

deTector:
a Topology-aware Loss Localization System
for Data Center Networks

Peng Yanghua

Losses have significant impact

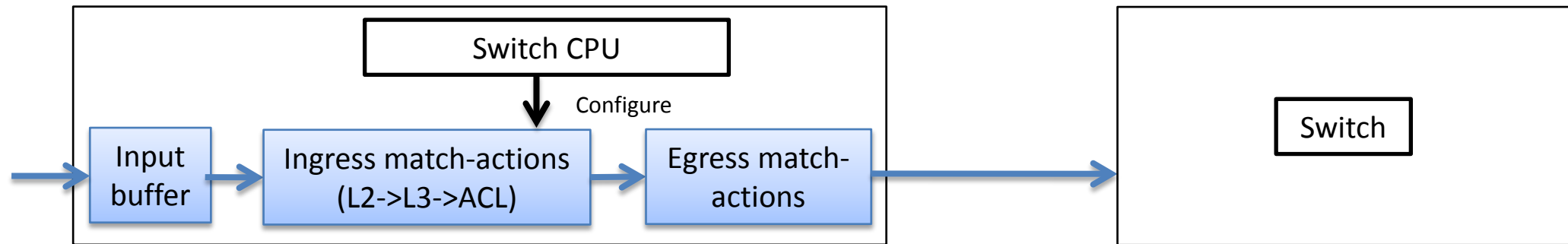
- Significant latency increase and throughput drop
 - Violating SLA and drop revenue
- Takes operators up to tens of hours to find and fix the problem

Losses have significant impact

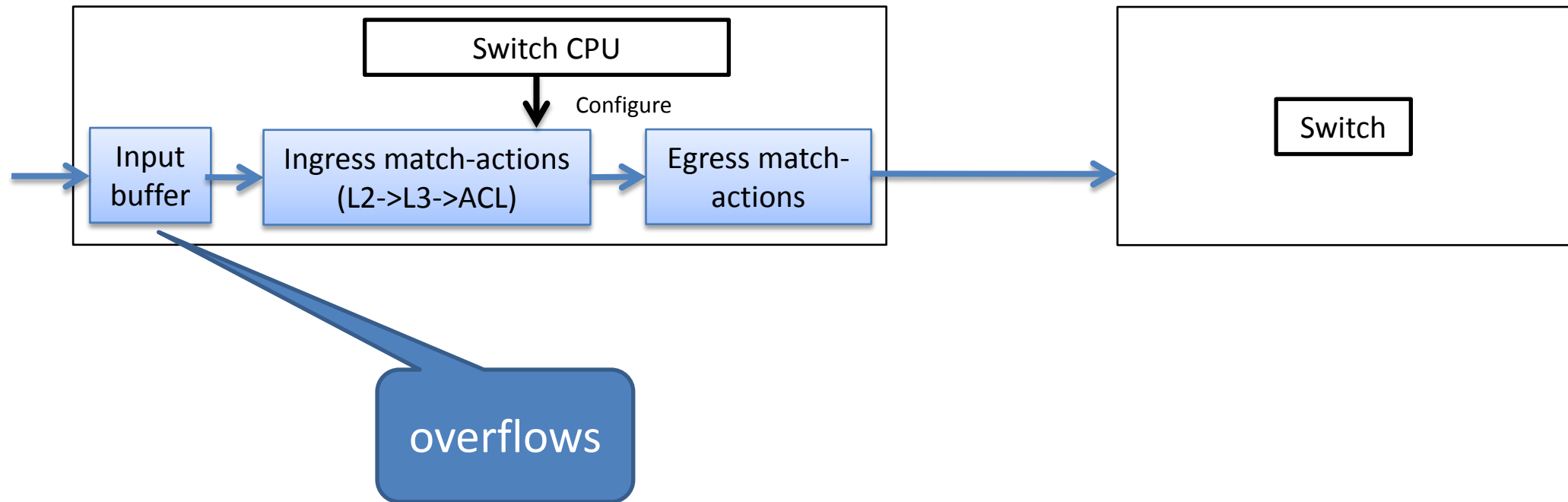
- Significant latency increase and throughput drop
 - Violating SLA and drop revenue
- Takes operators up to tens of hours to find and fix the problem

