Heading Off Correlated Failures through Independence-as-a-Service

Ennan Zhai[1]

Ruichuan Chen[2], David Isaac Wolinsky[1], Bryan Ford[1]

[1] Yale University [2] Bell Labs/Alcatel-Lucent

Background

- Cloud services ensure reliability by redundancy:

Background

- Cloud services ensure reliability by redundancy:
  - Amazon S3 replicates data on multiple racks

Background

- Cloud services ensure reliability by redundancy:
  - Amazon S3 replicates data on multiple racks
  - iCloud rents EC2 and Azure redundantly

Background

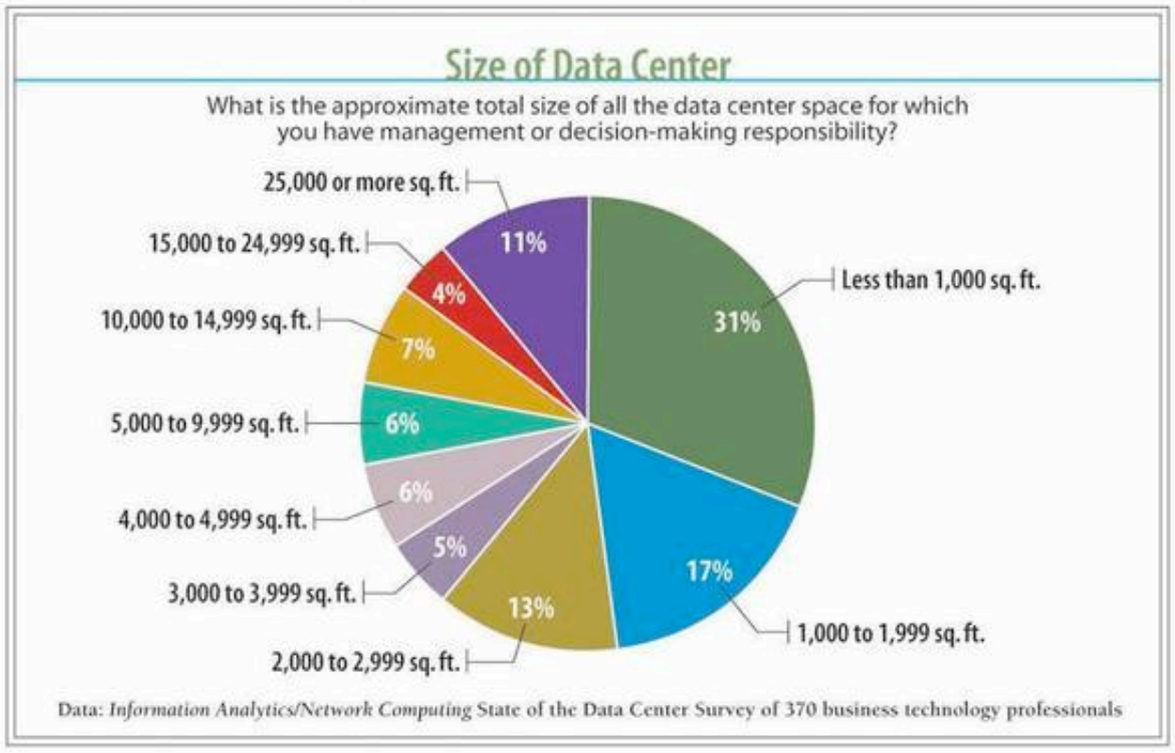
- Cloud services ensure reliability by redundancy:
  - Amazon S3 replicates data on multiple racks
  - iCloud rents EC2 and Azure redundantly

Unexpected common dependencies!

Service Outage Losses

Data Center Outages Generate Big Losses

Downtime in a data center can cost an average of $505,500 per incident, according to a Ponemon Institute study.

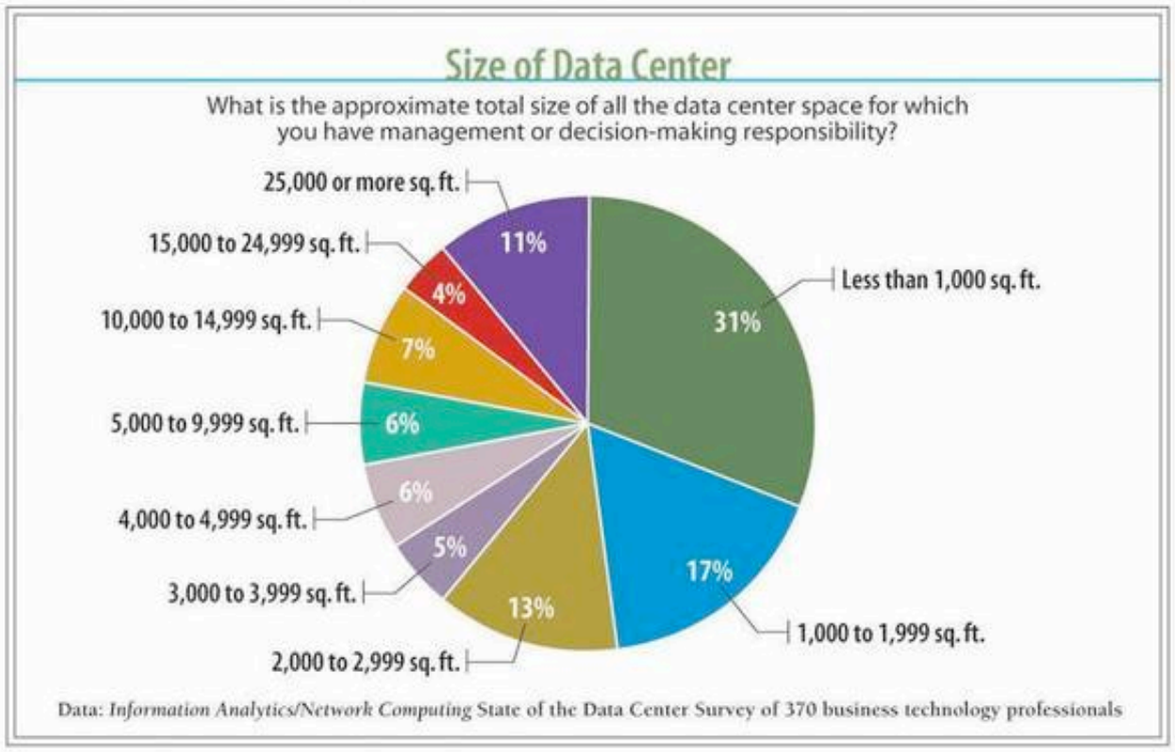


Analytics Slideshow: 2010 Data Center Operational Trends Report

Service Outage Losses

Data Center Outages Generate Big Losses

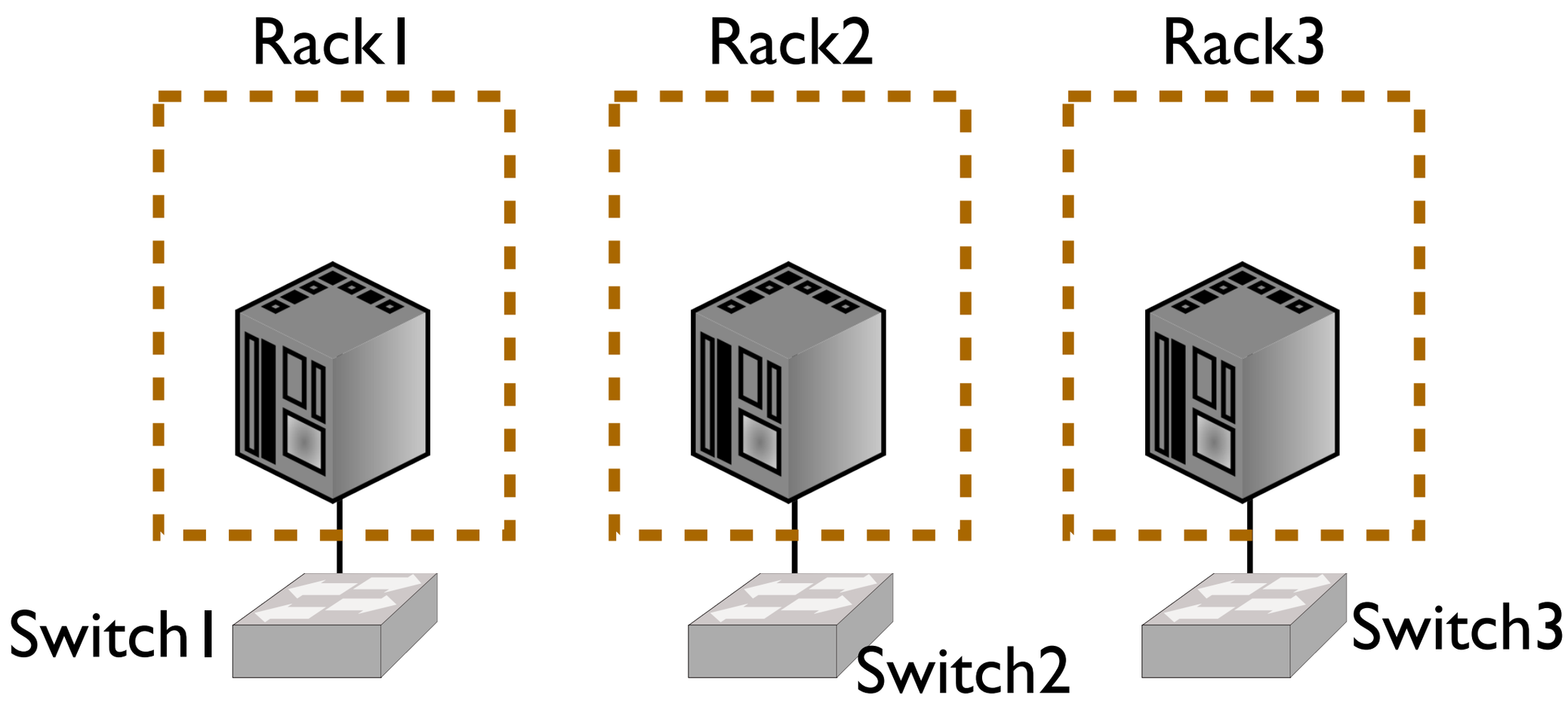
Downtime in a data center can cost an average of $505,500 per incident, according to a Ponemon Institute study.



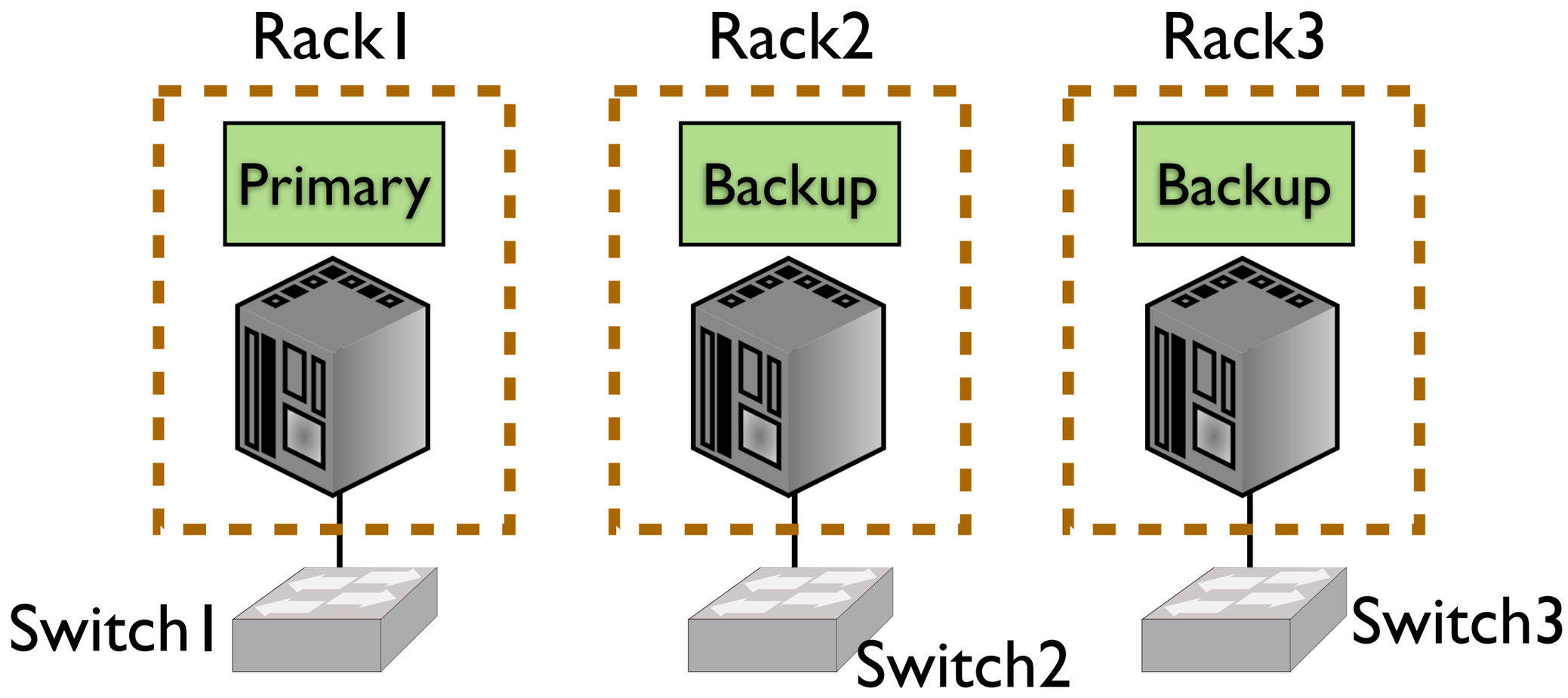
Analytics Slideshow: 2010 Data Center Operational Trends Report

What is correlated failure?

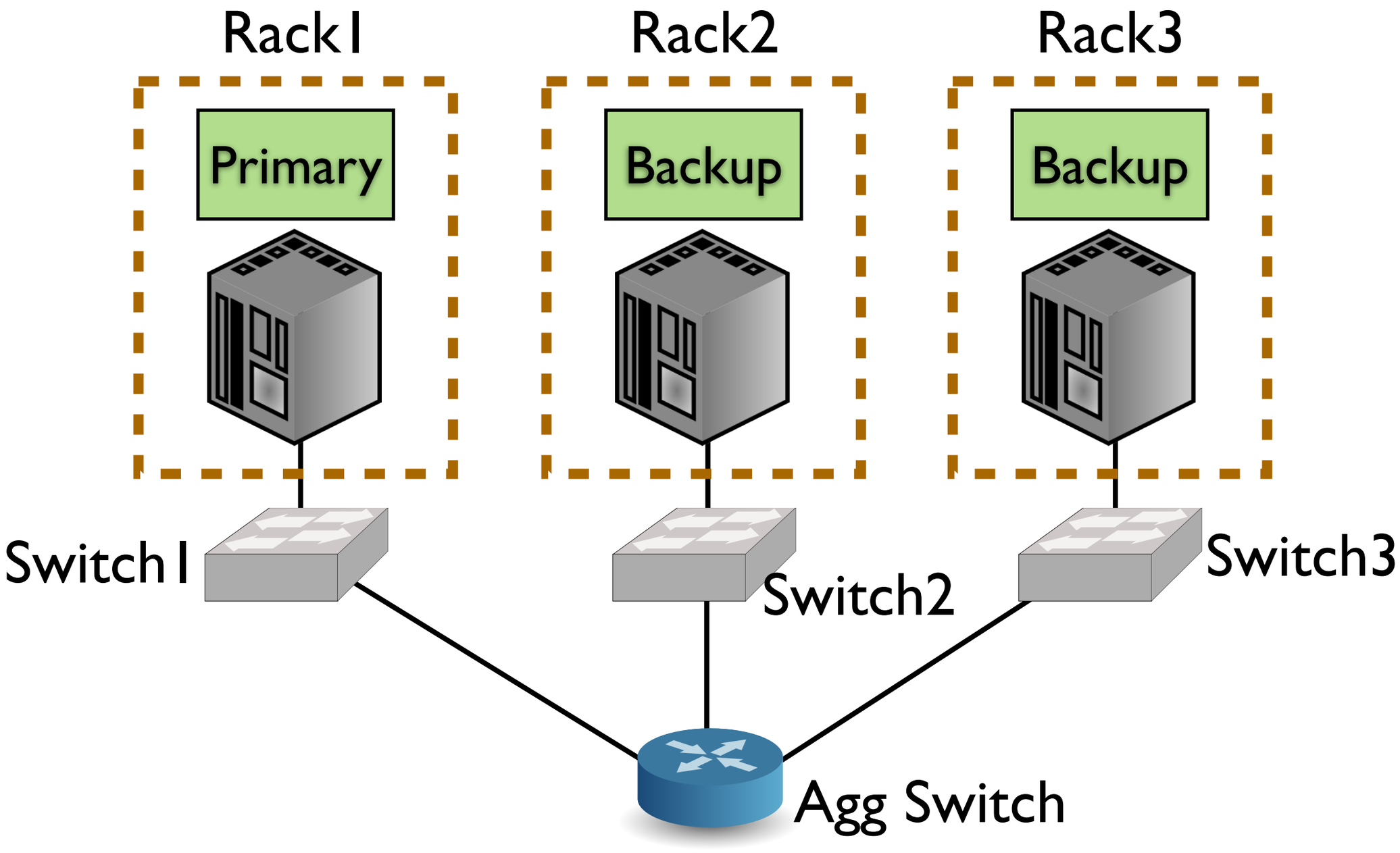
What is correlated failure?



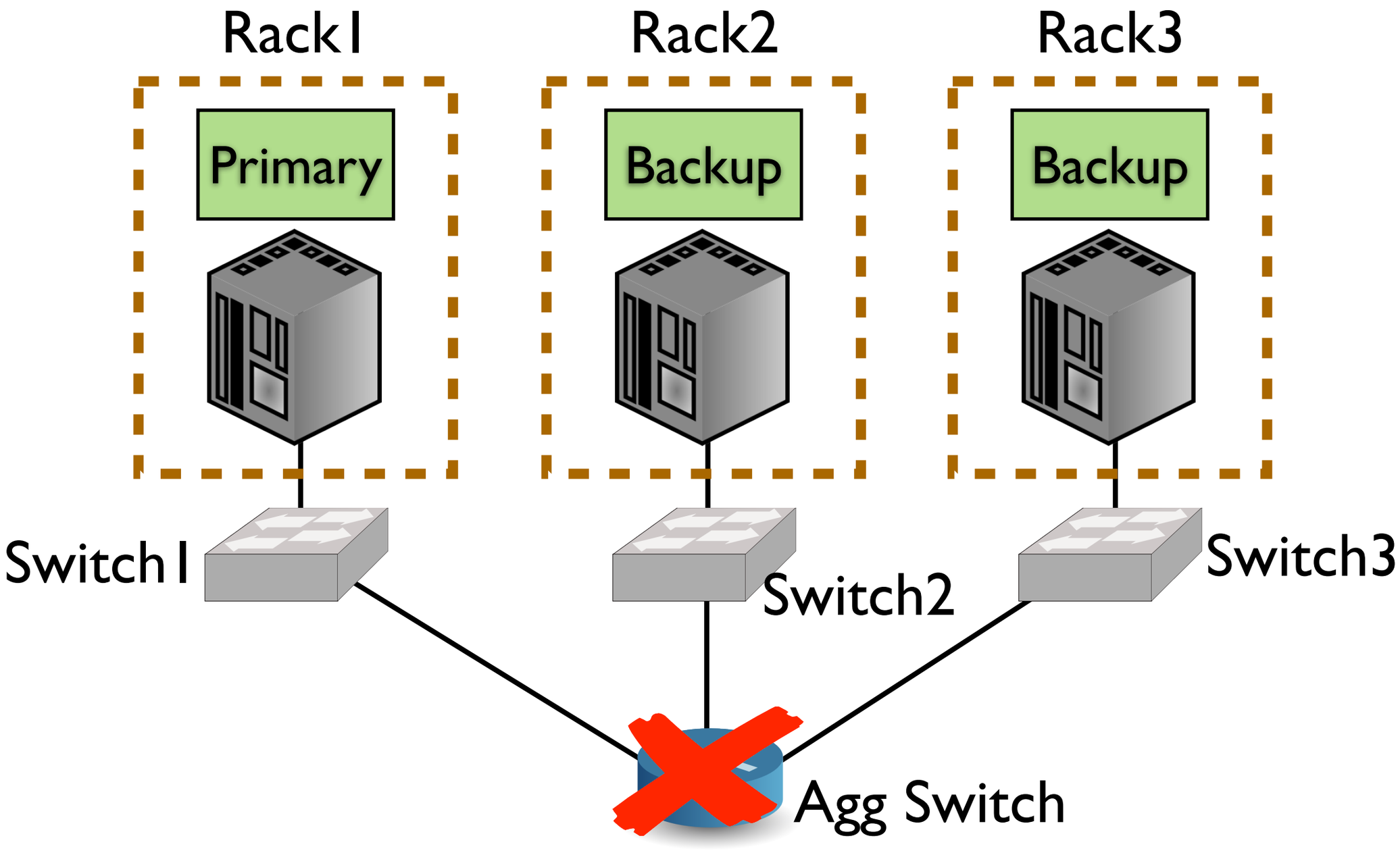
What is correlated failure?



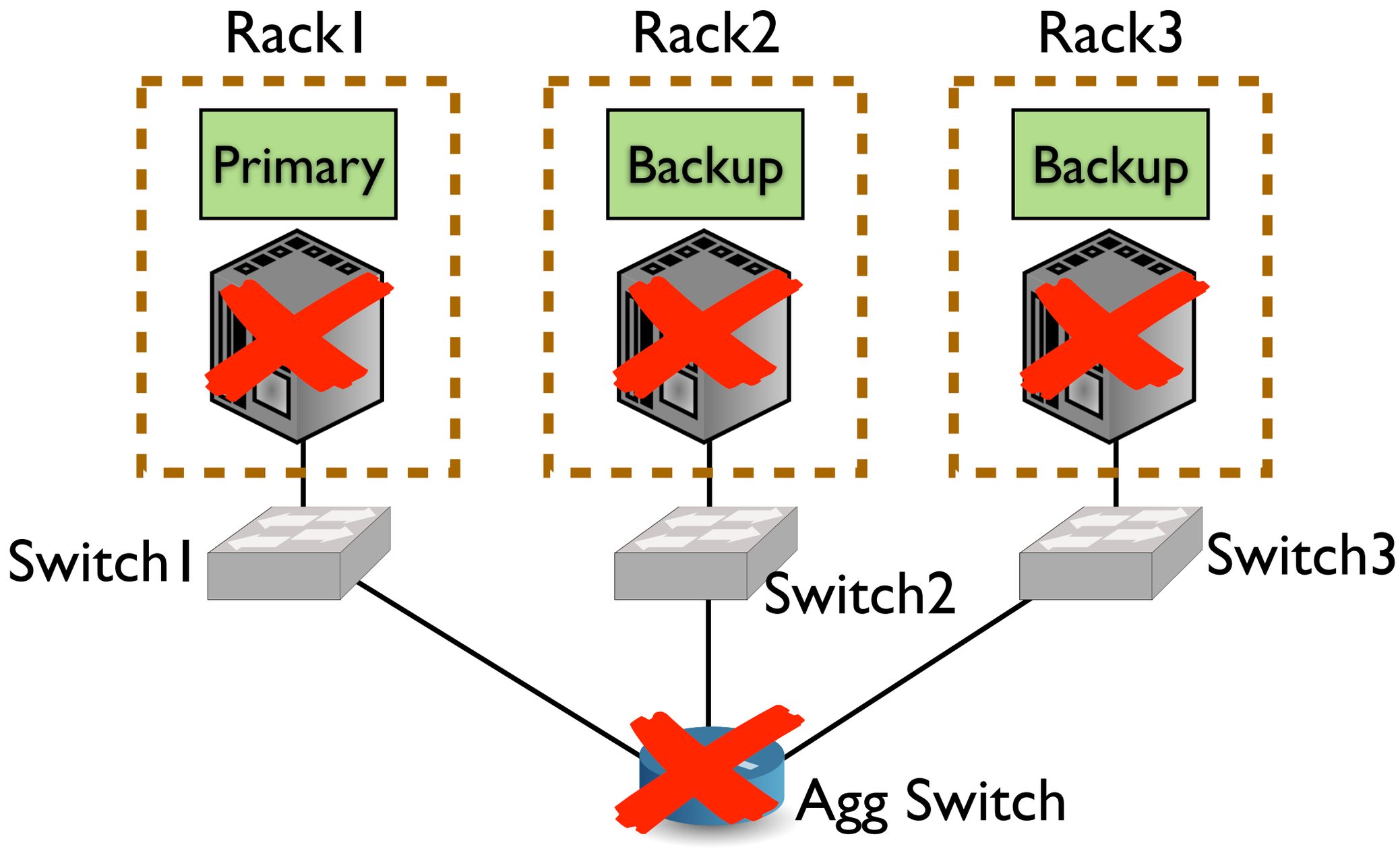
What is correlated failure?



What is correlated failure?



What is correlated failure?



Realistic Example

Realistic Example

Correlated failures resulting from EBS due to bugs in one EBS server

Summary of the October 22, 2012 AWS Service Event in the US-East Region

We'd like to share more about the service event that occurred on Monday, October 22nd in the US-East Region. We have now completed the analysis of the events that affected AWS customers, and we want to describe what happened, our understanding of how customers were affected, and what we are doing to prevent a similar issue from occurring in the future.

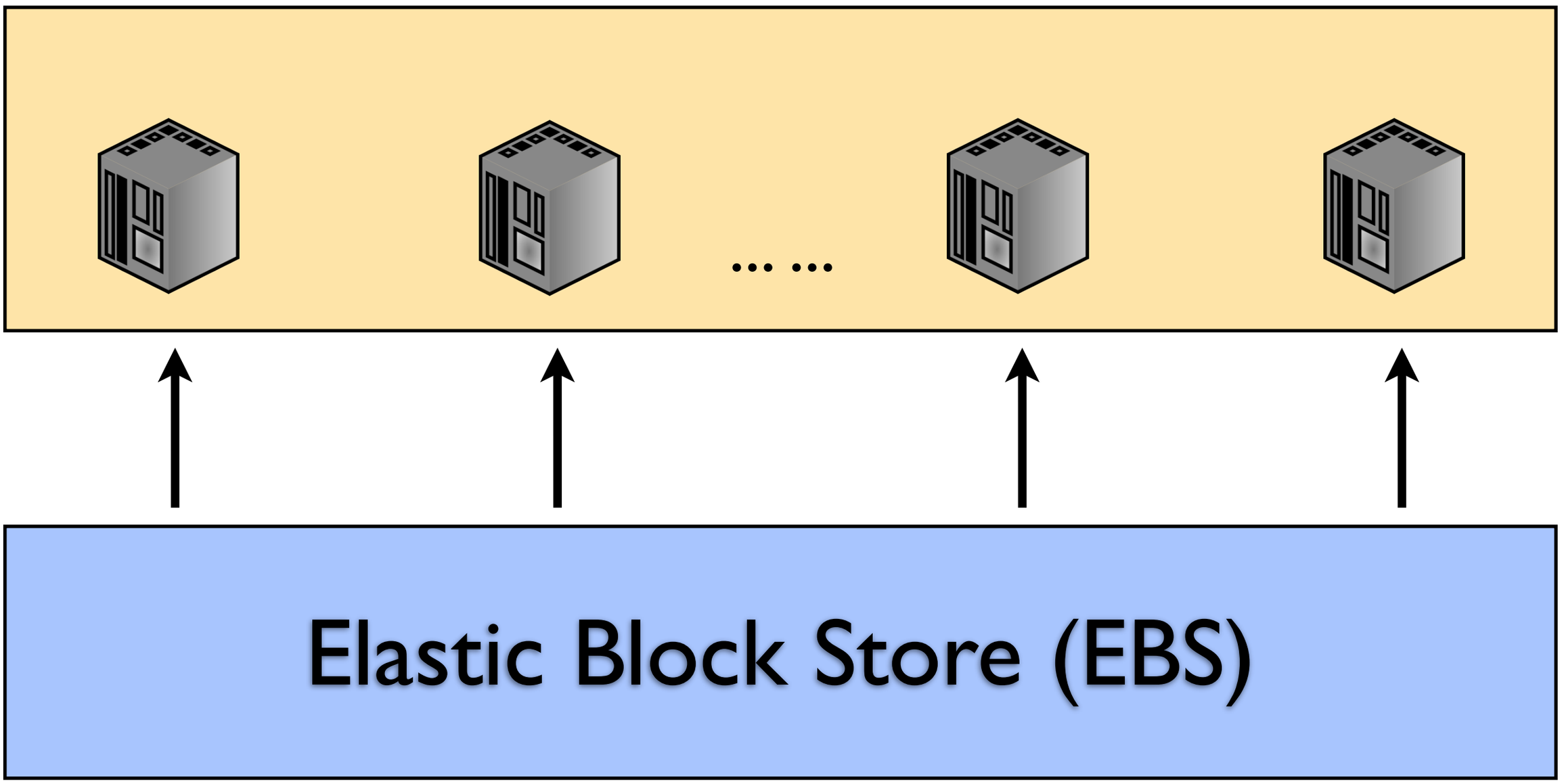
The Primary Event and the Impact to Amazon Elastic Block Store (EBS) and Amazon Elastic Compute Cloud (EC2)

Realistic Example

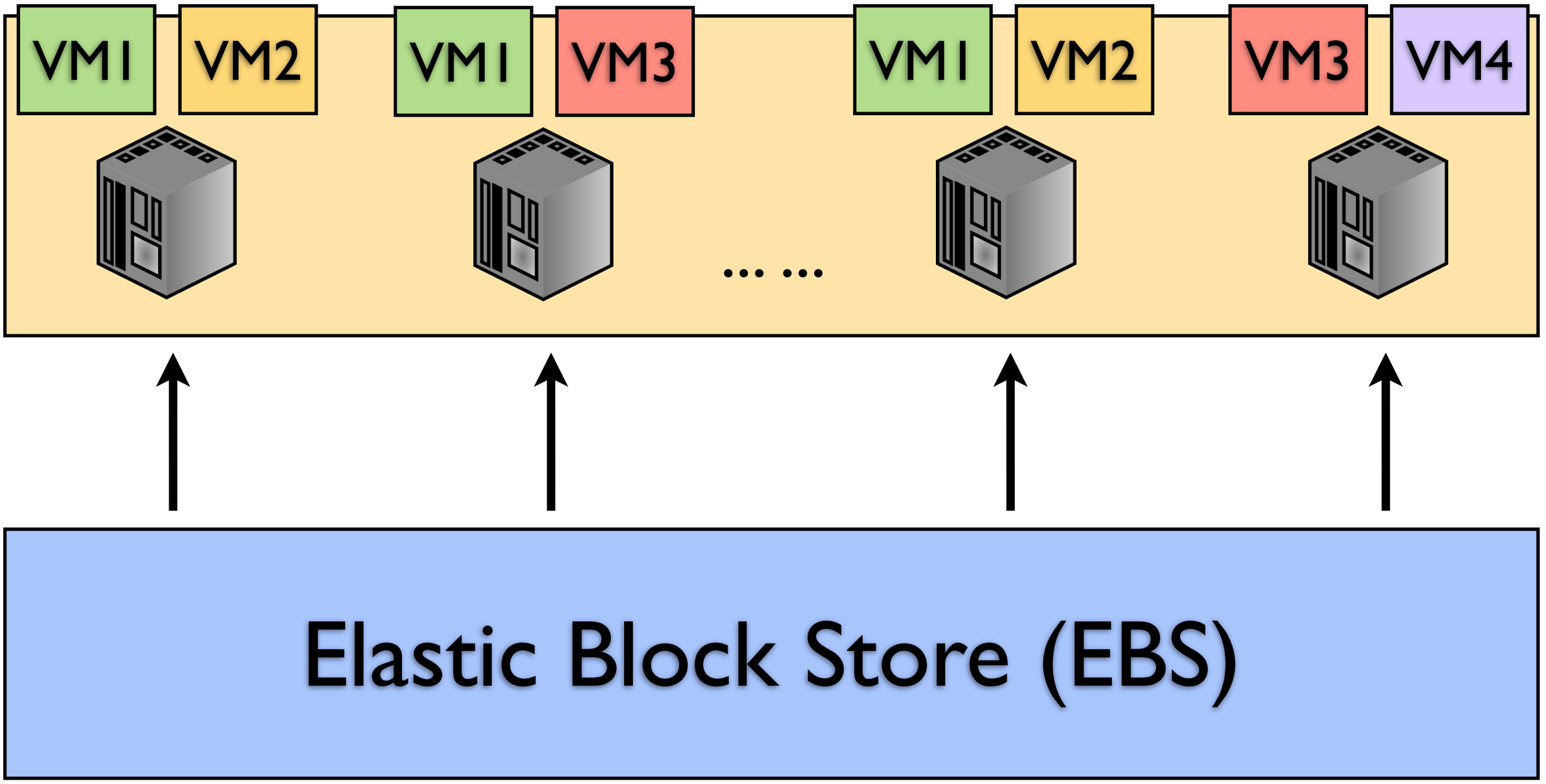
Elastic Compute Cloud (EC2)

Elastic Block Store (EBS)

