Gray Failure: The Achilles' Heel of Cloud-Scale Systems

Ryan Huang, Chuanxiong Guo, Lidong Zhou, Jacob R. Lorch,
Yingnong Dang, Murali Chintalapati, Randolph Yao

Microsoft

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Outline

- ❑ **Background & the gray failure problem**

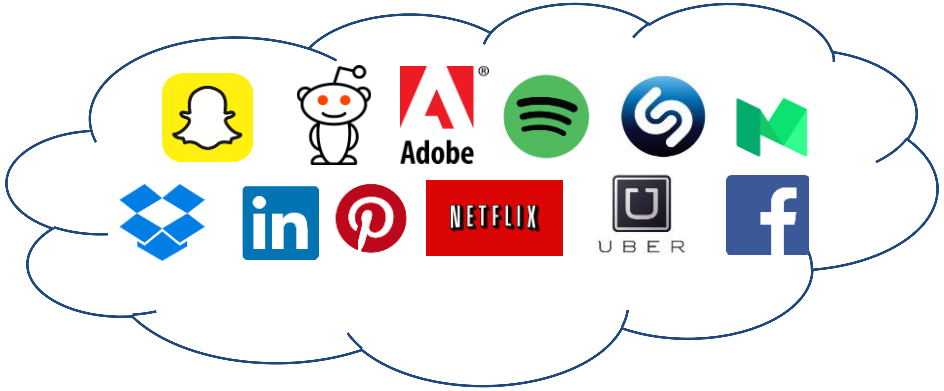
- ❑ **Real-world gray failure cases in Azure**

- ❑ **A model and a definition for gray failure**
  - ❖ differential observability

- ❑ **Potential future directions**

Rapid Growth of Cloud System Infra

- » **Software user shift**
  - direct: e.g., office 365, Google Drive
  - indirect: e.g., Netflix on AWS

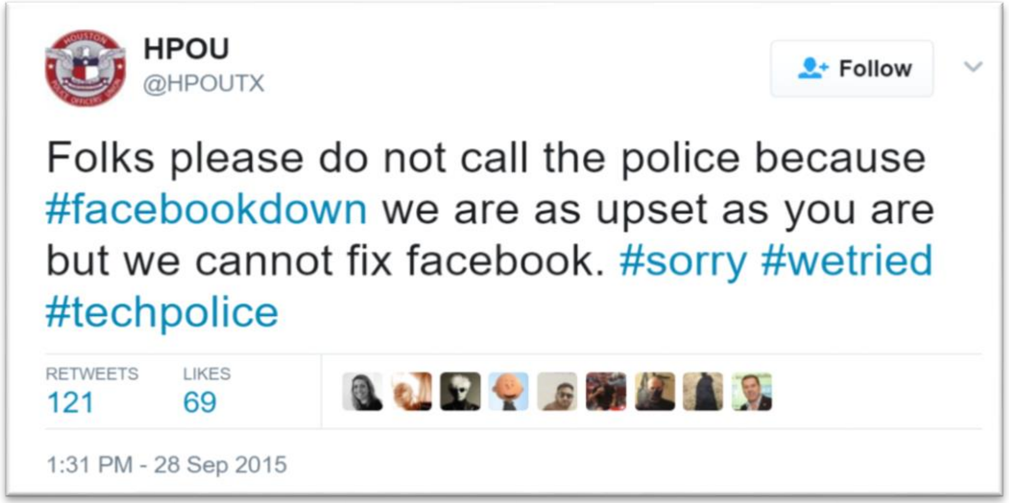


- » **Workload diversity**
  - website, workflow, big data, machine learning

- » **Internal composition**
  - more data centers, larger cluster, special h/w
  - containerization, micro-services

Demanding Requirement on Availability

- Users intolerant of service downtime



HPOU
@HPOUTX

Folks please do not call the police because #facebookdown we are as upset as you are but we cannot fix facebook. #sorry #wetried #techpolice

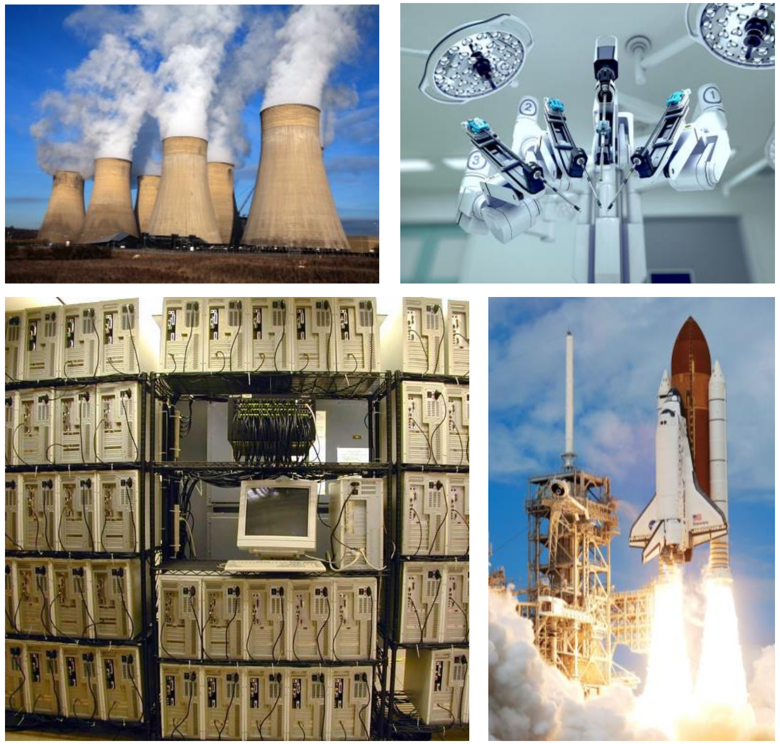
RETWEETS 121 LIKES 69

1:31 PM - 28 Sep 2015

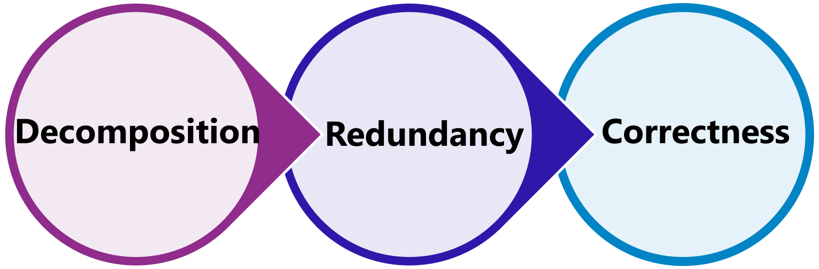
- Failure more costly
  - SLA violation, reputation hit, customer loss, engineering resource waste
- New availability bar
  - 3 nines to 5 or 6 nines

Key: Embrace Fault-tolerance!

Rich history since 1950s



Steps:



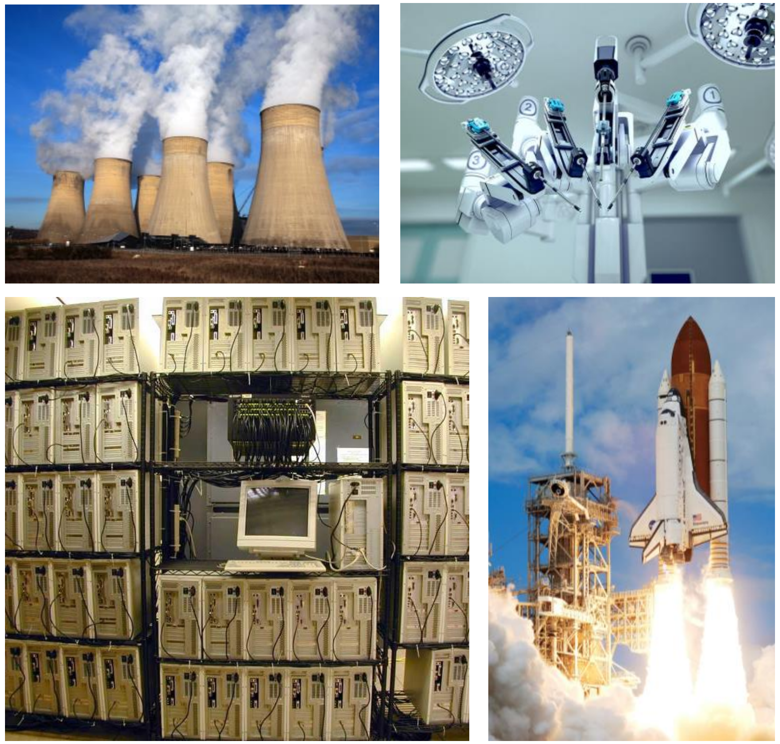
Decomposition

Redundancy

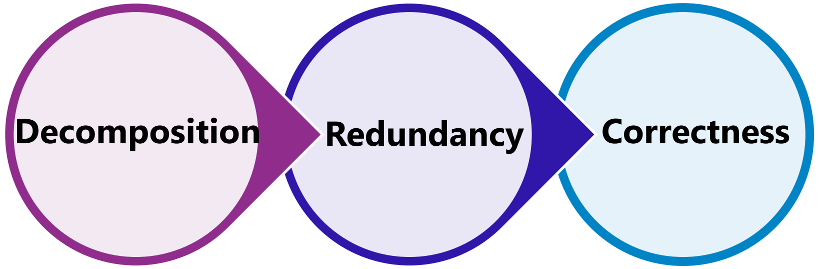
Correctness

Key: Embrace Fault-tolerance!

Rich history since 1950s



Steps:



Decomposition

Redundancy

Correctness

Building block:

process pair | triple modular redundancy | Primary/backup

RAID | state machine replication | checkpoint | chain replication

transaction | 2PC | Gossip | N-version | erasure coding

virtual synchrony | Paxos | Zab | PBFT | Zyzzyva …

Status Quo

Status Quo

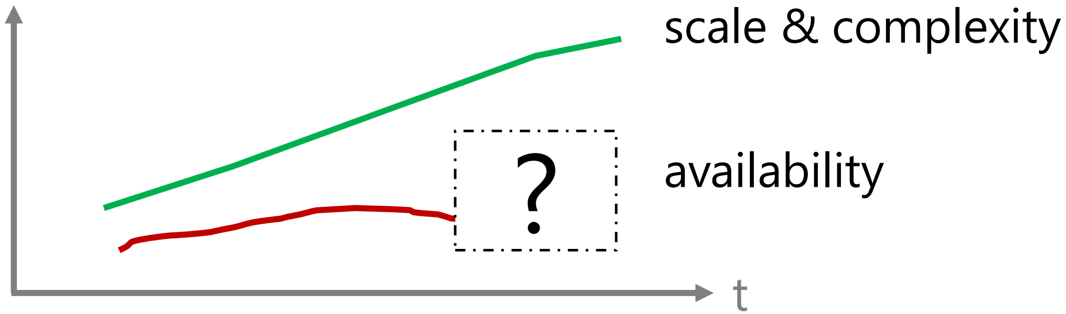
- **By and large, the efforts paid off**
  - many faults successfully detected, tolerated, and repaired every day
  - few global outages
  - 99.9% is achievable

- **But moving forward…**
  - simplistic assumptions start to break
  - reasoning about availability becomes hard
  - frequent bizarre phenomenon in production
  - 99.999% and beyond is challenging

Status Quo

- **By and large, the efforts paid off**
  - many faults successfully detected, tolerated, and repaired every day
  - few global outages
  - 99.9% is achievable

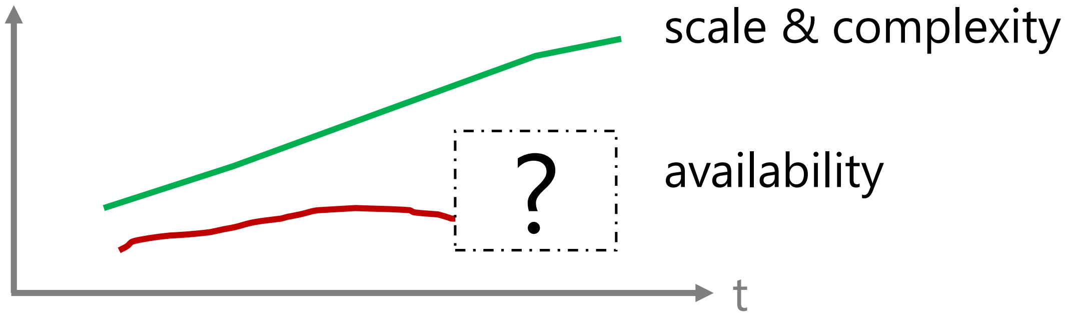
- **But moving forward…**
  - simplistic assumptions start to break
  - reasoning about availability becomes hard
  - frequent bizarre phenomenon in production
  - 99.999% and beyond is challenging



Status Quo

- By and large, the efforts paid off
  - many faults successfully detected, tolerated, and repaired every day
  - few global outages
  - 99.9% is achievable
- But moving forward…
  - simplistic assumptions start to break
  - reasoning about availability becomes hard
  - frequent bizarre phenomenon in production
  - 99.999% and beyond is challenging

A common theme: the overlooked gray failure problem



The Elephant in the Cloud - Gray Failure

A component either works correctly or stops

Fail-stop

process pair

TMR

RAID

primary backup

2PC

chain replication

Paxos

erasure coding

virtual synchrony

Zab

…

The Elephant in the Cloud - Gray Failure

A component either works correctly or stops

Fail-stop

process pair

TMR

RAID

primary backup

2PC

chain replication

Paxos

erasure coding

virtual synchrony

Zab

…

A component may behave arbitrarily

Byzantine

PBFT

Zyzzyva

Q/U

BAR Gossip

Aliph

UpRight

…

The Elephant in the Cloud - Gray Failure

A component either works correctly or stops

Fail-stop

- process pair
- TMR
- RAID
- primary backup
- 2PC
- chain replication
- Paxos
- erasure coding
- virtual synchrony
- Zab

…

A component appears to be still working but is in fact experiencing severe issue

Gray failure

?

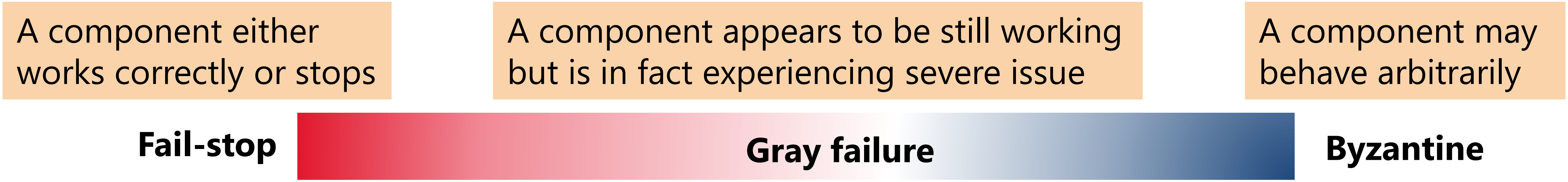
A component may behave arbitrarily

Byzantine

- PBFT
- Zyzzyva
- Q/U
- BAR Gossip
- Aliph
- UpRight

…

The Elephant in the Cloud - Gray Failure



A component either works correctly or stops

A component appears to be still working but is in fact experiencing severe issue

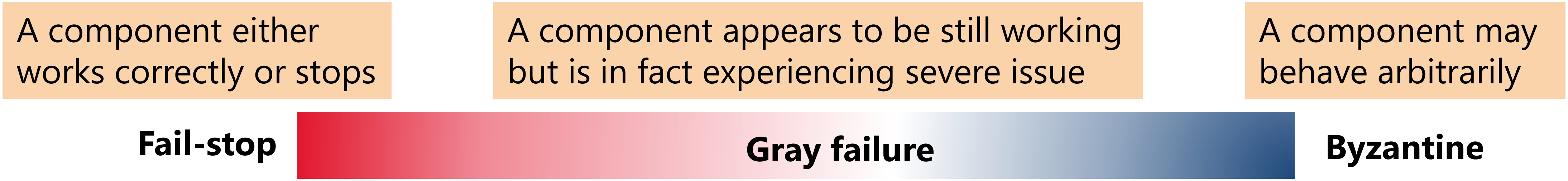
A component may behave arbitrarily

Fail-stop

Gray failure

Byzantine

The Elephant in the Cloud - Gray Failure



A component either works correctly or stops

A component appears to be still working but is in fact experiencing severe issue

A component may behave arbitrarily

symptom

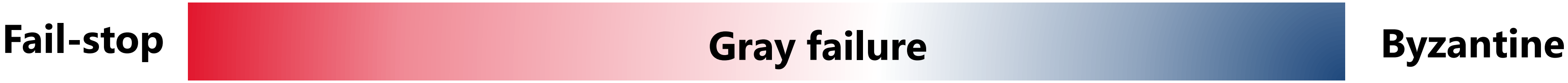
- subtle and ambiguous: e.g., switch random packet loss, non-fatal exceptions, memory thrashing, flaky disk I/O, overload..

The Elephant in the Cloud - Gray Failure

A component either works correctly or stops

A component appears to be still working but is in fact experiencing severe issue

A component may behave arbitrarily



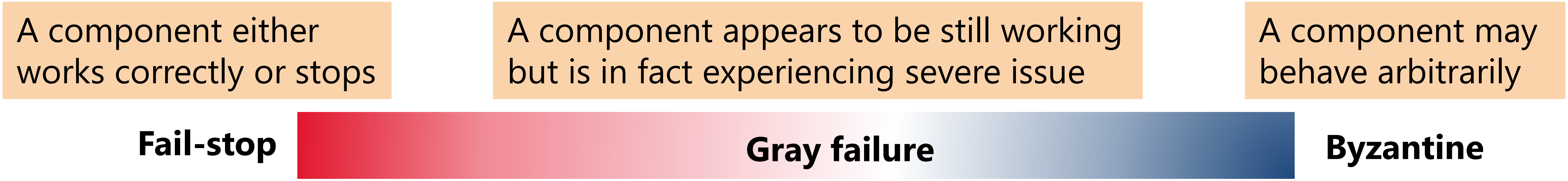
symptom

- subtle and ambiguous: e.g., switch random packet loss, non-fatal exceptions, memory thrashing, flaky disk I/O, overload..

occurrence

- across s/w and h/w stack in the infra due to various defects
- behind most service incidents we’ve seen in Azure

The Elephant in the Cloud - Gray Failure



A component either works correctly or stops

A component appears to be still working but is in fact experiencing severe issue

A component may behave arbitrarily

Fail-stop

Gray failure

Byzantine

symptom

- subtle and ambiguous: e.g., switch random packet loss, non-fatal exceptions, memory thrashing, flaky disk I/O, overload..

occurrence

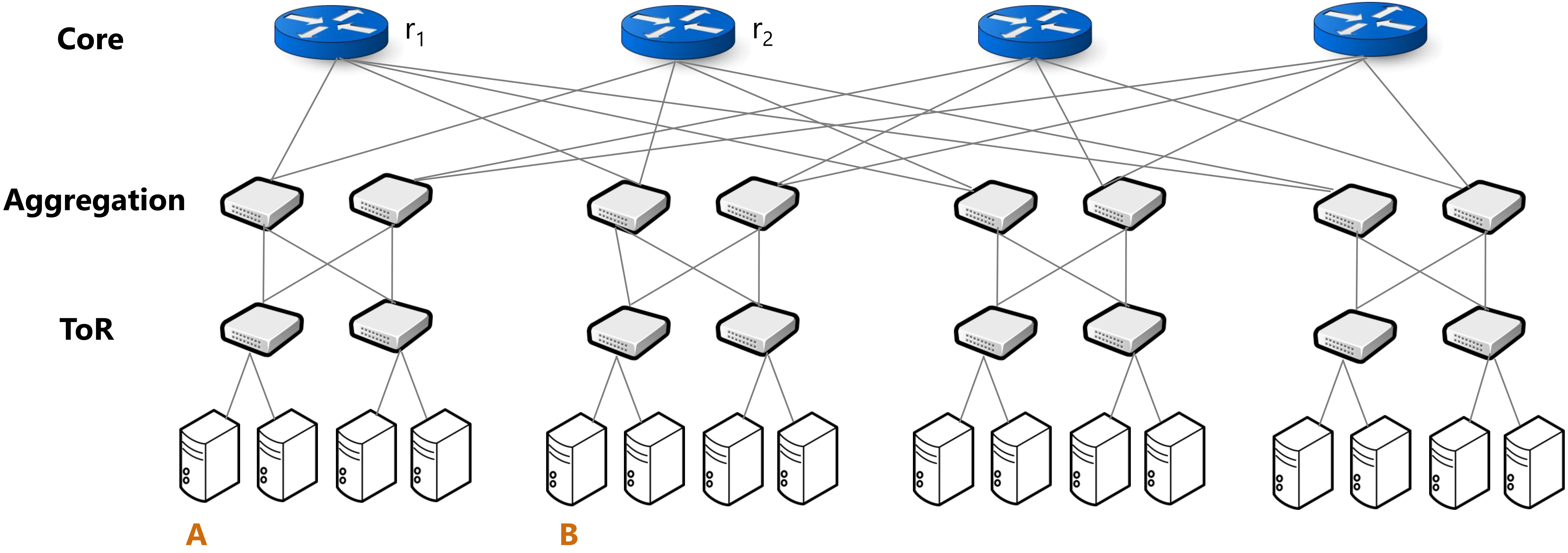
- across s/w and h/w stack in the infra due to various defects
- behind most service incidents we've seen in Azure

danger

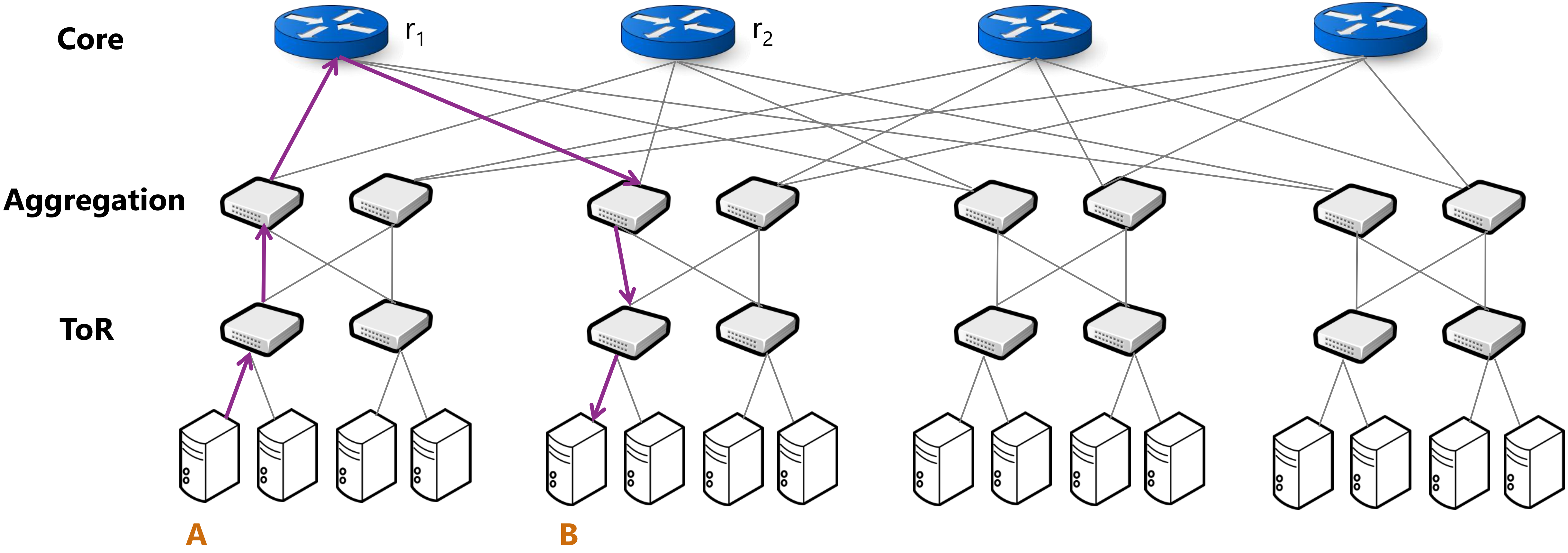
- fault-tolerance ineffective or counterproductive
- faults take engineers & designers huge efforts to nail down
- teams play the blame game with each other

Real-world Gray Failure Cases in Azure

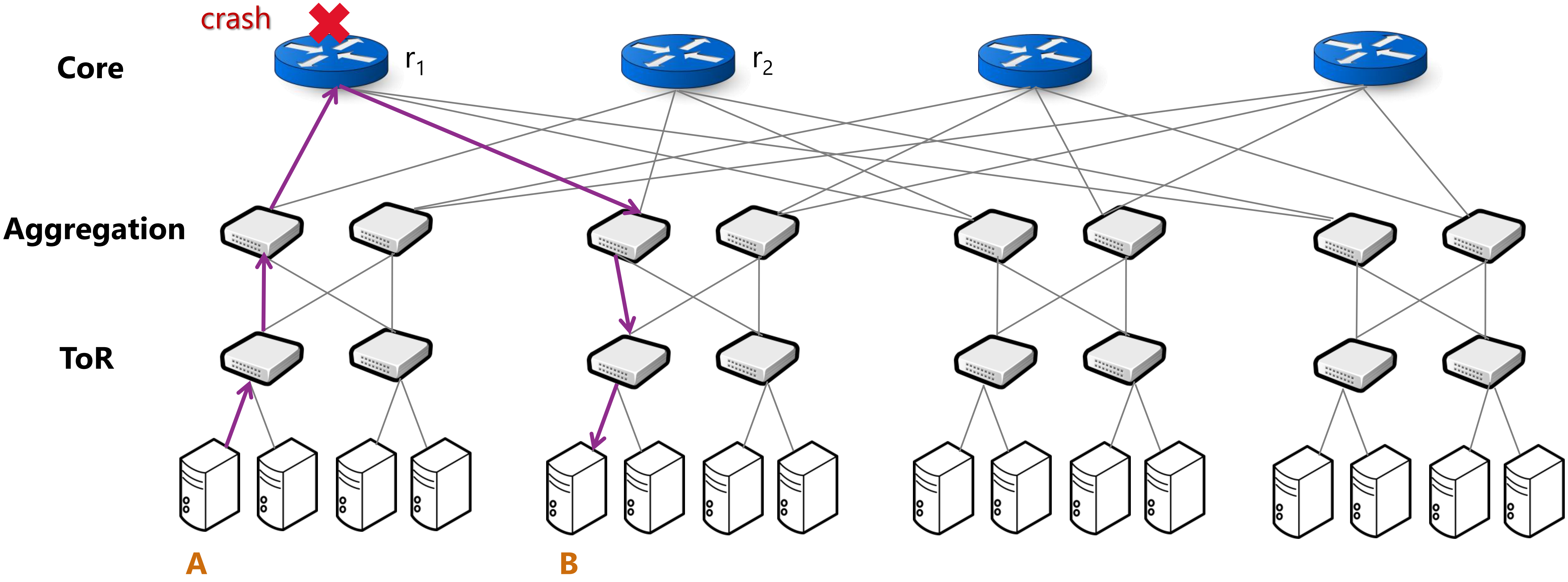
Case I: Redundancy in Datacenter Network (1)



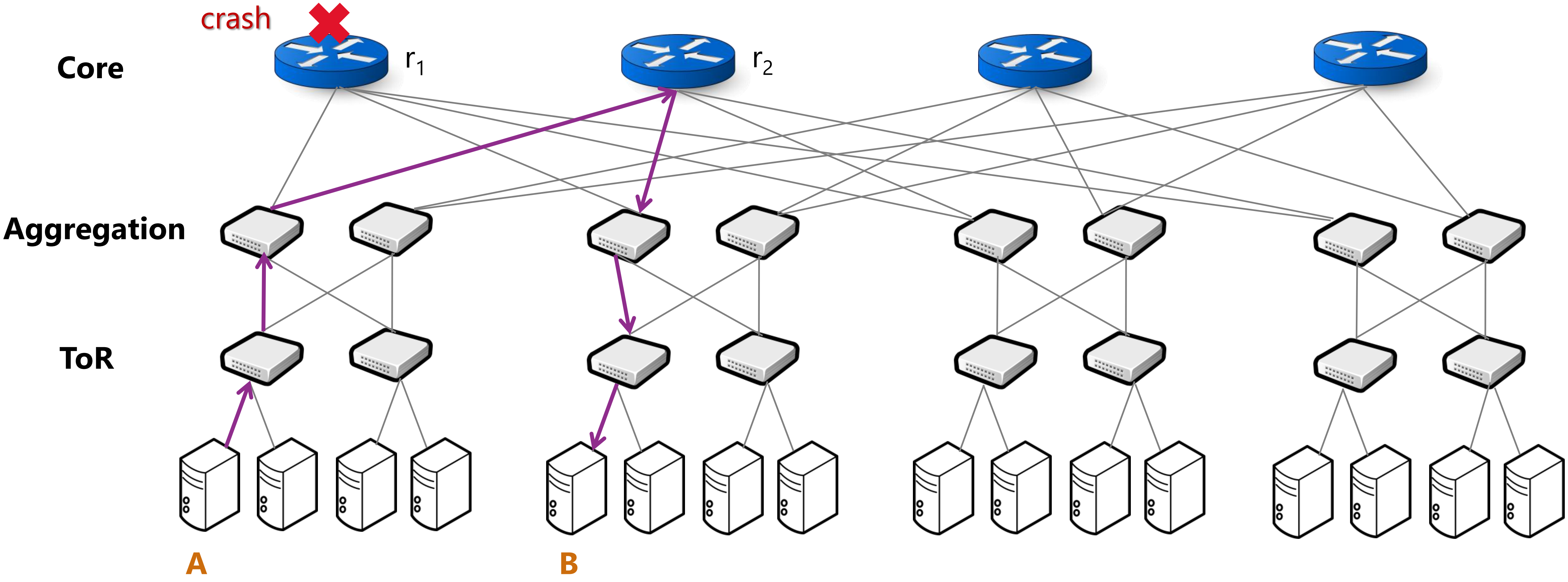
Case I: Redundancy in Datacenter Network (1)