We want to know loss ASAP

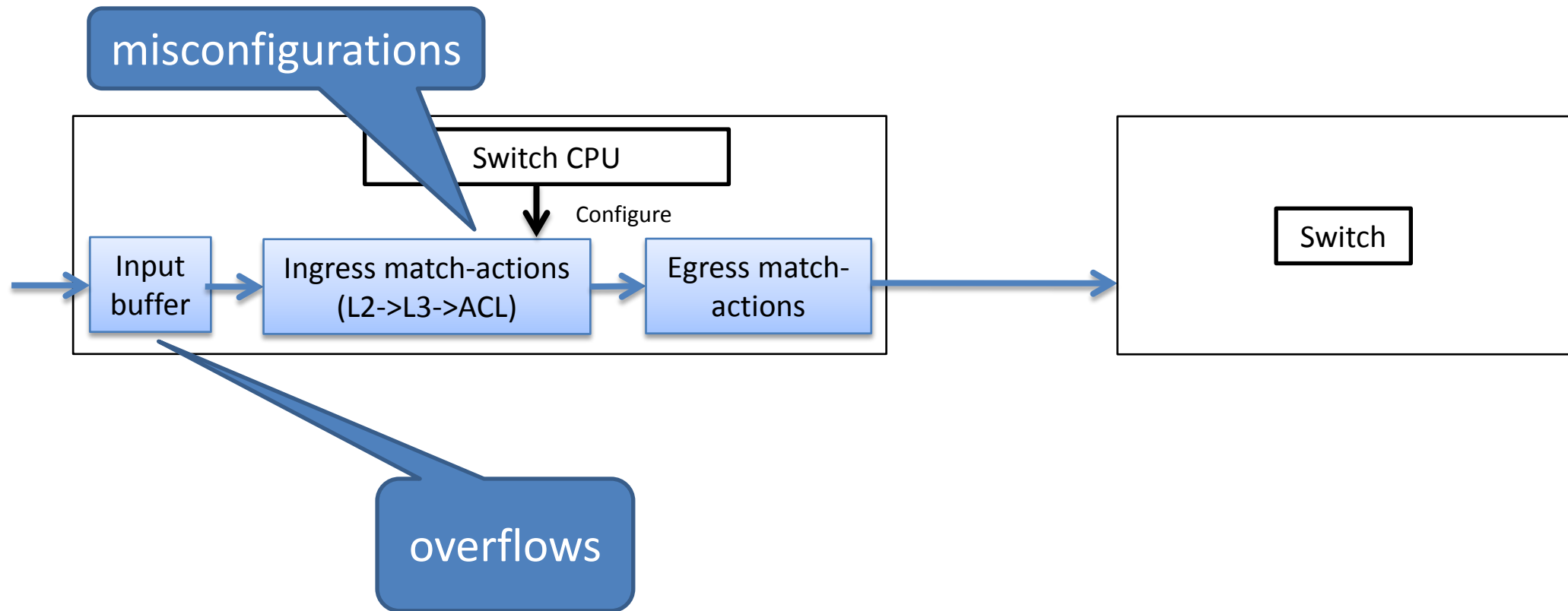
Challenge: many types of losses



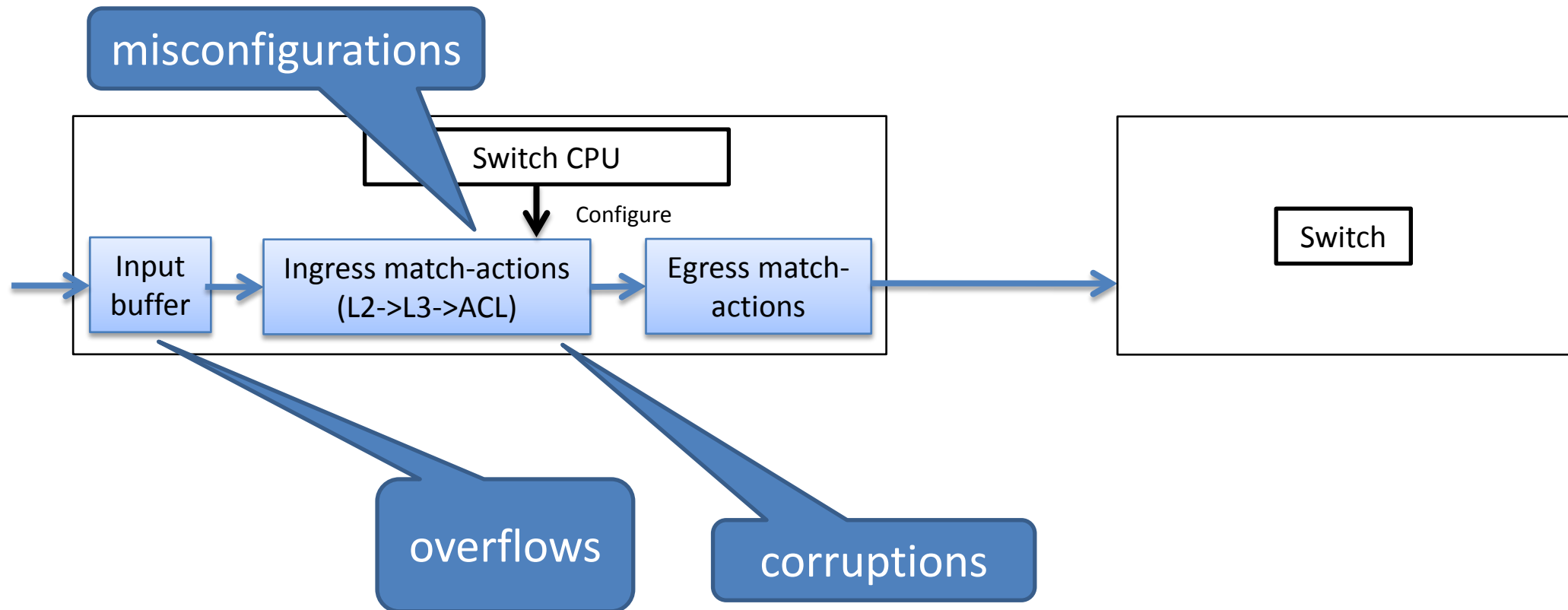
Challenge: many types of losses



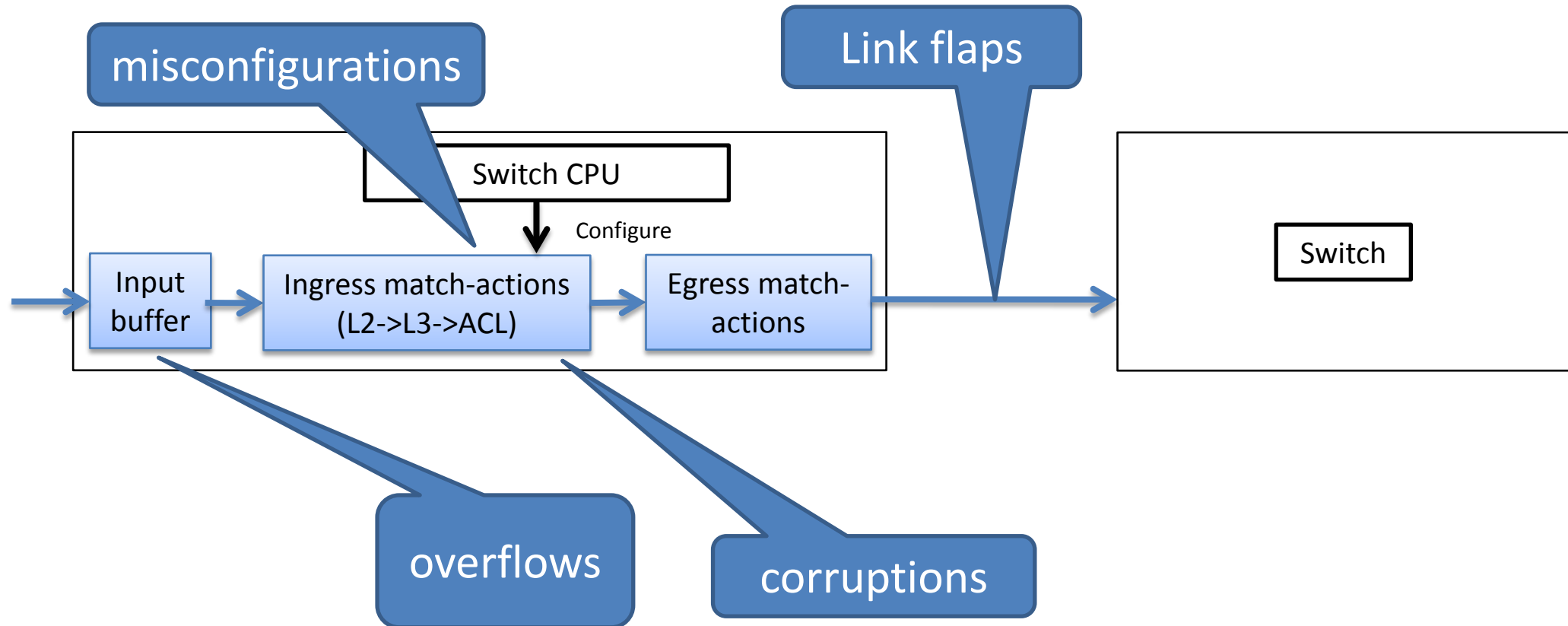
Challenge: many types of losses



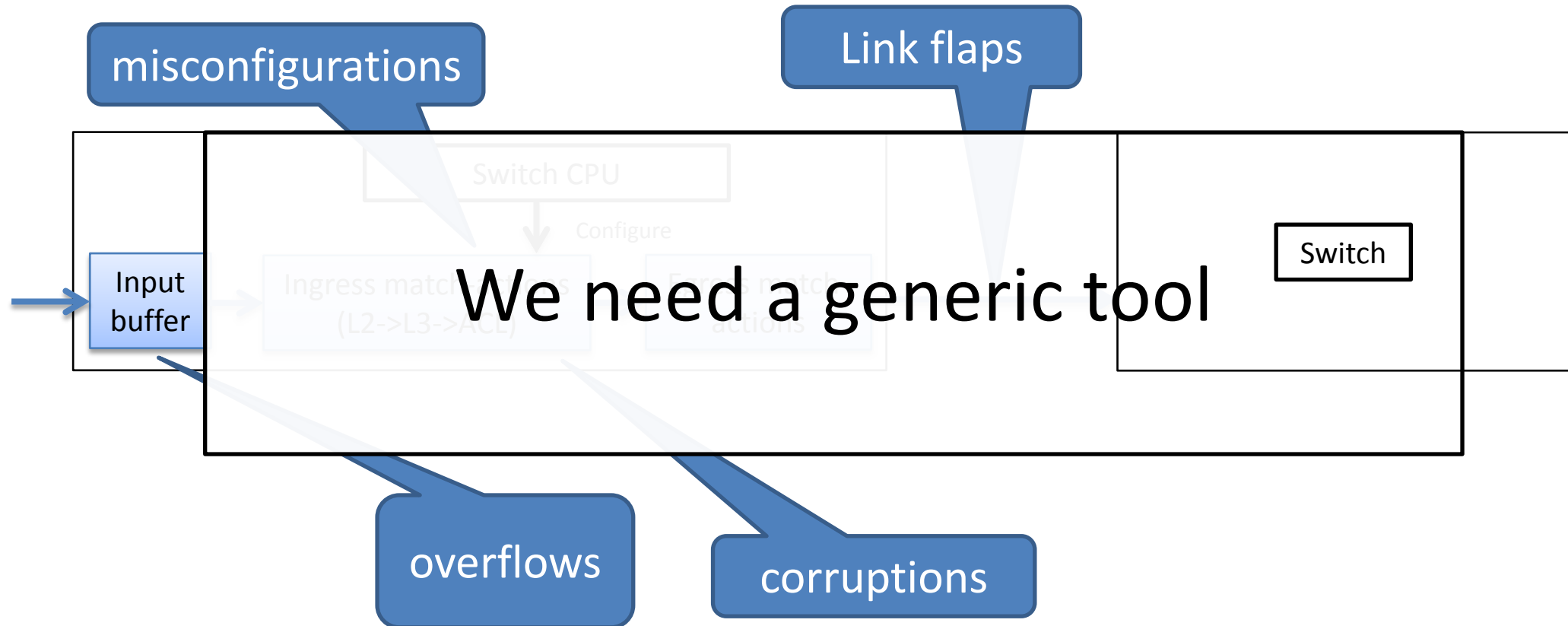
Challenge: many types of losses



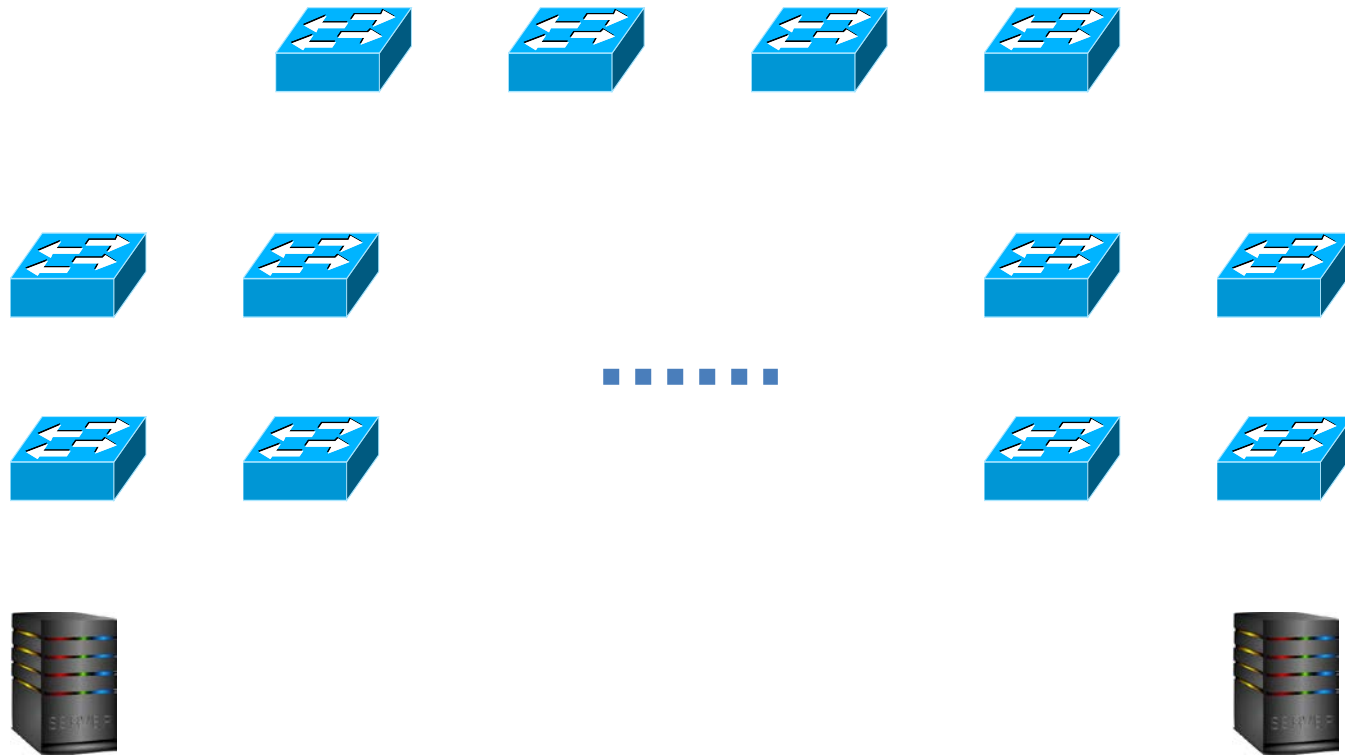
Challenge: many types of losses



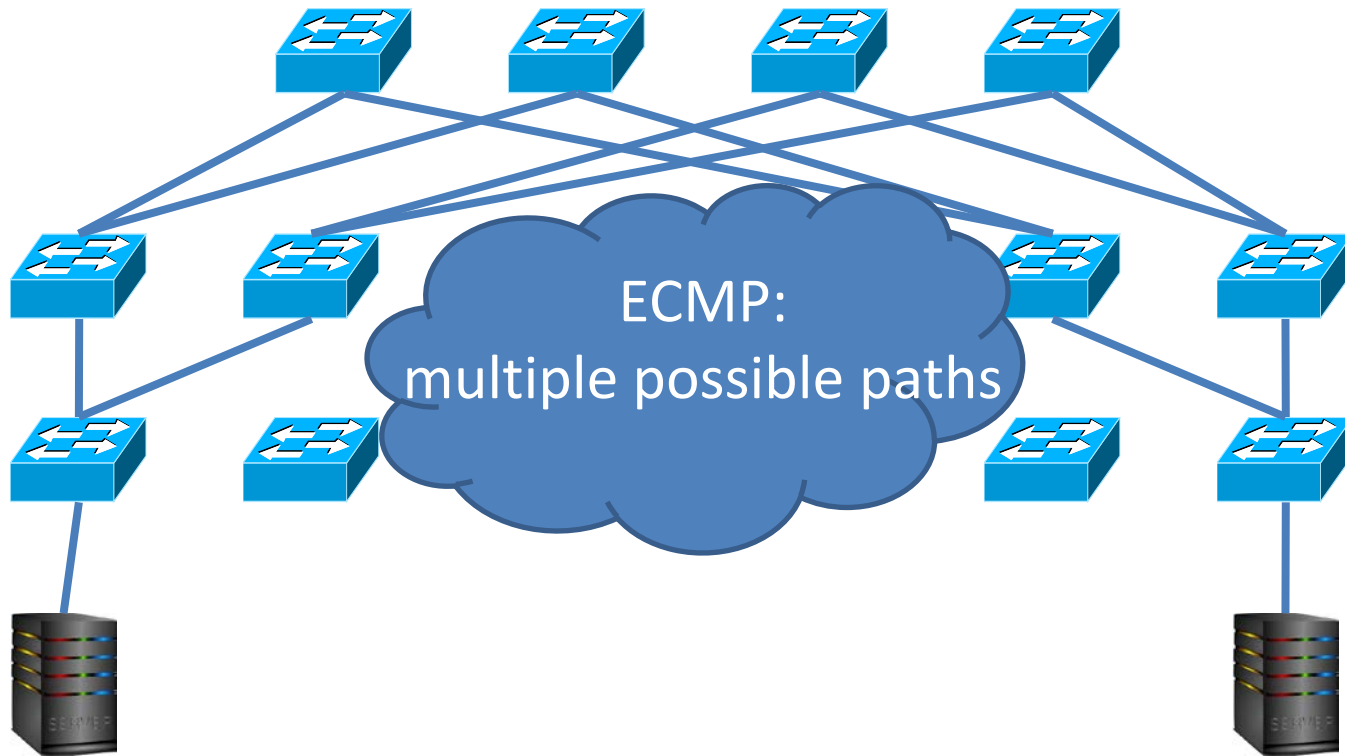
Challenge: many types of losses



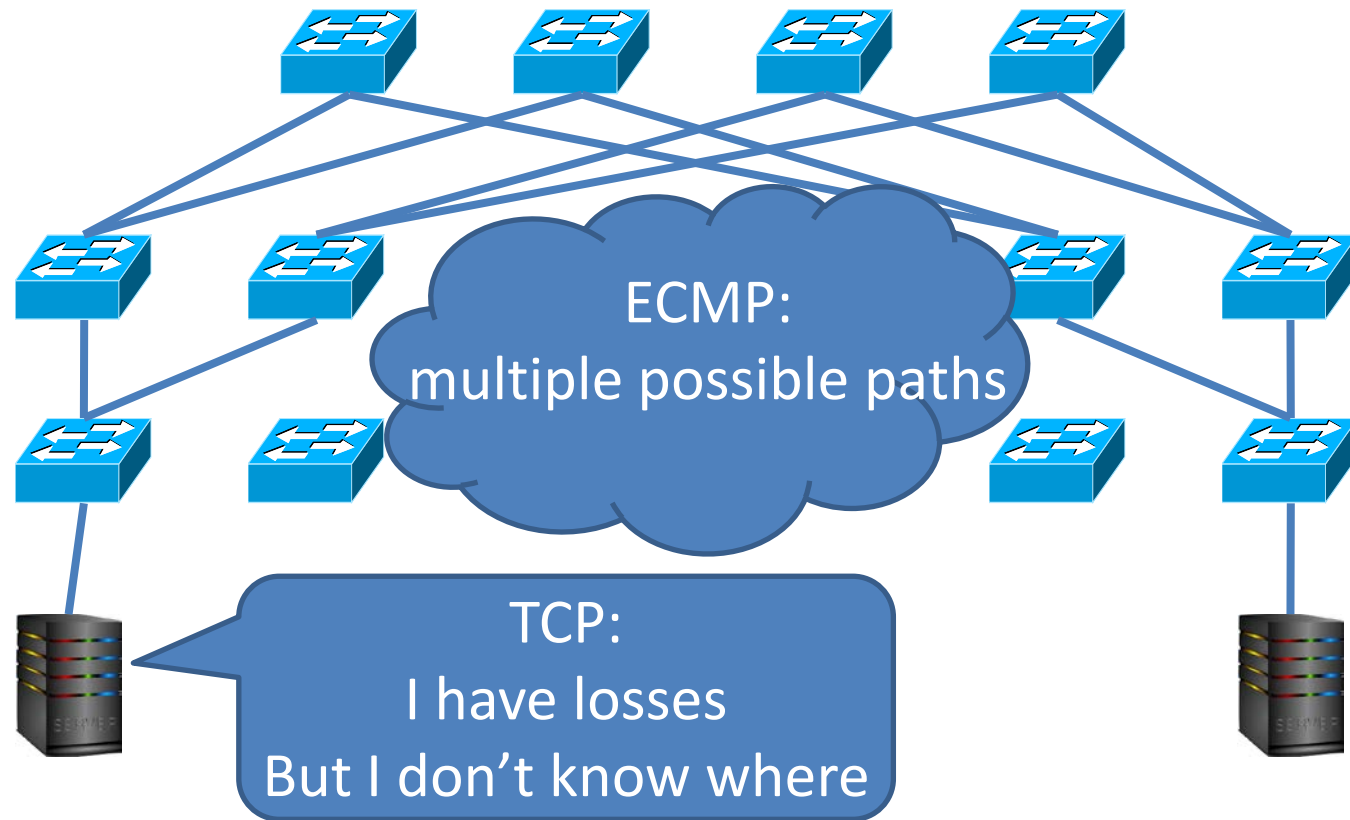
Challenge: losses could happen everywhere



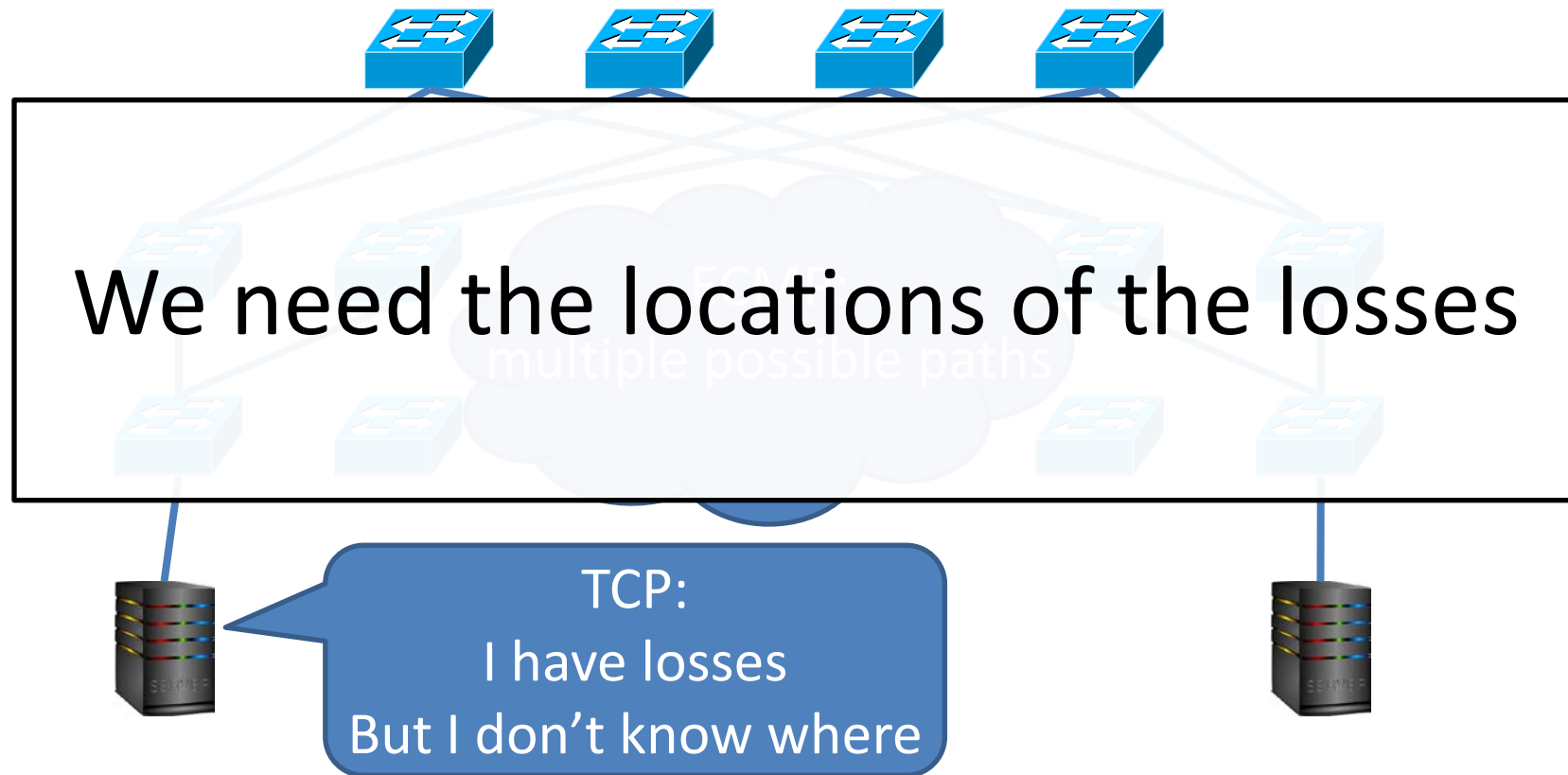
Challenge: losses could happen everywhere