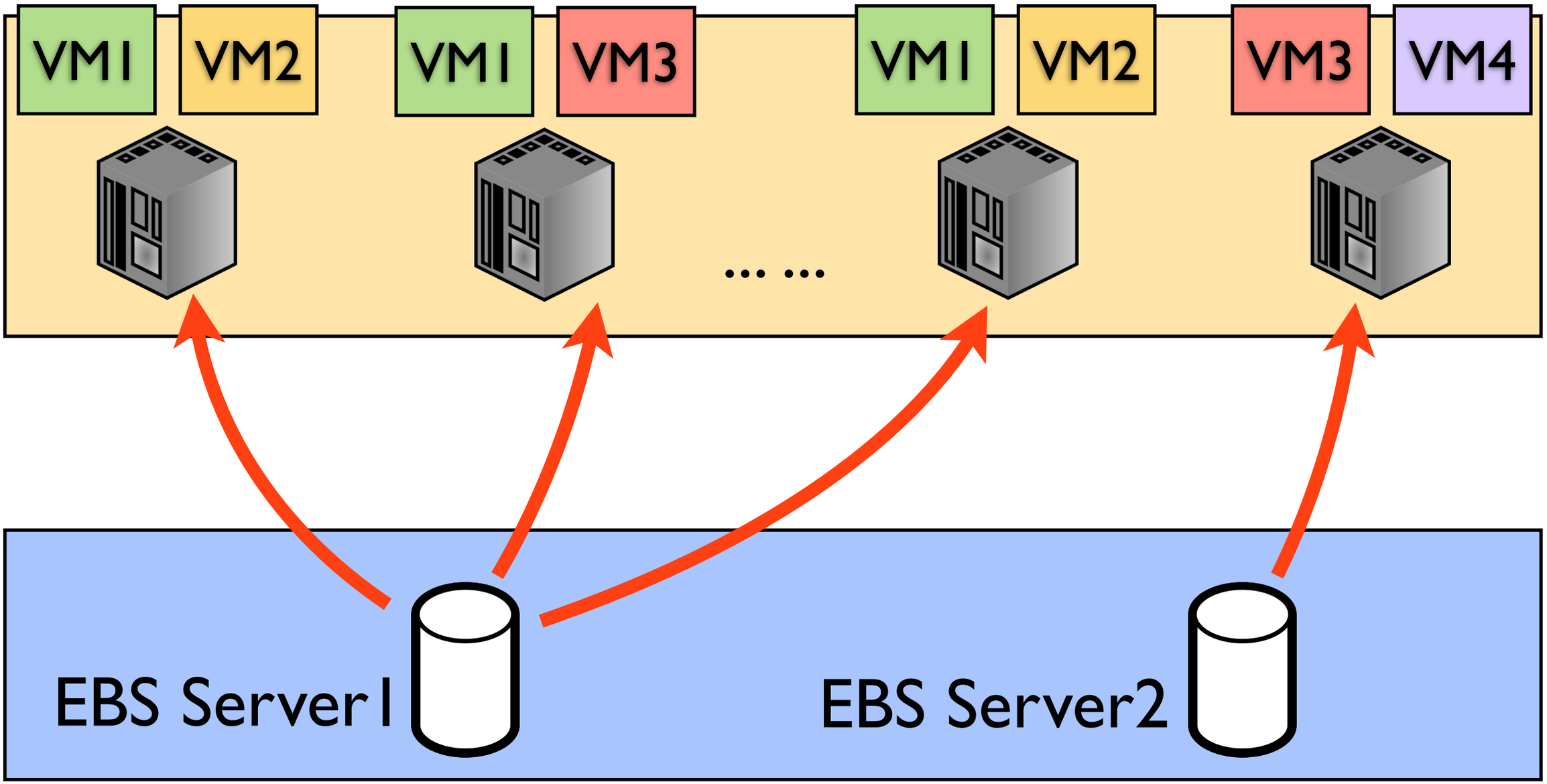
Realistic Example



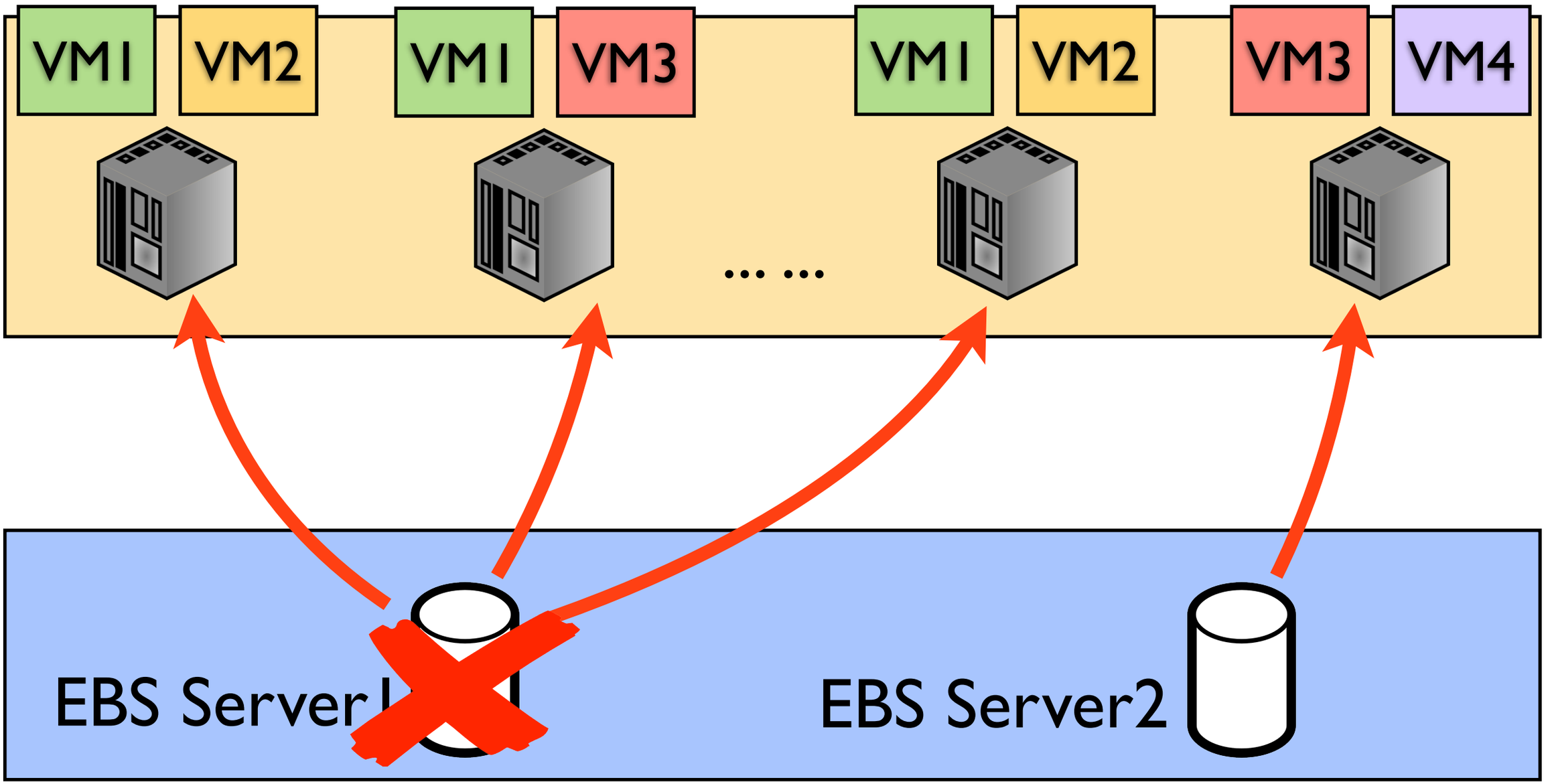
Realistic Example



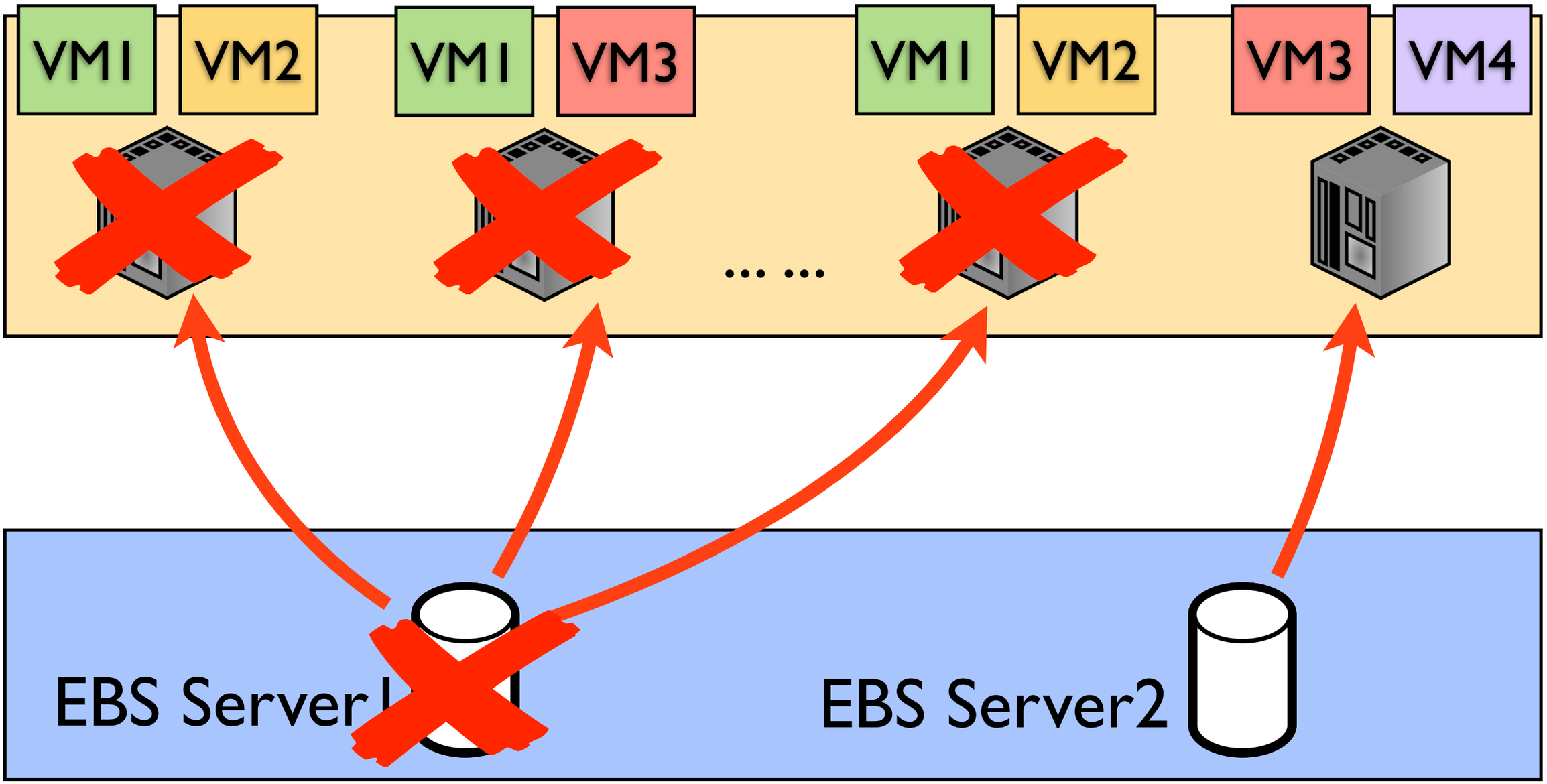
Realistic Example



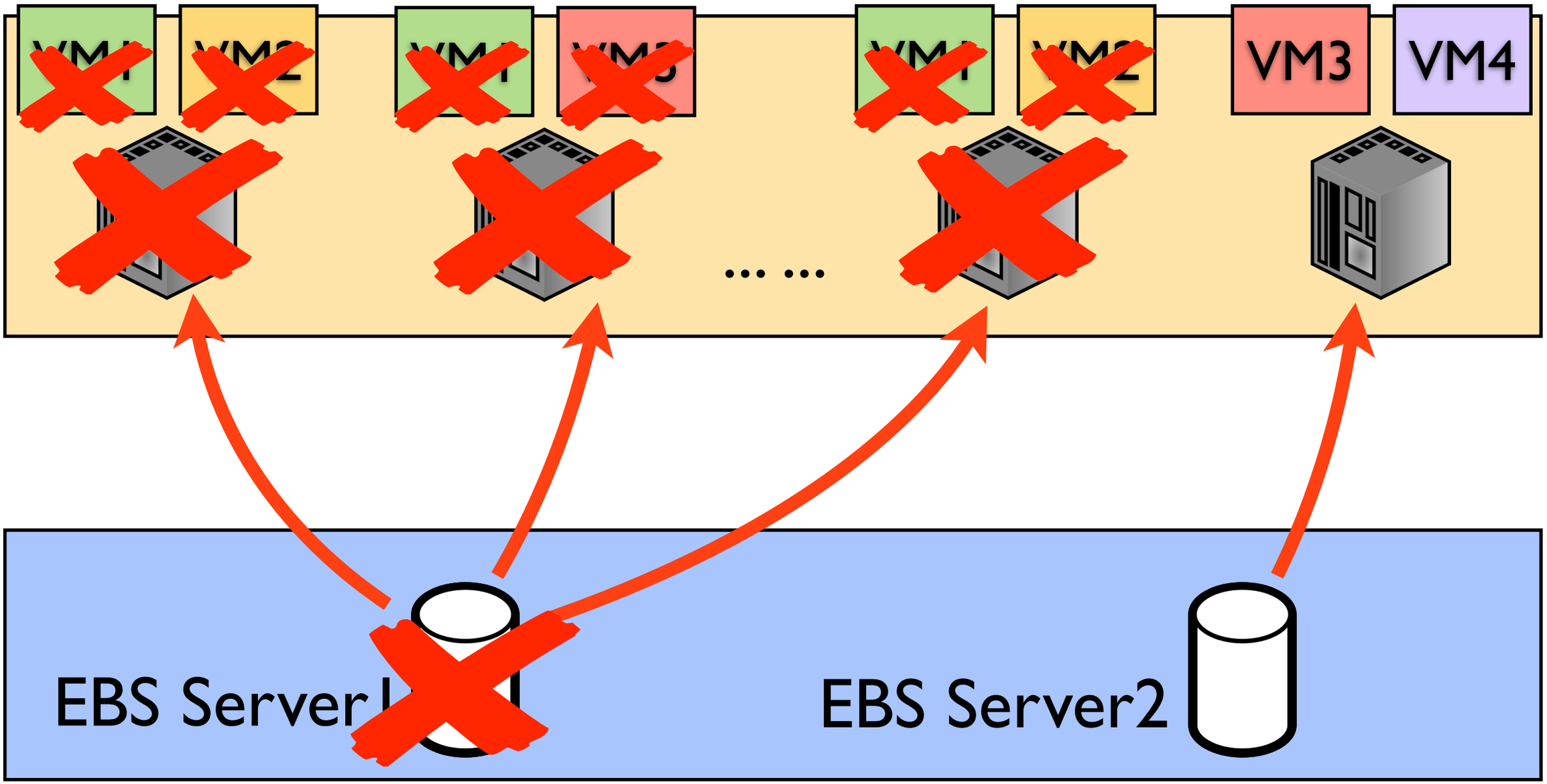
Realistic Example



Realistic Example



Realistic Example

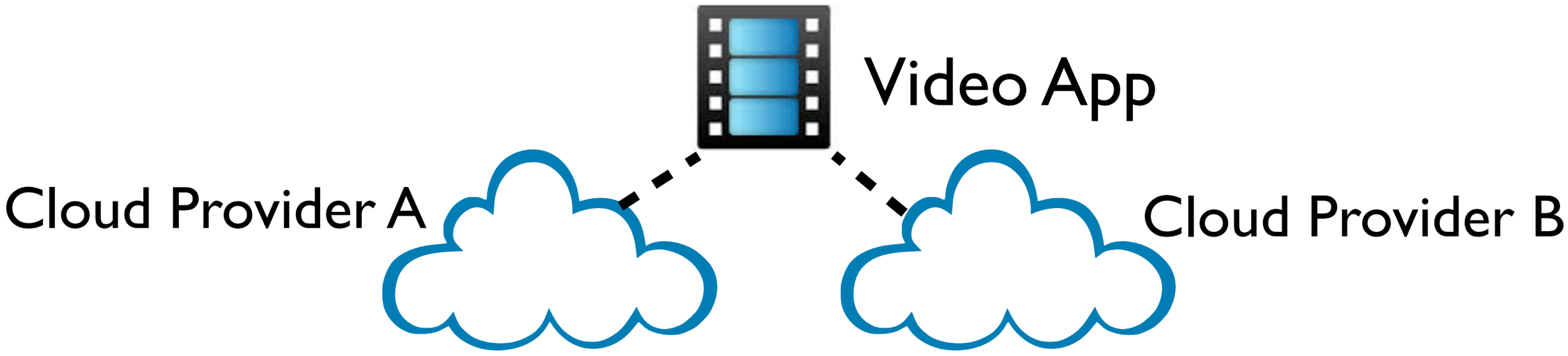


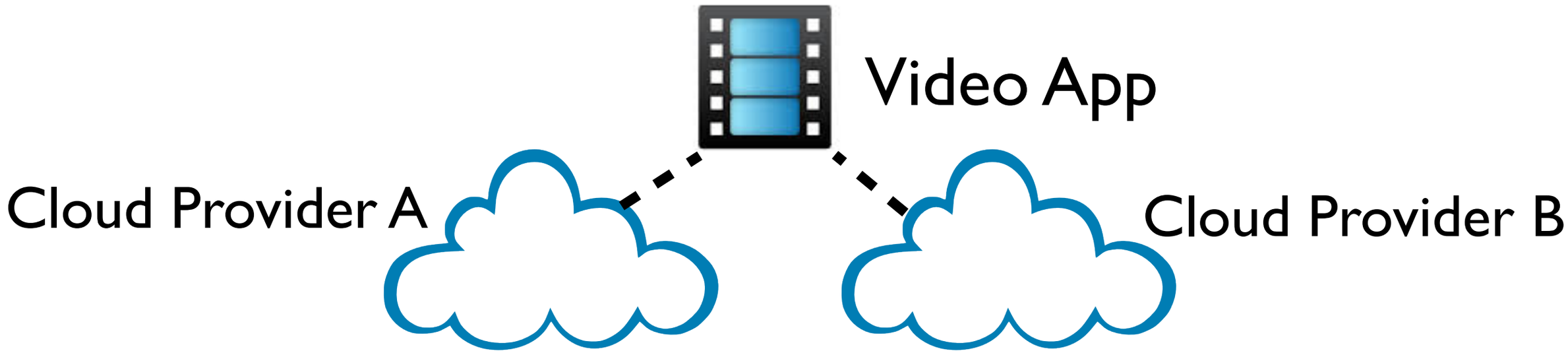
Even Worse

Lightning strikes Amazon's European cloud

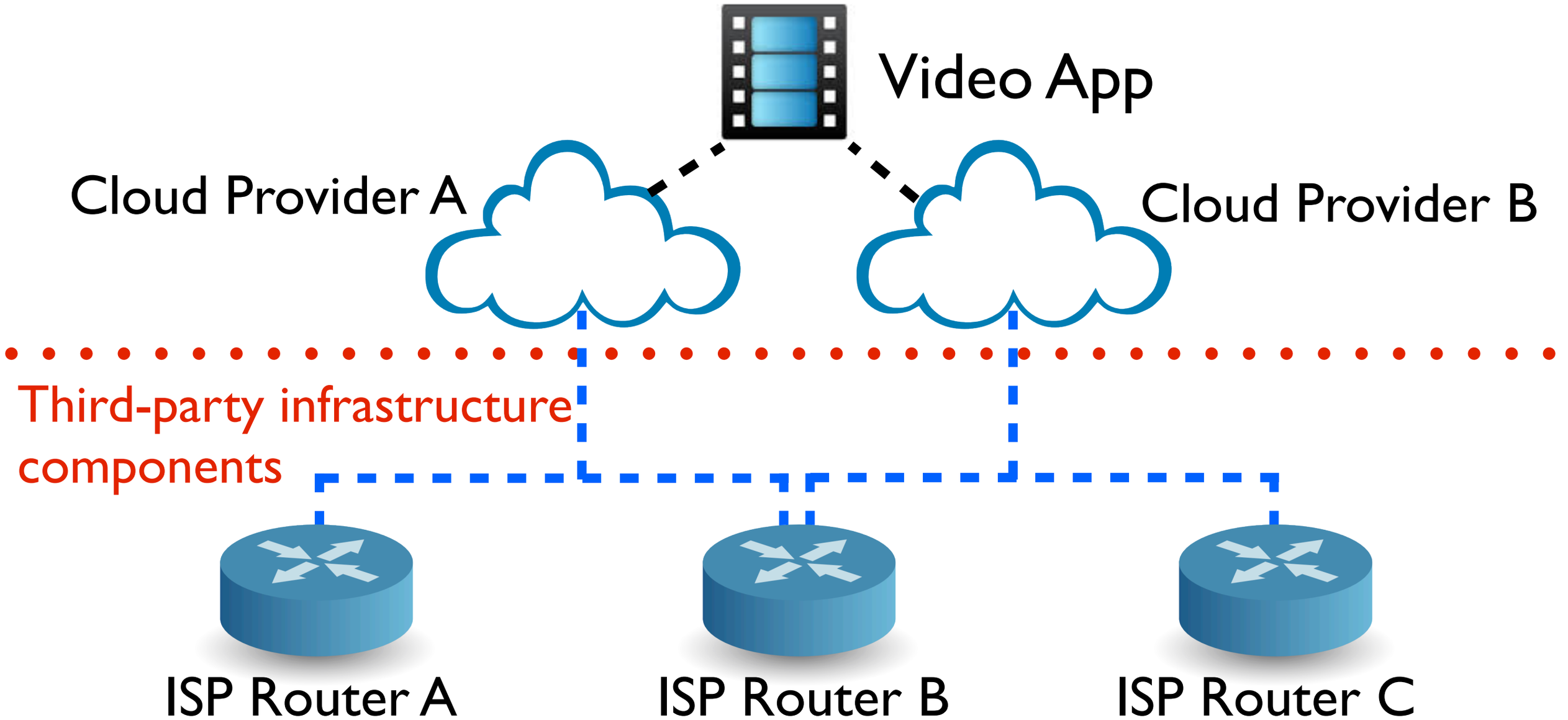
Summary: *The lightning strike damaged a power company's transformer, causing disruption to Amazon Web Services's European cloud, and may have affected Microsoft's BPOS as well*

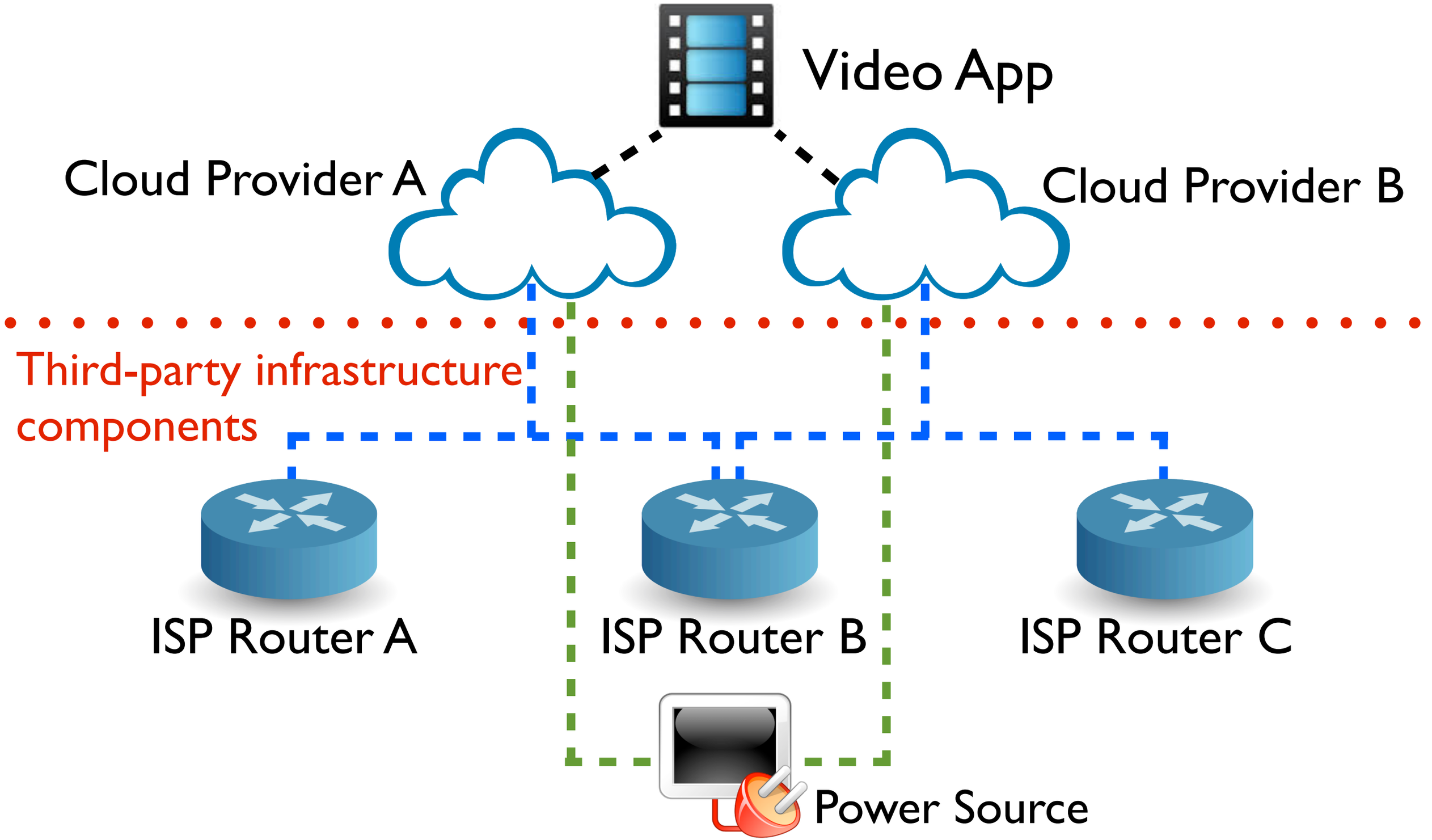
The outage, which Amazon Web Services (AWS) acknowledged on Sunday evening, affected its Dublin-based Elastic Compute Cloud (EC2) and Relational Database Service (RDS) cloud services, among others. The damage to the electricity infrastructure may have affected Microsoft's Business Productivity Online Services (BPOS) cloud as well, Microsoft said in a separate statement.

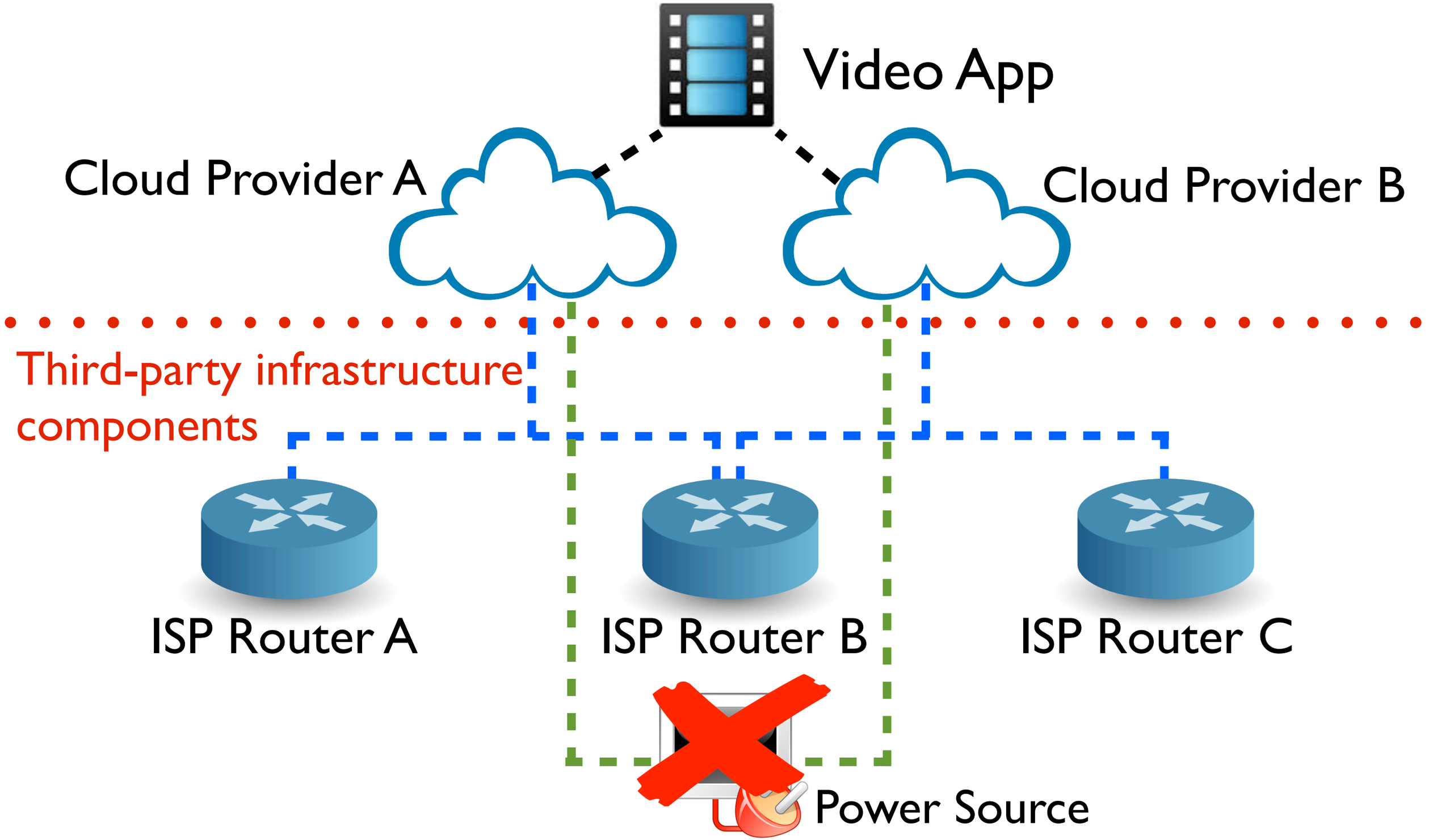

Video App
Cloud Provider A
Cloud Provider B

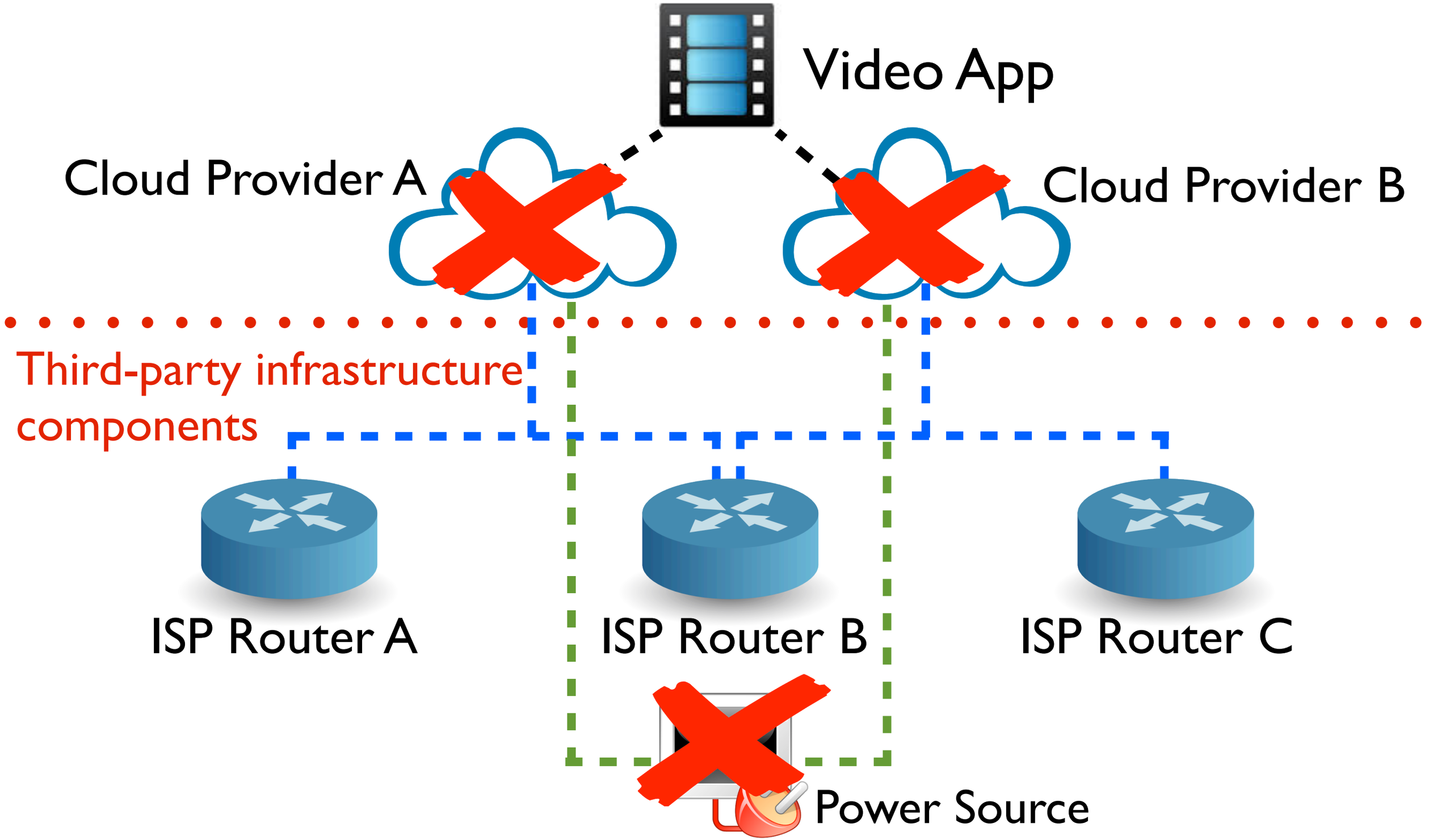


Third-party infrastructure components


Video App
Cloud Provider A
Cloud Provider B
Third-party infrastructure components
ISP Router A
ISP Router B
ISP Router C


Video App
Cloud Provider A
Cloud Provider B
Third-party infrastructure components
ISP Router A
ISP Router B
ISP Router C
Power Source


Video App
Cloud Provider A
Cloud Provider B
Third-party infrastructure components
ISP Router A
ISP Router B
ISP Router C
Power Source


Video App
Cloud Provider A
Cloud Provider B
Third-party infrastructure
components
ISP Router A
ISP Router B
ISP Router C
Power Source

Video App

Cl ler B

Third
comp

Cloud providers do not usually share information about their dependencies

ISP Router A

ISP Router B

ISP Router C

Power Source

Existing Efforts

- Cloud providers allocate or tolerate failures via:
  - diagnosis systems;
  - fault-tolerant systems.

Existing Efforts

- Cloud providers allocate or tolerate failures via:
  - diagnosis systems;
  - fault-tolerant systems.
- Solving the problem after outage occurs.

Existing Efforts

- Cloud providers allocate or tolerate failures via:
  - diagnosis systems;
  - fault-tolerant systems.
- Solving the problem after outage occurs.
- Prevent correlated failures before outage occurs.

Existing Efforts

- Cloud providers allocate or tolerate failures via:
  - diagnosis systems;
  - fault-tolerant systems.
- Solving the problem after outage occurs.
- Prevent correlated failures before outage occurs.

Independence-as-a-Service
(INDaaS)

INDaaS Workflow

Service Provider, Alice 

INDaaS

A Given Redundancy Configuration

INDaaS Workflow

Service Provider, Alice 

INDaaS

Two-Way Redundancy Configuration

Dependency
Data Source1

Dependency
Data Source2

INDaaS Workflow

Service Provider, Alice 

Independence of this two-way redundancy ?