Case I: Redundancy in Datacenter Network (1)

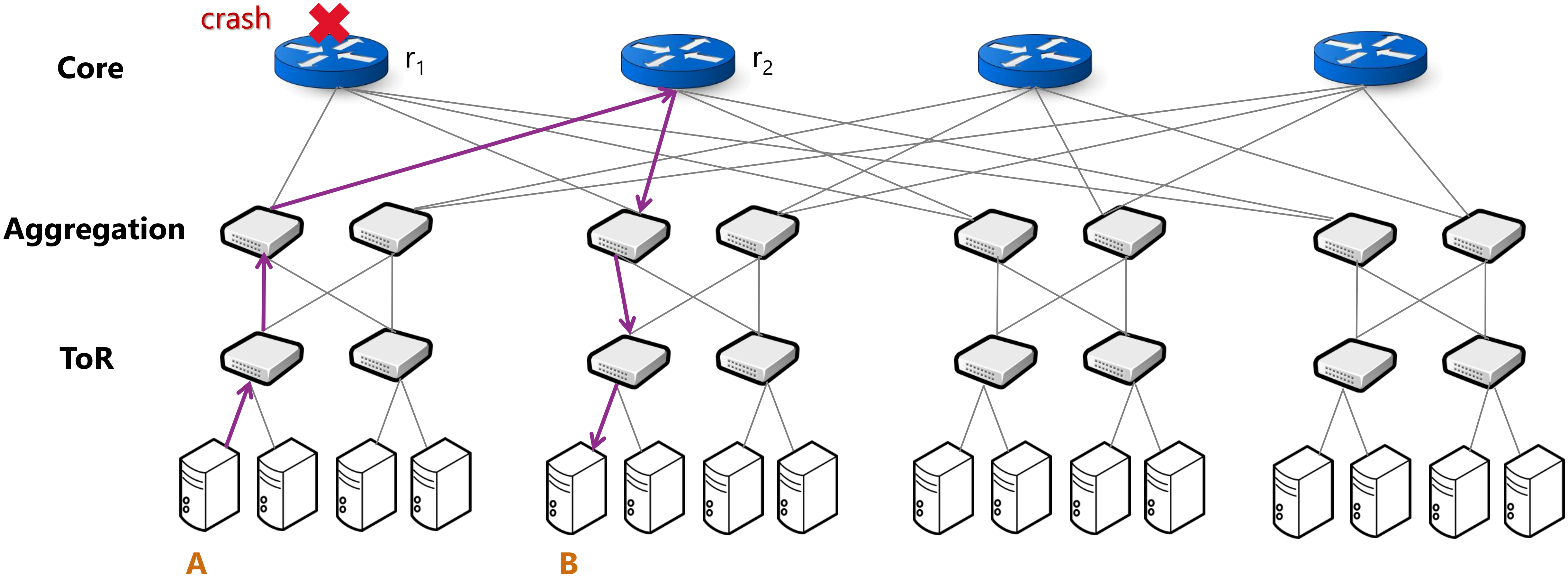


Case I: Redundancy in Datacenter Network (1)

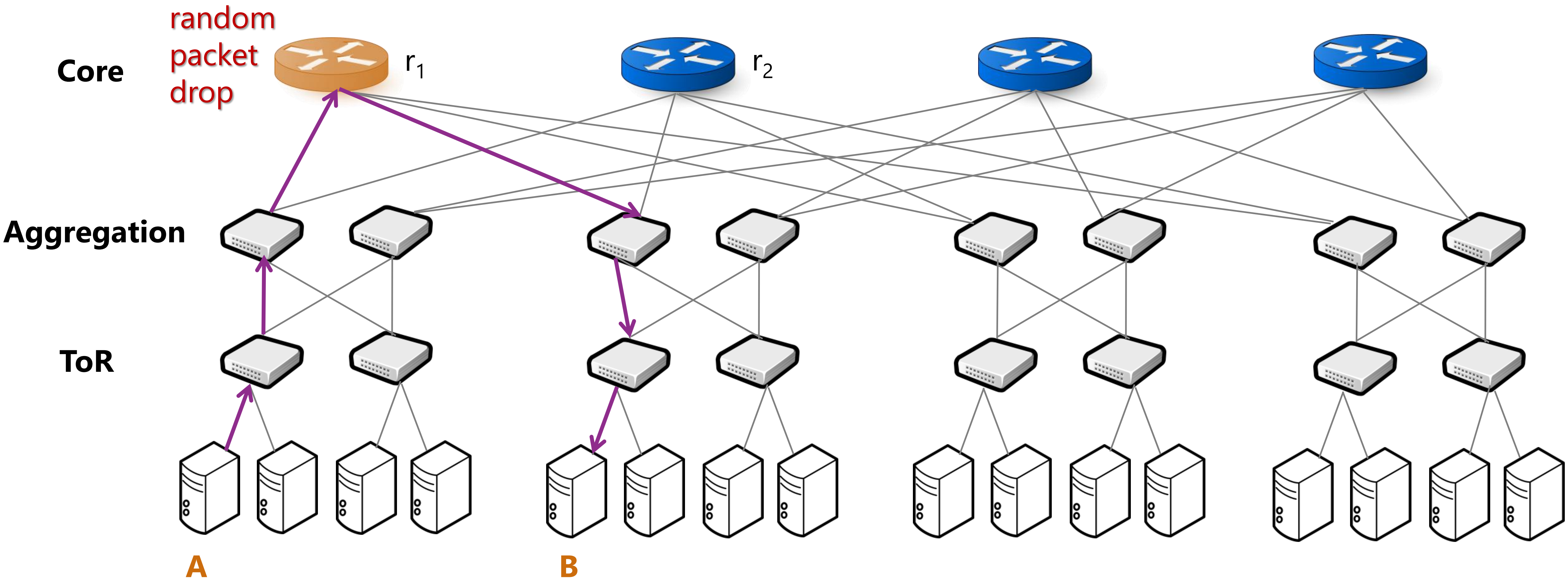


Case I: Redundancy in Datacenter Network (1)

increasing # of core switches helps with availability



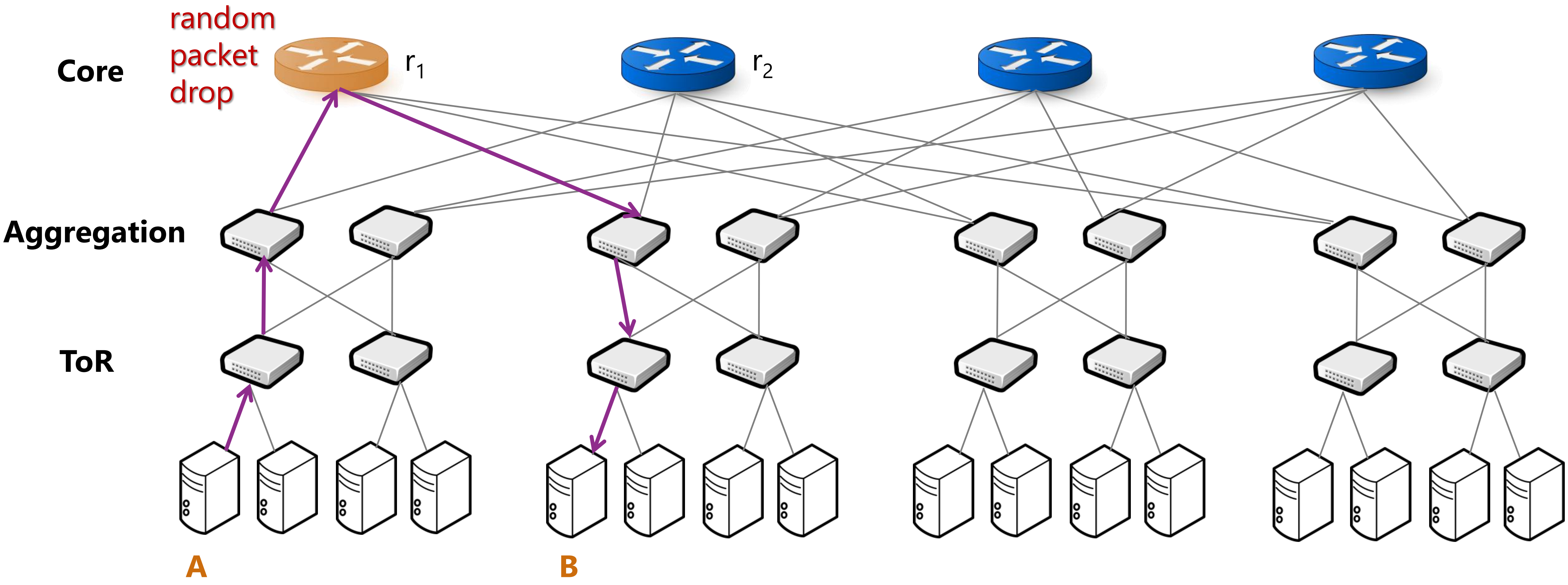
Case I: Redundancy in Datacenter Network (2)



Workload: single round trip

Case I: Redundancy in Datacenter Network (2)

- packets will not be re-routed → application glitches or increased latency
- increasing # of core switches may not affect chance of being affected *p*

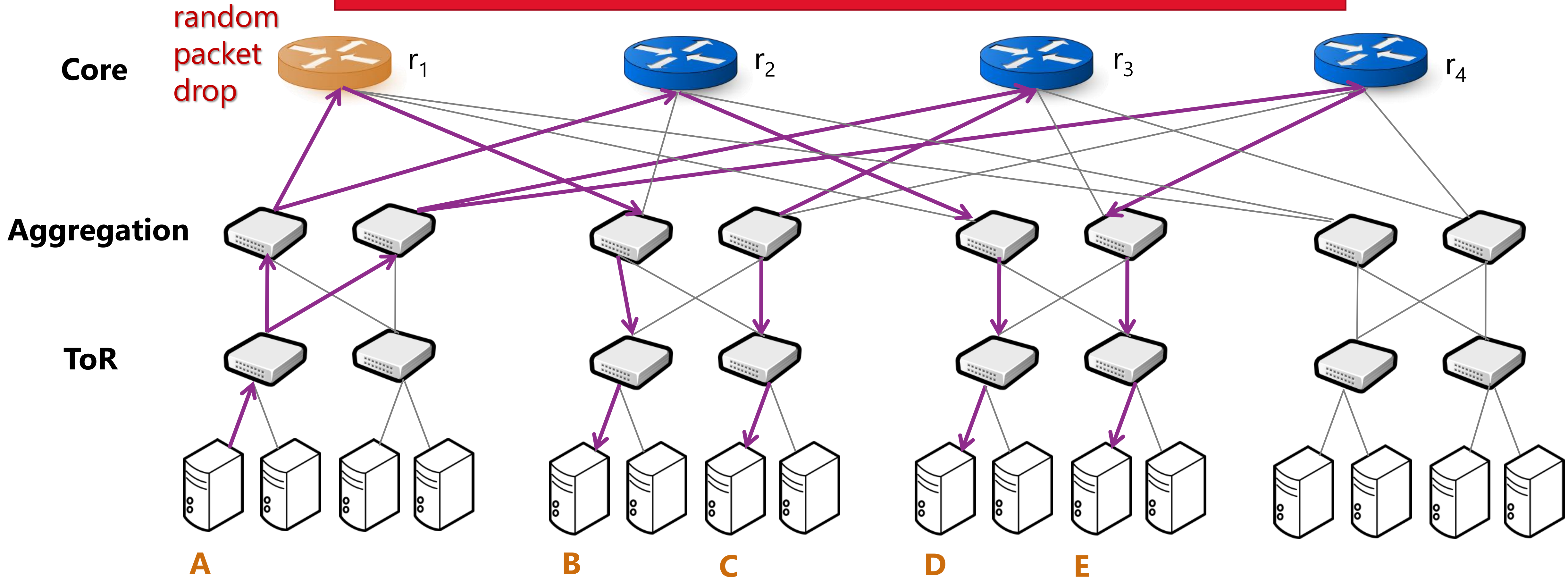


Workload: single round trip

Case I: Redundancy in Datacenter Network (3)

- high chance to involve every core switches for each front-end request
- gray failure at *any* core switch will cause delay
- more core switches → worse tail latencies

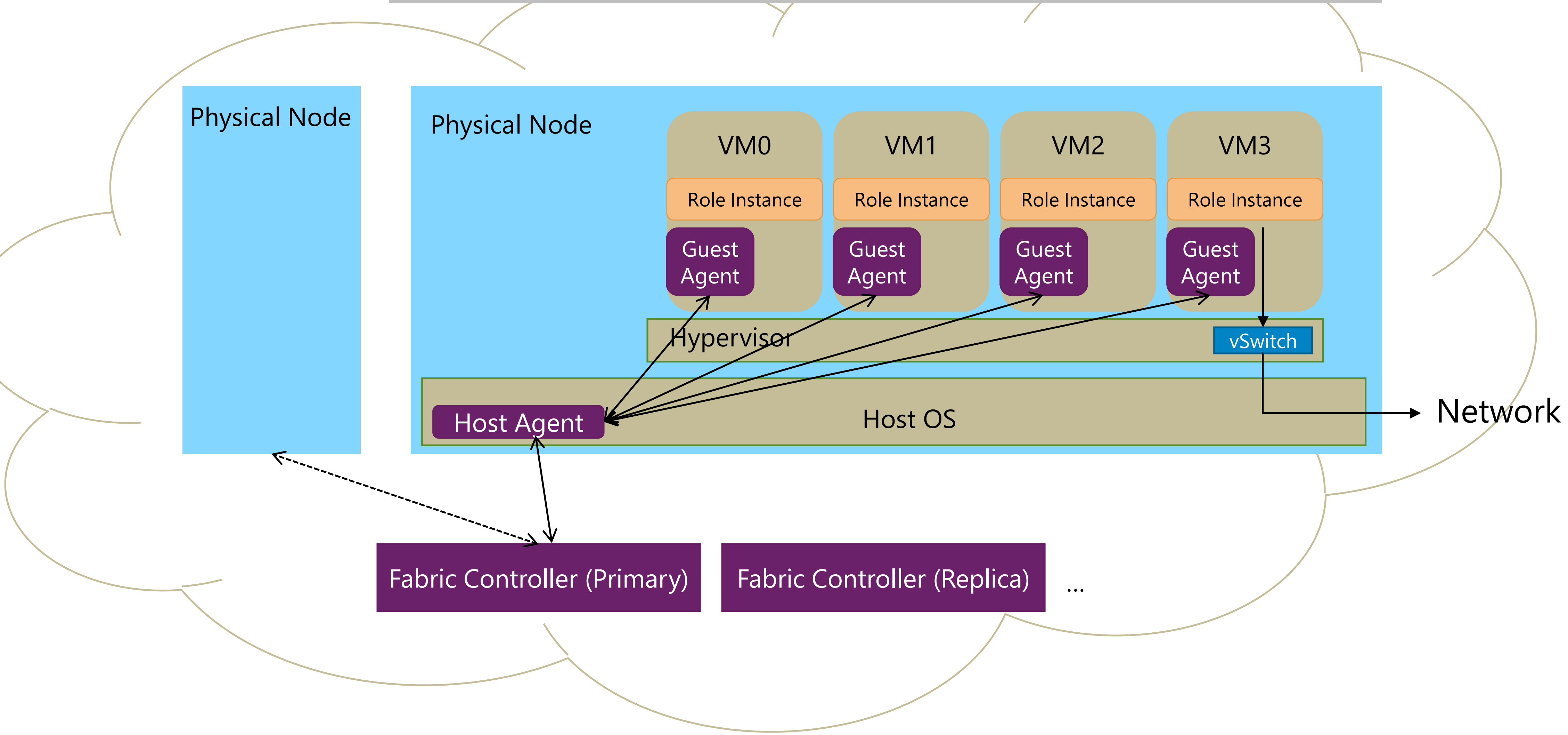
$$1 - (1 - p)^n$$



Workload: send multiple requests wait for all to finish (e.g., search)

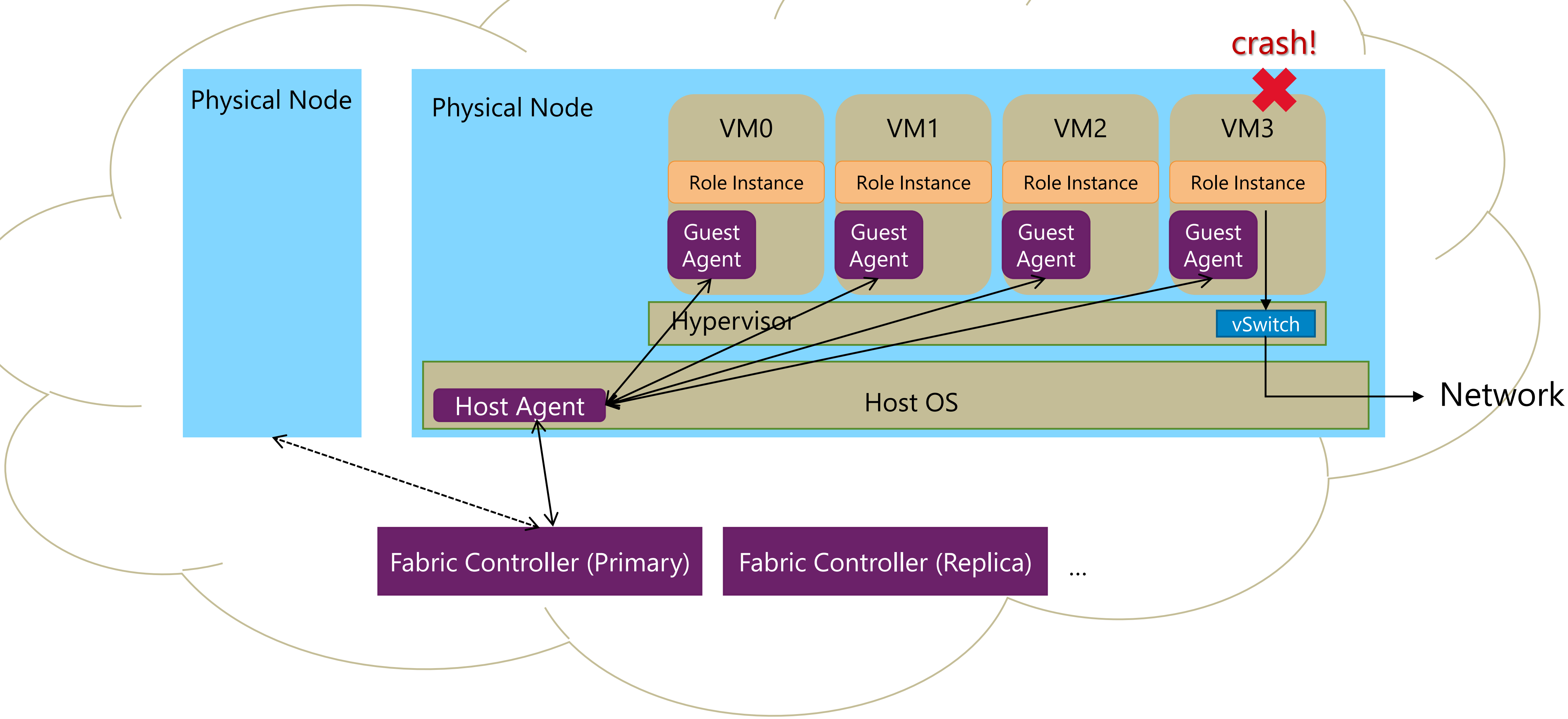
Case II: Failure Detector in Compute Service

Hierarchical agents to catch failure in different layers



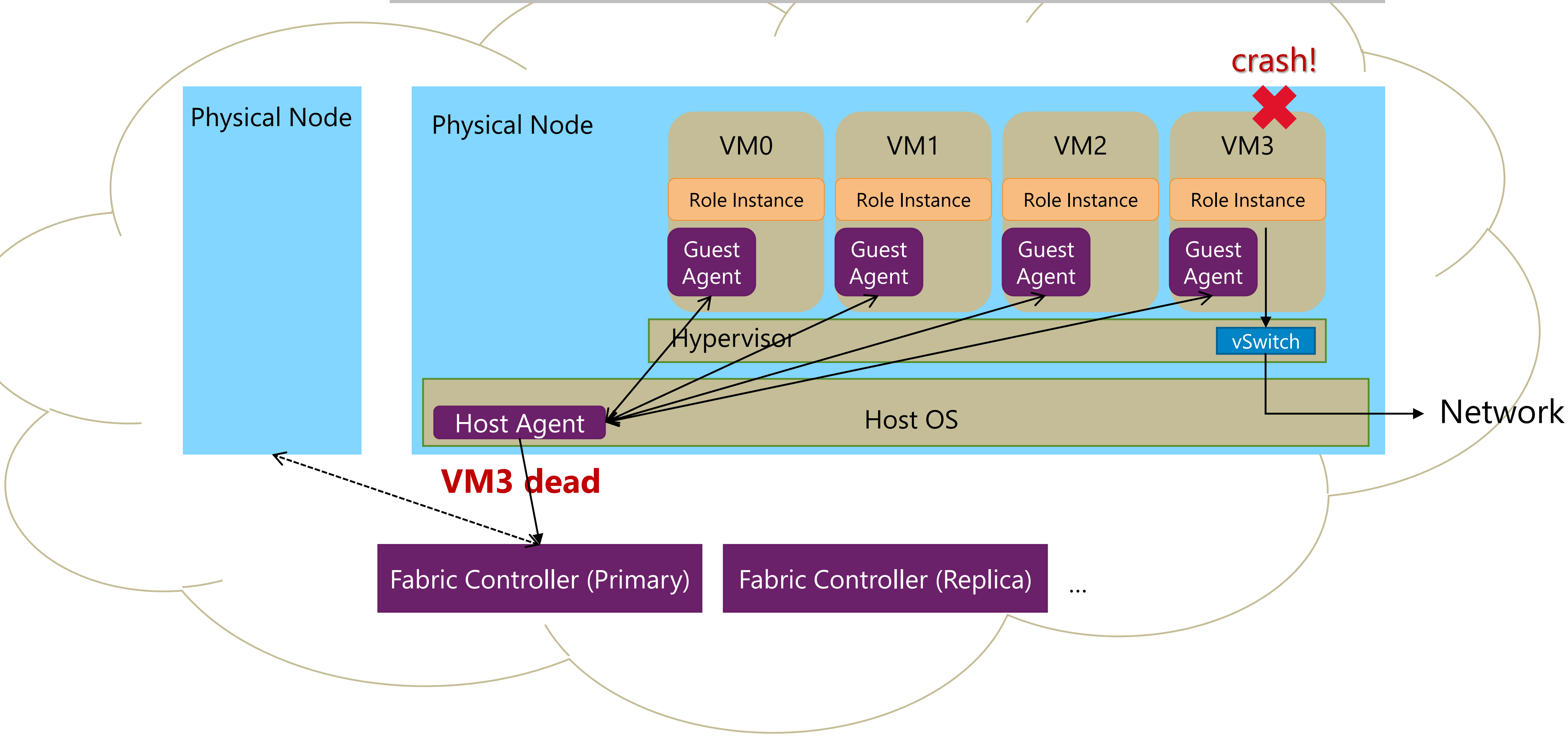
Case II: Failure Detector in Compute Service

Hierarchical agents to catch failure in different layers



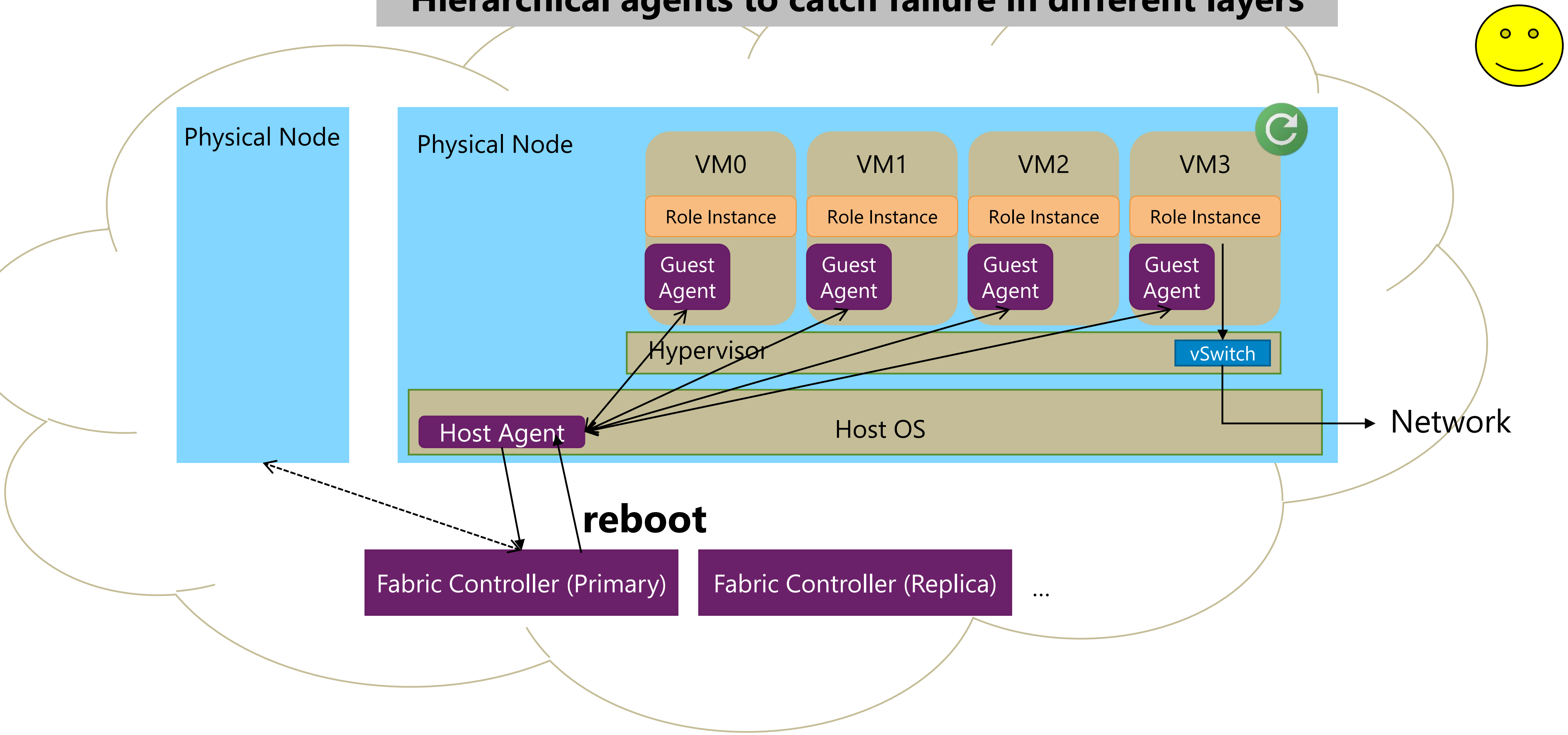
Case II: Failure Detector in Compute Service