Challenge: losses could happen everywhere



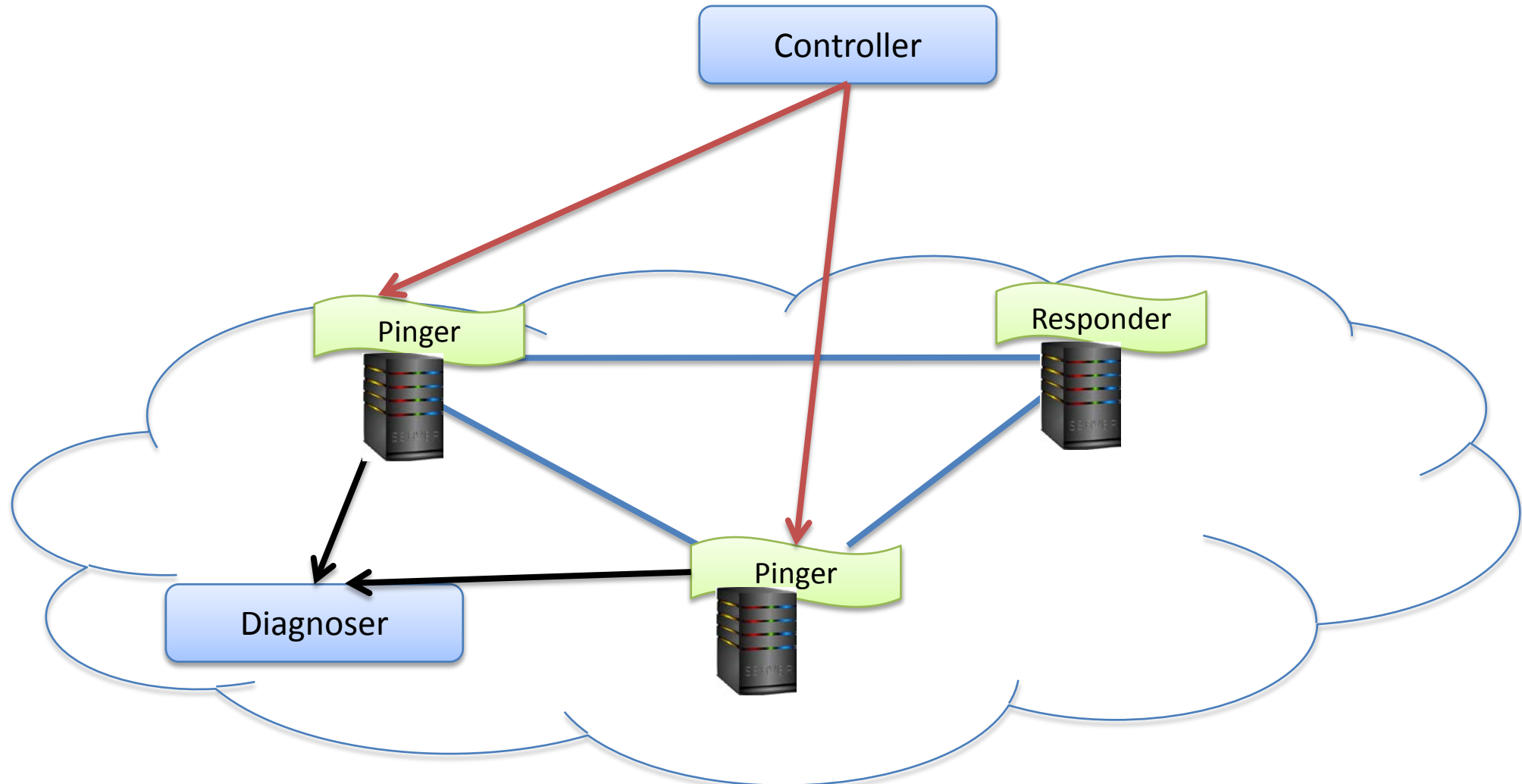
Challenge: losses could happen everywhere



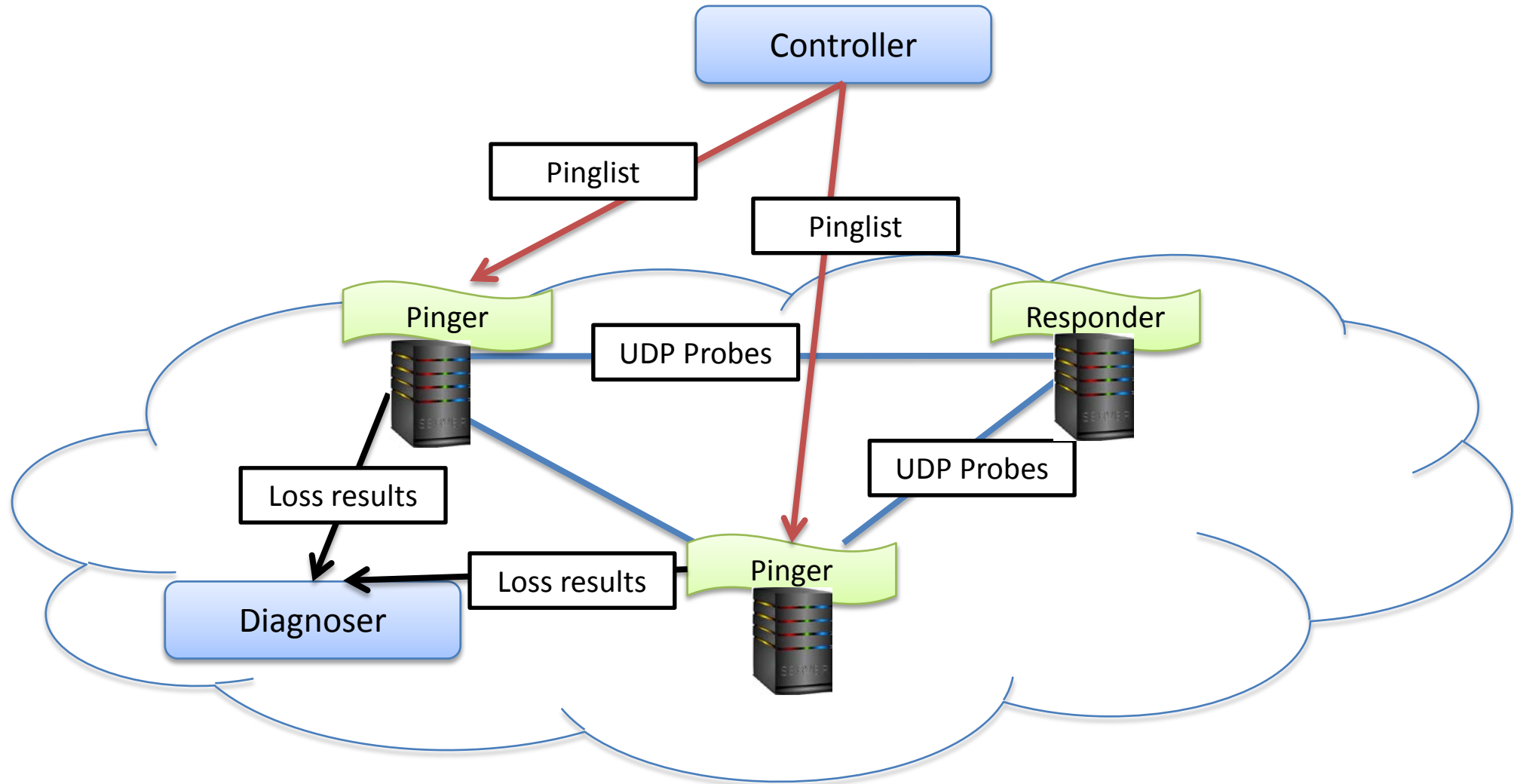
deTector

- ❑ deTector is a real-time, low-overhead and high-accuracy monitoring system for large-scale data center networks.