INDaaS

Two-Way Redundancy Configuration

Dependency Data Source1

Dependency Data Source2

INDaaS Workflow

Service Provider, Alice

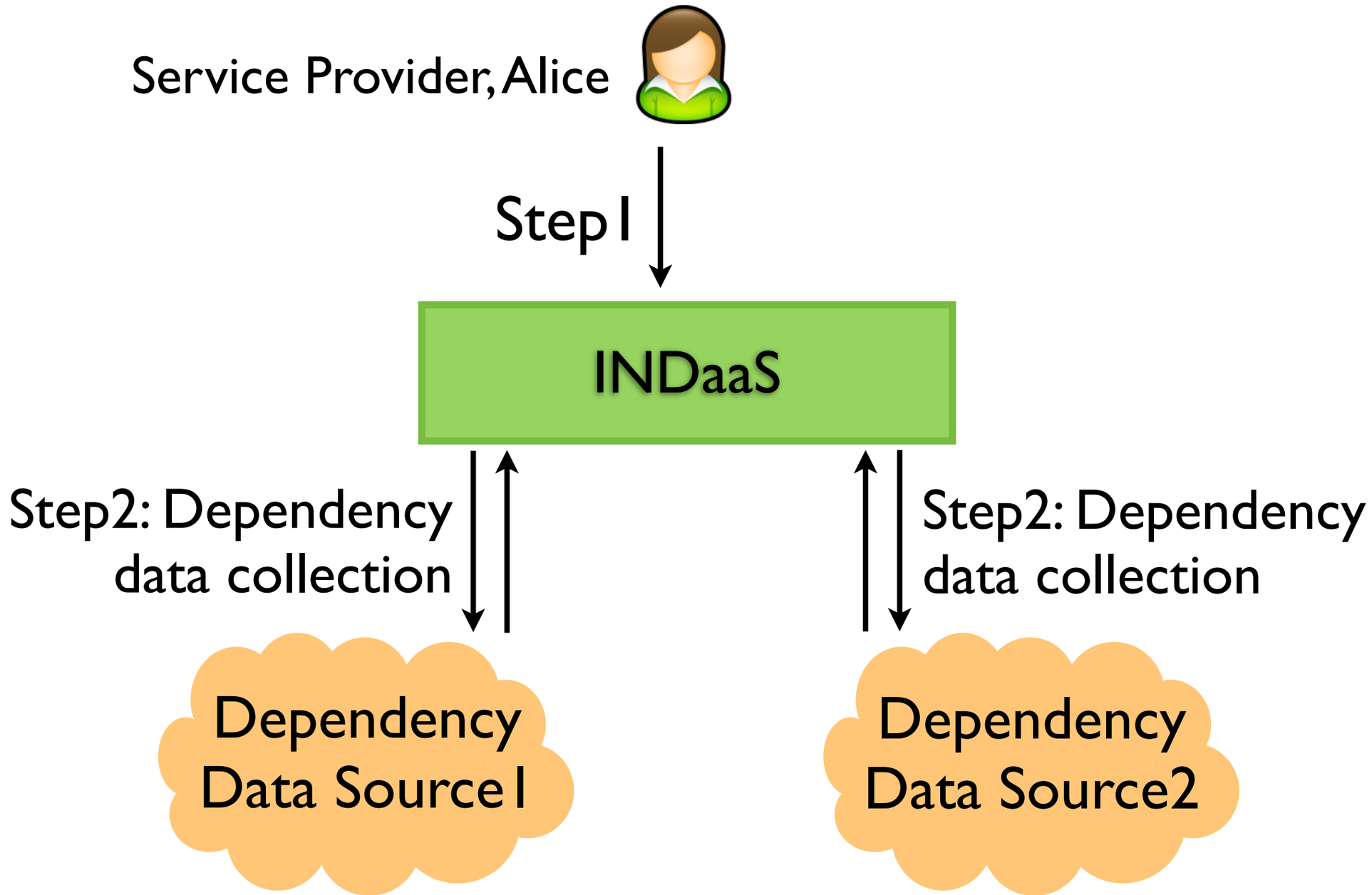
Step 1: Specification Submission

INDaaS

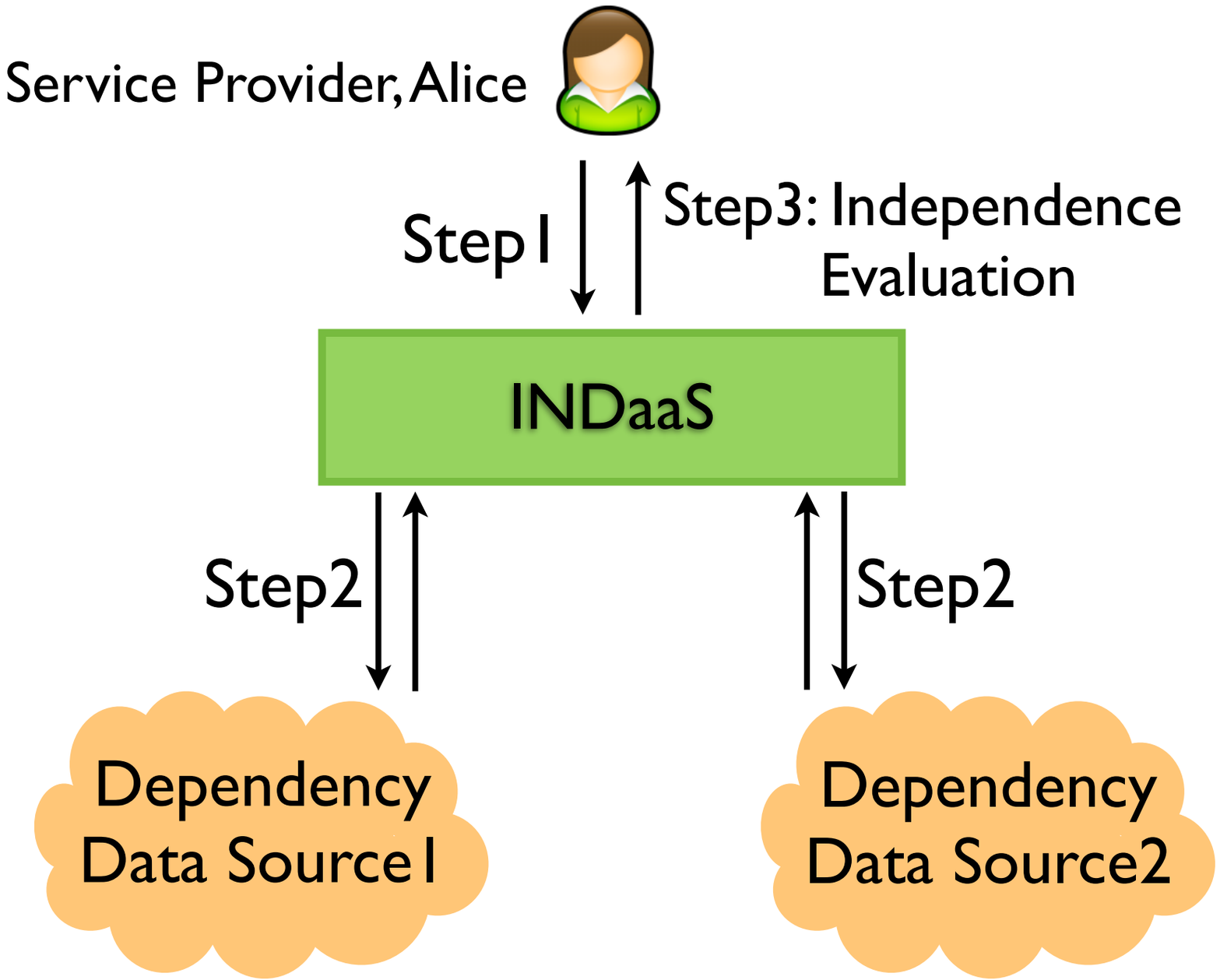
Dependency Data Source1

Dependency Data Source2

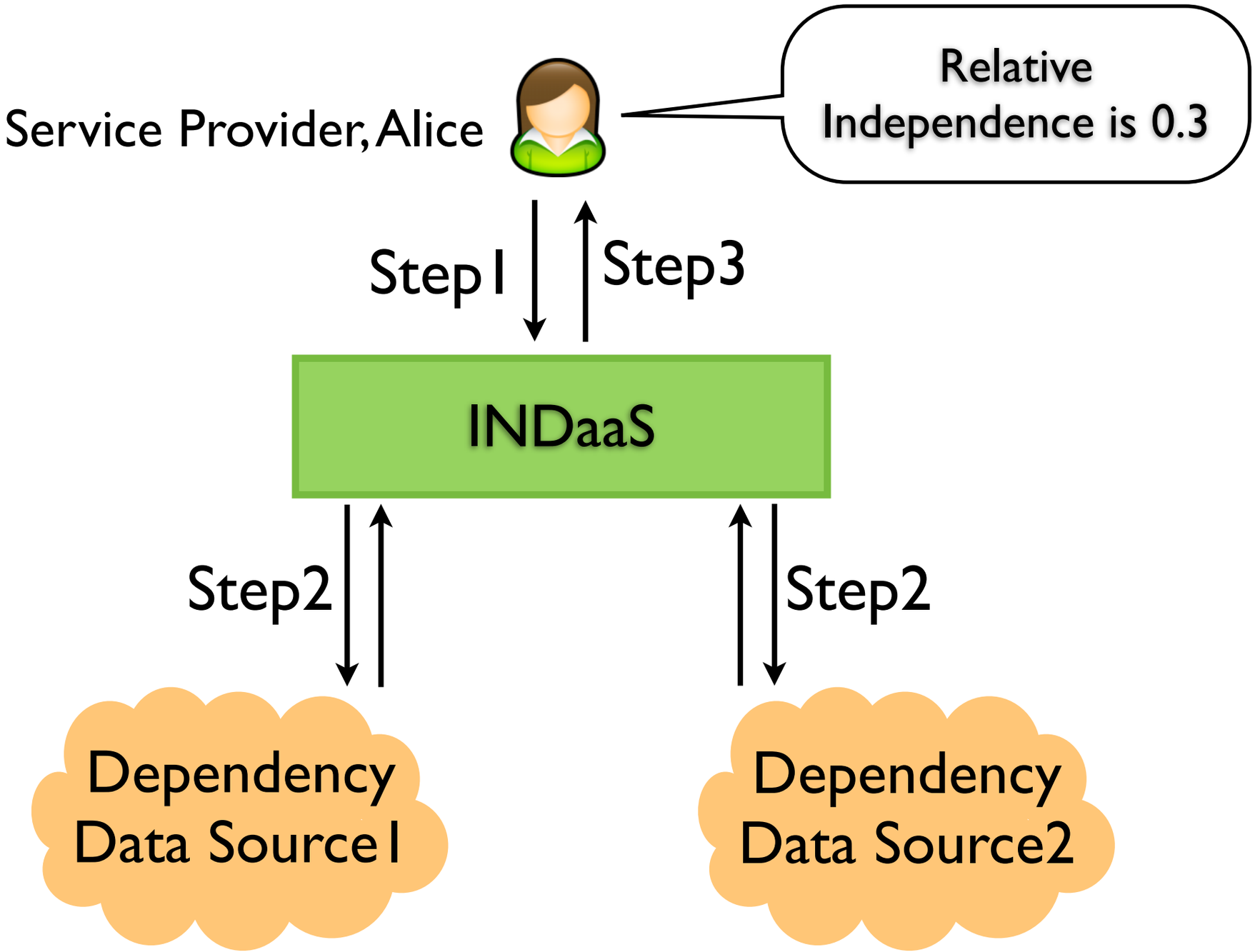
INDaaS Workflow



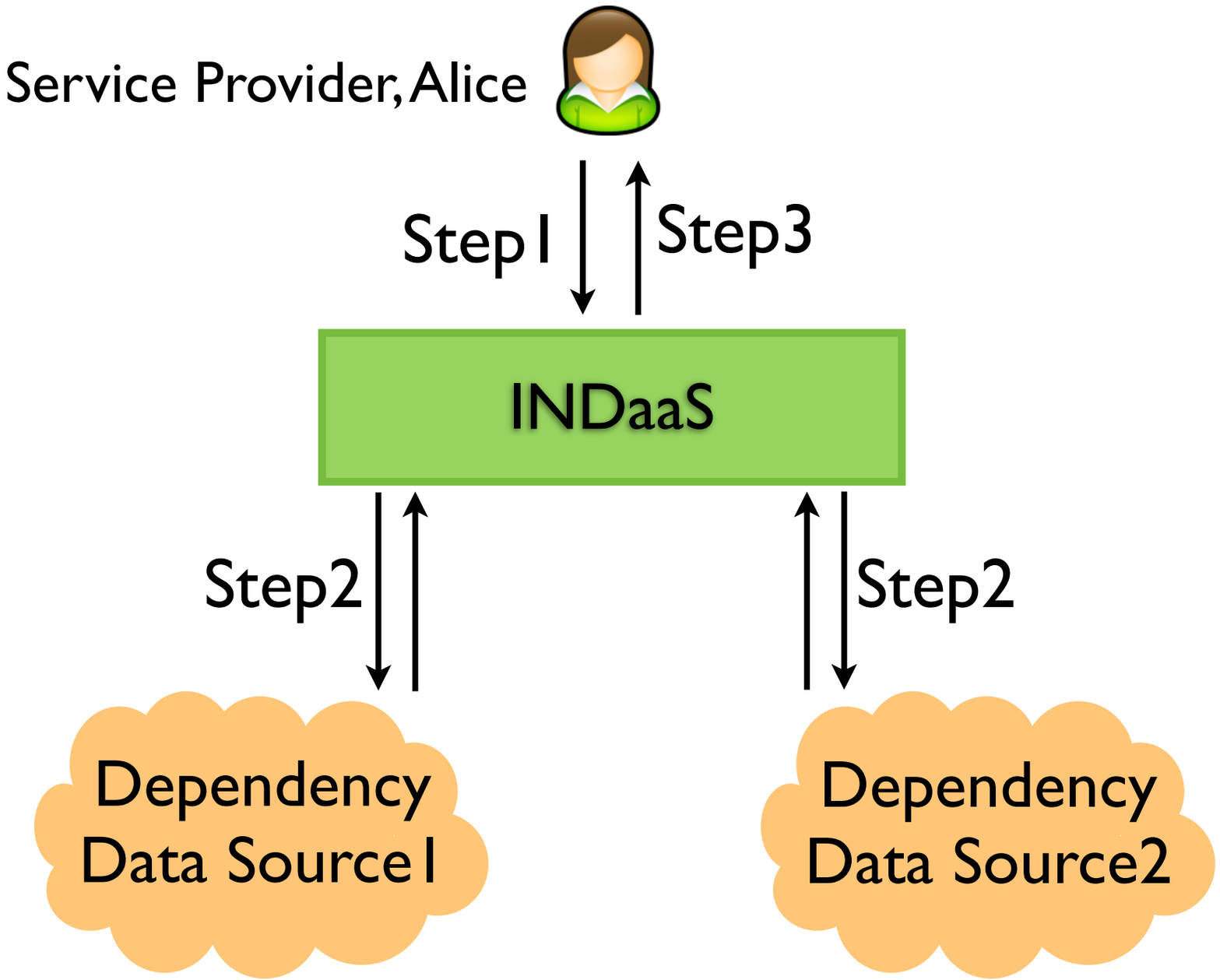
INDaaS Workflow



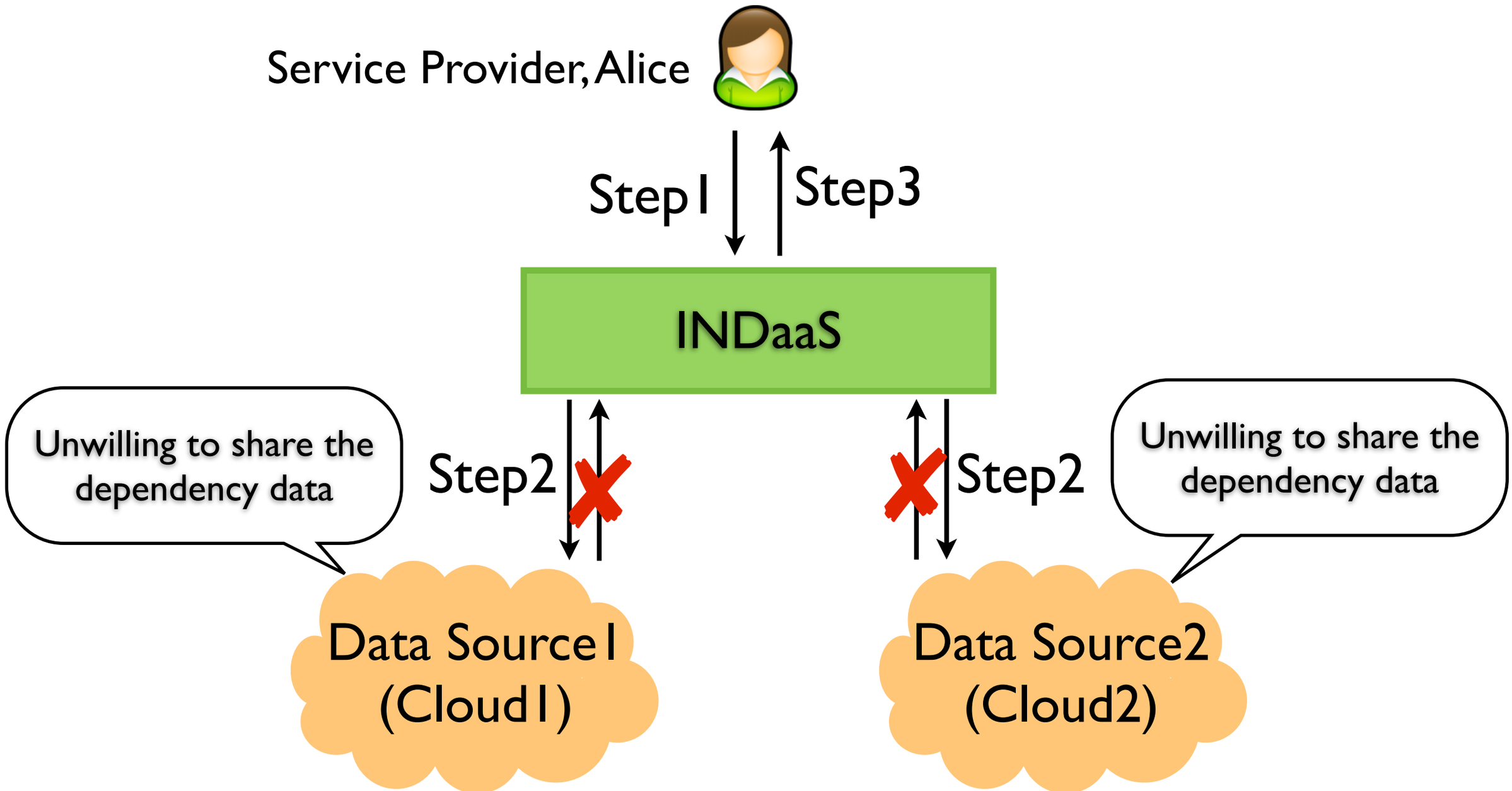
INDaaS Workflow



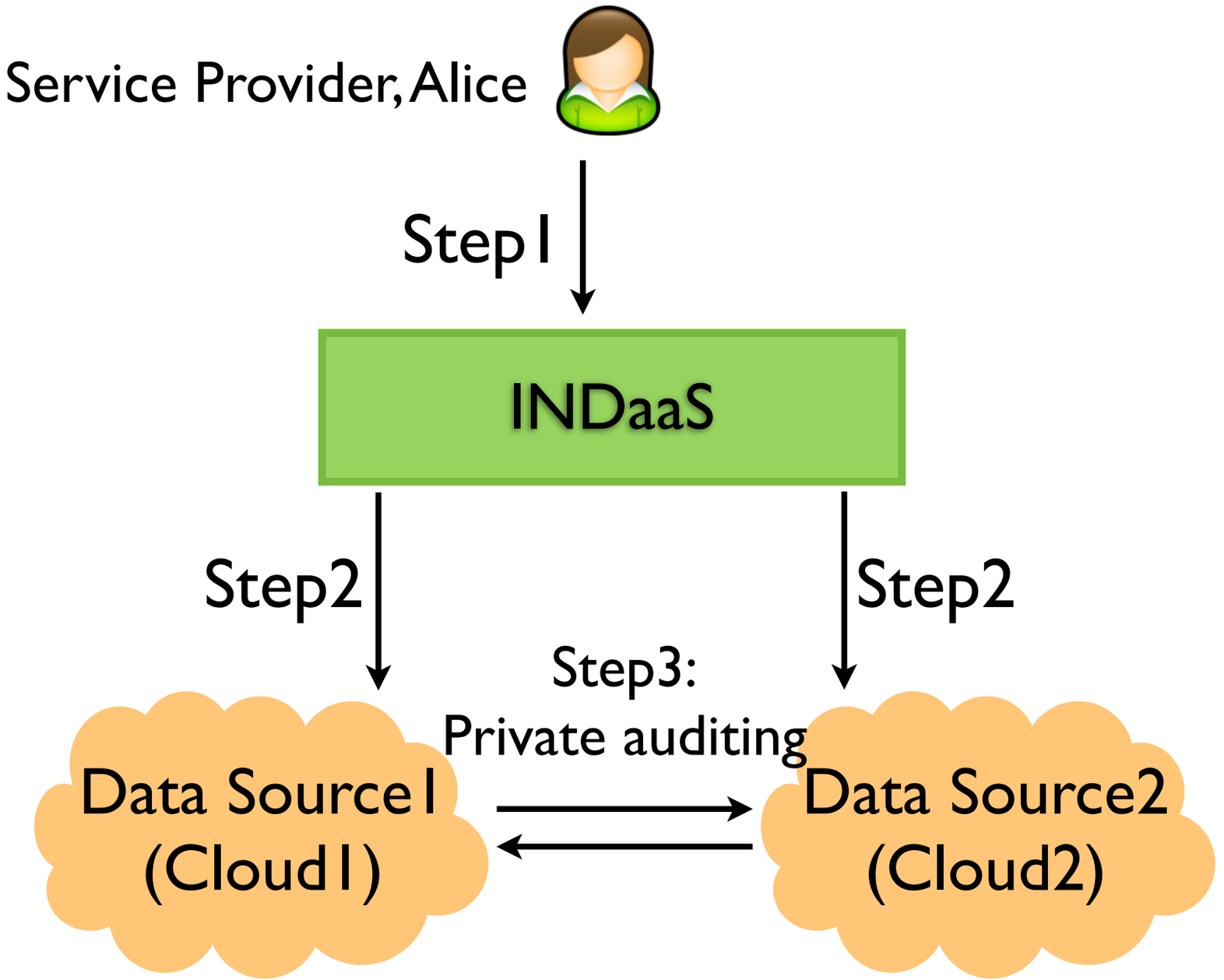
Multiple Cloud Providers



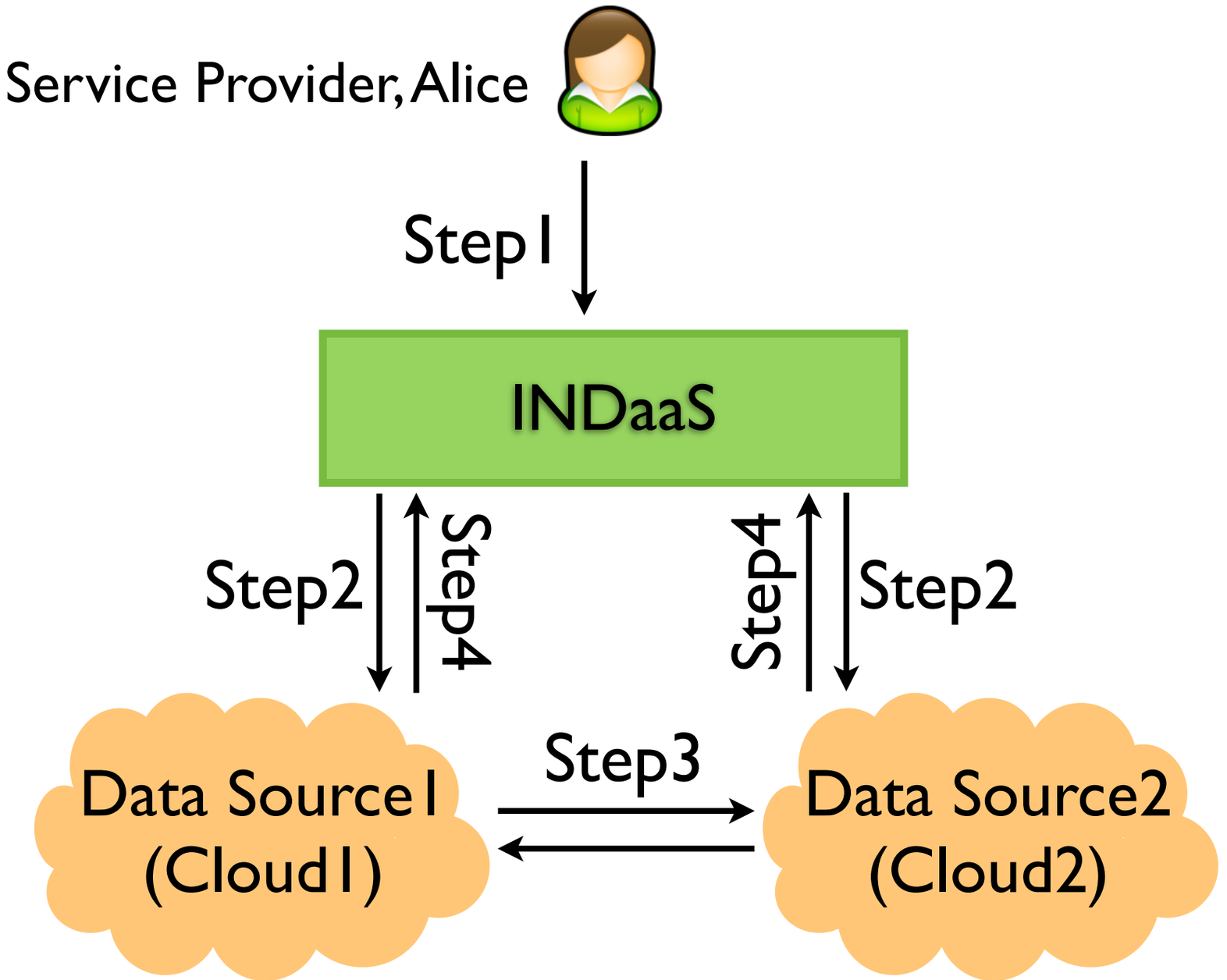
Multiple Cloud Providers



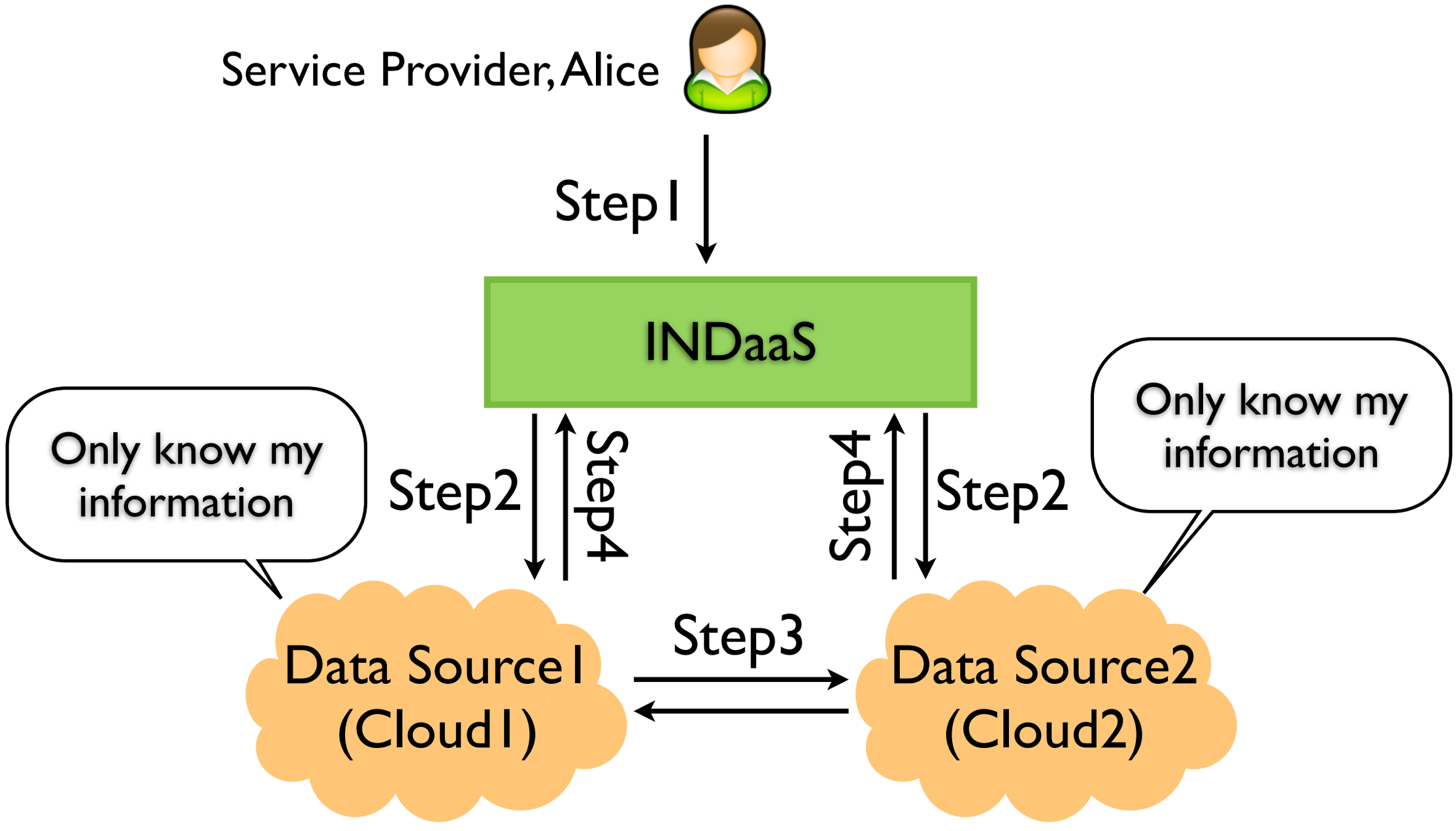
Private Independence Evaluation



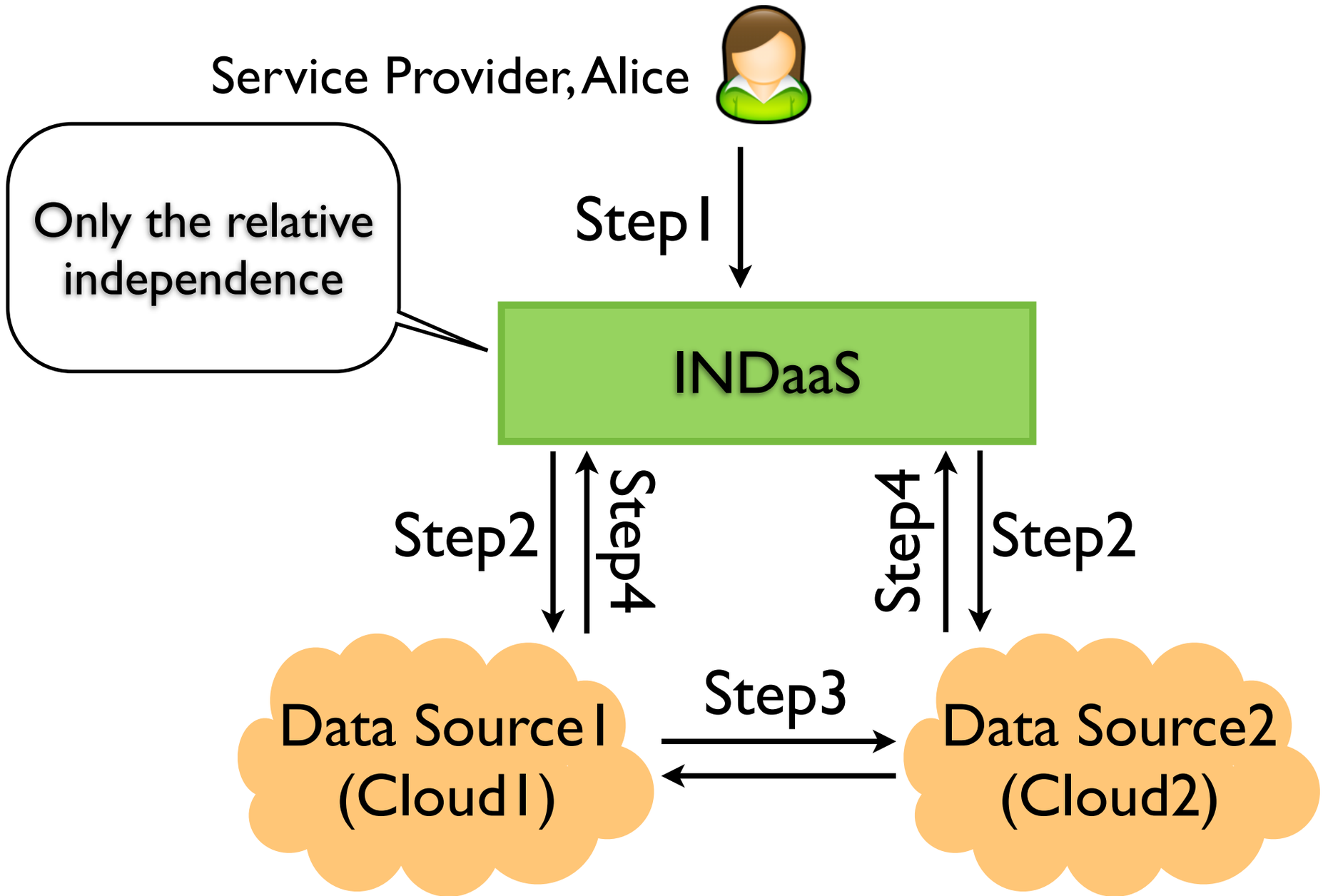
Private Independence Evaluation



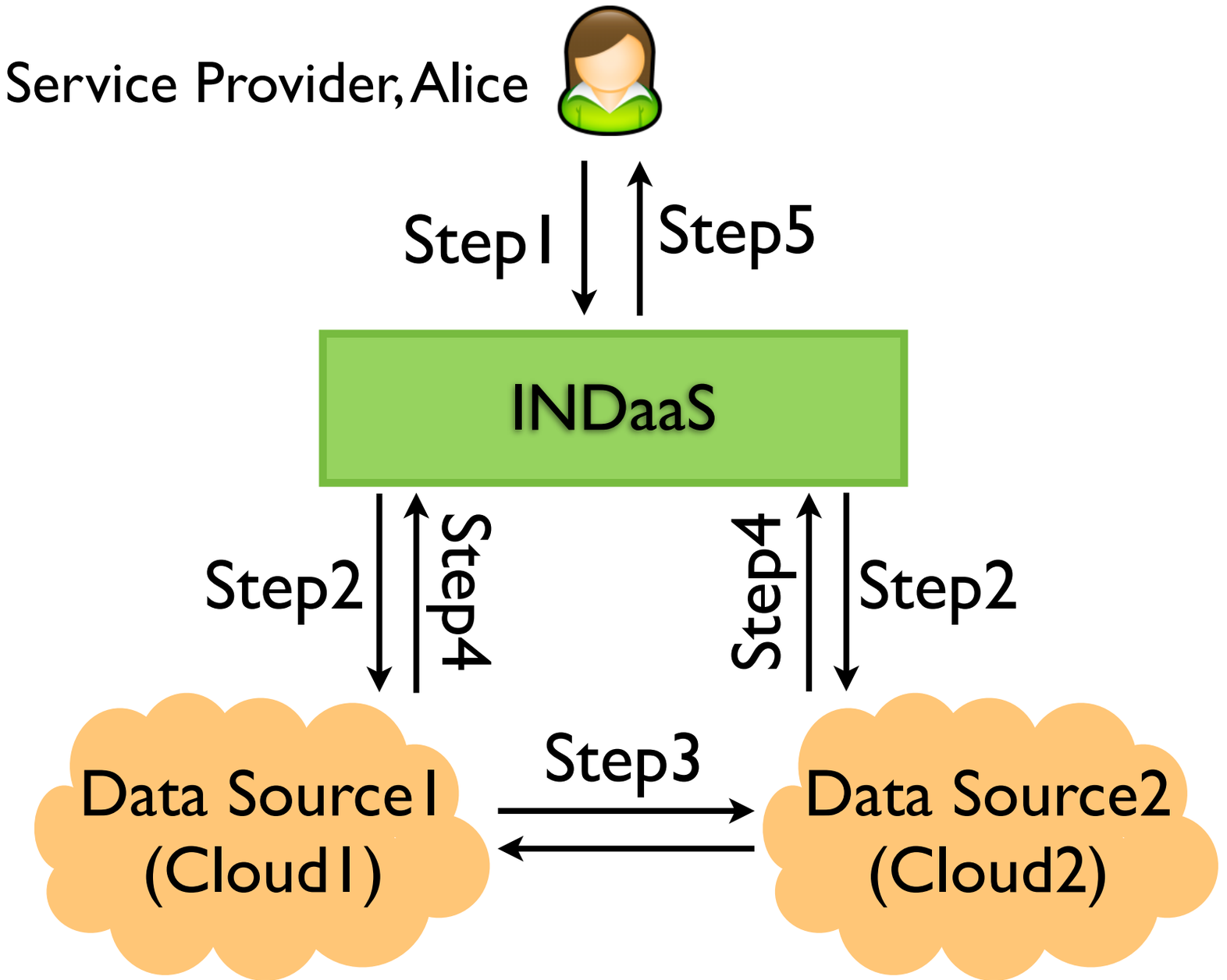
Private Independence Evaluation



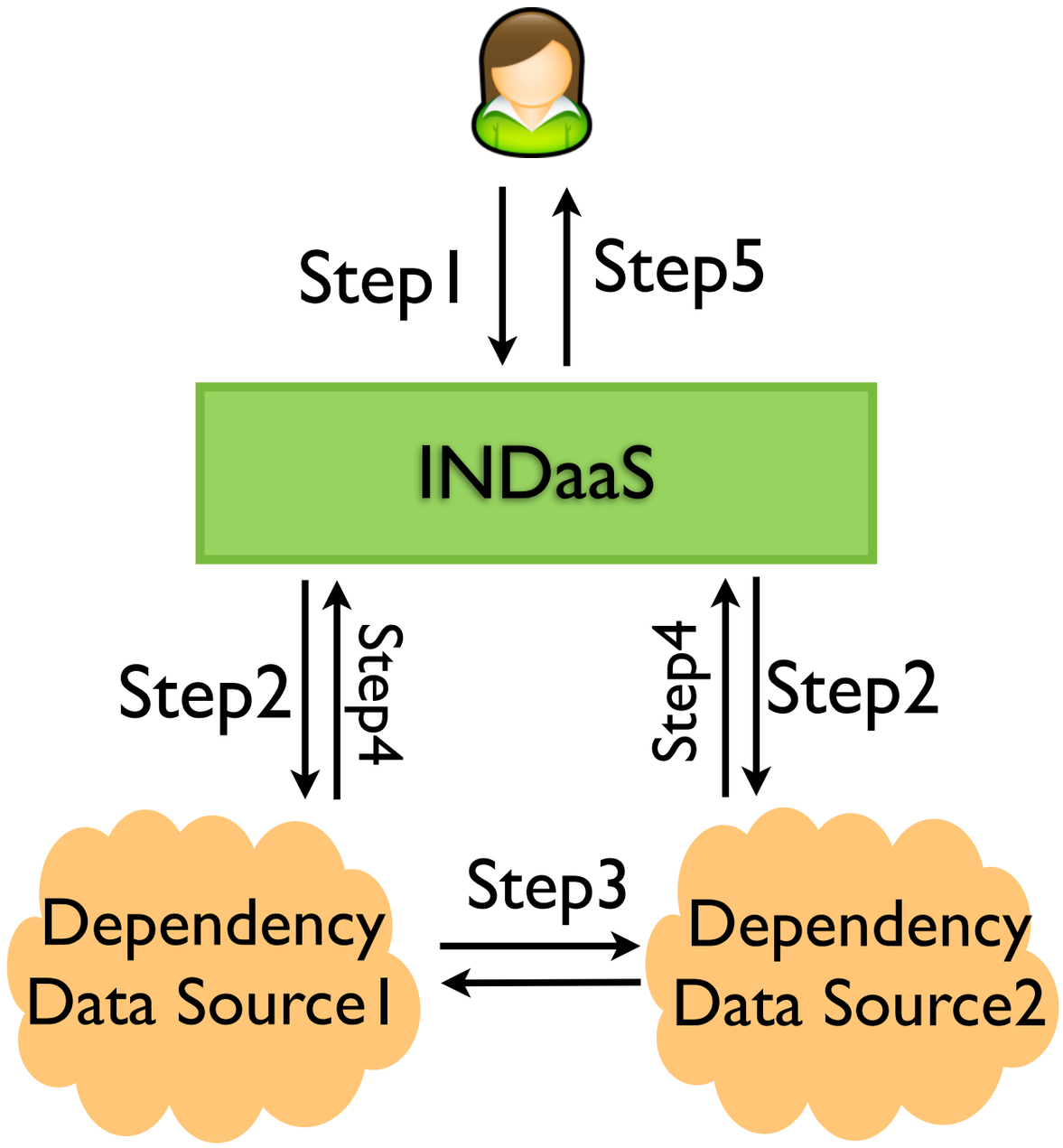
Private Independence Evaluation



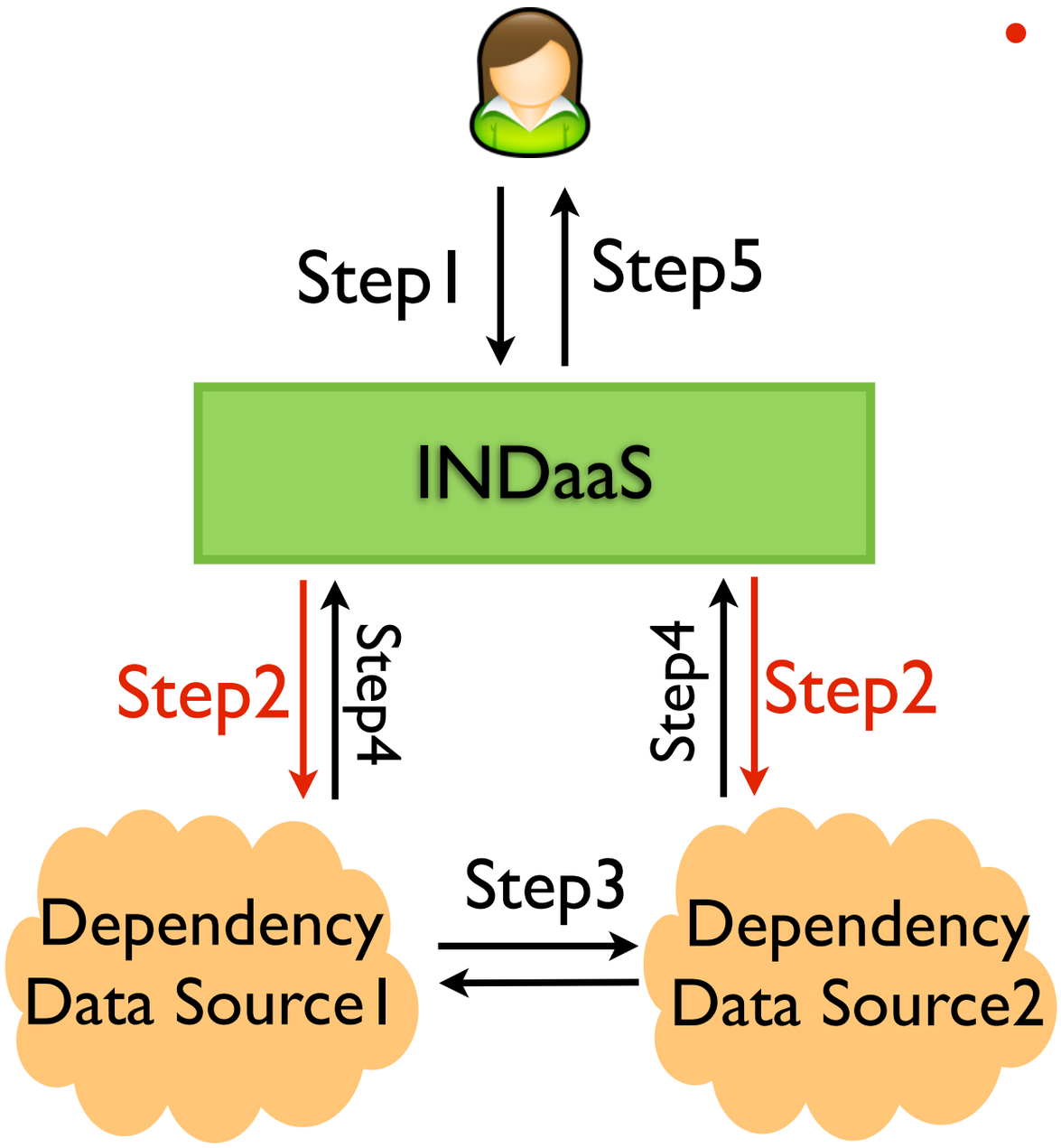
Private Independence Evaluation



Technical Challenges

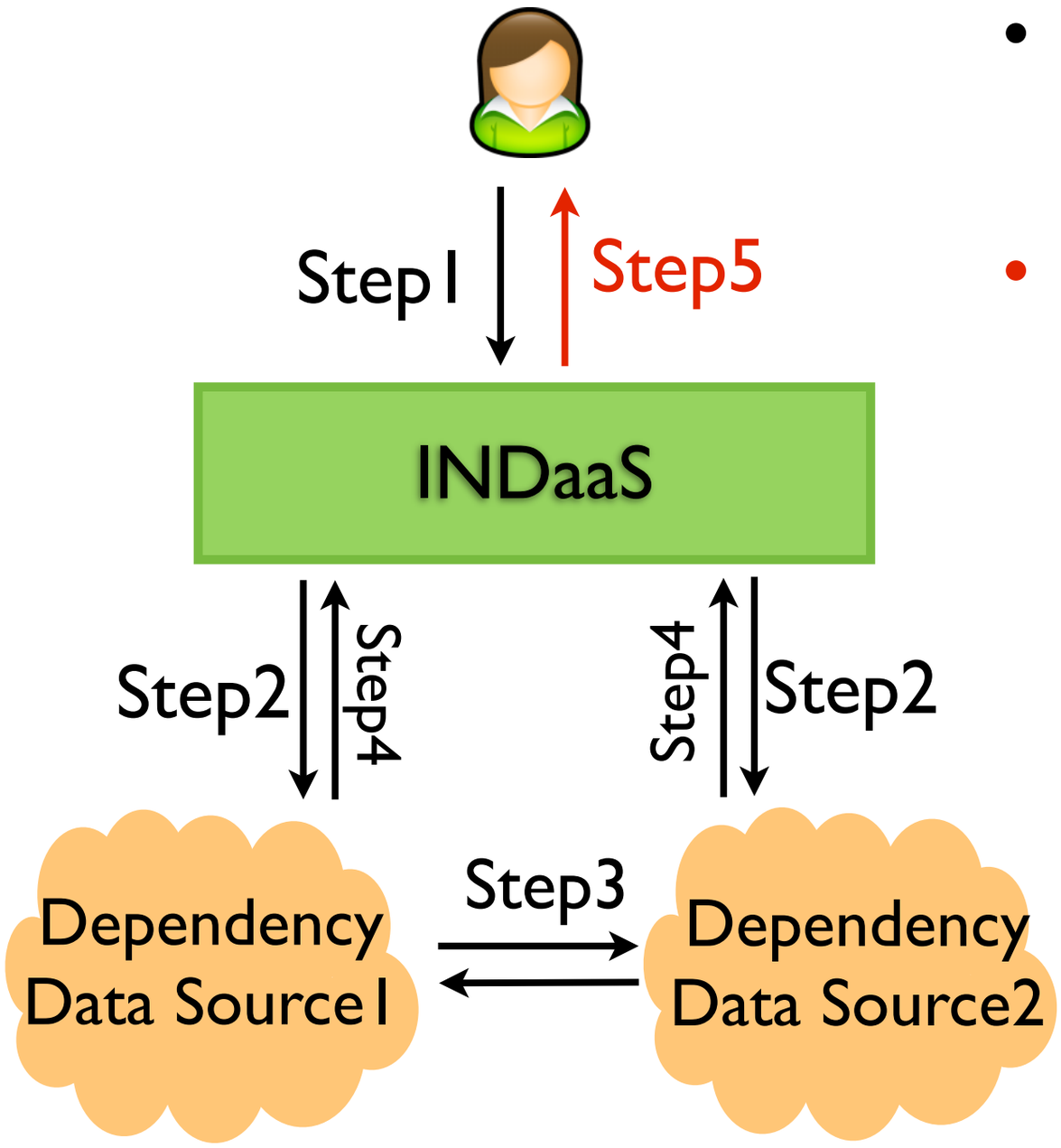


Technical Challenges



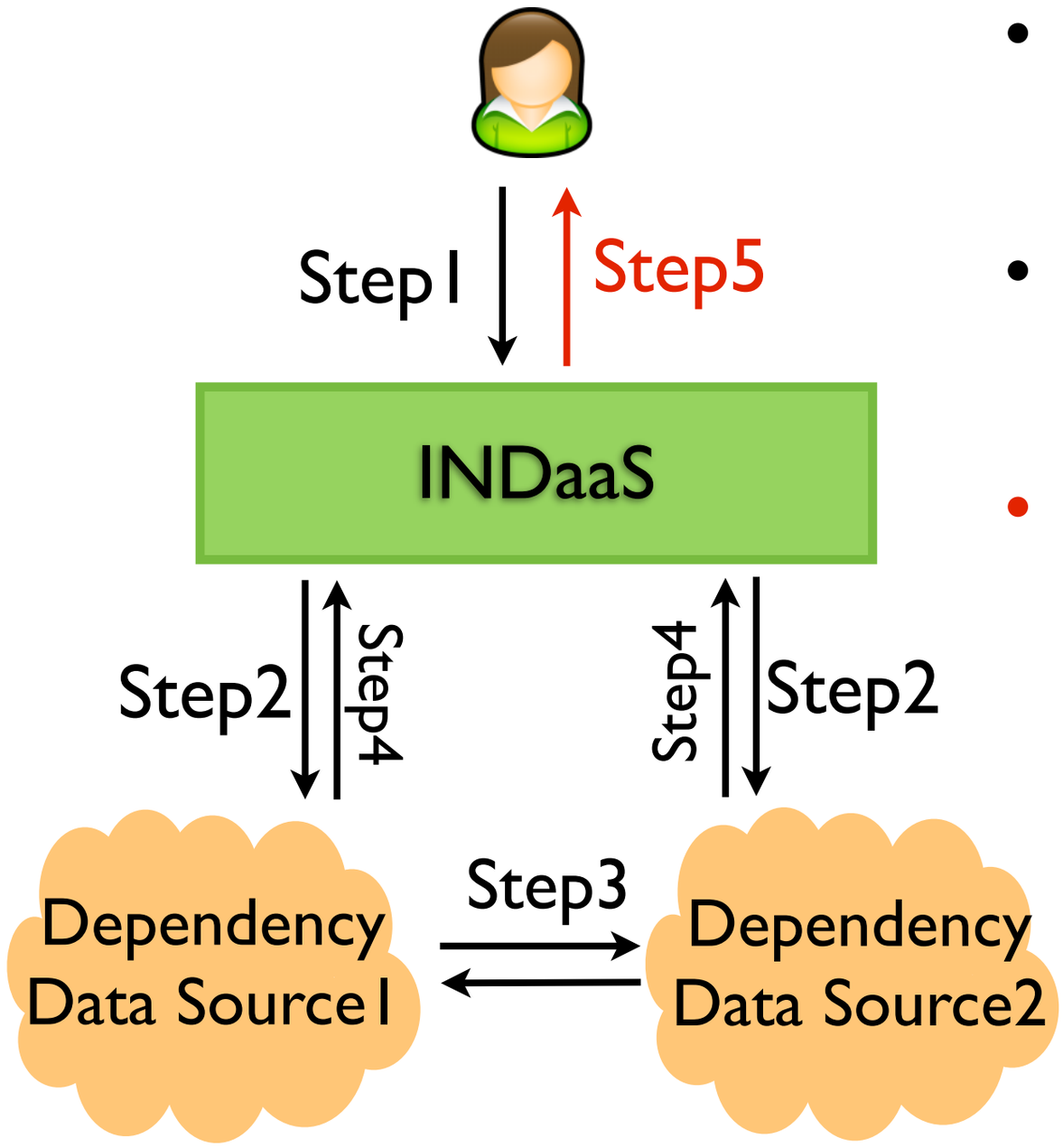
- #1: Dependency collections
  - Solution: Reusing existing tools

Technical Challenges



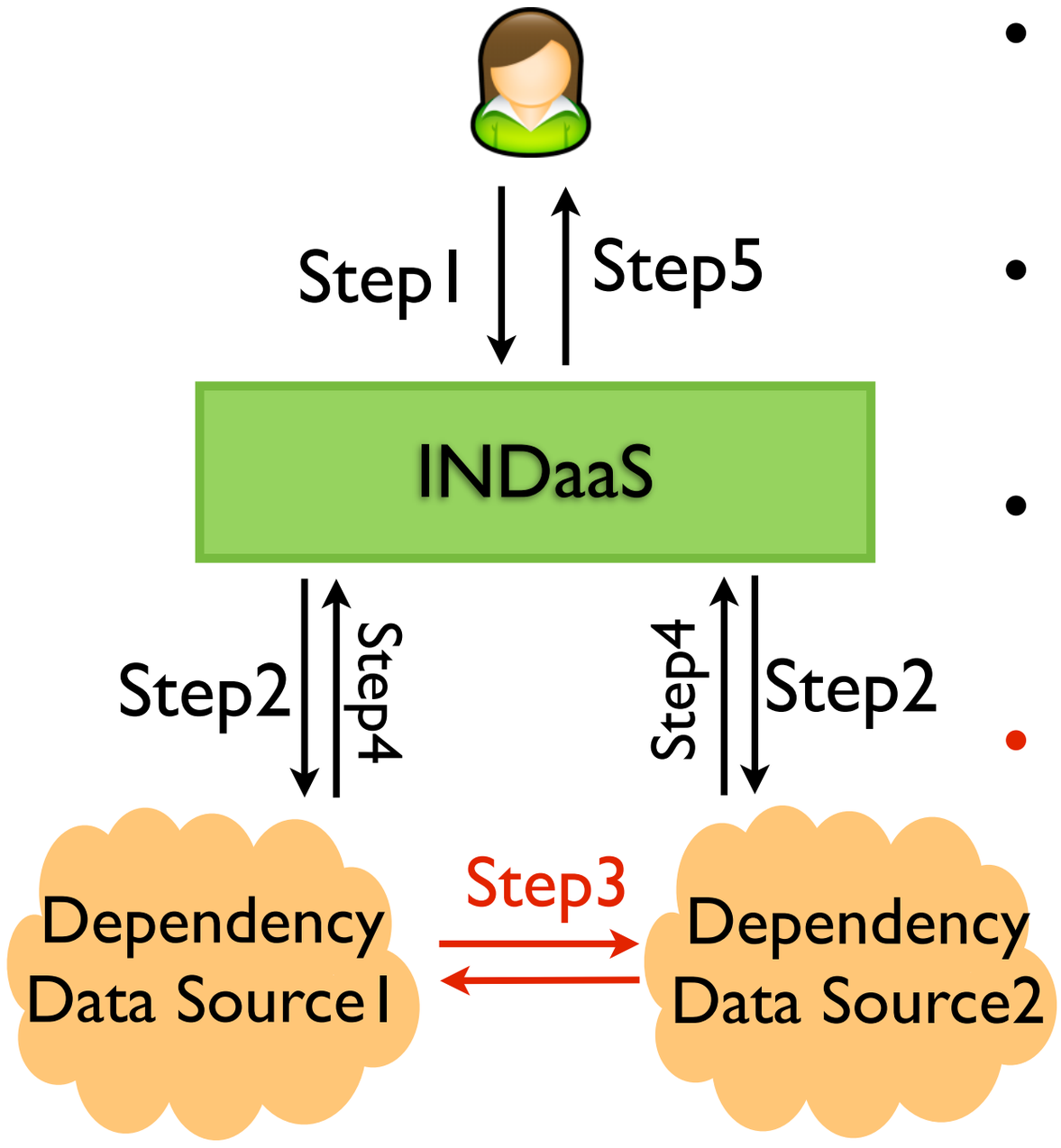
- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs

Technical Challenges



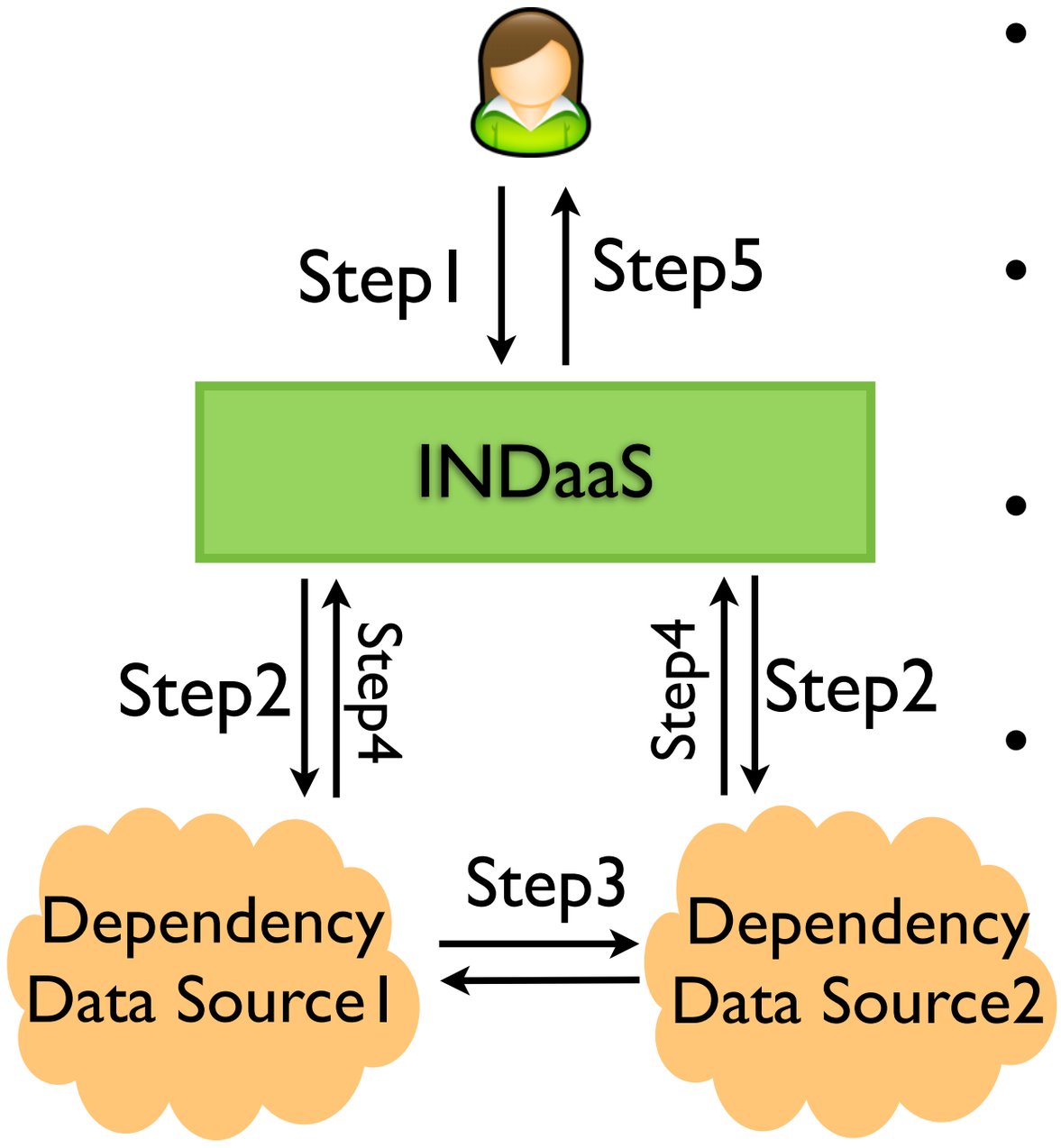
- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm

Technical Challenges



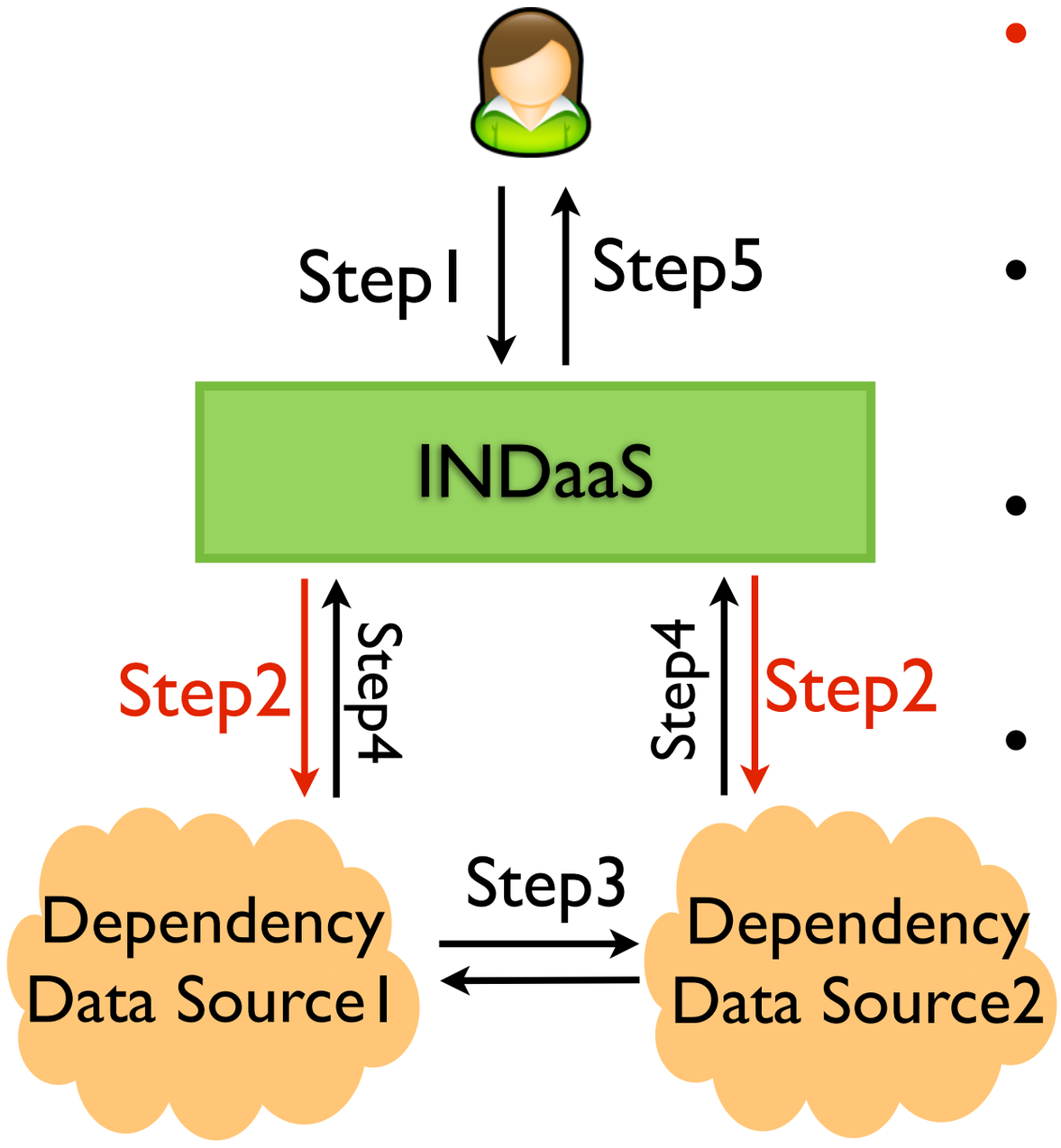
- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

RoadMap



- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

RoadMap



- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

Dependency Data Collections

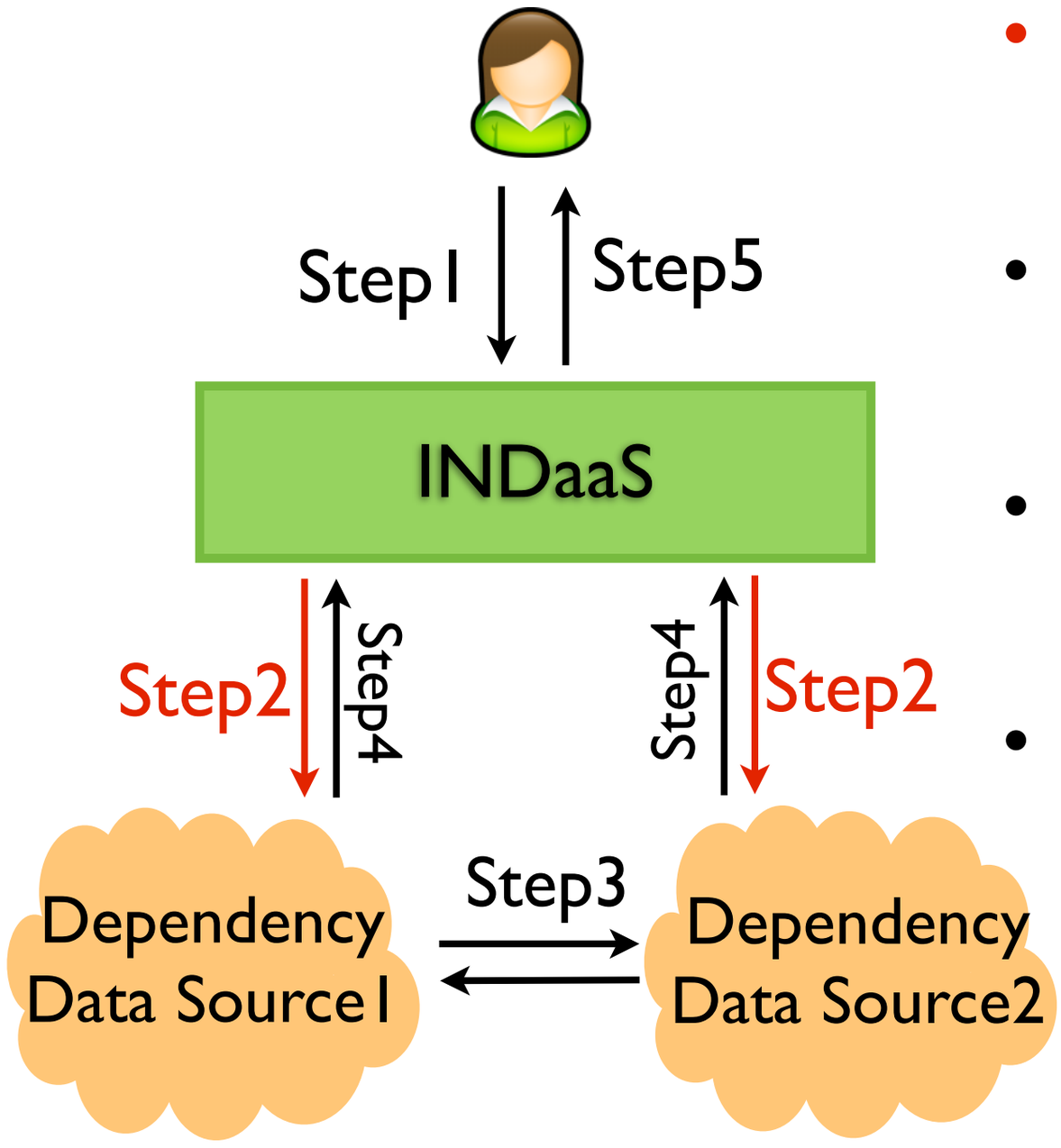
- Reuse existing data collection tools:
  - Convert the outputs to uniform format.
  - Three types of format: NET, HW and SW.