Hierarchical agents to catch failure in different layers



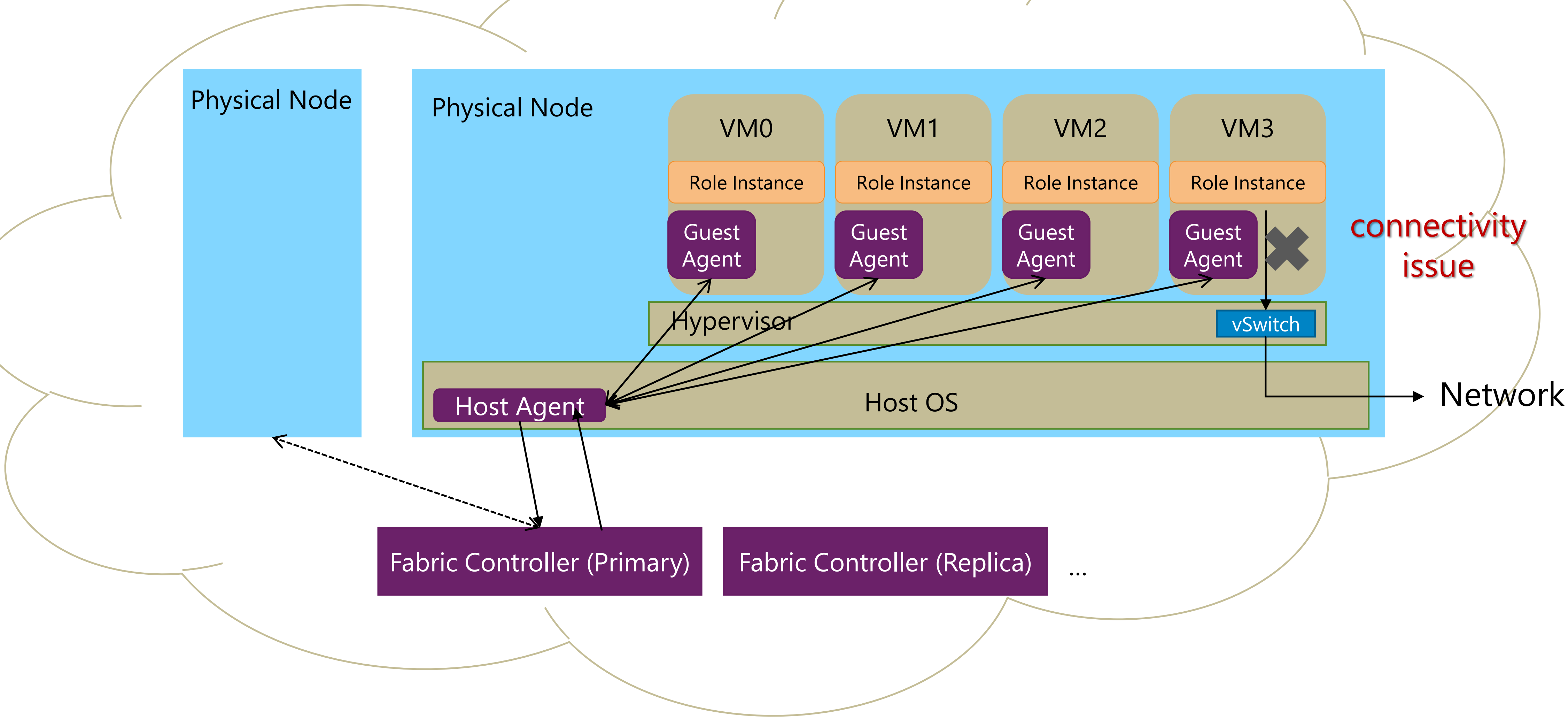
Case II: Failure Detector in Compute Service

Hierarchical agents to catch failure in different layers



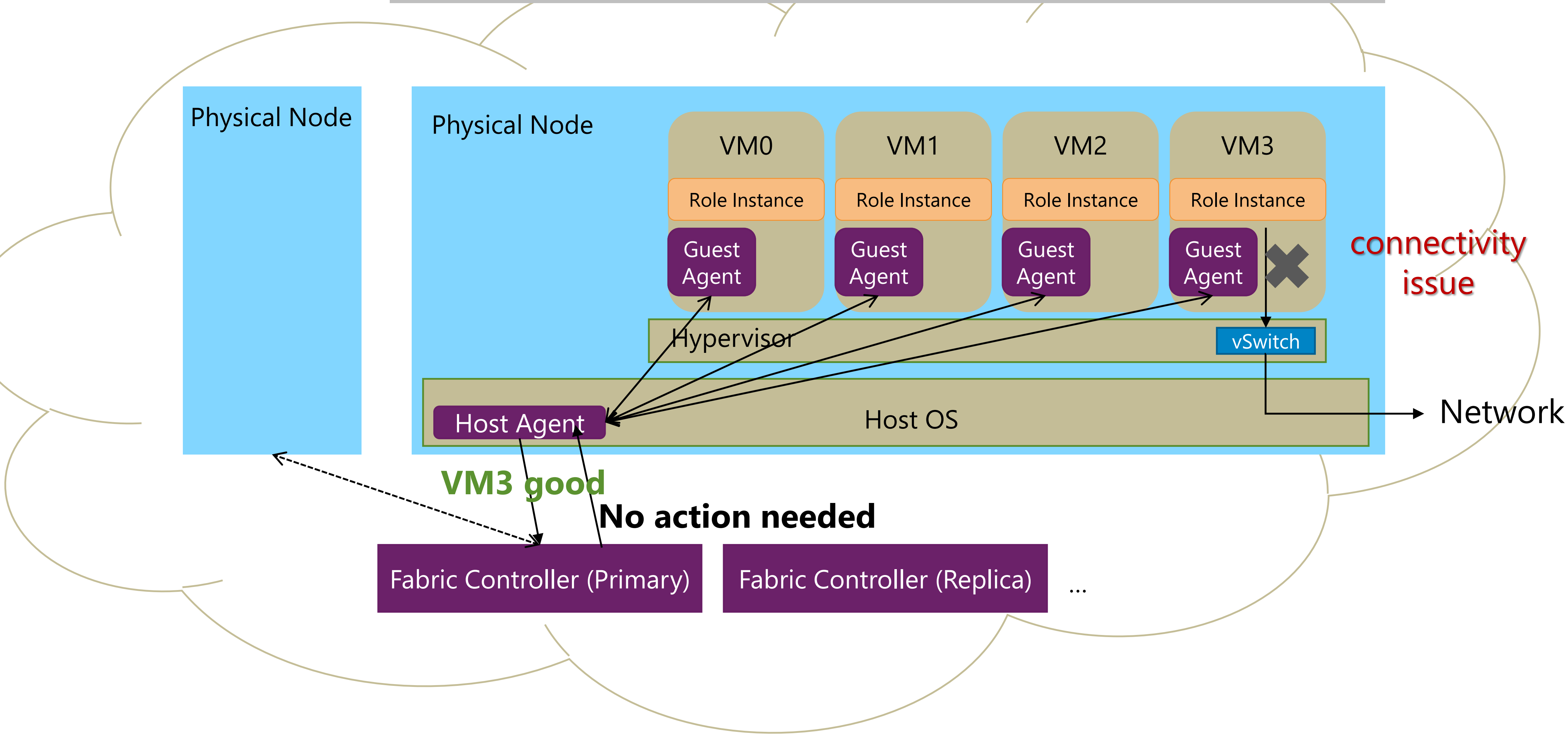
Case II: Failure Detector in Compute Service

Hierarchical agents to catch failure in different layers



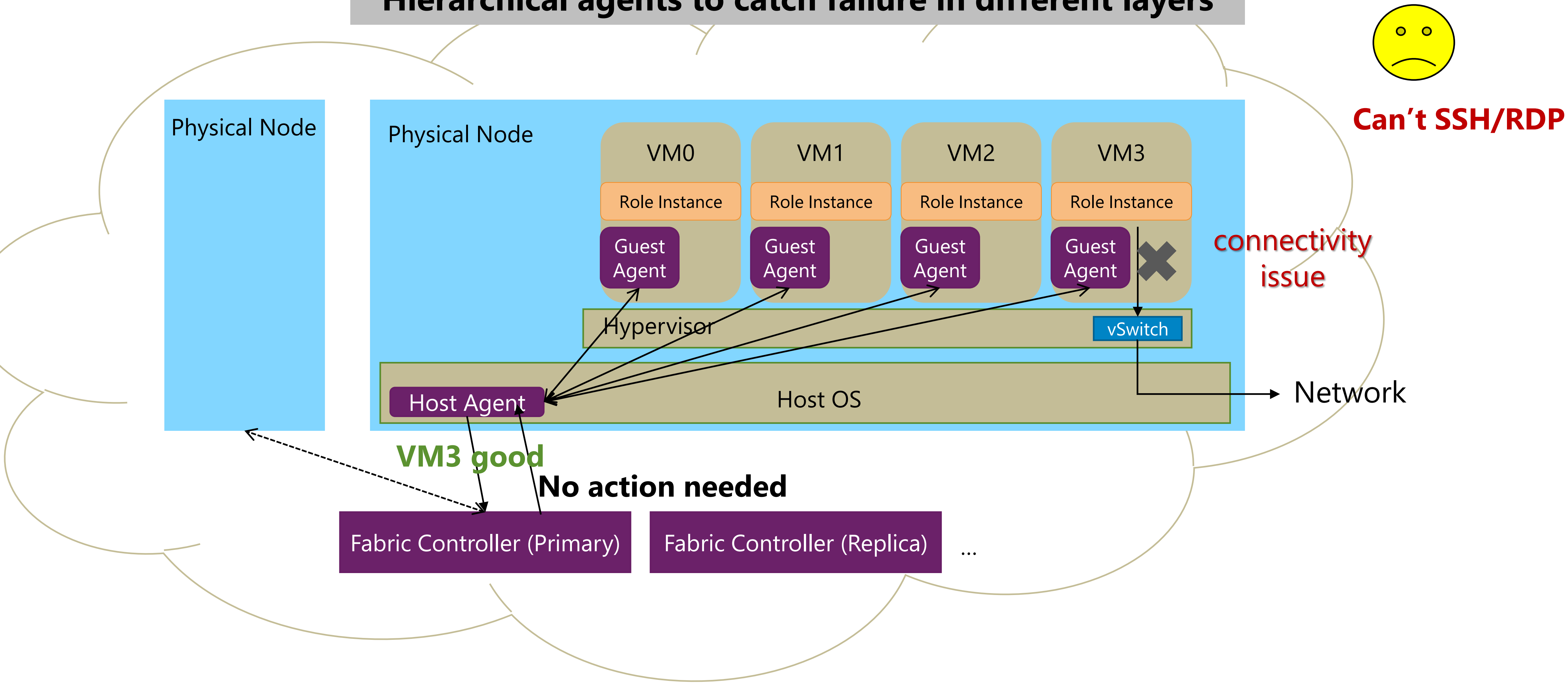
Case II: Failure Detector in Compute Service

Hierarchical agents to catch failure in different layers

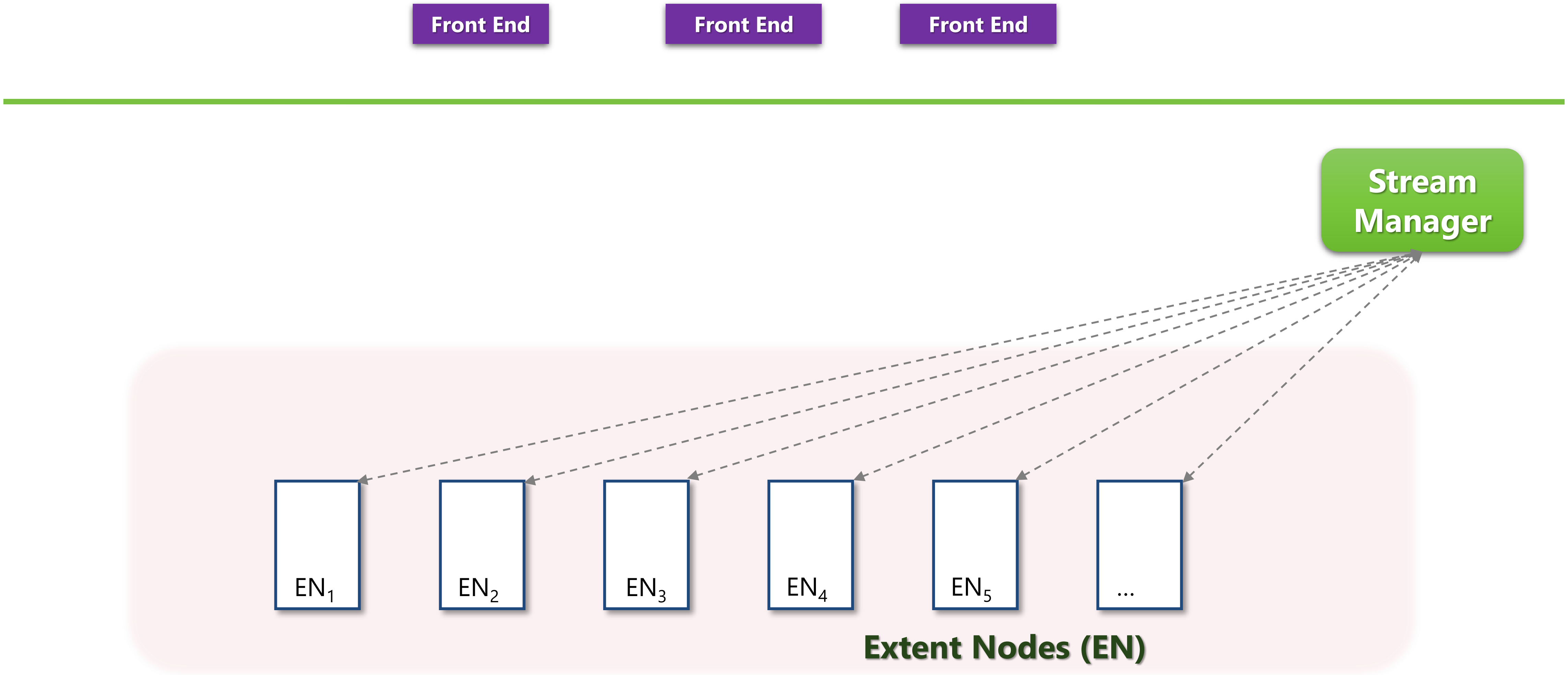


Case II: Failure Detector in Compute Service

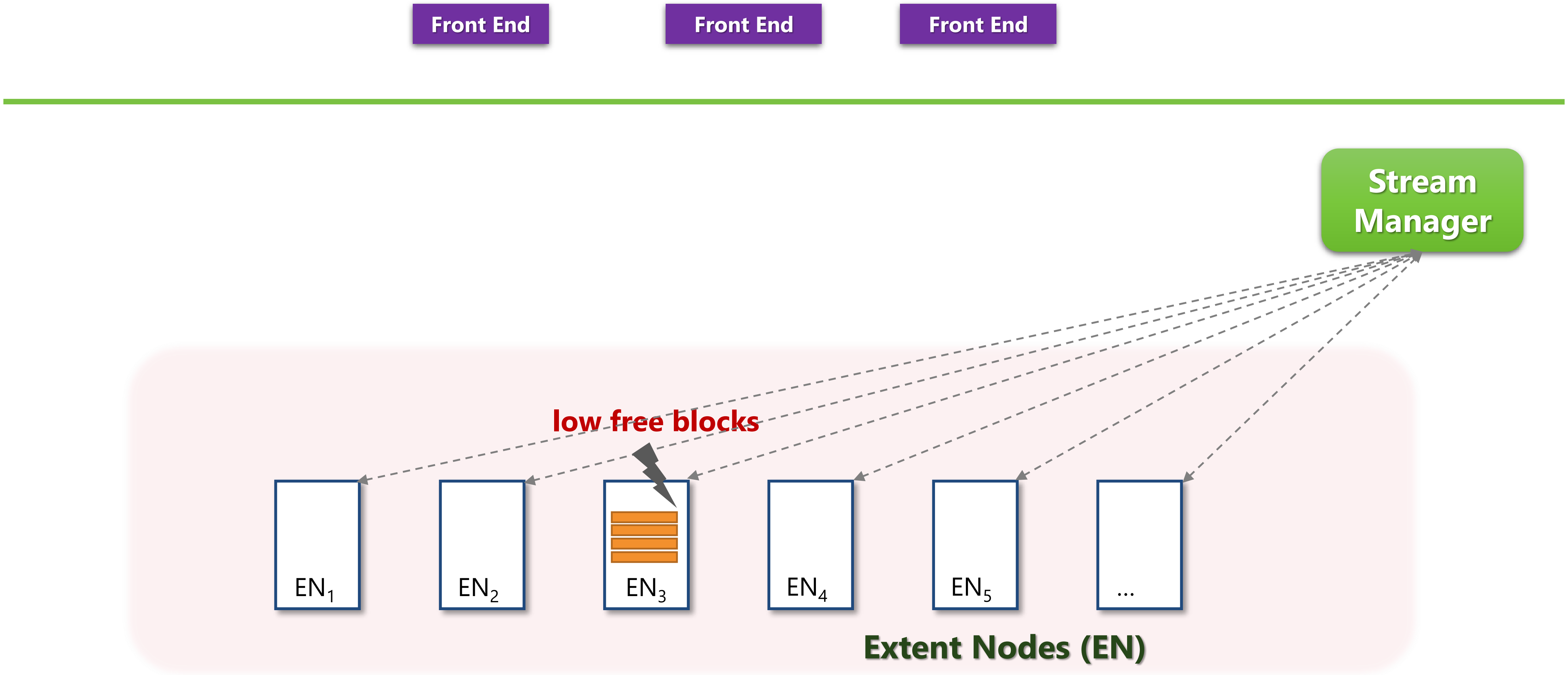
Hierarchical agents to catch failure in different layers