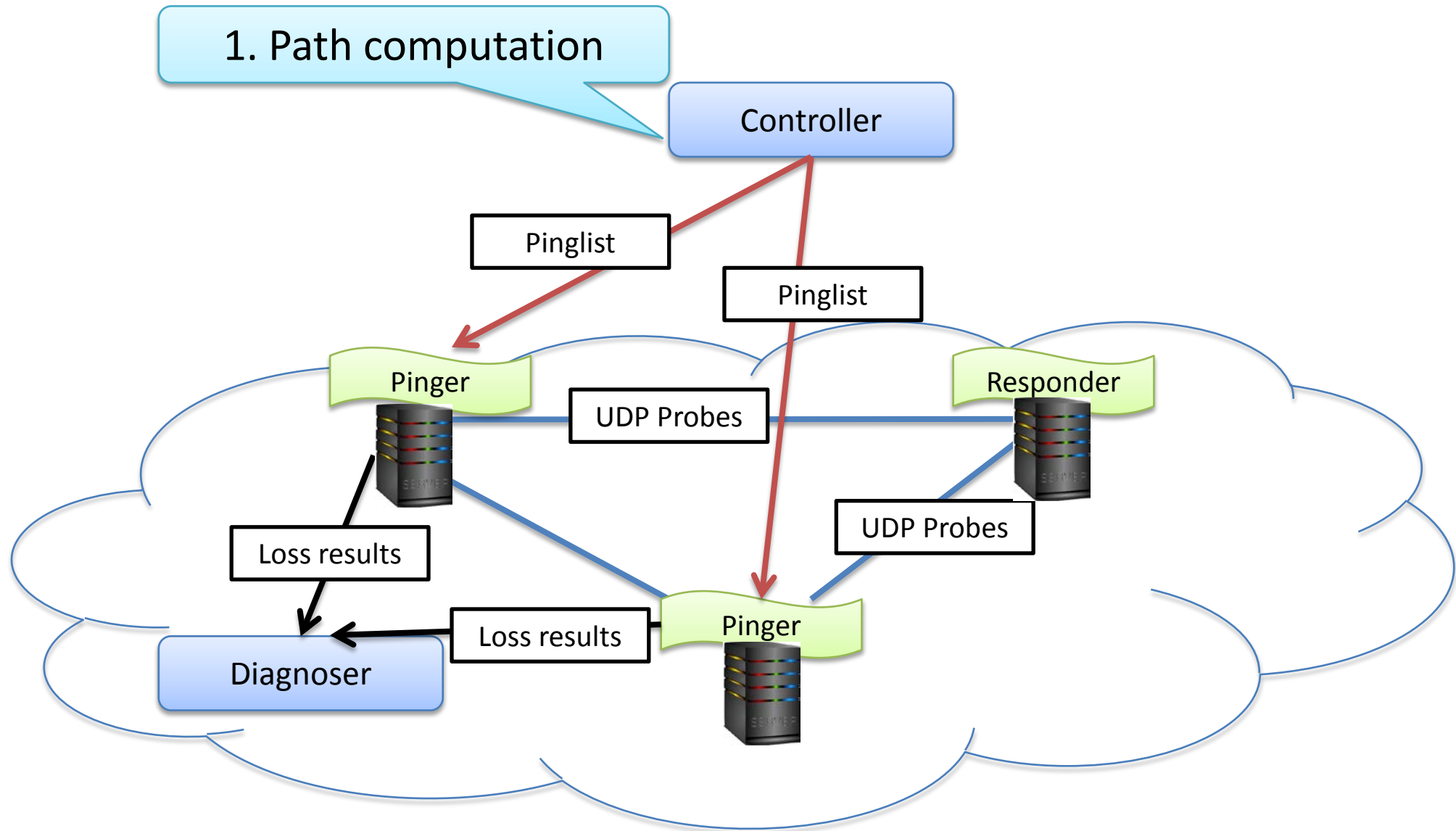
deTector architecture



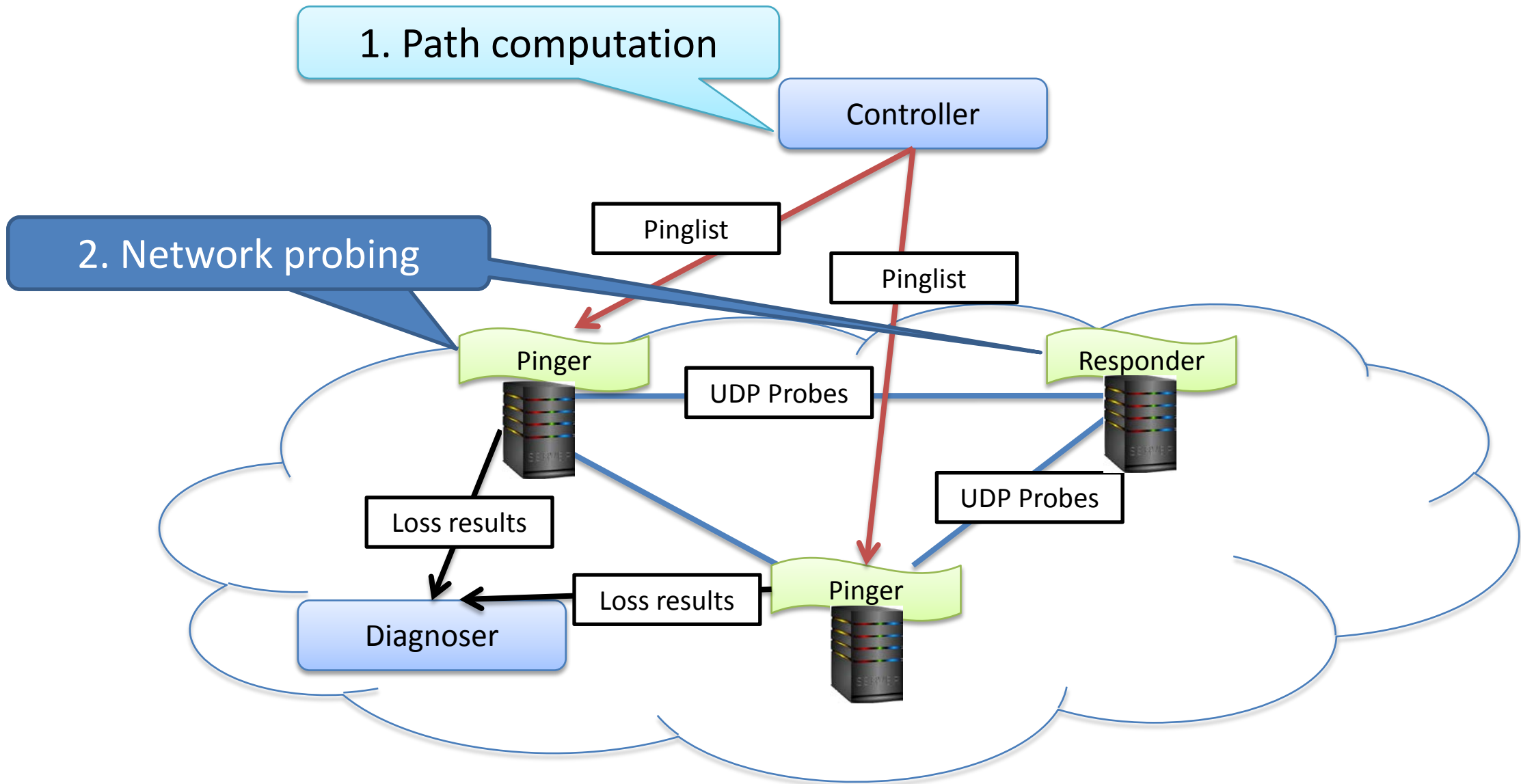
deTector workflow



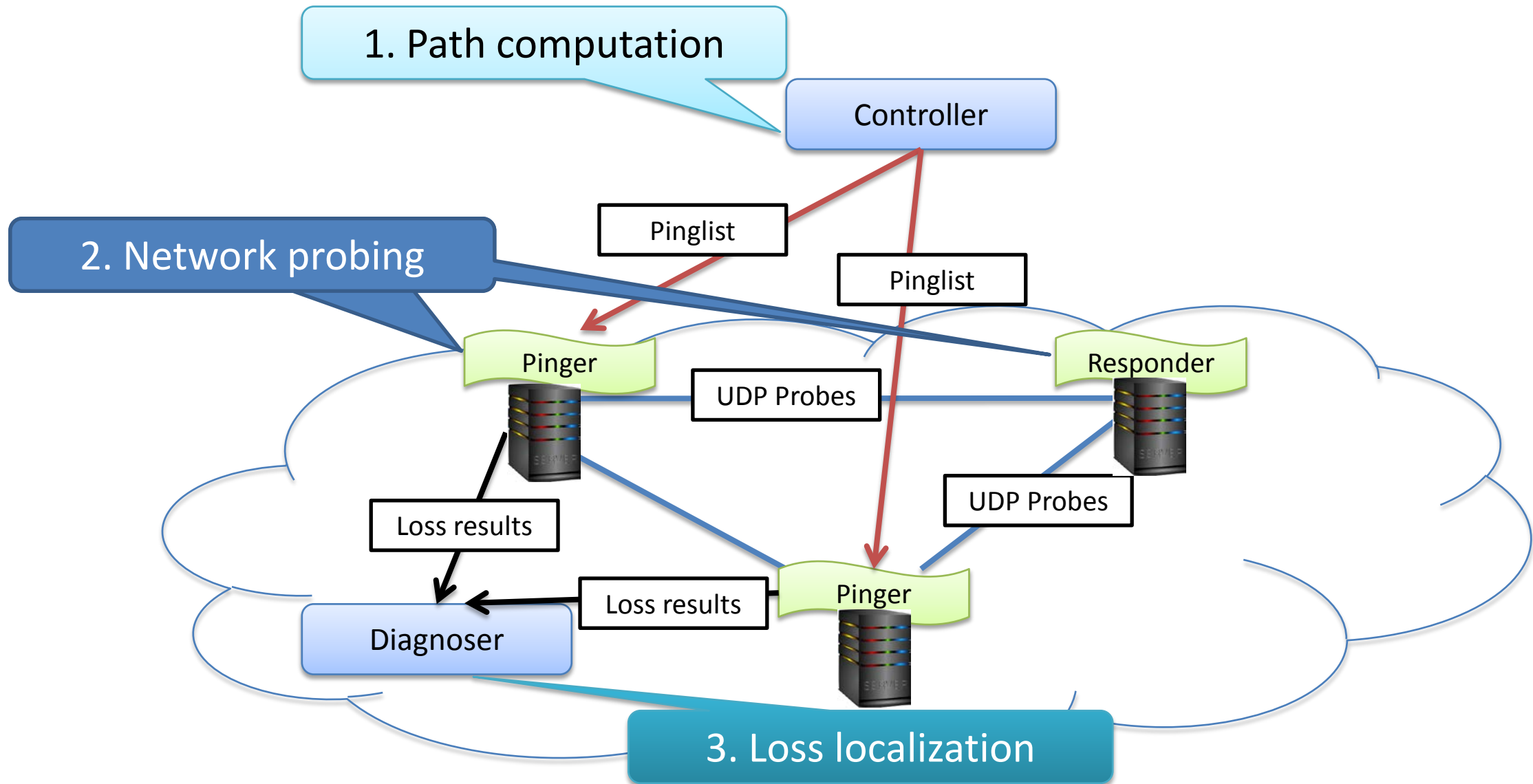
deTector workflow



deTector workflow



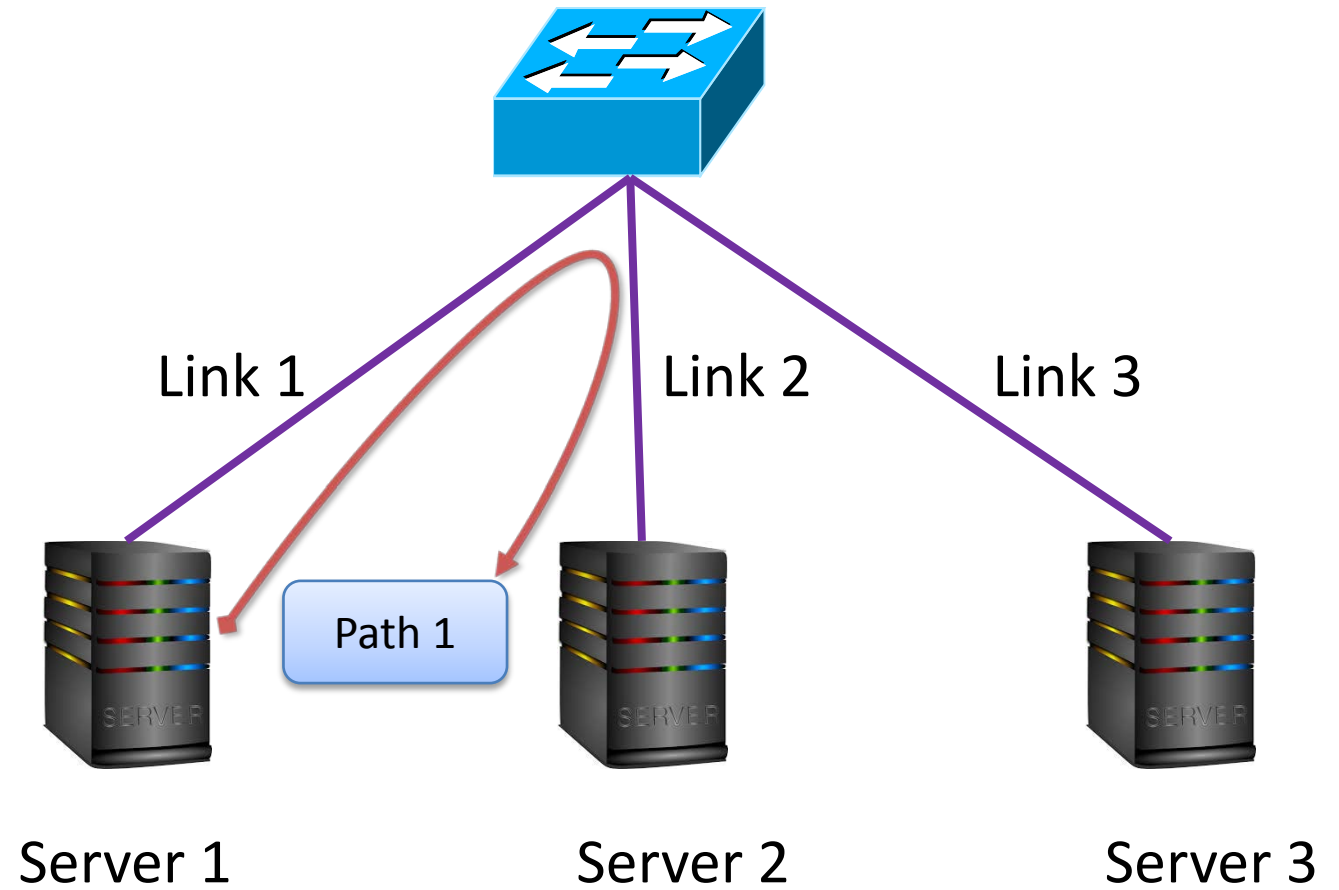
deTector workflow



Part I: Path Computation

Routing matrix

	Link 1	Link 2	Link 3
Path 1	1	1	0
Path 2	1	0	1
Path 3	0	0	1



Problem formulation

	Link 1	Link 2	Link 3
Path 1	1	1	0
Path 2	1	0	1
Path 3	0	0	1

- Given a routing matrix, select probing paths to send probes:

- (1) Minimize path number
- (2) α -coverage
- (3) β -identifiability

Problem formulation

Each link is covered by α paths

	Link 1	Link 2	Link 3
Path 1	1	1	0
Path 2	1	0	1
Path 3	0	0	1

- Given a routing matrix, select probing paths to send probes:

- (1) Minimize path number
- (2) α -coverage
- (3) β -identifiability

Problem formulation

Each link is covered by α paths

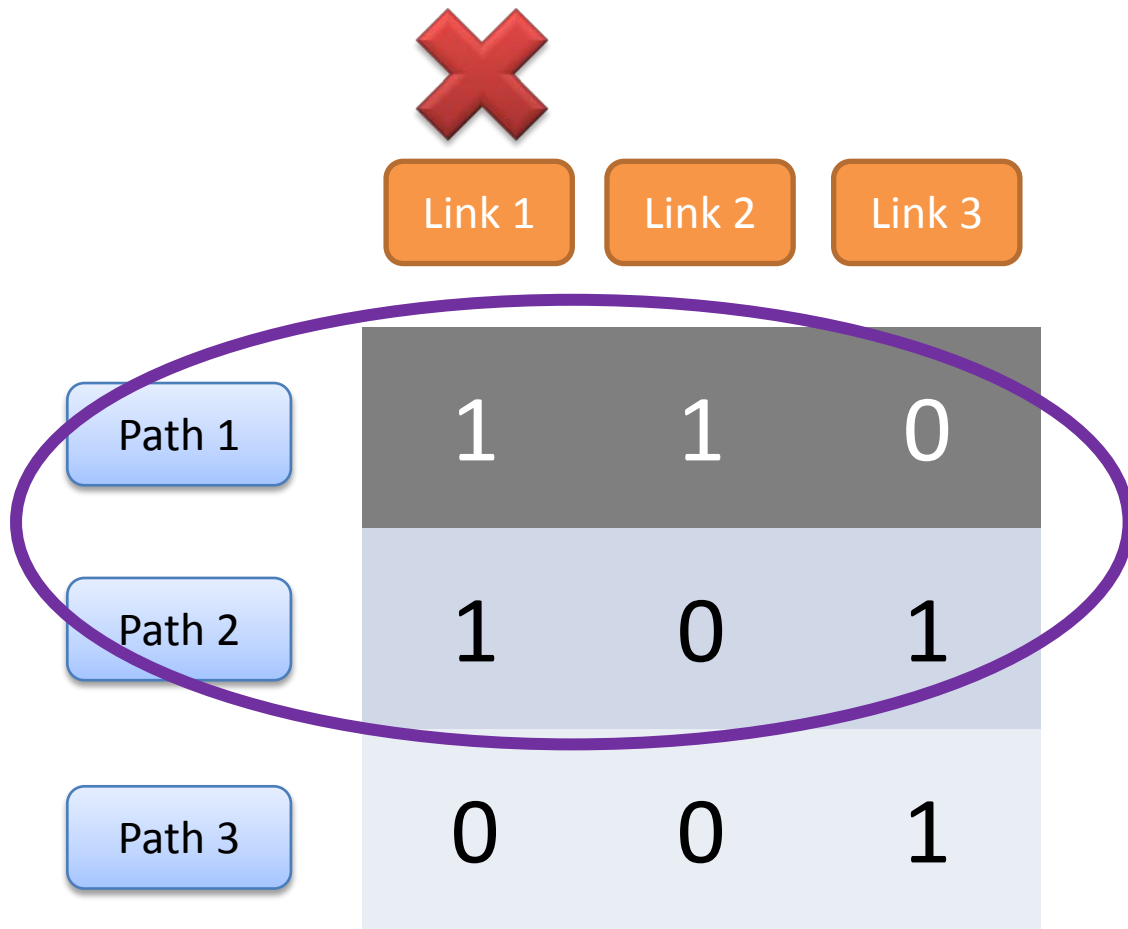
	Link 1	Link 2	Link 3
Path 1	1	1	0
Path 2	1	0	1
Path 3	0	0	1

- Given a routing matrix, select probing paths to send probes:

- (1) Minimize path number
- (2) α -coverage
- (3) β -identifiability

Any β failed links can be identified correctly

An example of β -identifiability



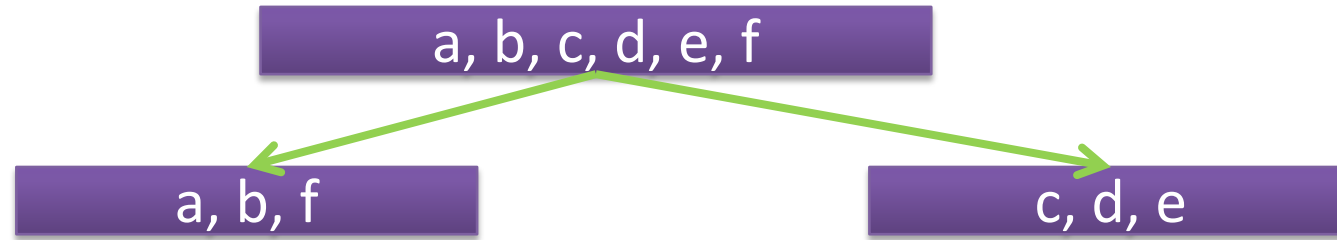
- Any β failed links can be identified correctly
- Probe matrix: path1 + path2
1-identifiability but not 2-identifiability