Our defined format

Type	Dependency Expression
Network	<src=”S” dst=”D” route=”x,y,z”/>
Hardware	<hw=”H” type=”T” dep=”x”/>
Software	<pgm=”S” hw=”H” dep=”x,y,z”/>

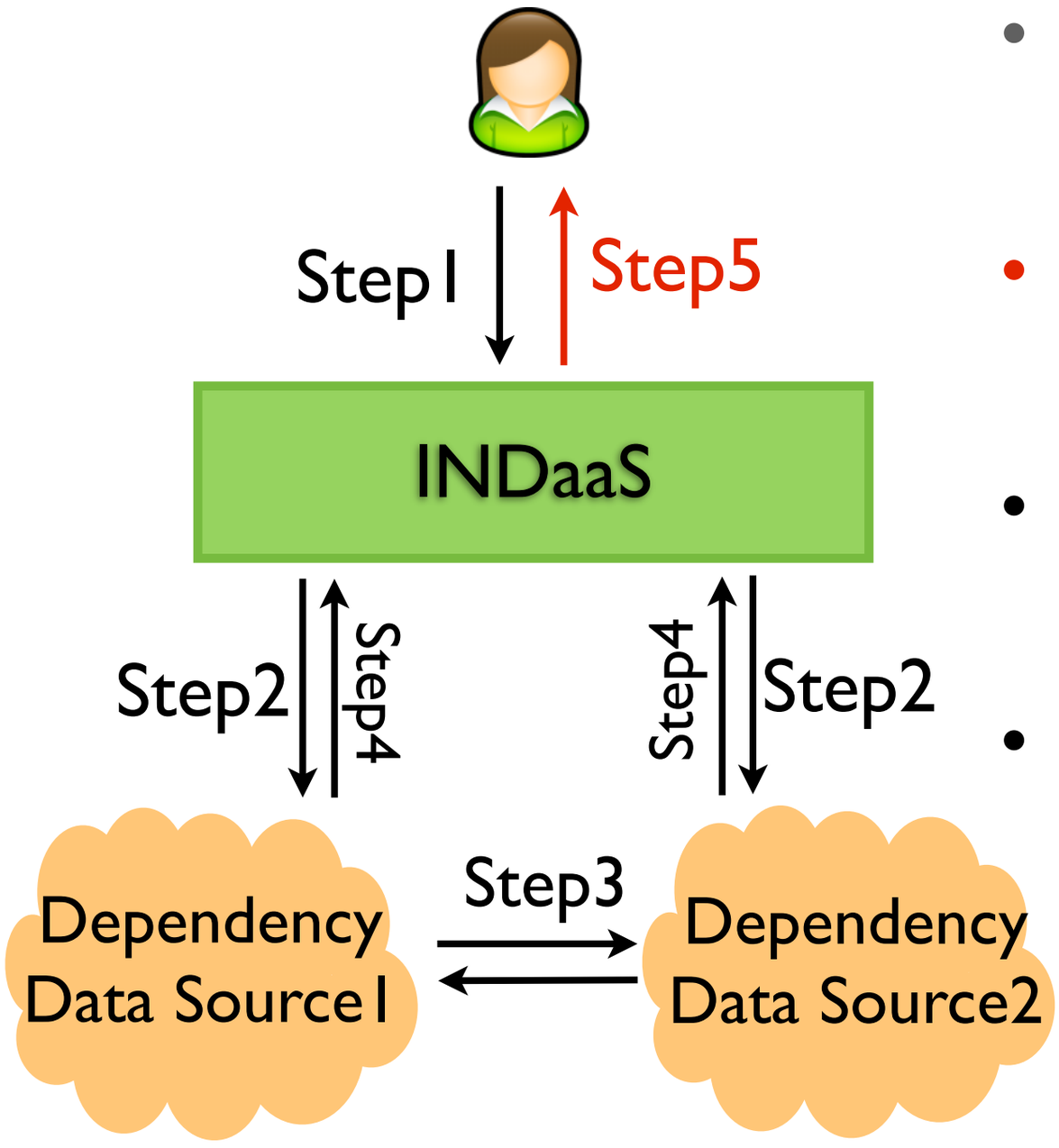
Please see our paper for more details

RoadMap



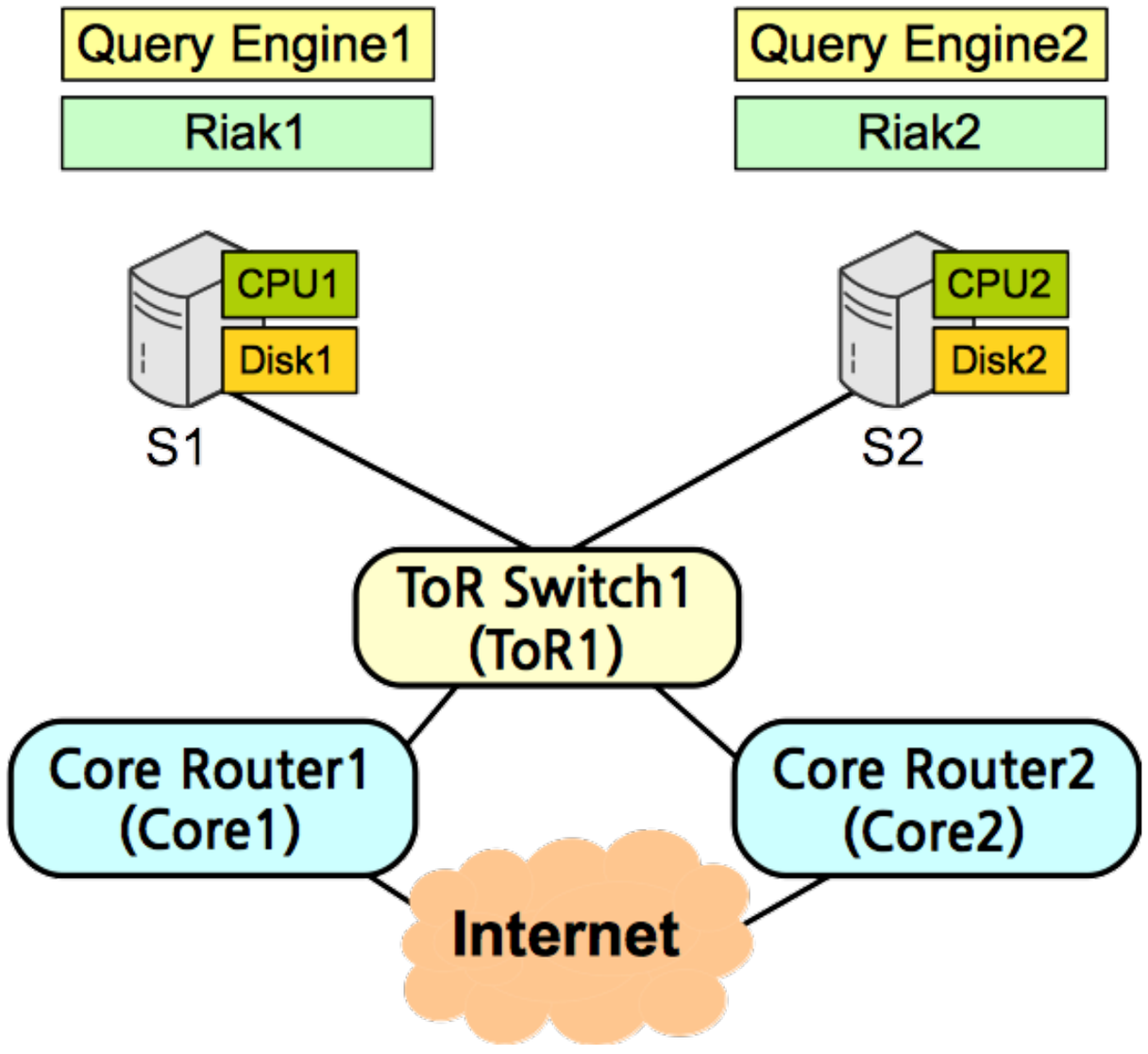
- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

RoadMap

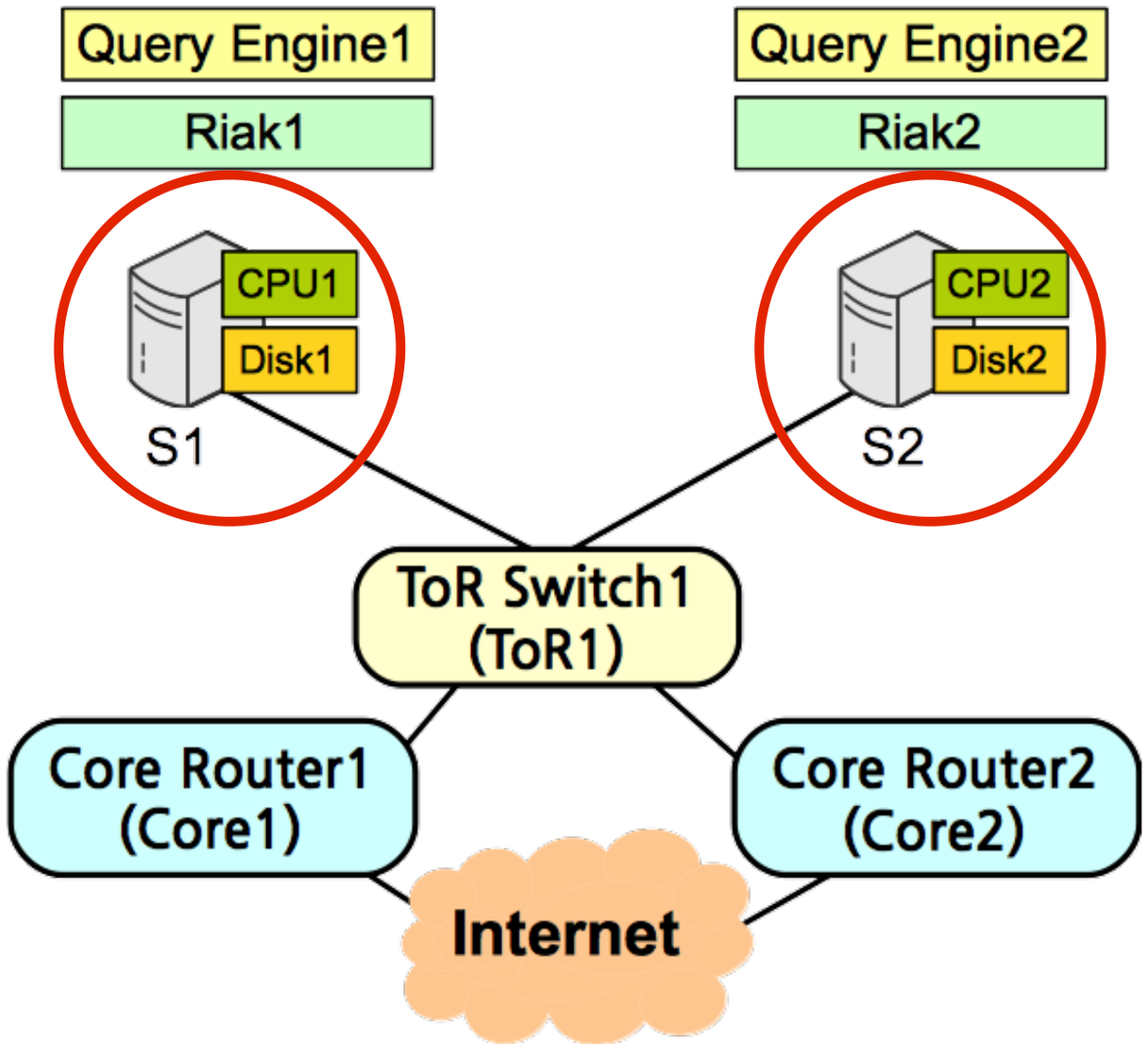


- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

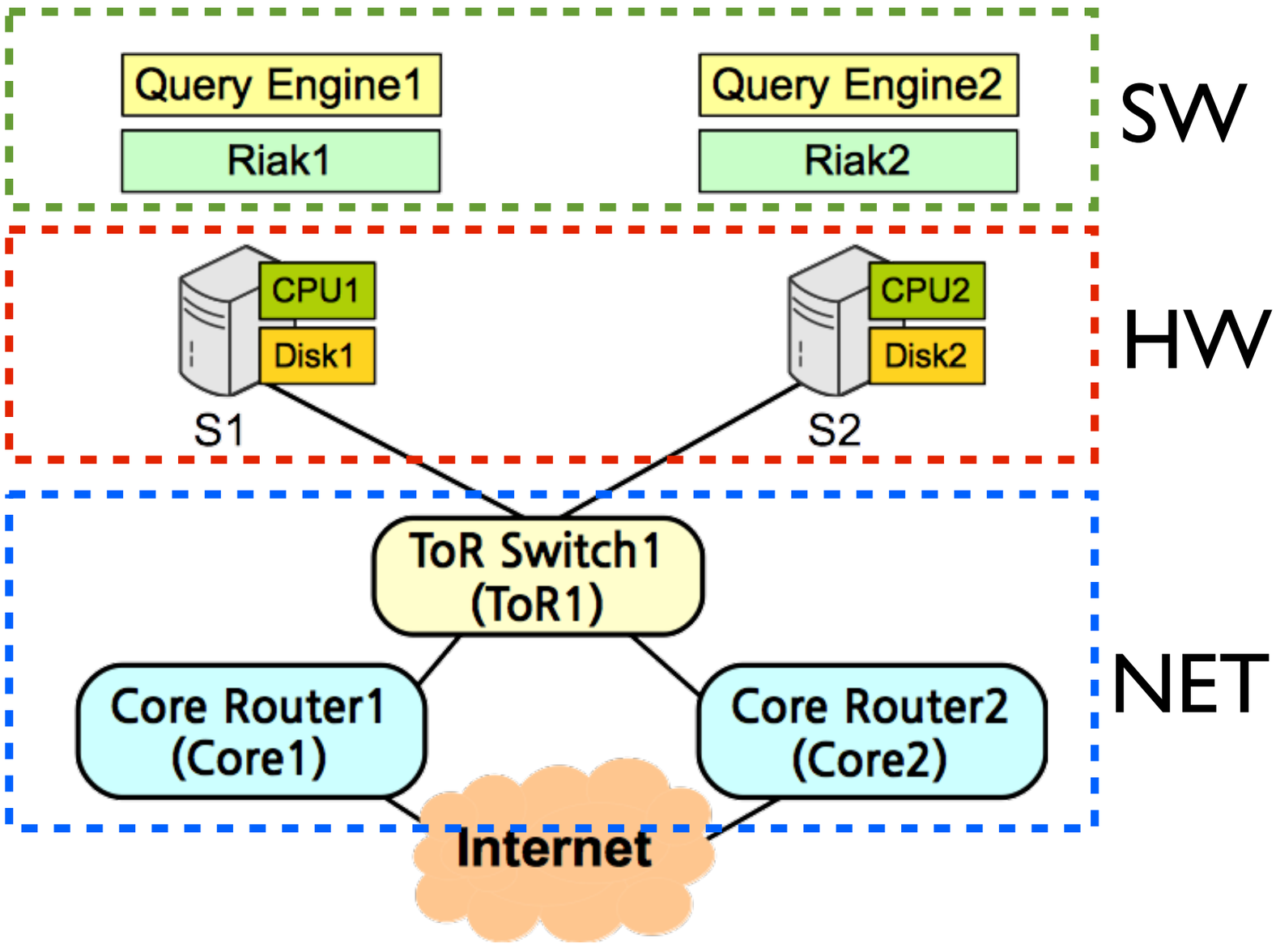
Example Redundancy



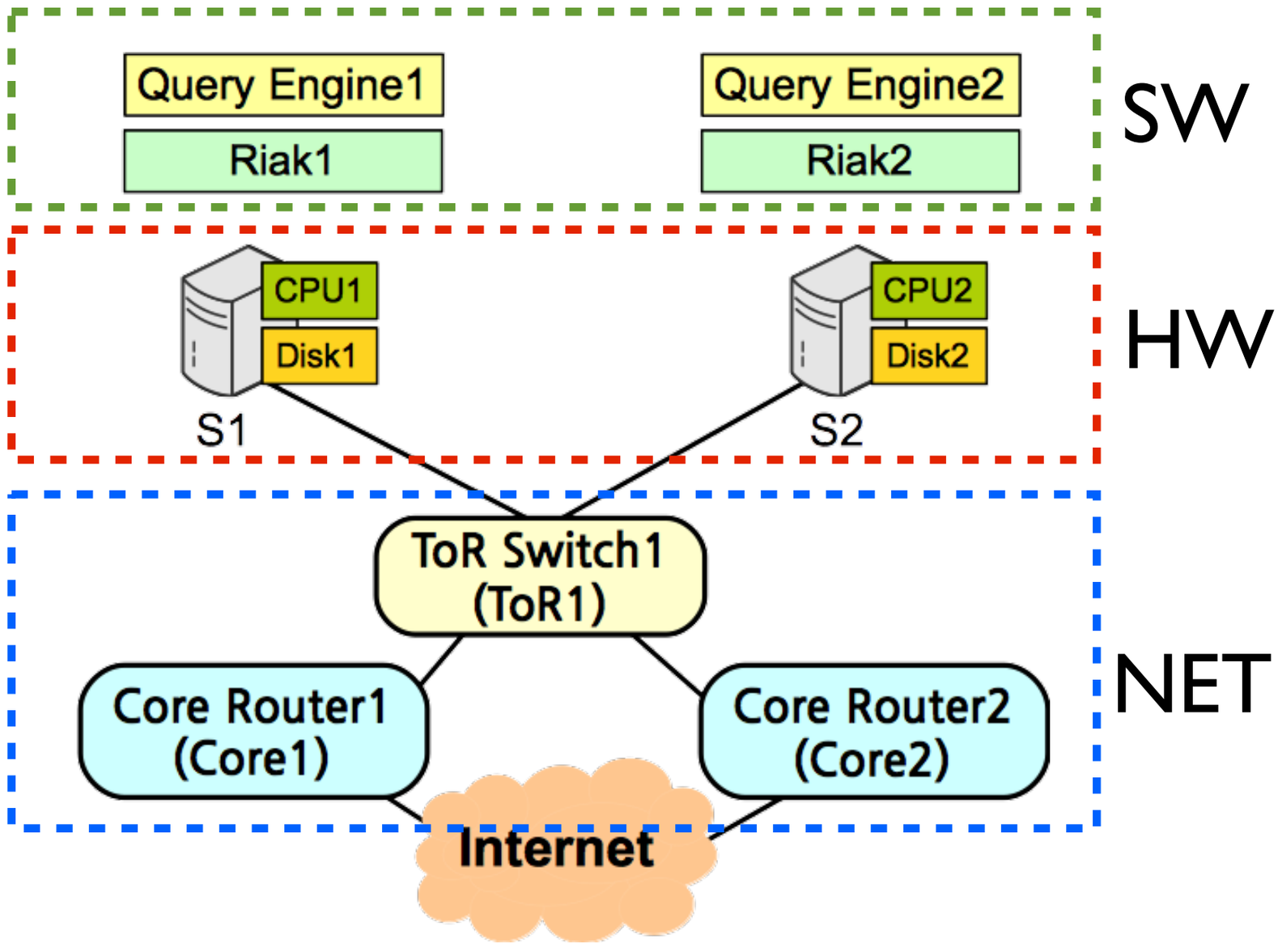
Example Redundancy



Example Redundancy



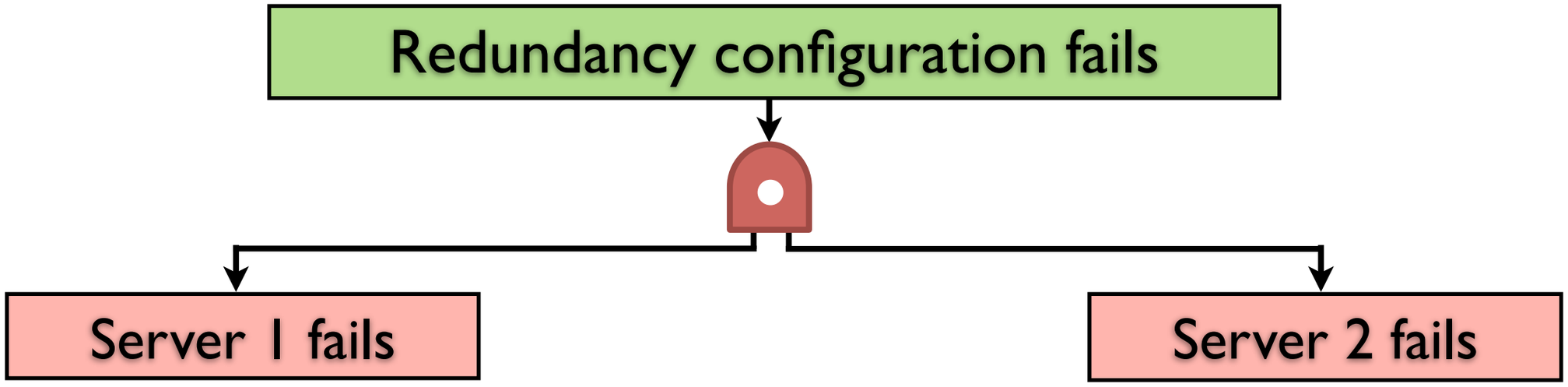
Building Fault Graph Top-to-Bottom



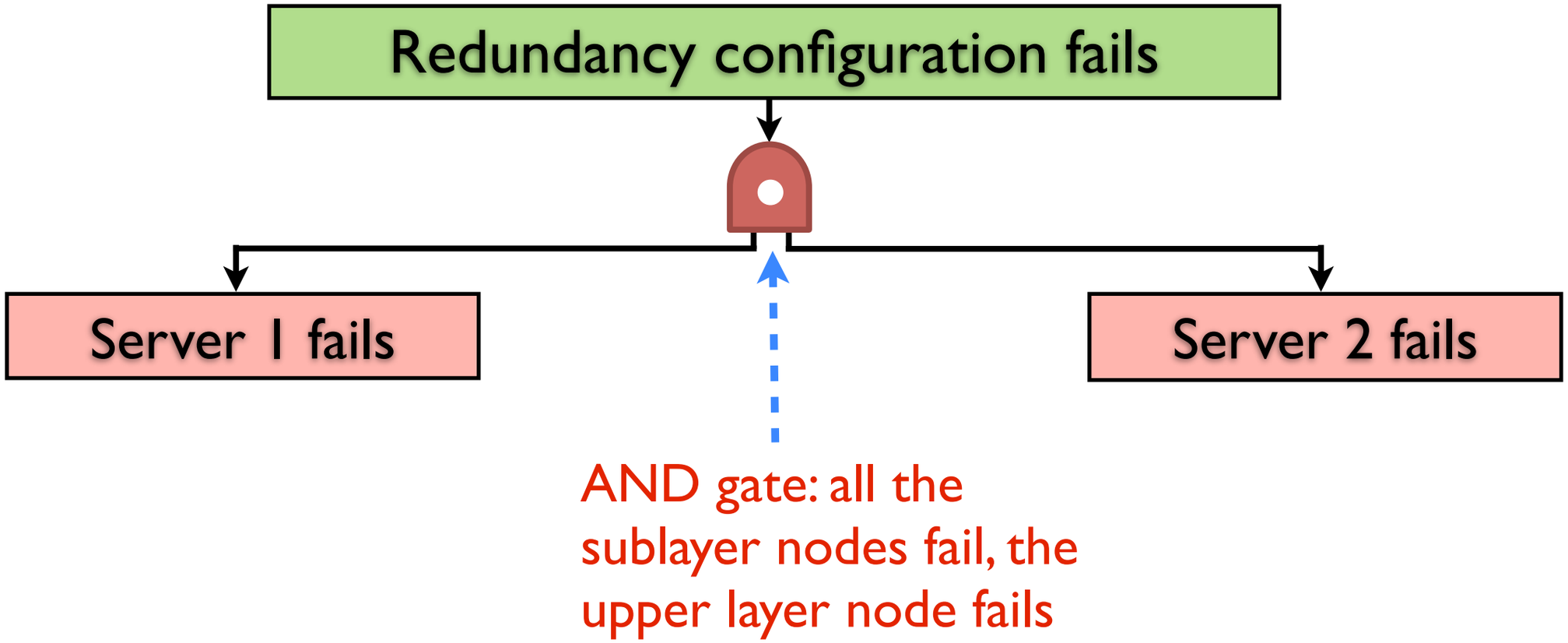
Step1: Root Node

Redundancy configuration fails

Step2: Server Nodes

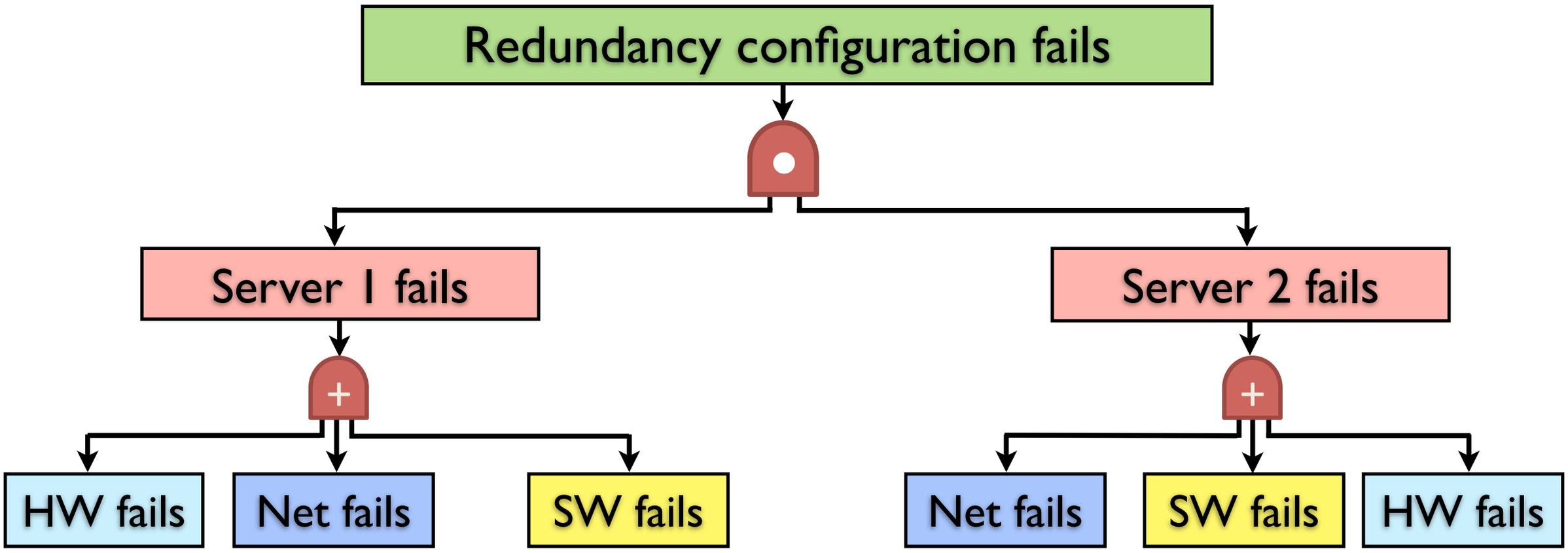


Step2: Server Nodes

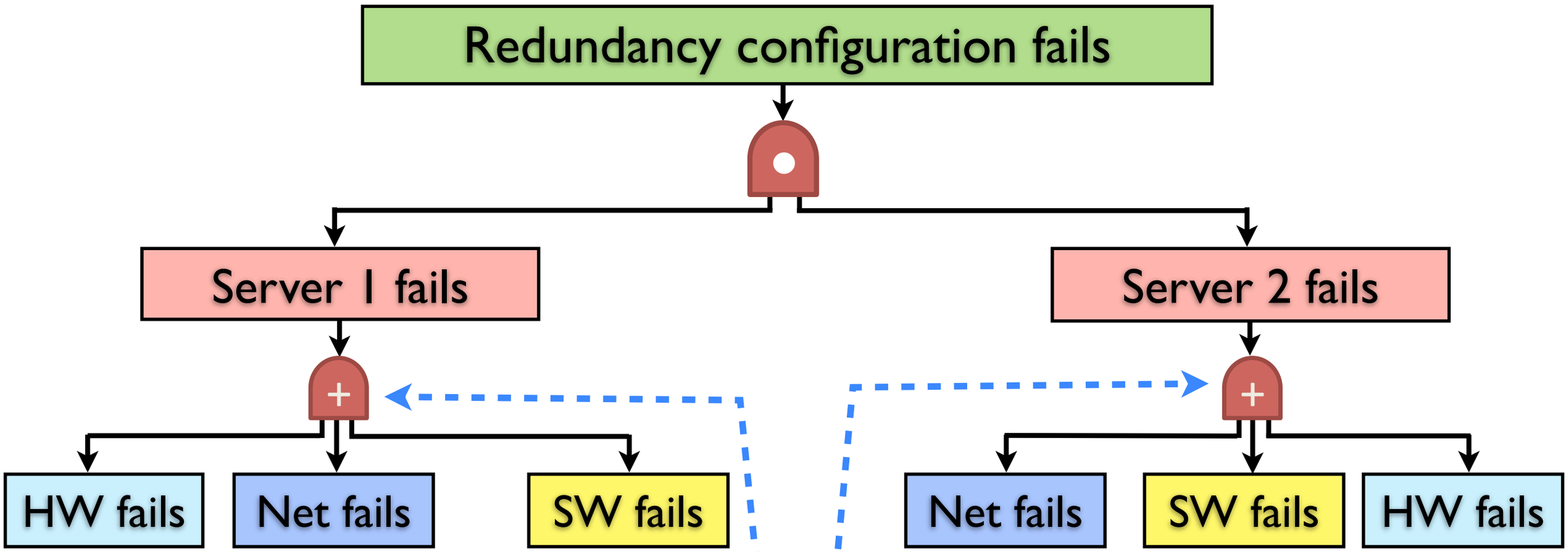


AND gate: all the sublayer nodes fail, the upper layer node fails

Step3: Dependency Nodes

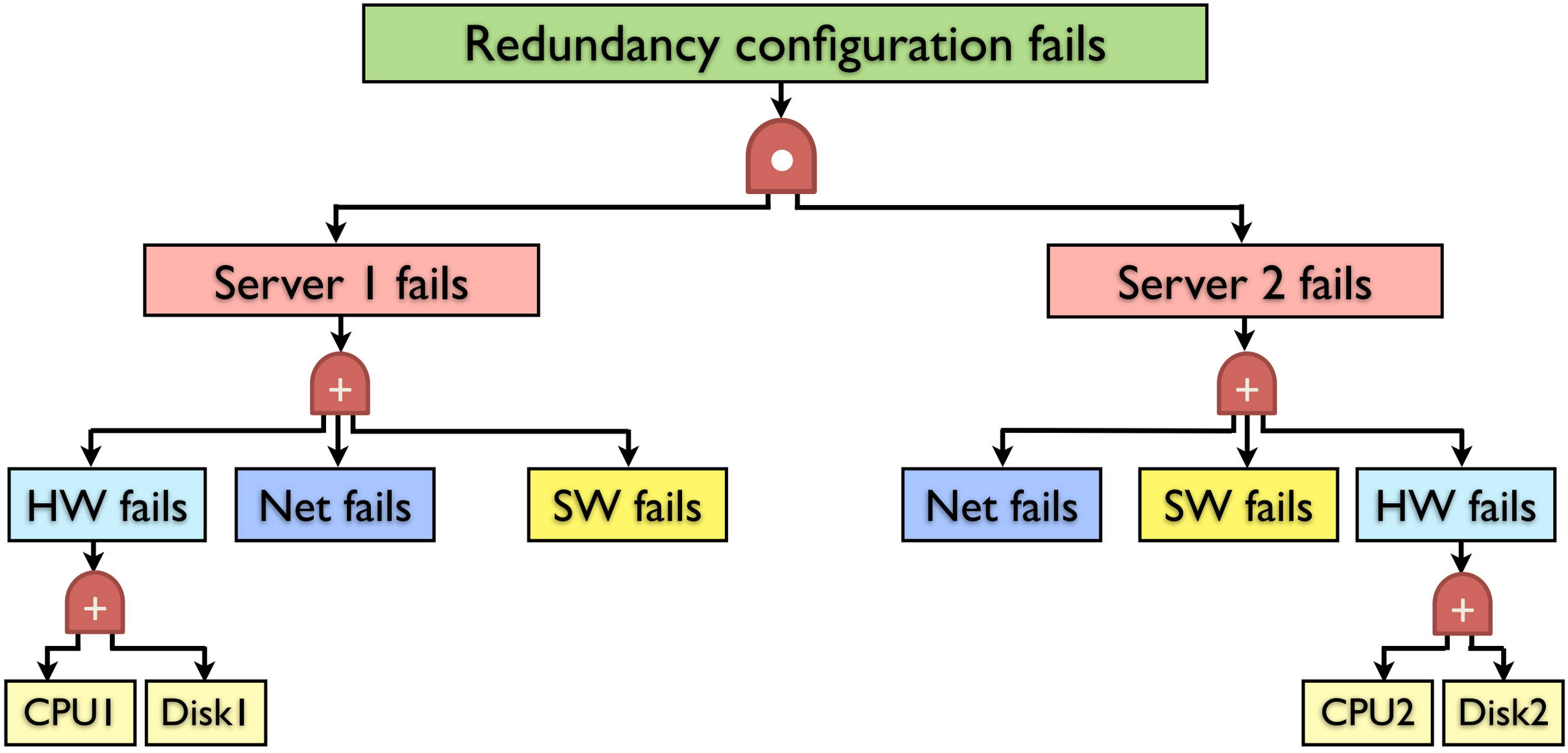


Step3: Dependency Nodes

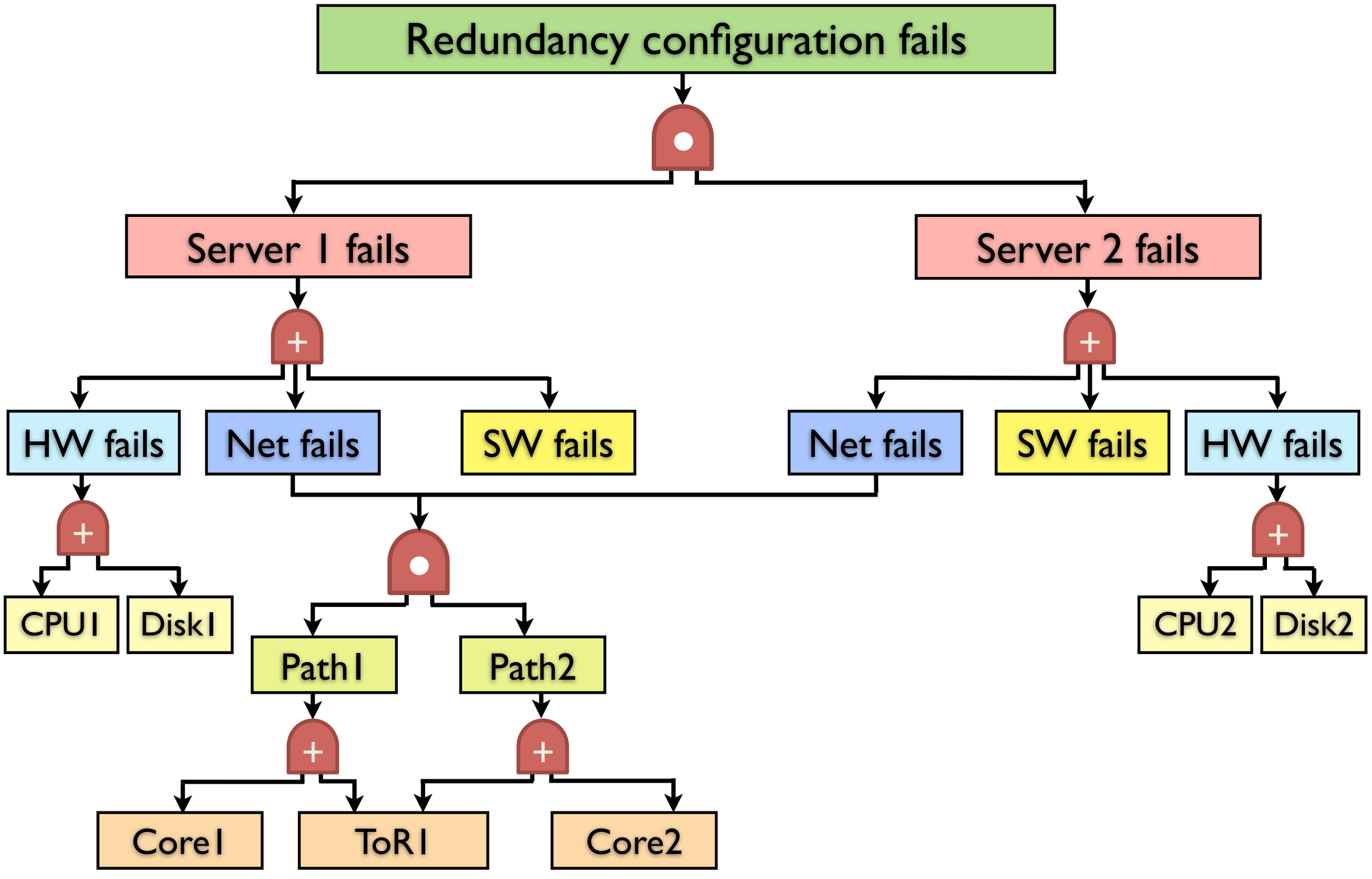


OR gate: one of the sublayer nodes fails,
the upper layer node fails

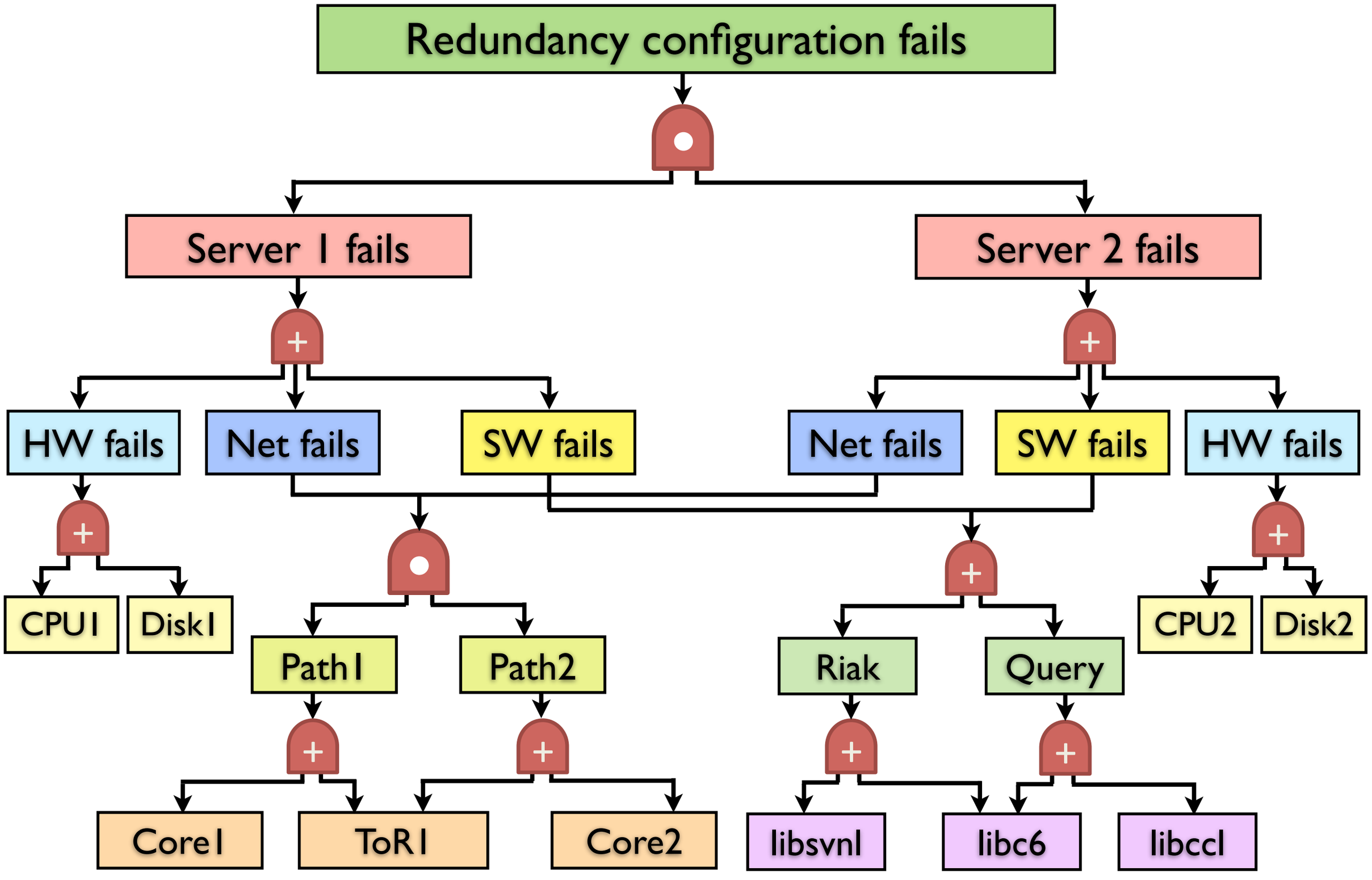
Step4: Hardware Dependency



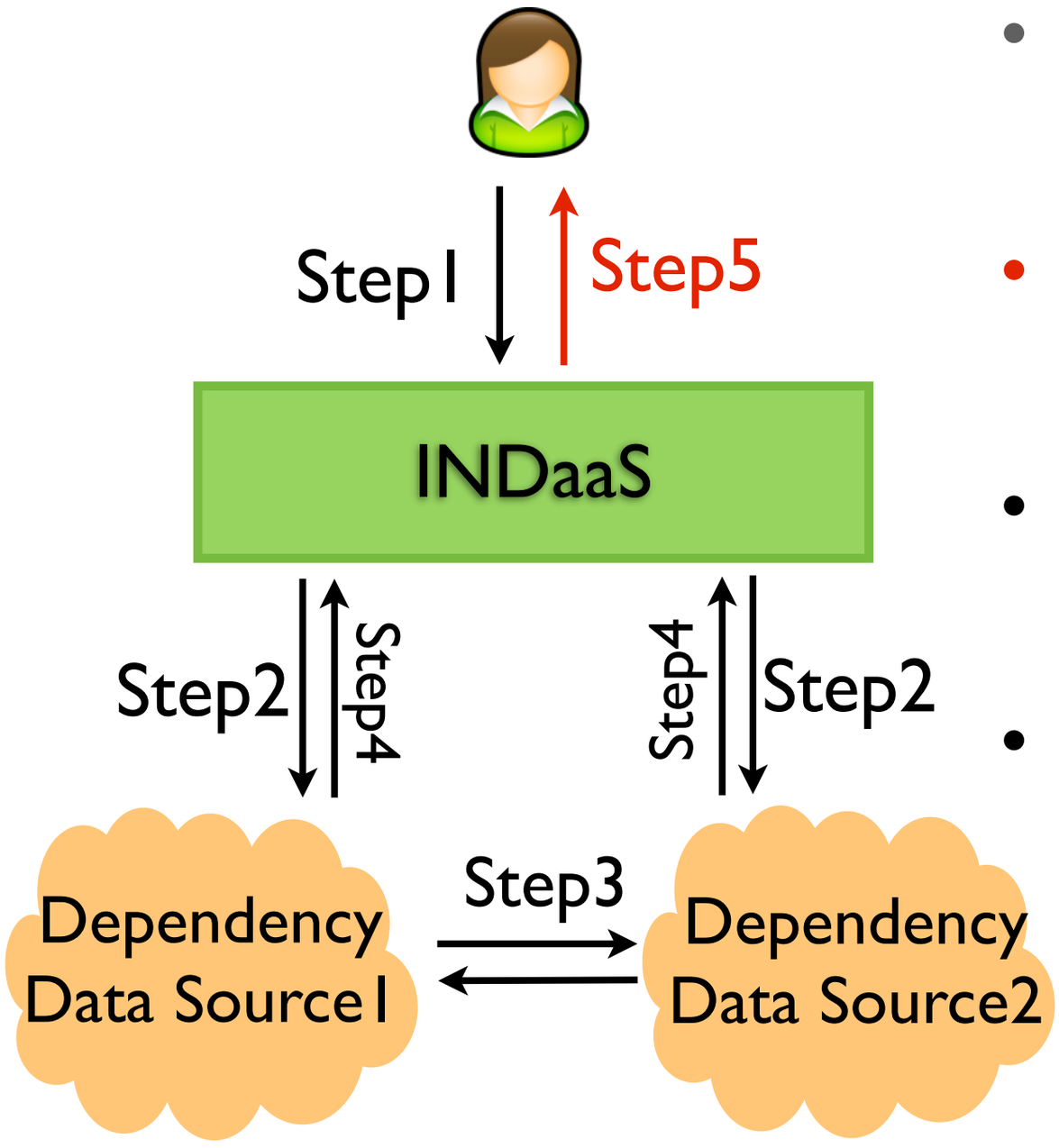
Step5: Network Dependency



Step6: Software Dependency

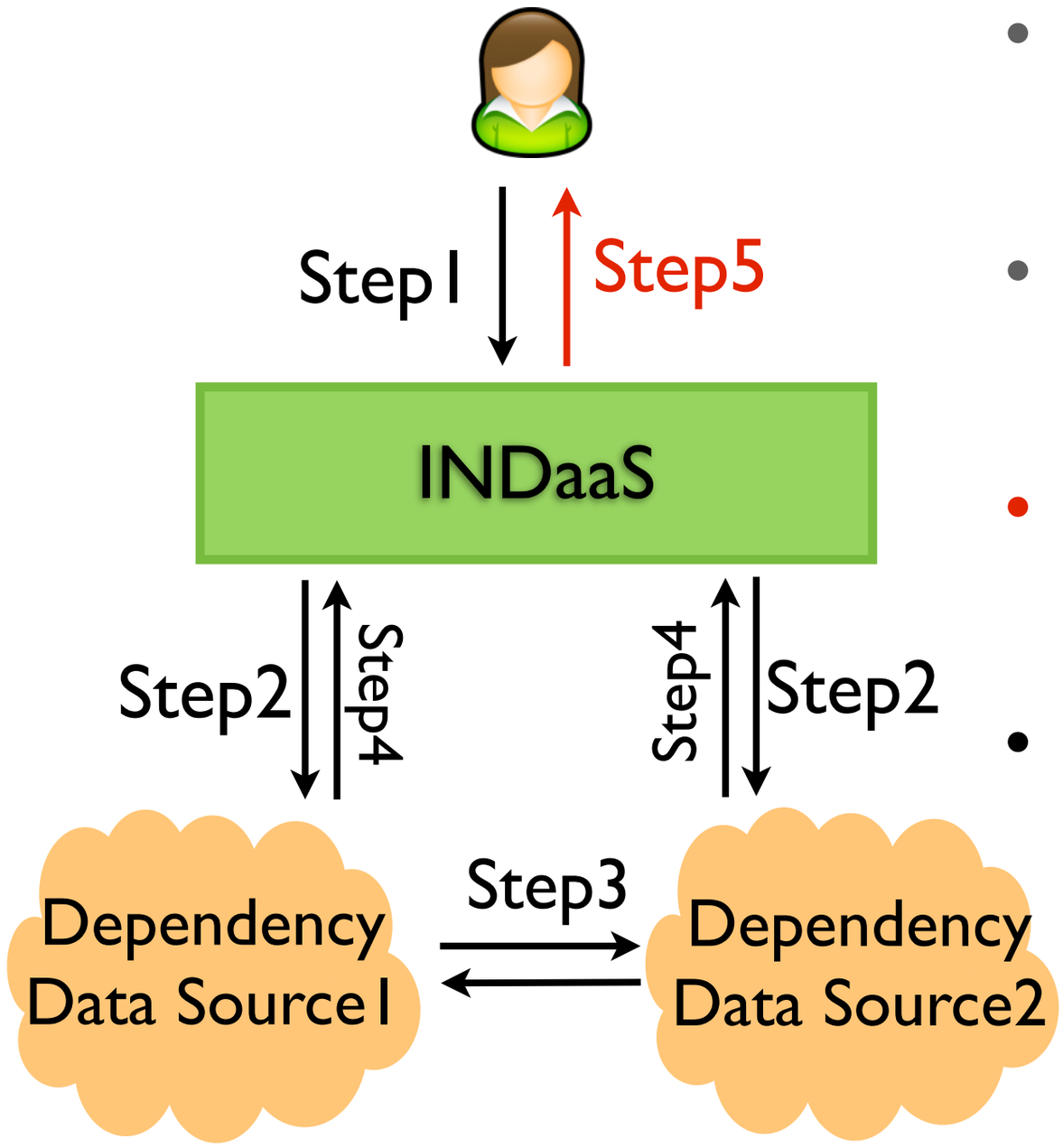


RoadMap



- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

RoadMap



- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

Efficient Auditing

- Two algorithms balancing cost and accuracy:
  - Minimal fault set algorithm
  - Failure sampling algorithm

Efficient Auditing

- Two algorithms balancing cost and accuracy:
  - Minimal fault set algorithm
  - Failure sampling algorithm

Minimal Fault Set Algorithm

- Traditional algorithm in safety engineering
  - Exponential complexity (NP-hard)

- We are the first to apply it in Cloud area:
  - Analyzing a fat tree with 30,528 with ~40 hours

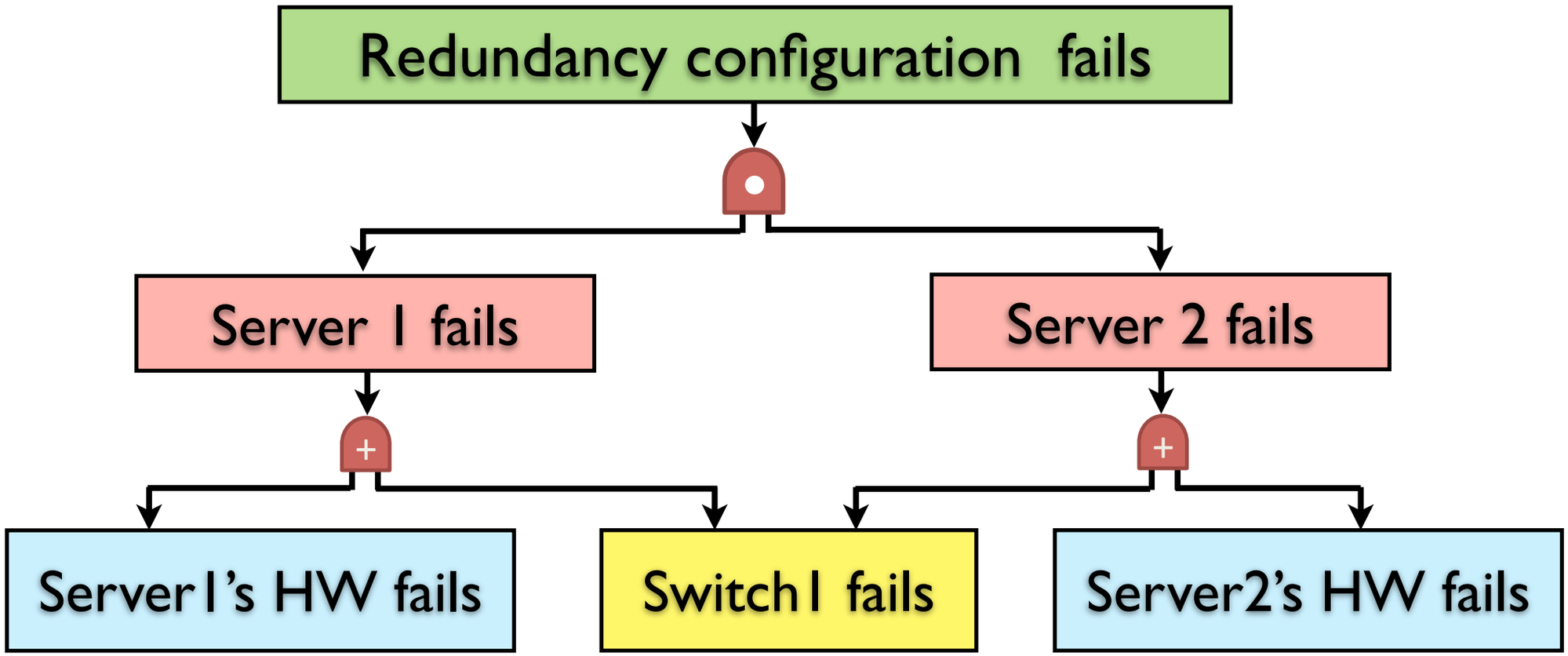
Minimal Fault Set Algorithm

- Traditional algorithm in safety engineer
  - Exponential complexity (NP-hard)

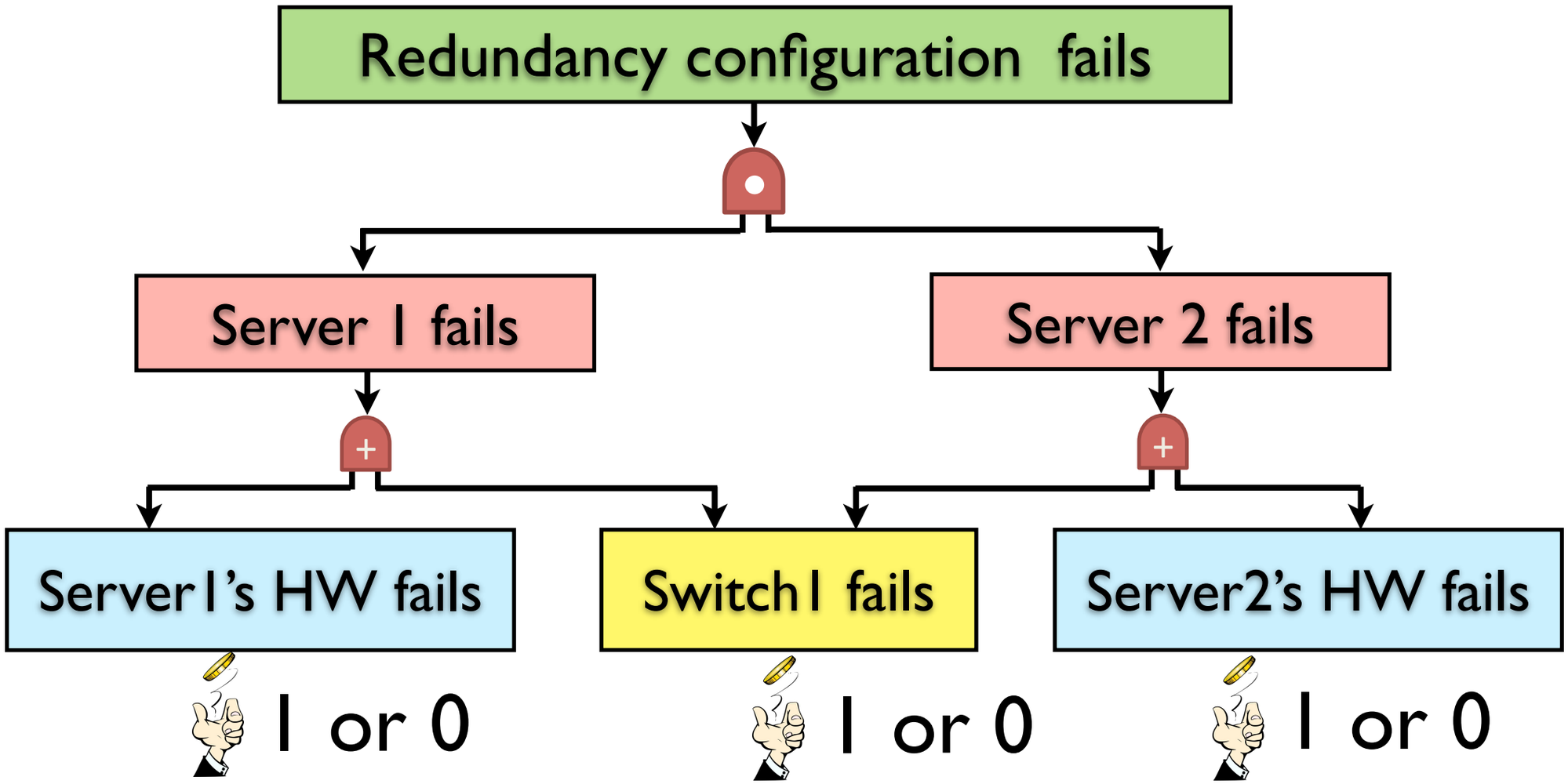
- We are the first to apply it in Cloud area:
  - Analyzing a fat tree with 30,528 with ~40 hours

- We propose efficient failure sampling algorithm.