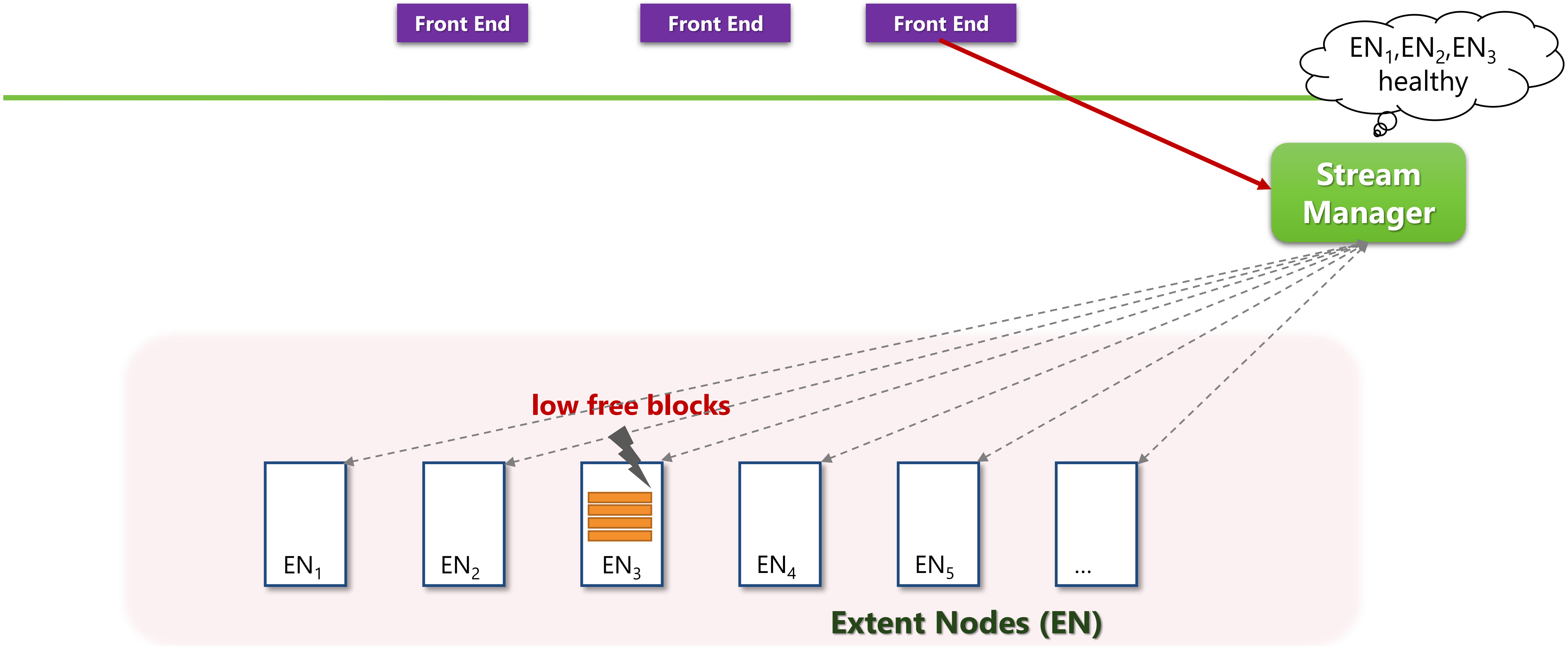
Case III: Recovery in Storage Service



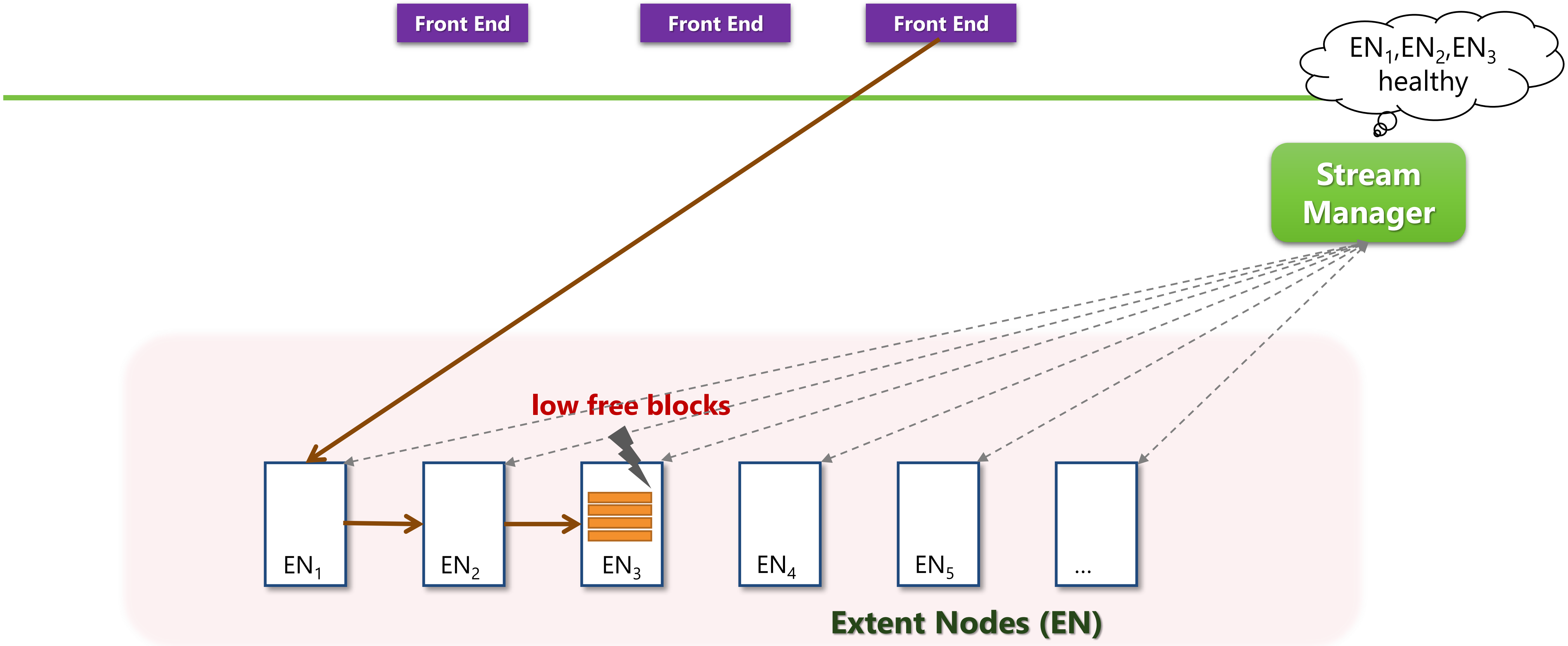
Case III: Recovery in Storage Service



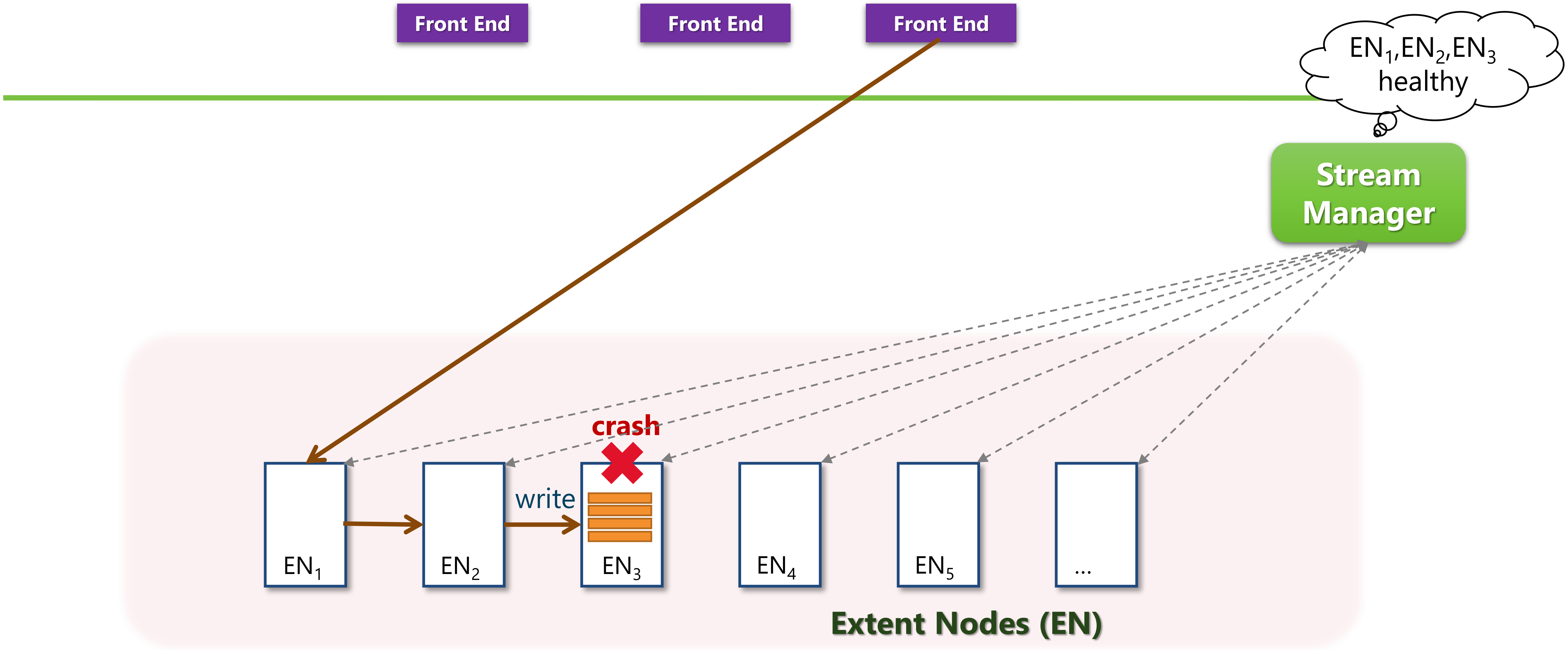
Case III: Recovery in Storage Service



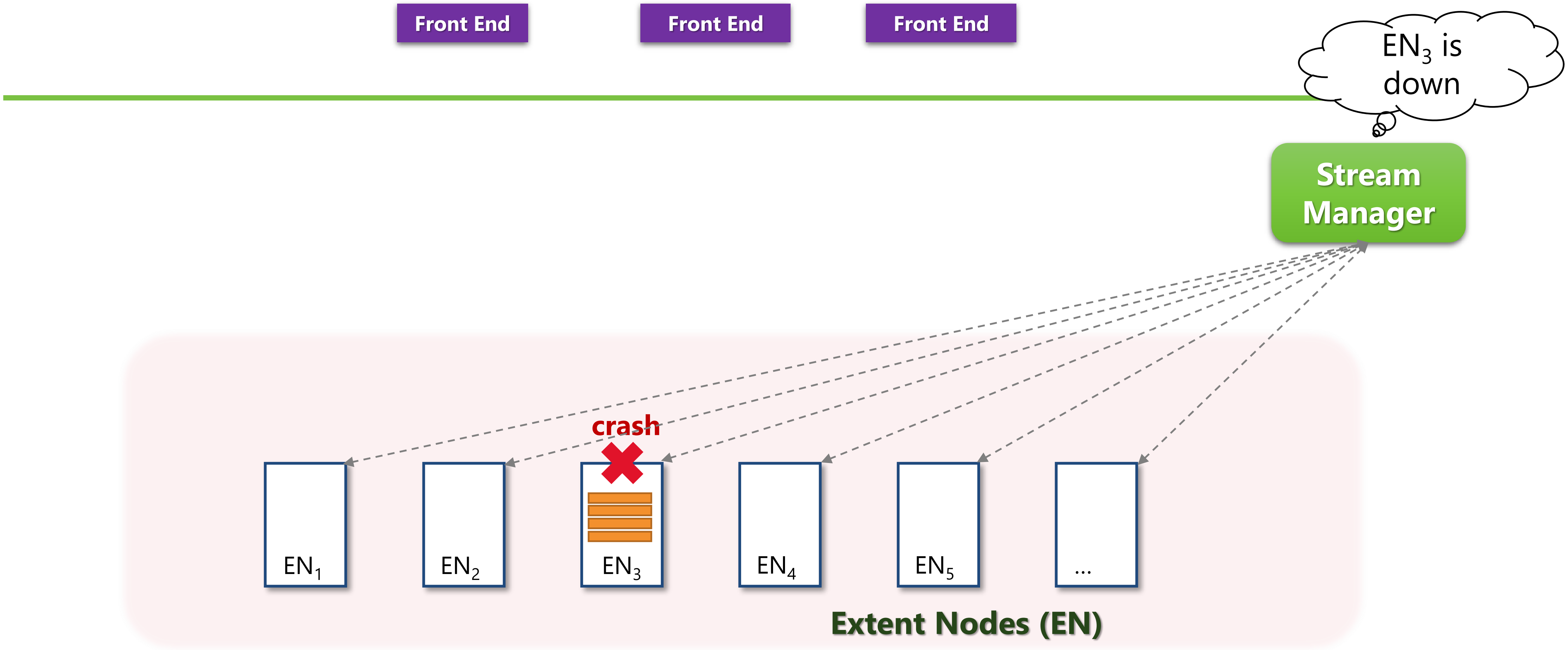
Case III: Recovery in Storage Service



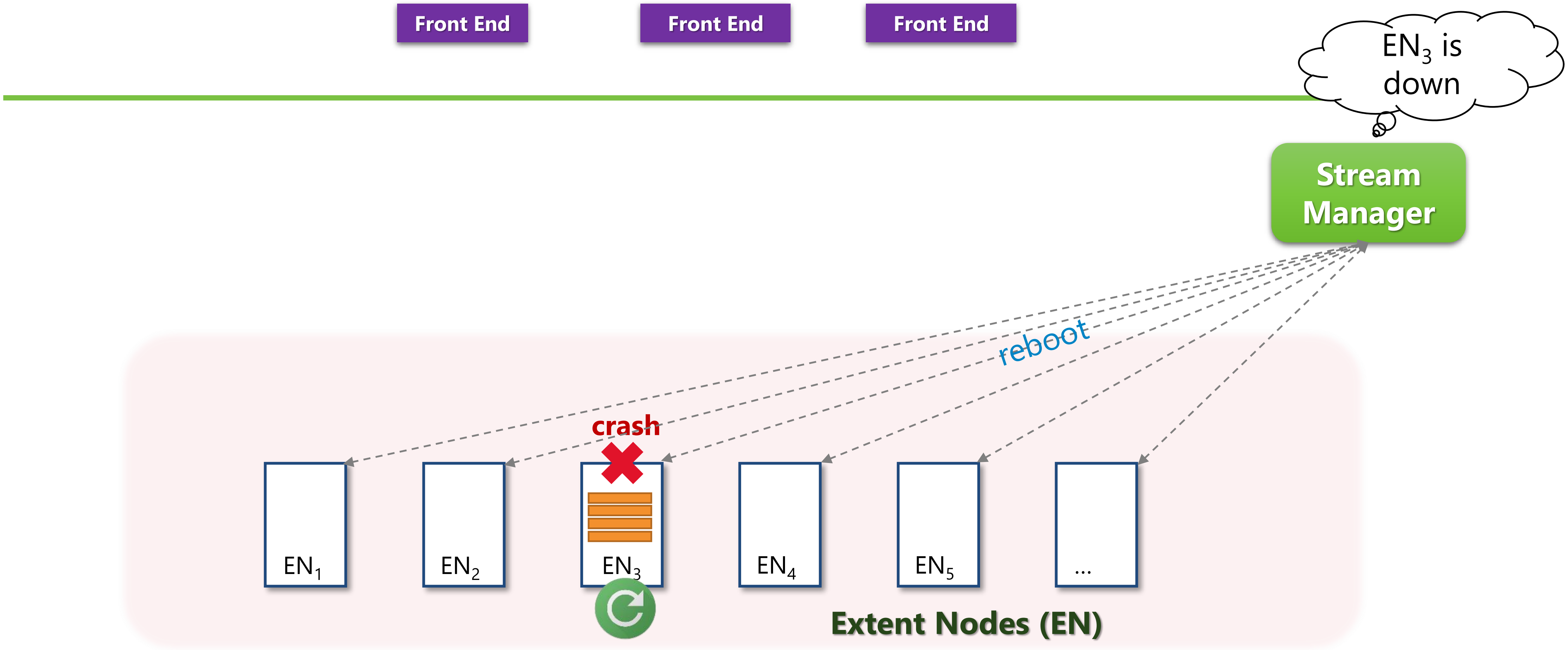
Case III: Recovery in Storage Service



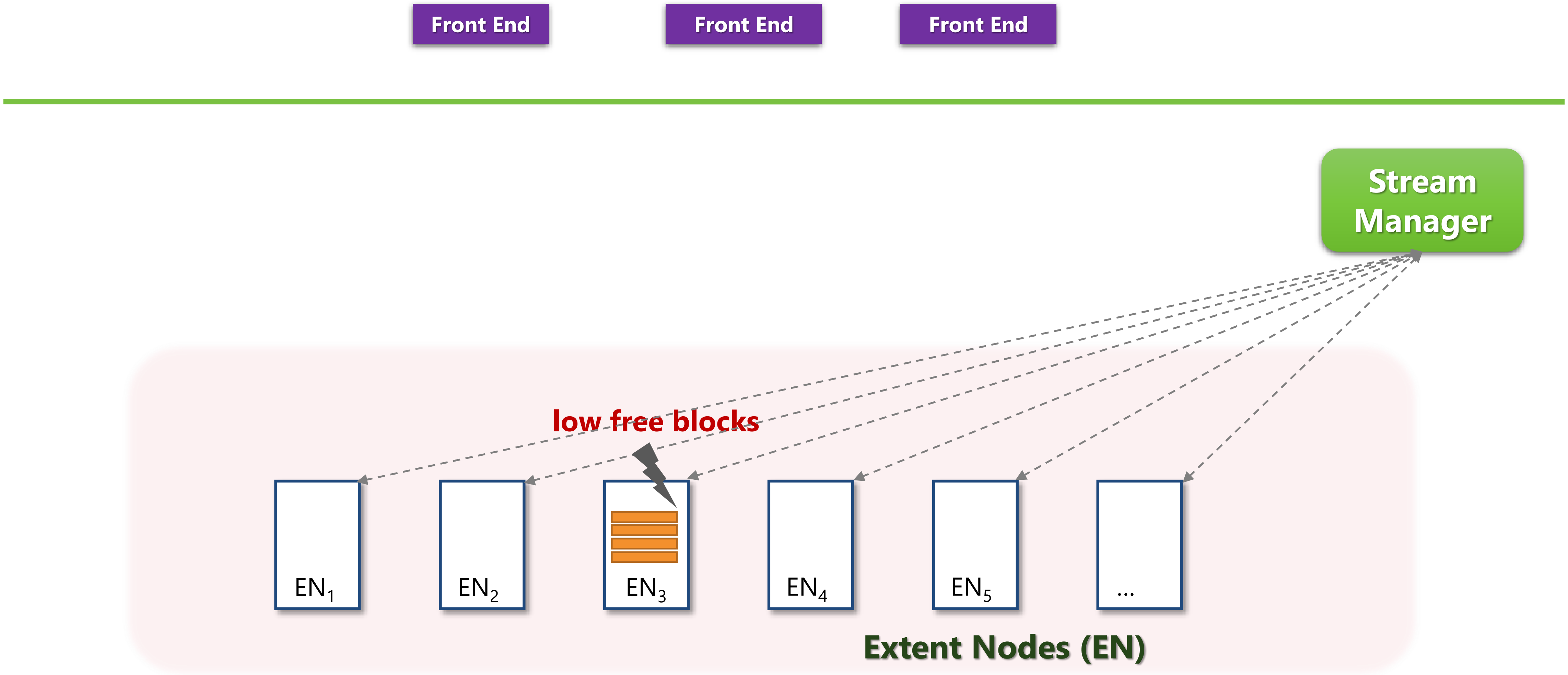
Case III: Recovery in Storage Service



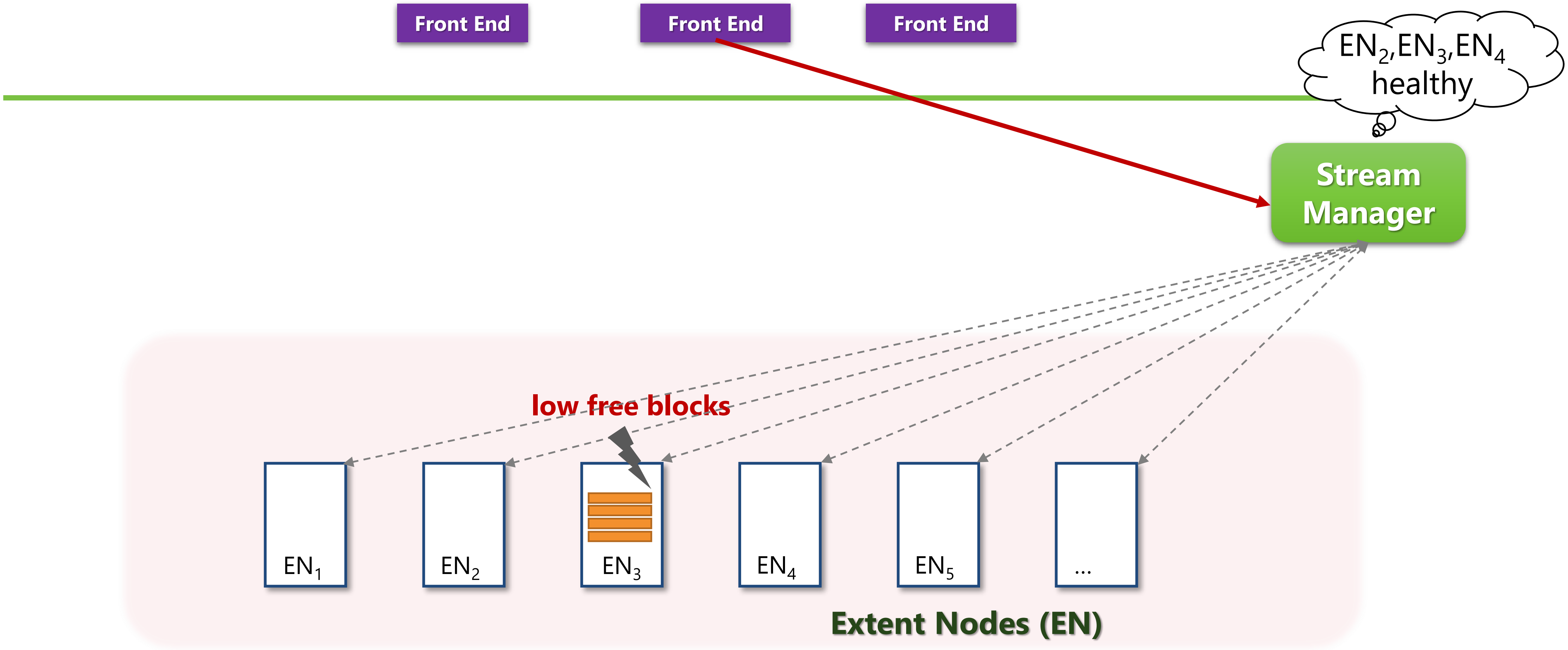
Case III: Recovery in Storage Service



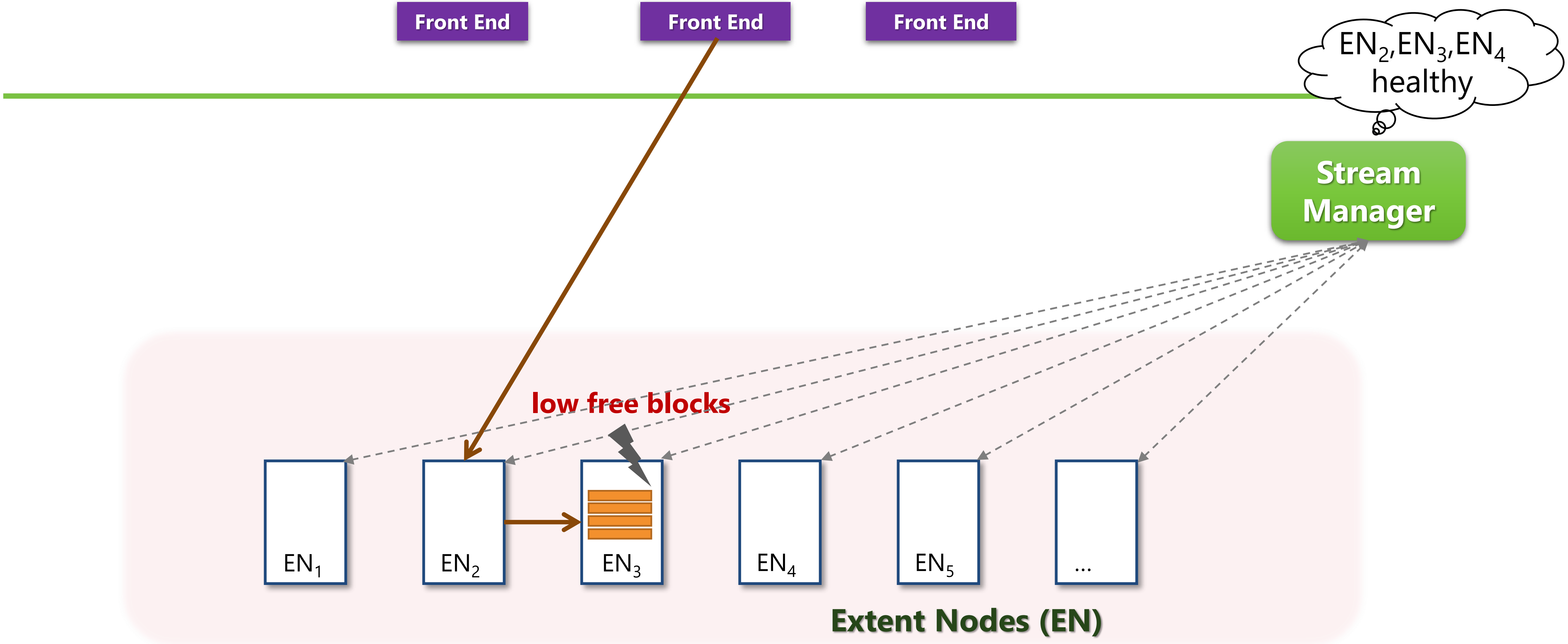
Case III: Recovery in Storage Service



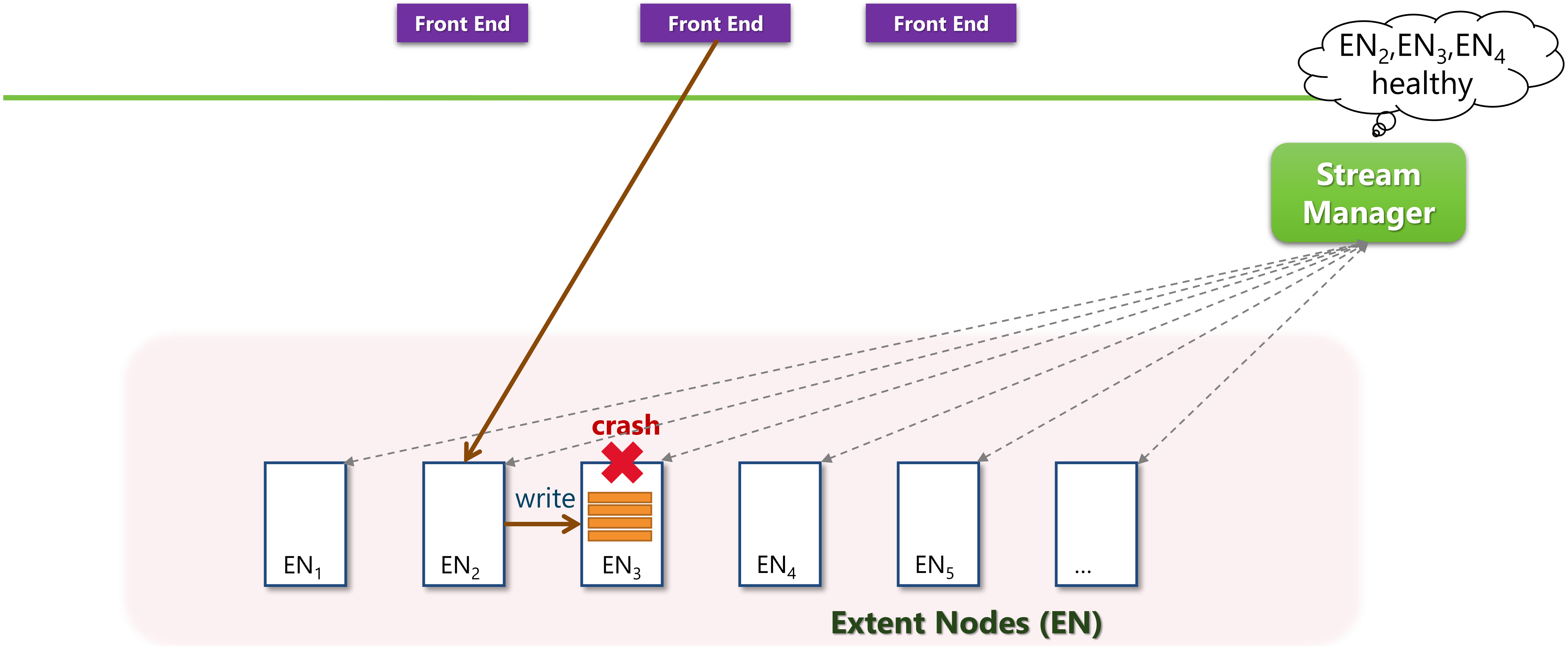
Case III: Recovery in Storage Service



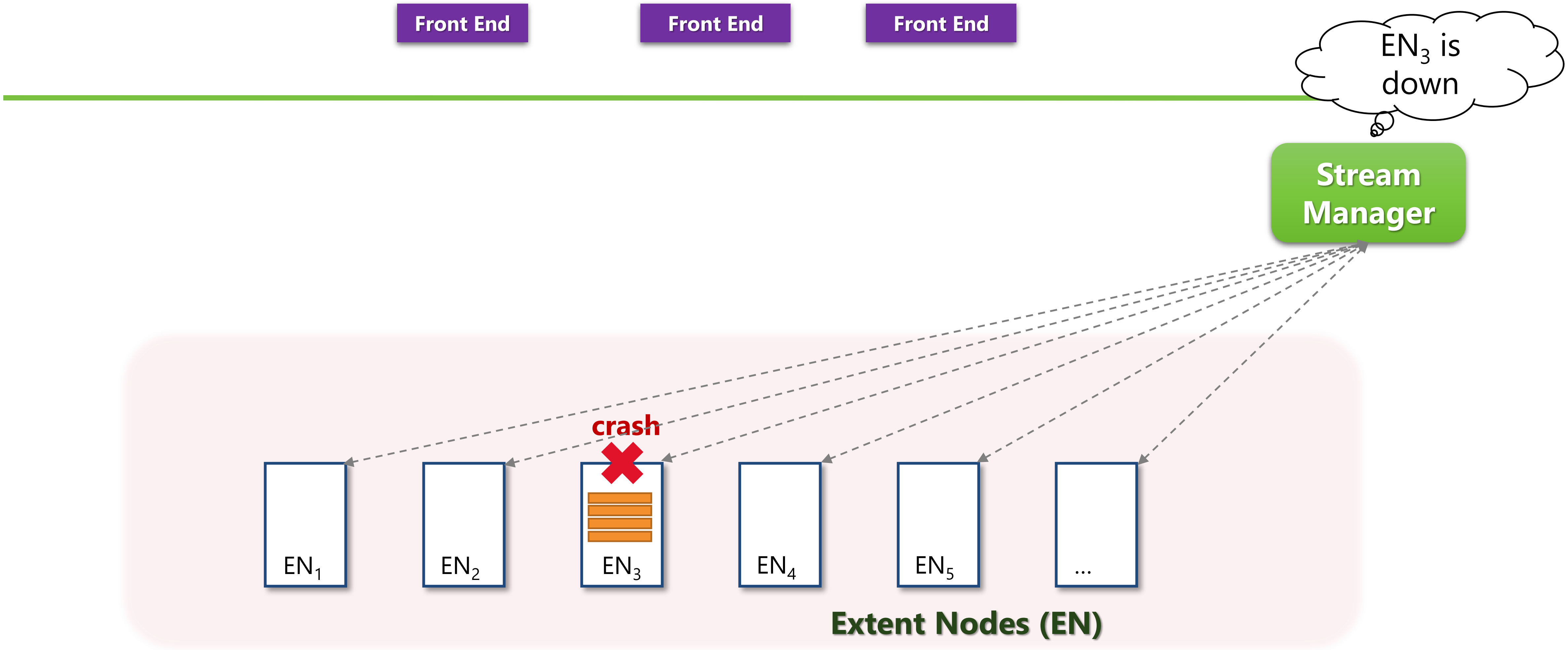
Case III: Recovery in Storage Service



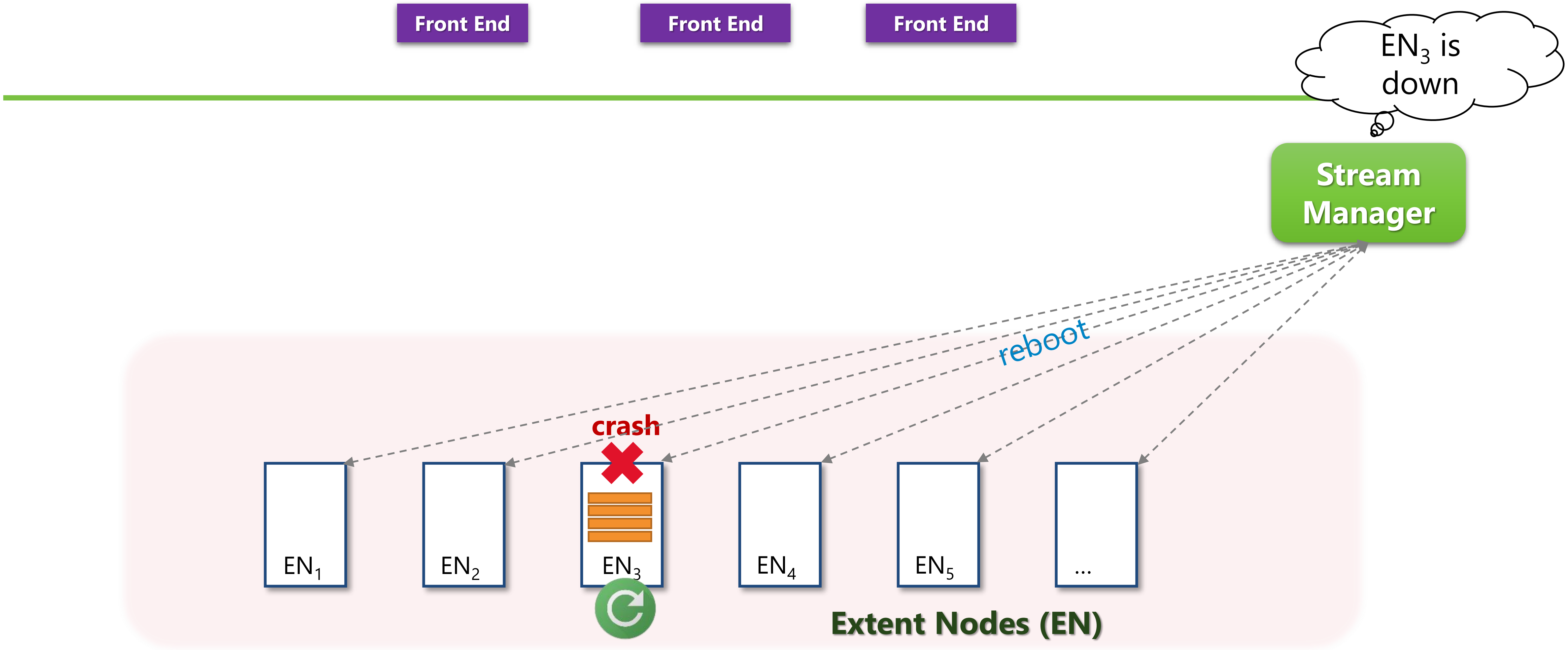
Case III: Recovery in Storage Service



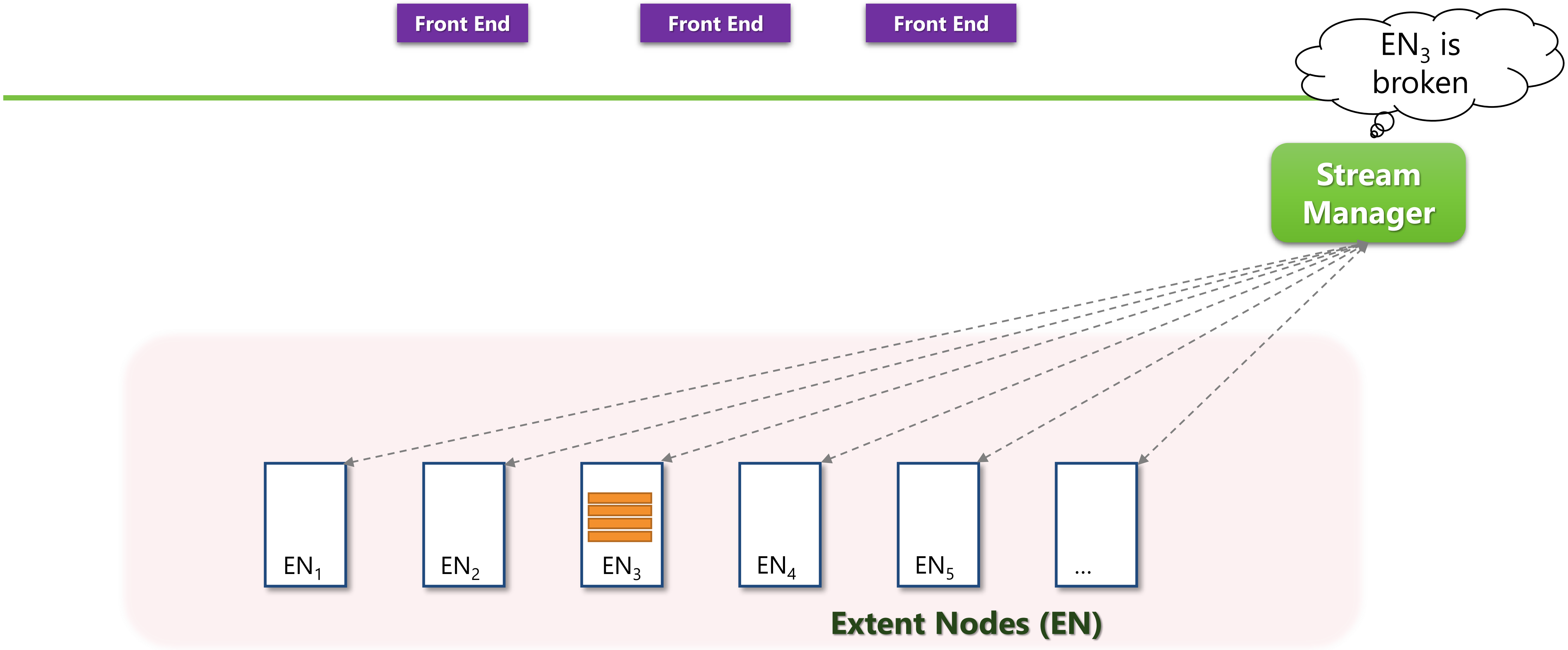
Case III: Recovery in Storage Service



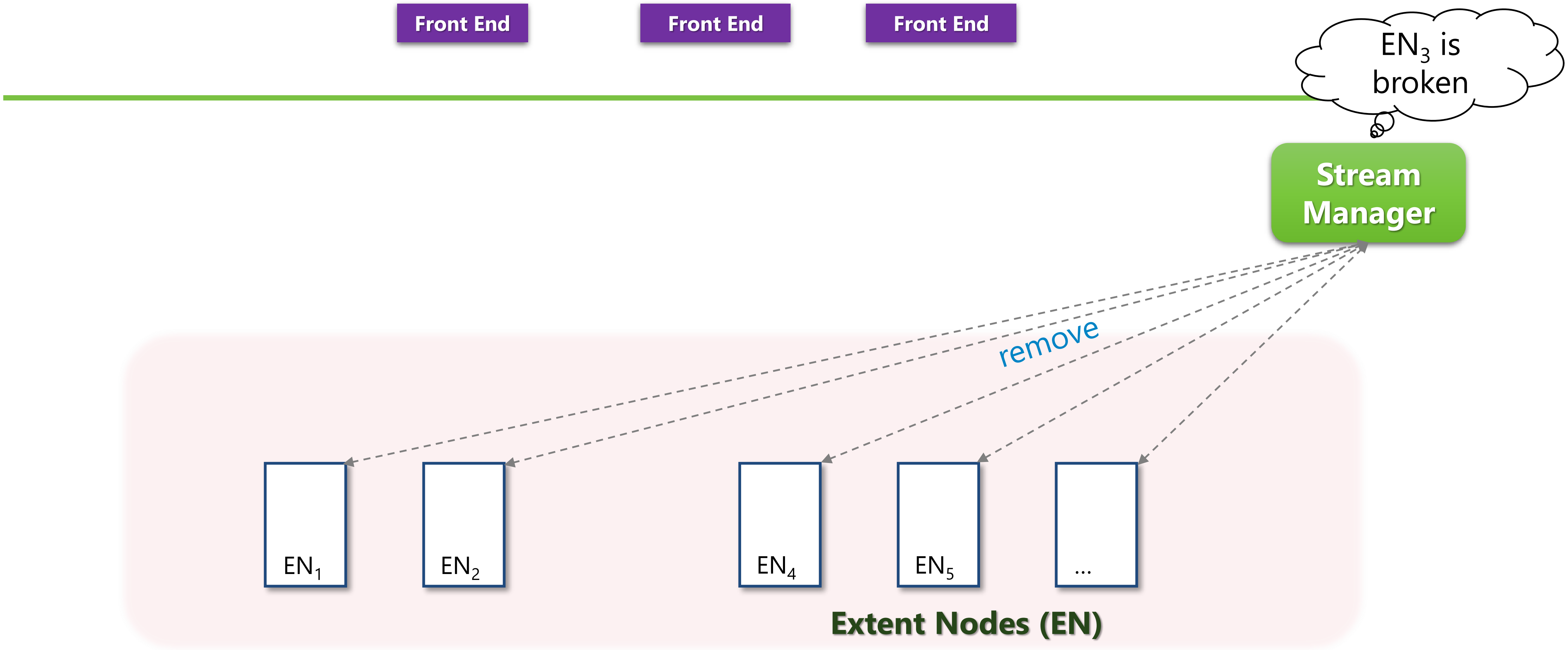
Case III: Recovery in Storage Service



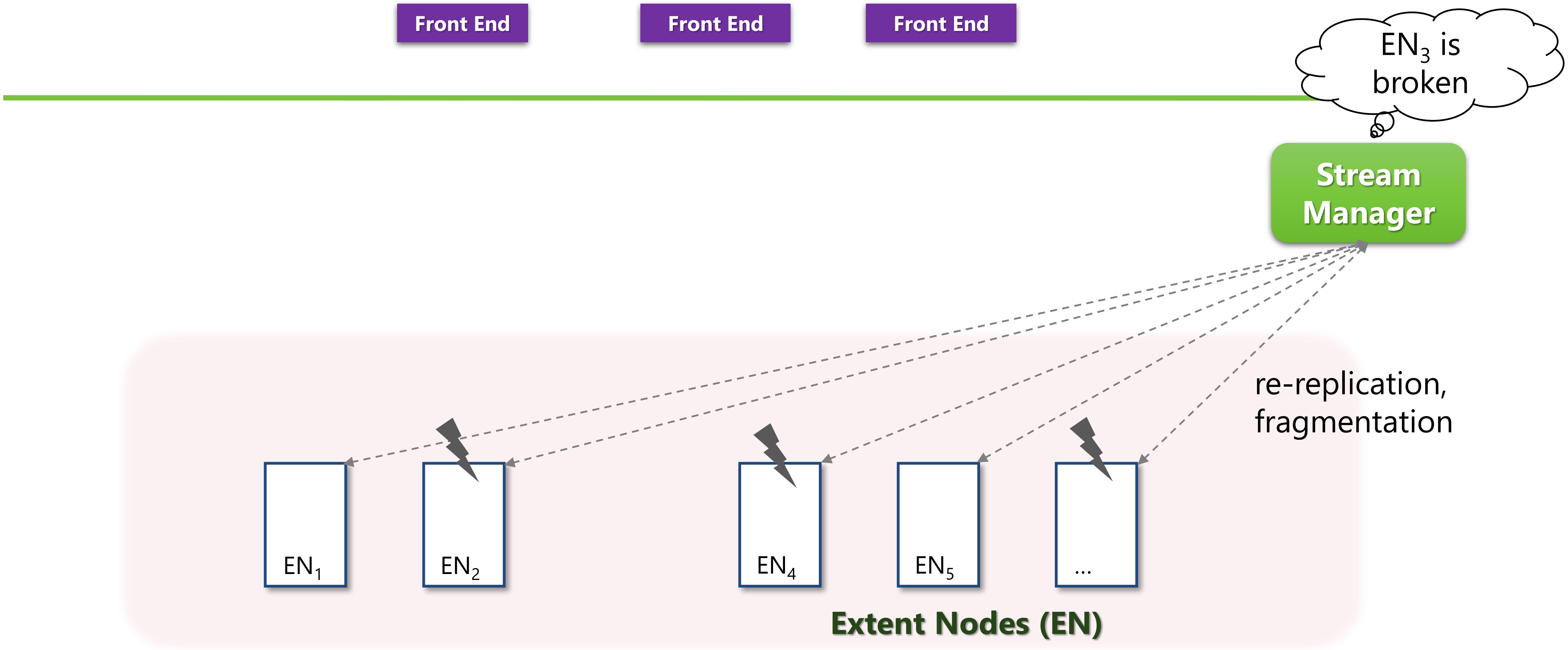
Case III: Recovery in Storage Service



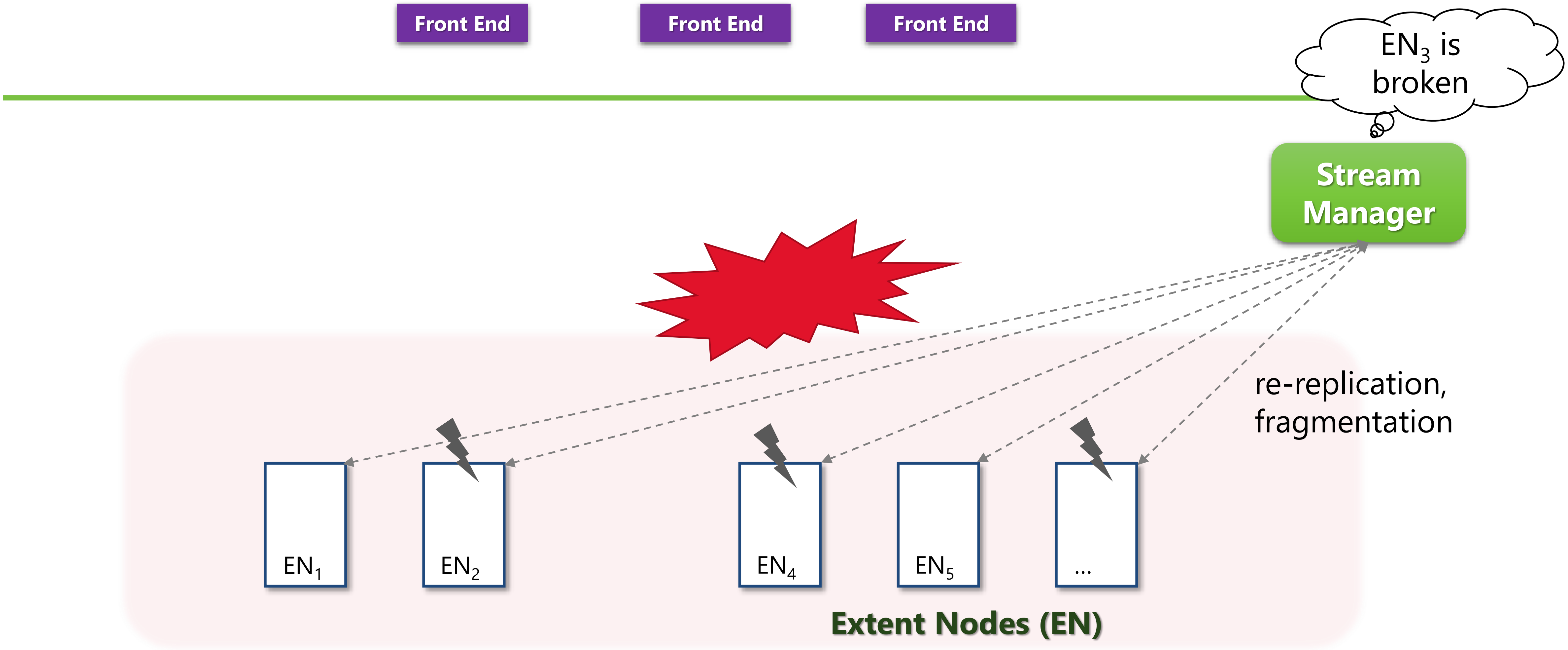
Case III: Recovery in Storage Service



Case III: Recovery in Storage Service



Case III: Recovery in Storage Service



Understanding Gray Failure

The Many Faces of Gray Failure

So, what is a gray failure?

The Many Faces of Gray Failure

So, what is a gray failure?

“ A performance issue. ”

The Many Faces of Gray Failure

So, what is a gray failure?

“ A performance issue. ”

“ A problem that some thinks is a failure but some thinks is not, e.g., a 2% packet loss. The ambiguity itself defines gray failure. If everyone agrees it is a problem, it is not a gray failure. ”

The Many Faces of Gray Failure

So, what is a gray failure?

“ A performance issue. ”

“ A problem that some thinks is a failure but some thinks is not, e.g., a 2% packet loss. The ambiguity itself defines gray failure. If everyone agrees it is a problem, it is not a gray failure. ”

