The Evolution of Traffic Routing in a Streaming World

Abhishek Srikanth

SOFTWARE ENGINEER,
FACEBOOK

A little about me

- Purdue University
- R&D Intern @ Bloomberg
- Program Manager Intern @ Microsoft
- Software Engineer @ Facebook

REAL-TIME

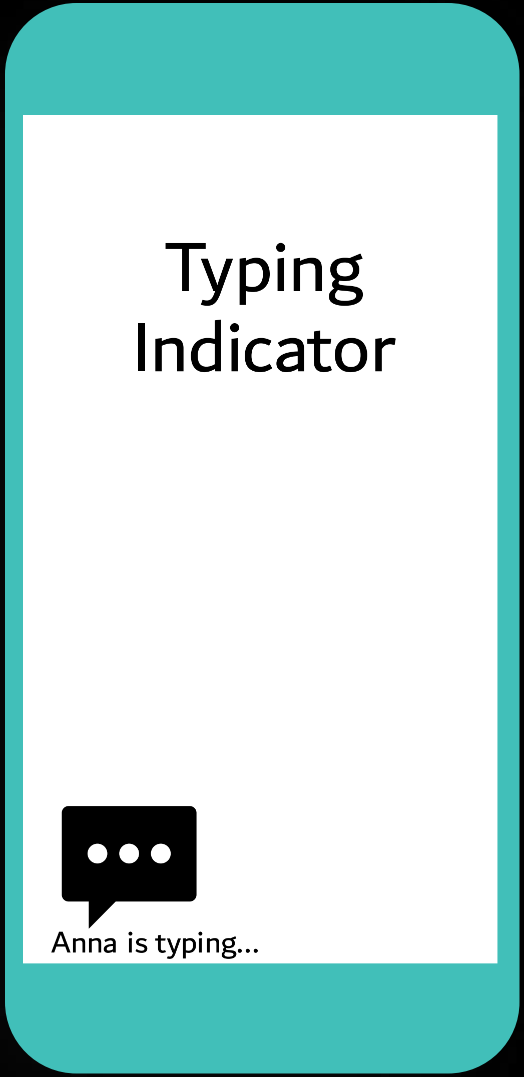
INFRASTRUCTURE

Live-Video Messages & Reactions

Pam: Looking forward to learning about traffic routing!

Tim: SRECon is awesome!

Jon: First!



Typing Indicator

Anna is typing...