Algorithm for 1-identifiability

a, b, c, d, e, f

Algorithm for 1-identifiability

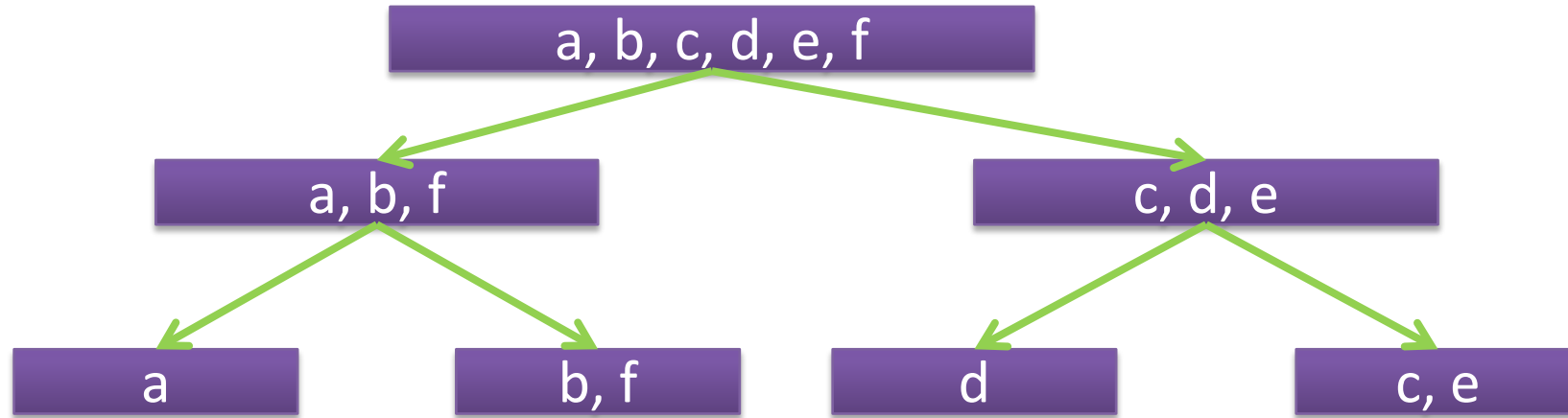
Path1: a->b->f



Algorithm for 1-identifiability

Path1: a->b->f

Path2: a->c->e

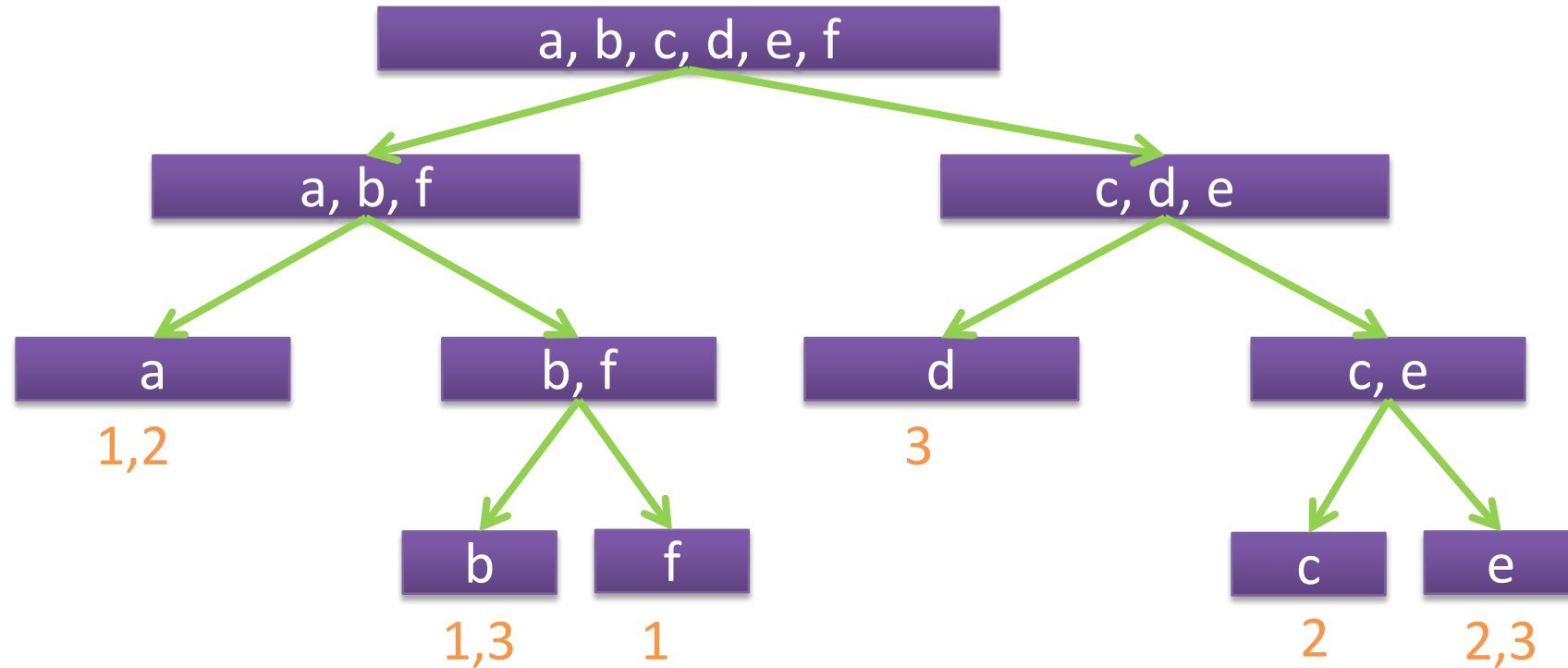


Algorithm for 1-identifiability

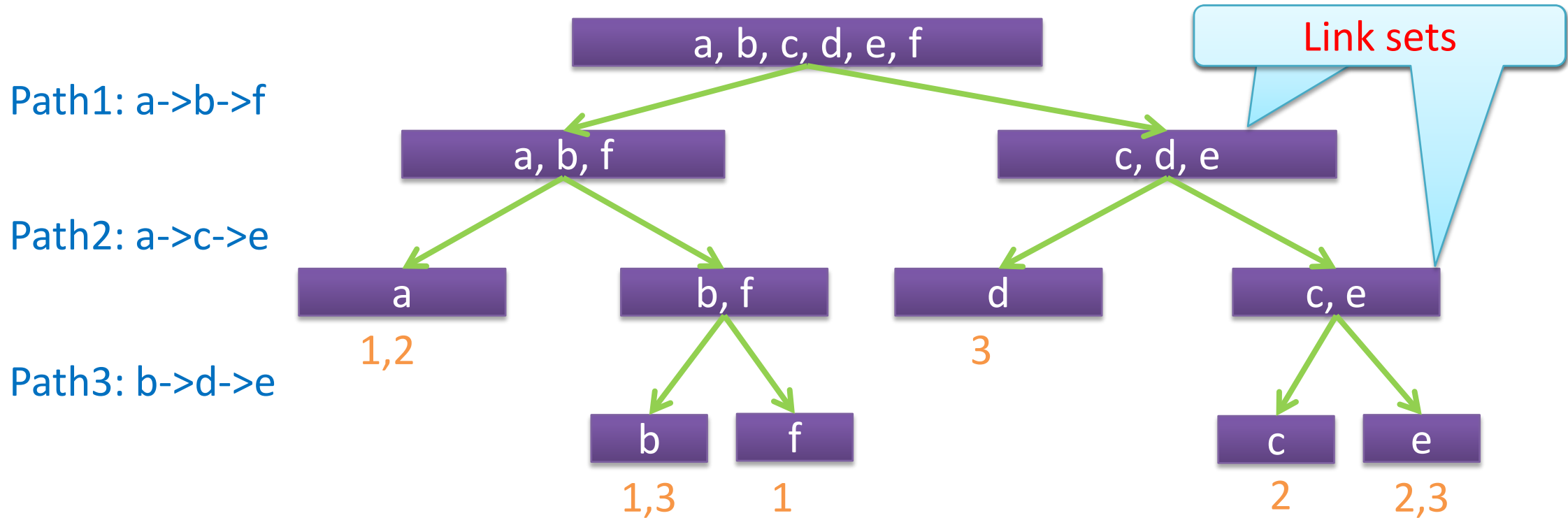
Path1: a->b->f

Path2: a->c->e

Path3: b->d->e

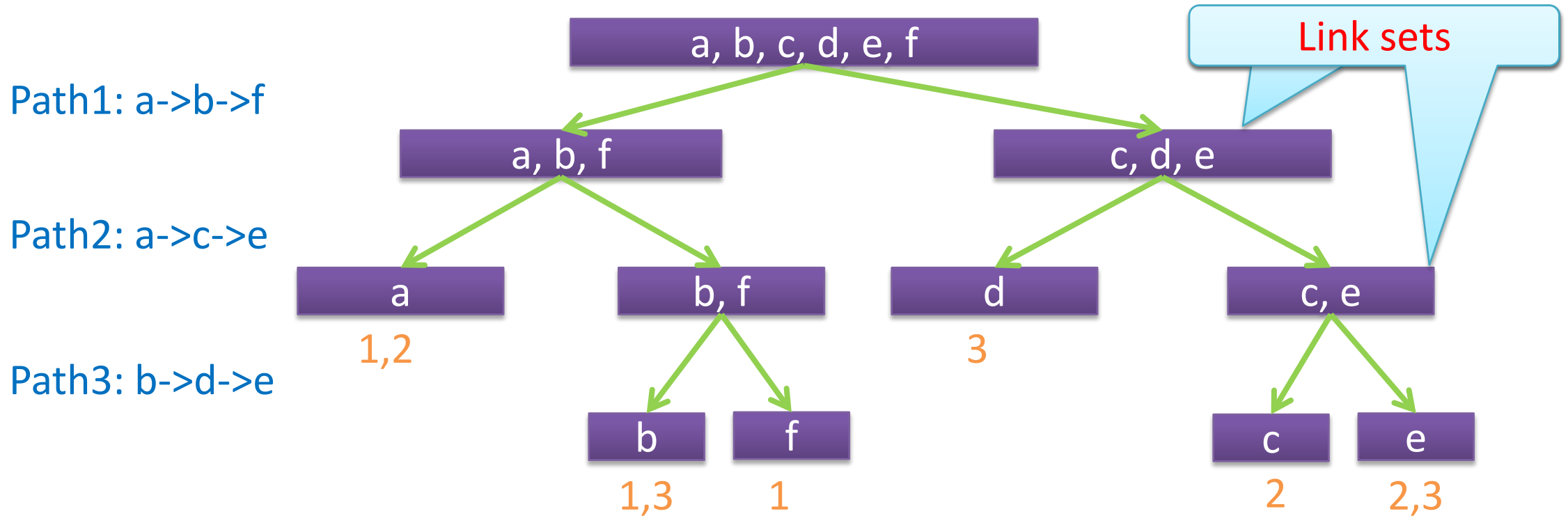


Algorithm for 1-identifiability



- Select the path that is present in the largest number of link sets to split link sets as much as possible.

Algorithm for 1-identifiability



- Select the path that is present in the largest number of link sets to split link sets as much as possible.
- Stops when the total number of link sets equals to the number of links

1-identifiability \Rightarrow β -identifiability

	Link 1	Link 2	Link 3	Link 12	Link 13	Link 23
Path 1	1	1	0	1	1	1
Path 2	1	0	1	1	1	1
Path 3	0	0	1	0	1	1

Physical links OR Virtual links

- Extend routing matrix with **virtual links**.
- If virtual link 12 is bad, we say both link 1 and link 2 have been failed.

α -coverage

	Link 1	Link 2	Link 3
Path 1	1	1	0
Path 2	1	0	1
Path 3	0	0	1

- Assign a counter $w[\text{link}]$ to each link to track the number of paths through it.
- If $w[\text{link}] > \alpha$, then the link has enough paths through it.

PMC algorithm

- Define a score for each path

$$score(path) = \sum_{link \in path} w[link] - \# \text{ of link sets on } path$$

PMC algorithm

- Define a score for each path

$$score(path) = \sum_{link \in path} w[link] - \# \text{ of link sets on } path$$

- Select a path with **minimal** score in each iteration
 - Lower $w[link]$, higher # of link sets \Rightarrow higher priority

PMC algorithm

- Define a score for each path

$$score(path) = \sum_{link \in path} w[link] - \# \text{ of link sets on } path$$

- Select a path with **minimal** score in each iteration
 - Lower $w[link]$, higher # of link sets \Rightarrow higher priority
- Stop when achieving α -coverage and β -identifiability

PMC algorithm

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

- 1: Initialize w , $score$ to $\mathbf{0}$, $setnum$ to 1, $selpaths$ to \emptyset
 - 2: $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$
 - 3: $paths \leftarrow$ all paths in \mathbf{R}' , $physlinks \leftarrow E$
 - 4: **while** ($setnum \neq |E| \parallel physlinks \neq \emptyset$) && $paths \neq \emptyset$ **do**
 - 5: **for** $path \in paths$ **do**
 - 6: update $score[path]$ according to (1)
 - 7: $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$
 - 8: $selpaths \leftarrow selpaths \cup \{path\}$
 - 9: $paths \leftarrow paths / \{path\}$
 - 10: **for** $physlink$ on $path$ **do**
 - 11: $w[physlink] \leftarrow w[physlink] + 1$
 - 12: **if** $w[physlink] > \alpha$ **then**
 - 13: $physlinks \leftarrow physlinks / \{physlink\}$
 - 14: update $setnum$ as the total number of link subsets after split by $path$
 - 15: return probe matrix constructed by paths in $selpaths$ (retaining only physical links on the paths)
-

Extend routing
matrix with virtual
links

PMC algorithm

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

- 1: Initialize w , $score$ to $\mathbf{0}$, $setnum$ to 1, $selpaths$ to \emptyset
 - 2: $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$
 - 3: $paths \leftarrow$ all paths in \mathbf{R}' , $physlinks \leftarrow E$
 - 4: **while** ($setnum \neq |E| \parallel physlinks \neq \emptyset$) && $paths \neq \emptyset$ **do**
 - 5: **for** $path \in paths$ **do**
 - 6: update $score[path]$ according to (1)
 - 7: $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$
 - 8: $selpaths \leftarrow selpaths \cup \{path\}$
 - 9: $paths \leftarrow paths / \{path\}$
 - 10: **for** $physlink$ on $path$ **do**
 - 11: $w[physlink] \leftarrow w[physlink] + 1$
 - 12: **if** $w[physlink] > \alpha$ **then**
 - 13: $physlinks \leftarrow physlinks / \{physlink\}$
 - 14: update $setnum$ as the total number of link subsets after split by $path$
 - 15: return probe matrix constructed by paths in $selpaths$ (retaining only physical links on the paths)
-

Extend routing
matrix with virtual
links

PMC algorithm

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

```
1: Initialize  $w$ ,  $score$  to  $\mathbf{0}$ ,  $setnum$  to 1,  $selpaths$  to  $\emptyset$ 
2:  $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$ 
3:  $paths \leftarrow$  all paths in  $\mathbf{R}'$ ,  $physlinks \leftarrow E$ 
4: while ( $setnum \neq |E| \parallel physlinks \neq \emptyset$ ) &&  $paths \neq \emptyset$  do
5:   for  $path \in paths$  do
6:     update  $score[path]$  according to (1)
7:      $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$ 
8:      $selpaths \leftarrow selpaths \cup \{path\}$ 
9:      $paths \leftarrow paths / \{path\}$ 
10:    for  $physlink$  on  $path$  do
11:       $w[physlink] \leftarrow w[physlink] + 1$ 
12:      if  $w[physlink] > \alpha$  then
13:         $physlinks \leftarrow physlinks / \{physlink\}$ 
14:    update  $setnum$  as the total number of link subsets
    after split by  $path$ 
15: return probe matrix constructed by paths in  $selpaths$ 
    (retaining only physical links on the paths)
```

Update path score

Extend routing
matrix with virtual
links

PMC algorithm

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

```
1: Initialize  $w$ ,  $score$  to  $\mathbf{0}$ ,  $setnum$  to 1,  $selpaths$  to  $\emptyset$ 
2:  $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$ 
3:  $paths \leftarrow$  all paths in  $\mathbf{R}'$ ,  $physlinks \leftarrow E$ 
4: while ( $setnum \neq |E| \parallel physlinks \neq \emptyset$ ) &&  $paths \neq \emptyset$  do
5:   for  $path \in paths$  do
6:     update  $score[path]$  according to (1)
7:      $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$ 
8:      $selpaths \leftarrow selpaths \cup \{path\}$ 
9:      $paths \leftarrow paths / \{path\}$ 
10:    for  $physlink$  on  $path$  do
11:       $w[physlink] \leftarrow w[physlink] + 1$ 
12:      if  $w[physlink] > \alpha$  then
13:         $physlinks \leftarrow physlinks / \{physlink\}$ 
14:    update  $setnum$  as the total number of link subsets
    after split by  $path$ 
15: return probe matrix constructed by paths in  $selpaths$ 
    (retaining only physical links on the paths)
```

Update path score

Select the path
with minimal score

PMC algorithm

Extend routing matrix with virtual links

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

```
1: Initialize  $w$ ,  $score$  to  $\mathbf{0}$ ,  $setnum$  to 1,  $selpaths$  to  $\emptyset$ 
2:  $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$ 
3:  $paths \leftarrow$  all paths in  $\mathbf{R}'$ ,  $physlinks \leftarrow E$ 
4: while ( $setnum \neq |E| \parallel physlinks \neq \emptyset$ ) &&  $paths \neq \emptyset$  do
5:   for  $path \in paths$  do
6:     update  $score[path]$  according to (1)
7:      $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$ 
8:      $selpaths \leftarrow selpaths \cup \{path\}$ 
9:      $paths \leftarrow paths / \{path\}$ 
10:    for  $physlink$  on  $path$  do
11:       $w[physlink] \leftarrow w[physlink] + 1$ 
12:      if  $w[physlink] > \alpha$  then
13:         $physlinks \leftarrow physlinks / \{physlink\}$ 
14:    update  $setnum$  as the total number of link subsets after split by  $path$ 
15: return probe matrix constructed by paths in  $selpaths$  (retaining only physical links on the paths)
```

Update path score

Update $w[link]$ and remove links that achieve α -coverage

Select the path with minimal score

PMC algorithm

Extend routing matrix with virtual links

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

```
1: Initialize  $w$ ,  $score$  to  $\mathbf{0}$ ,  $setnum$  to 1,  $selpaths$  to  $\emptyset$ 
2:  $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$ 
3:  $paths \leftarrow$  all paths in  $\mathbf{R}'$ ,  $physlinks \leftarrow E$ 
4: while ( $setnum \neq |E| \parallel physlinks \neq \emptyset$ ) &&  $paths \neq \emptyset$  do
5:   for  $path \in paths$  do
6:     update  $score[path]$  according to (1)
7:      $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$ 
8:      $selpaths \leftarrow selpaths \cup \{path\}$ 
9:      $paths \leftarrow paths / \{path\}$ 
10:    for  $physlink$  on  $path$  do
11:       $w[physlink] \leftarrow w[physlink] + 1$ 
12:      if  $w[physlink] > \alpha$  then
13:         $physlinks \leftarrow physlinks / \{physlink\}$ 
14:      update  $setnum$  as the total number of link subsets
        after split by  $path$ 
15: return probe matrix constructed by paths in  $selpaths$ 
    (retaining only physical links on the paths)
```

Update $w[link]$ and remove links that achieve α -coverage

Update path score

Select the path with minimal score

Update the total number of link sets

PMC algorithm

Extend routing matrix with virtual links

Algorithm 1 PMC: Probe Matrix Construction Algorithm

Require: $\mathbf{R}, \alpha, \beta$

```
1: Initialize  $w$ ,  $score$  to  $\mathbf{0}$ ,  $setnum$  to 1,  $selpaths$  to  $\emptyset$ 
2:  $\mathbf{R}' \leftarrow LINKOR(\mathbf{R}, \beta)$ 
3:  $paths \leftarrow$  all paths in  $\mathbf{R}'$ ,  $physlinks \leftarrow E$ 
4: while ( $setnum \neq |E| \parallel physlinks \neq \emptyset$ ) &&  $paths \neq \emptyset$  do
5:   for  $path \in paths$  do
6:     update  $score[path]$  according to (1)
7:      $path \leftarrow \operatorname{argmin}_{path' \in paths} score[path']$ 
8:      $selpaths \leftarrow selpaths \cup \{path\}$ 
9:      $paths \leftarrow paths / \{path\}$ 
10:    for  $physlink$  on  $path$  do
11:       $w[physlink] \leftarrow w[physlink] + 1$ 
12:      if  $w[physlink] > \alpha$  then
13:         $physlinks \leftarrow physlinks / \{physlink\}$ 
14:      update  $setnum$  as the total number of link subsets
        after split by  $path$ 
15: return probe matrix constructed by paths in  $selpaths$ 
    (retaining only physical links on the paths)
```

Update $w[link]$ and remove links that achieve α -coverage

Stop

Update path score

Select the path with minimal score

Update the total number of link sets

PMC algorithm

- Achieves 63% approximation ratio.
- Time complexity $O(n^2)$ where n is the number of paths.
- A Fattree(64) DCN has more than 2^{32} paths, running time > 24 hours

Four optimizations for speedup

- Routing matrix decomposition
- Partial score update
- Lazy update
- Symmetry reduction

Routing matrix decomposition

1	1	0	0	0
1	0	1	0	0
0	1	1	0	0
0	0	0	1	0
0	0	0	1	1



1	1	0
1	0	1
0	1	1

1	0
1	1

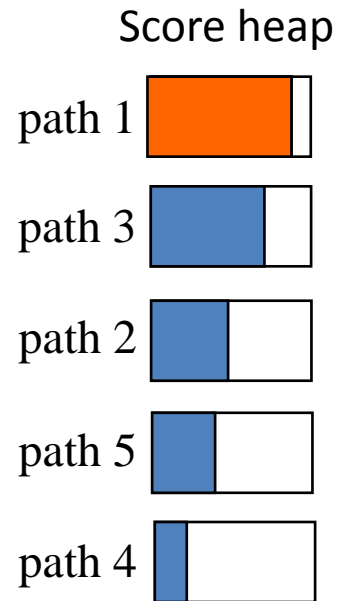
Partial update

	Link 1	Link 2	Link 3
Path 1	1	1	0
Path 2	1	0	1
Path 3	0	0	1

- After selecting path1, we only need to update the score of path2 since path3 shares no links with path1.