“ A Heisenbug, sometimes it occurs and sometimes it does not. ”

The Many Faces of Gray Failure

So, what is a gray failure?

“ A performance issue. ”

“ A problem that some thinks is a failure but some thinks is not, e.g., a 2% packet loss. The ambiguity itself defines gray failure. If everyone agrees it is a problem, it is not a gray failure. ”

“ A Heisenbug, sometimes it occurs and sometimes it does not. ”

“ The system is failing slowly, e.g., memory leak. ”

The Many Faces of Gray Failure

So, what is a gray failure?

> “A performance issue.”

> “A problem that some thinks is a failure but some thinks is not, e.g., a 2% packet loss. The ambiguity itself defines gray failure. If everyone agrees it is a problem, it is not a gray failure.”

> “A Heisenbug, sometimes it occurs and sometimes it does not.”

> “The system is failing slowly, e.g., memory leak.”

> “There is an increasing number of transient errors in the system, which results in reduced system capacity even if the system still manages to continue working.”

The Many Faces of Gray Failure

So, what is a gray failure?

“ A performance issue. ”

“ A pro ks is not,
e.g., a ay failure.
If eve re. ”

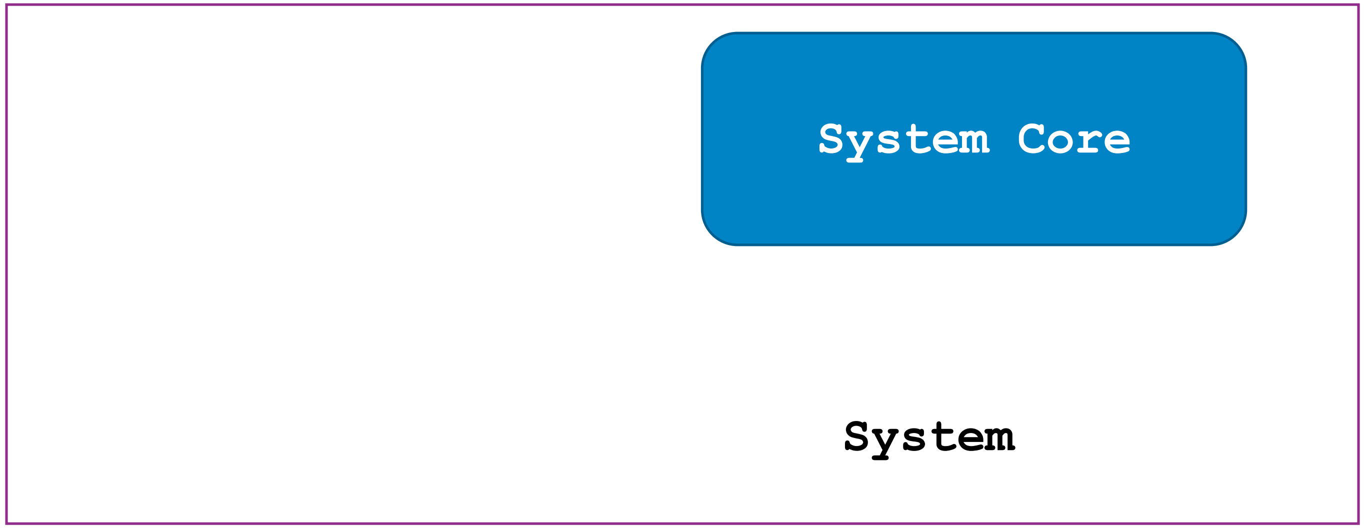
“ A He g oes not. ”

“ The system is failing slowly, e.g., memory leak. ”

“ There is an increasing number of transient errors in the system, which results in reduced system capacity even if the system still manages to continue working. ”

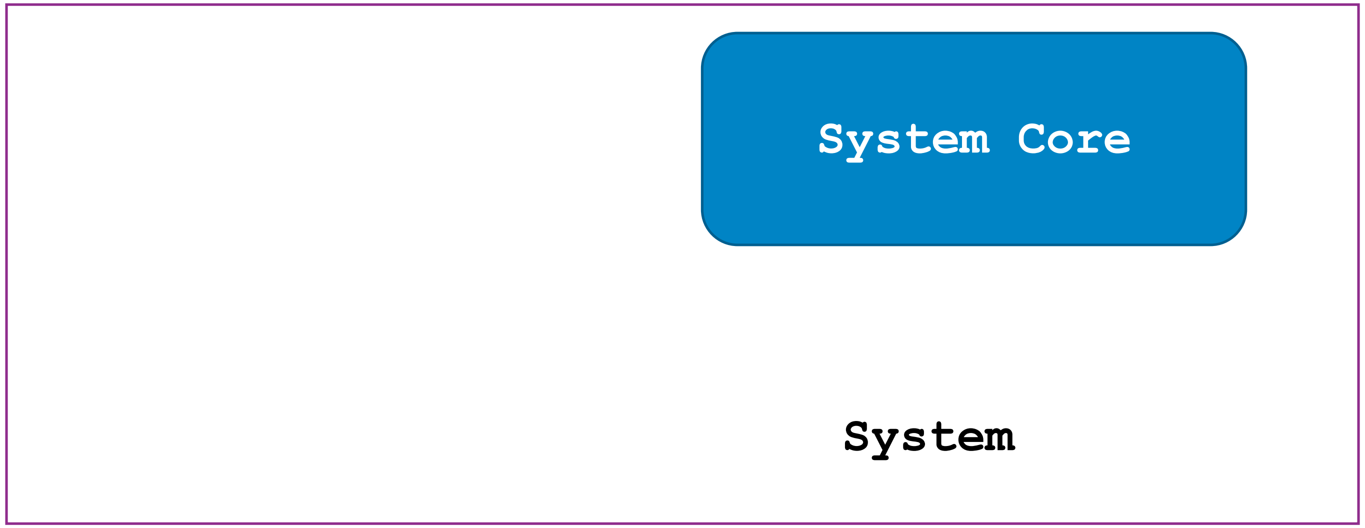
What is a formal way to define and study gray failure, one that potentially sheds light on how to address it?

An Abstract Model



Note: these are logical entities

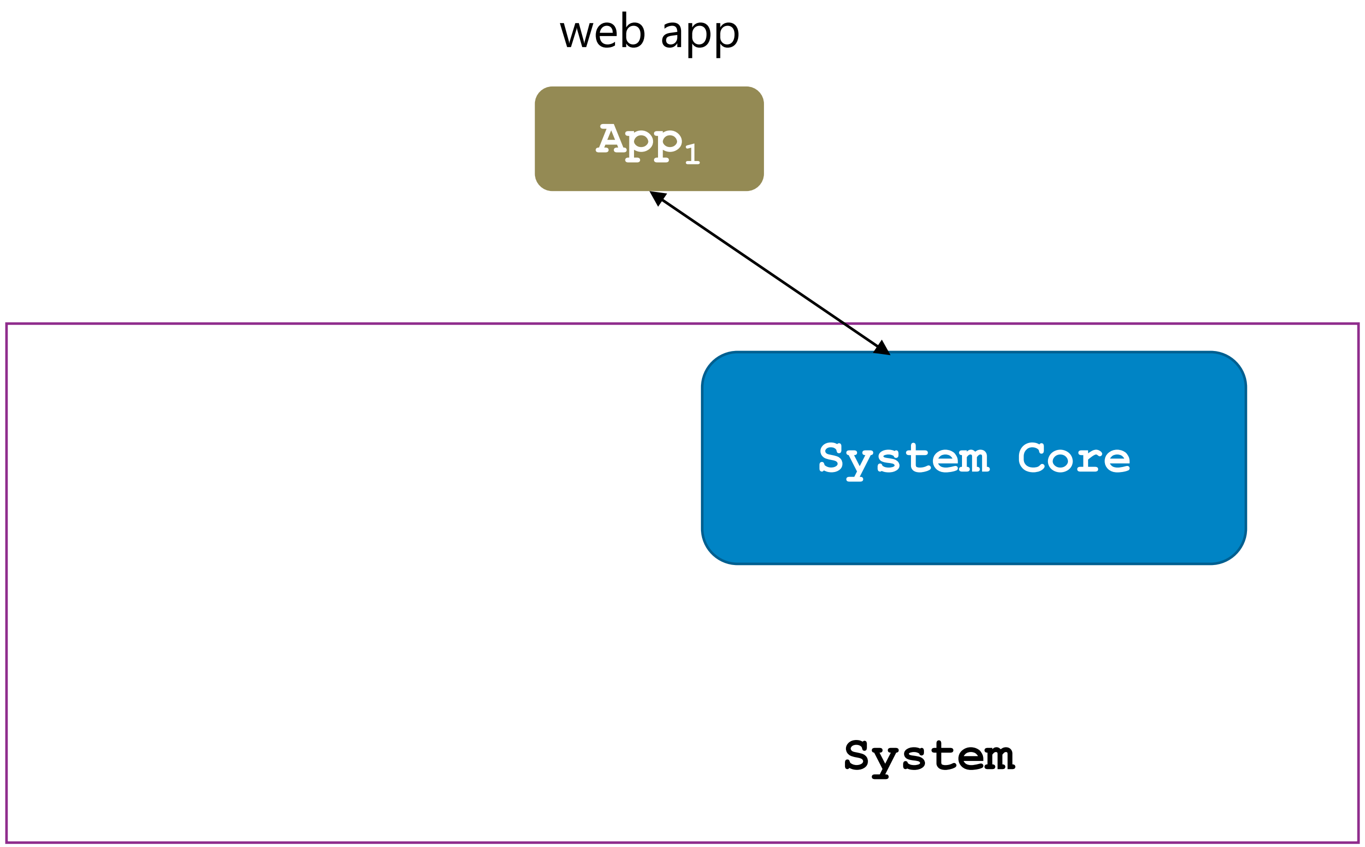
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

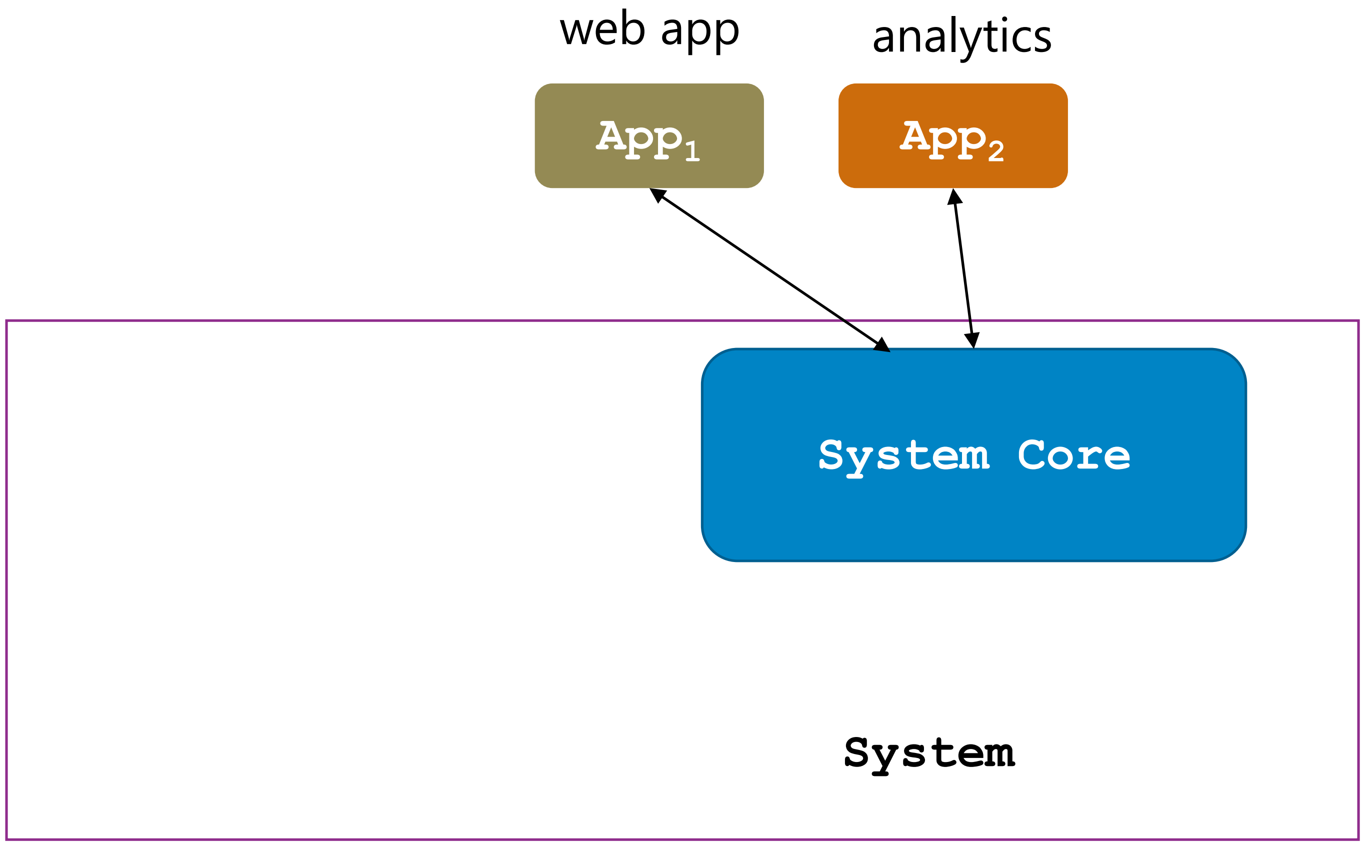
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

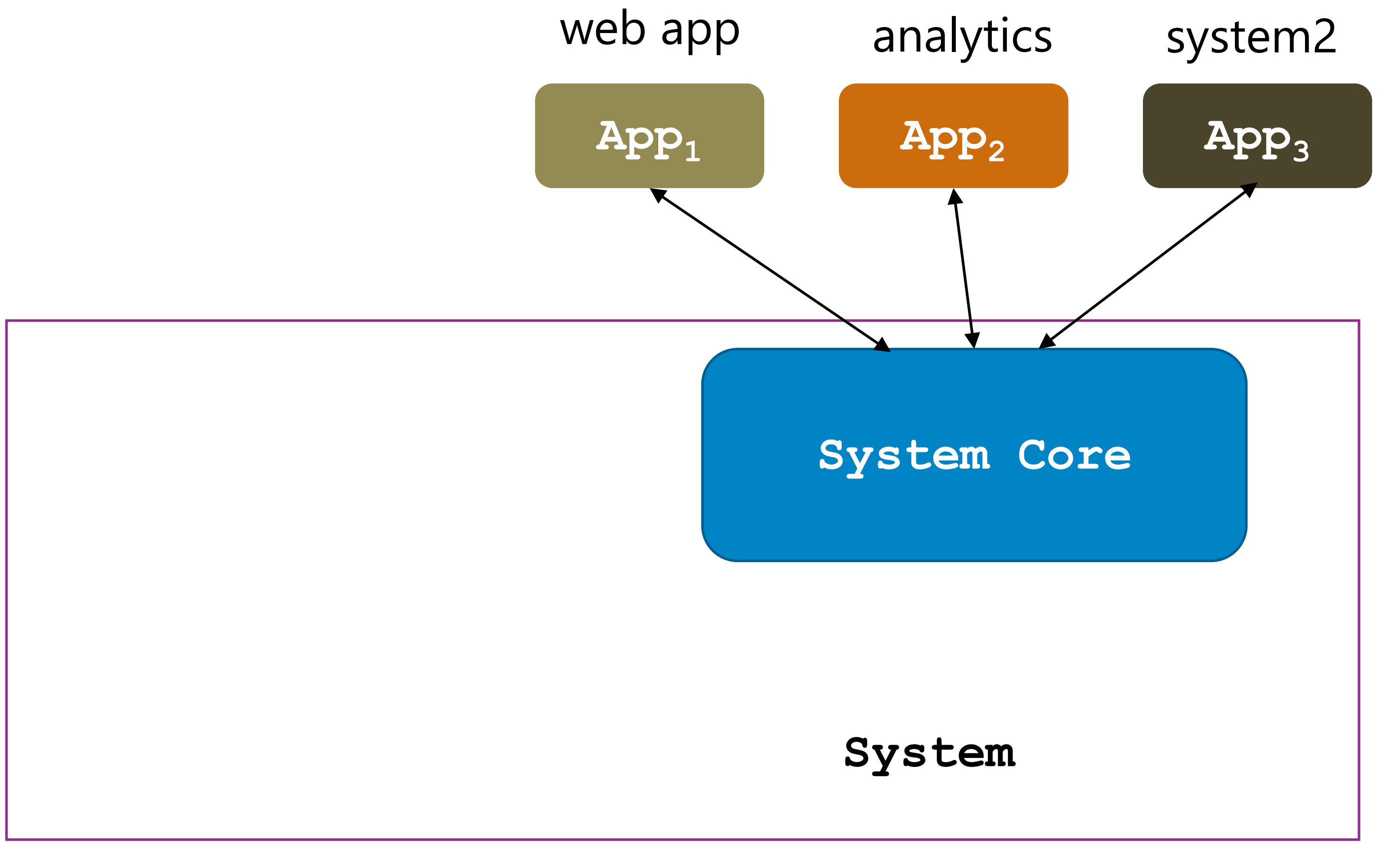
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

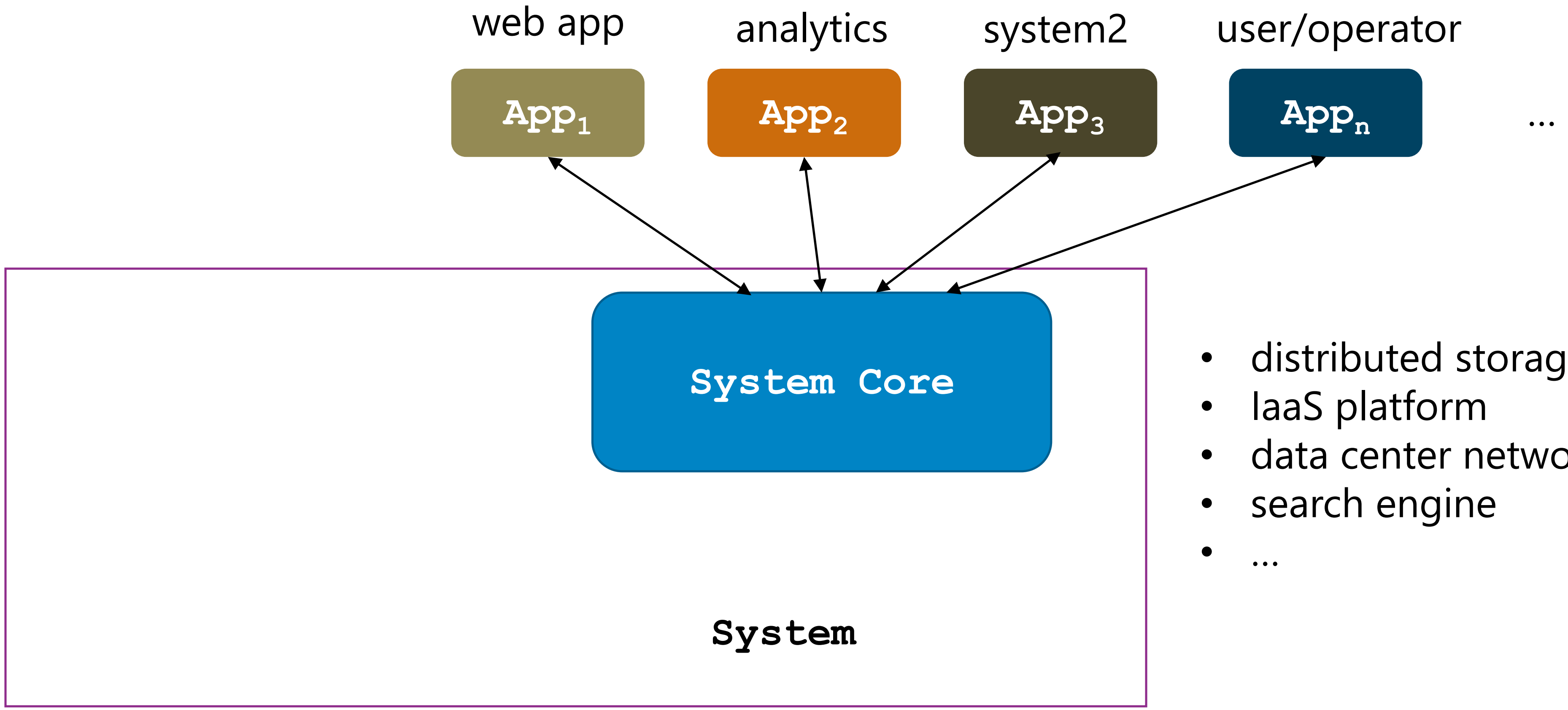
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