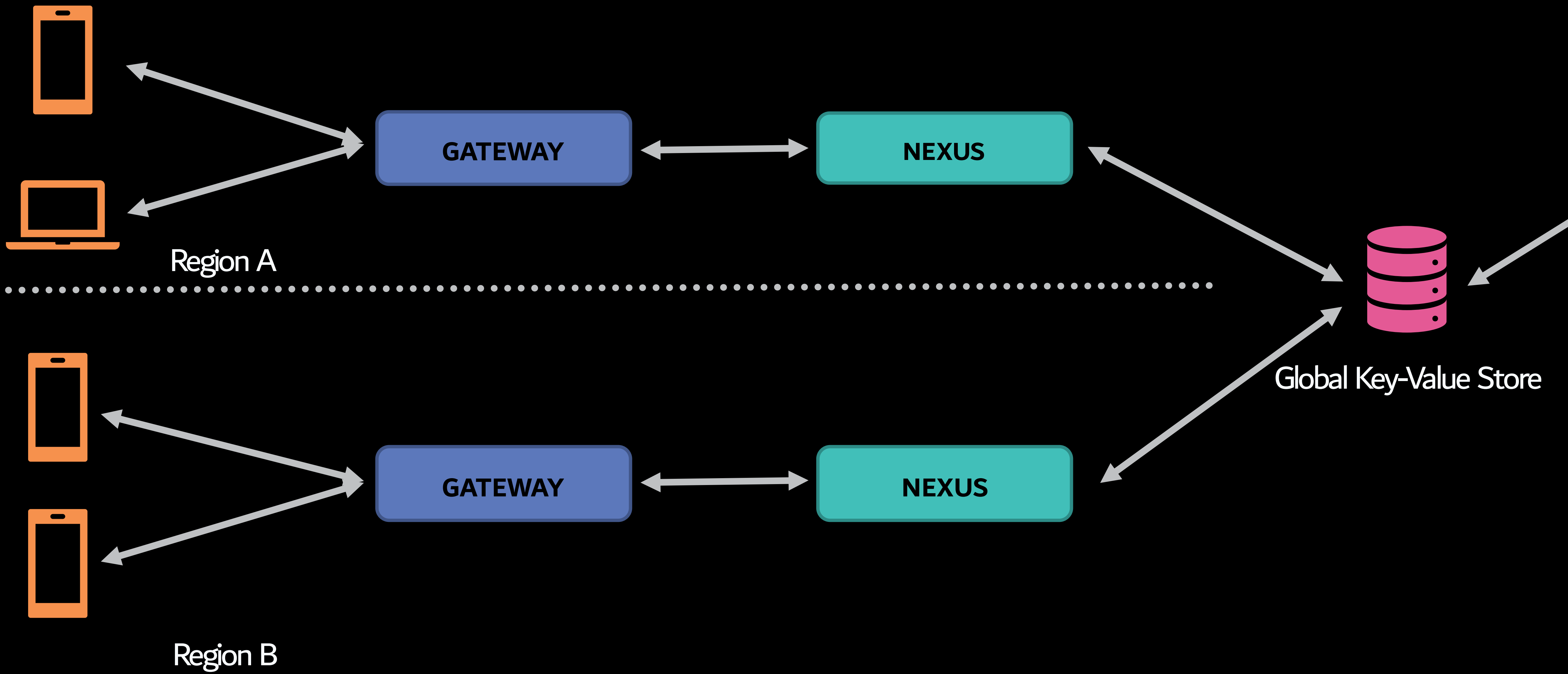
The Evolution

Collocated Architecture

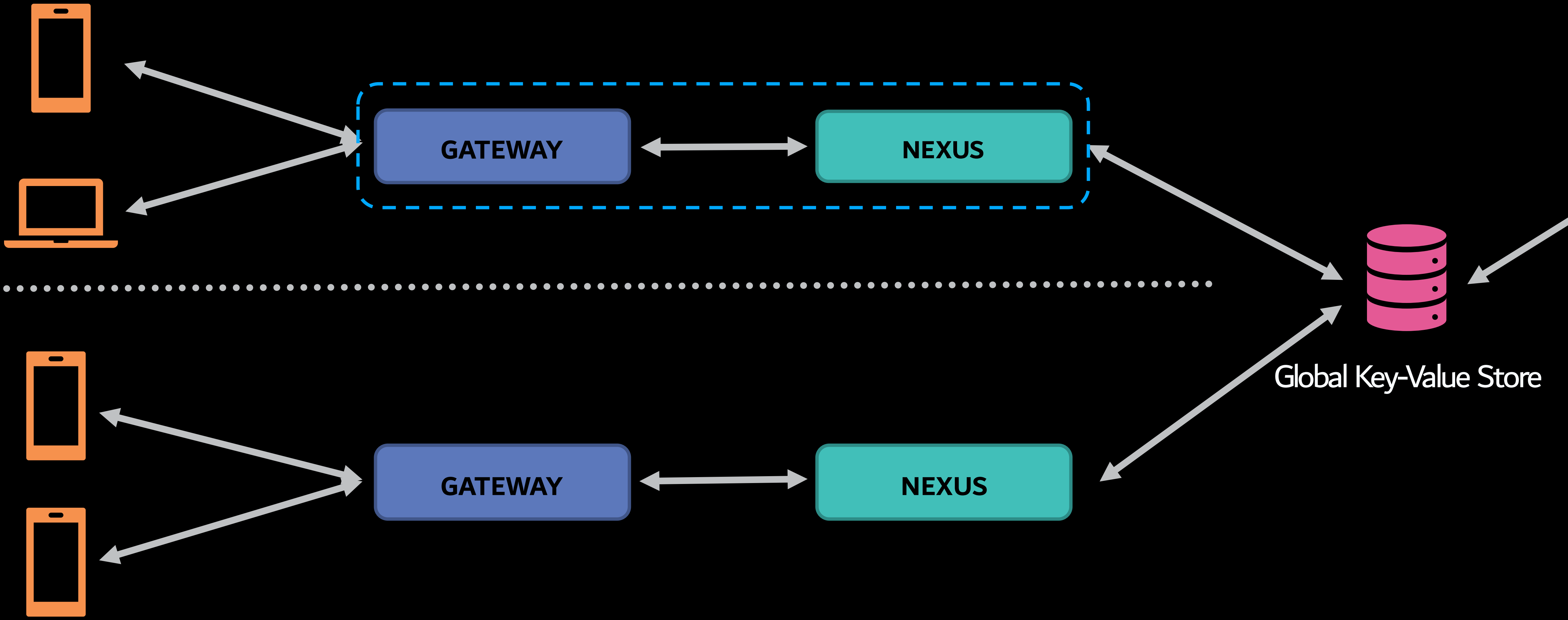
Off-box Architecture

Full-Mesh Architecture

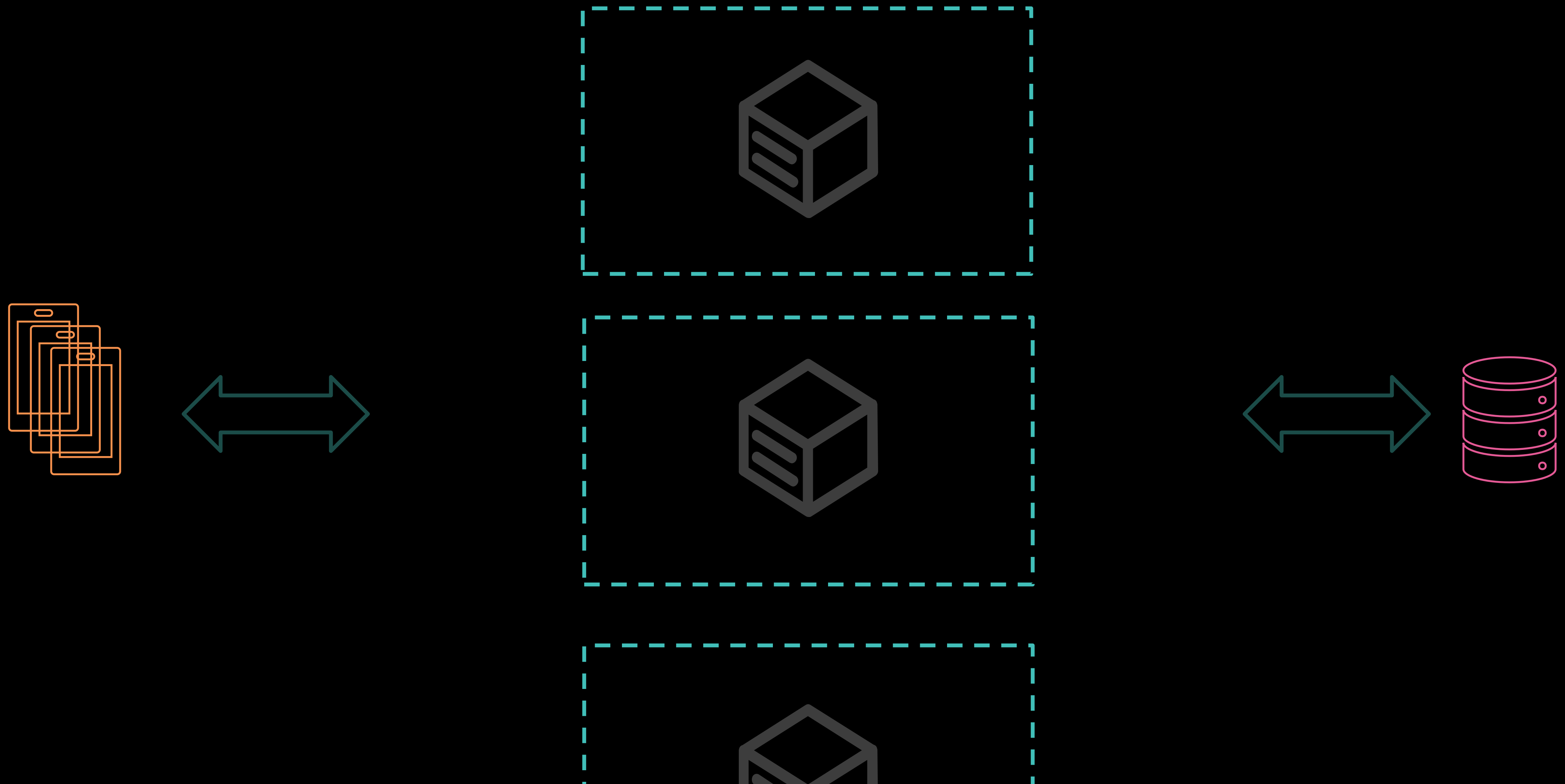
5000 ft view


GATEWAY
NEXUS
Region A
Global Key-Value Store
GATEWAY
NEXUS
Region B

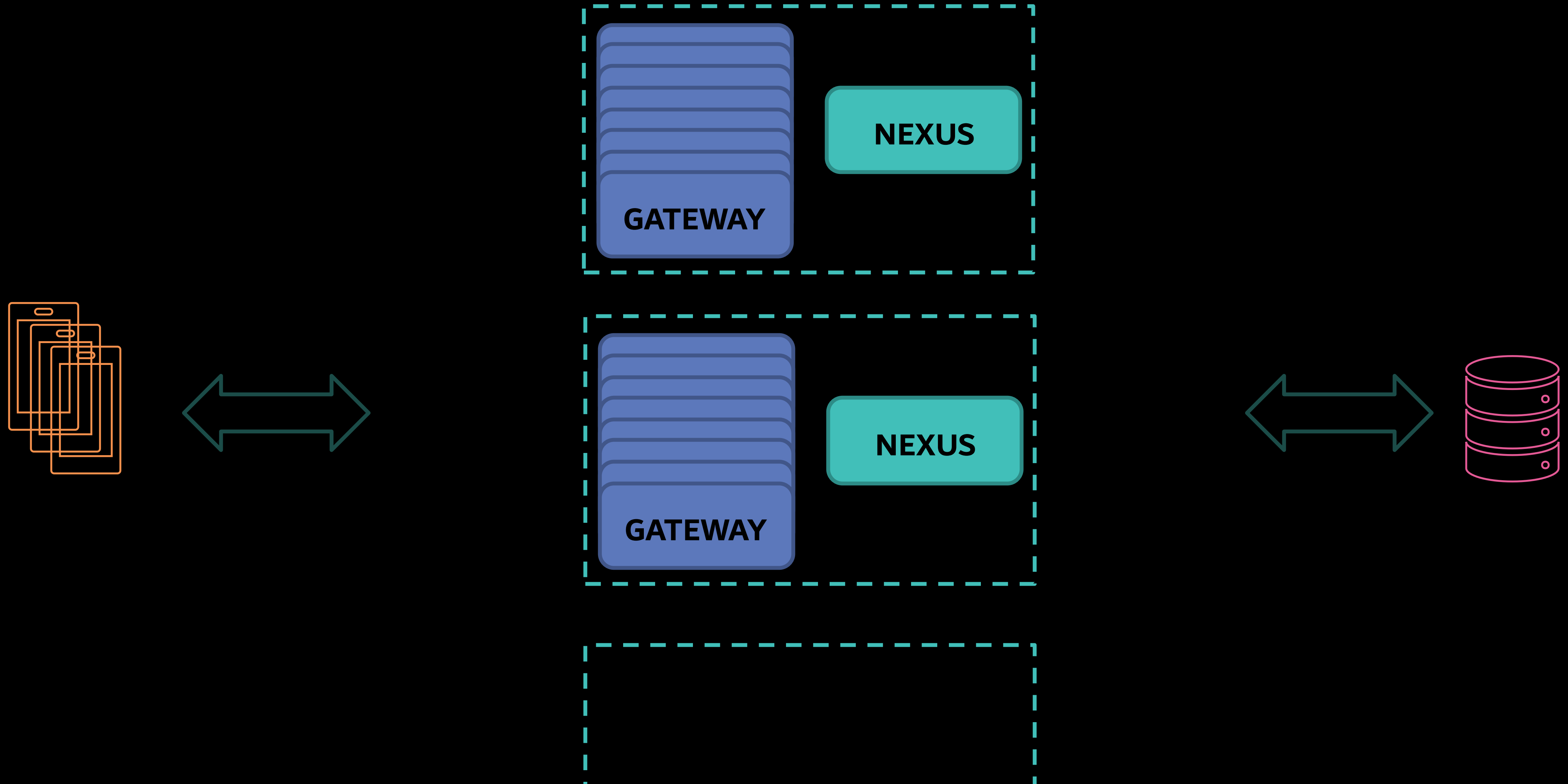
5000 ft view



Collocated Architecture

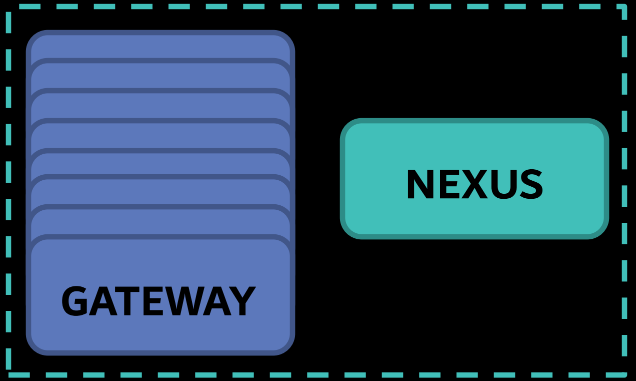


Collocated Architecture



Collocated Architecture

Problems

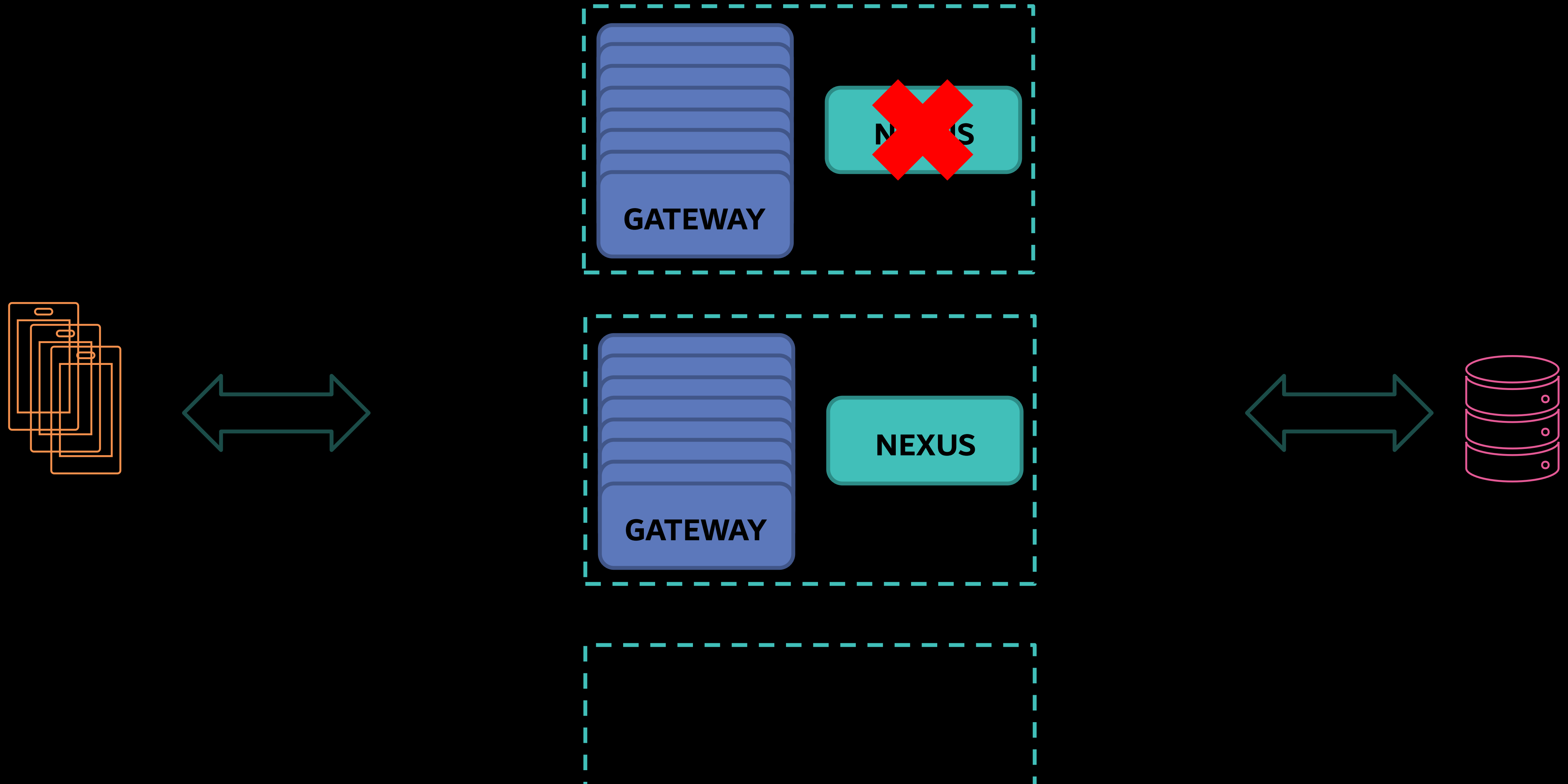


1. Shared Resources