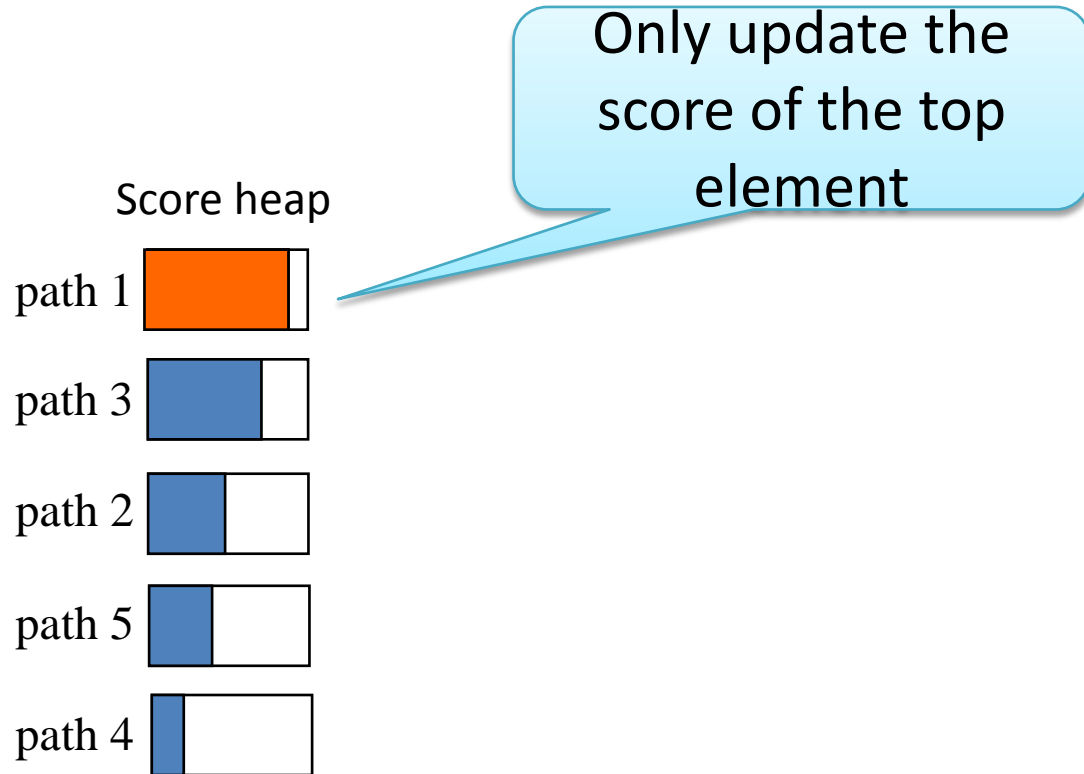
Lazy update

- Defer the score update of a path as much as possible until we have to.

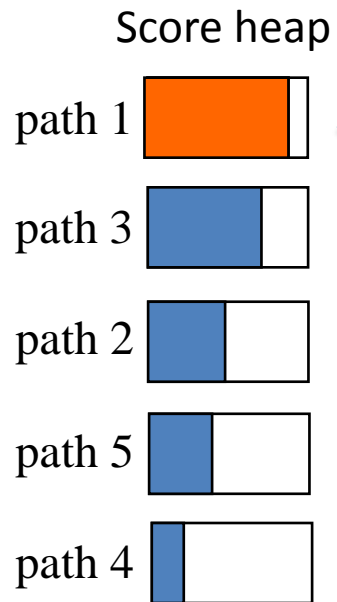


Lazy update

- Defer the score update of a path as much as possible until we have to.



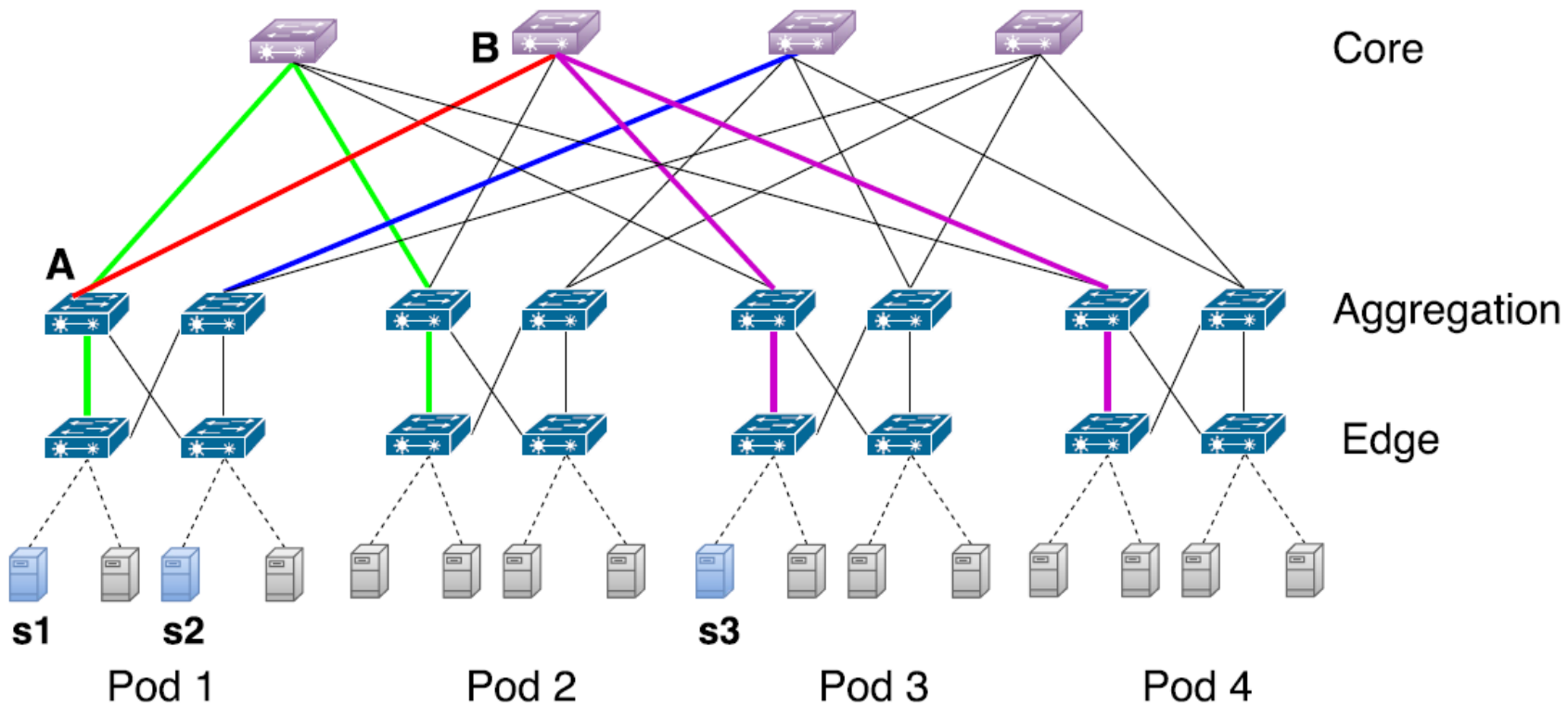
Lazy update



Only update the
score of the top
element

- Defer the score update of a path as much as possible until we have to.
- Correctness guaranteed by the submodularity of the objective function.

Symmetry reduction



- DCN topology is symmetric and thus we only need to compute paths for a small symmetry component.

PMC results

- Running time

DCNs	# of nodes	# of links	# of original paths	Strawman	Decomposition	Partial update	Lazy update	Symmetry
Fattree(12)	612	1296	184,032	231.458	5.216	1.509	0.506	0.126
Fattree(24)	4,176	10,368	11,902,464	> 24h	1381.226	99.778	23.254	0.280
Fattree(72)	99,792	279,936	8,703,770,112	> 24h	> 24h	> 24h	> 24h	17.054

PMC results

- Running time

DCNs	# of nodes	# of links	# of original paths	Strawman	Decomposition	Partial update	Lazy update	Symmetry
Fattree(12)	612	1296	184,032	231.458	5.216	1.509	0.506	0.126
Fattree(24)	4,176	10,368	11,902,464	> 24h	1381.226	99.778	23.254	0.280
Fattree(72)	99,792	279,936	8,703,770,112	> 24h	> 24h	> 24h	> 24h	17.054

- The number of selected paths

DCNs	# of original paths	# of selected paths with (α, β)		
		(1, 0)	(1, 1)	(3, 2)
Fattree(32)	66,977,792	4,096	7,680	12,288
Fattree(64)	4,292,870,144	32768	61,440	98,304

Part II: Network Probing

Network probing

- How to route probes: source routing to control paths
- How pingers send probes: frequency, QoS, source port
- How responders reply probes: timestamp

Part III: Loss Localization

Problem formulation

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	4
Path 2	1	0	1	1
Path 3	0	0	1	3

Problem formulation

4 probes are lost on path1

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	4
Path 2	1	0	1	1
Path 3	0	0	1	3

Problem formulation

4 probes are lost on path1

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	4
Path 2	1	0	1	1
Path 3	0	0	1	3

- Given the probe matrix and loss measurements, select the least number of links to explain the observation.
- NP-hard

PLL algorithm

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	4
Path 2	1	0	1	1
Path 3	0	0	1	2

PLL algorithm

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	4
Path 2	1	0	1	1
Path 3	0	0	1	2

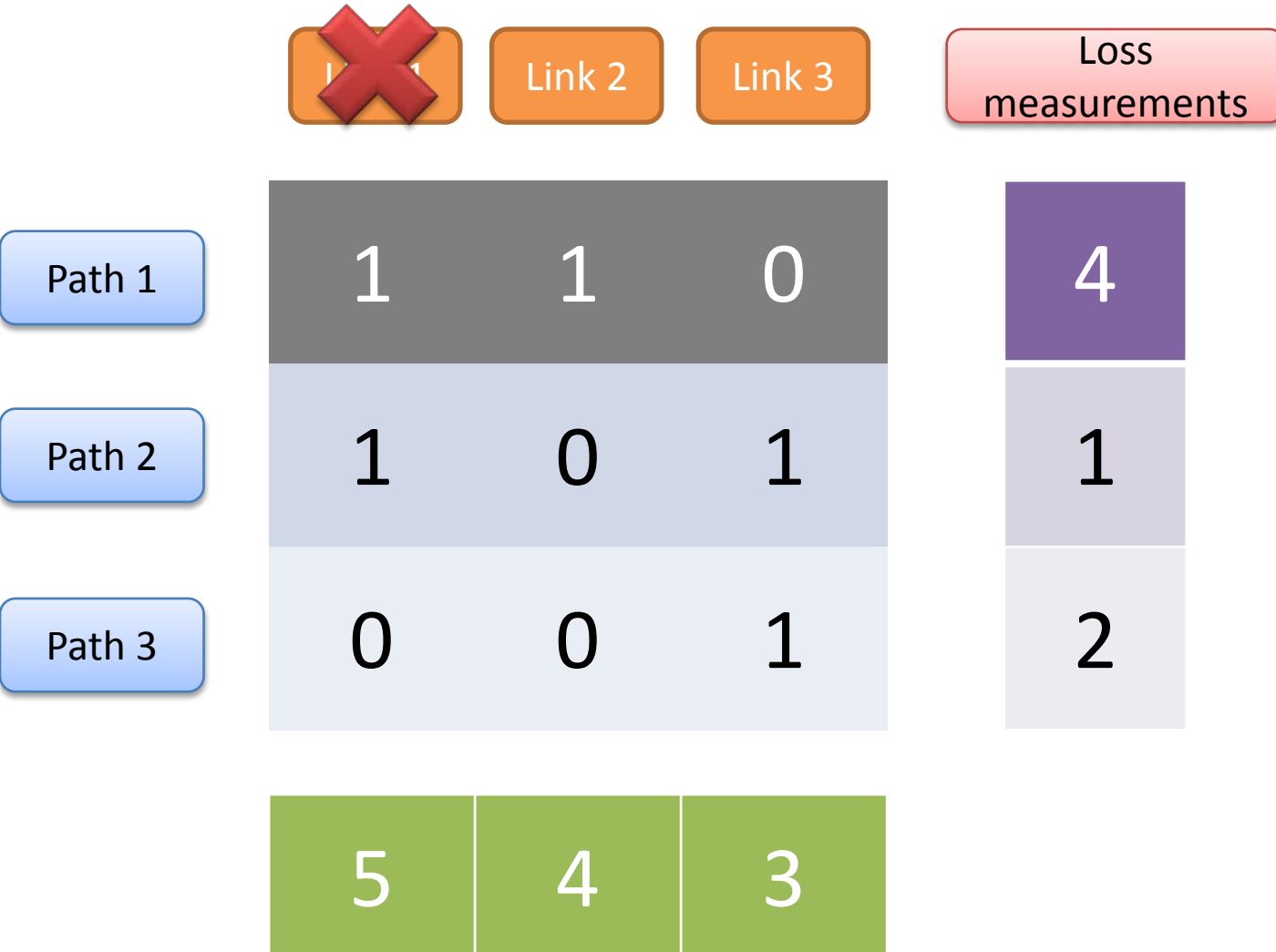
- In each iteration we select a link that can explain the largest number of probe losses until all are explained

PLL algorithm

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	4
Path 2	1	0	1	1
Path 3	0	0	1	2
				5 4 3

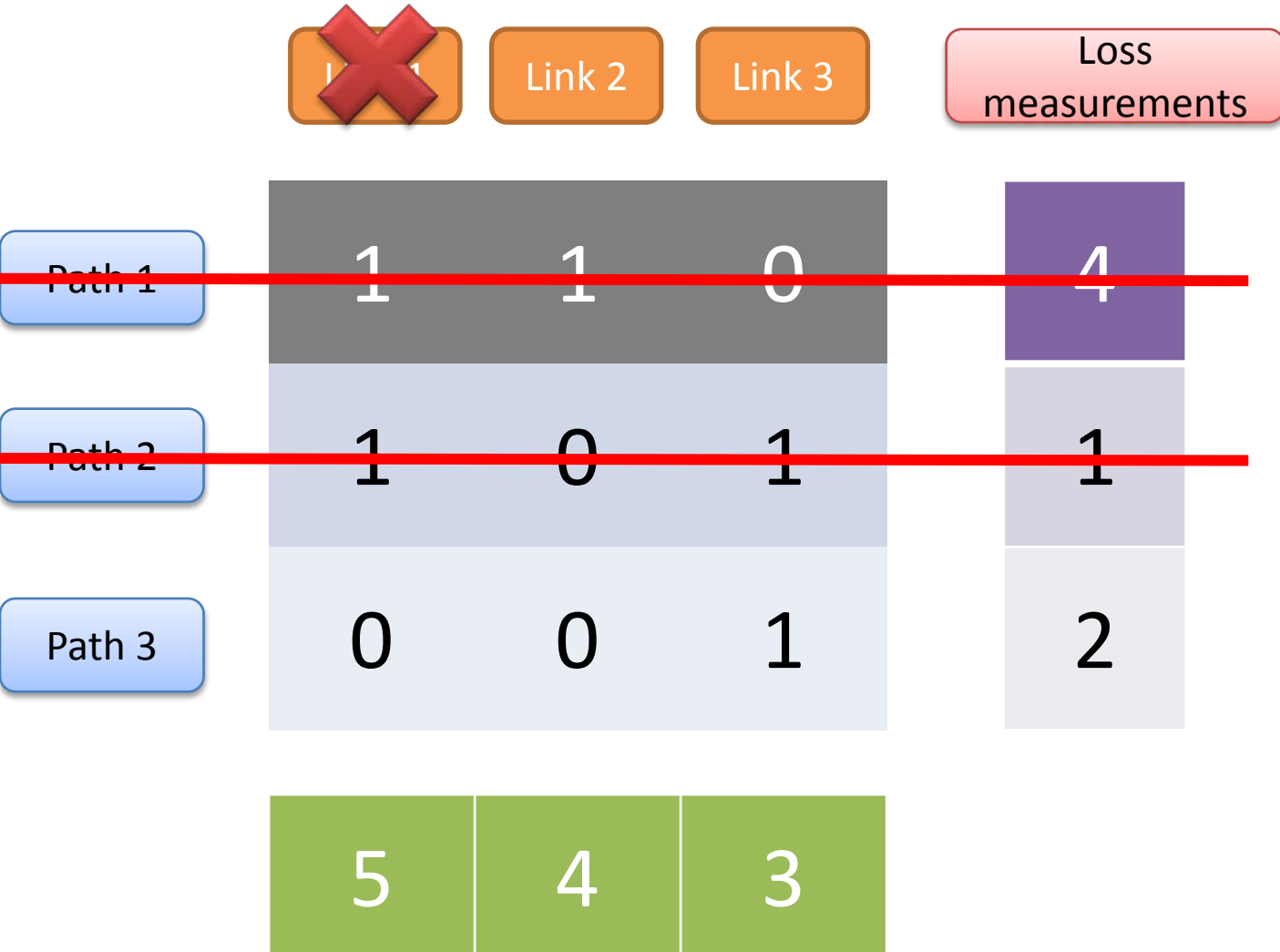
- In each iteration we select a link that can explain the largest number of probe losses until all are explained