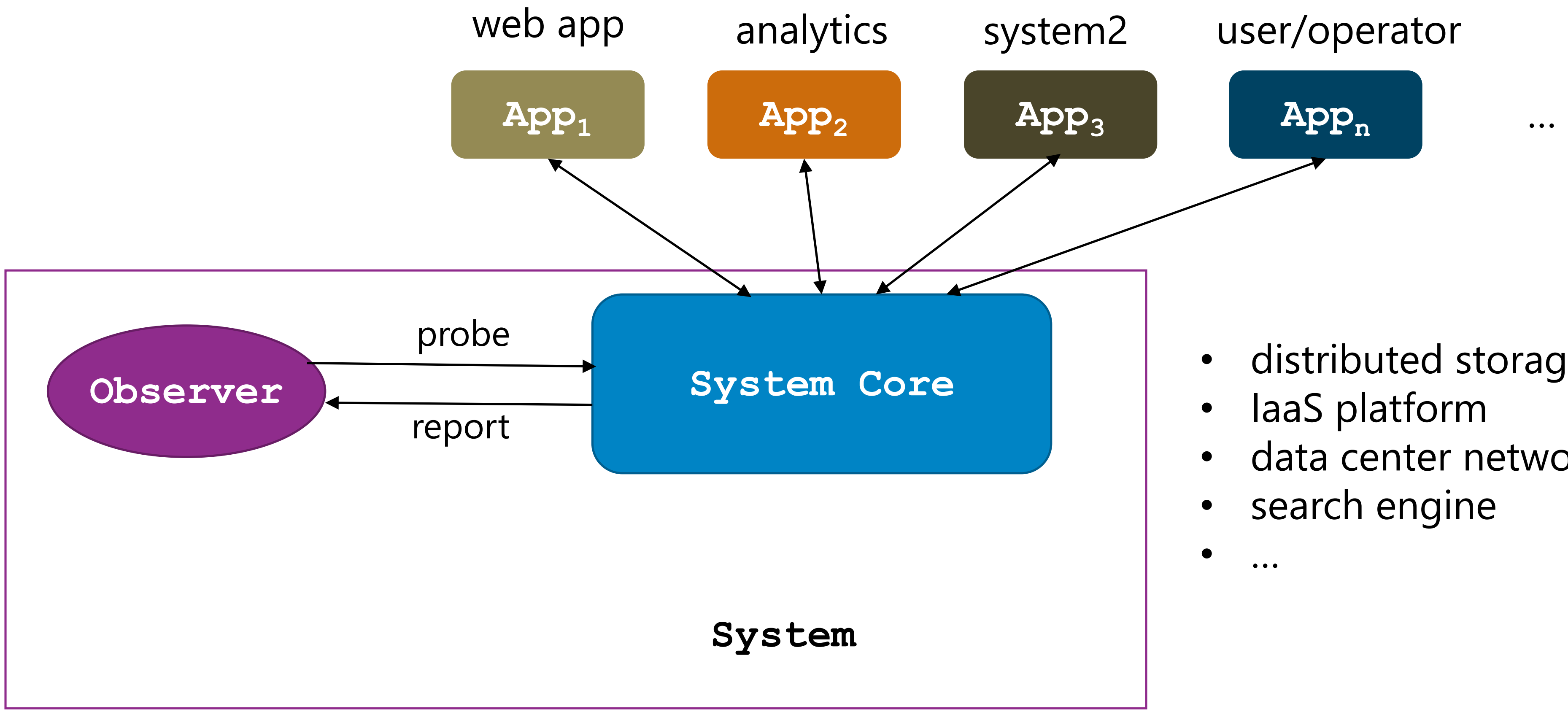
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

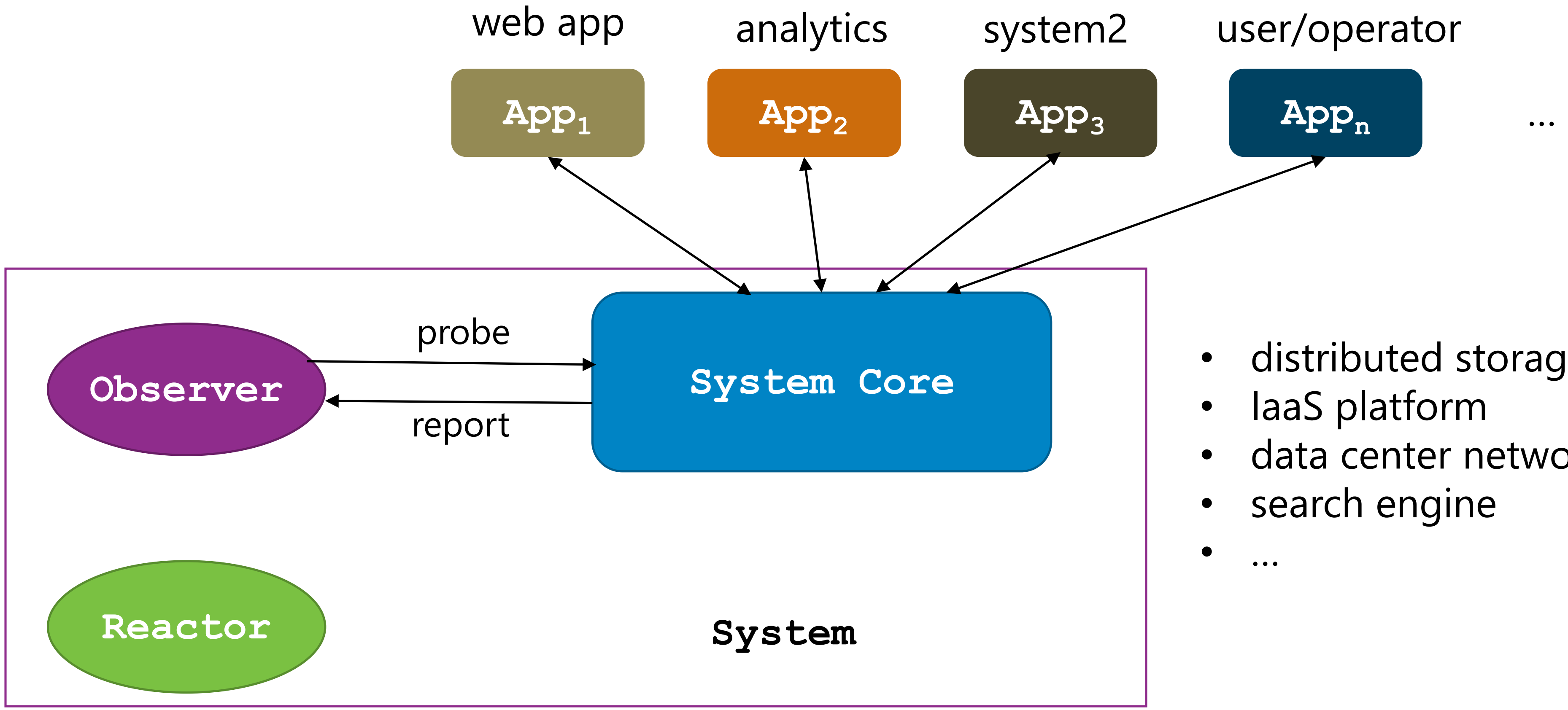
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

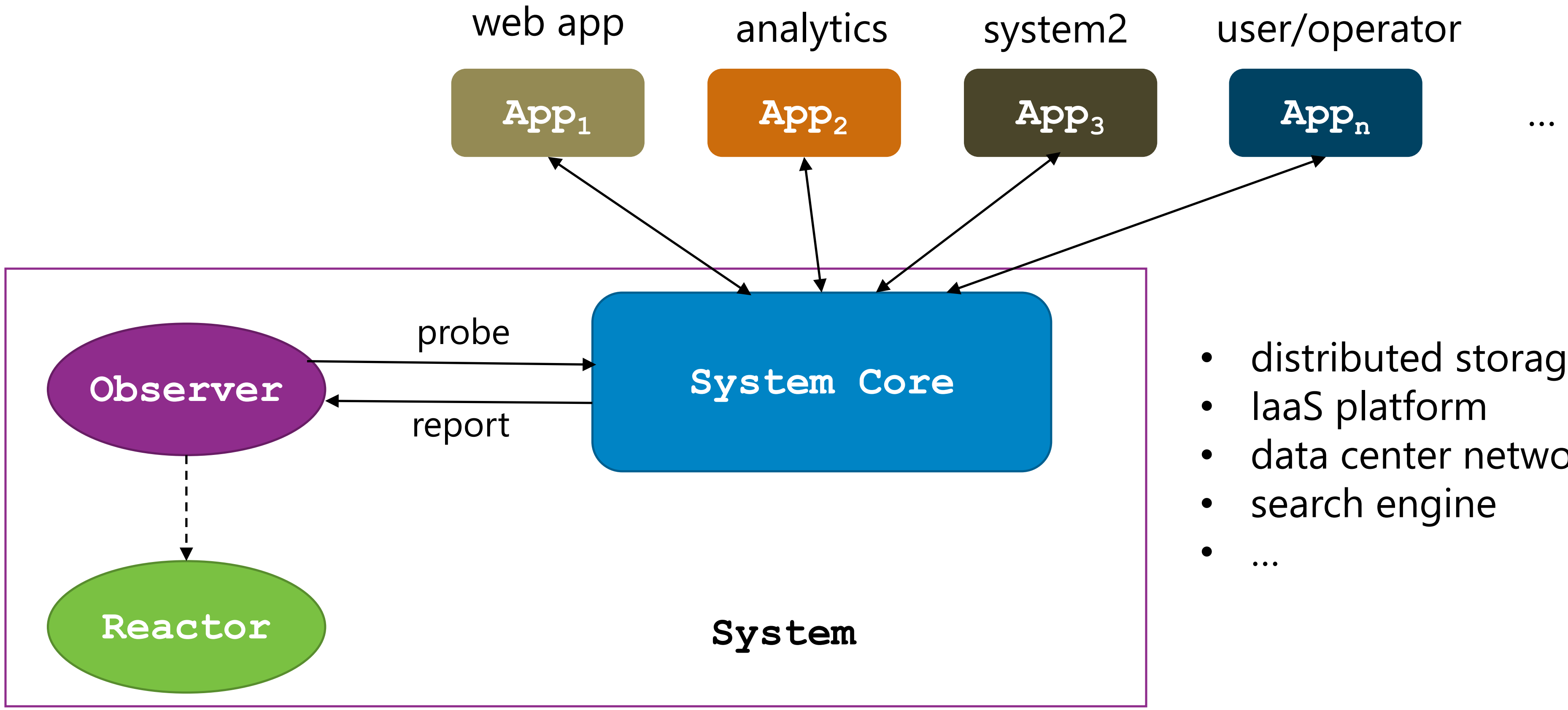
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

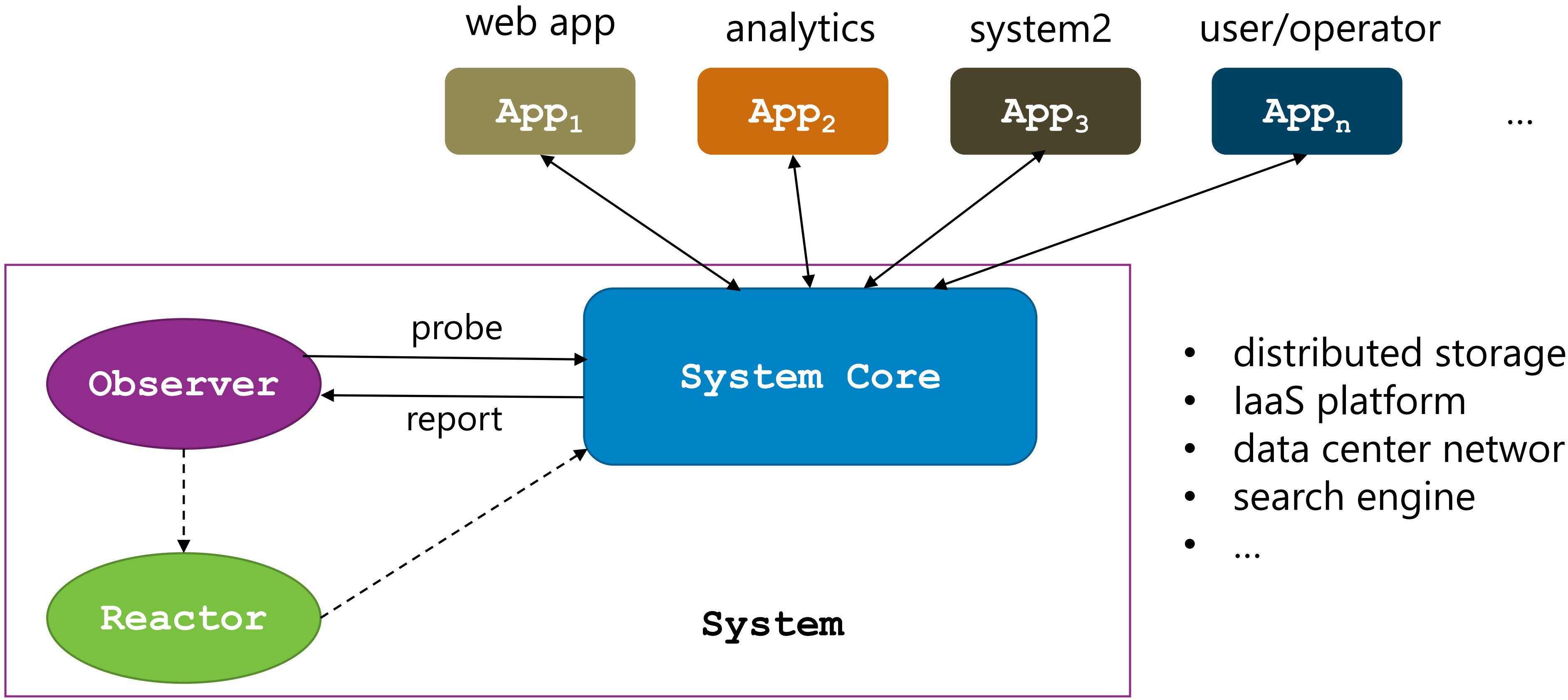
An Abstract Model



- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

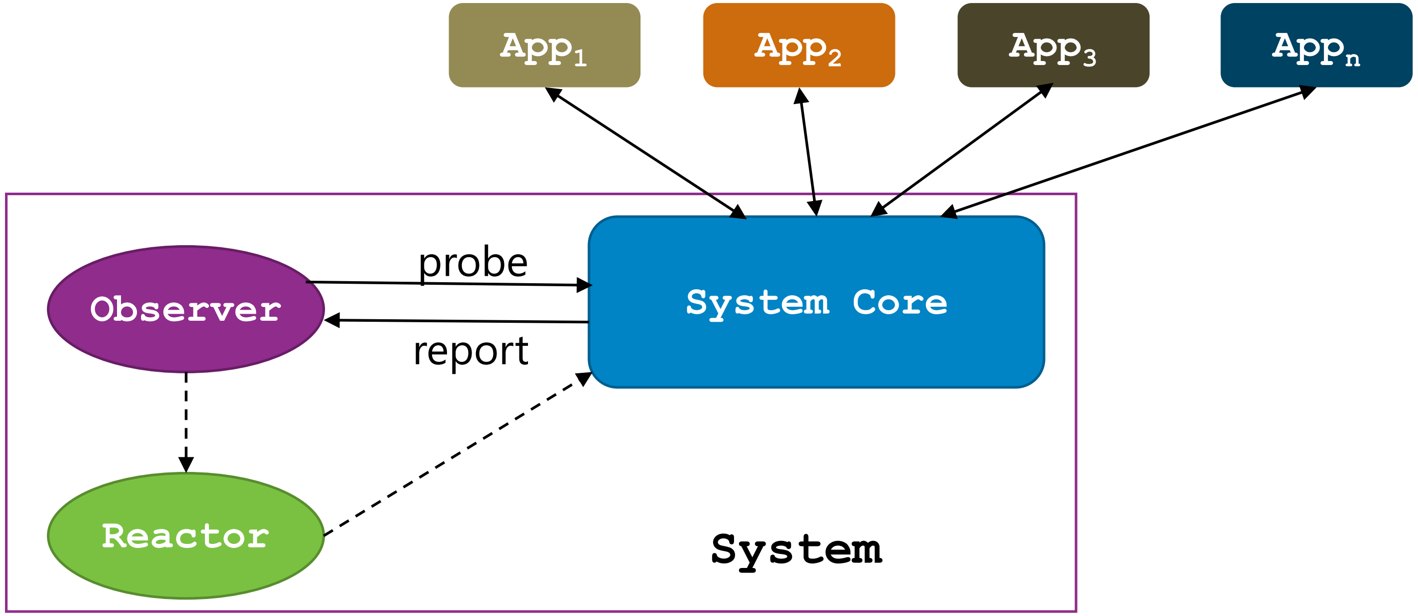
An Abstract Model



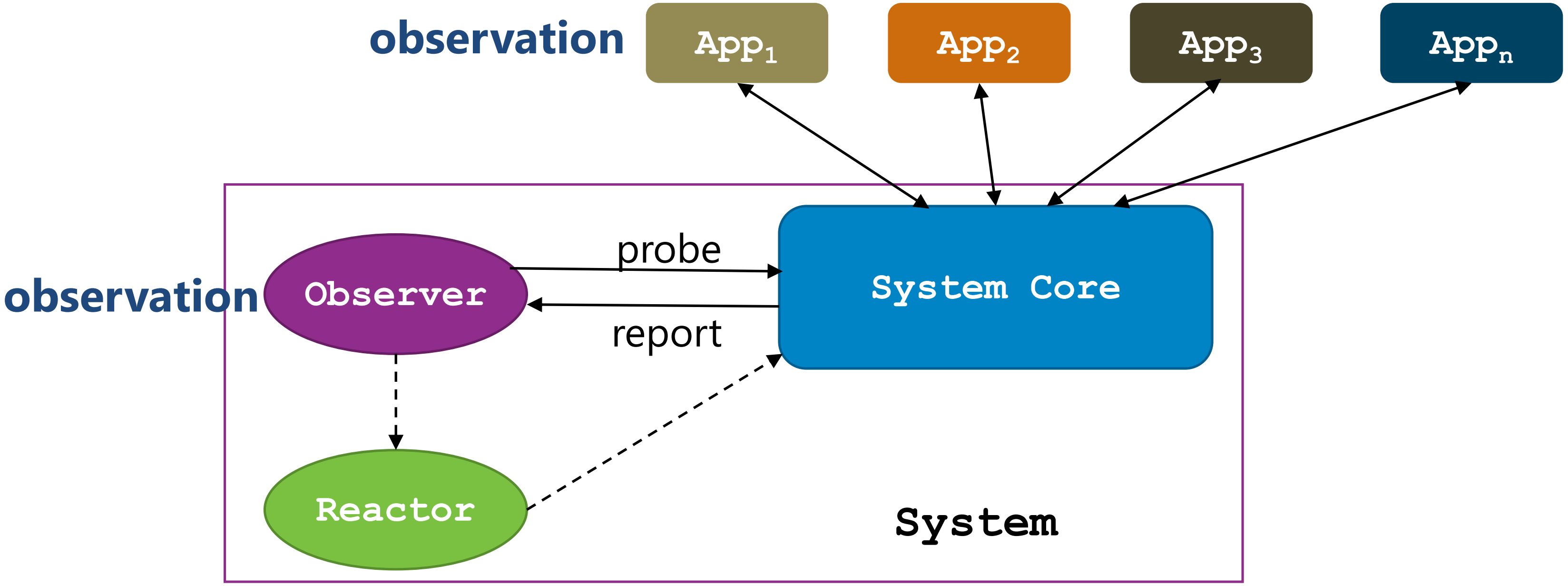
- distributed storage system
- IaaS platform
- data center network
- search engine
- …

Note: these are logical entities

Gray Failure Trait: *Differential Observability*

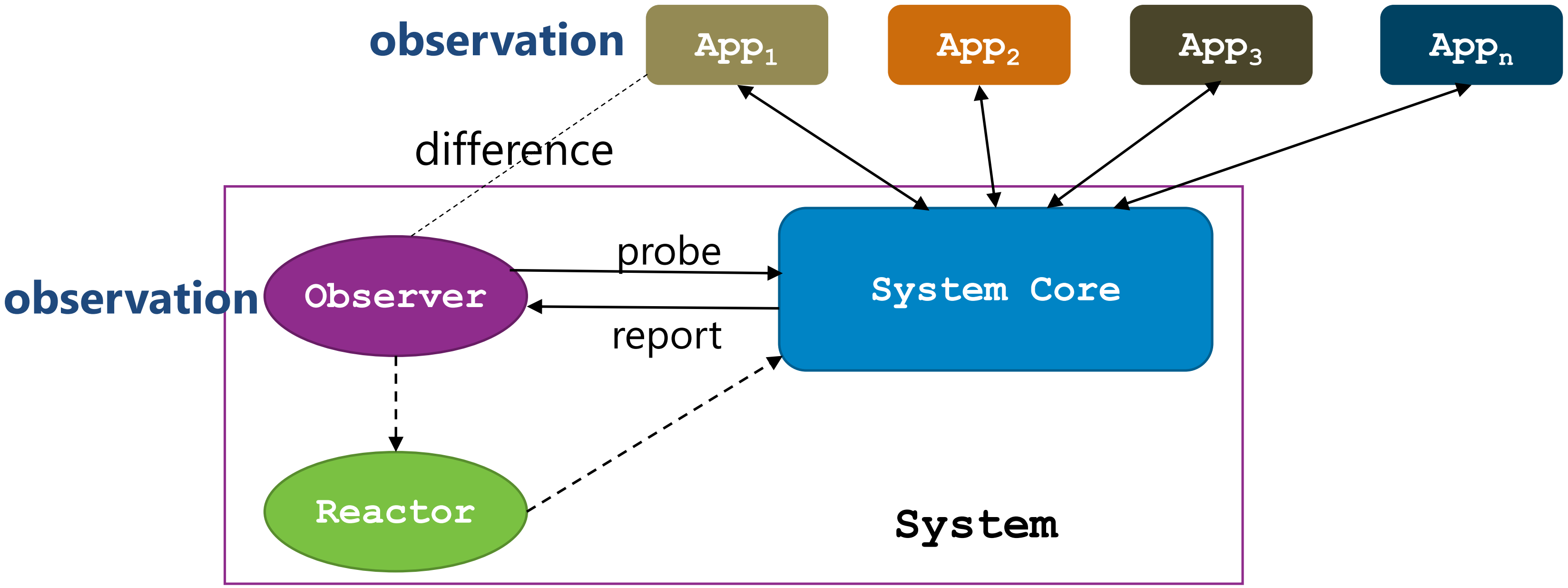


Gray Failure Trait: *Differential Observability*



Gray Failure Trait: *Differential Observability*

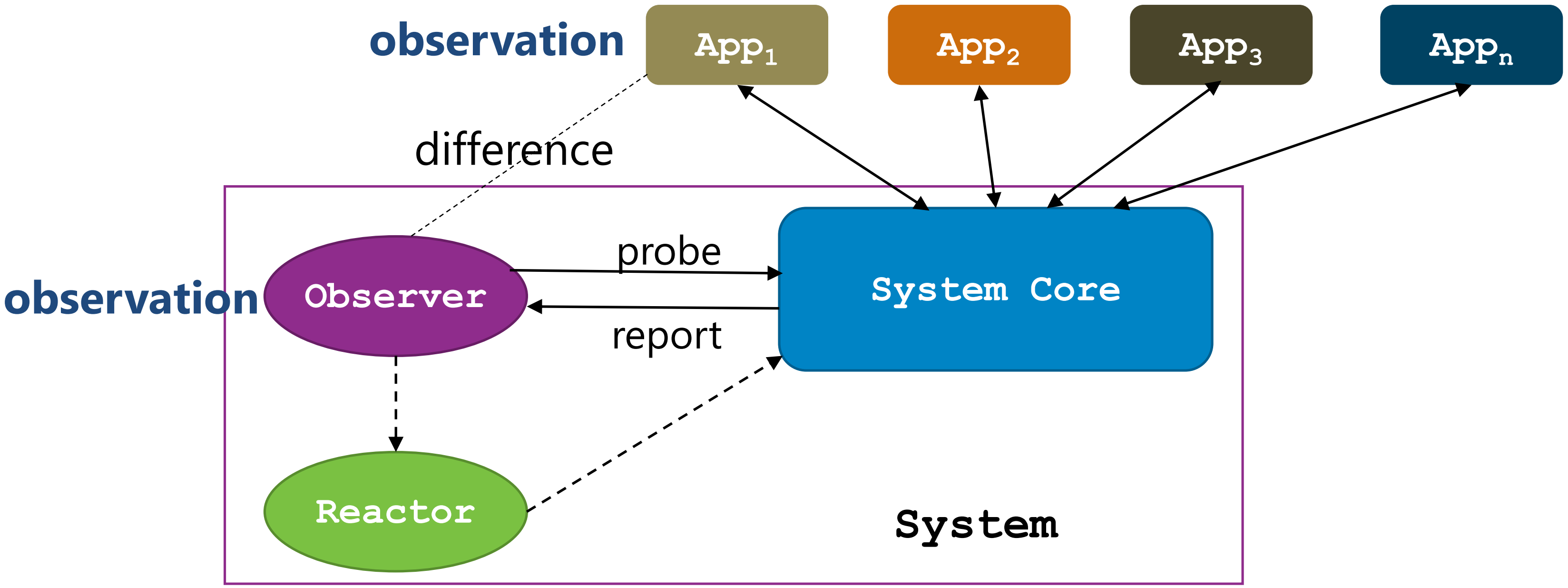
different entities come into different conclusions about whether a system is working or not



Gray Failure Trait: *Differential Observability*

	All apps deem system good	App$_i$ deems system bad
observer deems system good		
observer deems system bad		

Gray Failure!

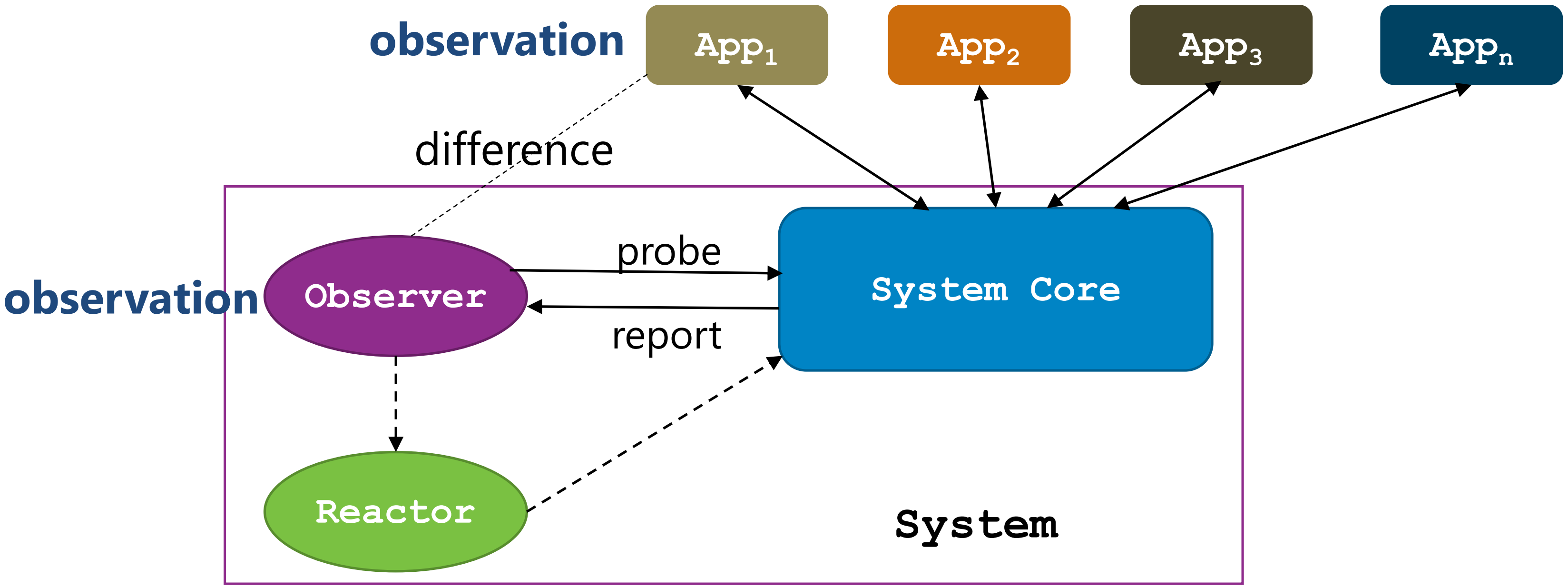


Gray Failure Trait: *Differential Observability*

Healthy or w/ minor latent fault

Gray Failure!