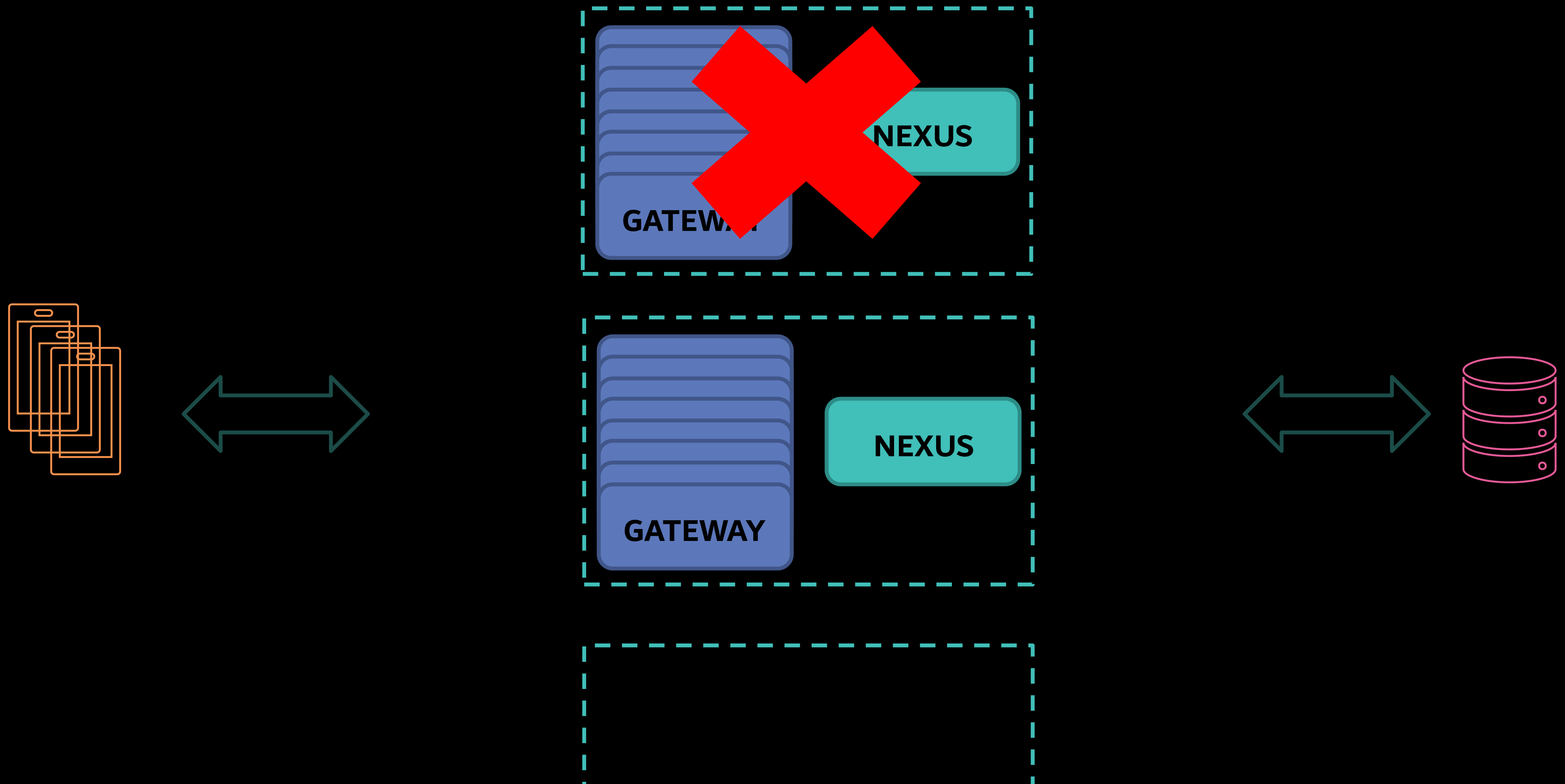
2. Independent Deployments

3. Low Fault Tolerance

Collocated Architecture

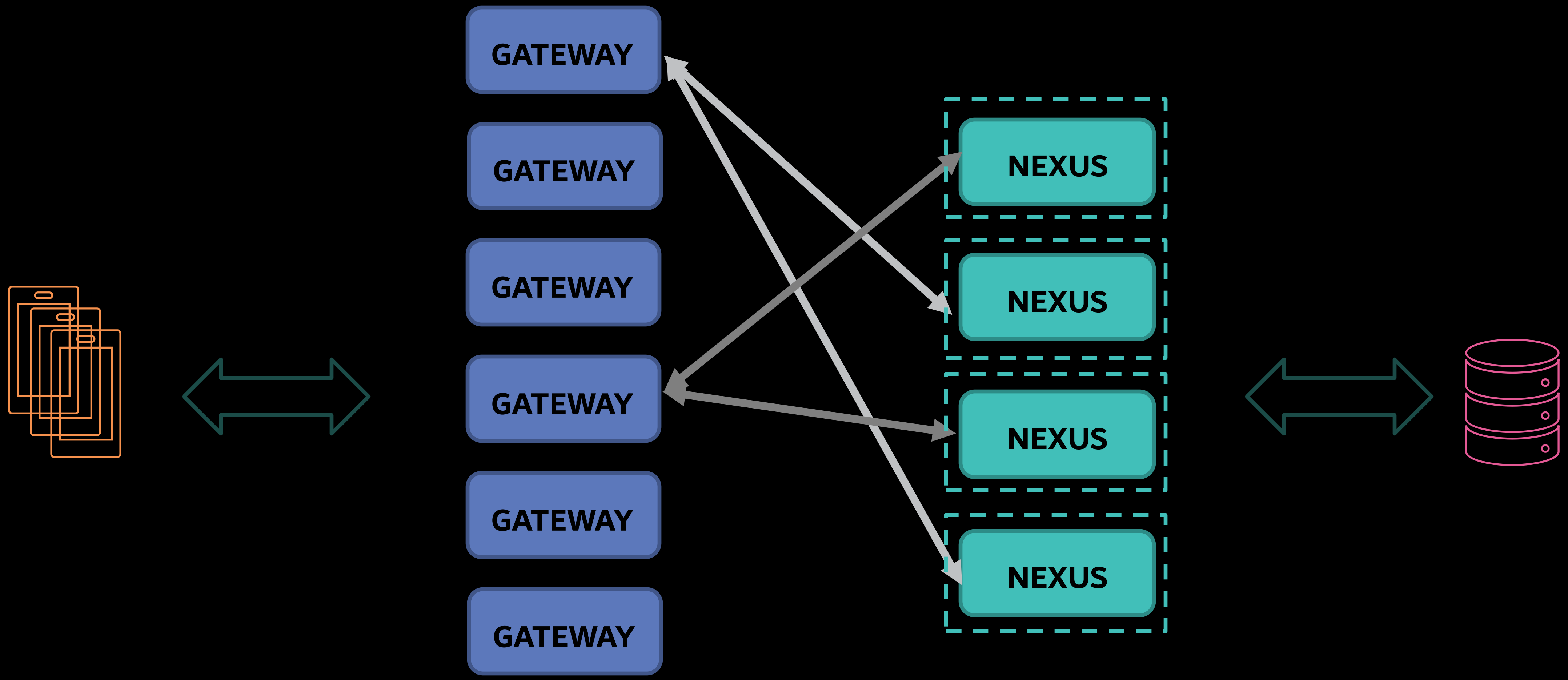


Collocated Architecture



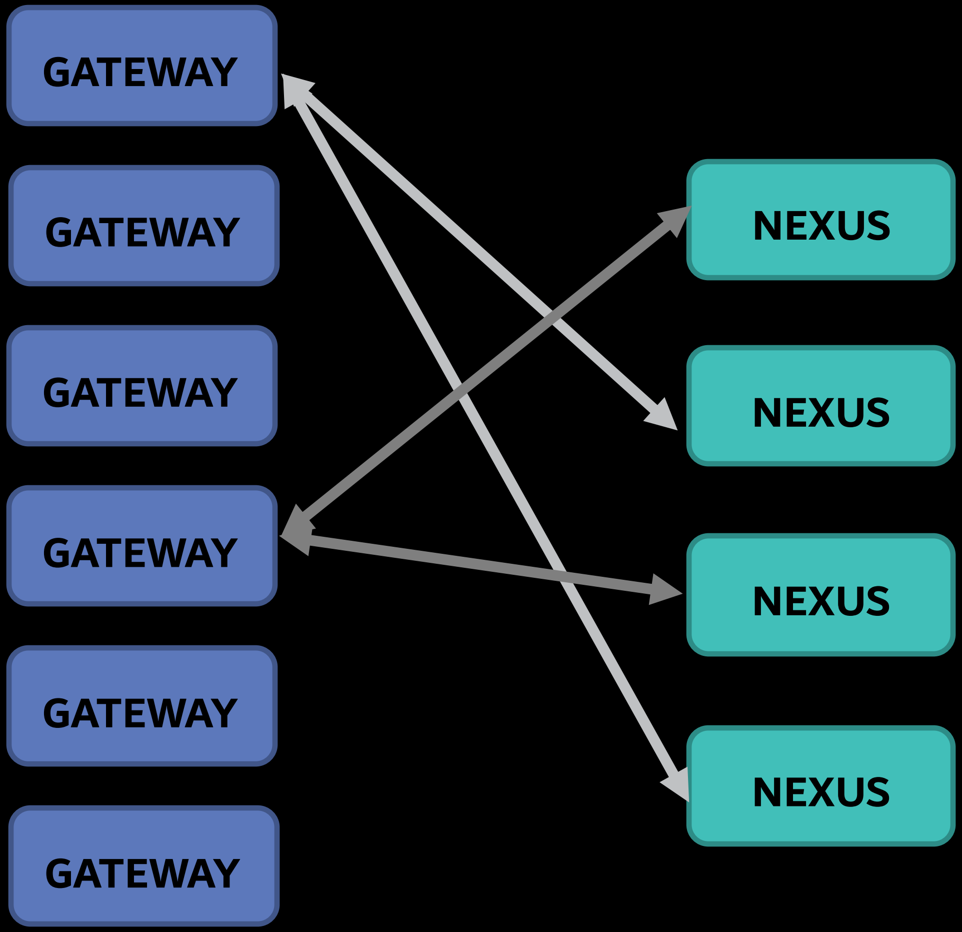
Off-box Architecture

Off-box Architecture



Off-box Architecture

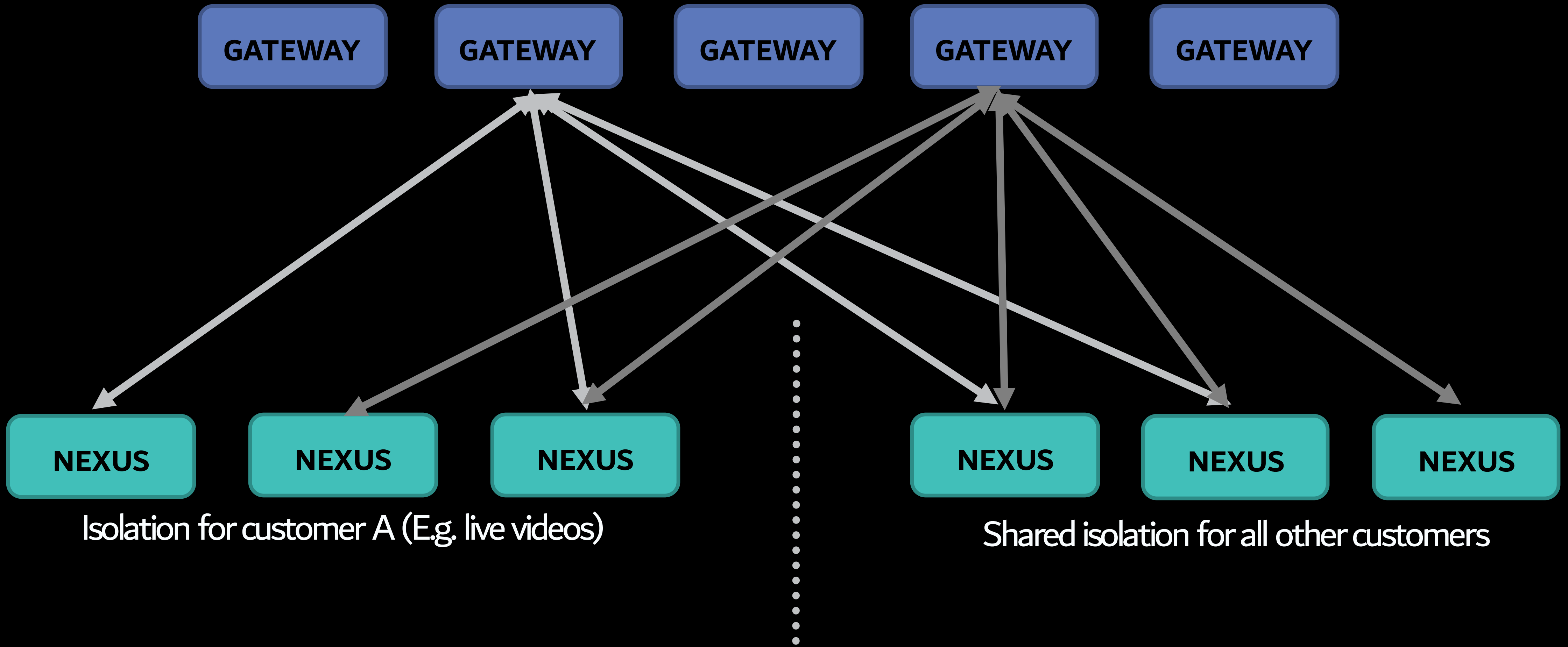
Advantages



1. Customer Traffic Isolation
2. High Fault Tolerance
3. Dynamic Load Balancing

Off-box Architecture

Advantages : Customer Traffic Isolation

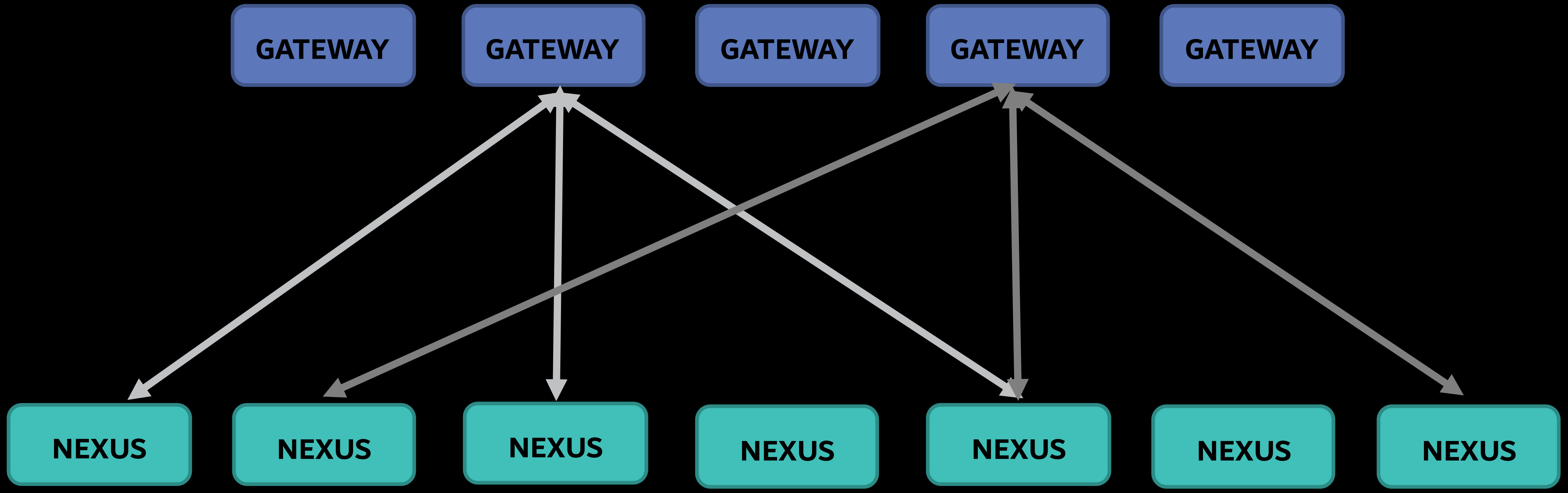


Isolation for customer A (E.g. live videos)