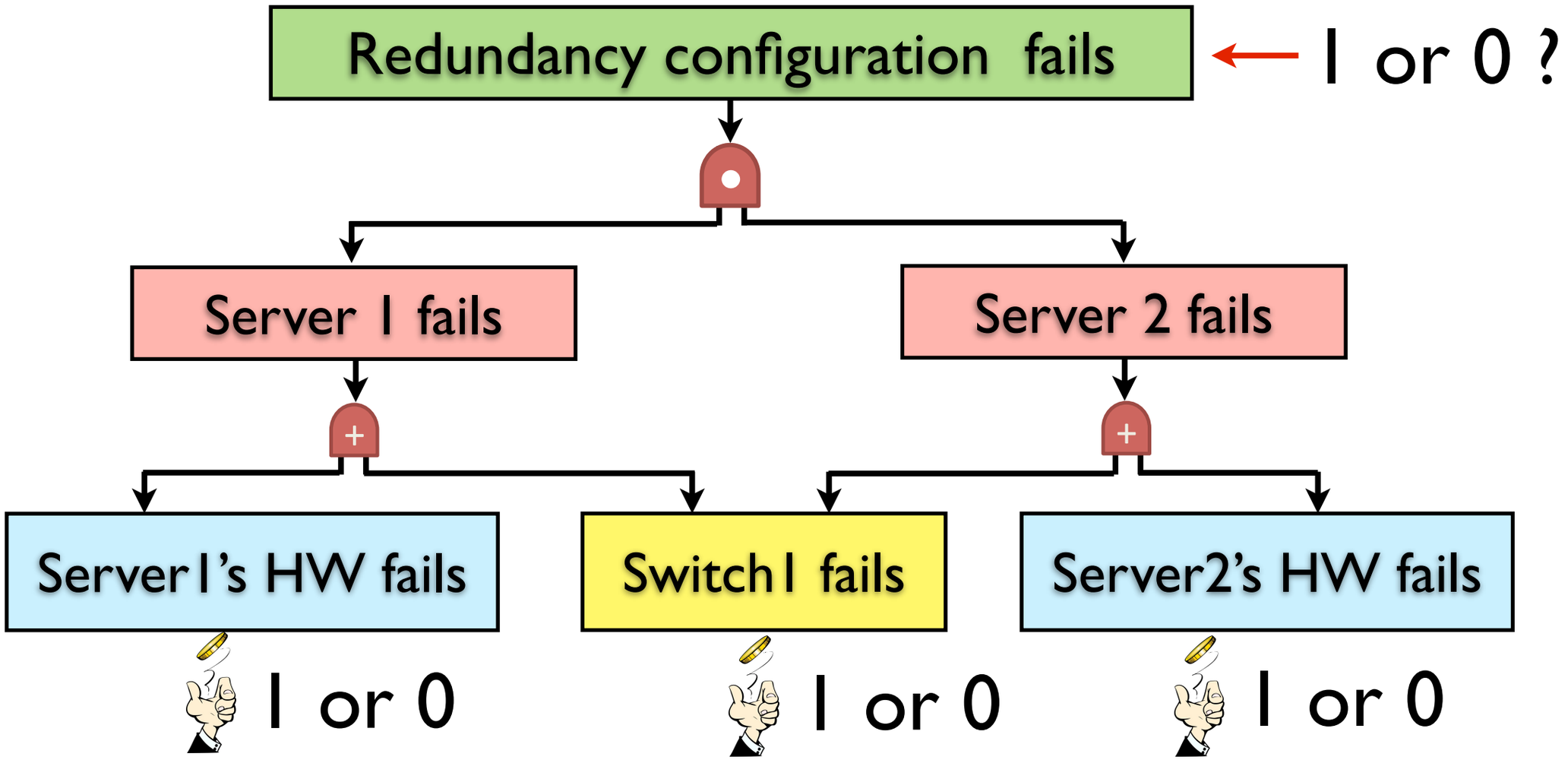
Failure Sampling Algorithm



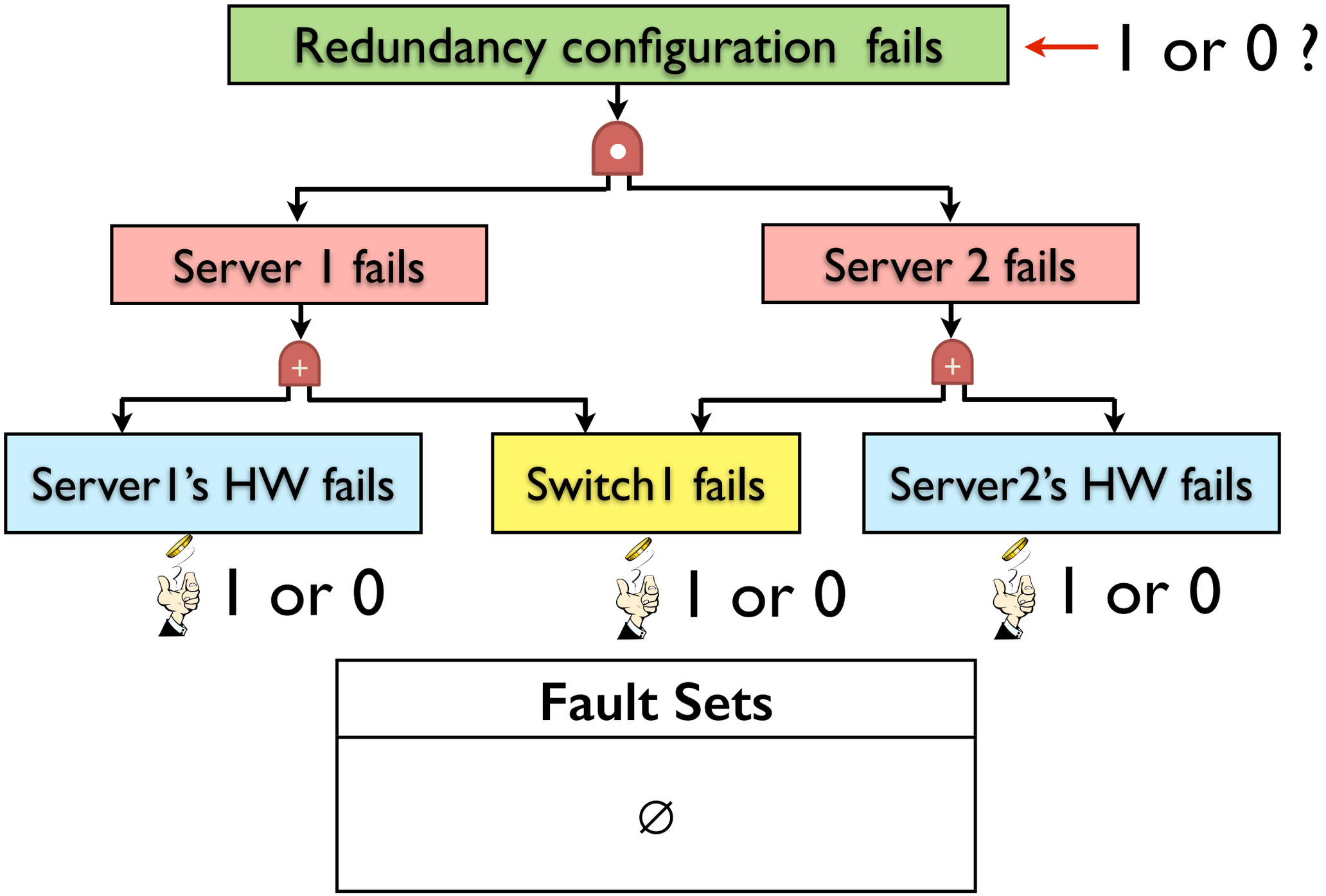
Failure Sampling Algorithm



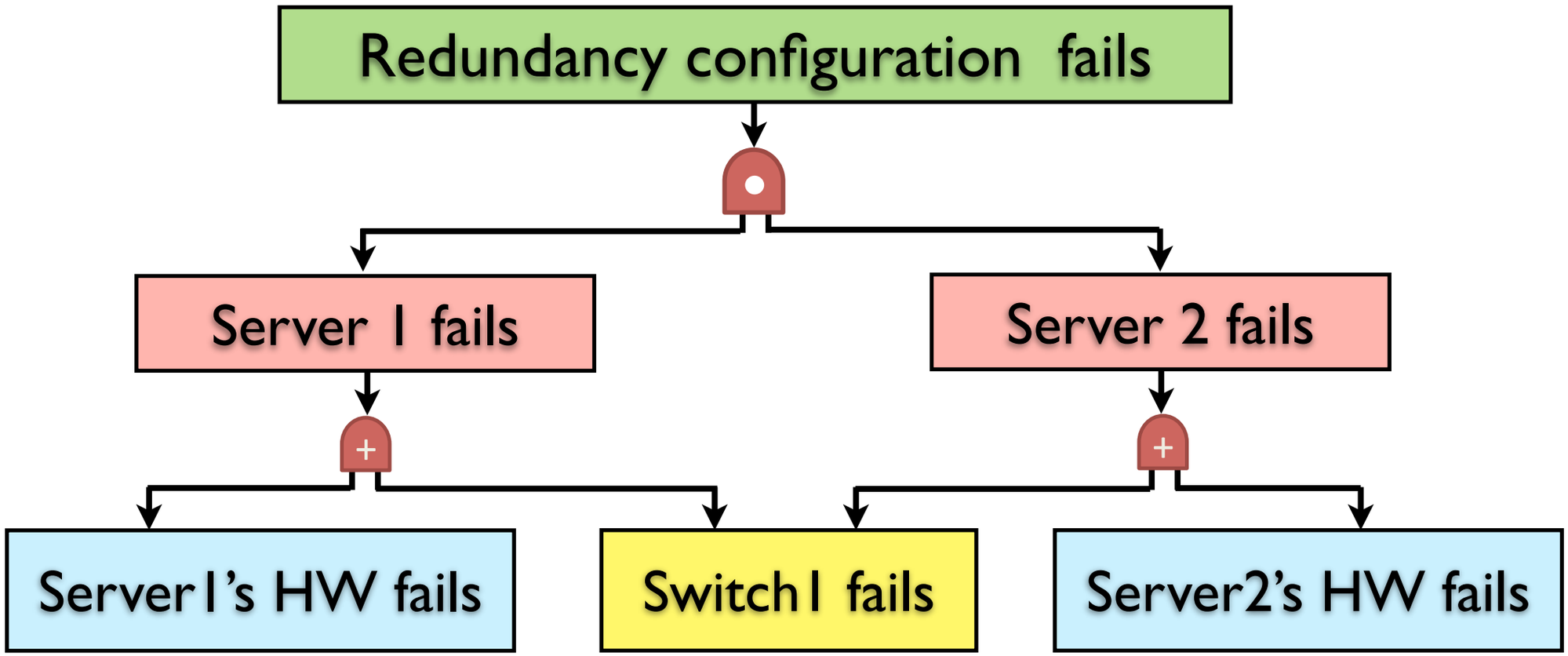
Failure Sampling Algorithm



Failure Sampling Algorithm

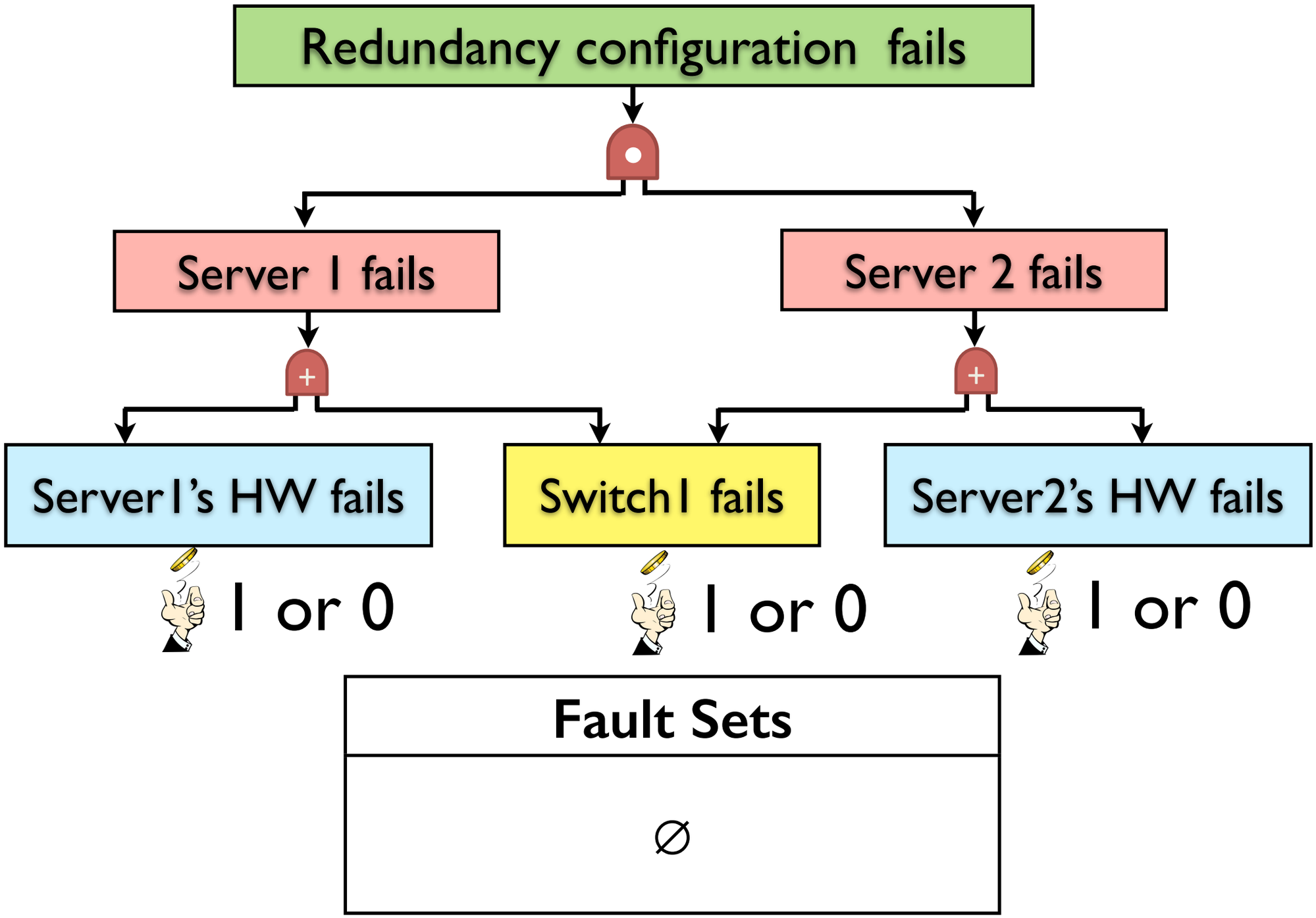


The 1st Sampling Round

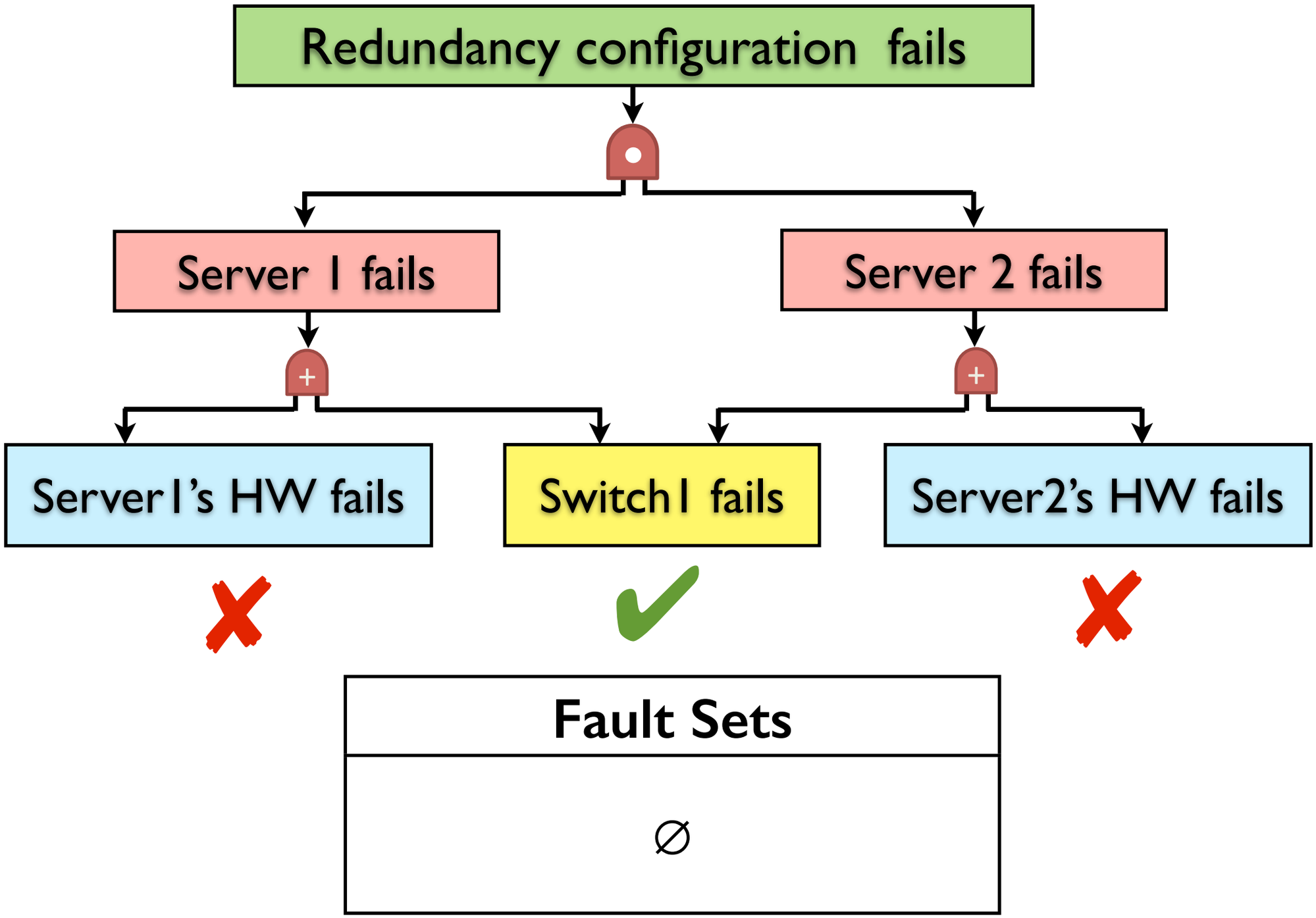


Fault Sets
∅

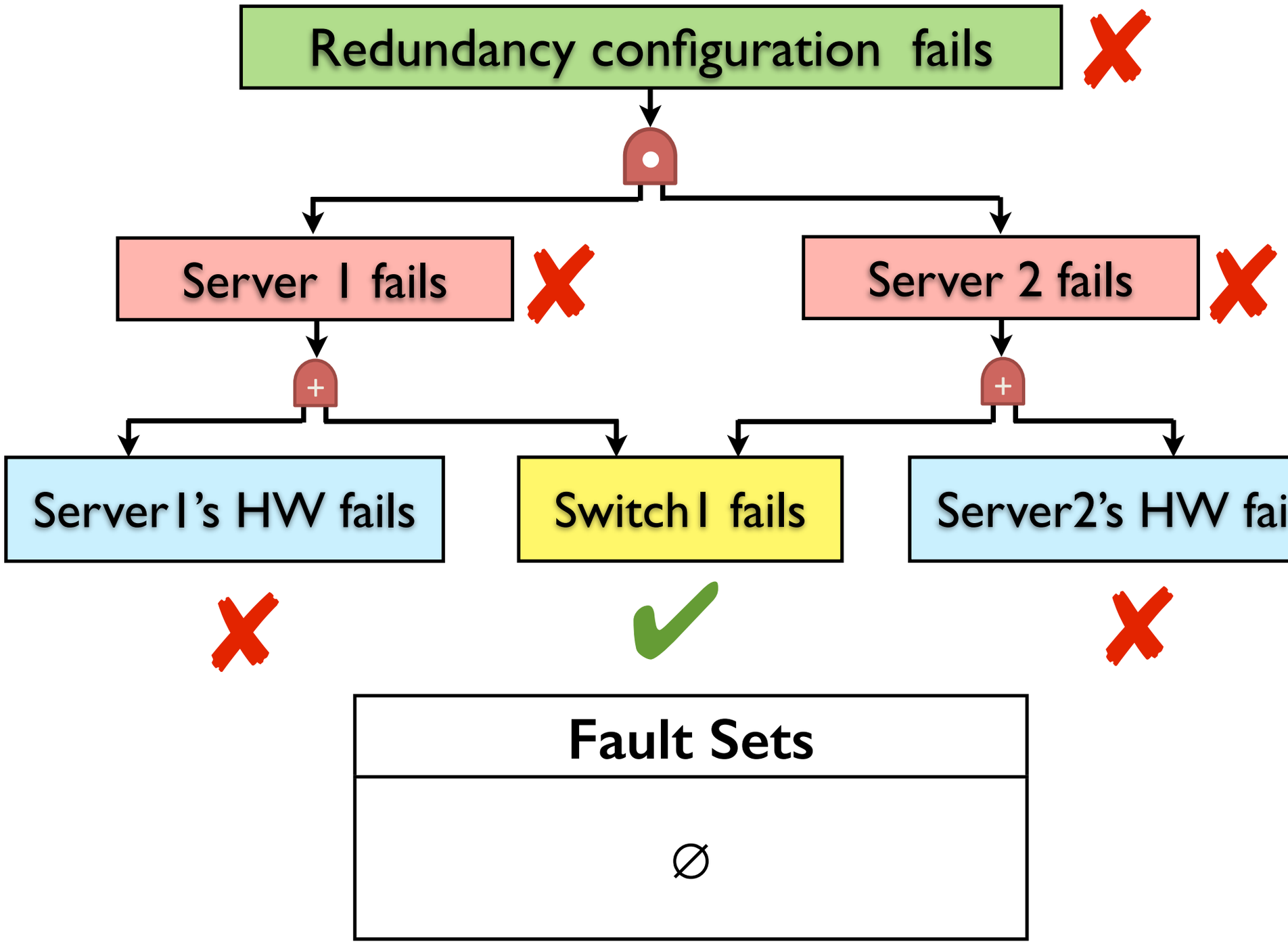
The 1st Sampling Round



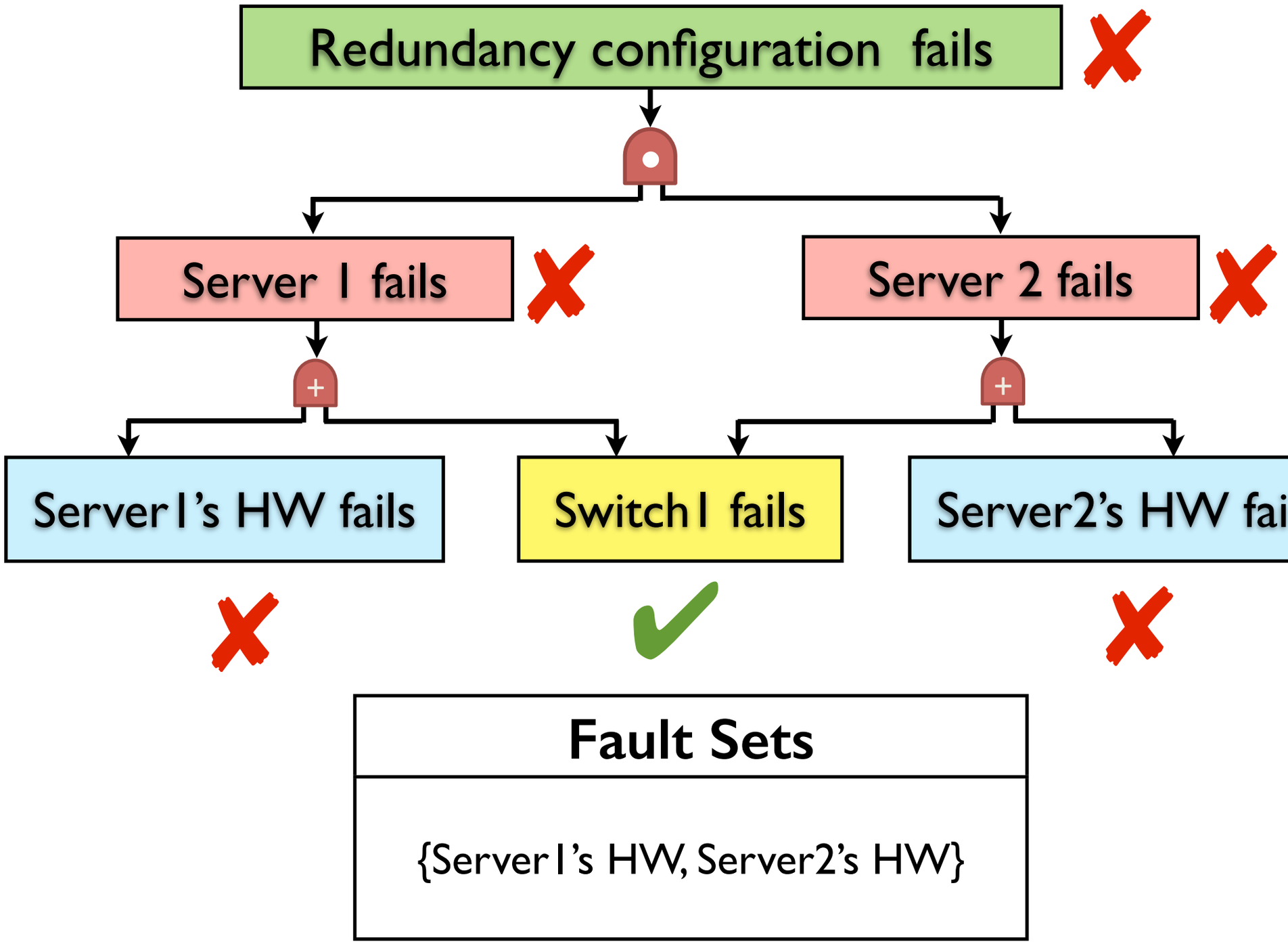
The 1st Sampling Round



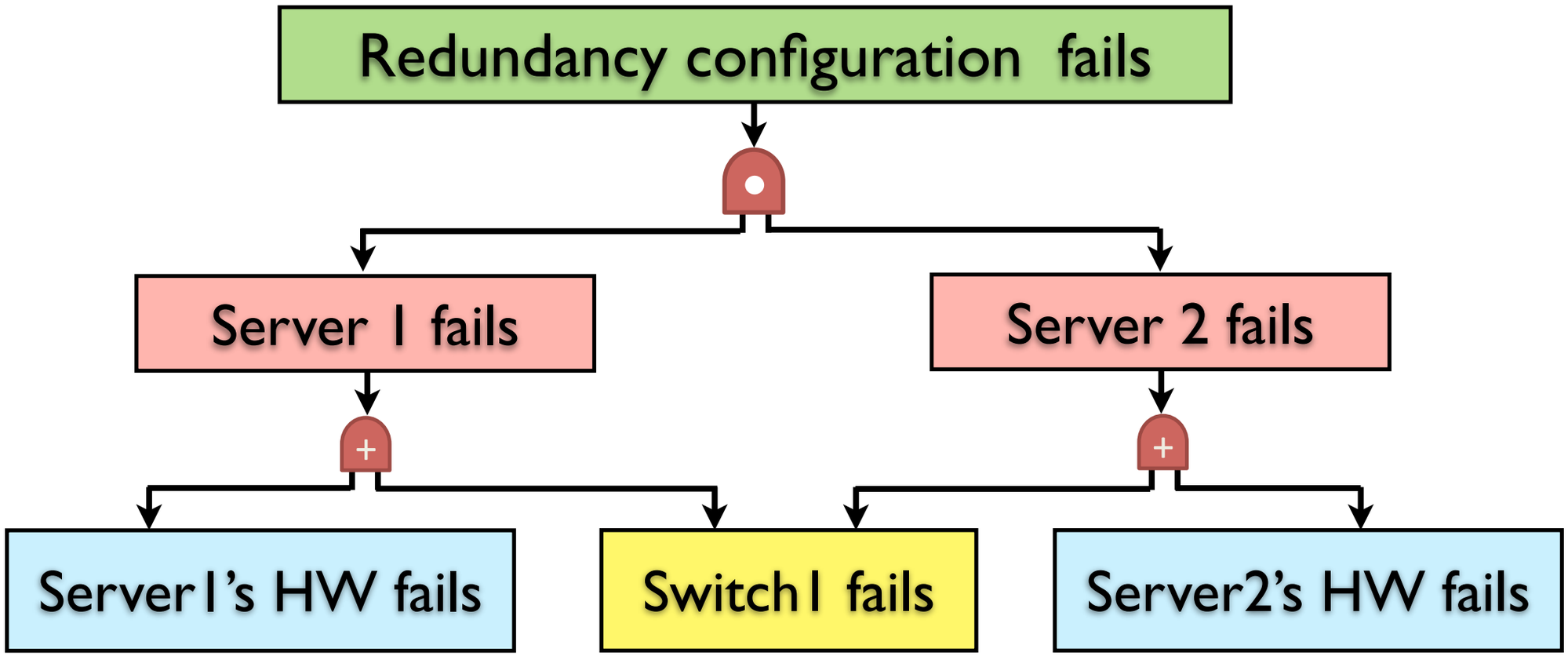
The 1st Sampling Round



The 1st Sampling Round

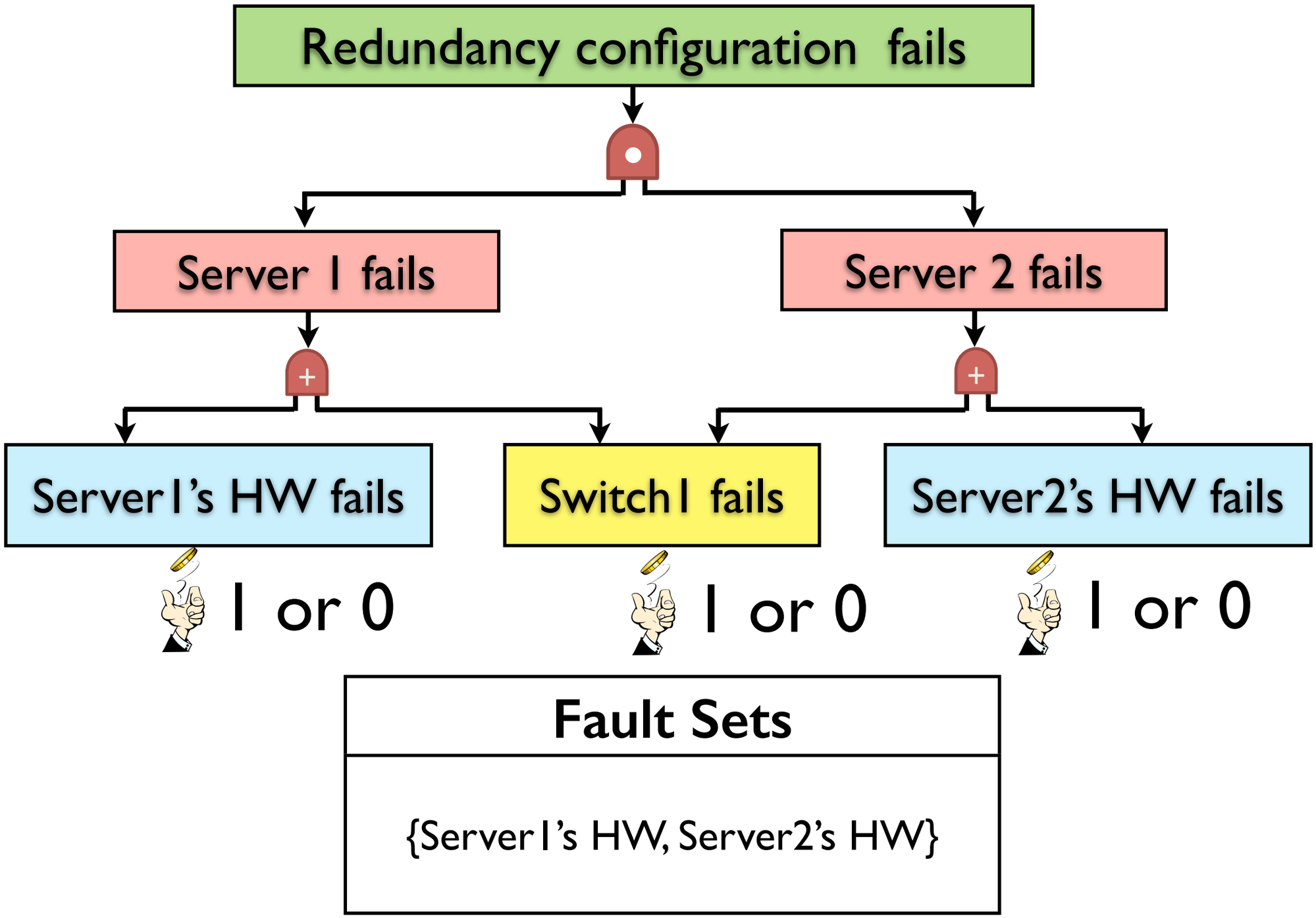


The 2nd Sampling Round

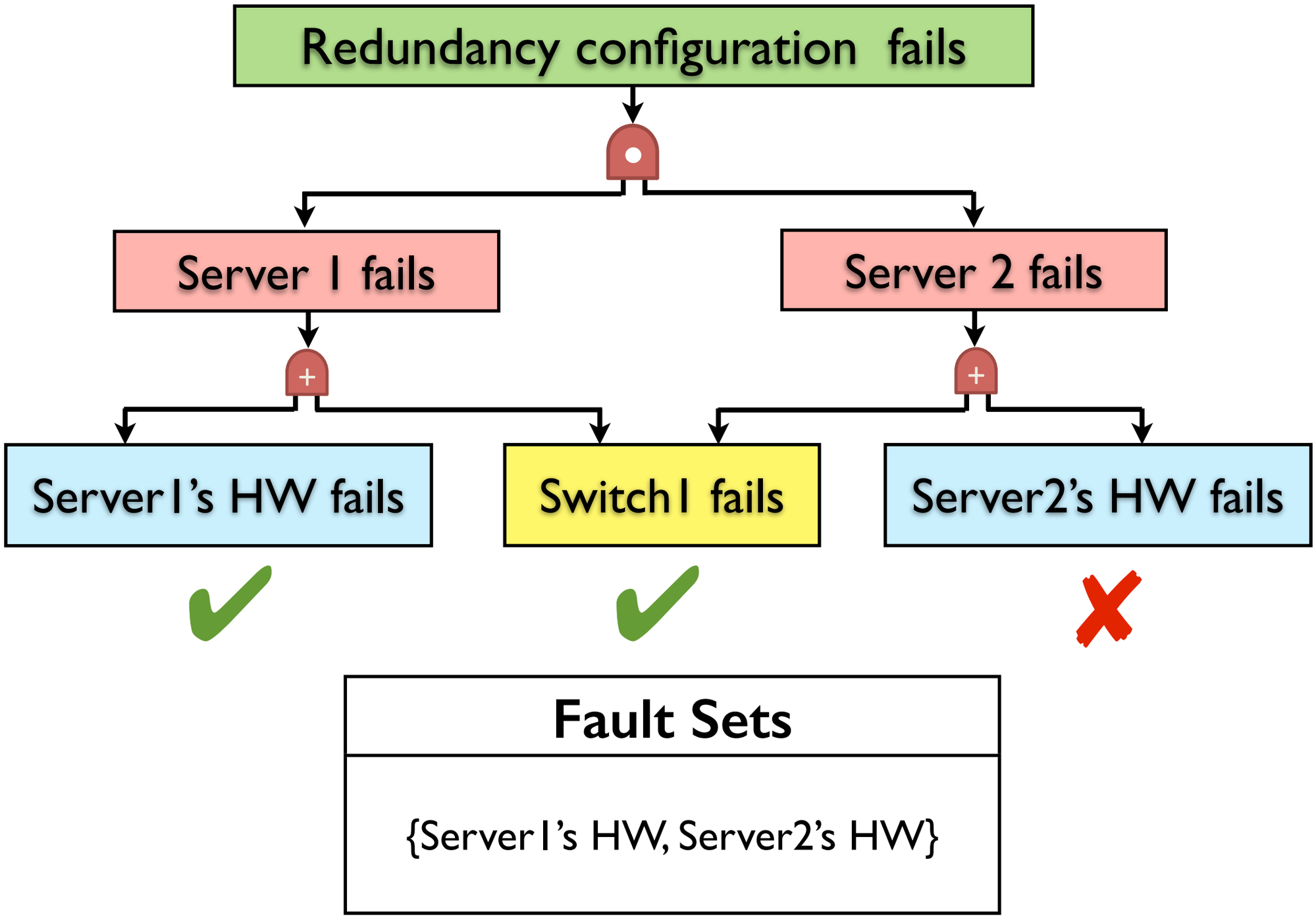


Fault Sets
{Server1's HW, Server2's HW}

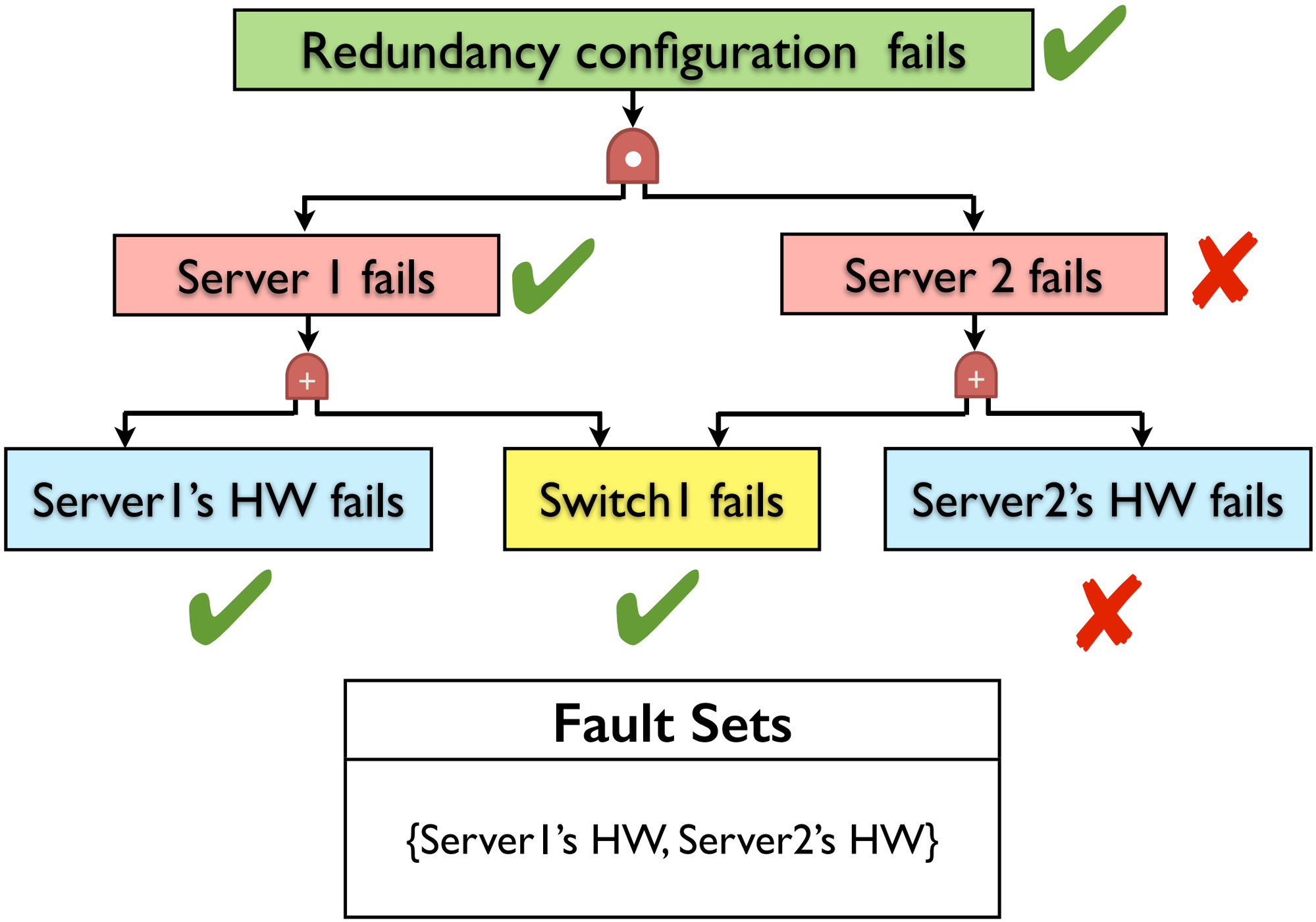
The 2nd Sampling Round



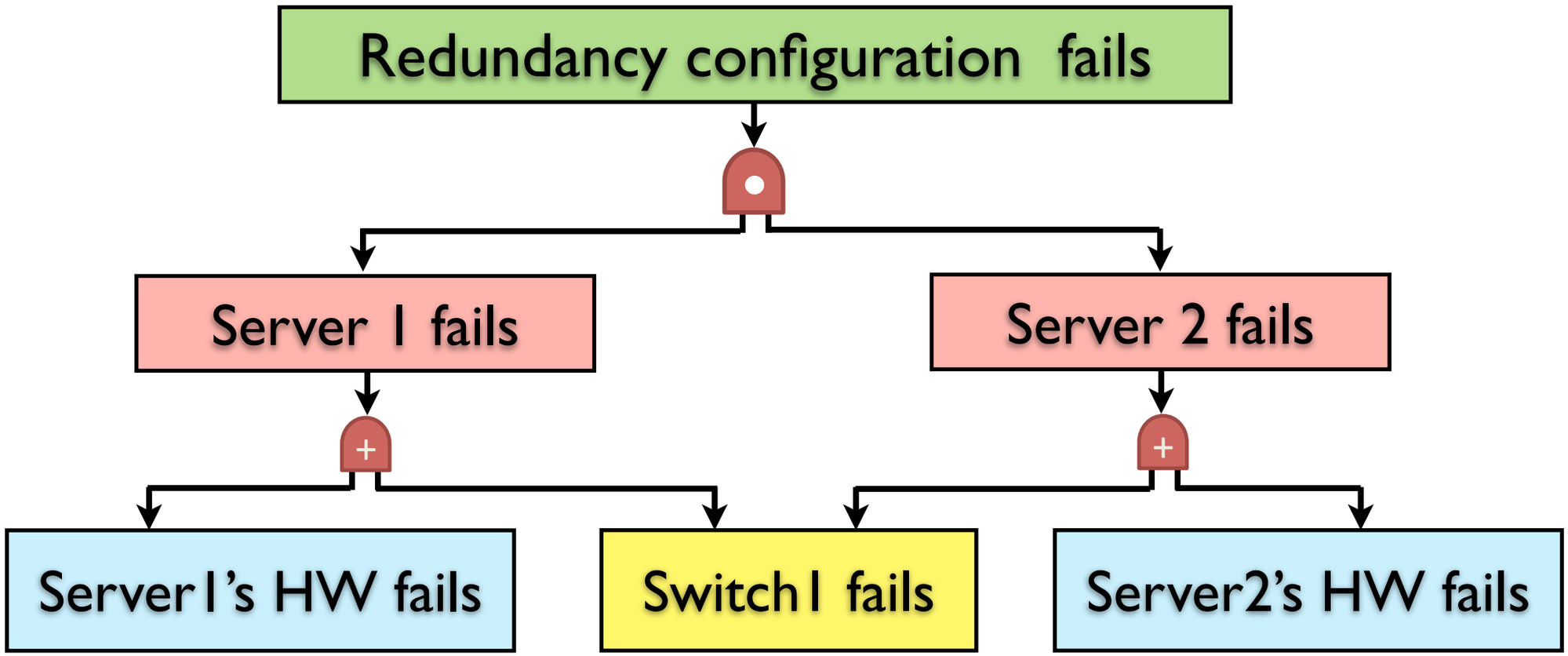
The 2nd Sampling Round



The 2nd Sampling Round

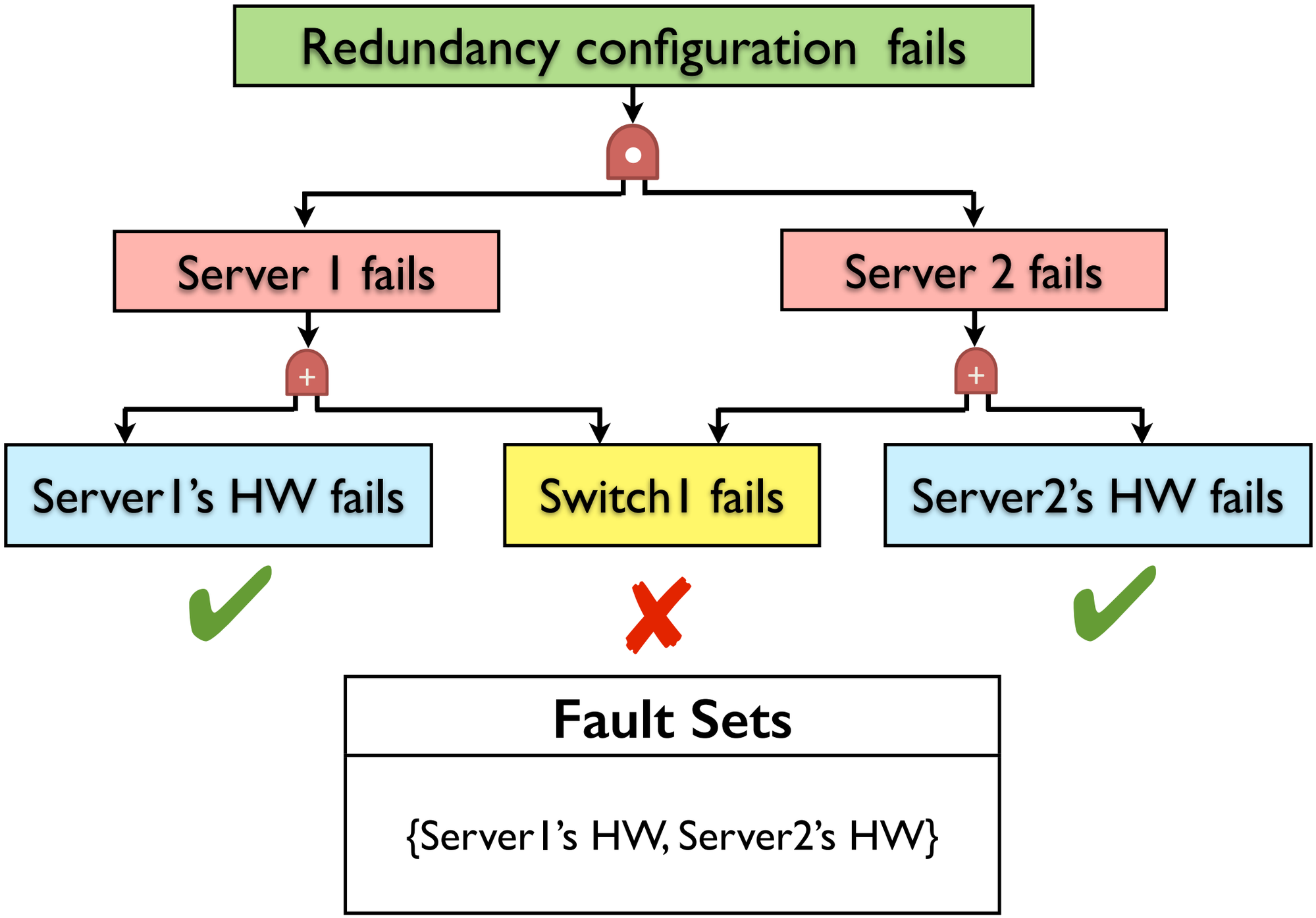


The 3rd Sampling Round

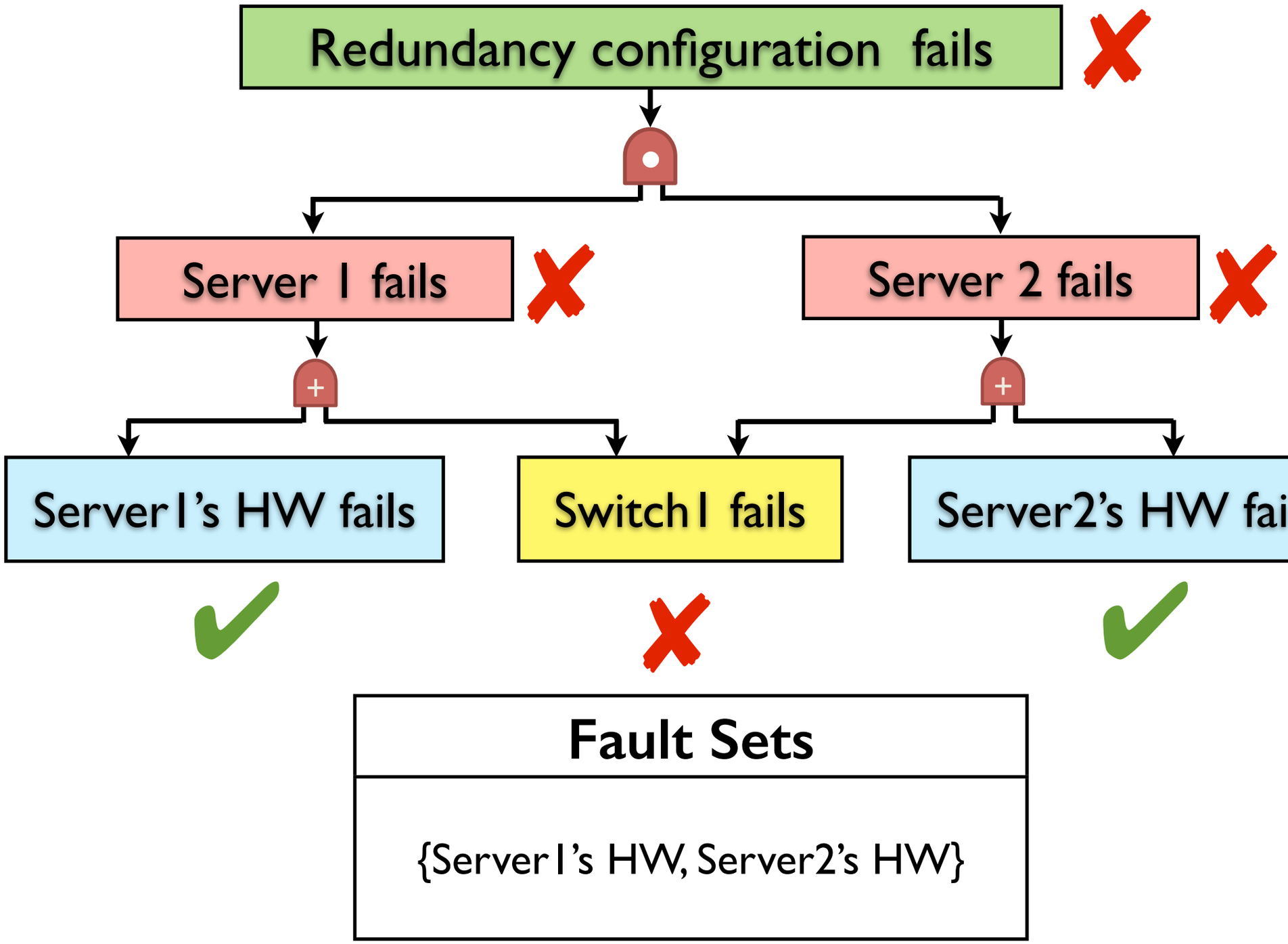


Fault Sets
{Server1's HW, Server2's HW}

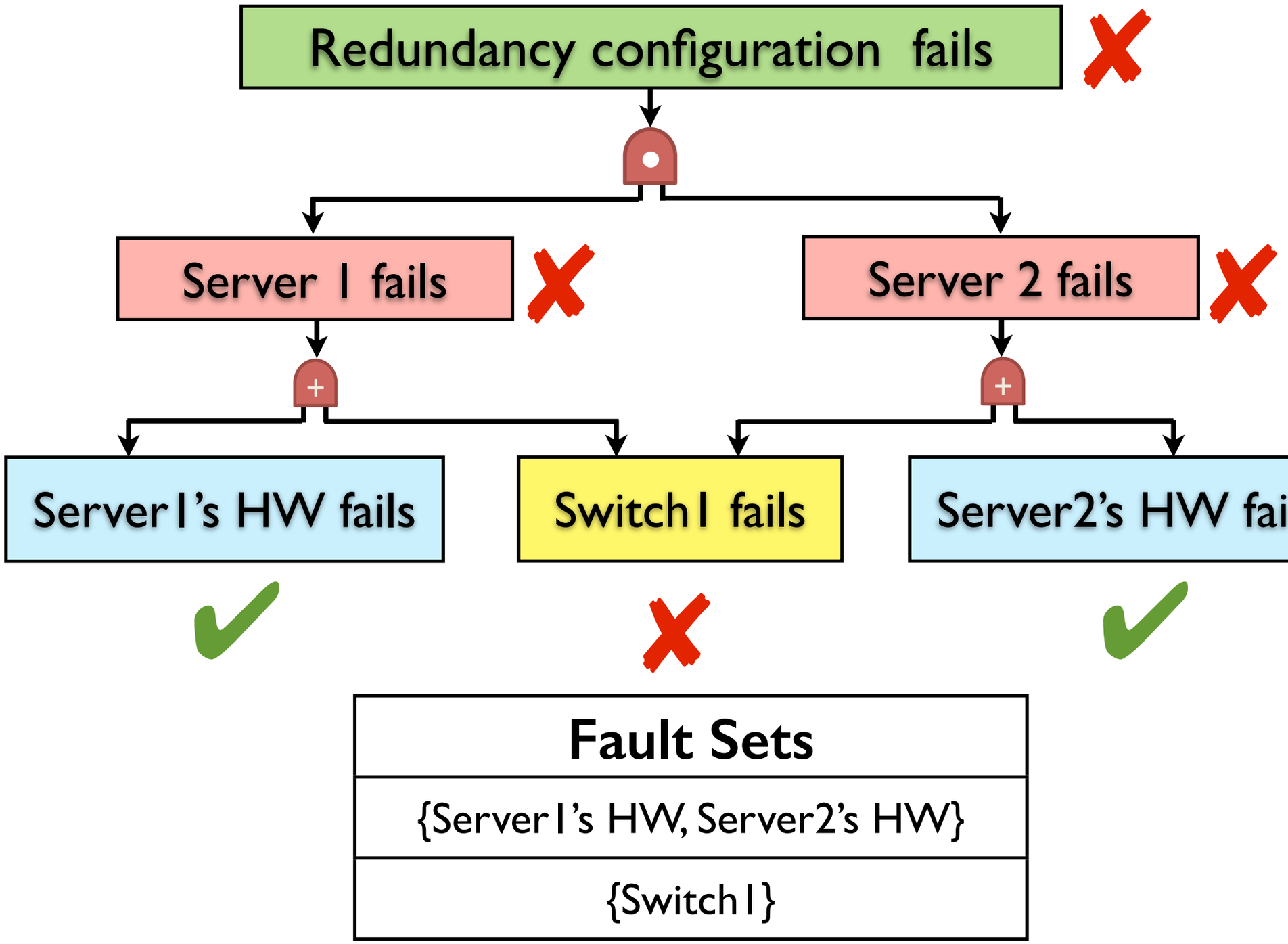
The 3rd Sampling Round



The 3rd Sampling Round



The 3rd Sampling Round



After Many (e.g., 10^7) Rounds

Fault Sets
{Server1's HW, Server2's HW}
{Switch1}
{Switch1, Server2's HW}
{Switch1}
{Switch1, Server2's HW}
… …

Size-Based Ranking

Fault Sets
{Server1's HW, Server2's HW}
{Switch1}
{Switch1, Server2's HW}
{Switch1}
{Switch1, Server2's HW}
… …

Size-Based Ranking

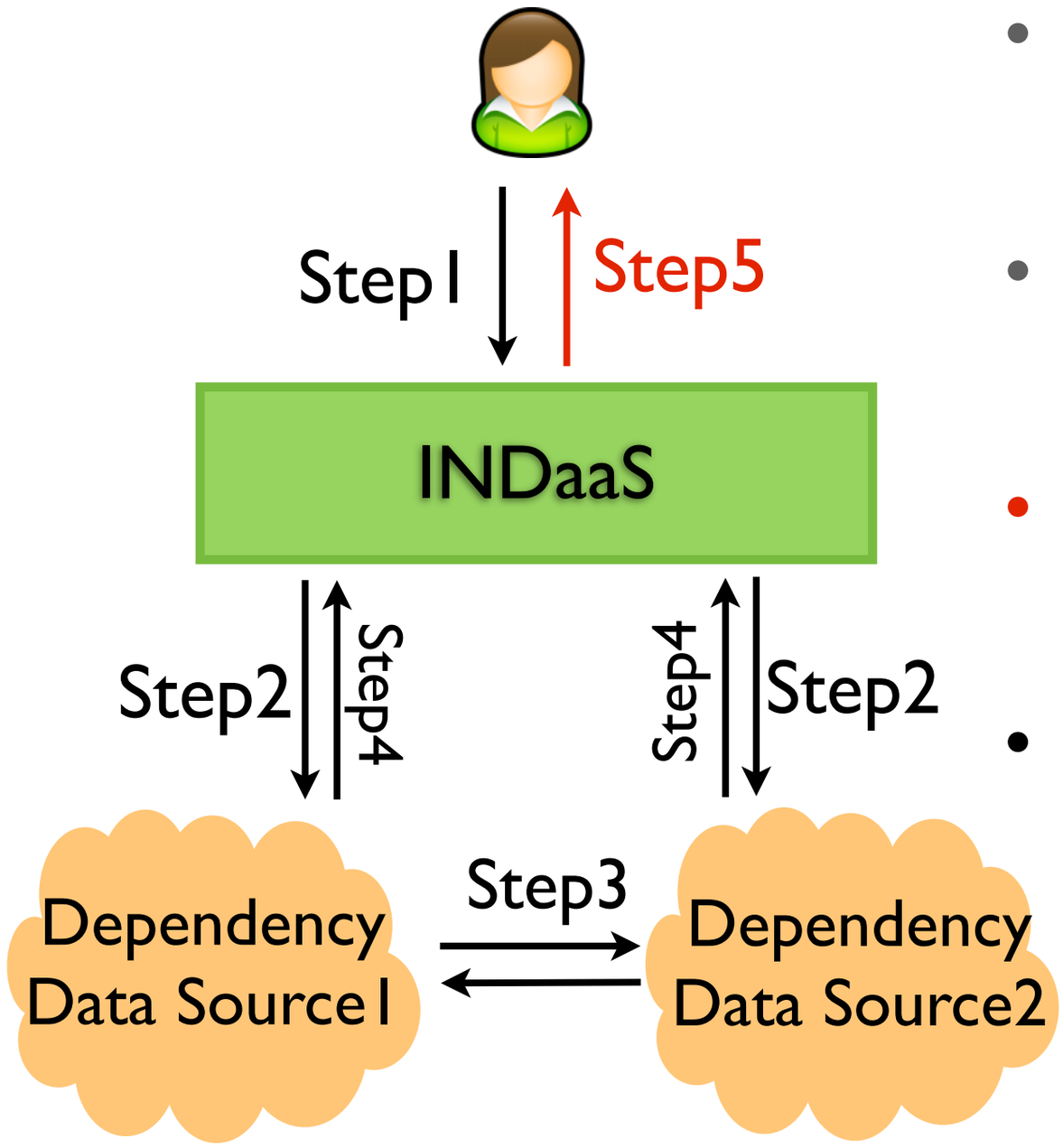
Fault Sets
{Switch1}
{Switch1}