	All apps deem system good	App$_i$ deems system bad
observer deems system good	❶	❷
observer deems system bad		



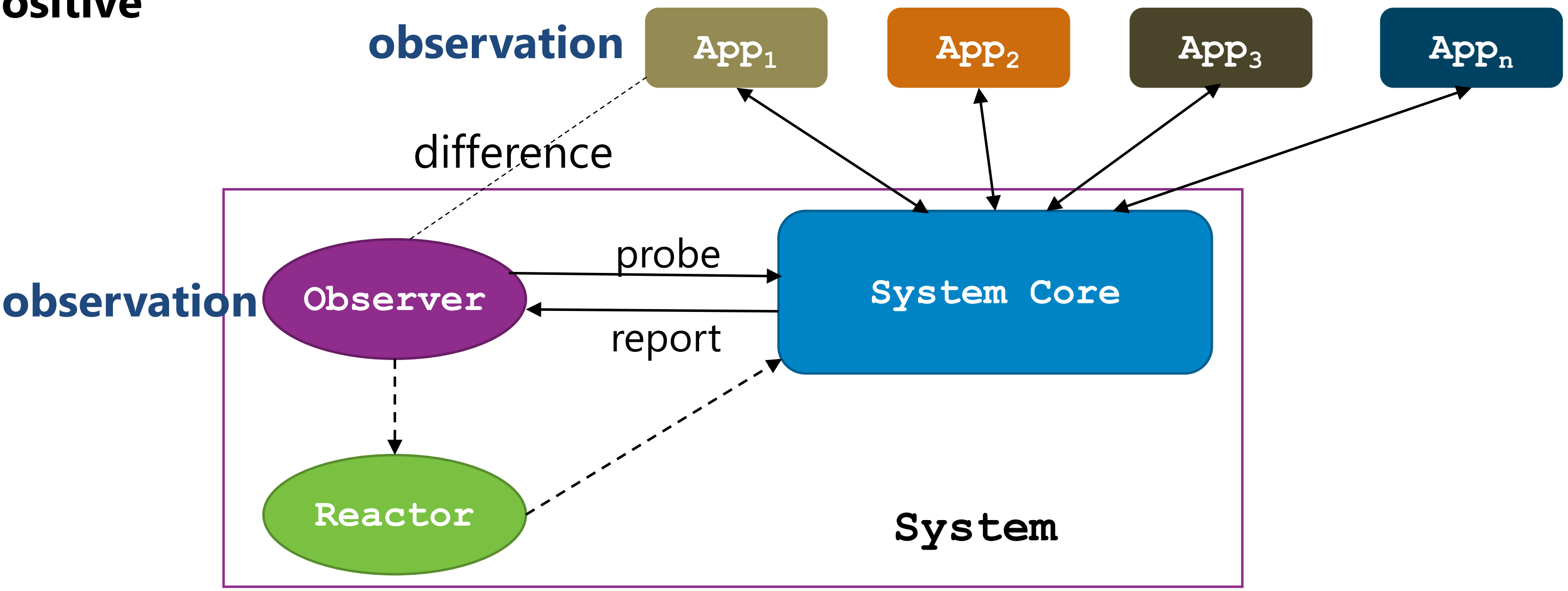
Gray Failure Trait: *Differential Observability*

Healthy or w/ minor latent fault

	All apps deem system good	App$_i$ deems system bad
observer deems system good	❶	❷
observer deems system bad	❸	

Gray Failure!

Fault tolerance at play, or a false positive



Gray Failure Trait: *Differential Observability*

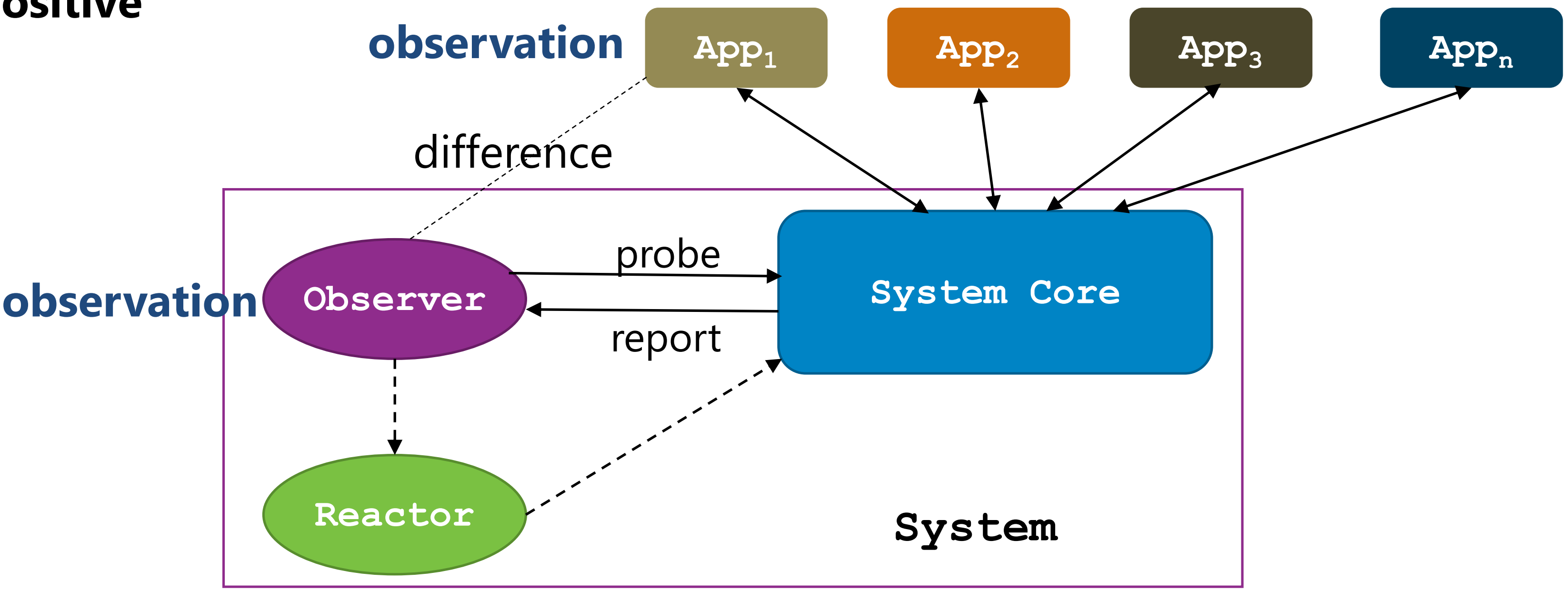
Healthy or w/ minor latent fault

	All apps deem system good	App$_i$ deems system bad
observer deems system good	❶	❷
observer deems system bad	❸	❹

Gray Failure!

Crash, fail-stop

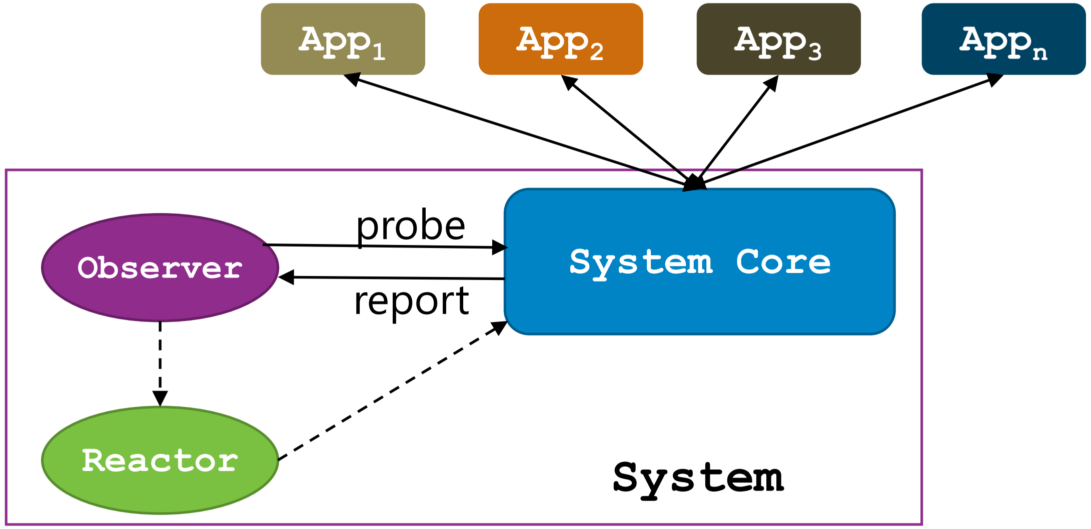
Fault tolerance at play, or a false positive



Applying the Model

Case I

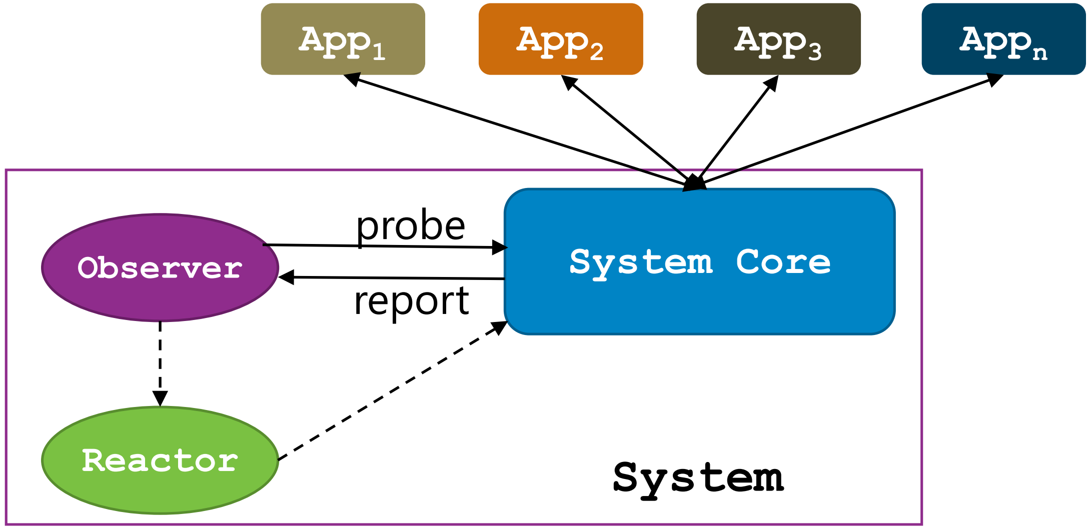
term	example
`system`	data center network
`observer`	switch peers
`reactor`	routing protocols
`app`$_1$	simple web server
`app`$_2$	search engine
gray failure	`app2` observed glitches but neighbors of the bad switch (and `app1`) didn't



Applying the Model

Case III

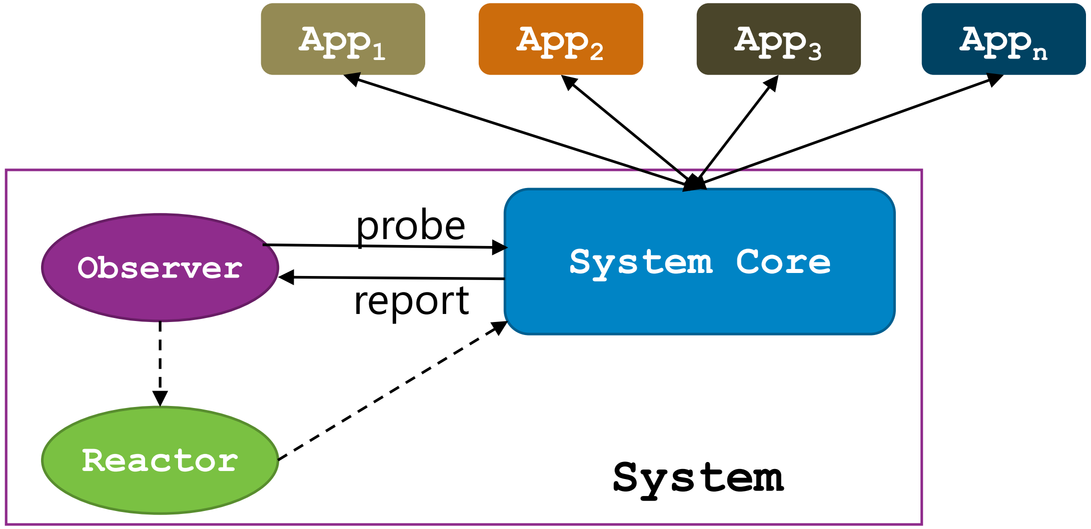
term	example
system	Azure storage service
observer	storage master
reactor	storage master
$app_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy



Applying the Model

Case III

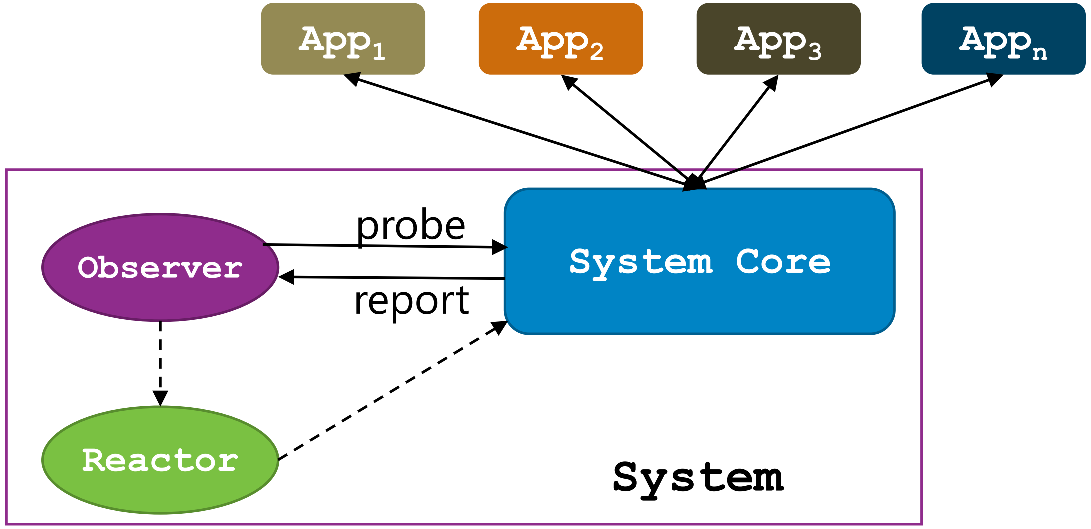
term	example
system	Azure storage service
observer	storage master
reactor	storage master
$app_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy



Applying the Model

Case III

term	example
system	Azure storage service
observer	storage master
reactor	storage master
$app_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy

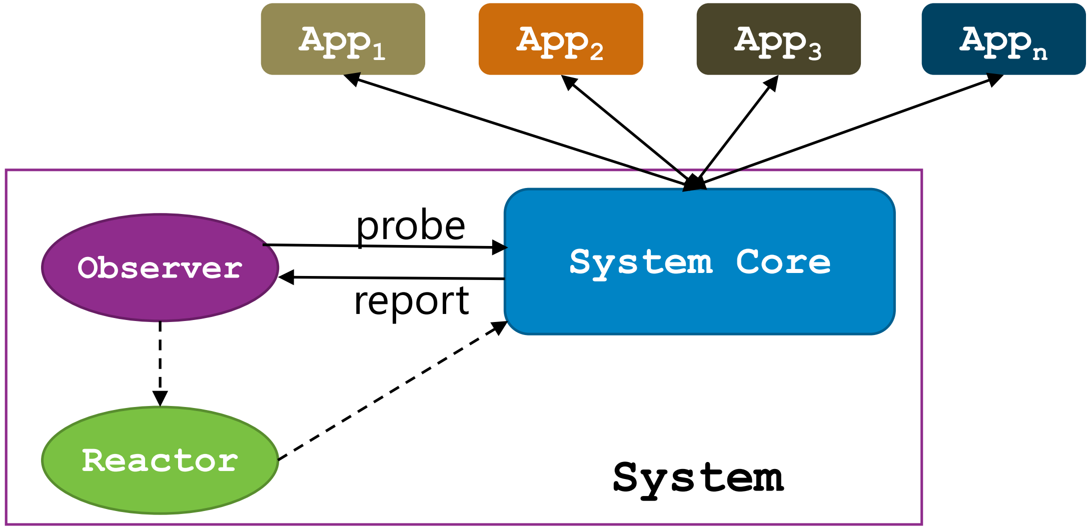


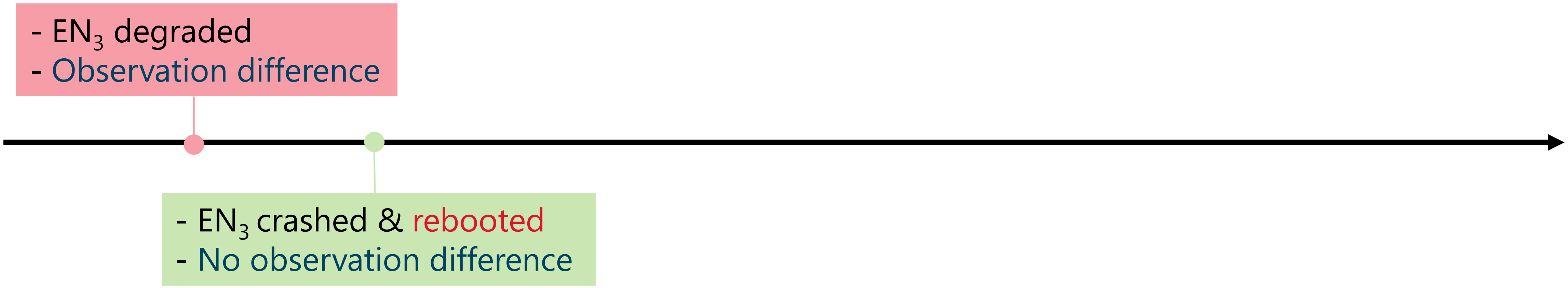
- EN_3 degraded
- Observation difference

Applying the Model

Case III

term	example
system	Azure storage service
observer	storage master
reactor	storage master
$\mathrm{app}_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy





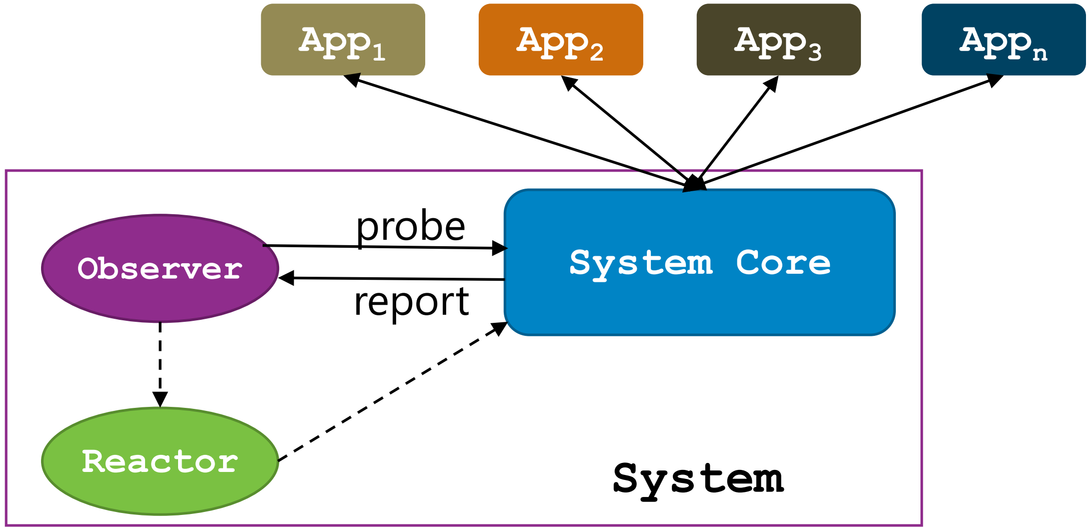
- EN_3 degraded
- Observation difference

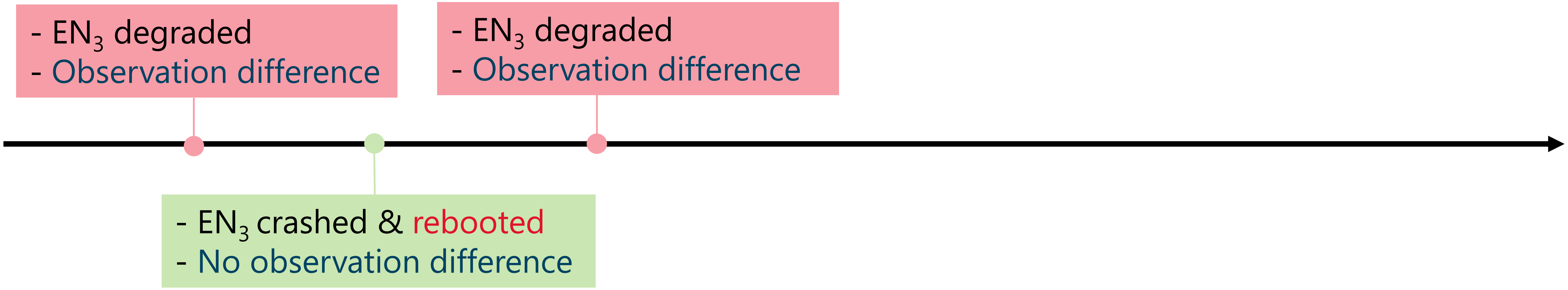
- EN_3 crashed & rebooted
- No observation difference

Applying the Model

Case III

term	example
system	Azure storage service
observer	storage master
reactor	storage master
$app_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy





- EN_3 degraded
- Observation difference

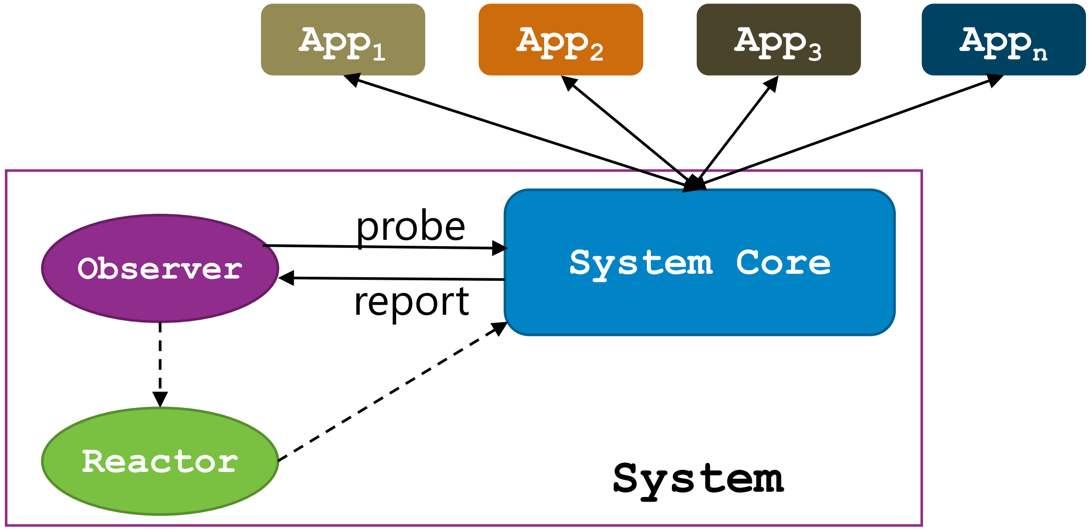
- EN_3 crashed & rebooted
- No observation difference

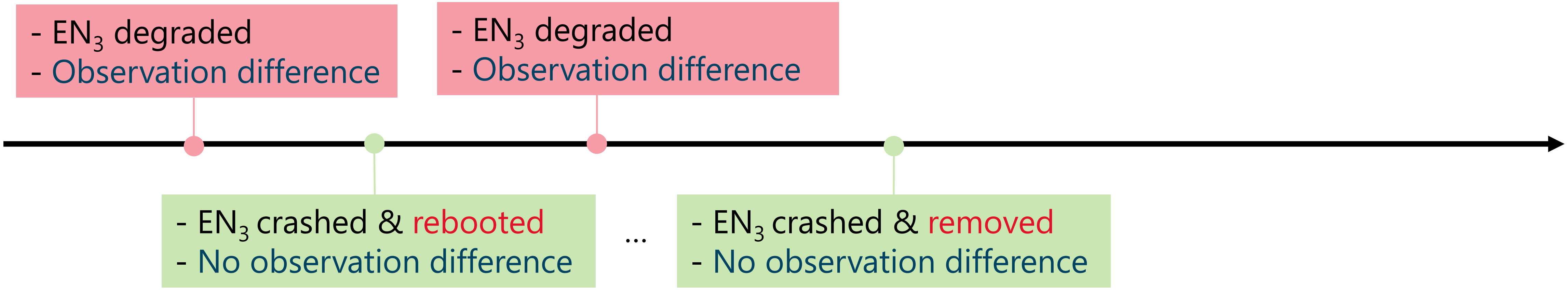
- EN_3 degraded
- Observation difference

Applying the Model

Case III

term	example
system	Azure storage service
observer	storage master
reactor	storage master
$app_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy





- EN_3 degraded
- Observation difference

- EN_3 crashed & rebooted
- No observation difference

- EN_3 degraded
- Observation difference

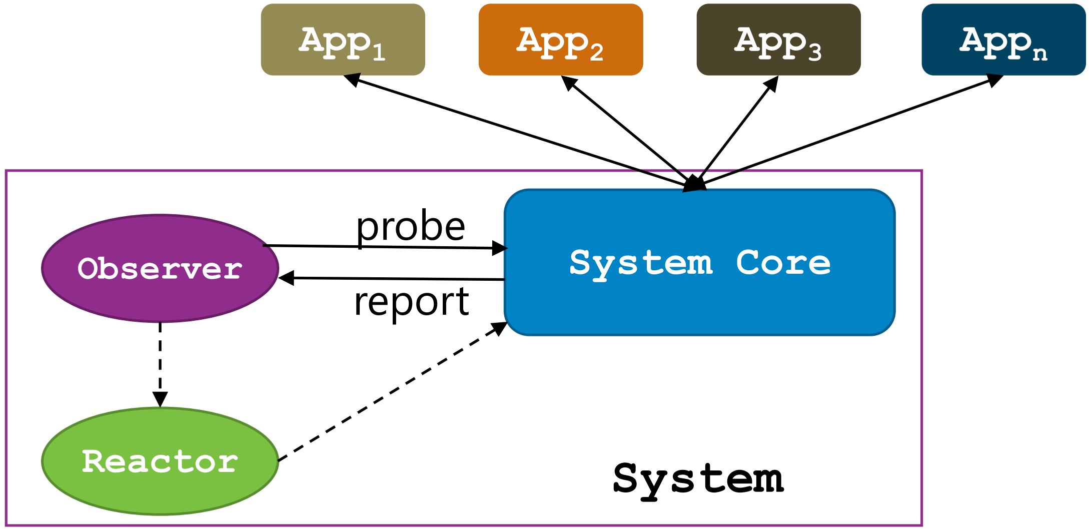
…

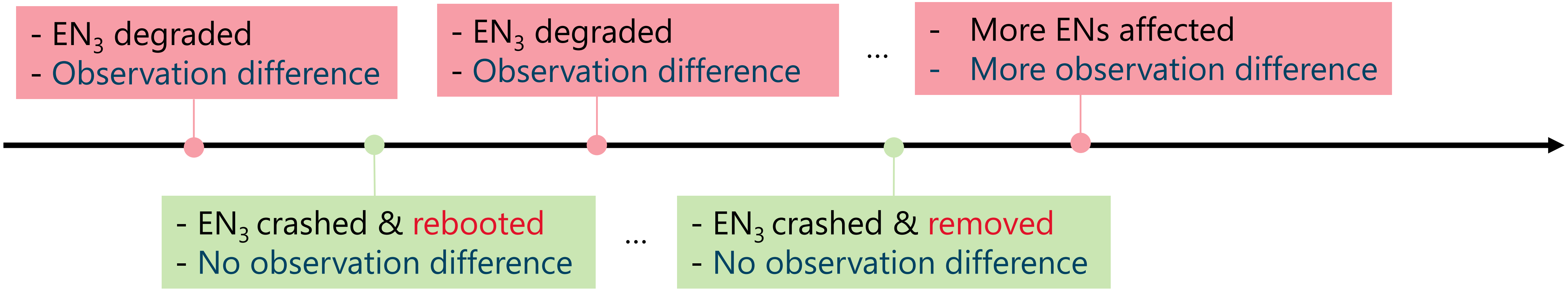
- EN_3 crashed & removed
- No observation difference

Applying the Model

Case III

term	example
system	Azure storage service
observer	storage master
reactor	storage master
$\text{app}_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy





- EN$_3$ degraded
- Observation difference

- EN$_3$ crashed & rebooted
- No observation difference

- EN$_3$ degraded
- Observation difference

…

…

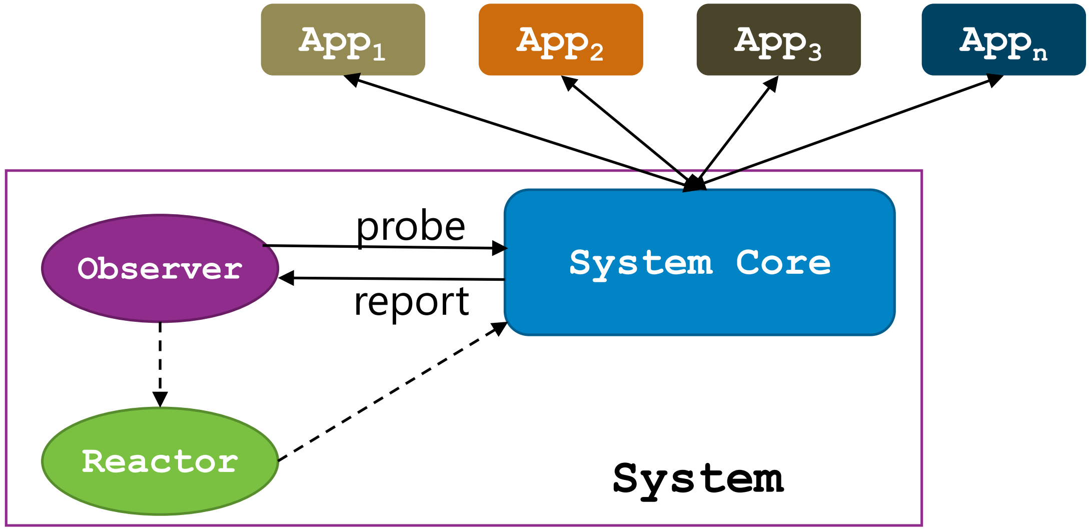
- EN$_3$ crashed & removed
- No observation difference