PLL algorithm



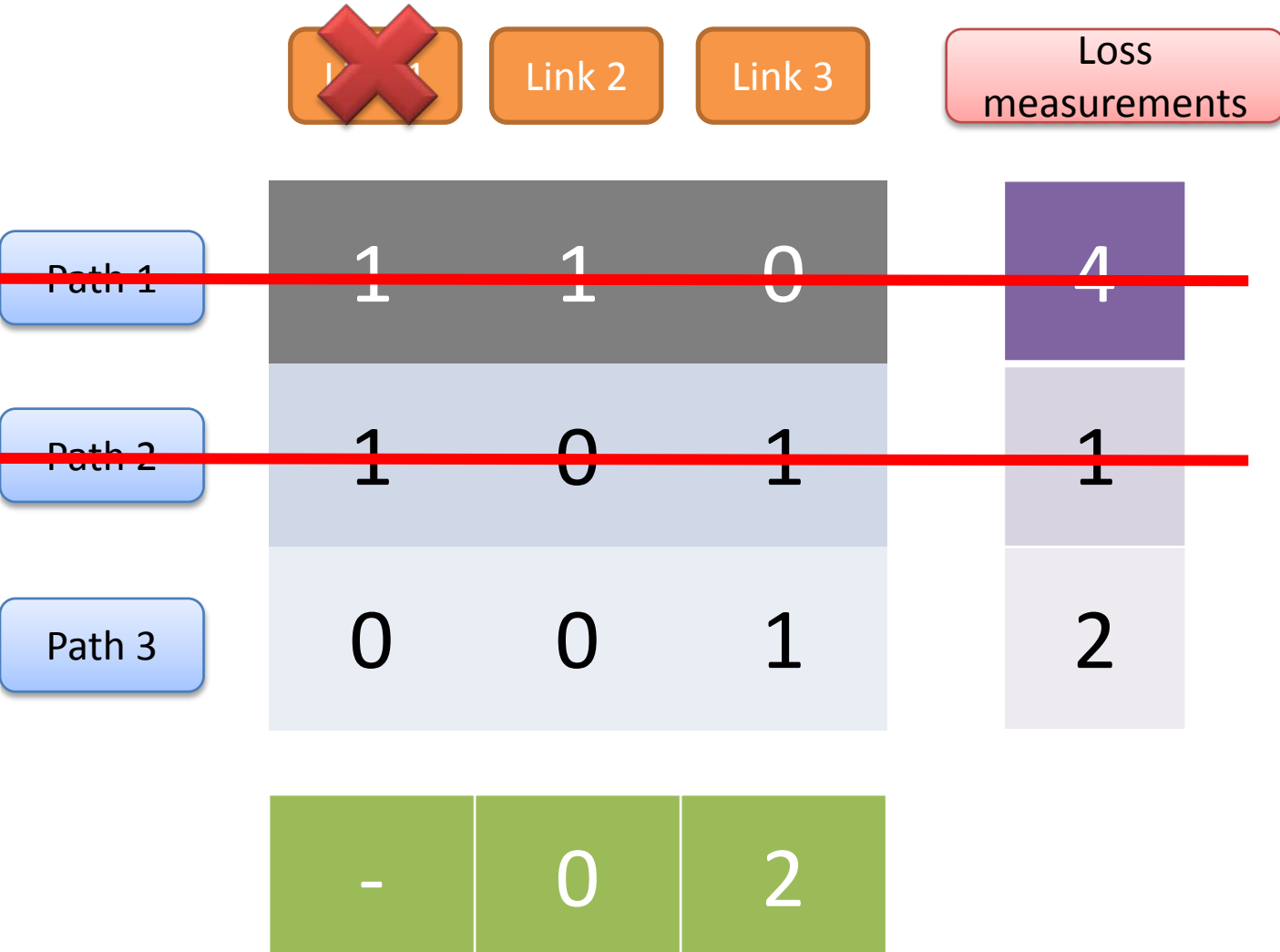
- In each iteration we select a link that can explain the largest number of probe losses until all are explained
- Link1 is bad.

PLL algorithm



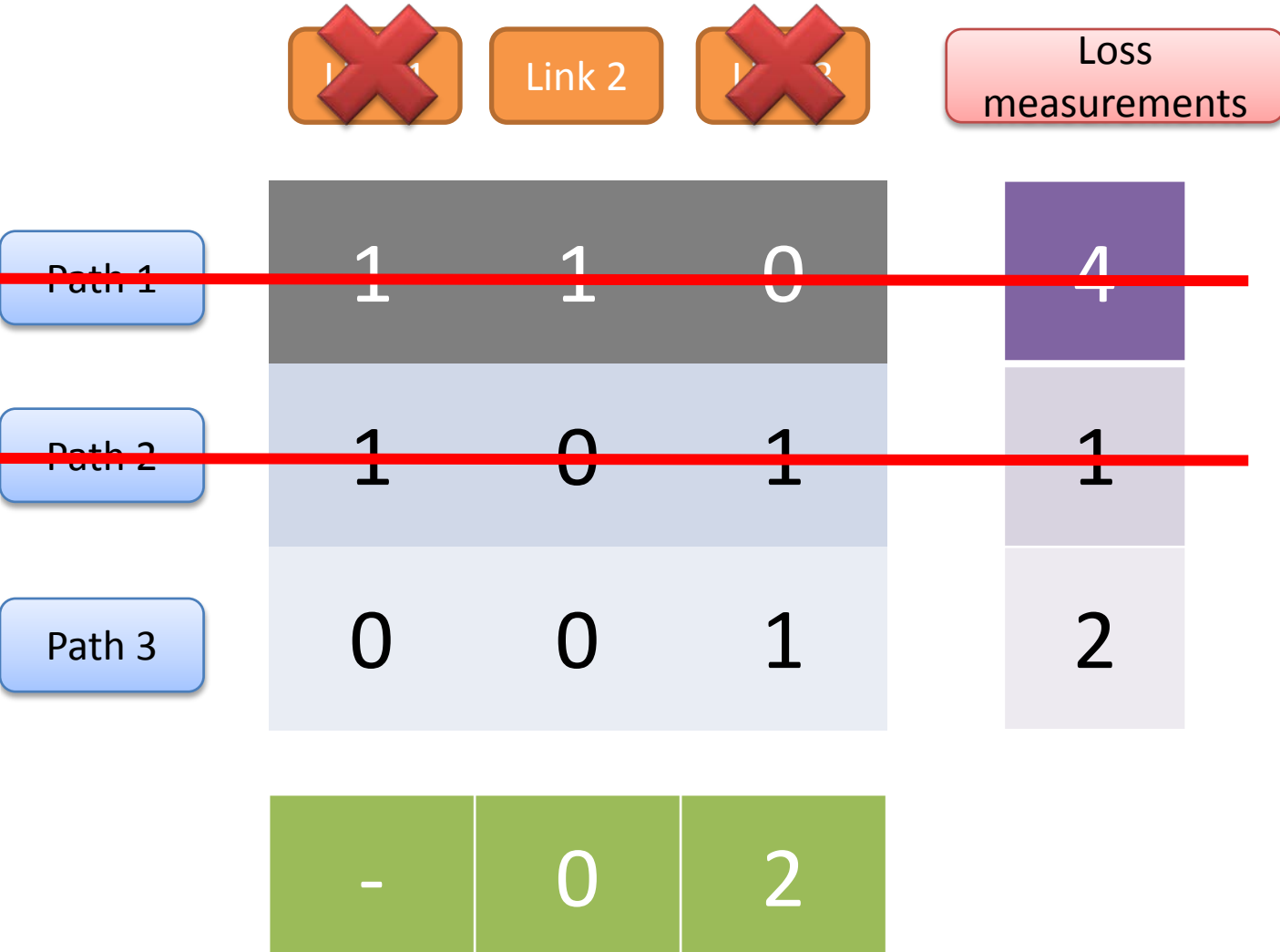
- In each iteration we select a link that can explain the largest number of probe losses until all are explained
- Link1 is bad.

PLL algorithm



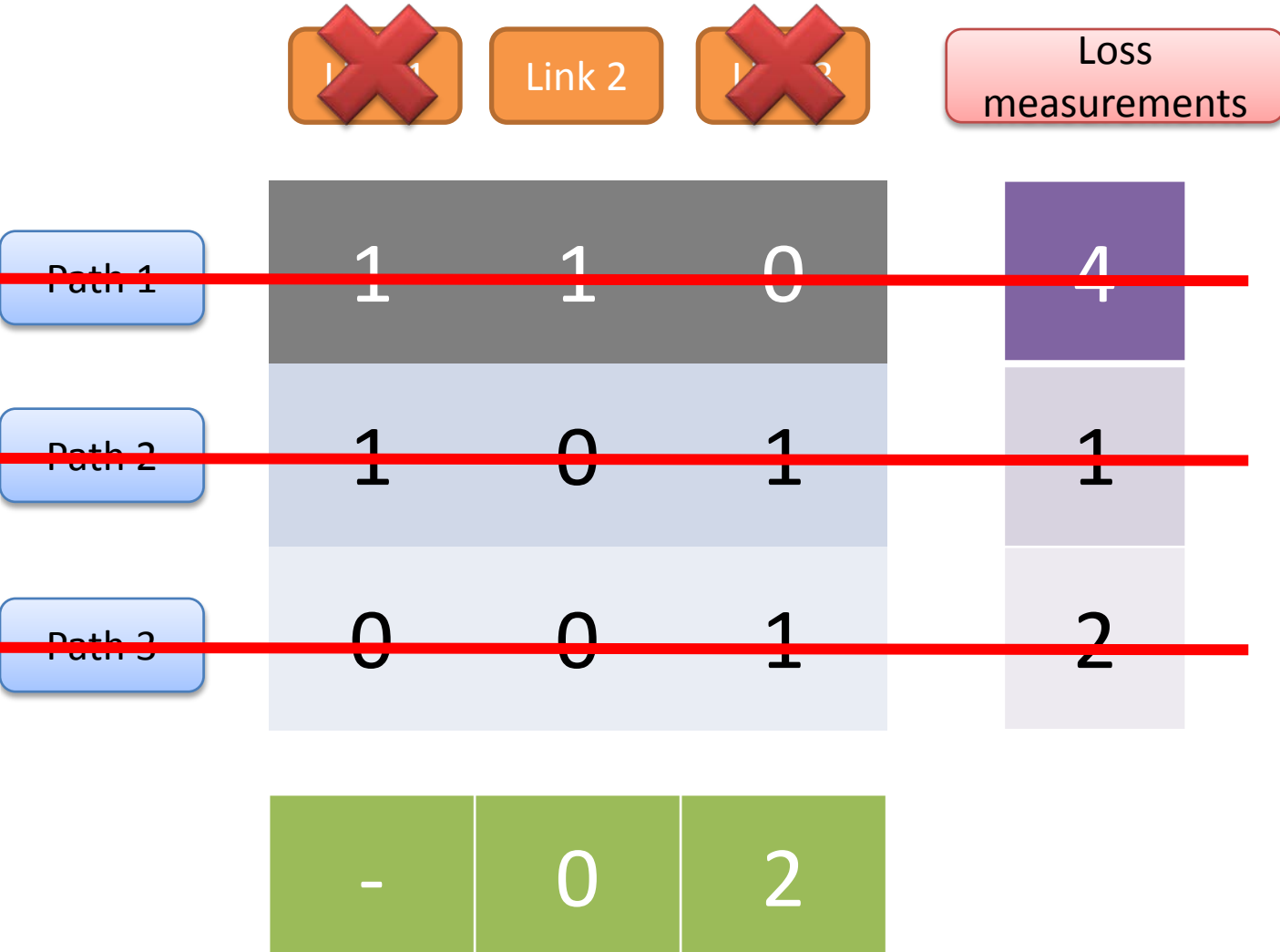
- In each iteration we select a link that can explain the largest number of probe losses until all are explained.

PLL algorithm



- In each iteration we select a link that can explain the largest number of probe losses until all are explained.
- Link3 is also bad.

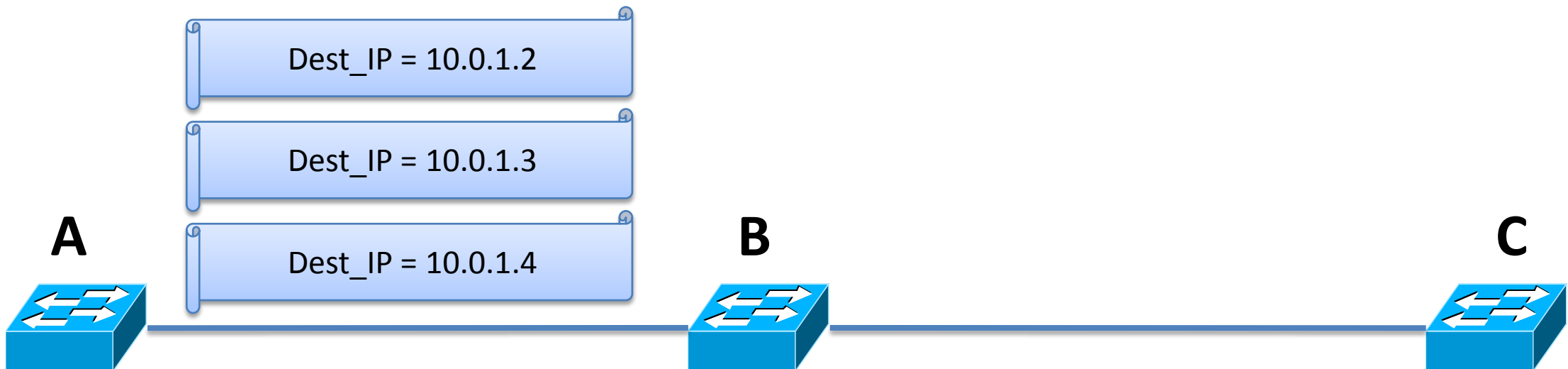
PLL algorithm



- In each iteration we select a link that can explain the largest number of probe losses until all are explained.
- Link3 is also bad.