{Switch1, Server2's HW}
{Switch1, Server2's HW}
{Server1's HW, Server2's HW}
… …

Independence Evaluation

- Multiple equations for option:
  - summation of sizes
  - weighted average of sizes

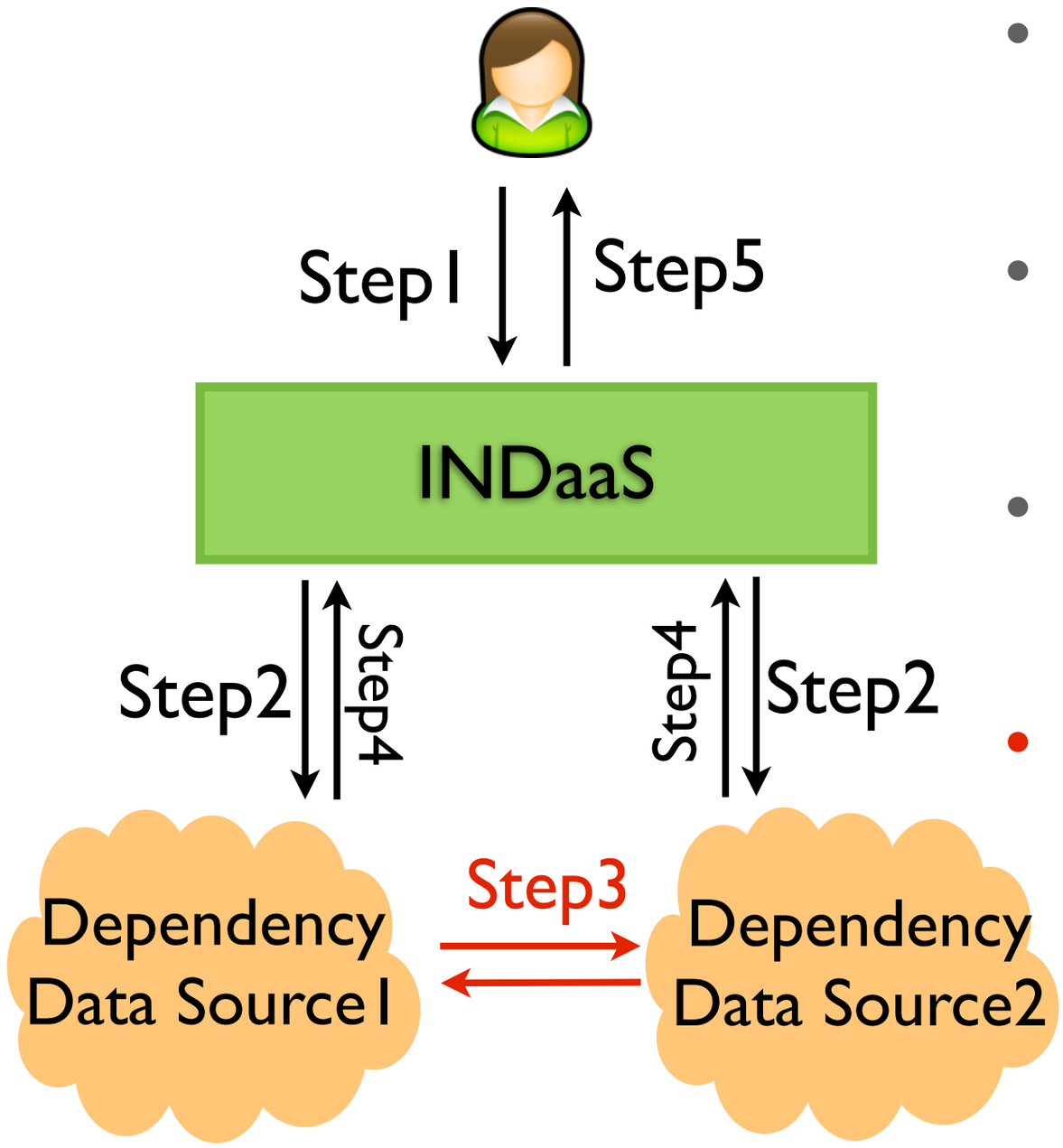
Fault Sets
{Switch1}
{Switch1, Server2's HW}
{Switch1, Server2's HW}
{Server1's HW, Server2's HW}
… …

RoadMap



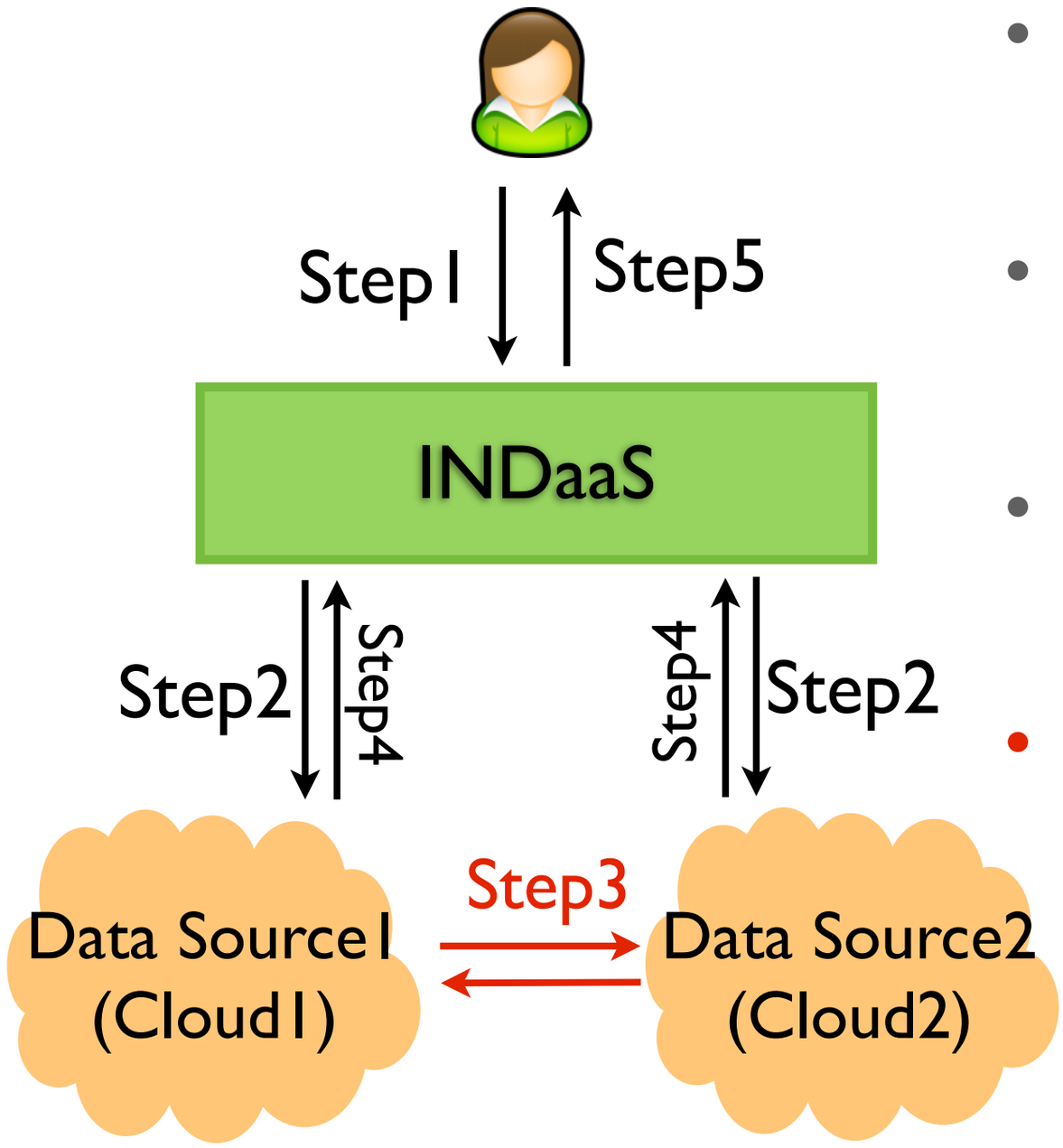
- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

RoadMap



- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

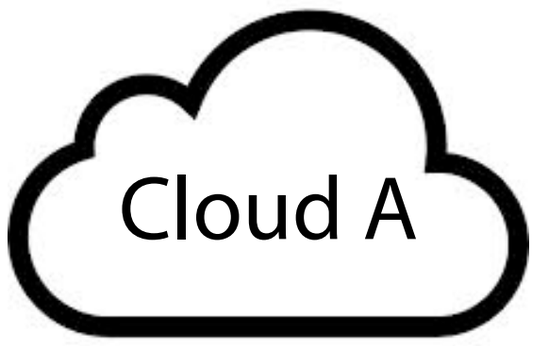
RoadMap

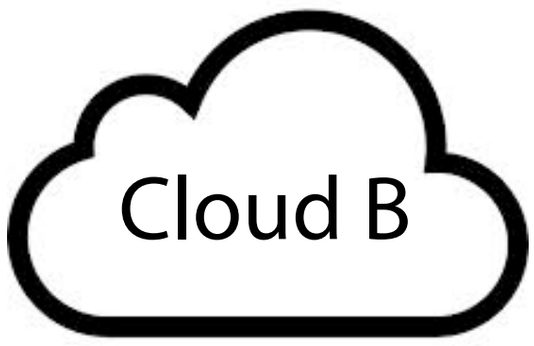


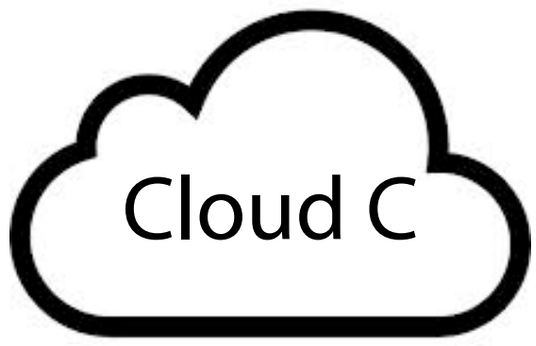
- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

Service Provider

INDaaS Agent

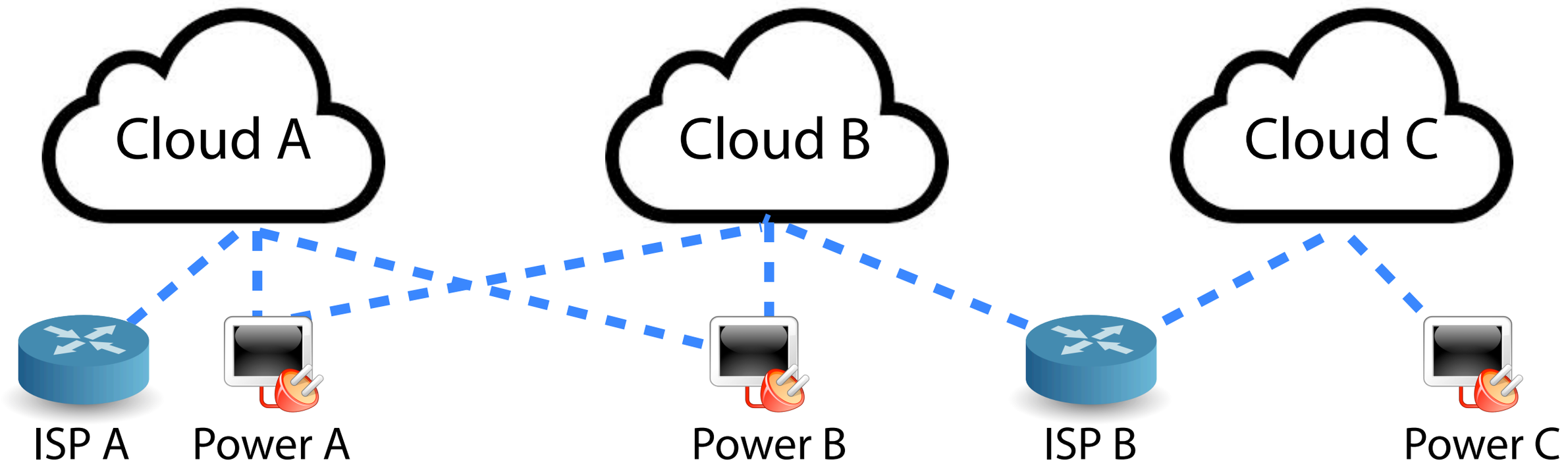






Service Provider

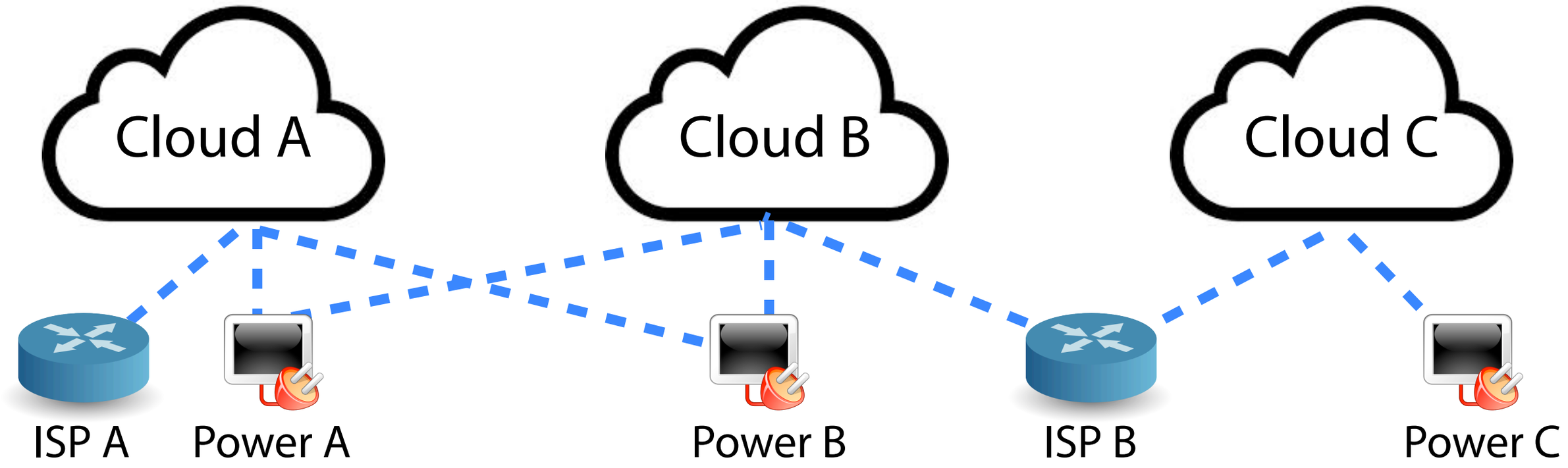
INDaaS Agent



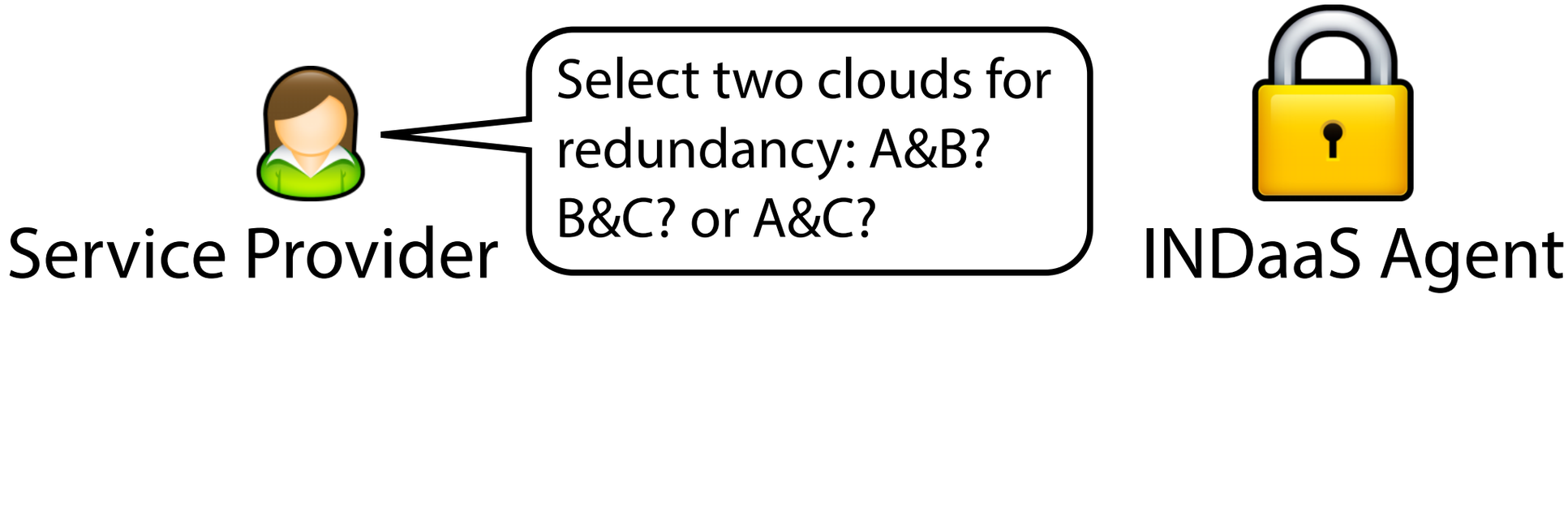
Select two clouds for redundancy: A&B? B&C? or A&C?

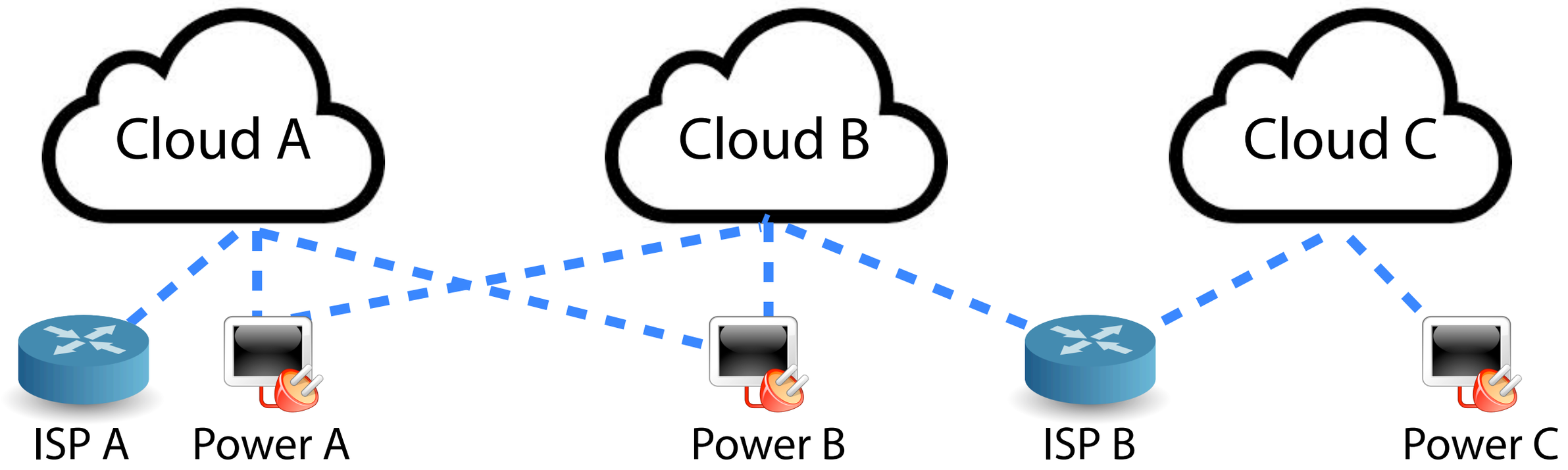
Service Provider

INDaaS Agent

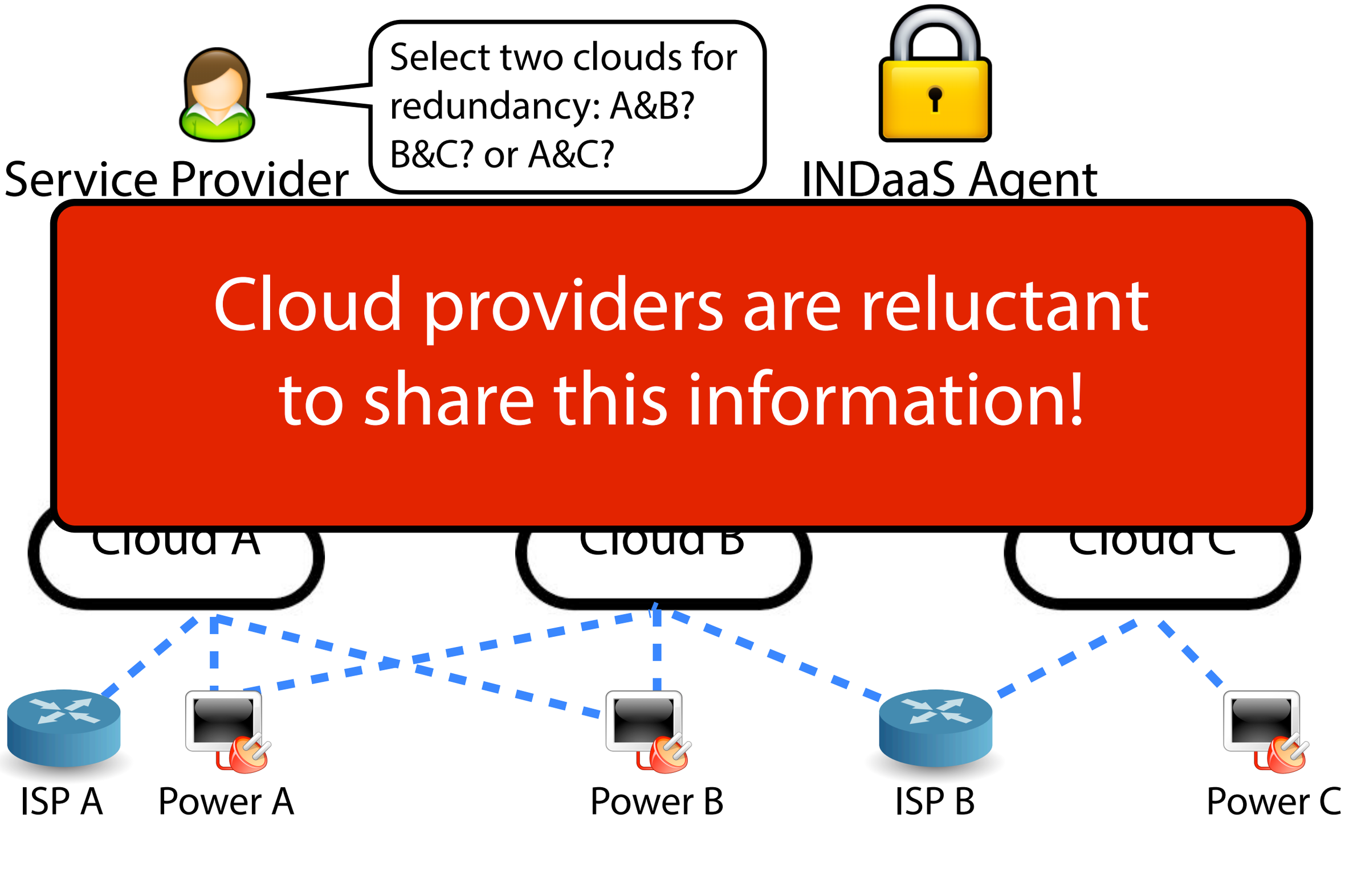


Trusted Third-Party



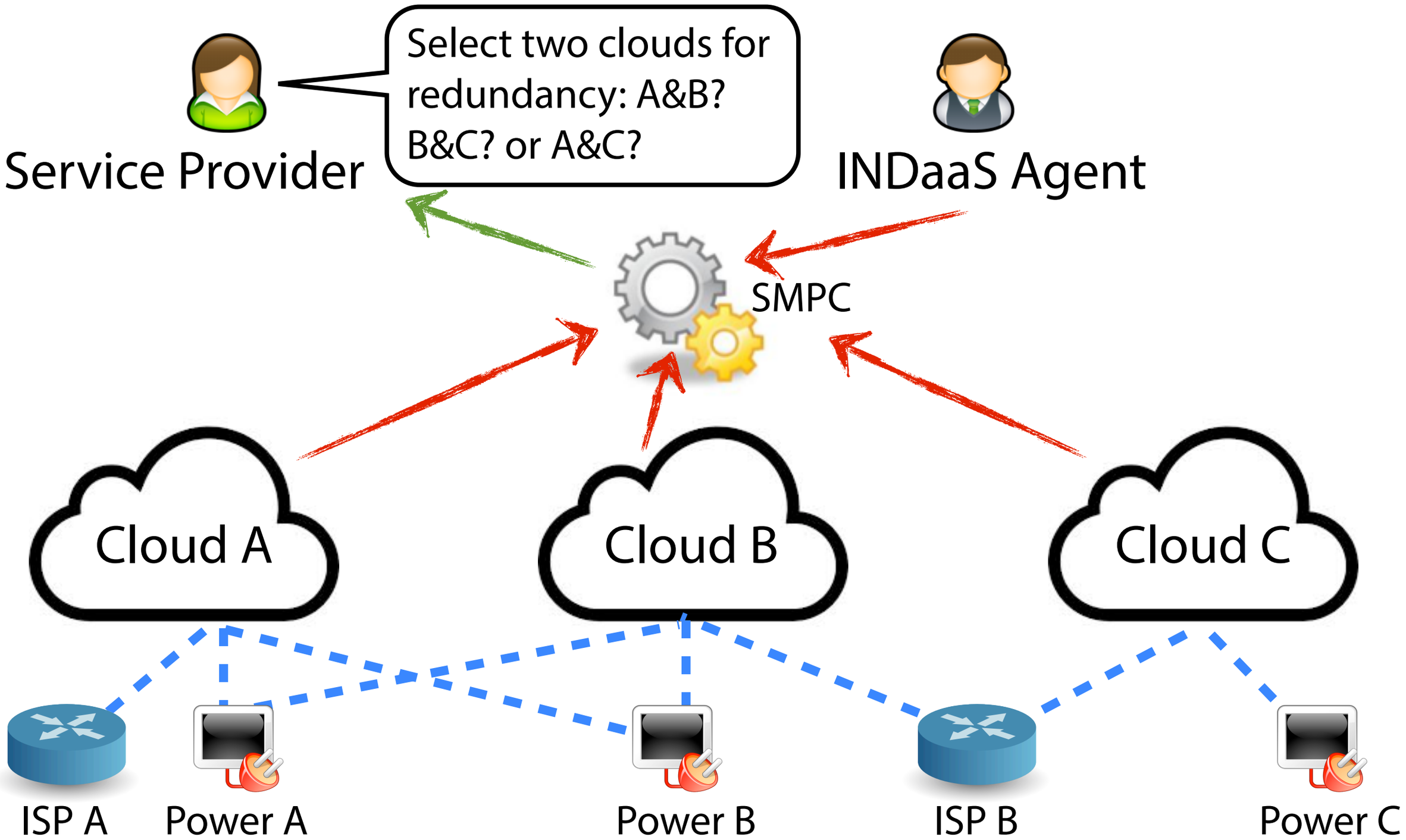


Trusted Third-Party

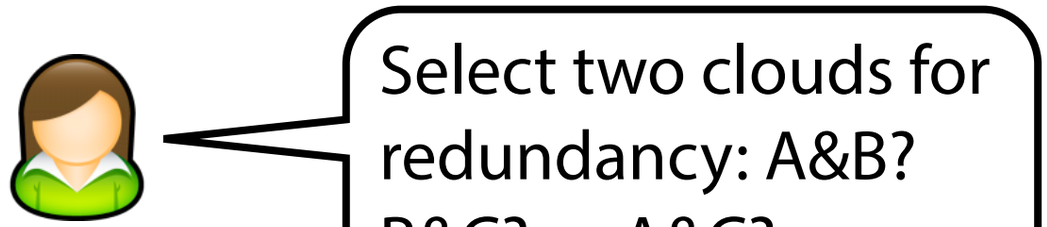


Cloud providers are reluctant to share this information!

Secure Multiparty Computation (SMPC)



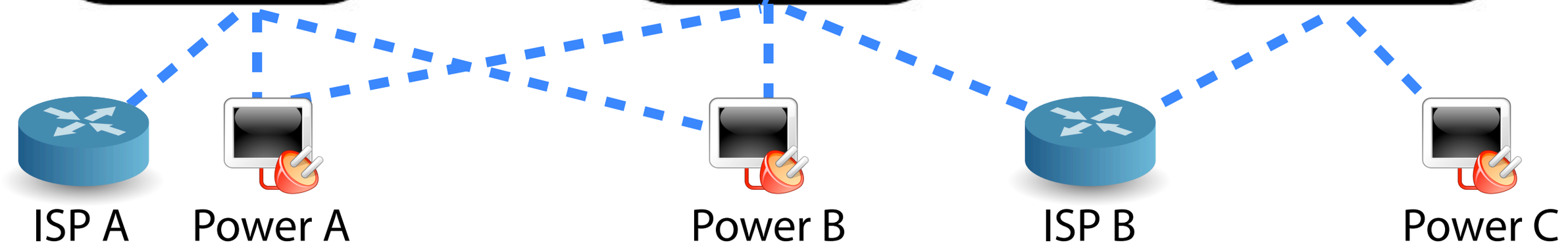
Secure Multiparty Computation (SMPC)



IND^2-CCA Agent

SMPC is hard to scale!

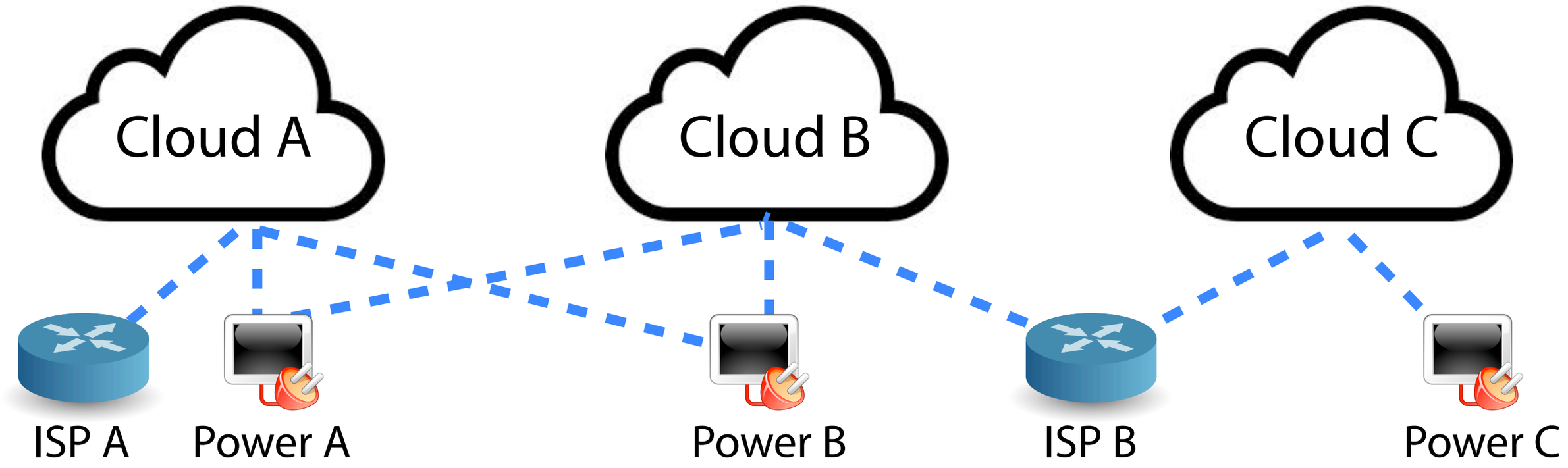
[Xiao et al. CCSW'13]



Select two clouds for redundancy: A&B? B&C? or A&C?