Shared isolation for all other customers

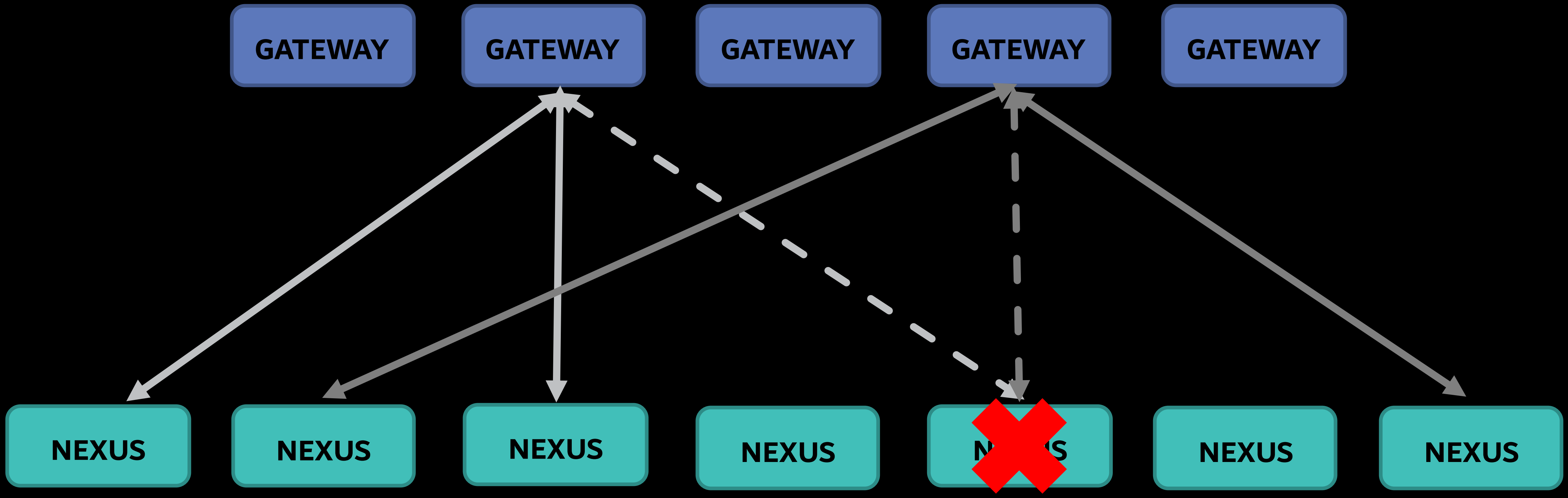
Off-box Architecture

Advantages : High Fault Tolerance



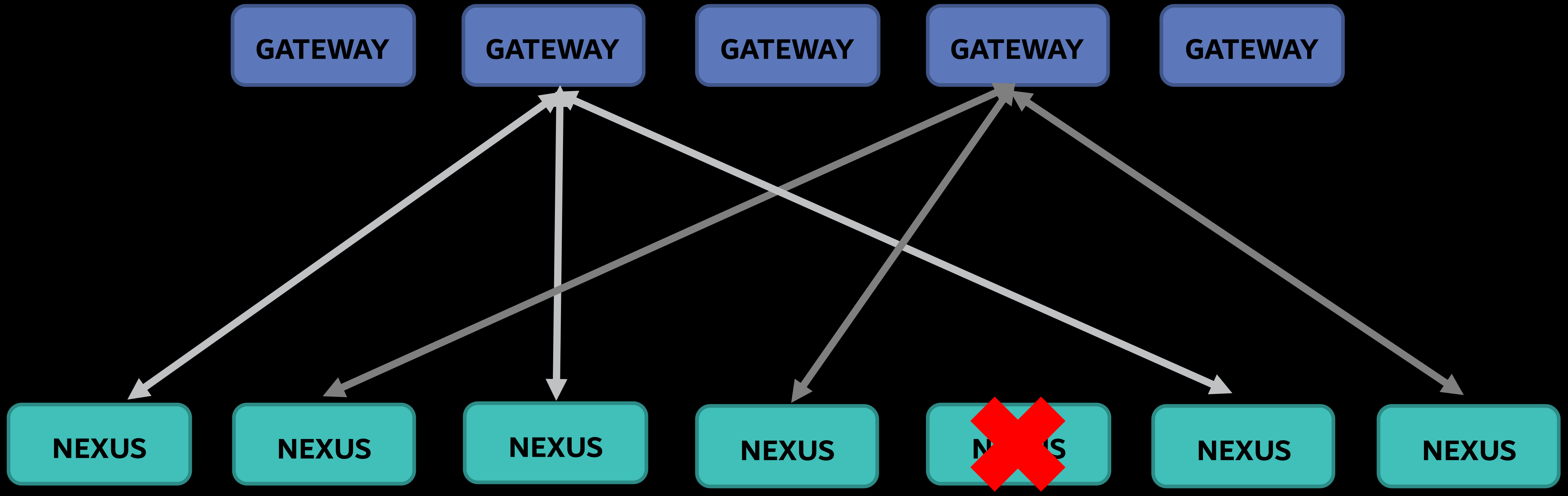
Off-box Architecture

Advantages : High Fault Tolerance



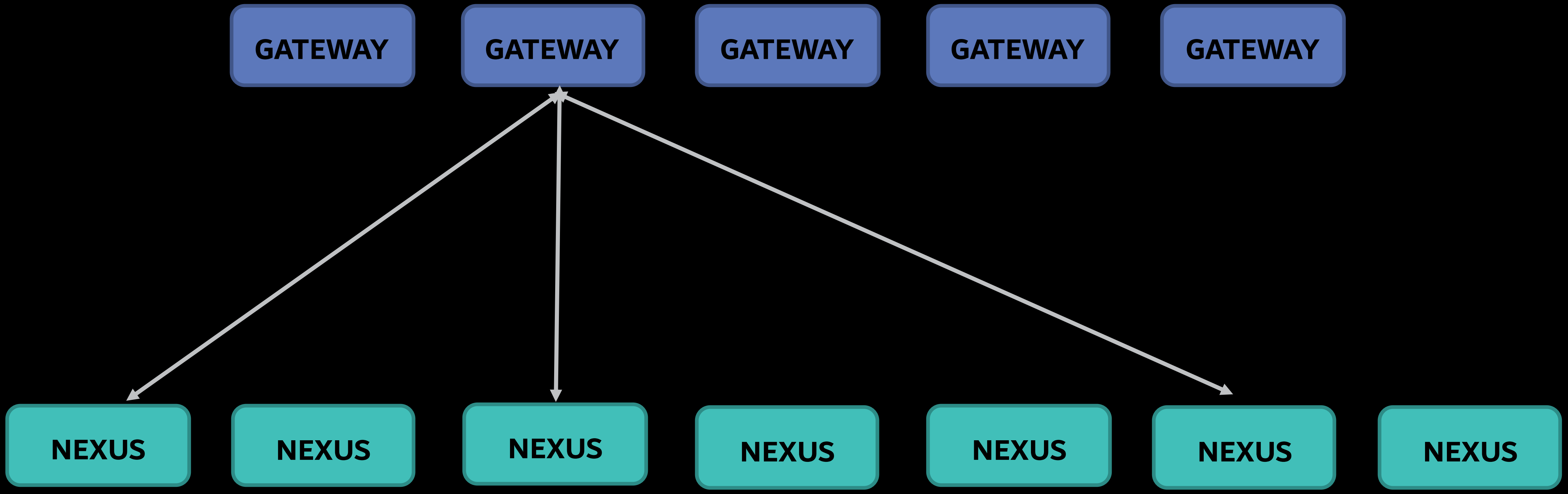
Off-box Architecture

Advantages : High Fault Tolerance



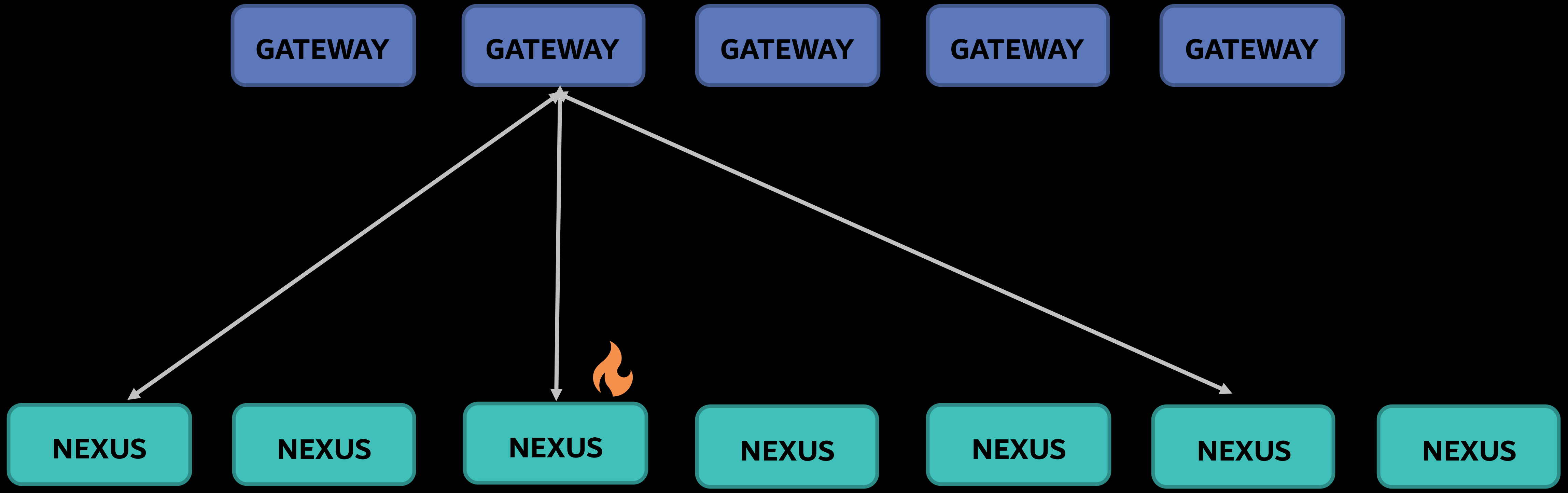
Off-box Architecture

Advantages : Dynamic Load Balancing



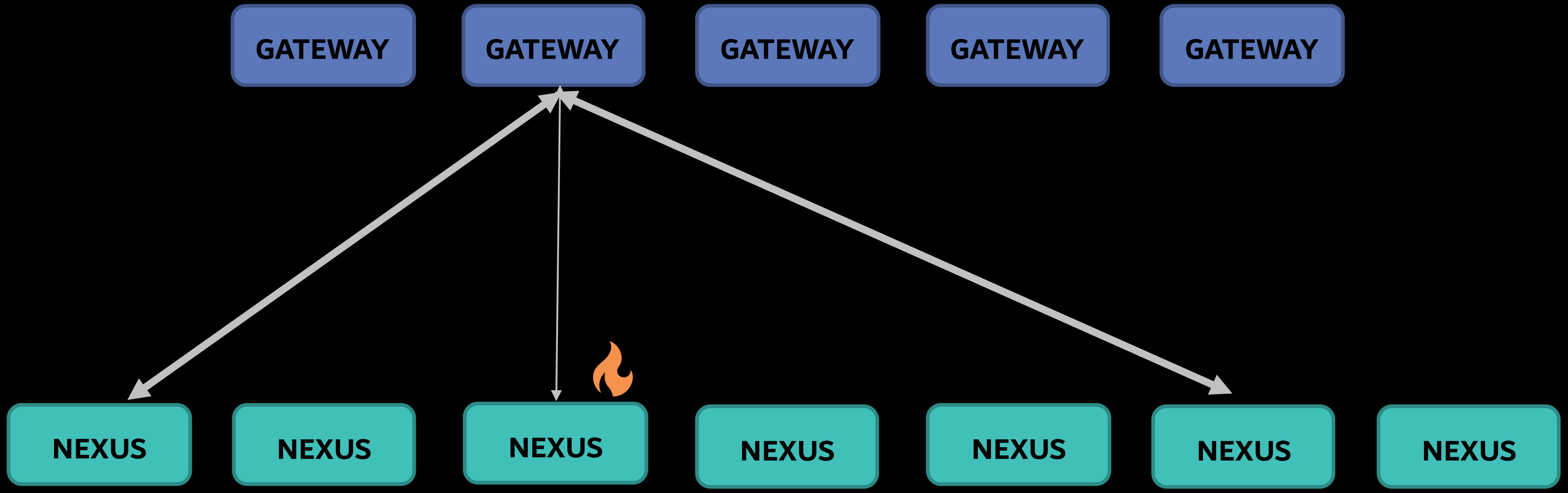
Off-box Architecture

Advantages : Dynamic Load Balancing



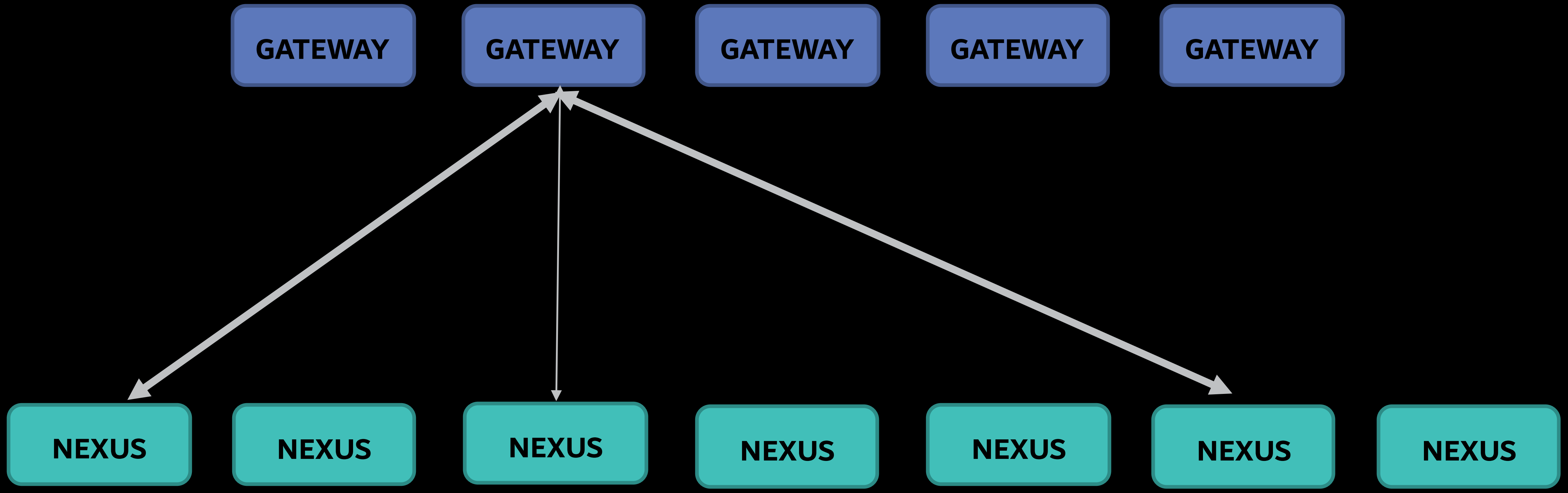
Off-box Architecture

Advantages : Dynamic Load Balancing



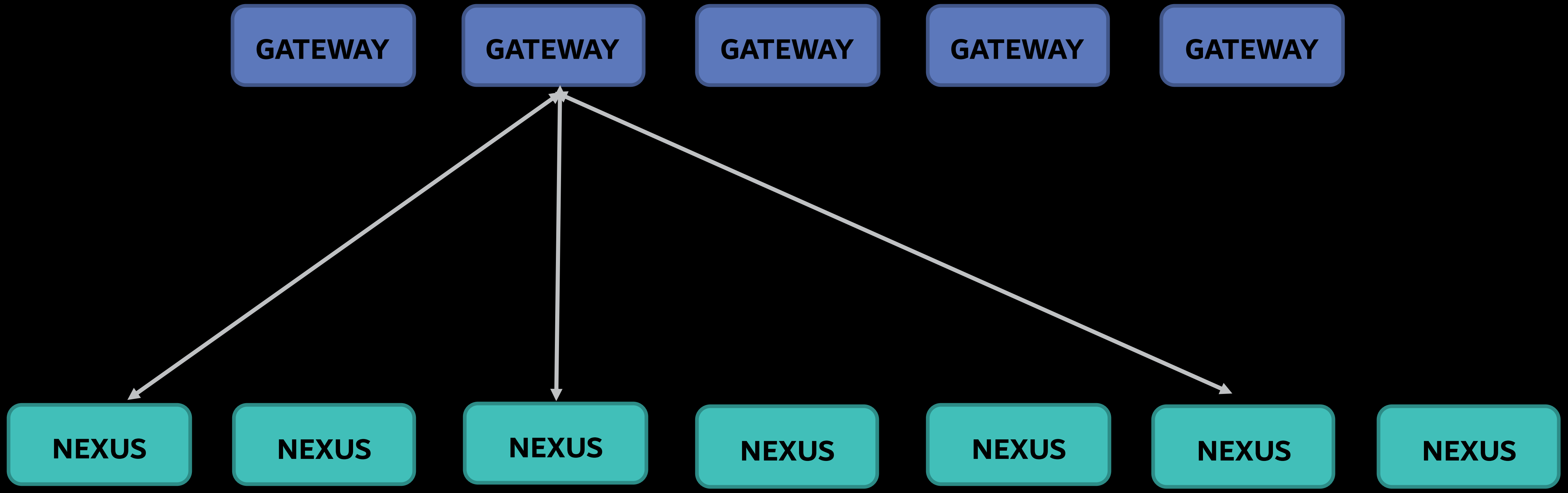
Off-box Architecture

Advantages : Dynamic Load Balancing



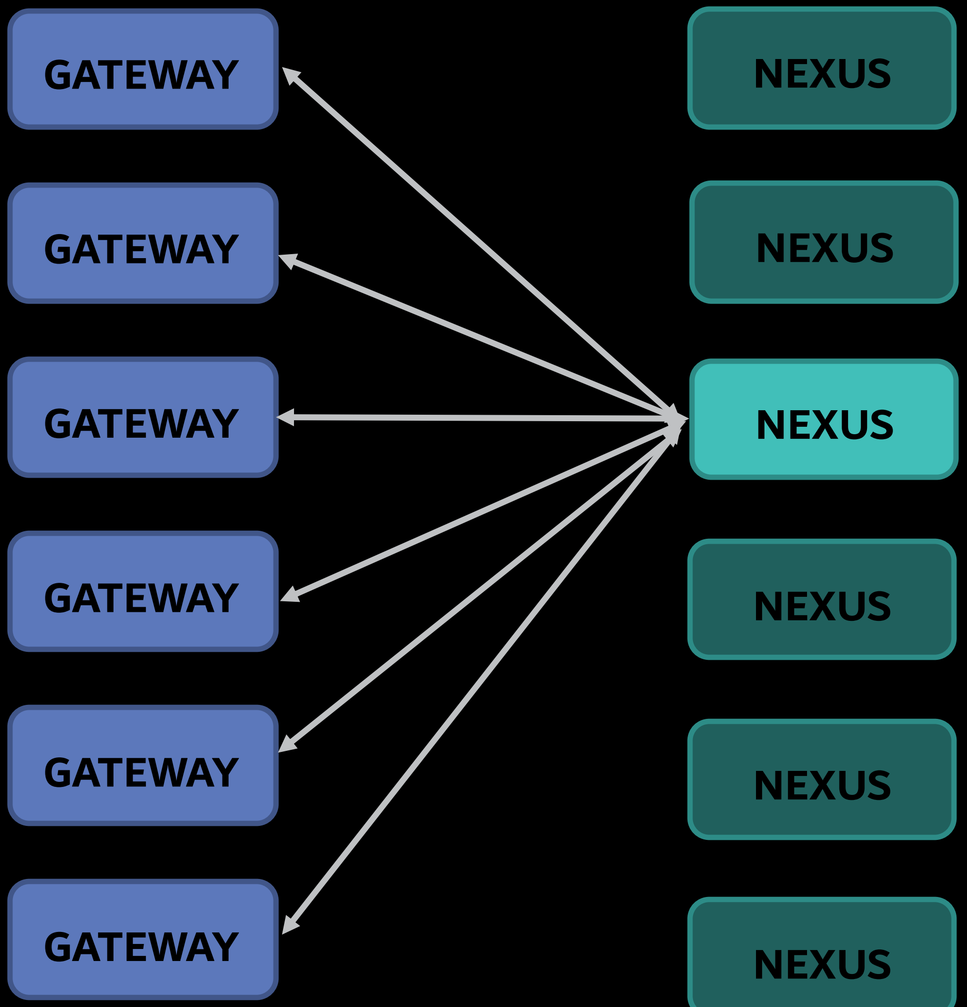
Off-box Architecture

Advantages : Dynamic Load Balancing



Off-box Architecture

Challenges: Socket Connections



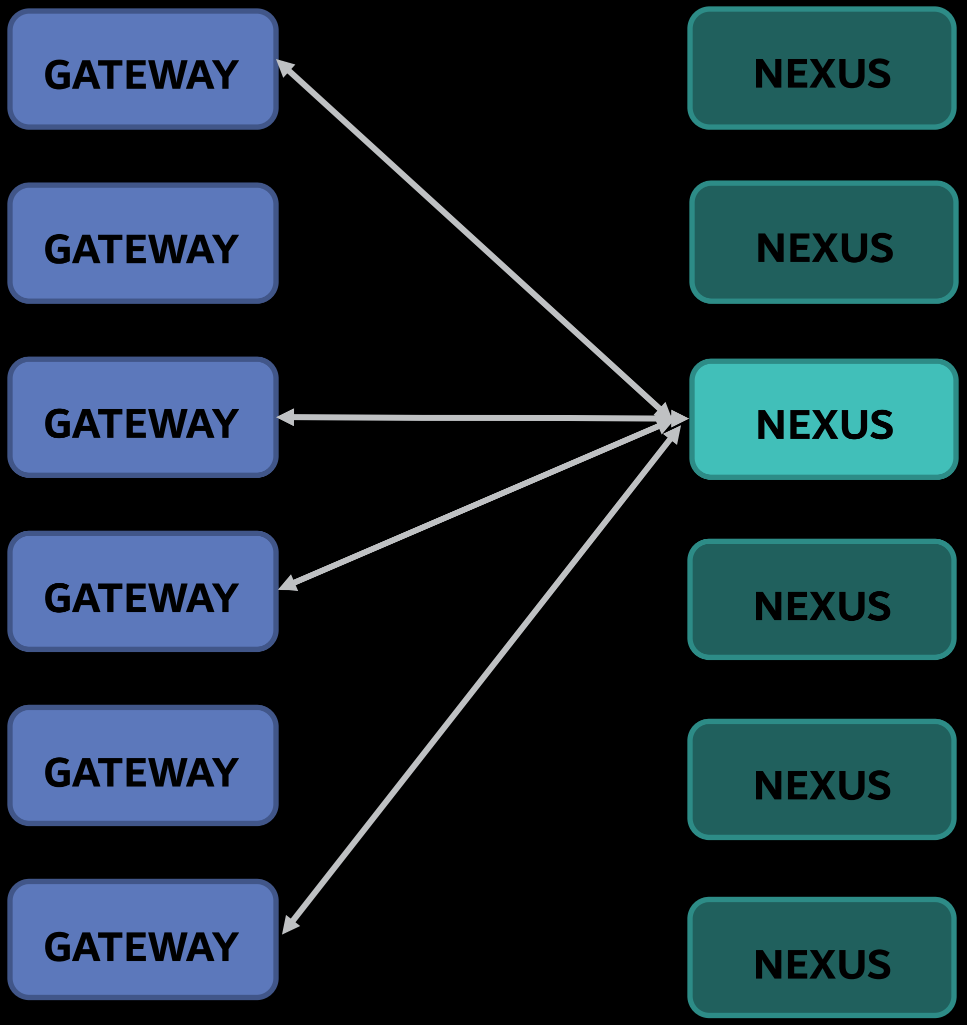
In a Full-Mesh connection:

Total Number of Connections: G * N

Number of Connections per Nexus: G

Off-box Architecture

Challenges: Socket Connections



In a Full-Mesh connection:

Total Number of Connections: G * N

Number of Connections per Nexus: G

If each Gateway Connected to only 'K' Nexuses:

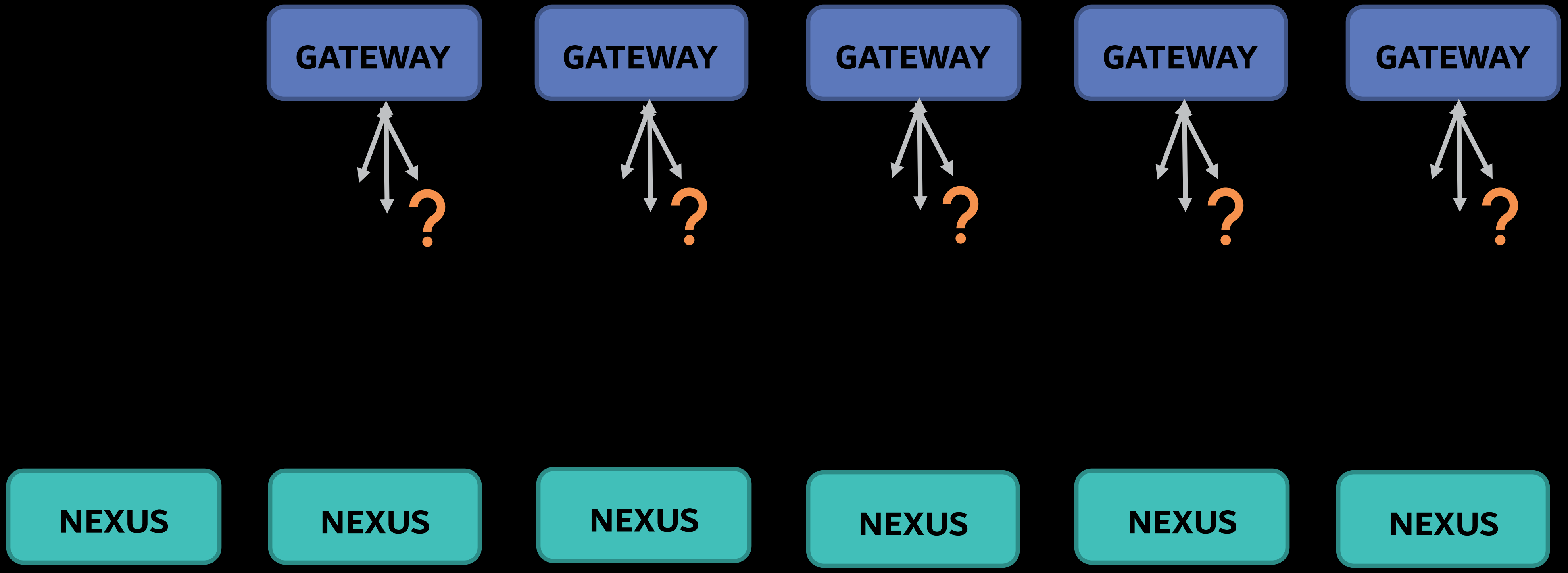
Total Number of Connections: G * K

Number of Connections per Nexus: G * K / N

For our system, K = 3 provided desirable results.