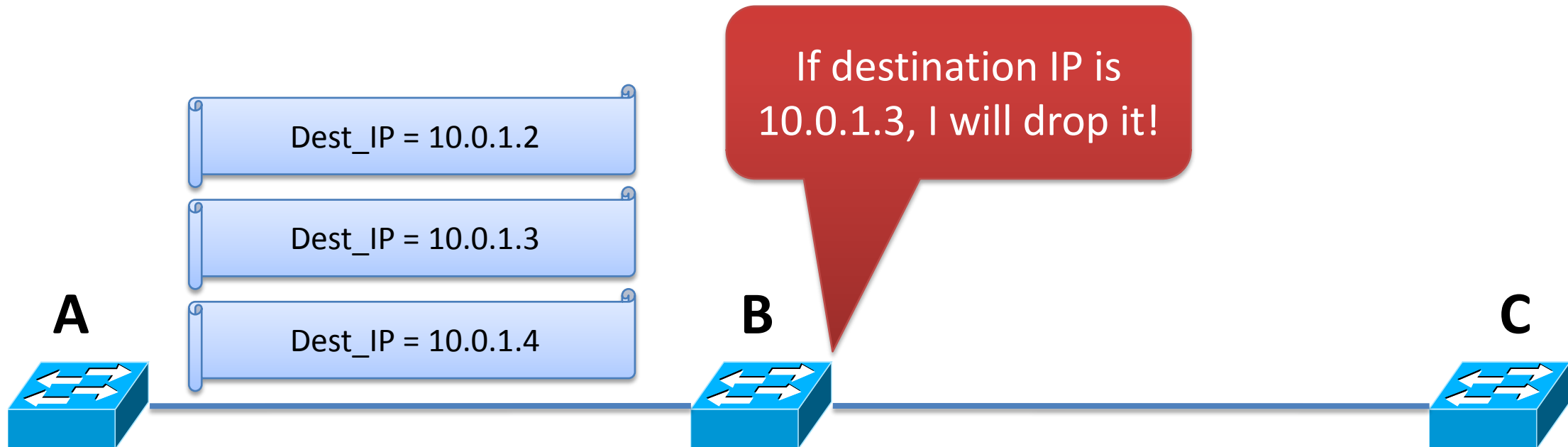
Losses are more complicated in reality

- Partial link loss: if a link is bad, **not all** probing paths through it will observe probe losses.
 - Packet blackhole



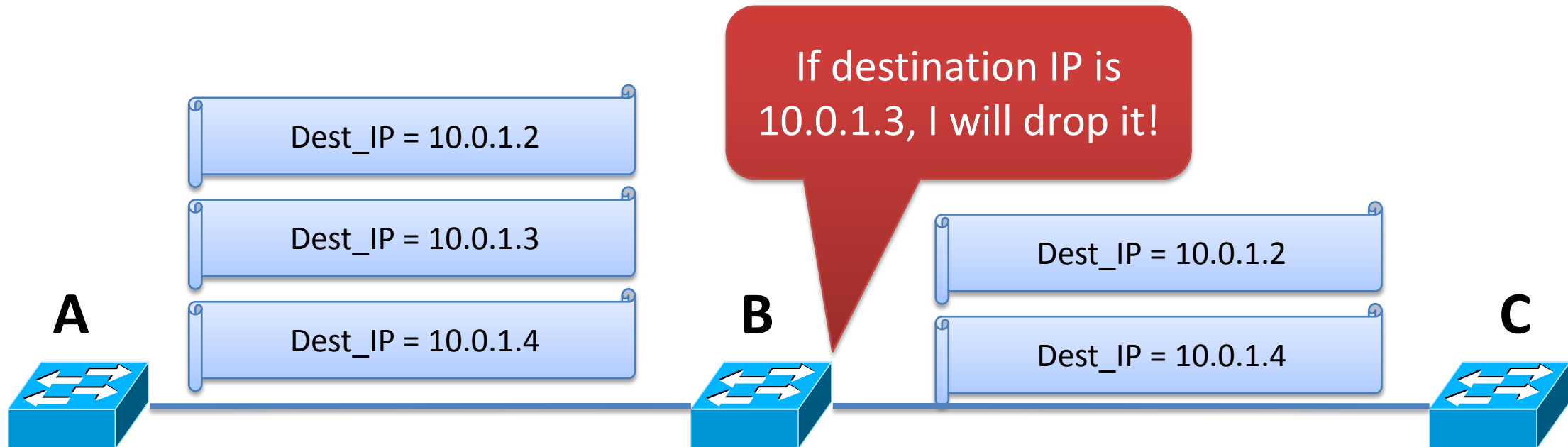
Losses are more complicated in reality

- Partial link loss: if a link is bad, **not all** probing paths through it will observe probe losses.
 - Packet blackhole



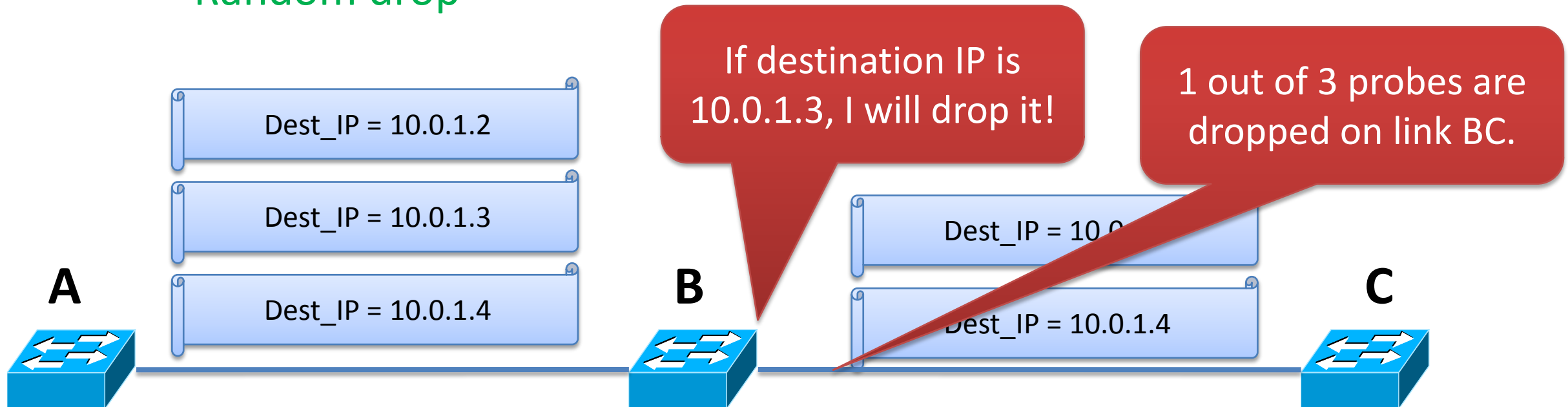
Losses are more complicated in reality

- Partial link loss: if a link is bad, **not all** probing paths through it will observe probe losses.
 - Packet blackhole



Losses are more complicated in reality

- Partial link loss: if a link is bad, **not all** probing paths through it will observe probe losses.
 - Packet blackhole
 - **Random drop**



Hit ratio

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	0
Path 2	1	0	1	2
Path 3	0	0	1	2

- Hit ratio of a link: the number of lossy paths divided by all paths through the link

Hit ratio

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	0
Path 2	1	0	1	2
Path 3	0	0	1	2
Hit ratio	0.5	0	1	

- Hit ratio of a link: the number of lossy paths divided by all paths through the link

Hit ratio

	Link 1	Link 2	Link 3	Loss measurements
Path 1	1	1	0	0
Path 2	1	0	1	2
Path 3	0	0	1	2
Hit ratio	0.5	0	1	

- Hit ratio of a link: the number of lossy paths divided by all paths through the link
- Only if the hit ratio of a link is **larger than a threshold** do we think the link is bad.

The tradeoff of hit ratio

- Larger hit ratio: more paths of a link experience loss and we have higher confidence that the link is faulty.

The tradeoff of hit ratio

- Larger hit ratio: more paths of a link experience loss and we have higher confidence that the link is faulty.
- But some failures only incur packet loss on one or two specified paths.
 - Our algorithm may miss them.

Conclusion

- ❑ deTector is a real-time, low-overhead and high-accuracy monitoring system for large-scale data center networks.
 - Real-time → without help of other diagnosis tools
 - Low overhead → PMC algorithm minimizes the probing cost
 - High accuracy → carefully design probe matrix + PLL algorithm
 - Large scale → optimizations to speed up the computation of PMC and PLL

Thanks

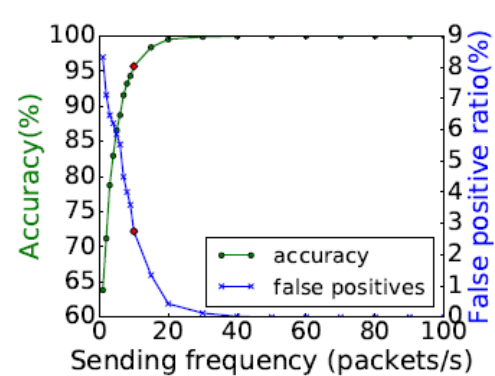
Evaluation

- Fault localization accuracy and running time in a 48-radix Fattree

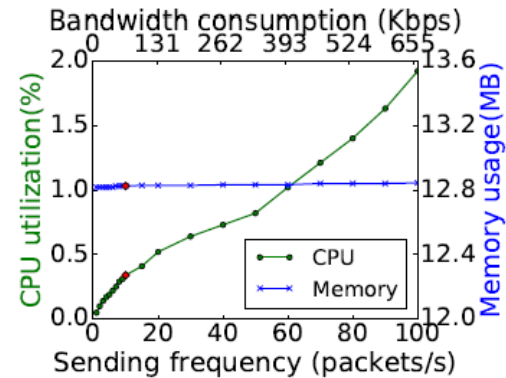
Algorithms \ # of failed links	Accuracy (%)				False positive ratio (%)				Running time (seconds)	
	1	5	20	50	1	5	20	50	5	50
<i>Sherlock</i> [2]	33.90	5.57	1.45	1.24	10.26	14.43	34.58	56.10	79.200	225.600
<i>SCORE</i> [27]	93.86	90.27	90.02	88.77	3.68	4.37	4.42	5.19	0.720	9.600
<i>OMP</i> [35]	89.74	91.38	49.43	19.37	6.82	11.61	23.22	44.62	14.400	16.320
<i>Tomo</i> [9]	94.48	90.45	89.88	88.87	3.74	4.61	4.51	5.31	0.672	9.360
<i>PLL</i>	96.00	91.87	91.24	89.87	1.73	2.95	3.18	3.84	0.486	0.772

Evaluation

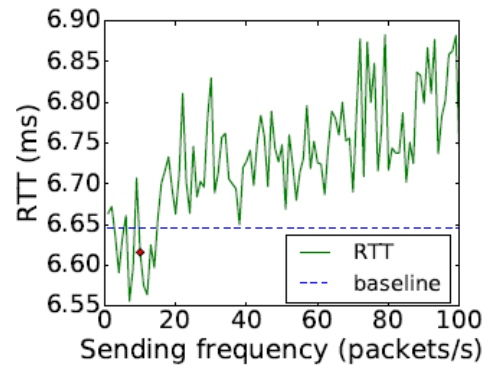
- Sensitivity test of sending frequency



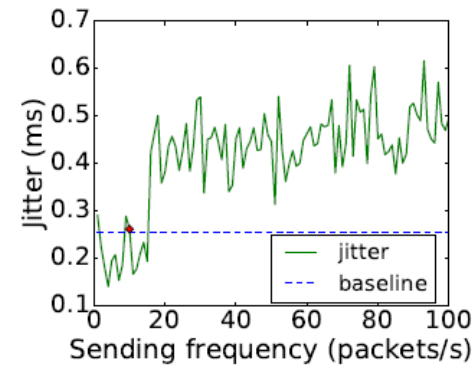
(a) PLL performance with one failure



(b) CPU, memory and bandwidth overhead on pingers



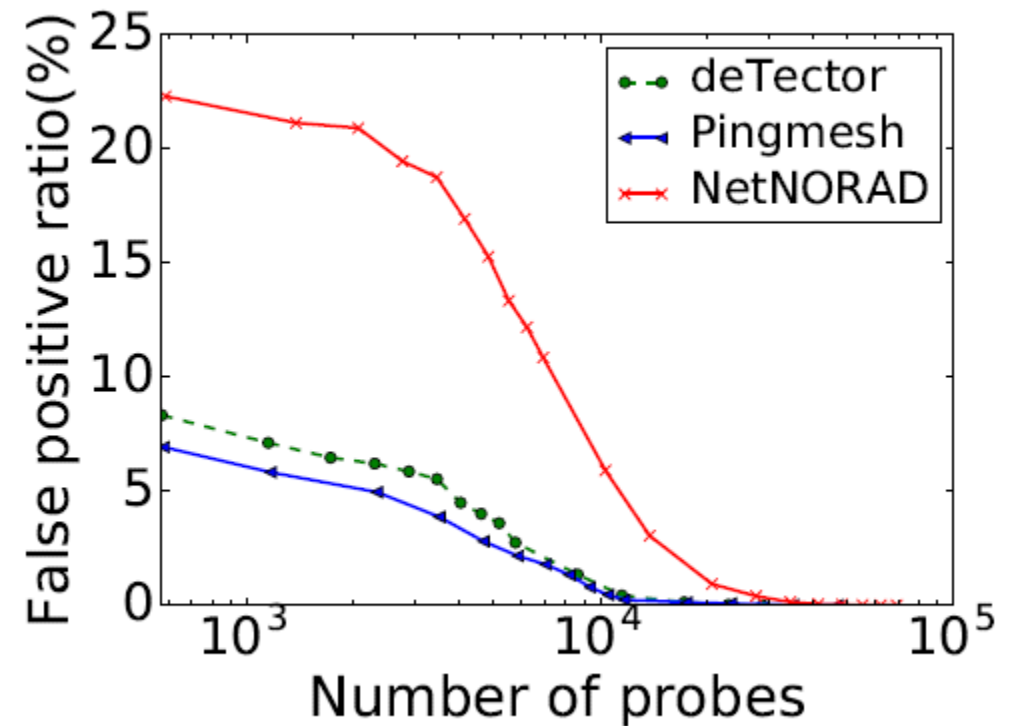
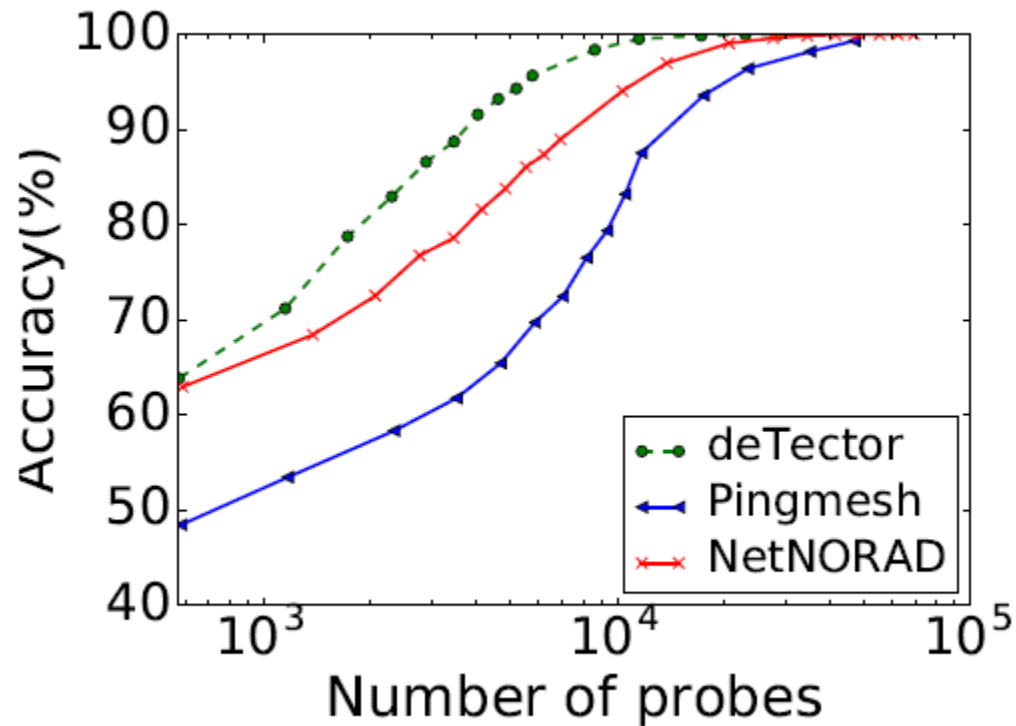
(c) RTT of background traffic



(d) Jitter of background traffic

Evaluation

- Accuracy and false positives of three monitoring systems with different number of probes



Evaluation

- Fault localization performance with probe matrix of 2-identifiability in a 48-ary Fattree

# of failed links	1	5	10	20	50
Accuracy (%)	98.95	98.99	98.98	98.93	98.87
False positive (%)	0.01	0.02	0.02	0.02	0.02
False negative (%)	1.05	1.01	1.02	1.07	1.13

Thanks