Service Provider

INDaaS Agent



Service Provider

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

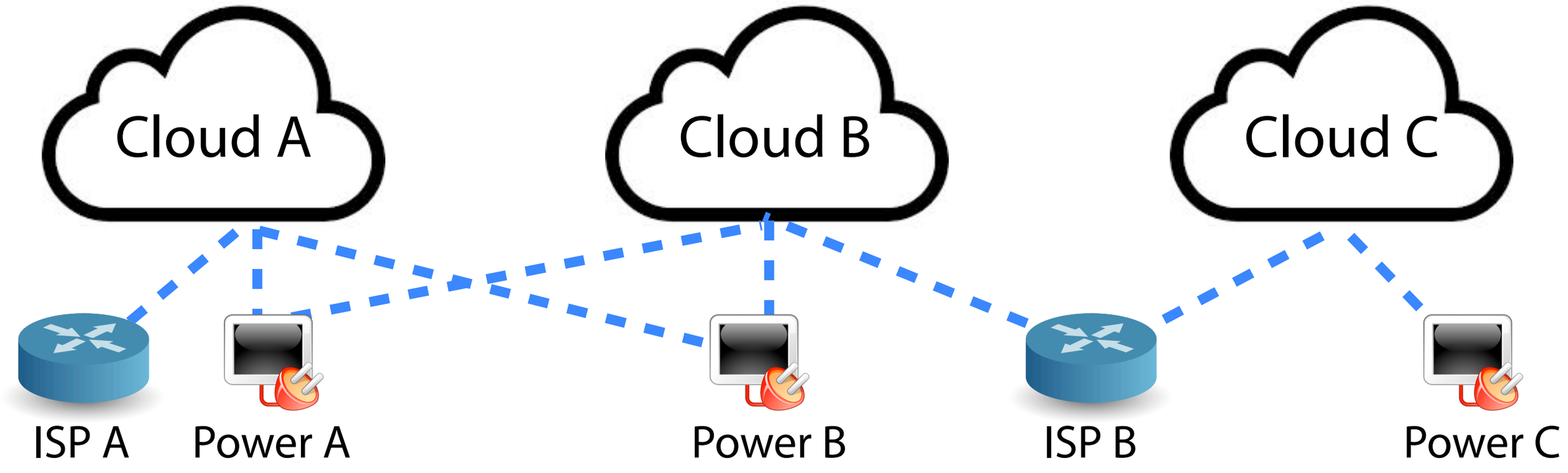
ISP B

Power C

Service Provider

INDaaS Agent

Evaluating independence by the dataset similarity between clouds



Service Provider

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

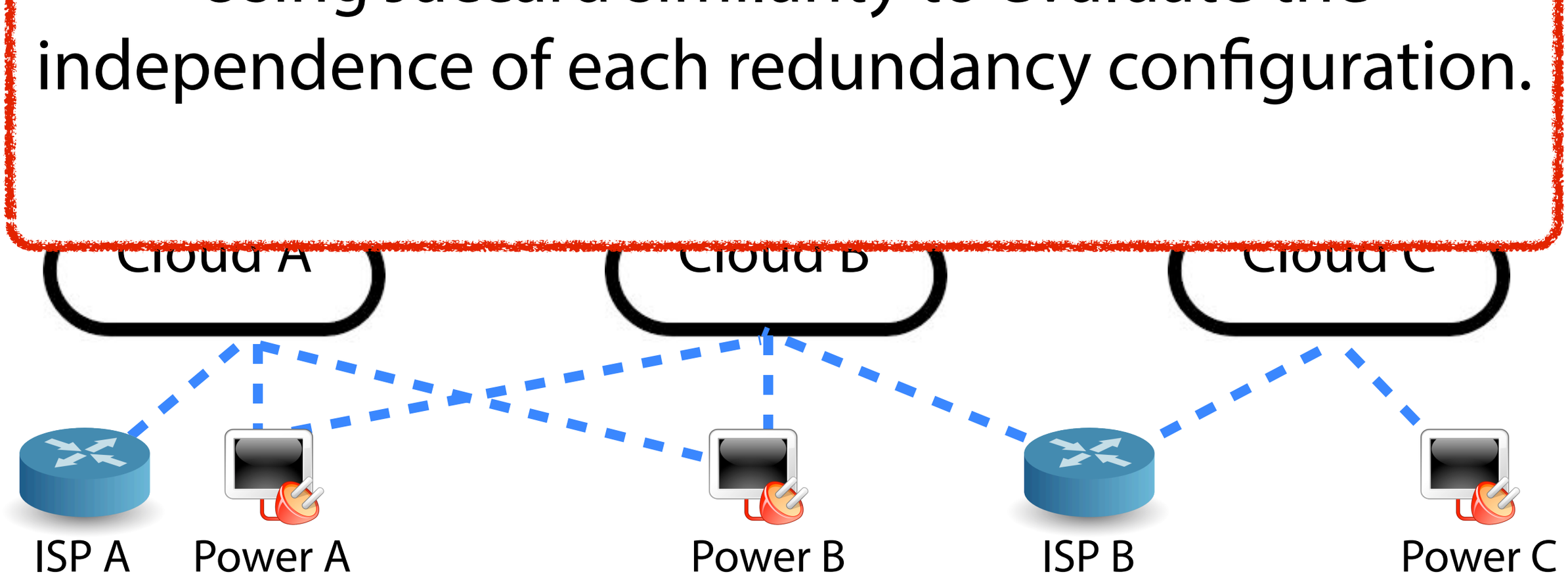
Power A

Power B

ISP B

Power C

Using Jaccard similarity to evaluate the independence of each redundancy configuration.

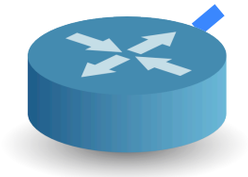


$$J(S_1, S_2, \ldots, S_n) = \frac{| S_1 \cap S_2 \cap \ldots\ldots \cap S_n |}{| S_1 \cup S_2 \cup \ldots\ldots \cup S_n |}$$

Cloud A

Cloud B

Cloud C

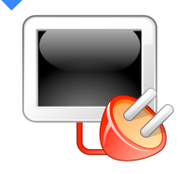


ISP A

Power A

Power B

ISP B



Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

=2

=4

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

ISP B
Power C

$\cap$ =2

$\cup$ =4

J = 2/4

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5

Service Provider

INDaaS Agent

ISP A
Power A
Power B

=2

=4

J = 2/4

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

ISP B
Power C

$\cap = 1$

$\cup = 4$

$J = 1/4$

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25
Cloud A&C	0

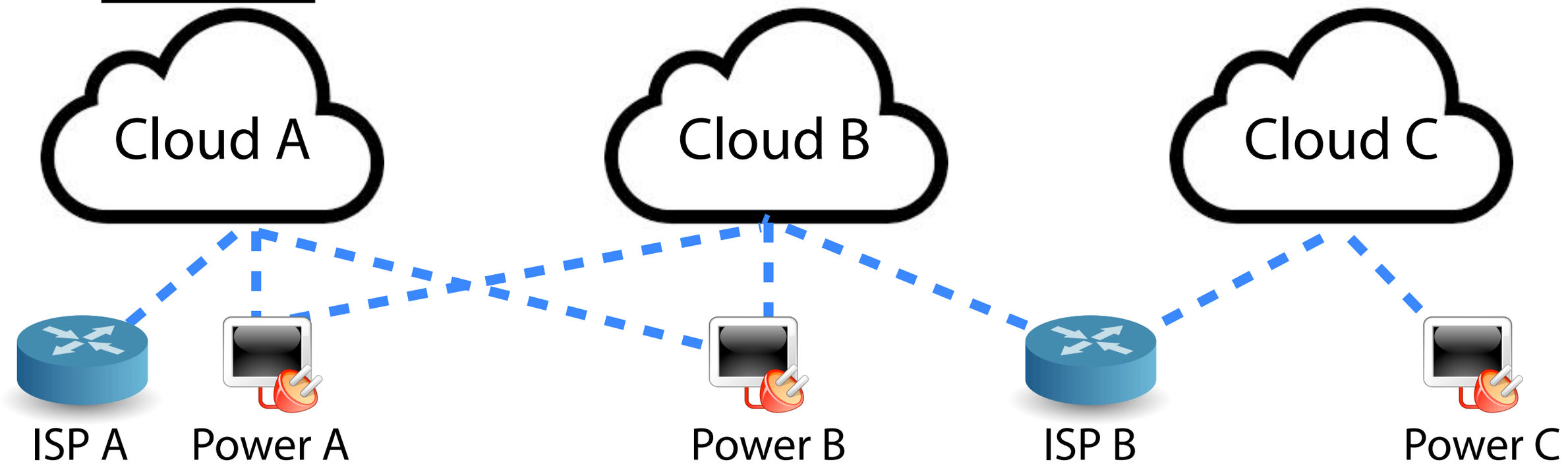
Service Provider

INDaaS Agent

ISP A
Power A
Power B

$\cap$ =0, $\cup$ =5 , J = 0/5

ISP B
Power C



Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25
Cloud A&C	0

Service Provider

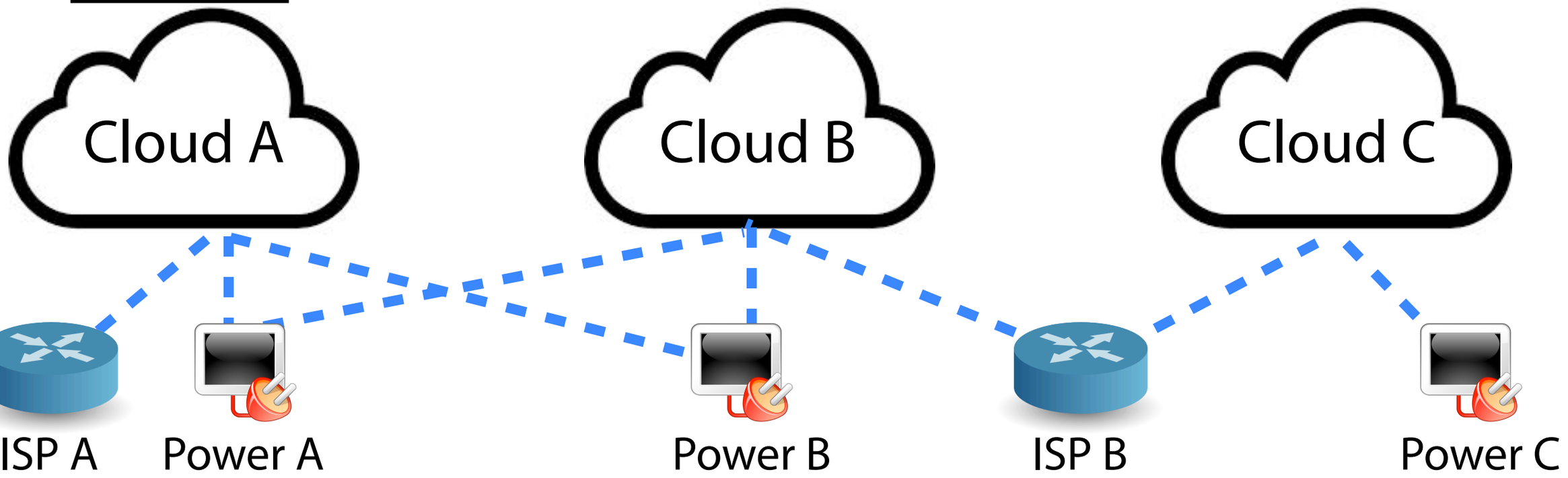
0 means fully independent

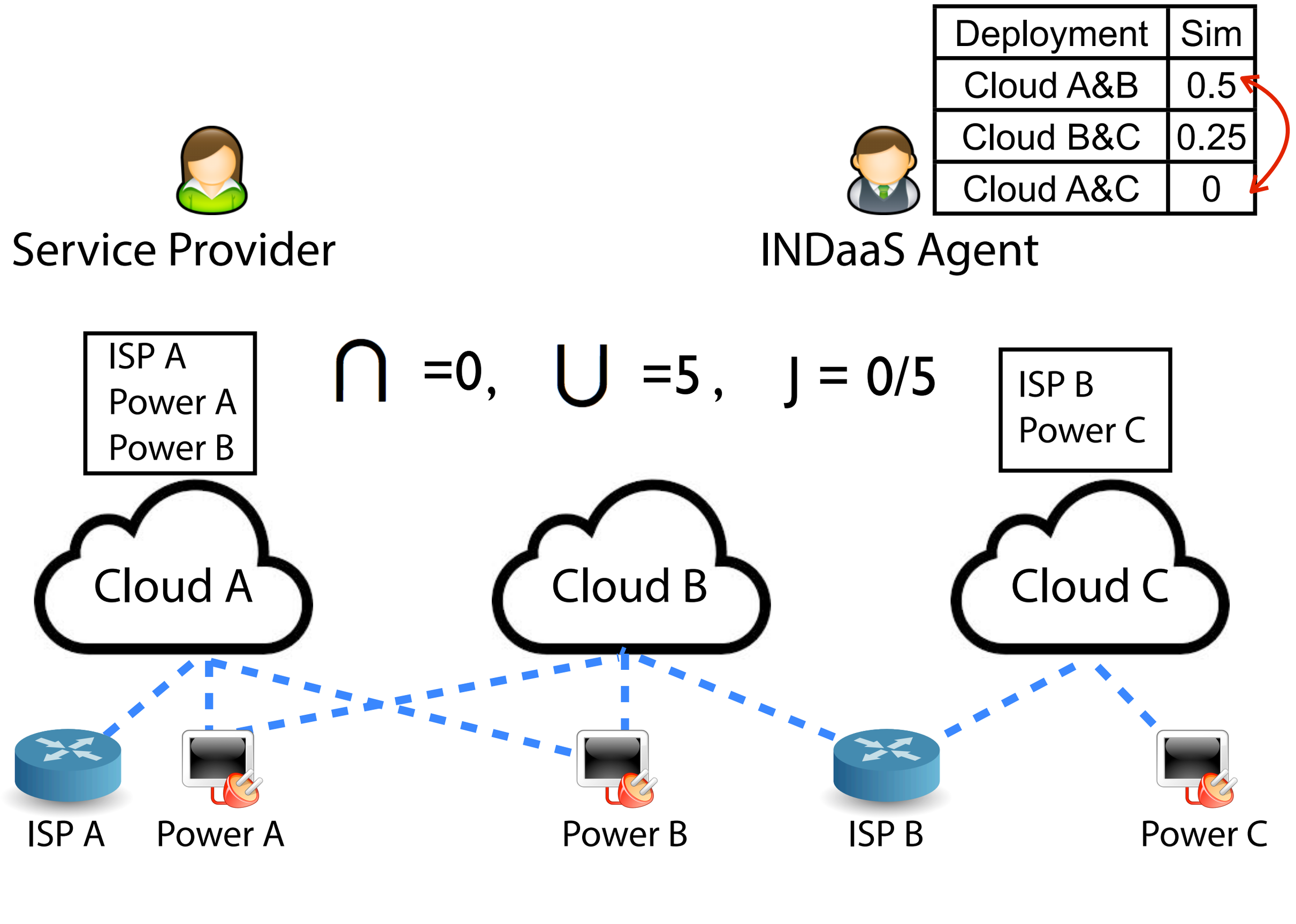
DaaS Agent

ISP A
Power A
Power B

$\cap$ =0, $\cup$ =5 , J = 0/5

ISP B
Power C




Deployment
Sim
Cloud A&B
0.5
Cloud B&C
0.25
Cloud A&C
0
Service Provider
INDaaS Agent
ISP A
Power A
Power B
∩ =0,
∪ =5 ,
J = 0/5
ISP B
Power C
Cloud A
Cloud B
Cloud C
ISP A
Power A
Power B
ISP B
Power C

Deployment	Sim
Cloud A&C	0
Cloud B&C	0.25
Cloud A&B	0.5

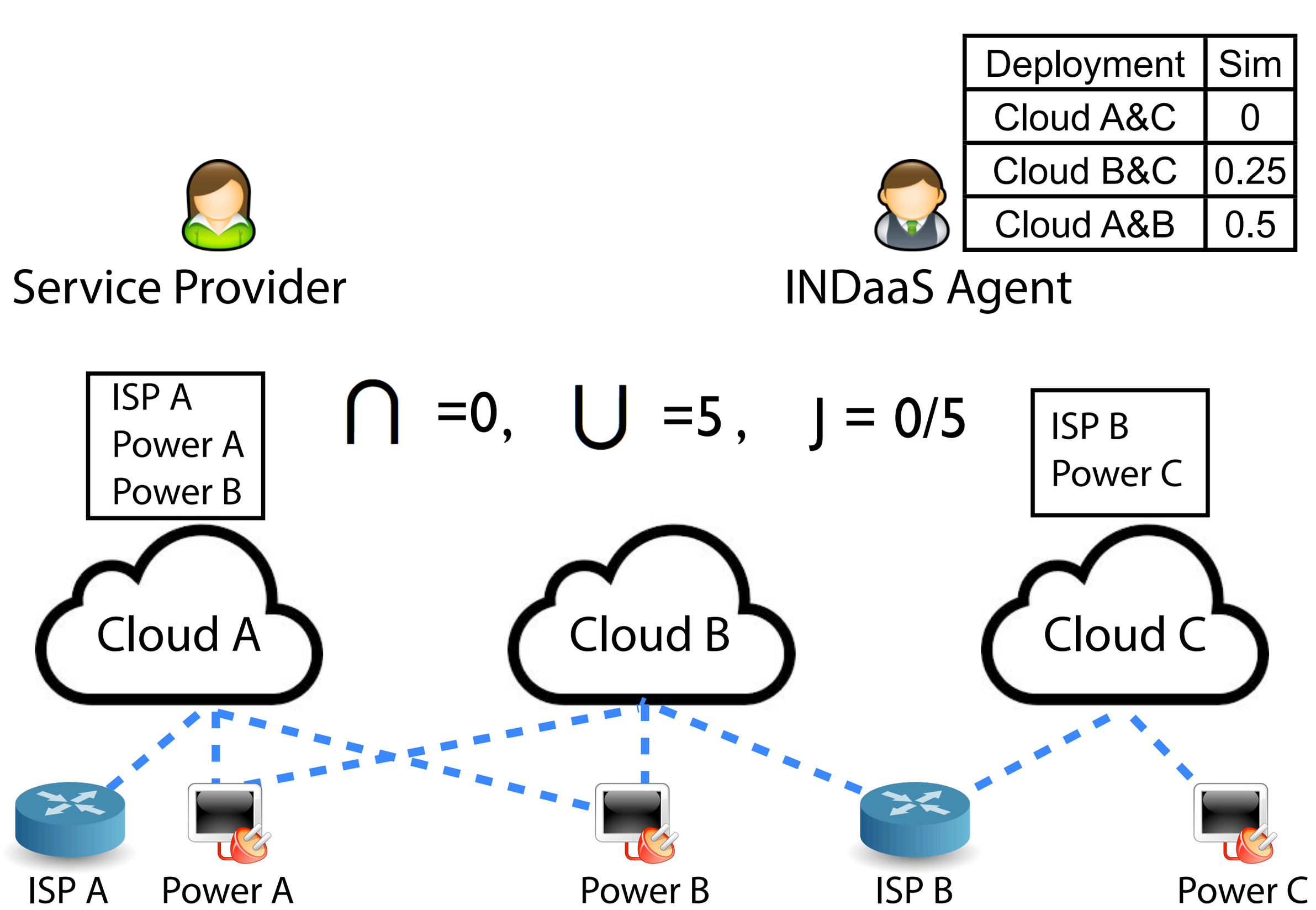
Service Provider

INDaaS Agent

ISP A
Power A
Power B

$\cap$ =0, $\cup$ =5 , J = 0/5

ISP B
Power C



Deployment	Sim
Cloud A&C	0
Cloud B&C	0.25
Cloud A&B	0.5

Service Provider

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

P-SOP [Vaidya et al. JCS05]

- We apply Private Set Operation Protocol (P-SOP):
  - Private set intersection cardinality.
  - Private set union cardinality.

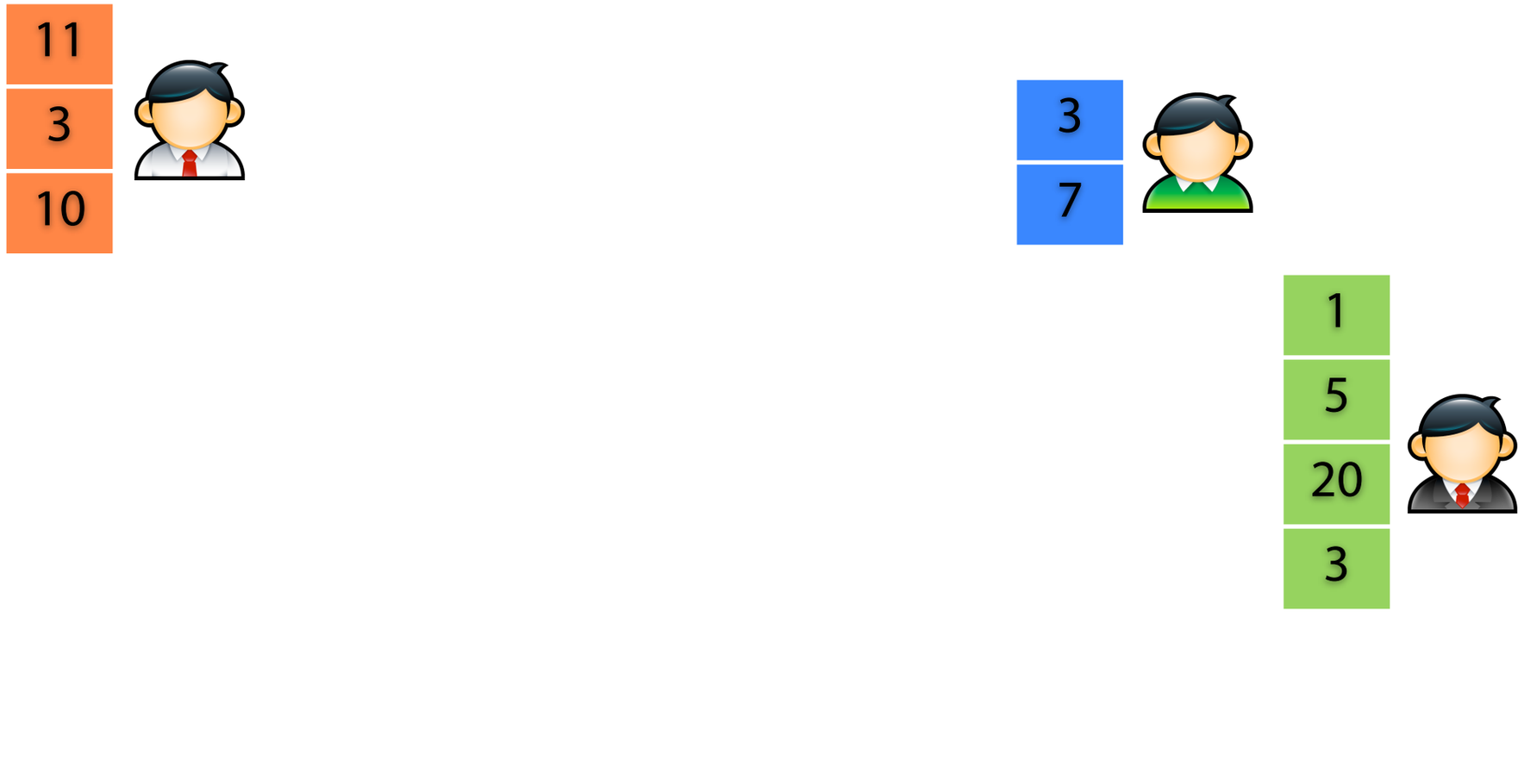
$$J(S_1, S_2, \ldots, S_n) = \frac{|\, S_1 \cap S_2 \cap \ldots\ldots \cap S_n \,|}{|\, S_1 \cup S_2 \cup \ldots\ldots \cup S_n \,|}$$

P-SOP [Vaidya et al. JCS05]

- Allow k parties to compute both intersection and union cardinalities without learning other information.

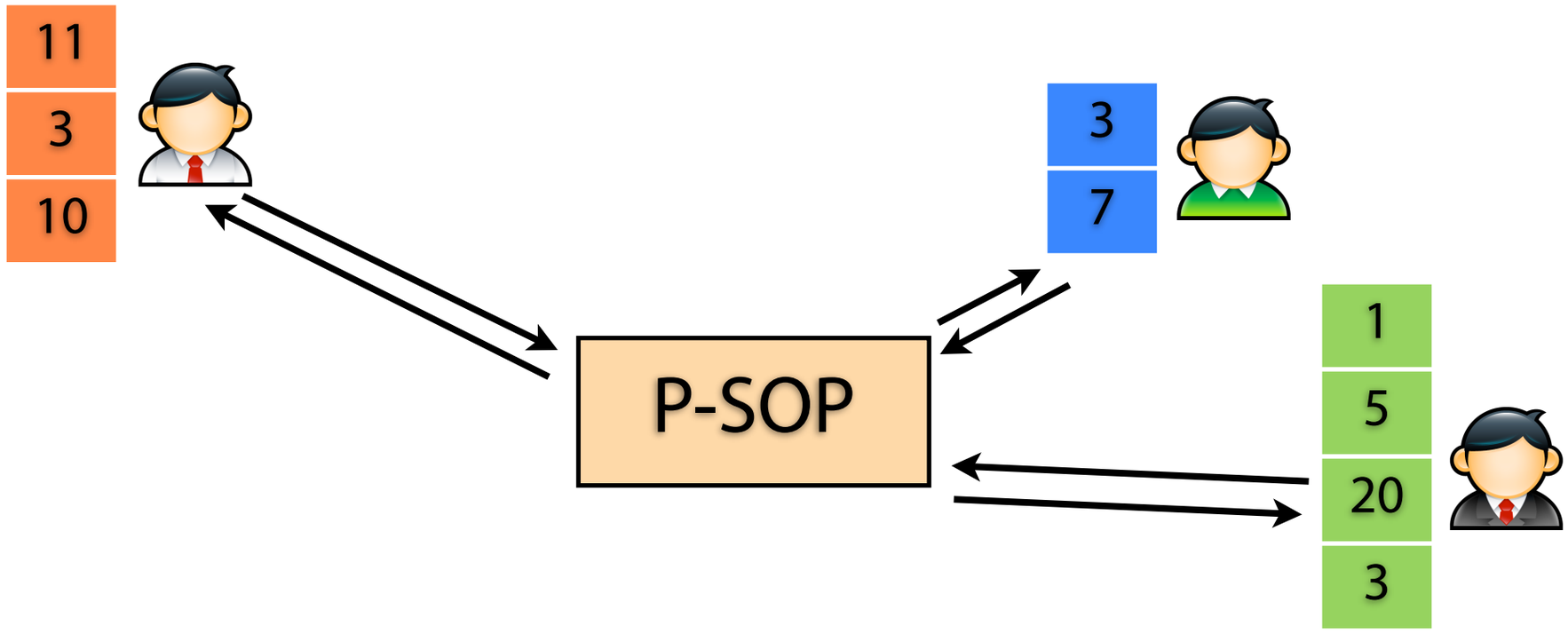
P-SOP [Vaidya et al. JCS05]

- Allow k parties to compute both intersection and union cardinalities without learning other information.



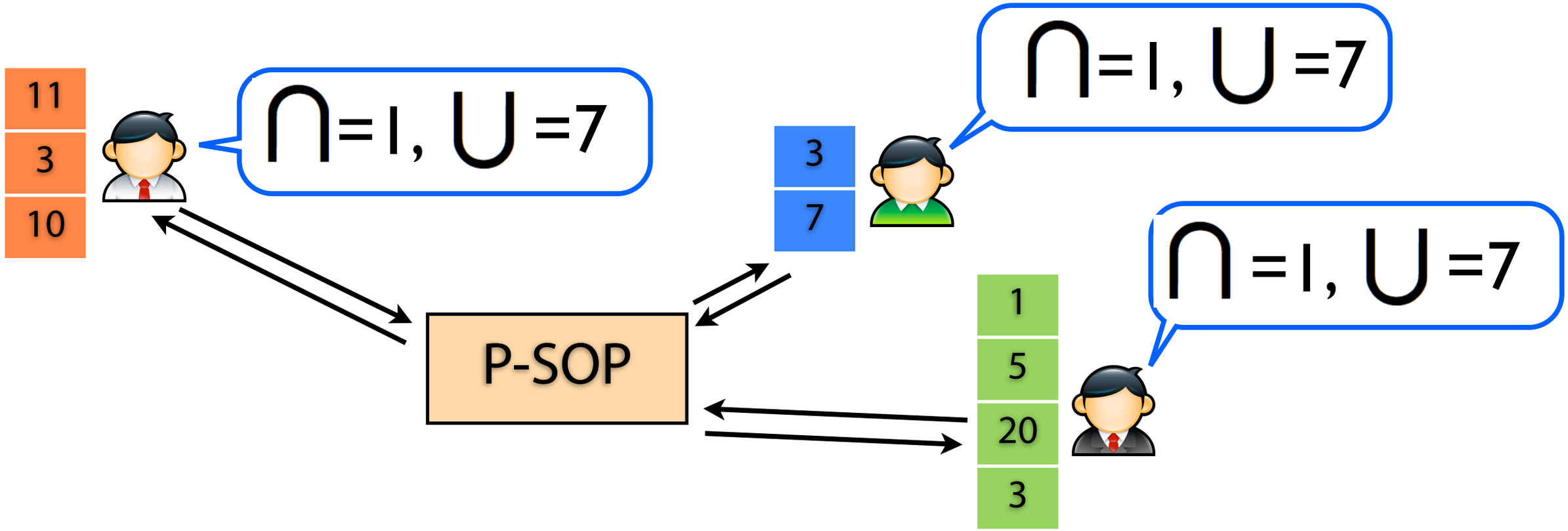
P-SOP [Vaidya et al. JCS05]

- Allow k parties to compute both intersection and union cardinalities without learning other information.



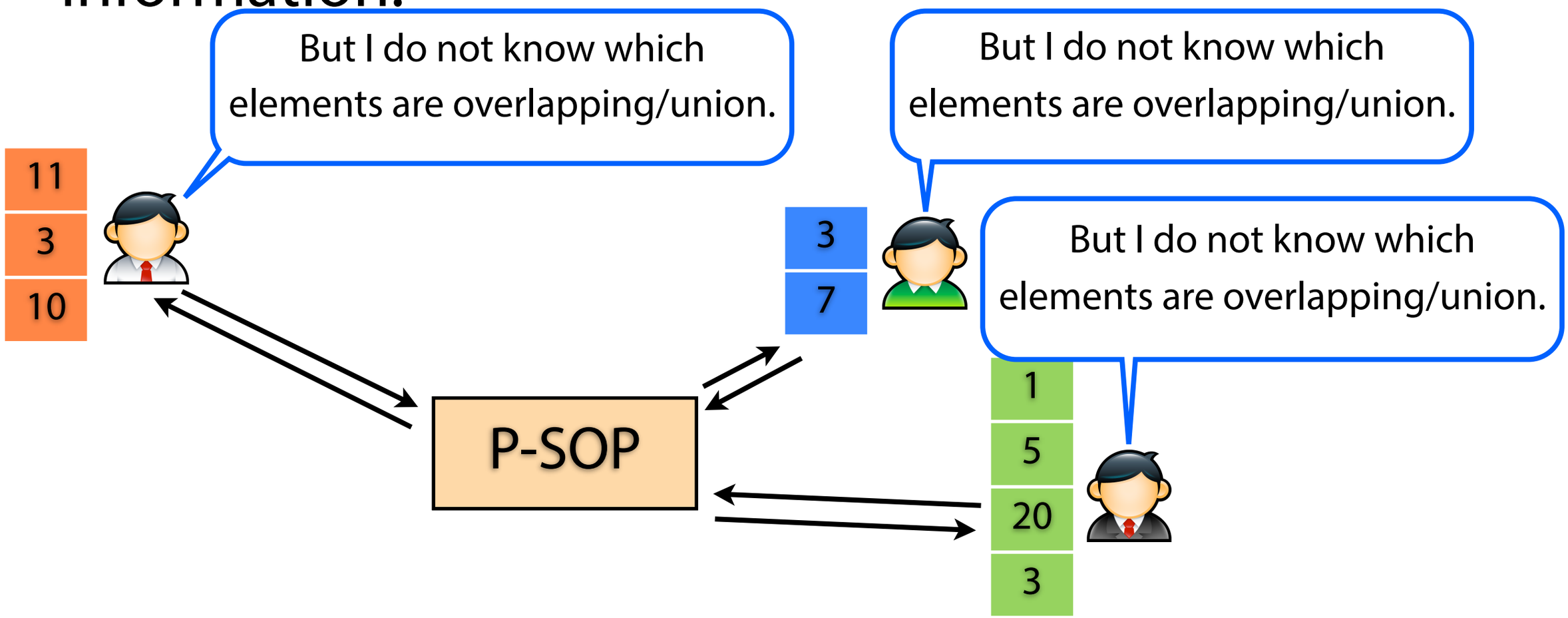
P-SOP [Vaidya et al. JCS05]

- Allow k parties to compute both intersection and union cardinalities without learning other information.



P-SOP [Vaidya et al. JCS05]

- Allow k parties to compute both intersection and union cardinalities without learning other information.



P-SOP [Vaidya et al. JCS05]

Each party maintains a commutative encryption key

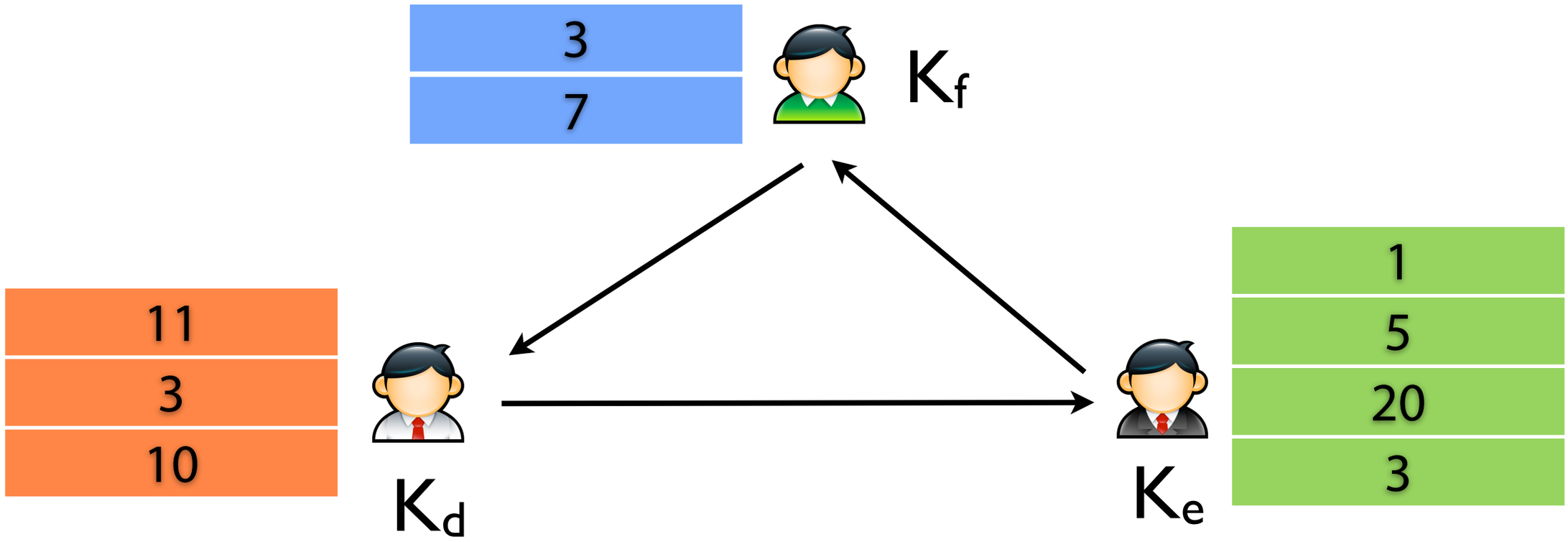
Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

P-SOP [Vaidya et al. JCS05]

Each party maintains a commutative encryption key

Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

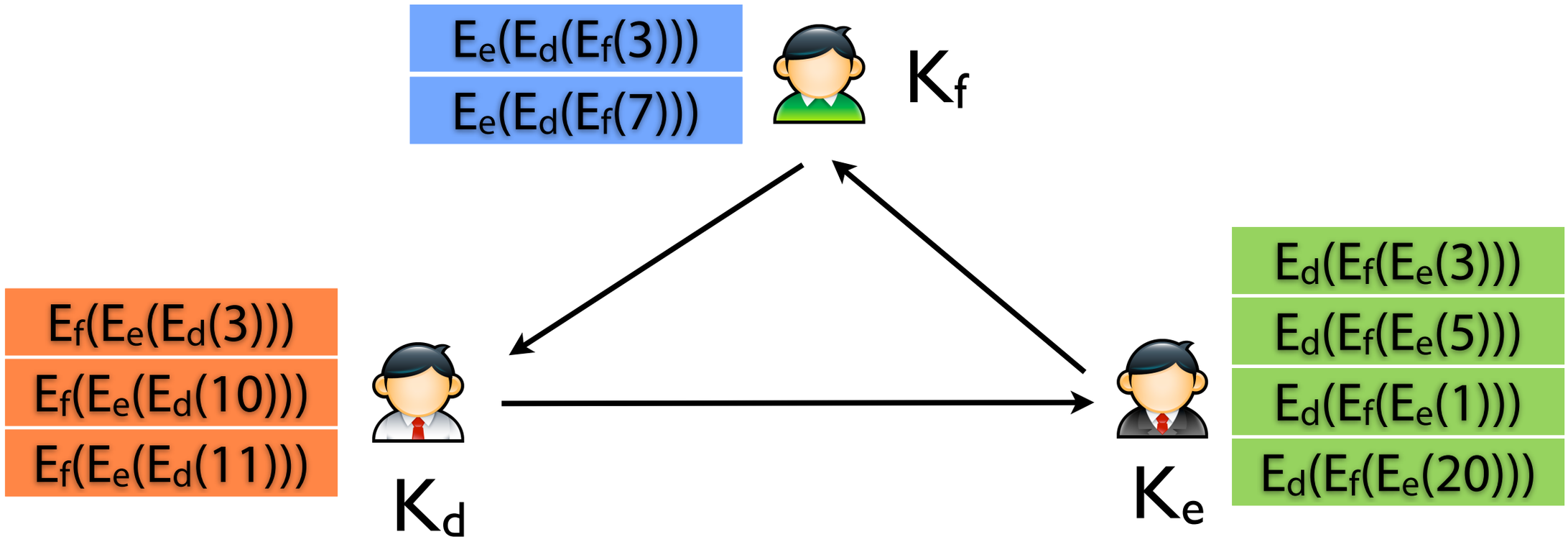
P-SOP [Vaidya et al. JCS05]



Each party maintains a commutative encryption key

Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

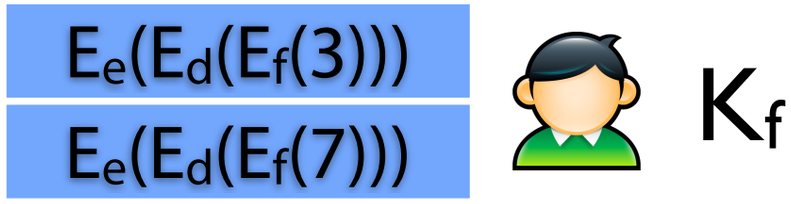
P-SOP [Vaidya et al. JCS05]



Each party maintains a commutative encryption key

Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

P-SOP [Vaidya et al. JCS05]



Each party maintains a commutative encryption key

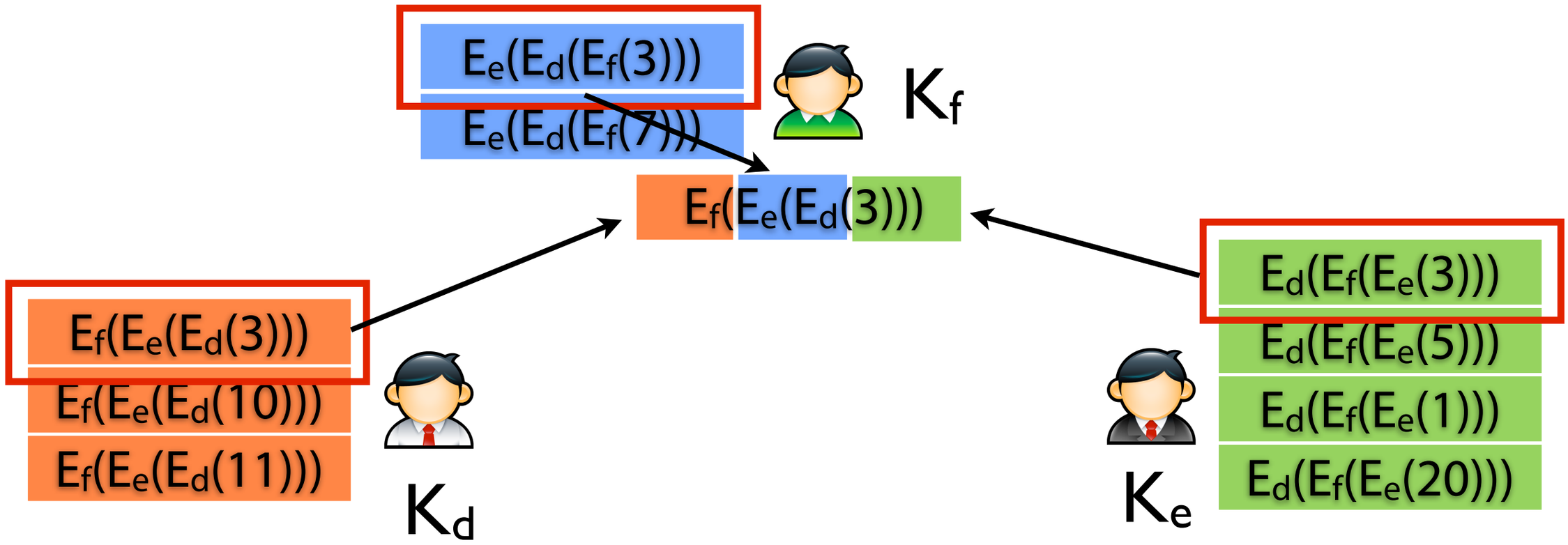
Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

P-SOP [Vaidya et al. JCS05]

Each party maintains a commutative encryption key

Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

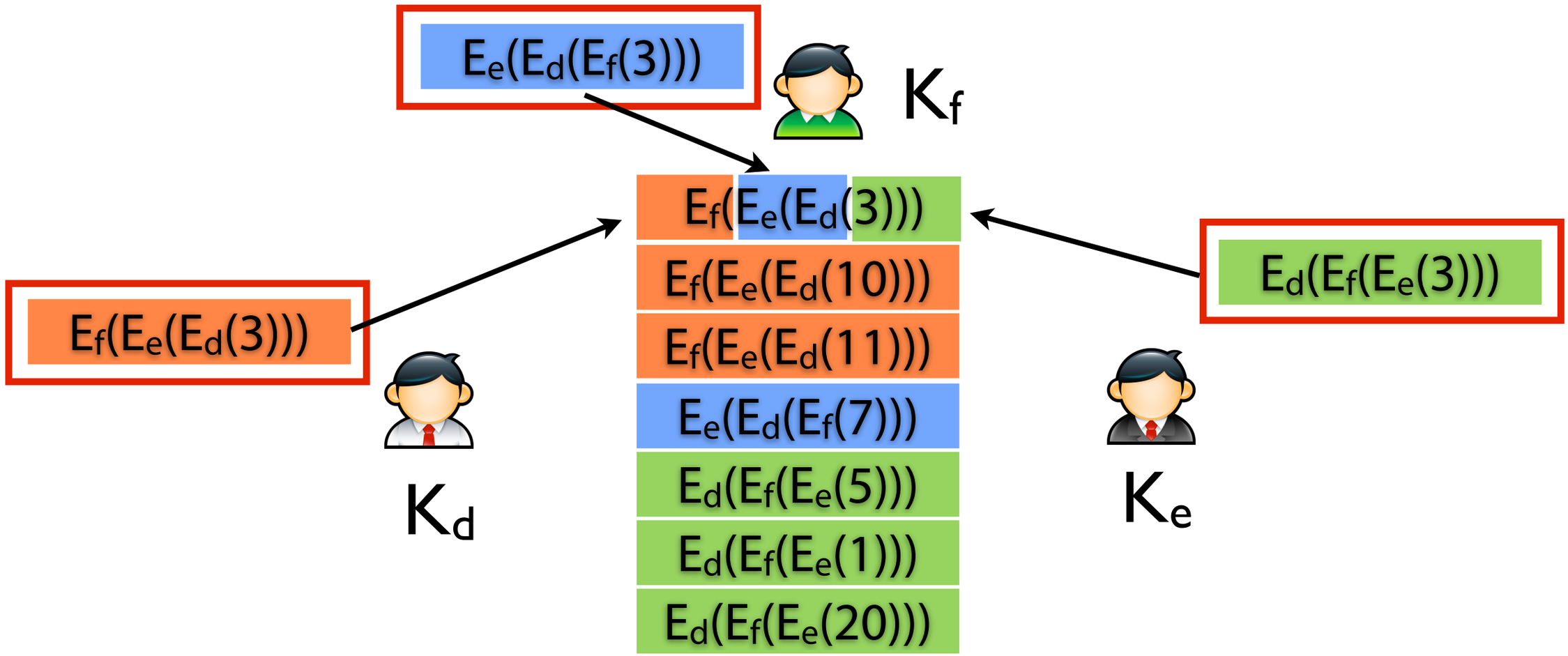
P-SOP [Vaidya et al. JCS05]



Each party maintains a commutative encryption key

Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

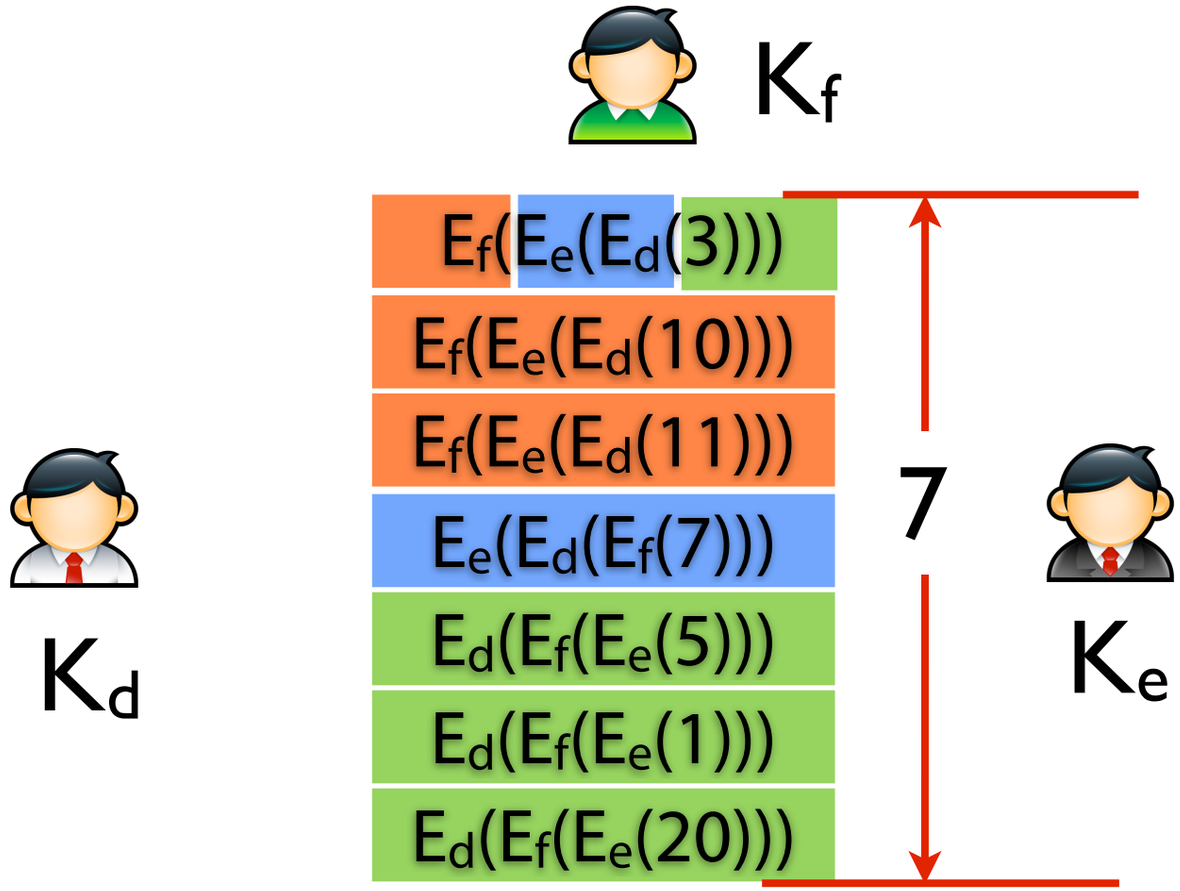
P-SOP [Vaidya et al. JCS05]



Each party maintains a commutative encryption key

Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

P-SOP [Vaidya et al. JCS05]



Each party maintains a commutative encryption key

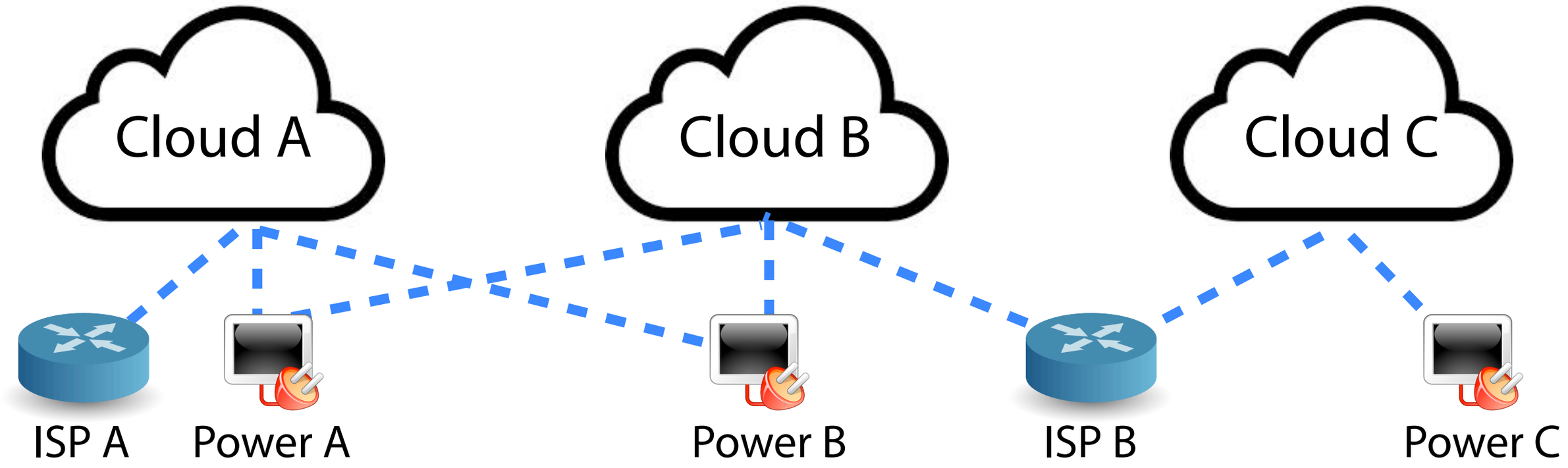
Commutative encryption holds: $E_x(E_y(m)) = E_y(E_x(m))$

Private Independence Evaluation

Select two clouds for redundancy: A&B? B&C? or A&C?