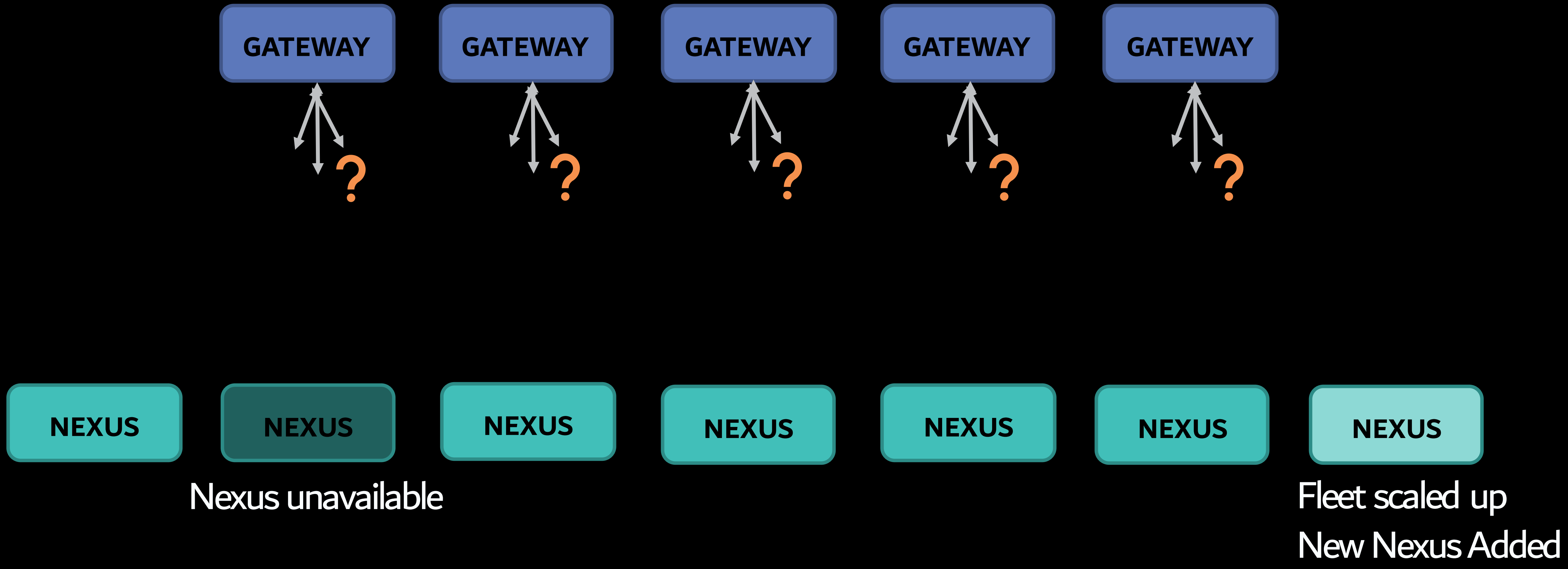
Off-box Architecture

Challenges: Which Nexus to talk to?



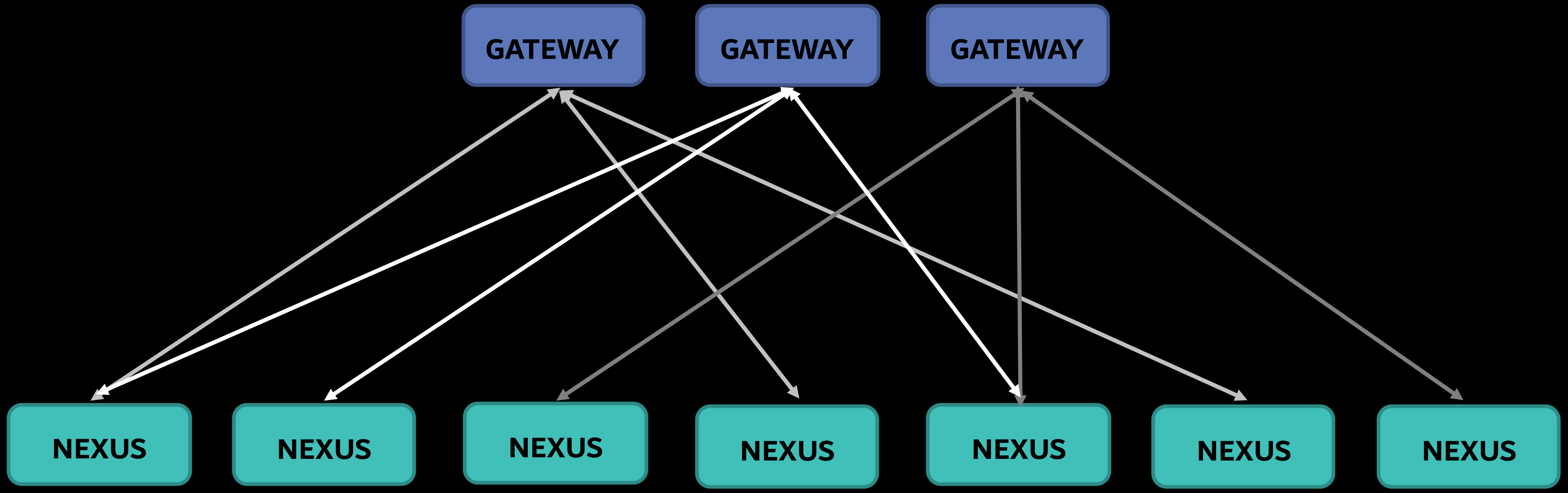
Off-box Architecture

Challenges: Which Nexus to talk to?



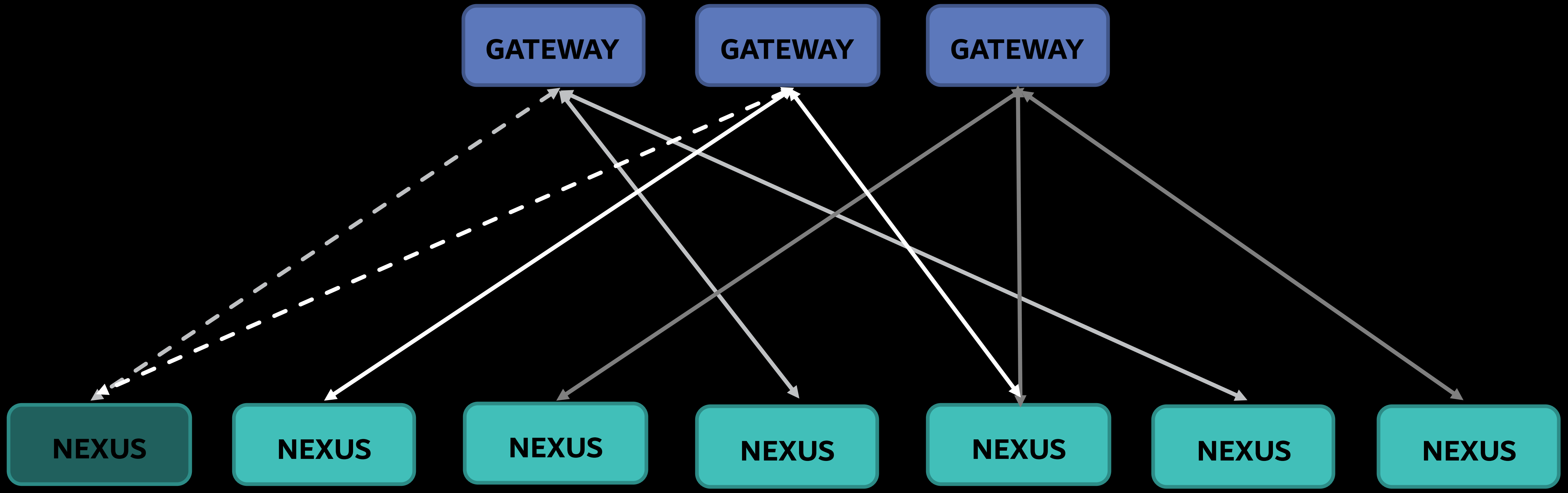
Off-box Architecture

Challenges: Which Nexus to talk to? A Naïve Solution



Off-box Architecture

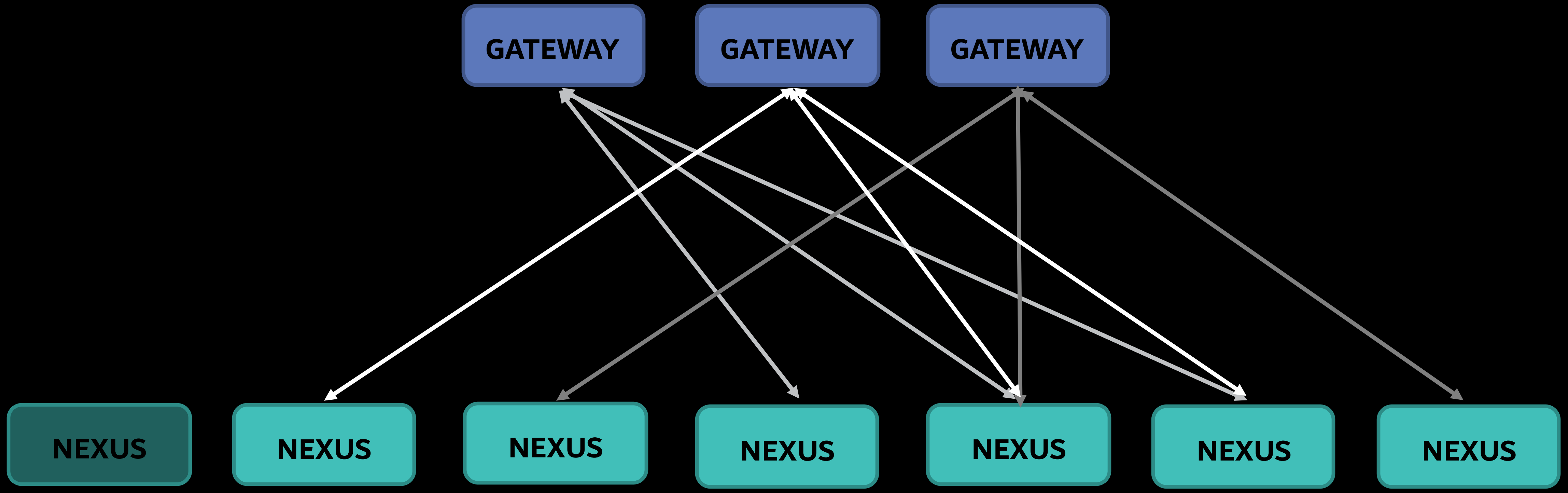
Challenges: Which Nexus to talk to? A Naïve Solution



Unavailable
(Due to deployment)

Off-box Architecture

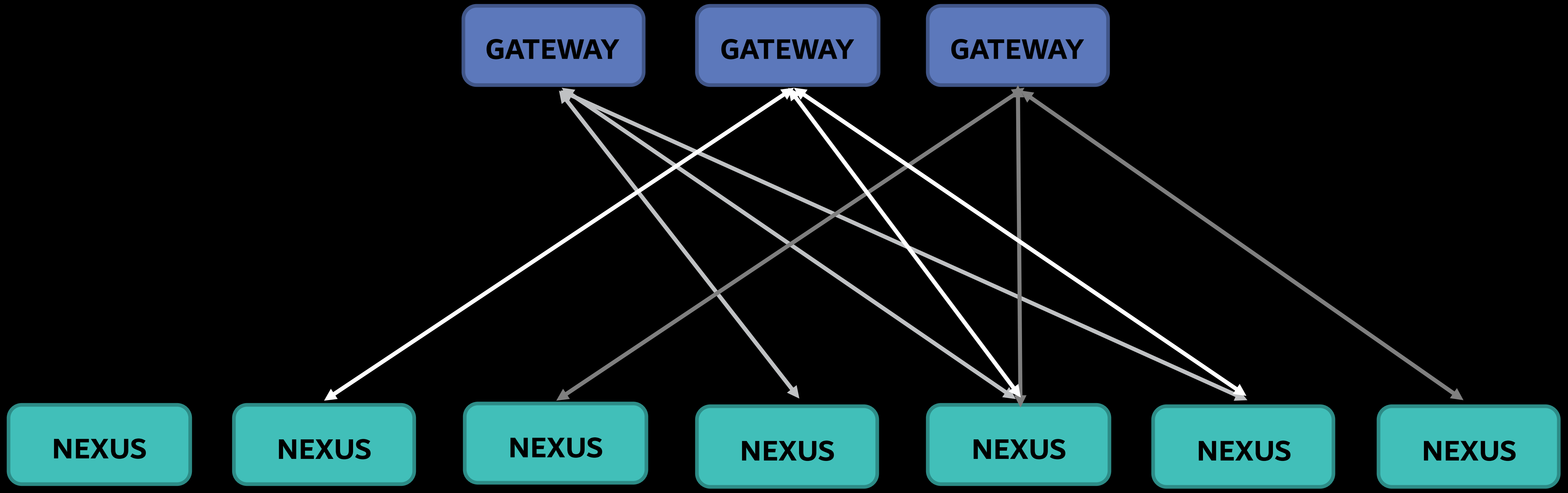
Challenges: Which Nexus to talk to? A Naïve Solution



Unavailable
(Due to deployment)

Off-box Architecture

Challenges: Which Nexus to talk to? A Naïve Solution



Available Again

Off-box Architecture

Challenges: Which Nexus to talk to?

