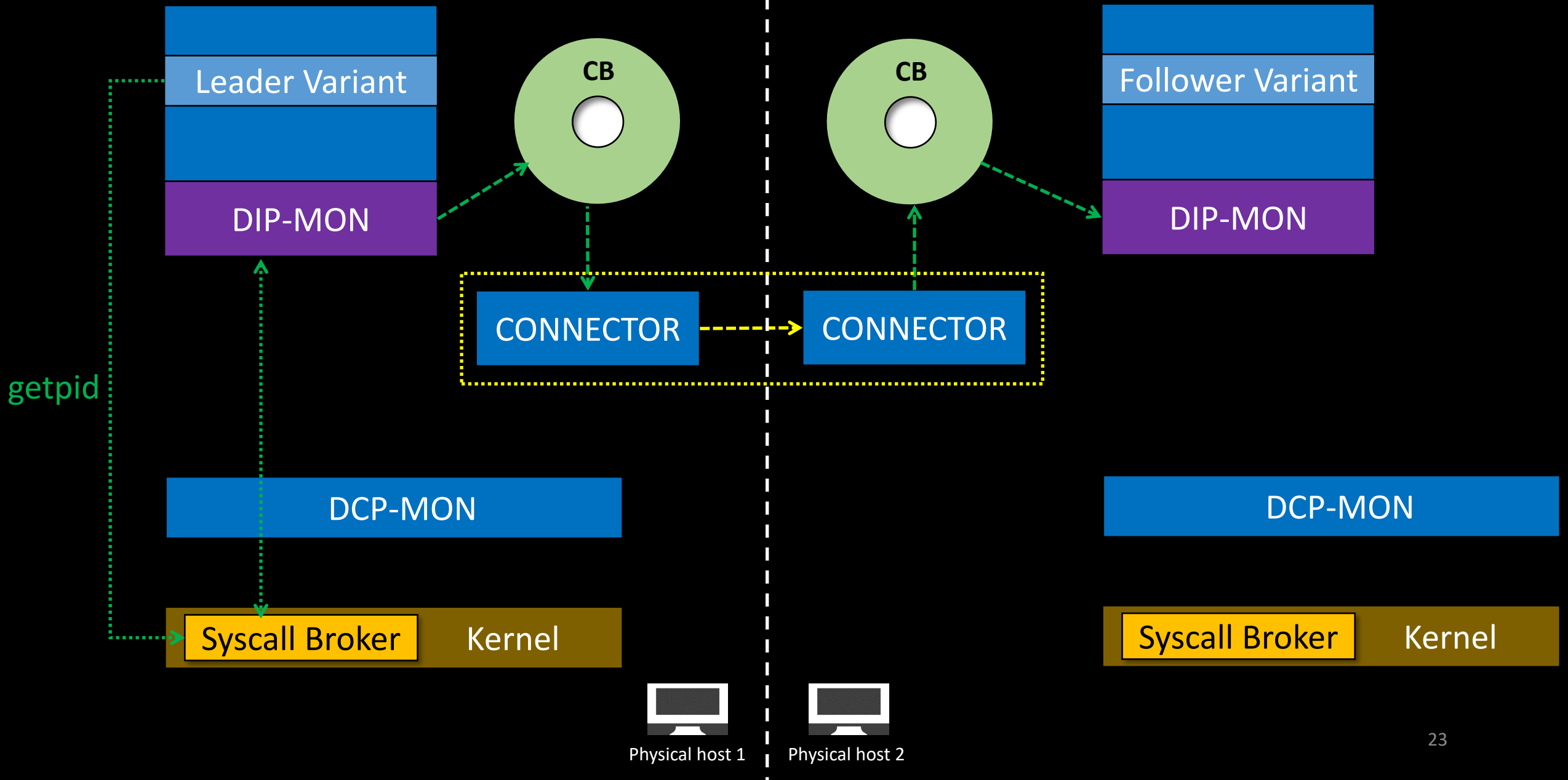
Core Components



Core Components

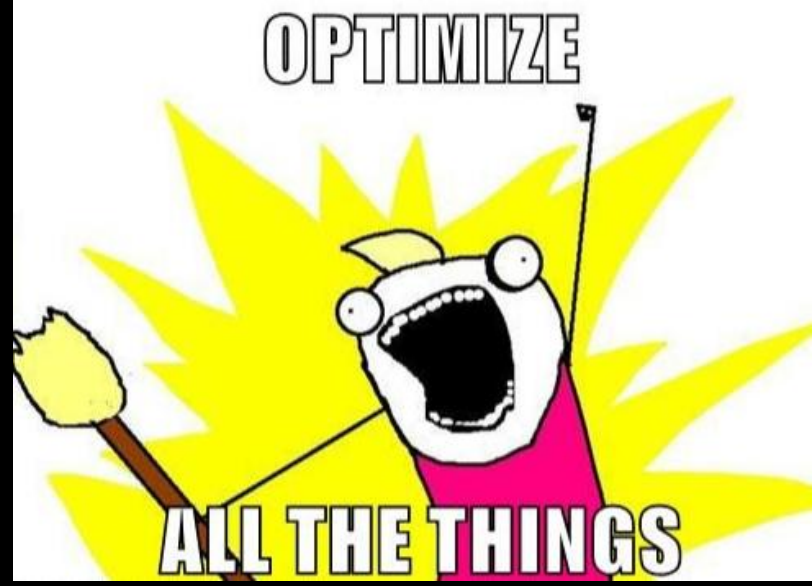