Service Provider

INDaaS Agent



Service Provider

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

P-SOP

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

∩ =2

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

=2

=4

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

ISP B
Power C

=2

=4

J = 2/4

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5

Service Provider

INDaaS Agent

ISP A
Power A
Power B

=2

=4

J = 2/4

ISP B
Power A
Power B

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

P-SOP

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

$= I$

$\cup = 4$

$J = 1/4$

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25

Service Provider

INDaaS Agent

ISP A
Power A
Power B

ISP B
Power A
Power B

$= I \cup = 4$

$J = 1/4$

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25

Service Provider

INDaaS Agent

ISP A
Power A
Power B

P-SOP

ISP B
Power C

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25

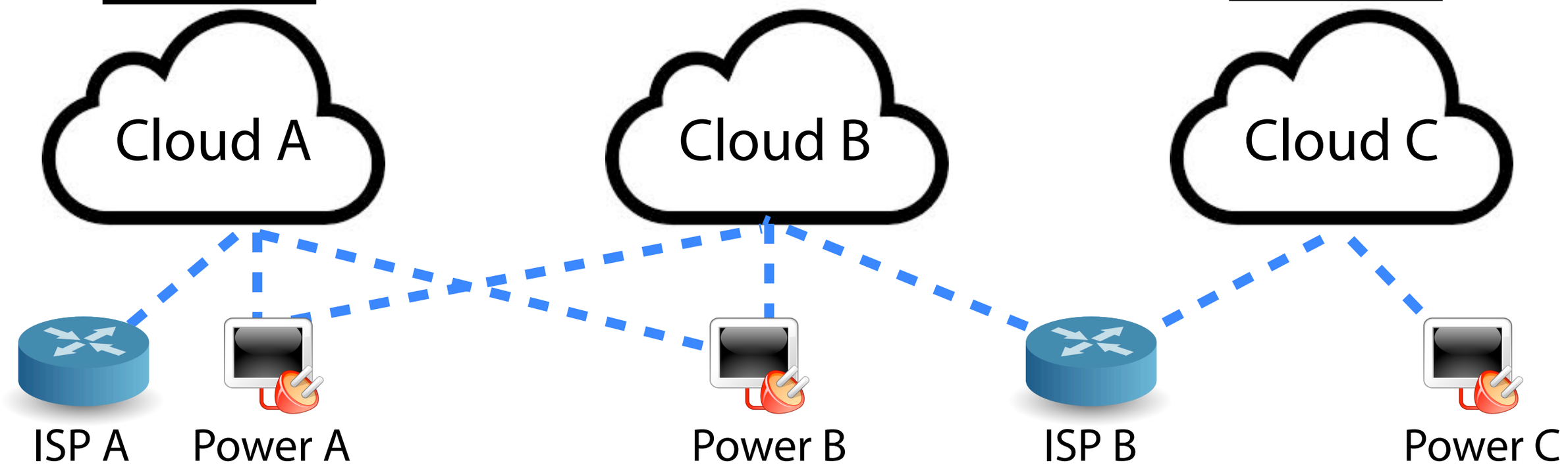
Service Provider

INDaaS Agent

ISP A
Power A
Power B

$\cap$ =0, $\cup$ =5 , J = 0/5

ISP B
Power C



Deployment	Sim
Cloud A&B	0.5
Cloud B&C	0.25
Cloud A&C	0

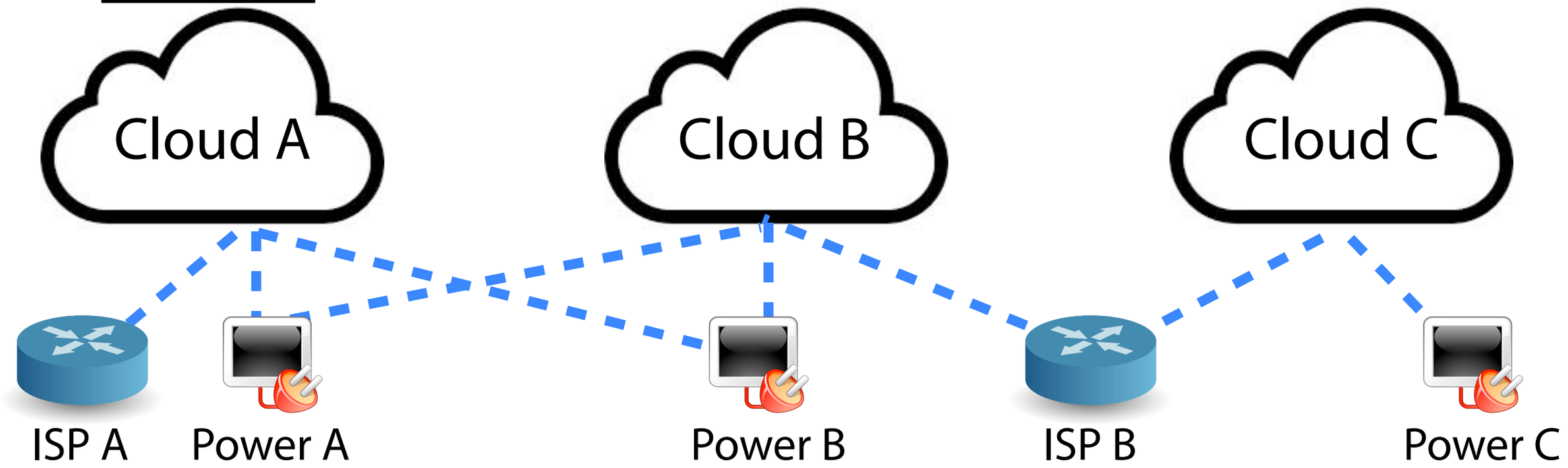
Service Provider

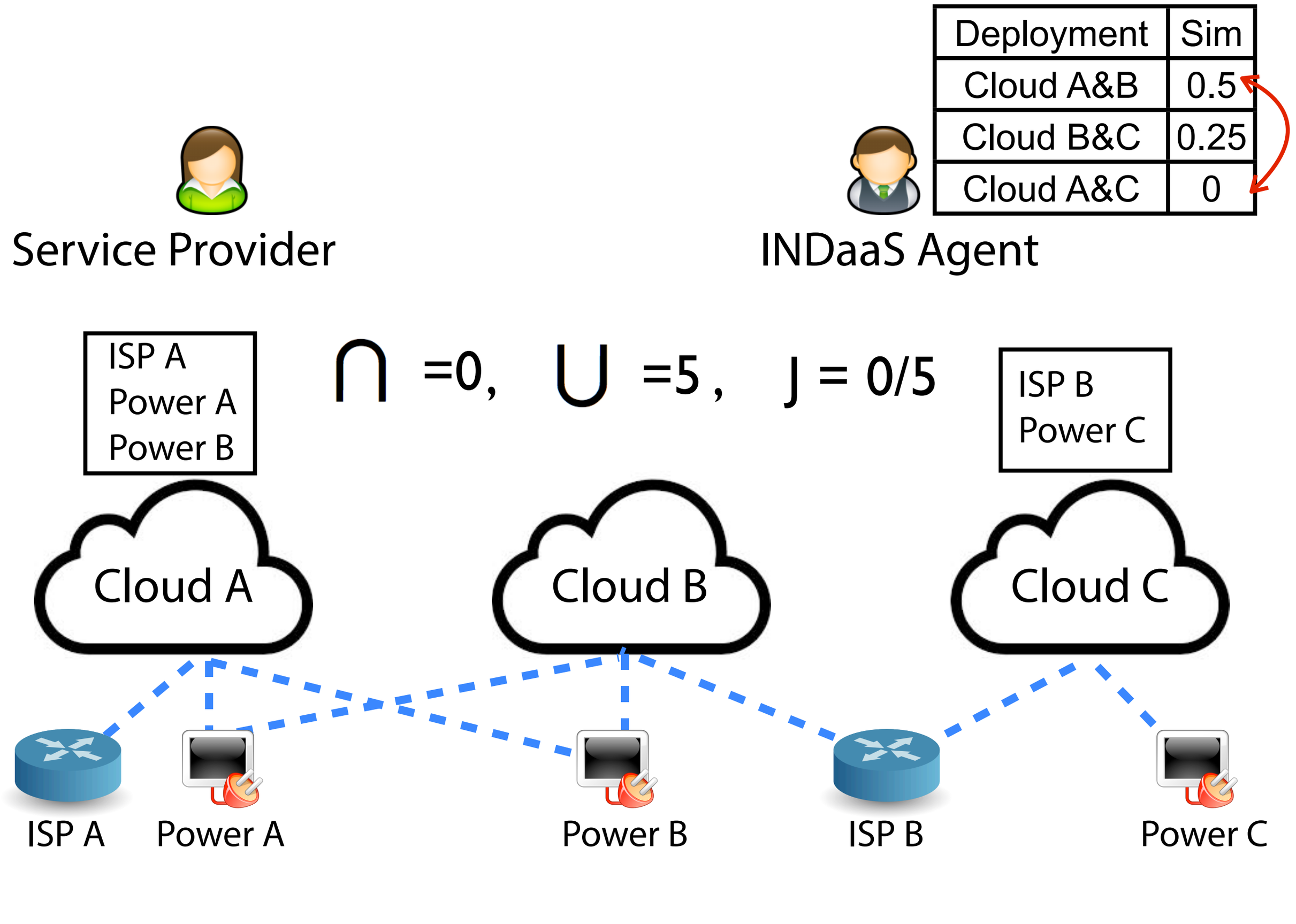
INDaaS Agent

ISP A
Power A
Power B

$\cap$ =0, $\cup$ =5 , J = 0/5

ISP B
Power C




Deployment
Sim
Cloud A&B
0.5
Cloud B&C
0.25
Cloud A&C
0
Service Provider
INDaaS Agent
ISP A
Power A
Power B
∩ =0,
∪ =5 ,
J = 0/5
ISP B
Power C
Cloud A
Cloud B
Cloud C
ISP A
Power A
Power B
ISP B
Power C

Deployment	Sim
Cloud A&C	0
Cloud B&C	0.25
Cloud A&B	0.5

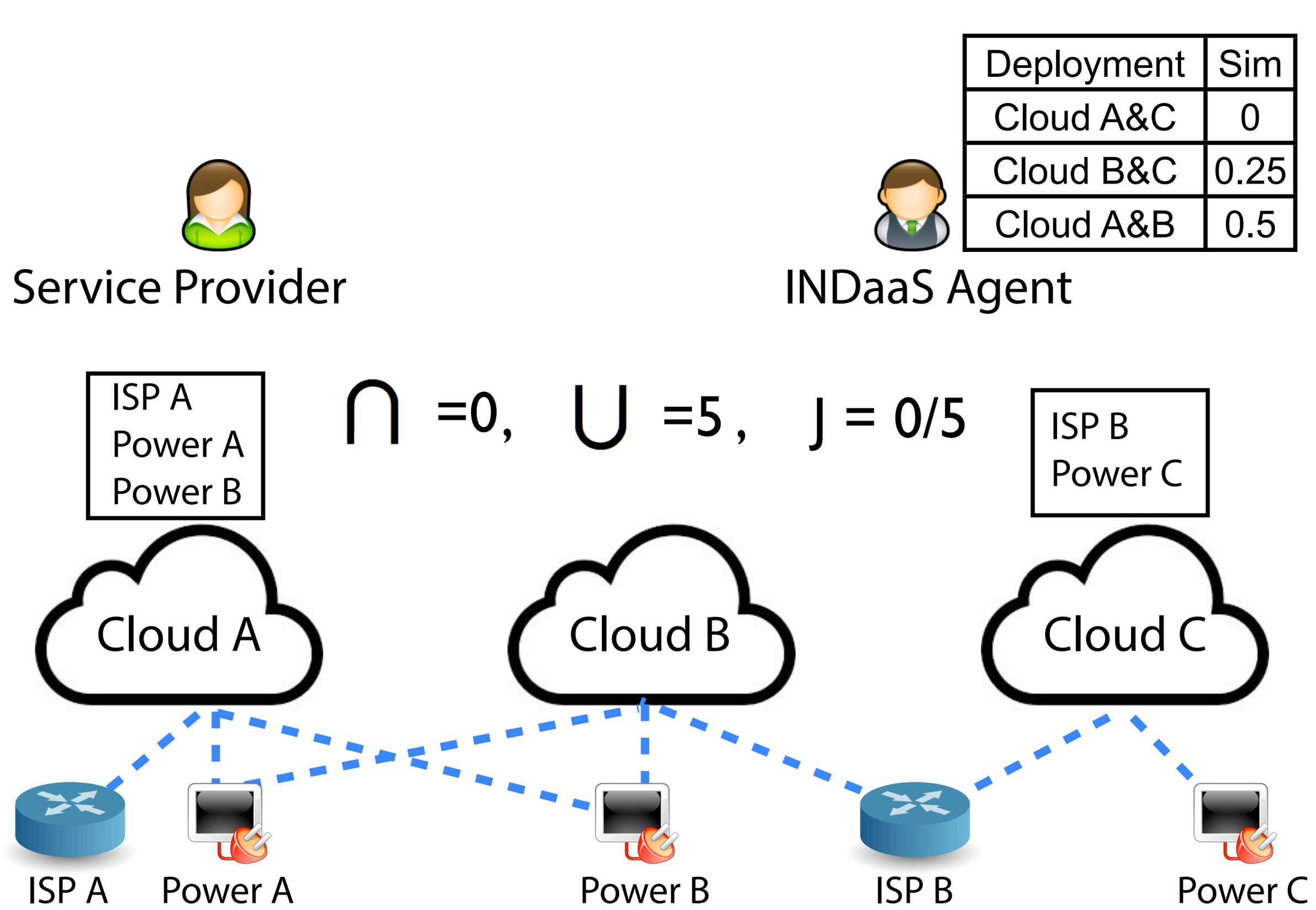
Service Provider

INDaaS Agent

ISP A
Power A
Power B

$\cap$ =0, $\cup$ =5 , J = 0/5

ISP B
Power C



Deployment	Sim
Cloud A&C	0
Cloud B&C	0.25
Cloud A&B	0.5

Service Provider

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

Power A

Power B

ISP B

Power C

Service Provider

Deployment
Cloud A&C
Cloud B&C
Cloud A&B

INDaaS Agent

Cloud A

Cloud B

Cloud C

ISP A

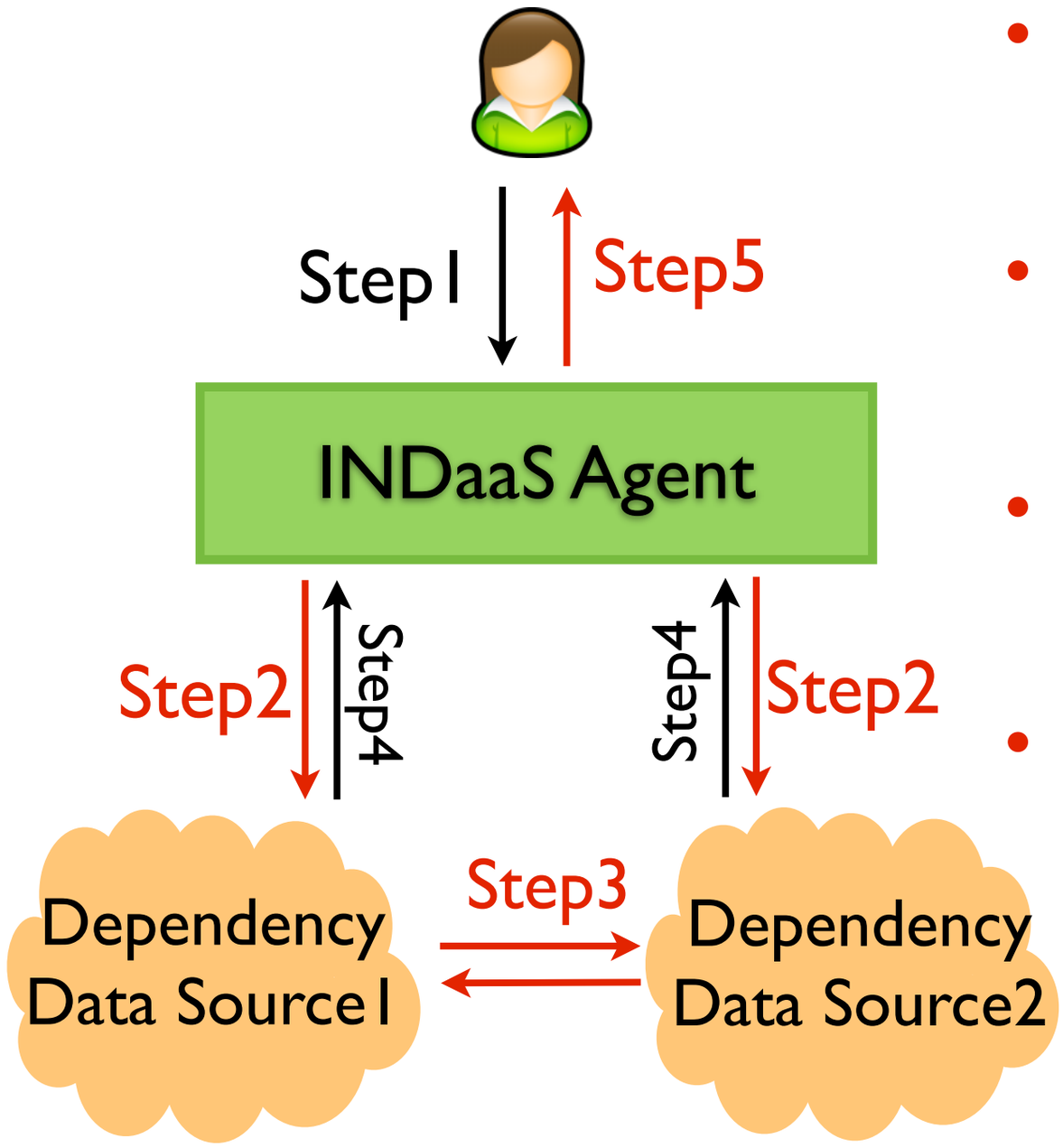
Power A

Power B

ISP B

Power C

RoadMap



- #1: Dependency collections
  - Solution: Reusing existing tools
- #2: Dependency representation
  - Solution: Fault graphs
- #3: Efficient auditing
  - Solution: Failure sampling algorithm
- #4: Private independence audit
  - Solution: Private Jaccard similarity

Evaluation

Evaluation

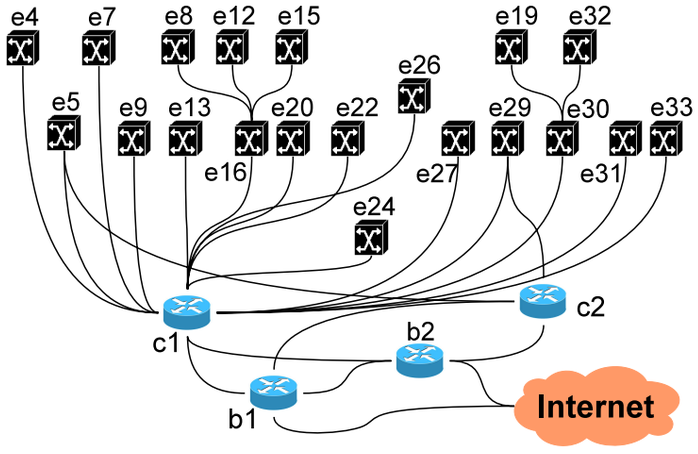
- Three realistic case studies.
- Tradeoff between auditing algorithms
- Overhead of P-SOP protocol

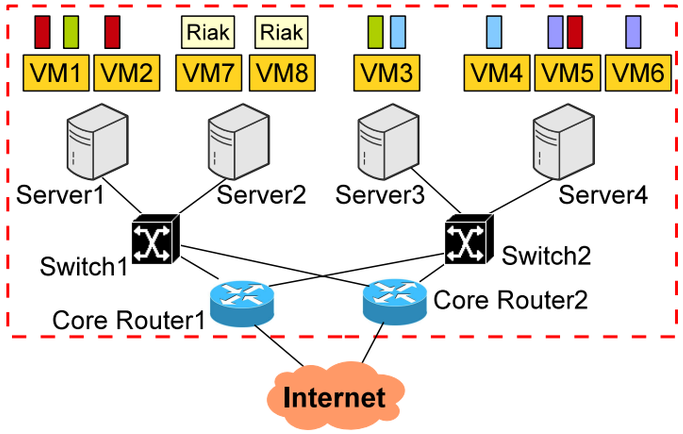
Evaluation

- Three realistic case studies.
- Tradeoff between auditing algorithms
- Overhead of P-SOP protocol

Three Case Studies

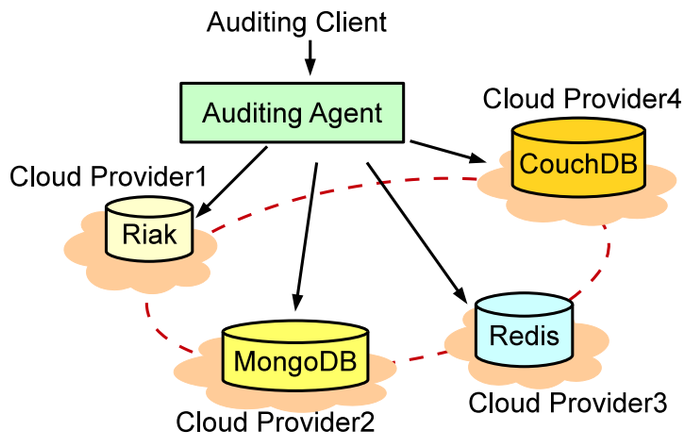
- Common network dependency





- Common hardware dependency

- Common software dependency



Please see our paper for more details

Evaluation

- Three realistic case studies.
- Tradeoff between auditing algorithms
- Overhead of P-SOP protocol

Tradeoff Evaluation

- We evaluate efficiency/accuracy tradeoff.
- We generate topology based on fat tree model.

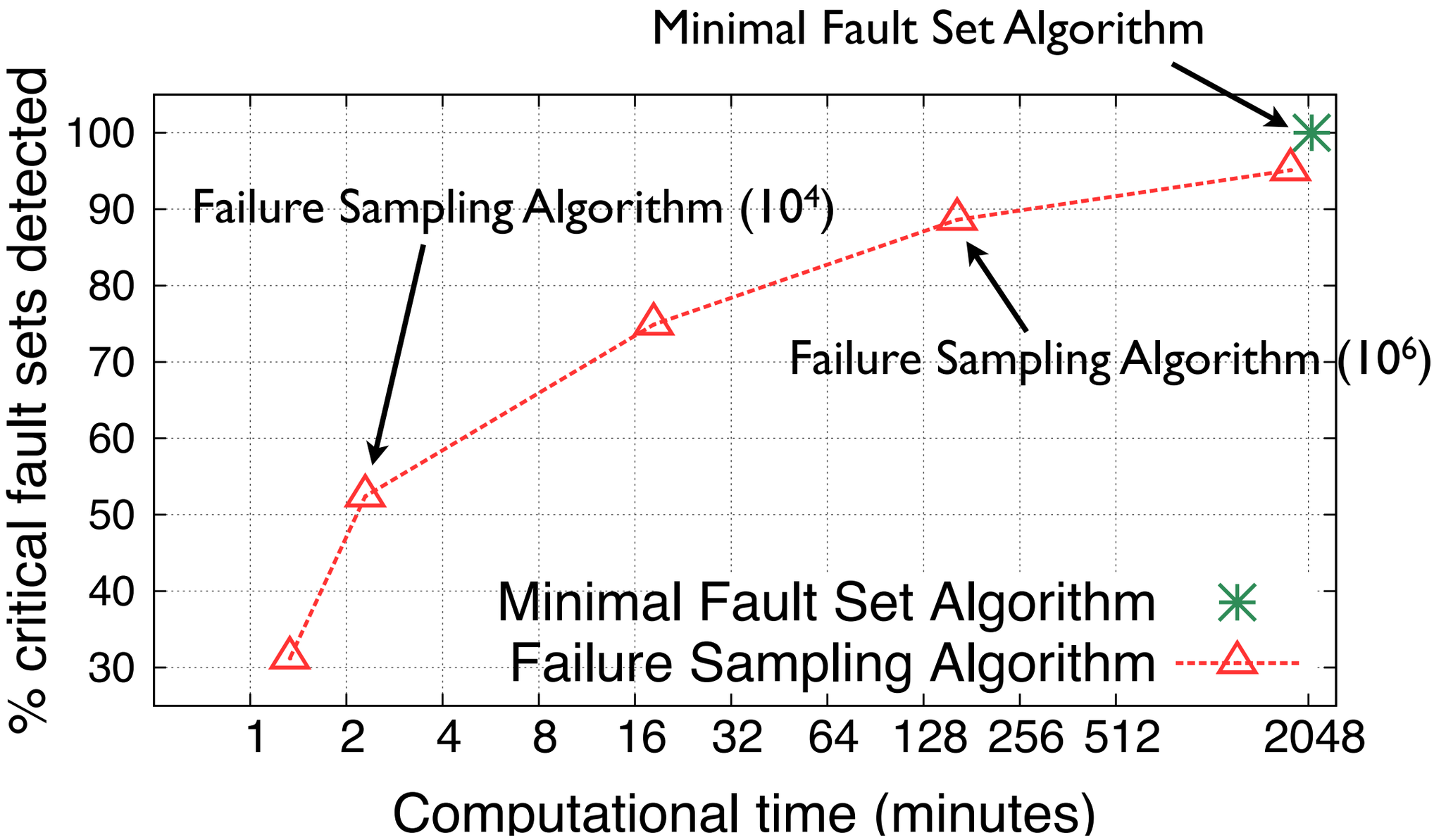
Tradeoff Evaluation

	Topology A	Topology B	Topology C
# of Core Routers	64	144	576
# of Agg Switches	128	288	1,152
# of ToR Switches	128	288	1,152
# of Servers	1,024	3,456	27,648
Total # of devices	1,344	4,176	30,528

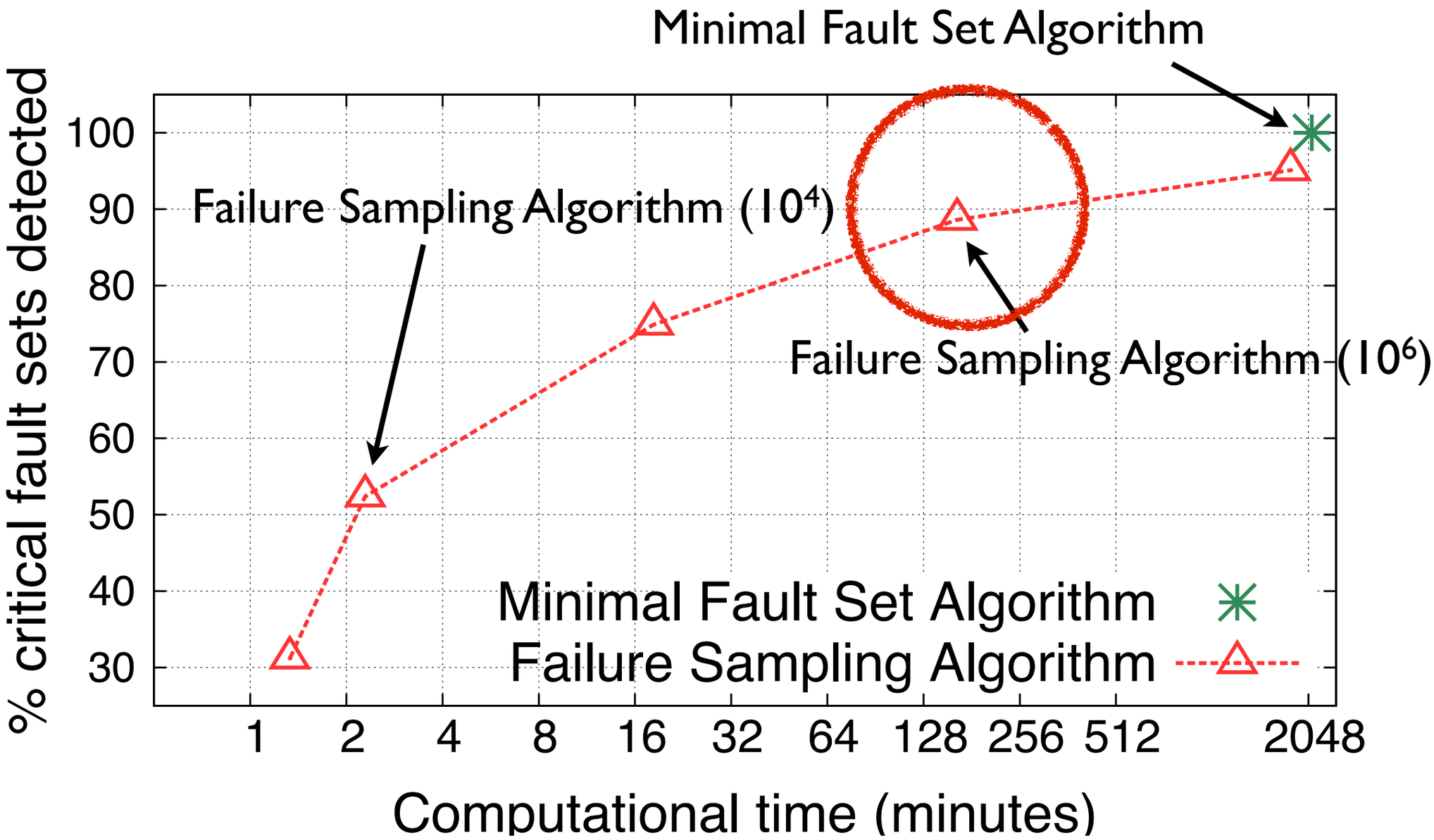
Tradeoff Evaluation

	Topology A	Topology B	Topology C
# of Core Routers	64	144	576
# of Agg Switches	128	288	1,152
# of ToR Switches	128	288	1,152
# of Servers	1,024	3,456	27,648
Total # of devices	1,344	4,176	30,528

Topology C: 30,528 Devices



Topology C: 30,528 Devices



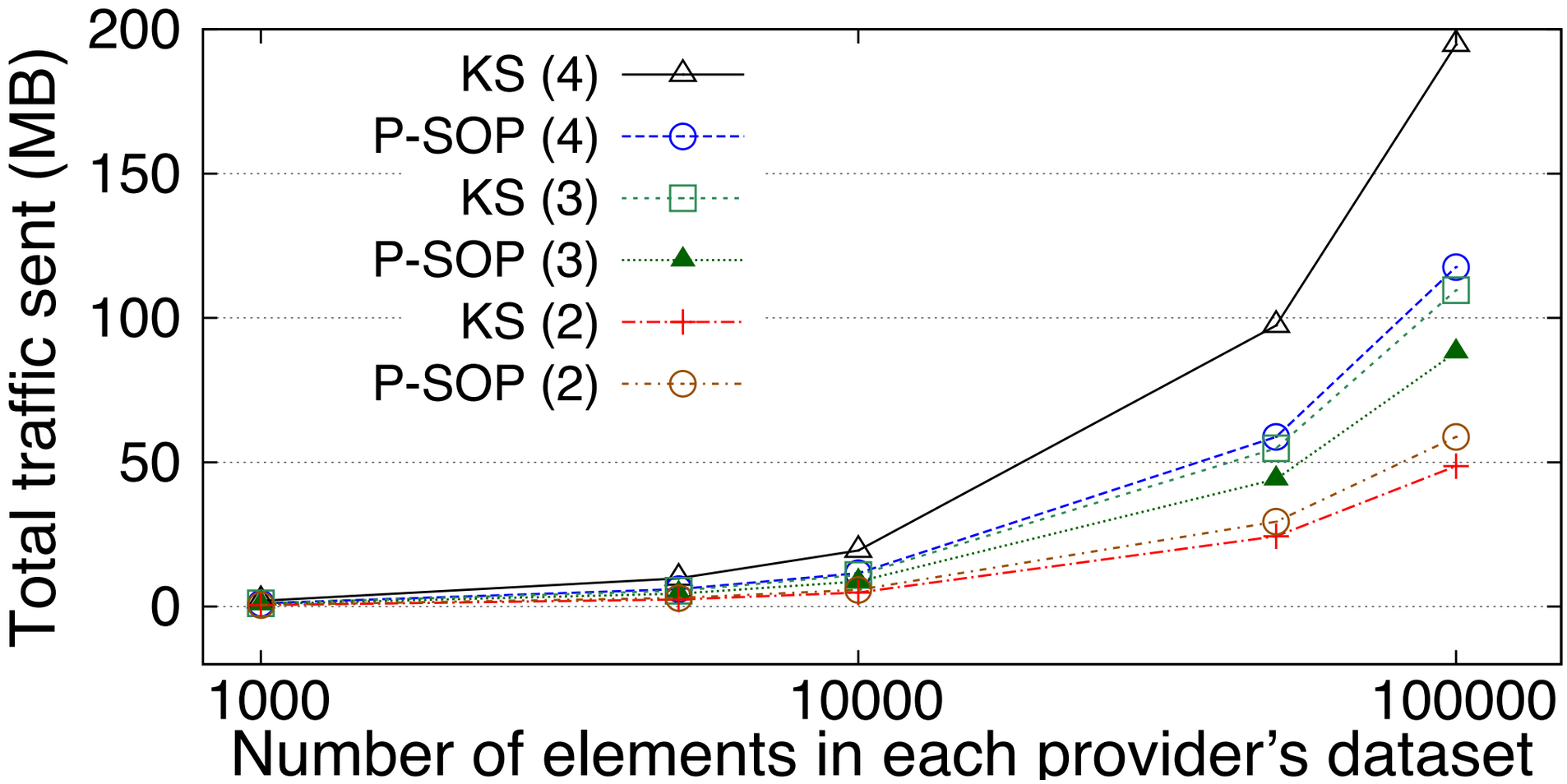
Evaluation

- Three realistic case studies.
- Tradeoff between auditing algorithms
- Overhead of P-SOP protocol

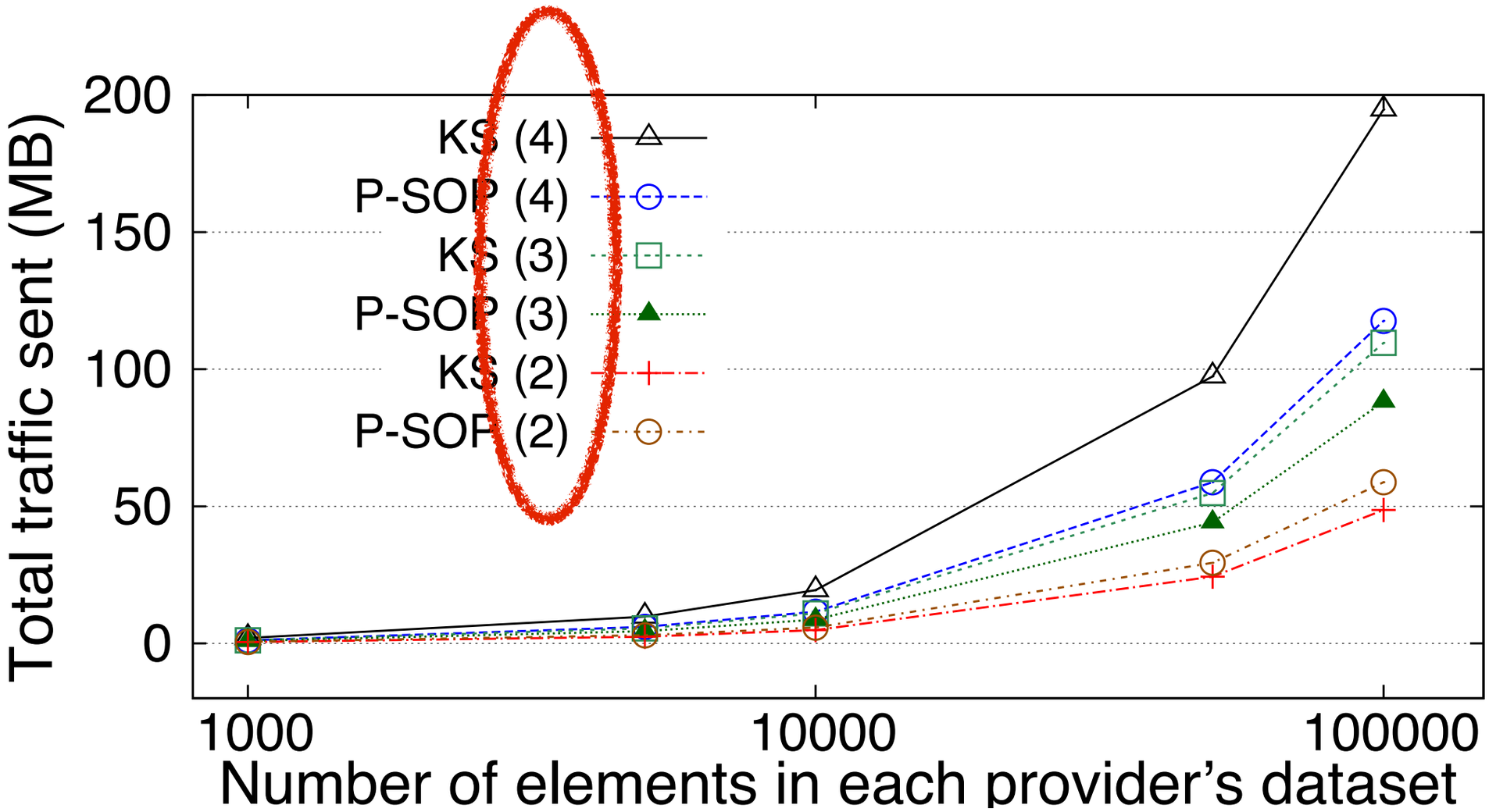
What we evaluate?

- Kissner and Song (KS) protocol for comparison.
- Bandwidth overhead of P-SOP and KS.
- Computational overhead of P-SOP and KS.

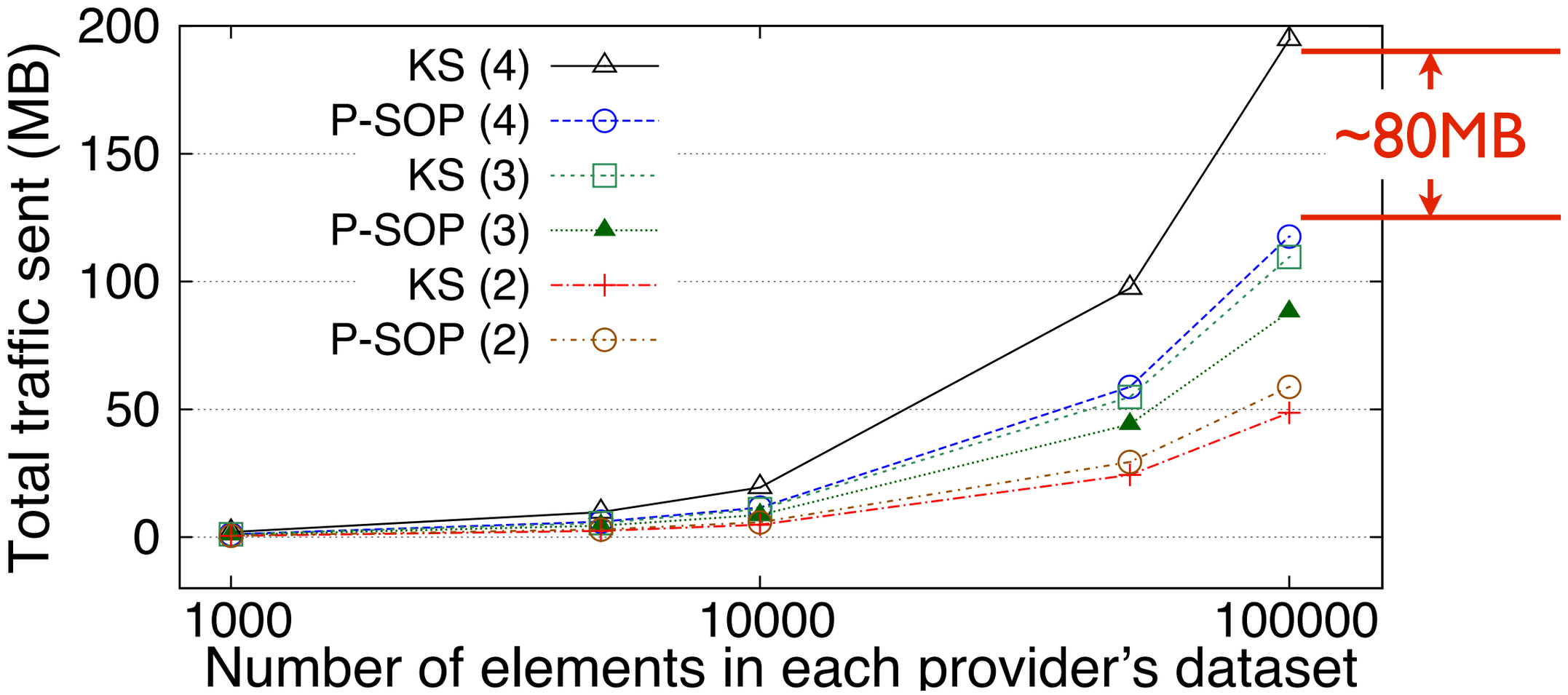
Bandwidth Overhead



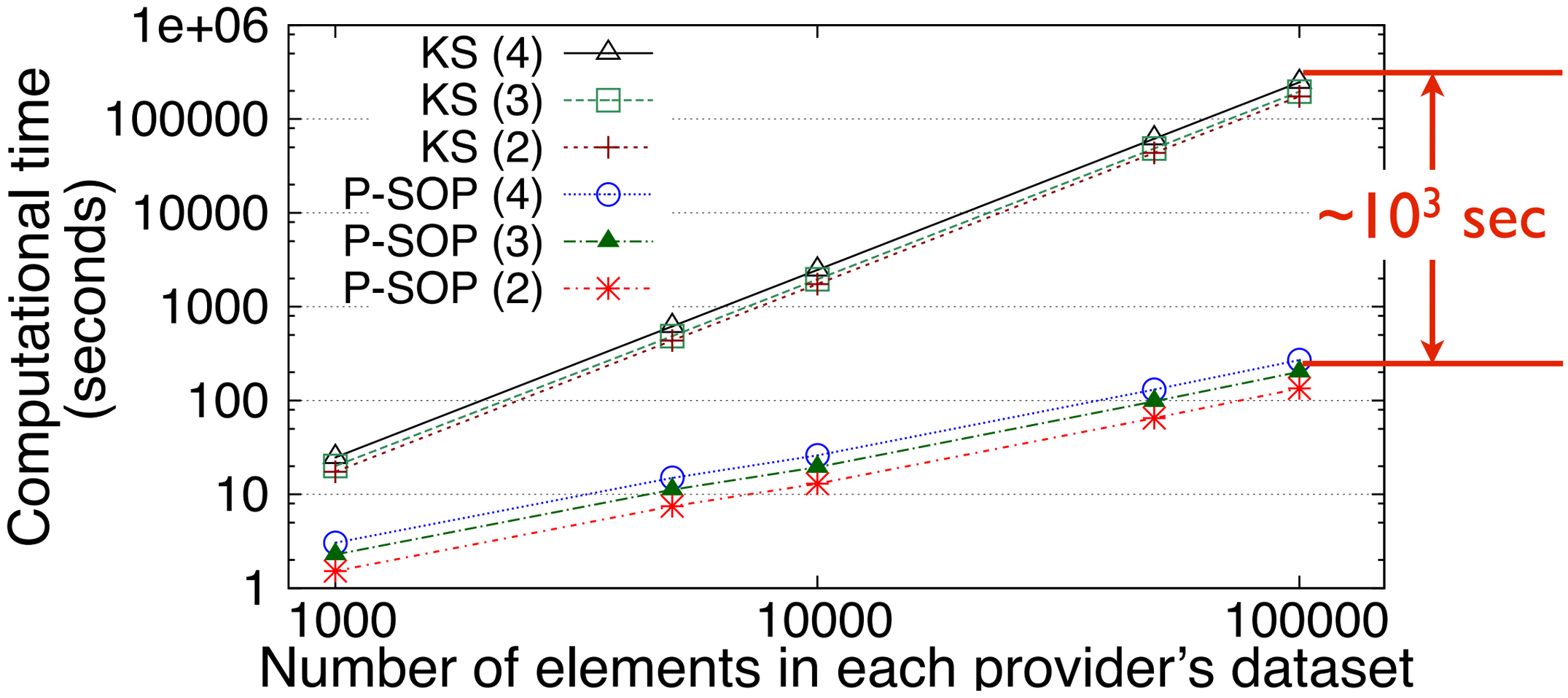
Bandwidth Overhead



Bandwidth Overhead



Computational Overhead



Conclusions

- INDaaS is a first step towards reliable clouds:
  - Dependency collections
  - Dependency representations
  - Efficient auditing
  - Private independence auditing

- We evaluated INDaas with three realistic case studies and large-scale simulations.

Thanks, questions?

- INDaaS: Heading off correlated failures in clouds

- Find out more at:
  - http://dedis.cs.yale.edu/cloud/

- We will be at the poster session tonight.