Conditions:

- Independent Gateway Decisions
- Even Distribution of Connections
- Minimal Disruptions

Rendezvous Hash

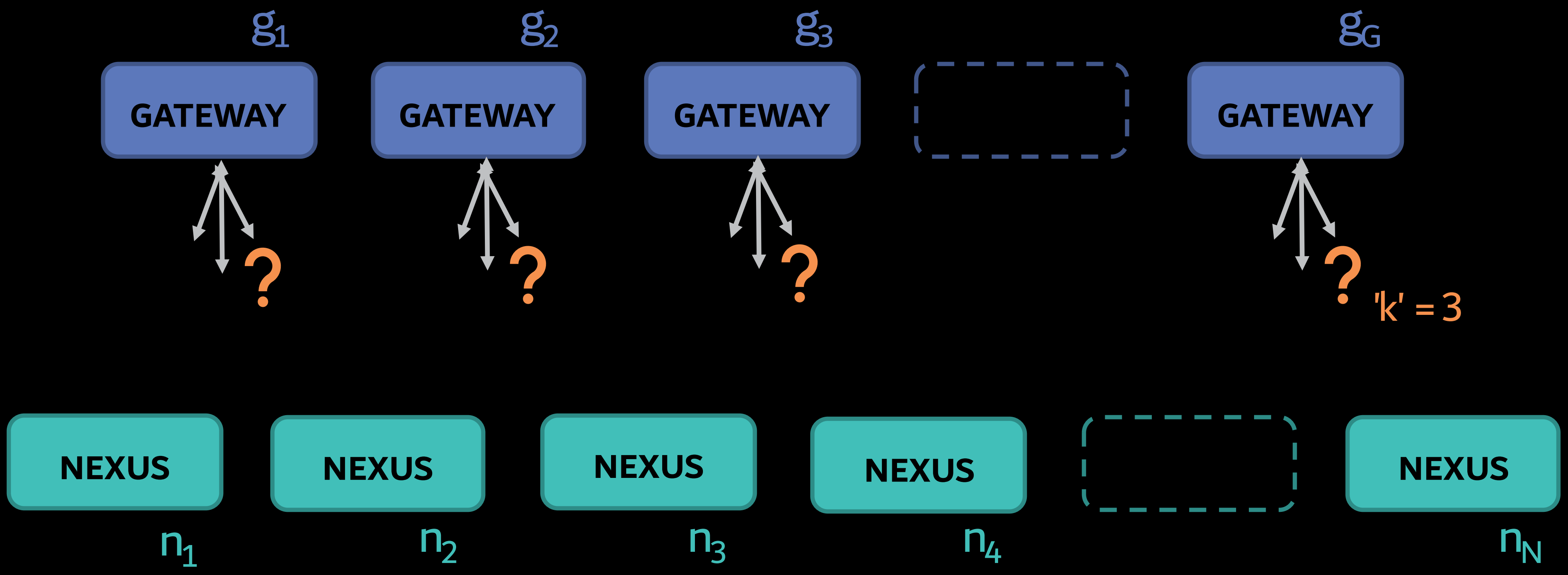
What is it?

Rendezvous hashing is an algorithm that allows clients to achieve distributed agreement on a set of 'k' options out of a possible set of 'n' options.

Rendezvous Hash

What is it?

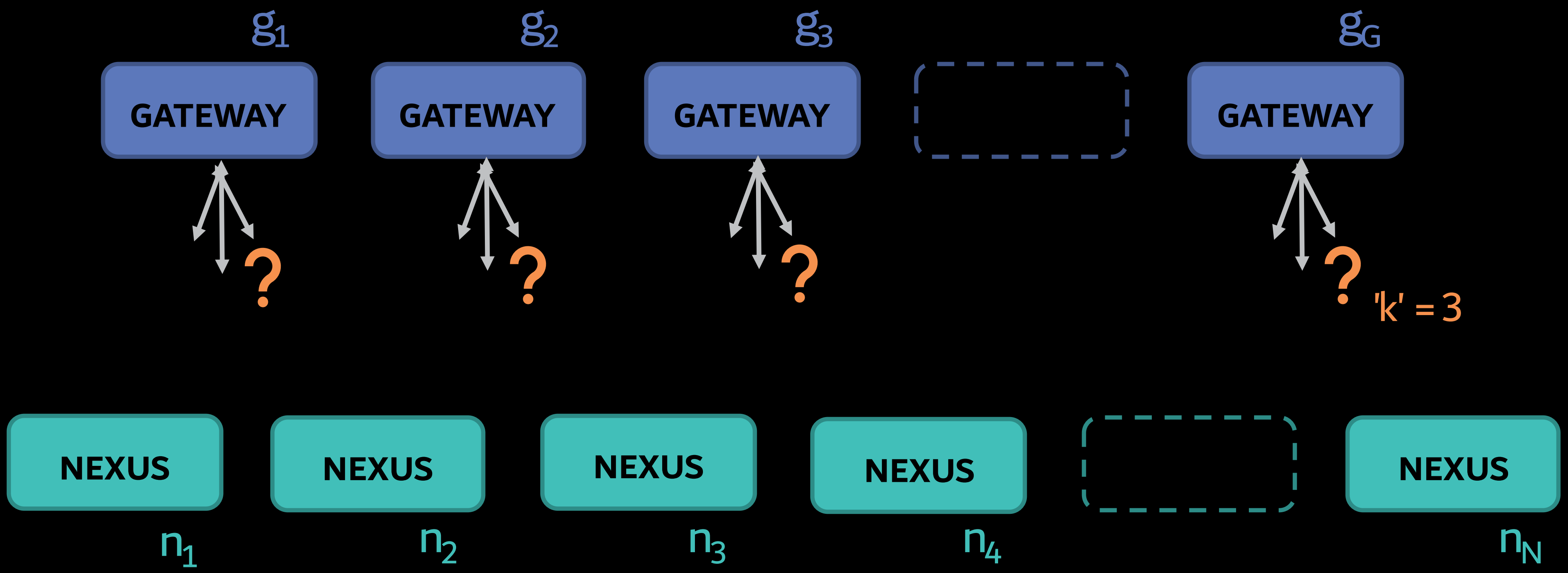
Rendezvous hashing is an algorithm that allows clients to achieve distributed agreement on a set of 'k' options out of a possible set of 'n' options.



Rendezvous Hash

How does it work?

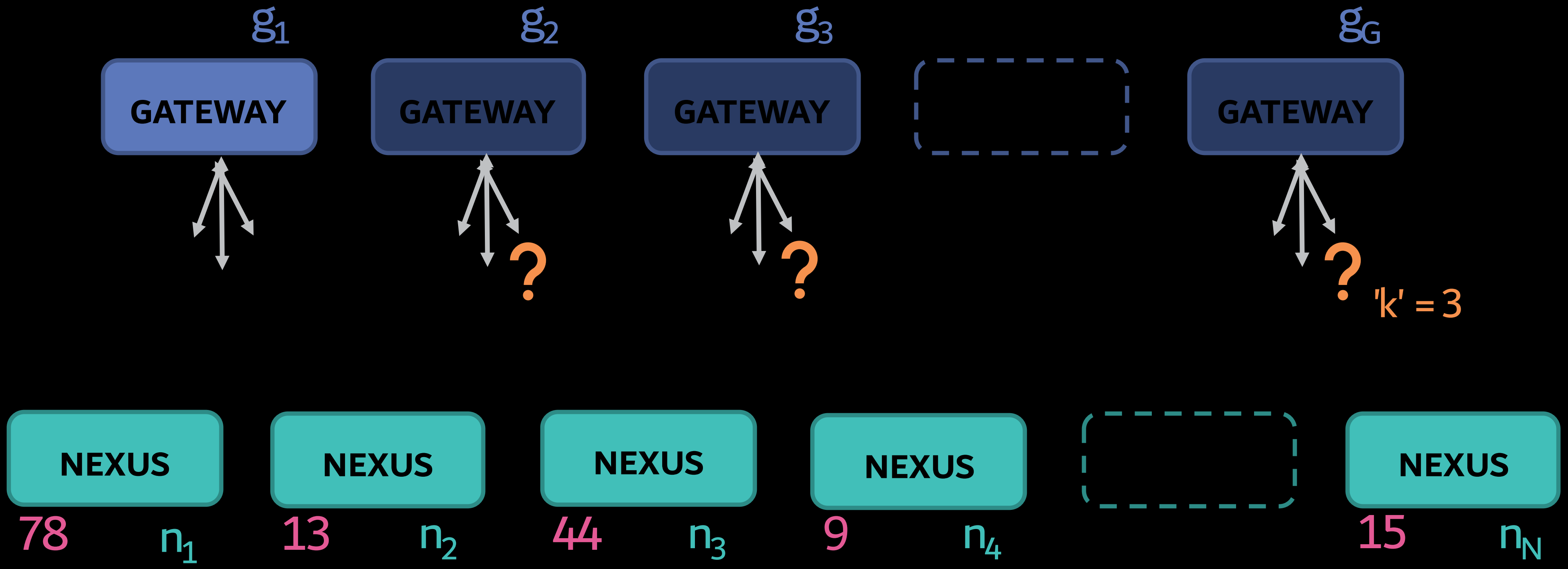
```
On Gateway $g_i$:
For $n_j$ in N:
    weight_of_$n_j$ = HASH( $g_i$ , $n_j$ )
Connect to nexuses with the top 'k' weights values
```



Rendezvous Hash

How does it work?

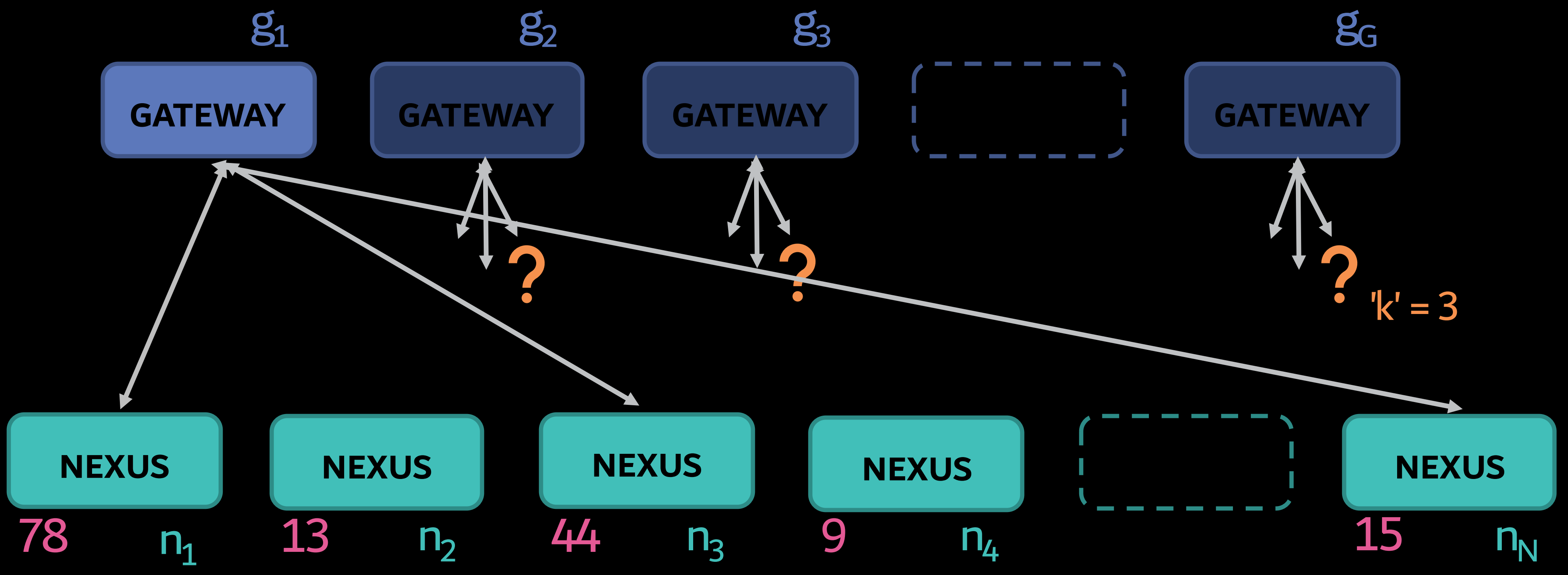
```
On Gateway $g_i$:
For $n_j$ in N:
    weight_of_$n_j$ = HASH( $g_i$ , $n_j$ )
Connect to nexuses with the top 'k' weights values
```



Rendezvous Hash

How does it work?

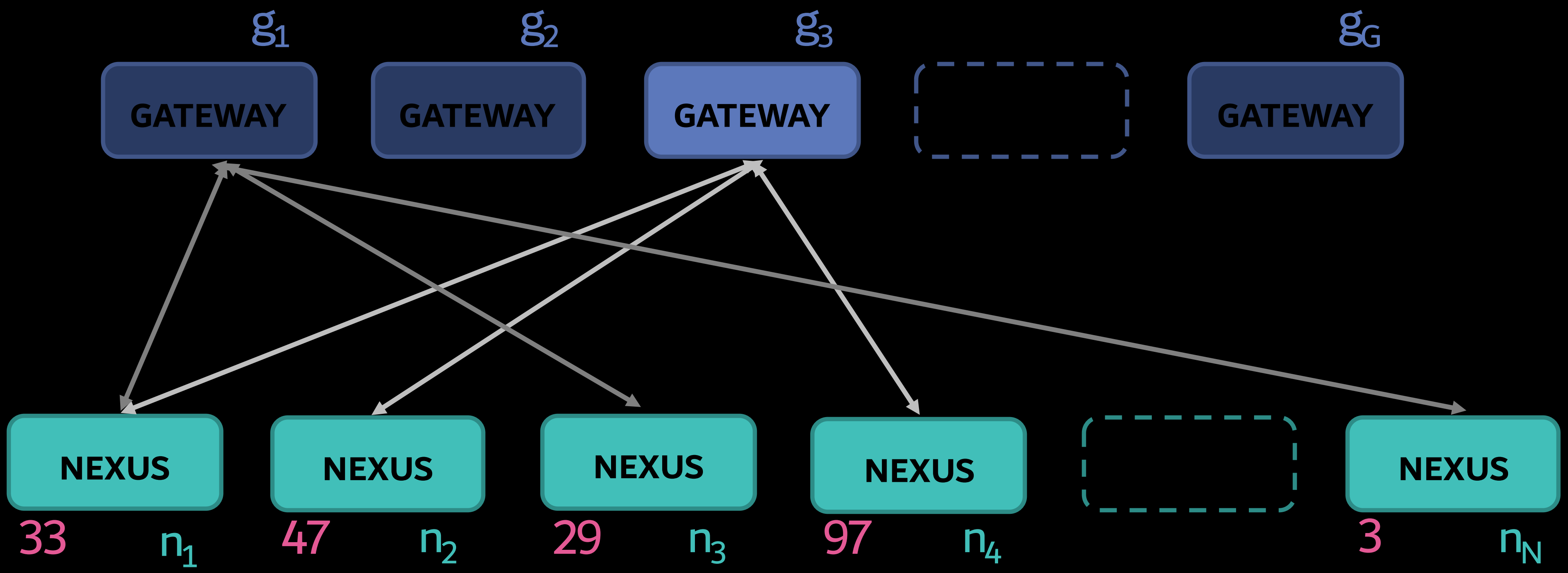
```
On Gateway $g_i$:
For $n_j$ in N:
    weight_of_$n_j$ = HASH( $g_i$ , $n_j$ )
Connect to nexuses with the top 'k' weights values
```



Rendezvous Hash

How does it work?

```
On Gateway $g_i$:
For $n_j$ in N:
    $weight\_of\_n_j$ = HASH( $g_i$ , $n_j$ )
Connect to nexuses with the top 'k' weights values
```