Core Components

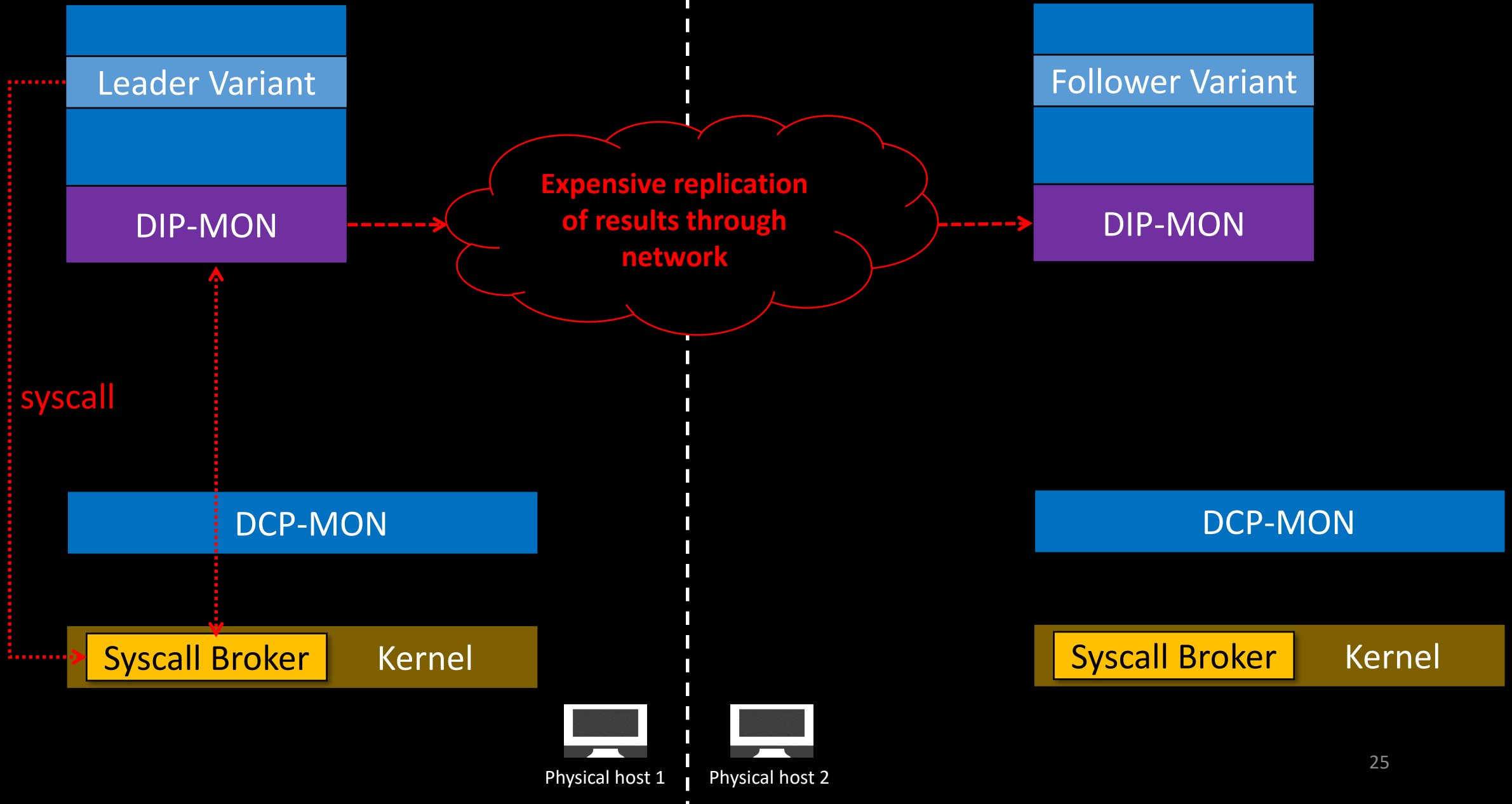


Additional Optimizations

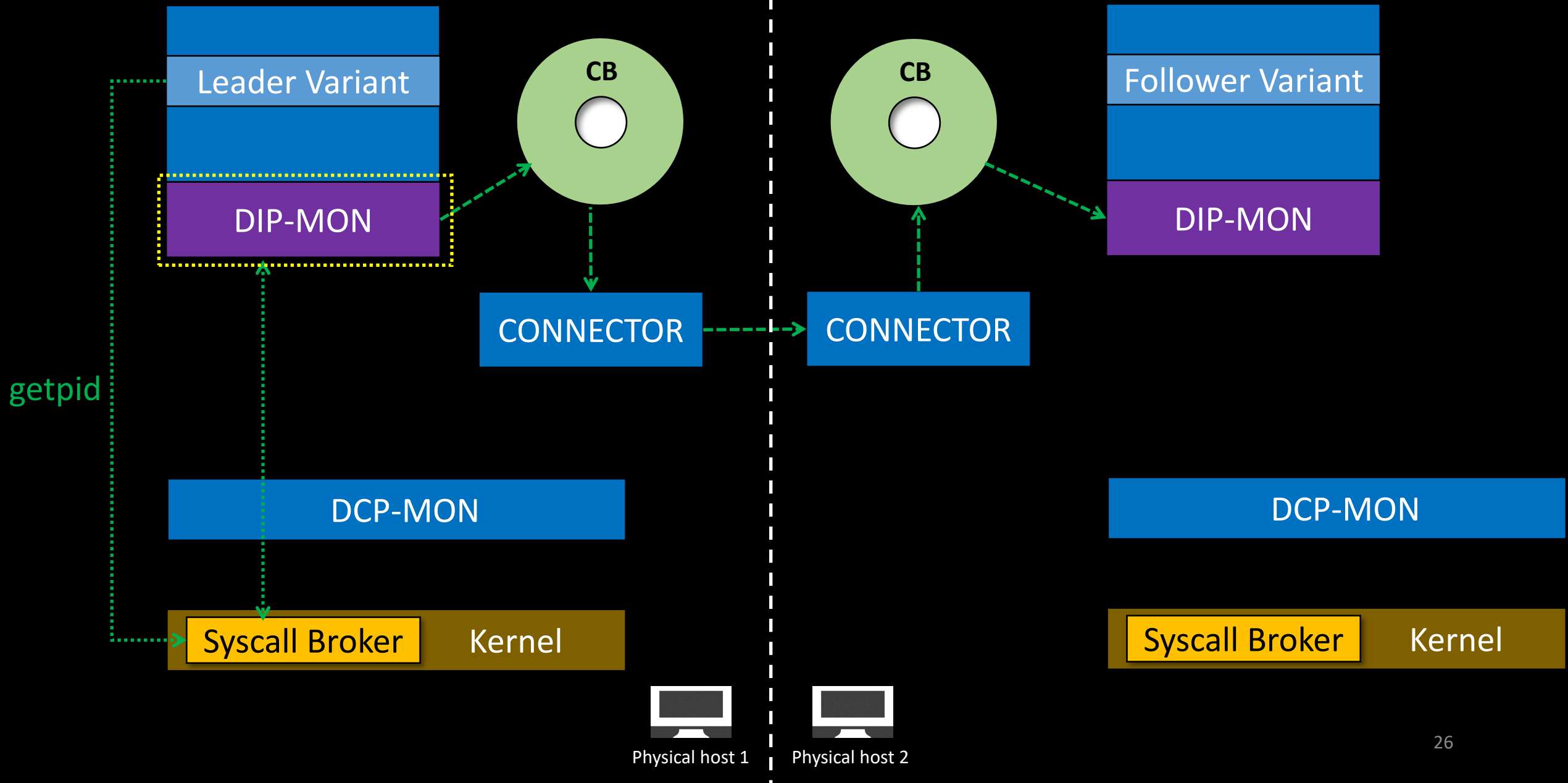
- Replication is *still* expensive
- Asynchronous replication
- Avoid replication when possible