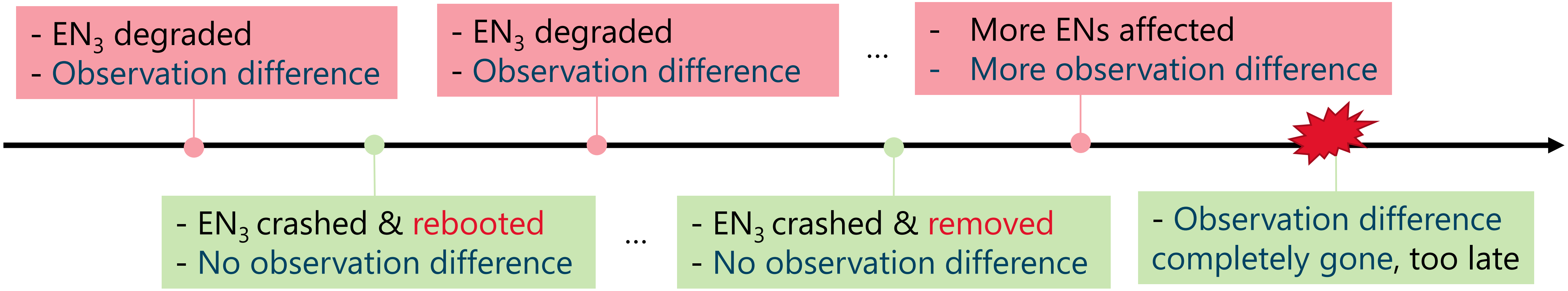
- More ENs affected
- More observation difference

Applying the Model

Case III

term	example
system	Azure storage service
observer	storage master
reactor	storage master
app$_{1..n}$	Azure VMs
gray failure	some VMs hit remote I/O exceptions while storage master deems ENs healthy




- EN$_3$ degraded
- Observation difference

- EN$_3$ crashed & rebooted
- No observation difference

- EN$_3$ degraded
- Observation difference

…

- EN$_3$ crashed & removed
- No observation difference

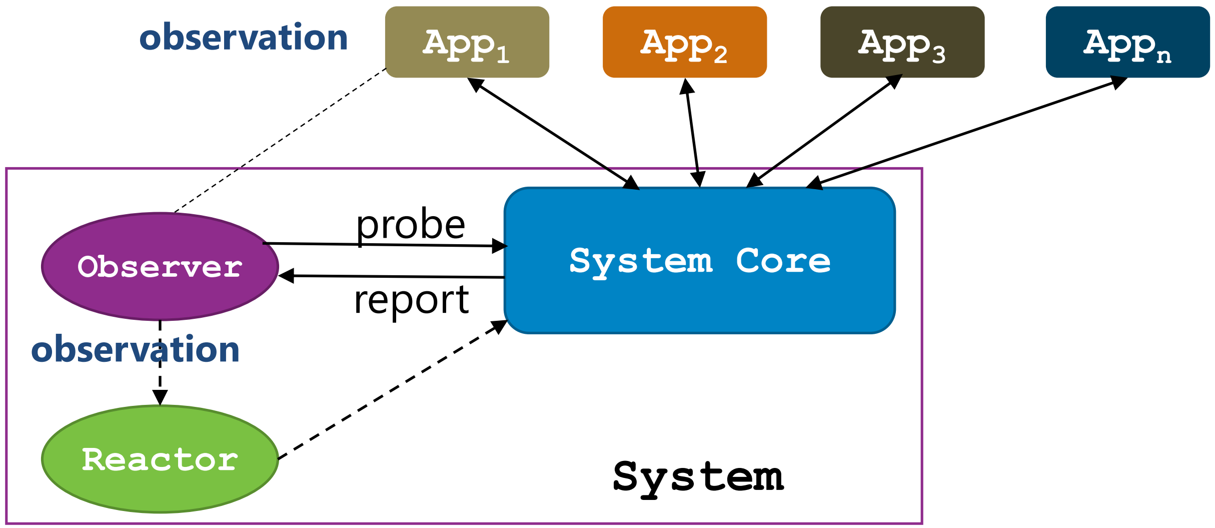
…

- More ENs affected
- More observation difference

- Observation difference completely gone, too late

Direction 1: Close Observation Gap

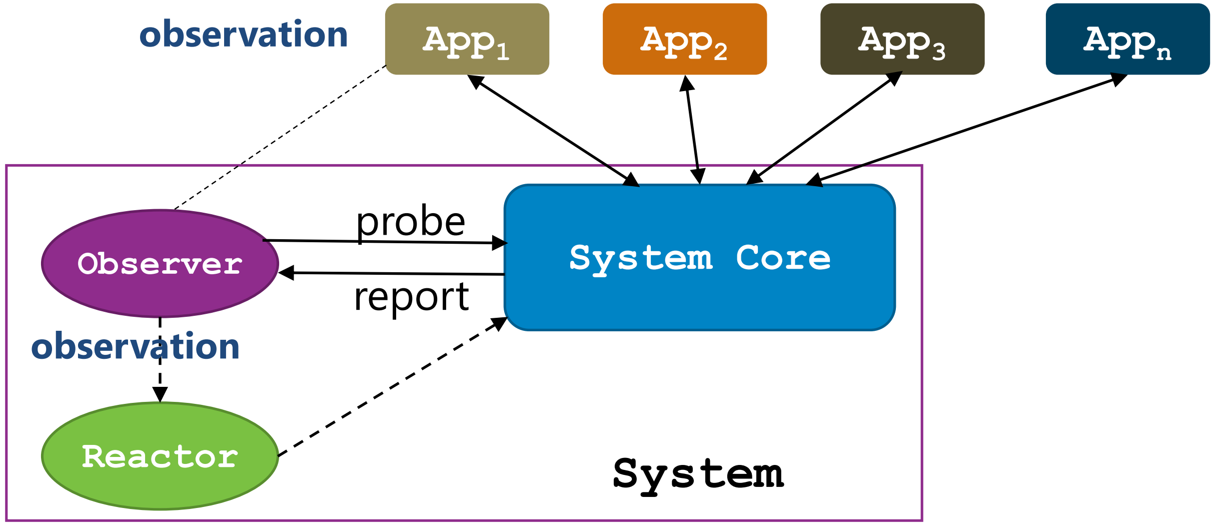
Traditional failure detector → multi-dimensional health monitor



Direction 1: Close Observation Gap

Traditional failure detector → multi-dimensional health monitor

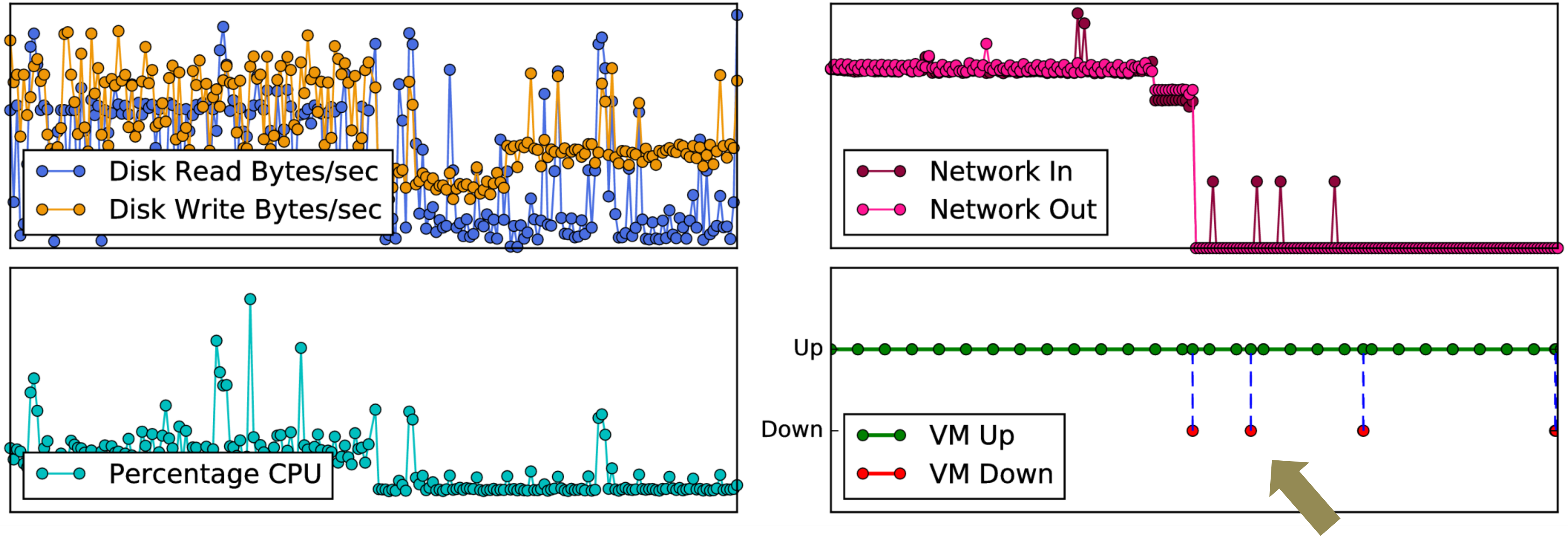
Doctors can’t use heartbeat as the only signal of a patient’s health



Direction 1: Close Observation Gap

Traditional failure detector ➔ multi-dimensional health monitor

Doctors can’t use heartbeat as the only signal of a patient’s health

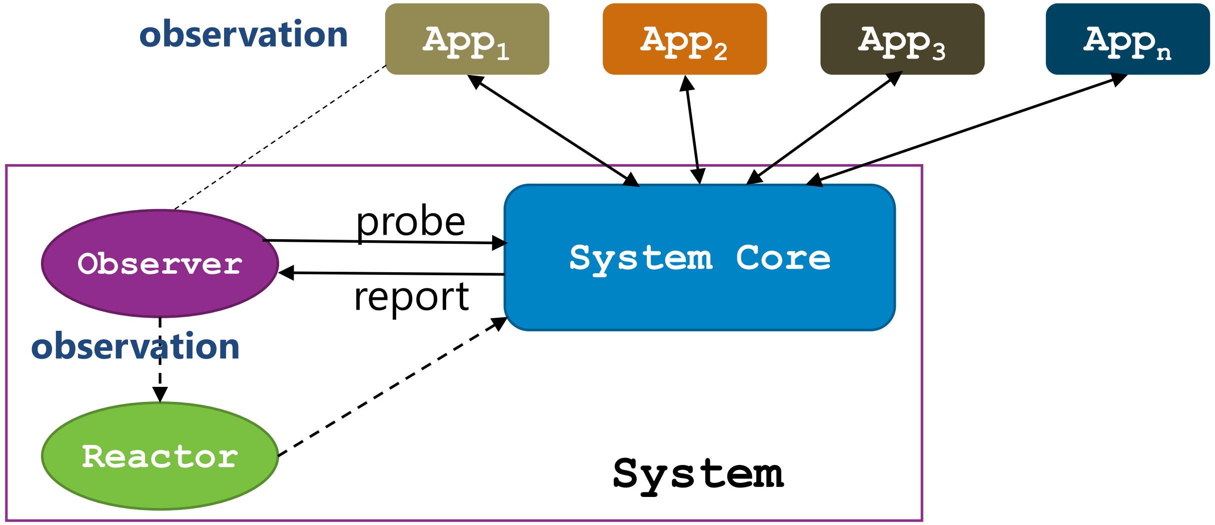


In-VM performance counters before and during the gray failure incident (case II)

Heartbeat-based hierarchical failure detector

Direction 2: Approximate Application View

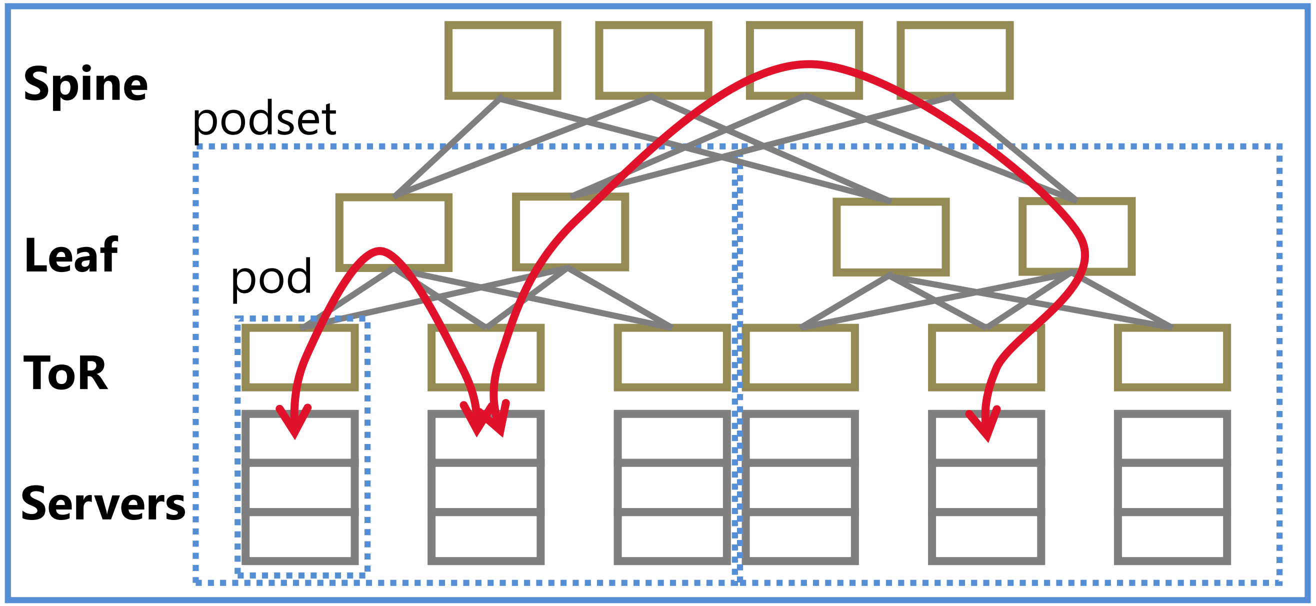
- Infeasible to completely eliminate differential observability due to multi-tenancy and modularity constraints
- System sends probes to emulate common application usage



Direction 2: Approximate Application View

- Infeasible to completely eliminate differential observability due to multi-tenancy and modularity constraints
- System sends probes to emulate common application usage

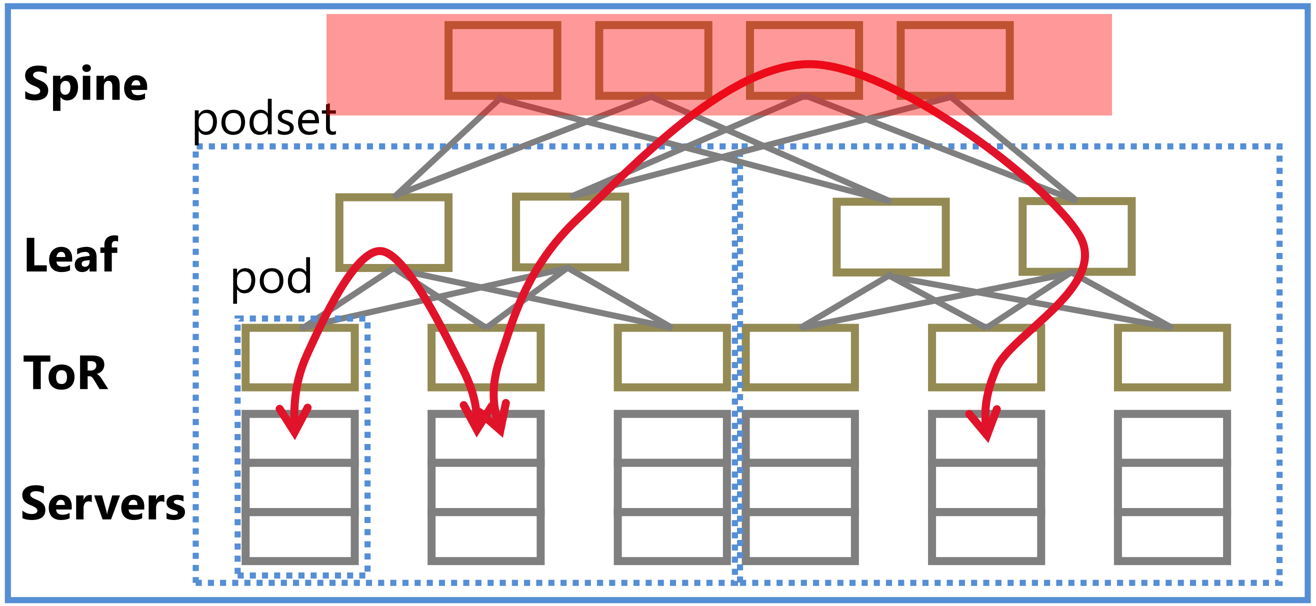
PingMesh [SIGCOMM '15]

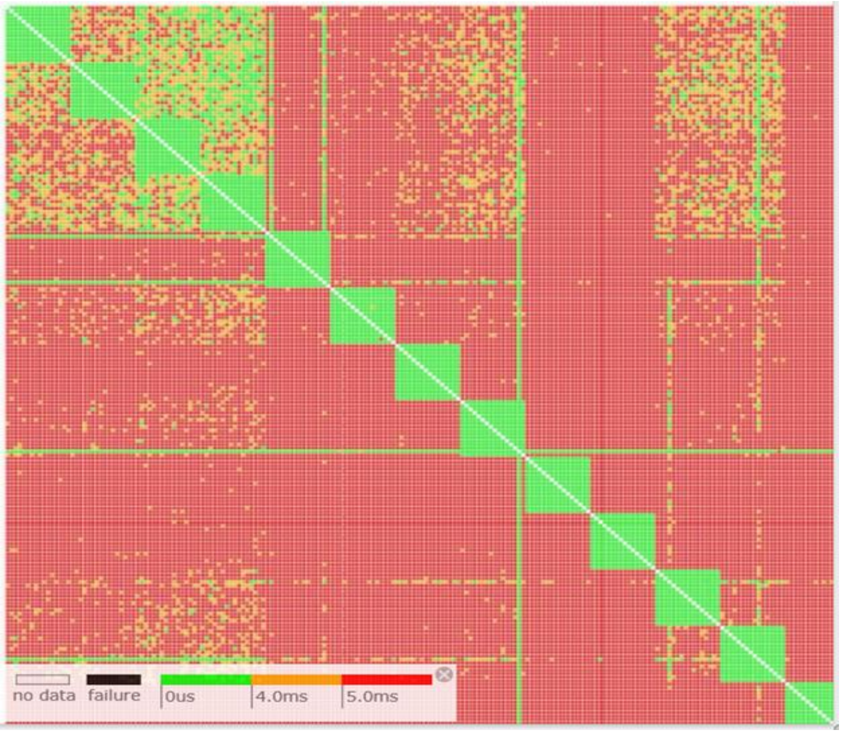


Direction 2: Approximate Application View

- Infeasible to completely eliminate differential observability due to multi-tenancy and modularity constraints
- System sends probes to emulate common application usage

PingMesh [SIGCOMM '15]

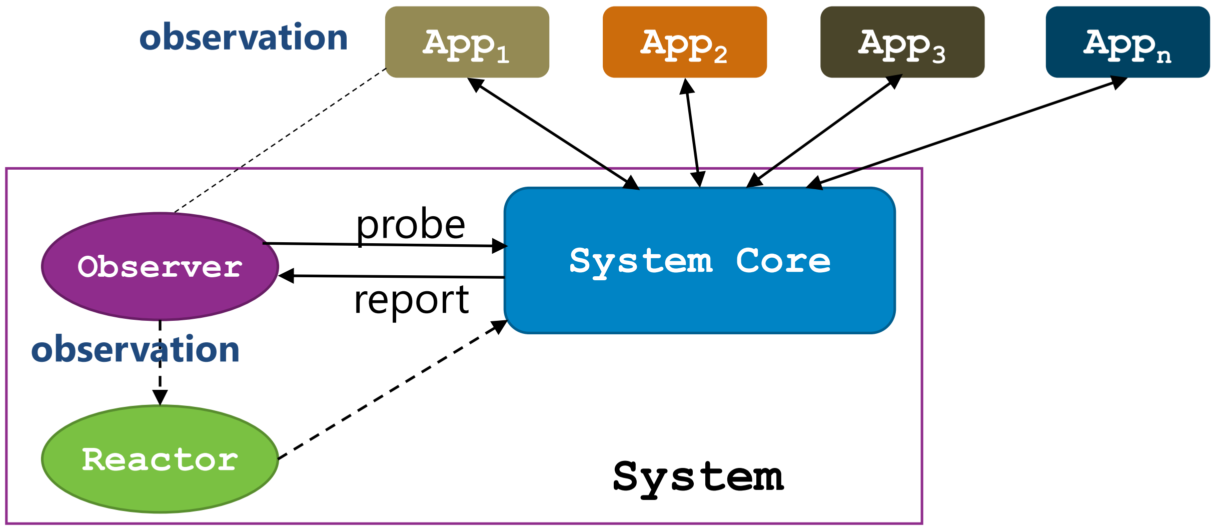




Failure in Spine

Direction 3: Leverage Power of Scale

Break the observation silos to complement each other



Direction 3: Leverage Power of Scale

Break the observation silos to complement each other

Observable vs. Detectable

- although end-to-end probe may expose differential observability, it may not detect *who* is responsible for the difference
- example solution: infer signals from many network devices to localize fault

Blame game

- *A* thinks *B* is problematic, *B* thinks *A* is problematic
- example solution: correlate VM failure events with topology info to judge

Direction 3: Leverage Power of Scale

Break the observation silos to complement each other

Observable vs. Detectable

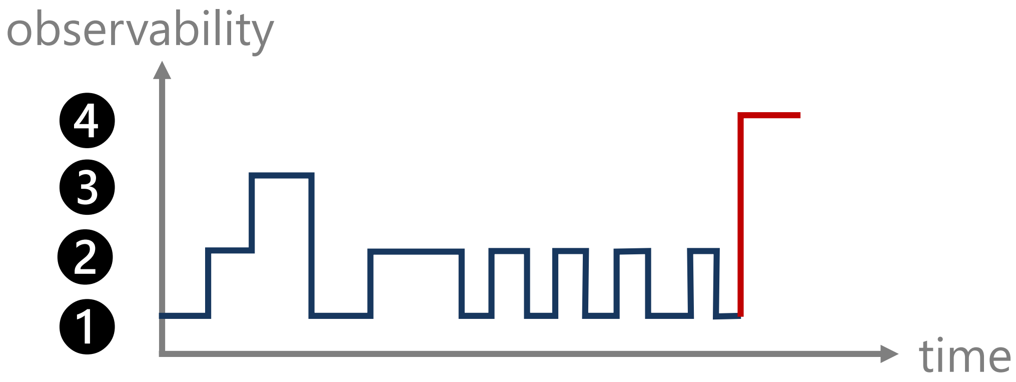
- although end-to-end probe may expose differential observability, it may not detect *who* is responsible for the difference
- example solution: infer signals from many network devices to localize fault

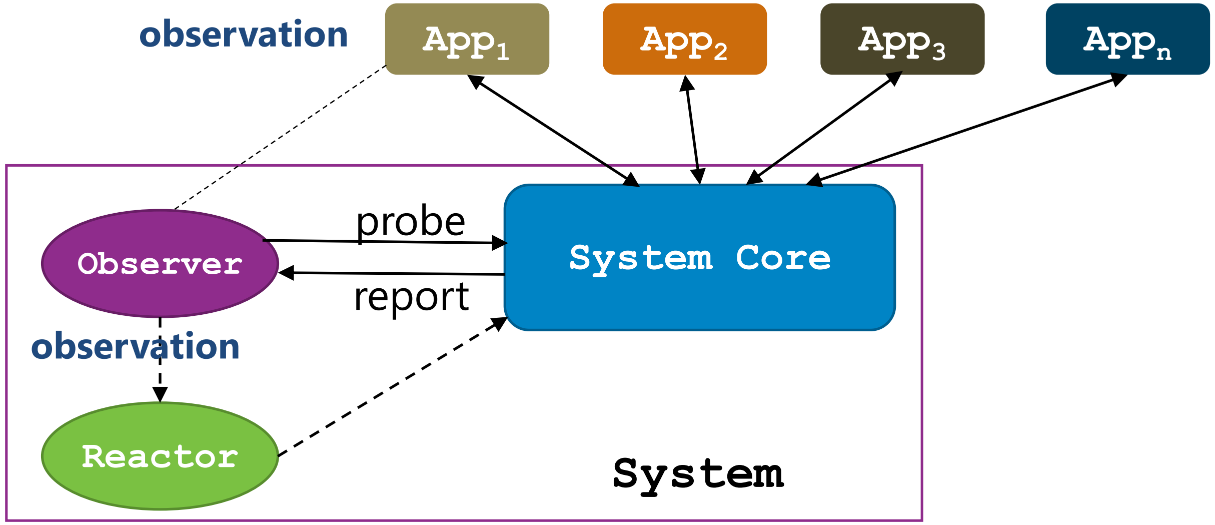
Blame game

- *A* thinks *B* is problematic, *B* thinks *A* is problematic
- example solution: correlate VM failure events with topology info to judge

Direction 4: Harness Temporal Patterns

Time series and trend analysis of key health signals

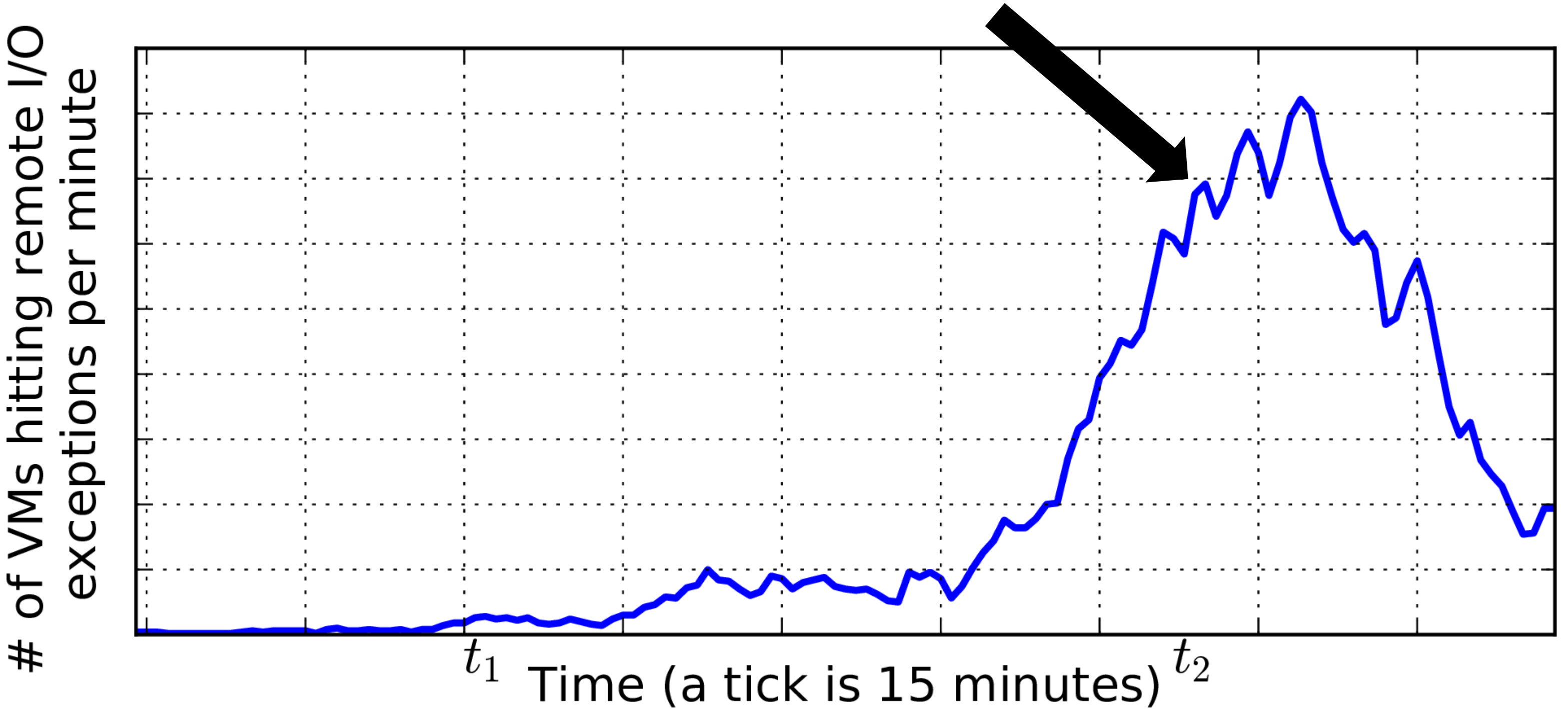




Direction 4: Harness Temporal Patterns

Time series and trend analysis of key health signals

Storage side observed availability issue



Conclusion

Cloud system are adept at handling crash and fail-stop failures

- » decades of efforts and research advances have paid off

Gray failure is a major challenge moving forward

- » behind many service incidents
- » an acute pain for system designers and engineers
- » fault tolerance 1.0 → 2.0

A first attempt to define and explore this problem domain

- » differential observability is a fundamental trait
- » addressing this trait is key to tackle gray failure
- » potential future directions with open challenges

Discussion

Why does differential observability occur?

- simplistic model and assumption about component behavior
- modularity principle, unexpected dependency, improper error handling
- focus on narrow point availability but dismissed *app* availability

Should (can) academia work on this problem?

- practitioners have been troubled by these issues for quite a while, relying on intuition, workaround, resources and process
- hungry for principled approach to understand and tackle the problem
- many issues exist in open-source distributed system stack as well