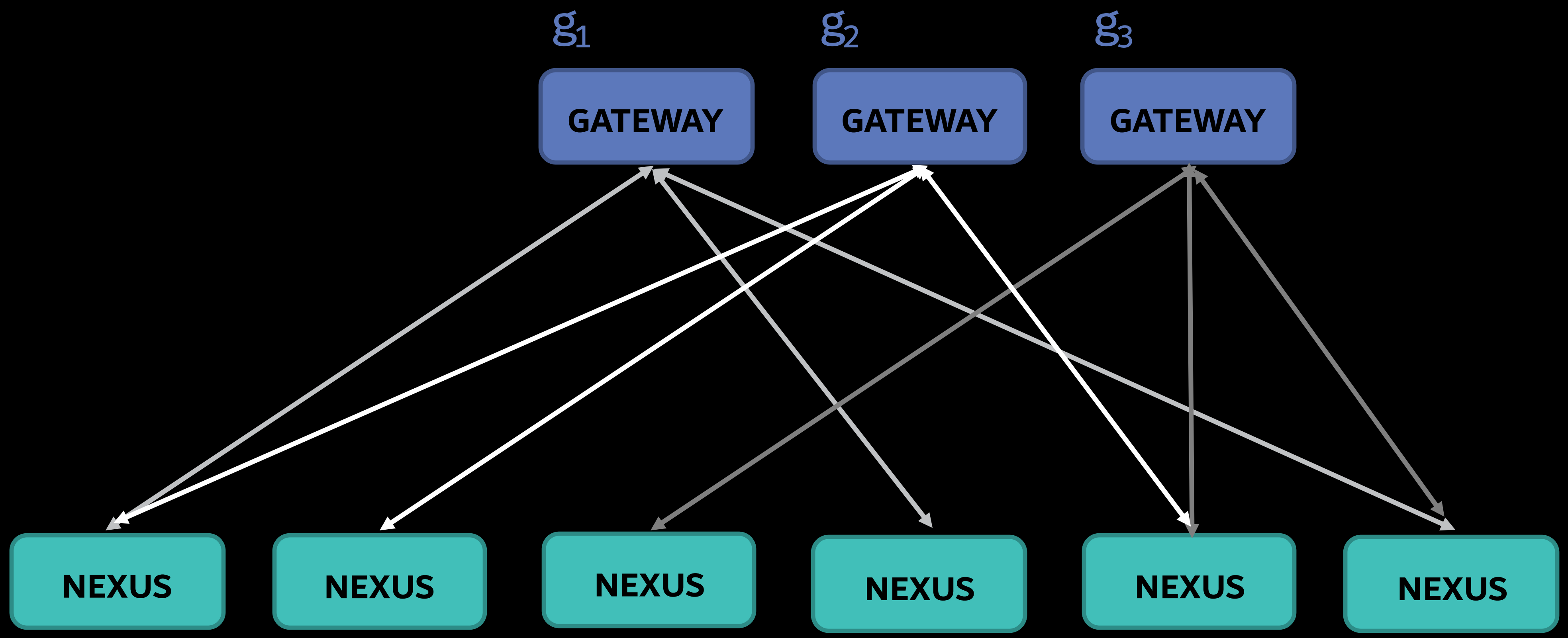
Rendezvous Hash

A Note on the Hash and Distribution

```
On Gateway $g_i$:
For $n_j$ in N:
    weight_of_$n_j$ = HASH( $g_i$ , $n_j$ )
Connect to nexuses with the top 'k' weights values
```

Rendezvous Hash

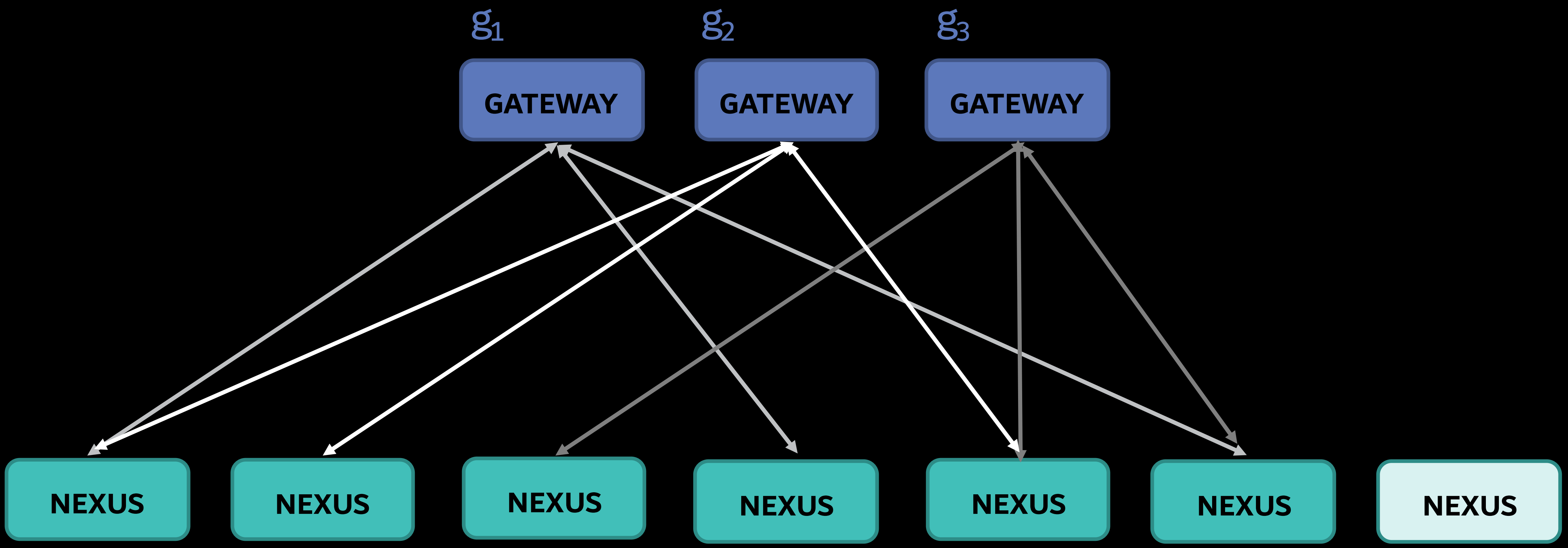
Minimal Disruption



g_1	88	12	37	78	45	58
g_2	99	98	3	9	21	6
g_3	6	23	52	26	49	45

Rendezvous Hash

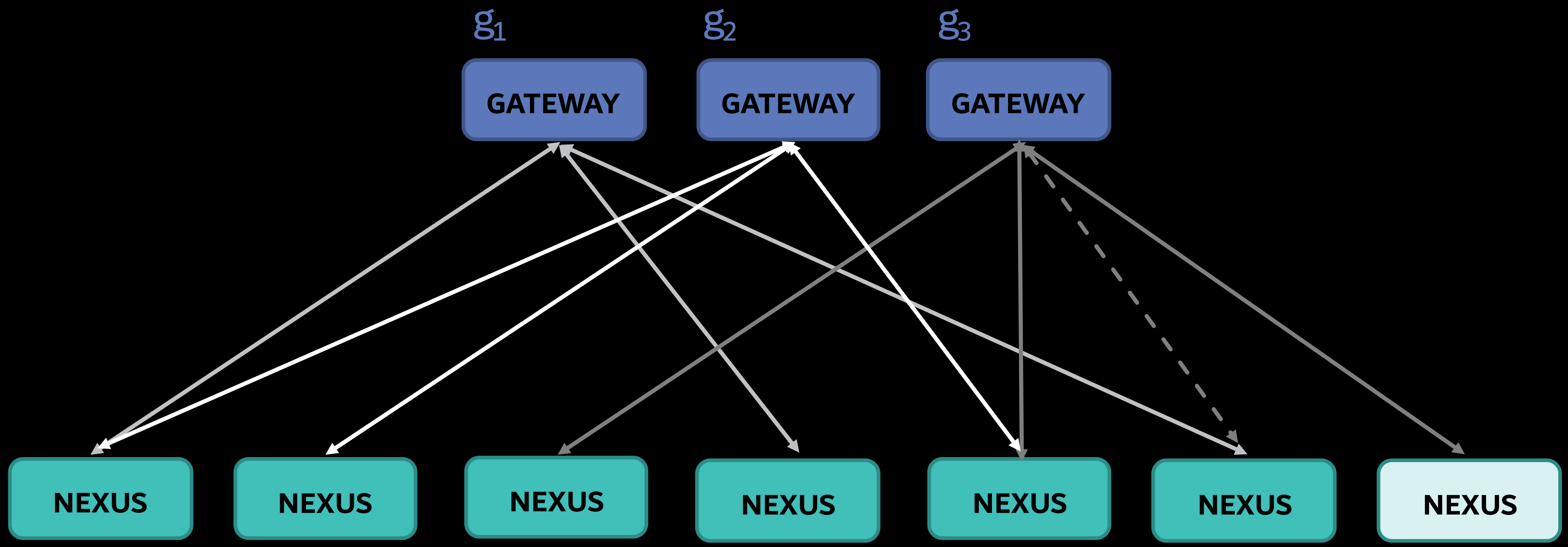
Minimal Disruption



g_1	88	12	37	78	45	58	3
g_2	99	98	3	9	21	6	18
g_3	6	23	52	26	49	45	73

Rendezvous Hash

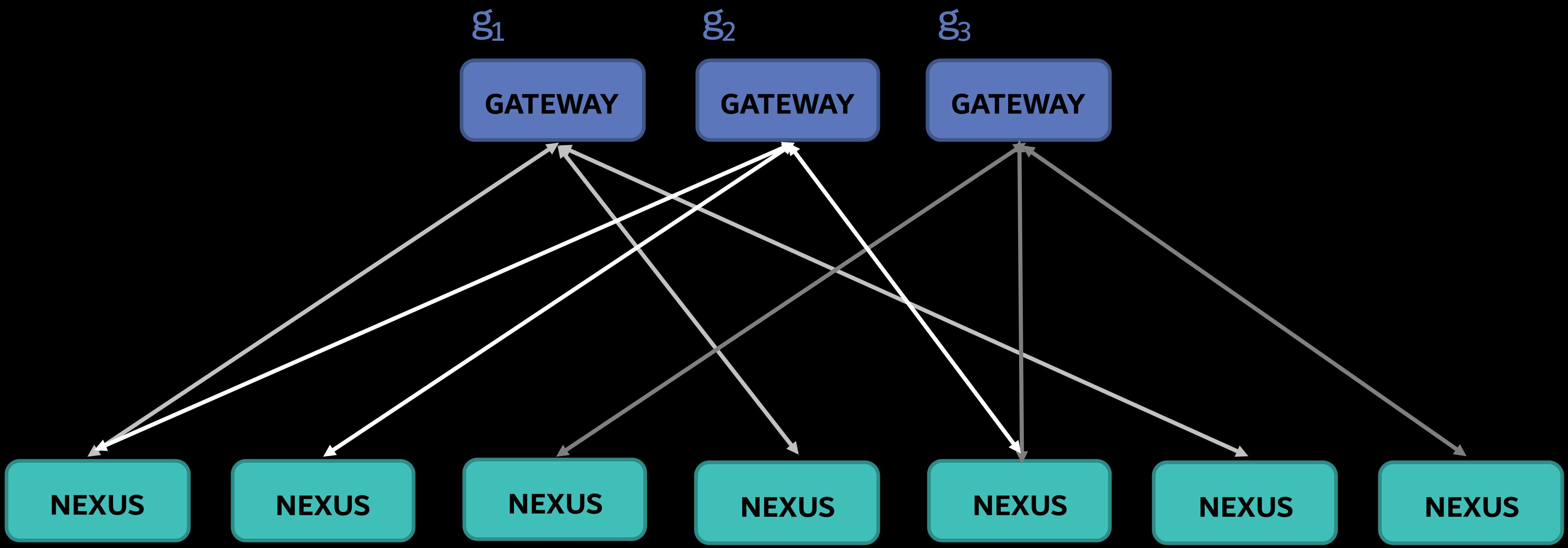
Minimal Disruption



g_1	88	12	37	78	45	58	3
g_2	99	98	3	9	21	6	18
g_3	6	23	52	26	49	45	73

Rendezvous Hash

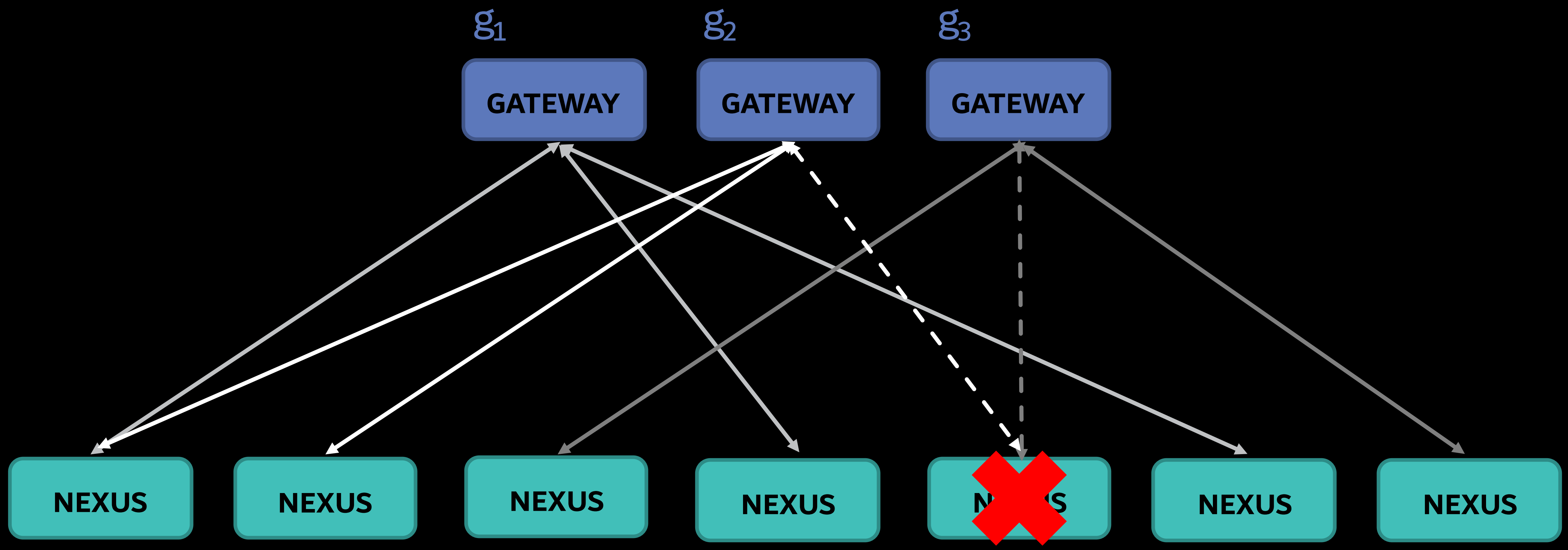
Minimal Disruption



g_1	88	12	37	78	45	58	3
g_2	99	98	3	9	21	6	18
g_3	6	23	52	26	49	45	73

Rendezvous Hash

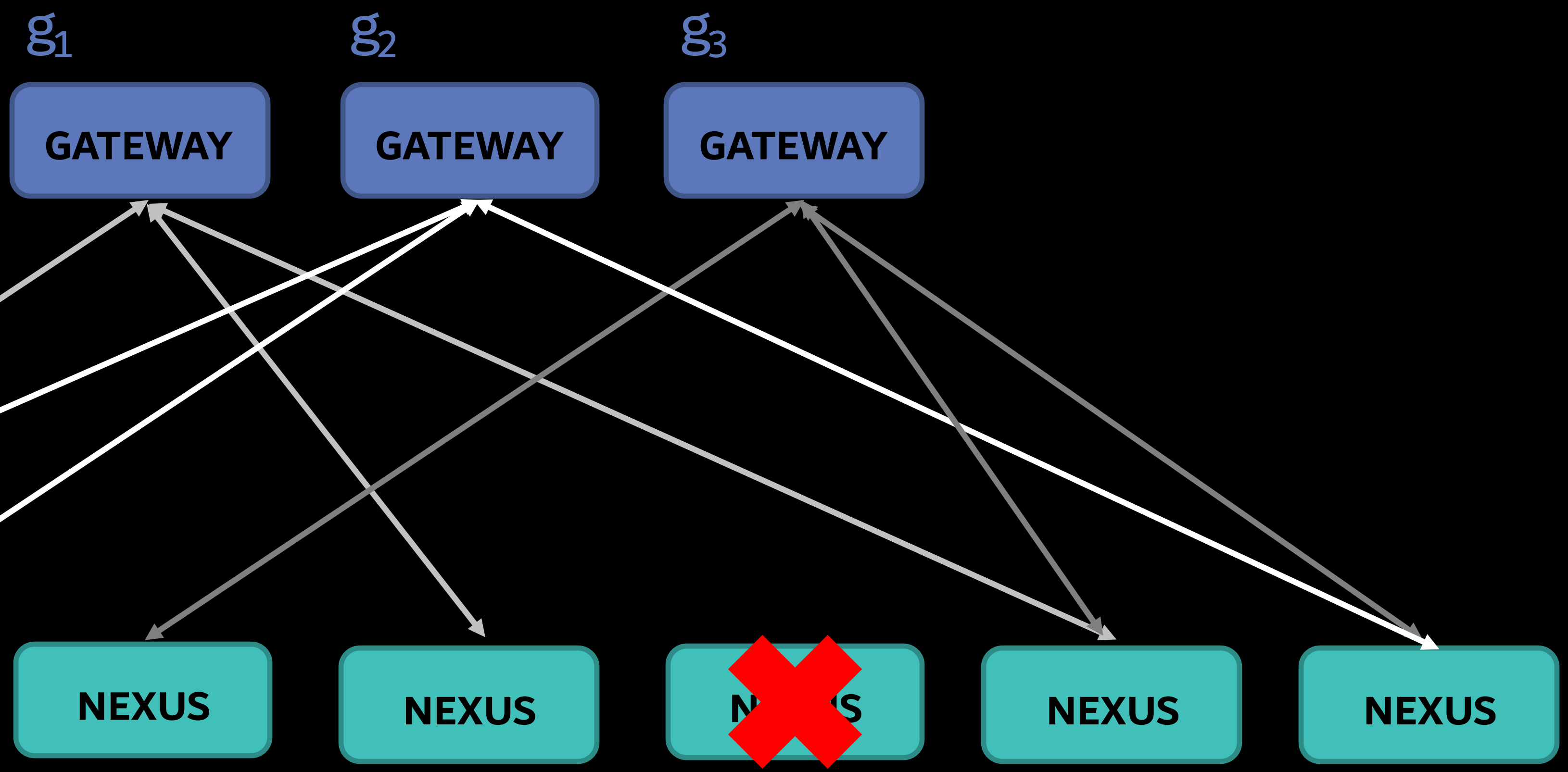
Minimal Disruption



g_1	88	12	37	78	~~45~~	58	3
g_2	99	98	3	9	~~21~~	6	18
g_3	6	23	52	26	~~49~~	45	73