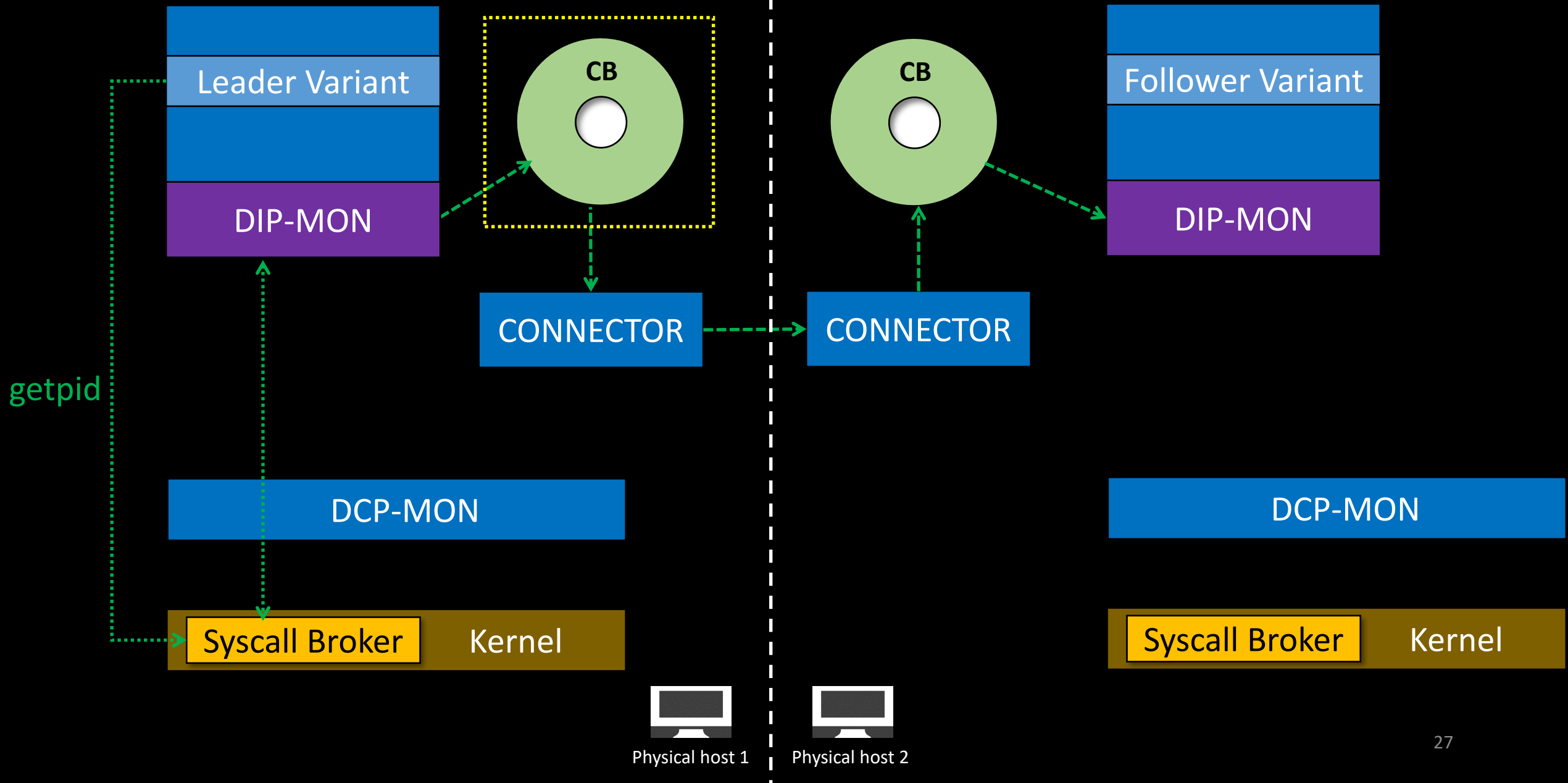
Replication



Asynchronous Replication



Asynchronous Replication



Asynchronous Replication

Leader Variant

DIP-MON

Follower Variant

DIP-MON

OR

getpid

DCP-MON

DCP-MON

Syscall Broker

Kernel

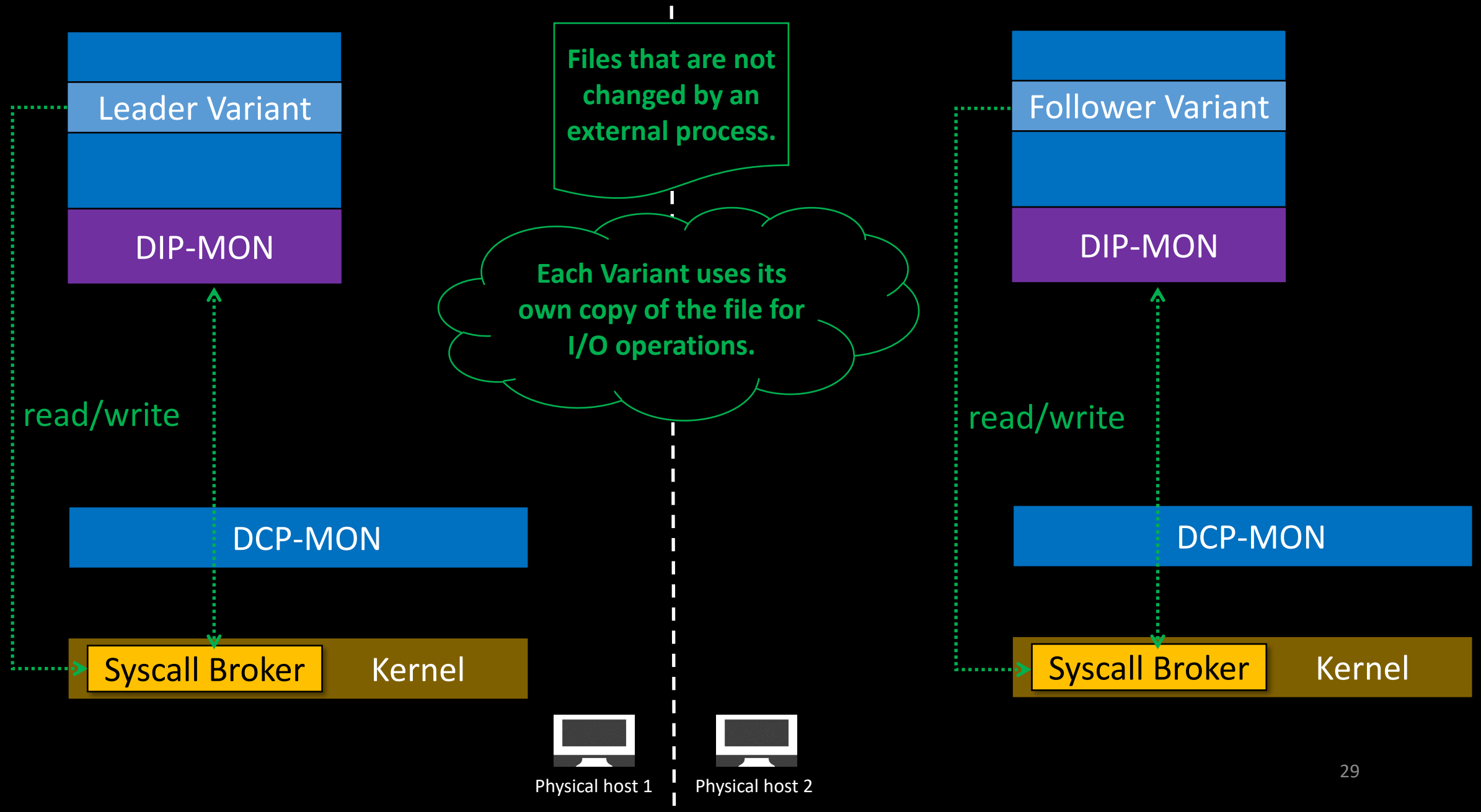
Syscall Broker

Kernel

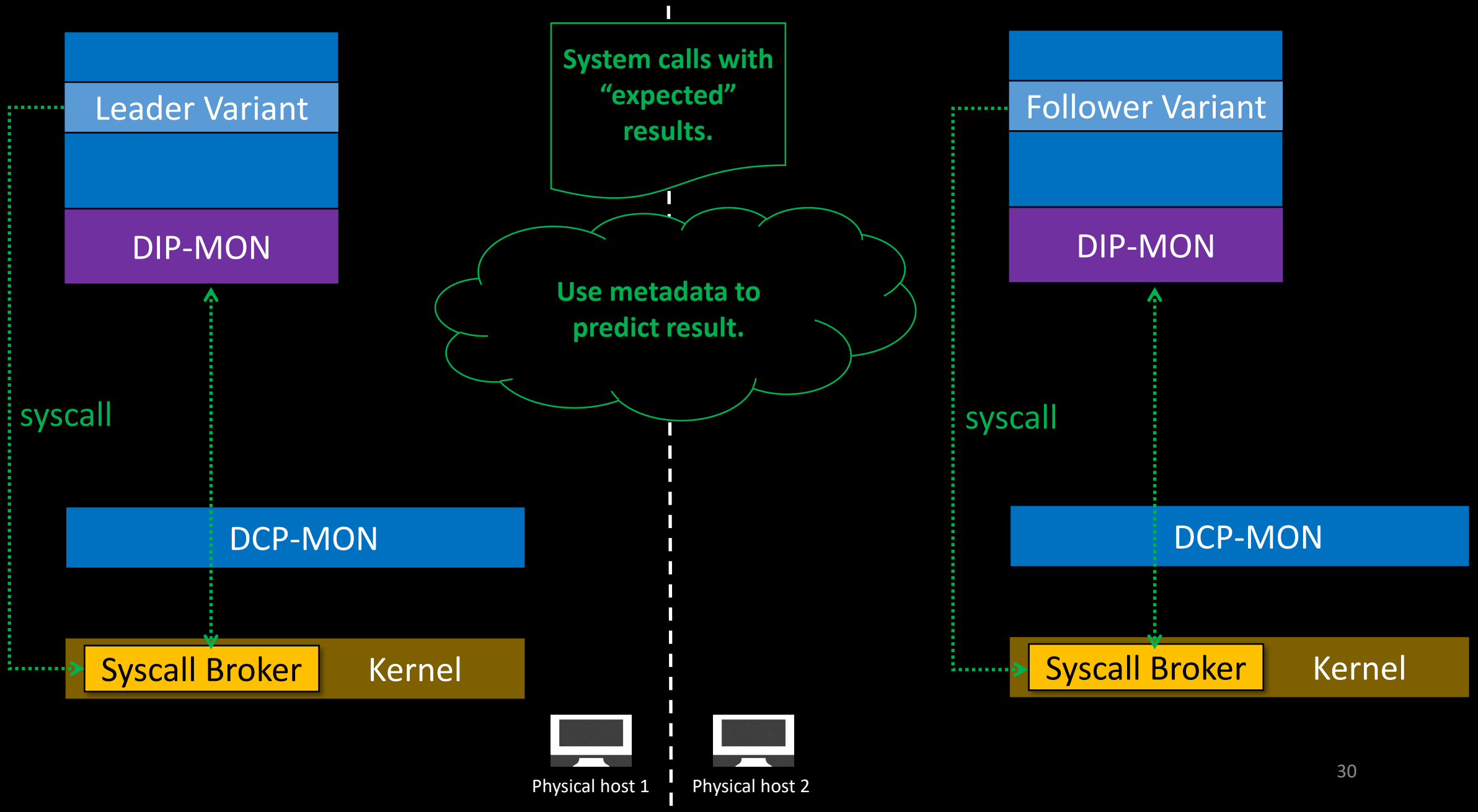
Physical host 1

Physical host 2

Selective Replication (1)



Selective Replication (2)



Security of dMVX

- Security-sensitive system calls are always monitored
- CONNECTOR is a separate process
- Information hiding to protect the in-process monitors and sensitive values

Case Studies

Benchmark	DMON	dMVX
READ	37.04×	**6.78**×
GETCWD	39.39×	**2.79**×
SCHED_YIELD	37.90×	**2.87**×
Lighttpd	5.43×	**3.1%**

Case Studies

Benchmark	DMON	dMVX
READ	37.04×	**6.78**×
GETCWD	39.39×	**2.79**×
SCHED_YIELD	37.90×	**2.87**×
Lighttpd	5.43×	**3.1%**

Conslusion

- dMVX: new distributed hybrid MVX design
  - Low system call interception cost
  - Avoid monitoring and replication when possible
  - Provide similar security guarantees with other distributed MVX systems
- Evaluation
  - Microbenchmarks
  - Lighttpd

- Alexios Voulimeneas
- Email: alex.voulimeneas@kuleuven.be