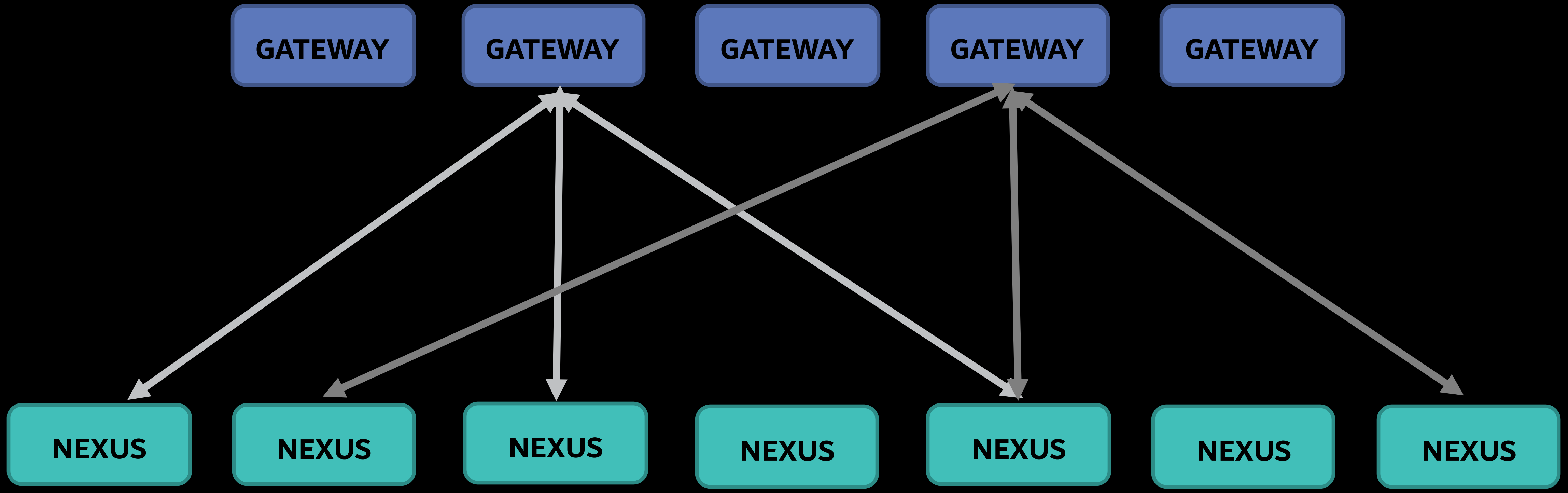
Rendezvous Hash

Minimal Disruption



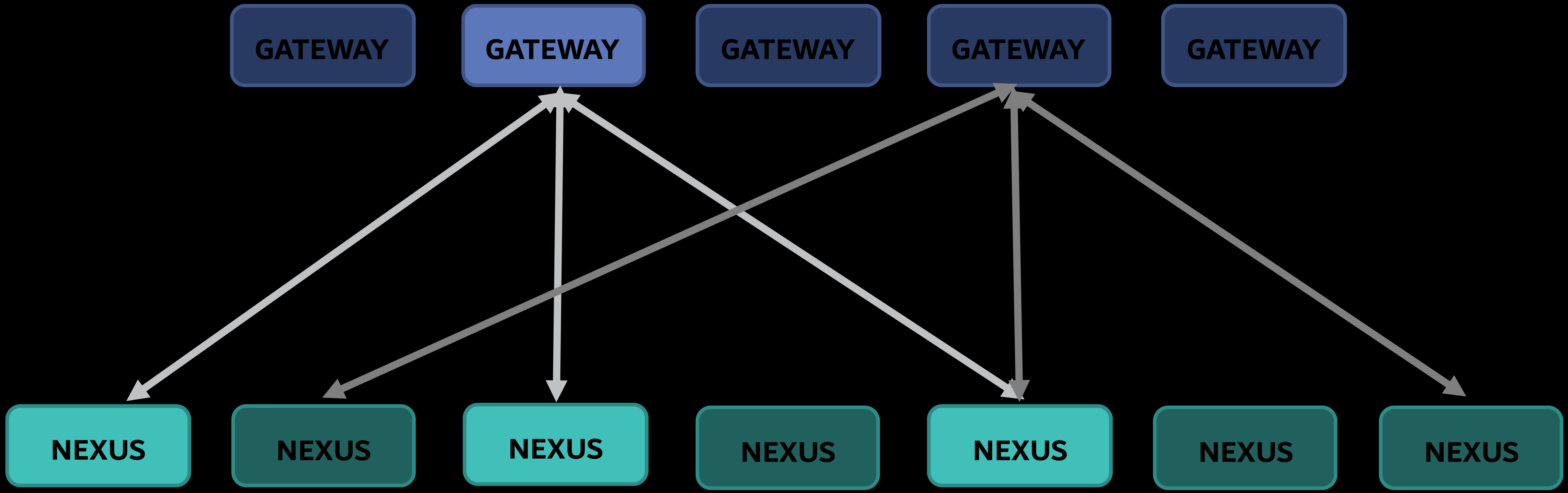
g_1	88	12	37	78	~~45~~	58	3
g_2	99	98	3	9	~~21~~	6	18
g_3	6	23	52	26	~~49~~	45	73

Off-box Architecture



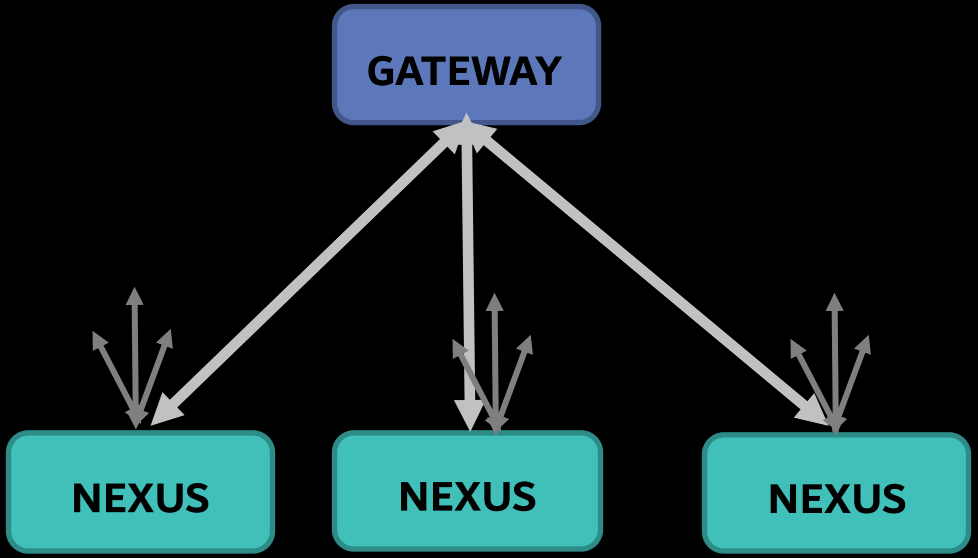
Off-box Architecture

Traffic Routing



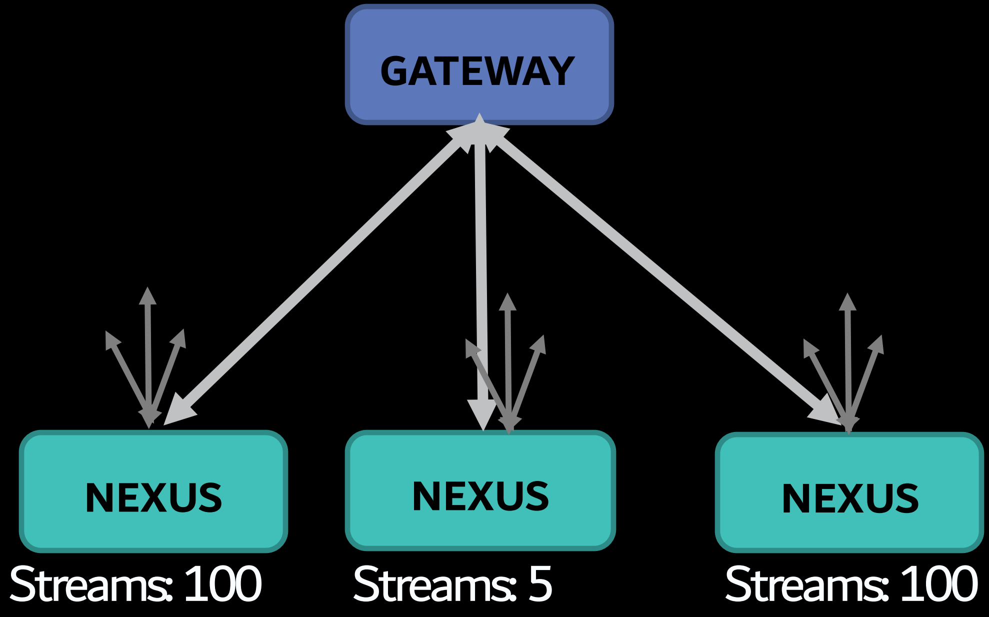
Off-box Architecture

Traffic Routing



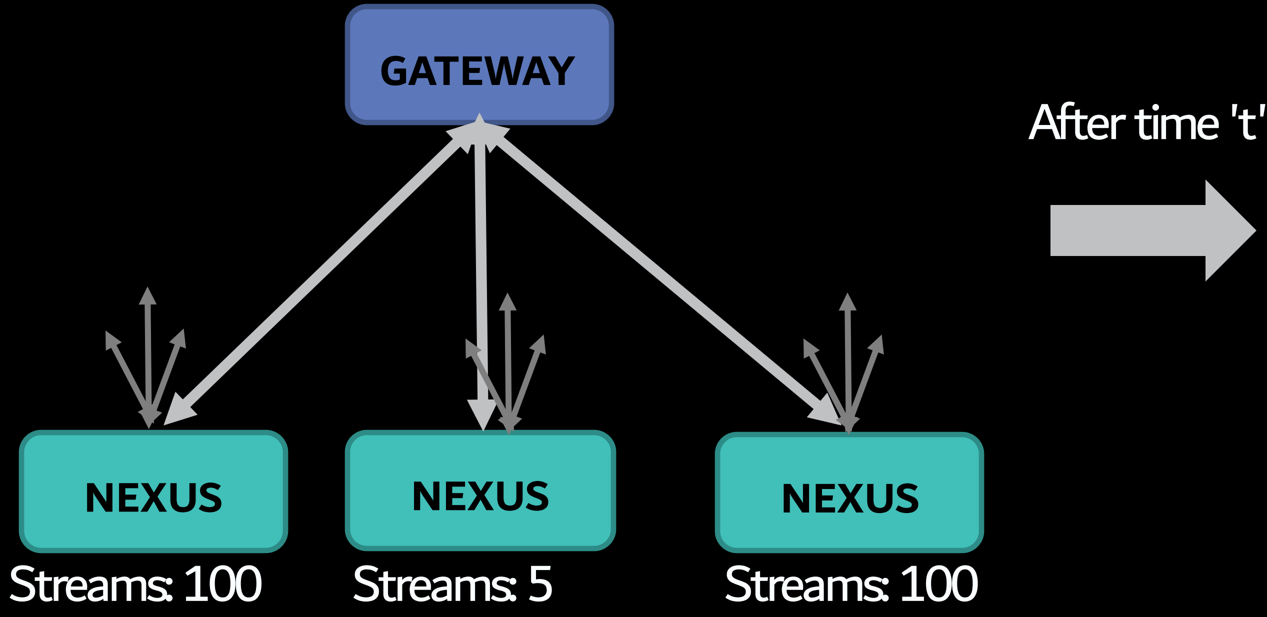
Off-box Architecture

Traffic Routing



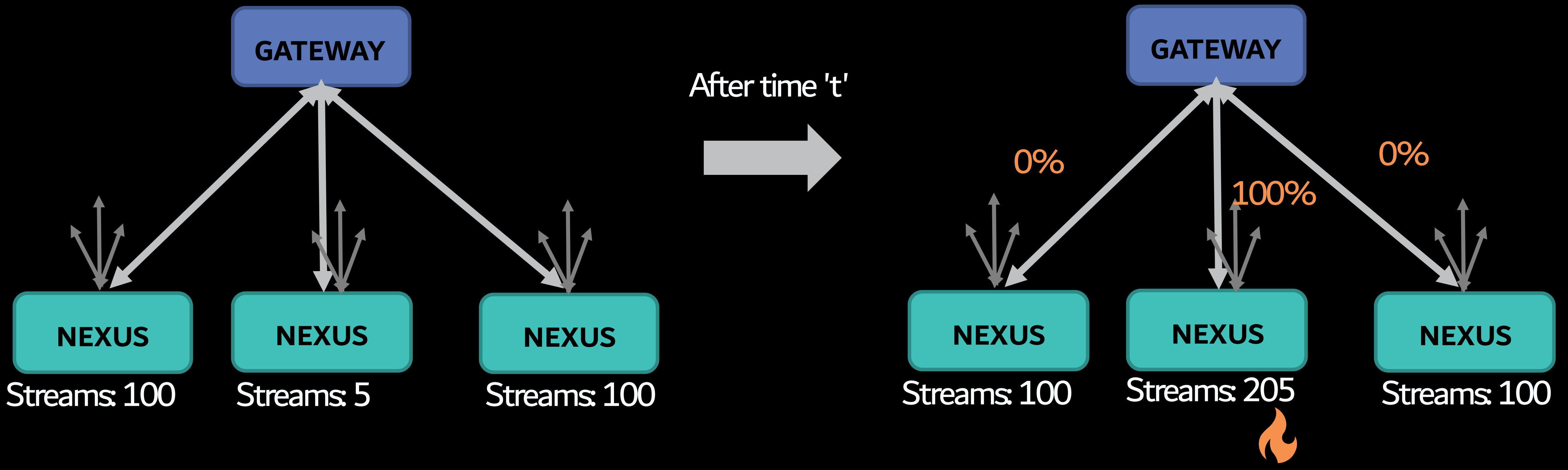
Off-box Architecture

Traffic Routing



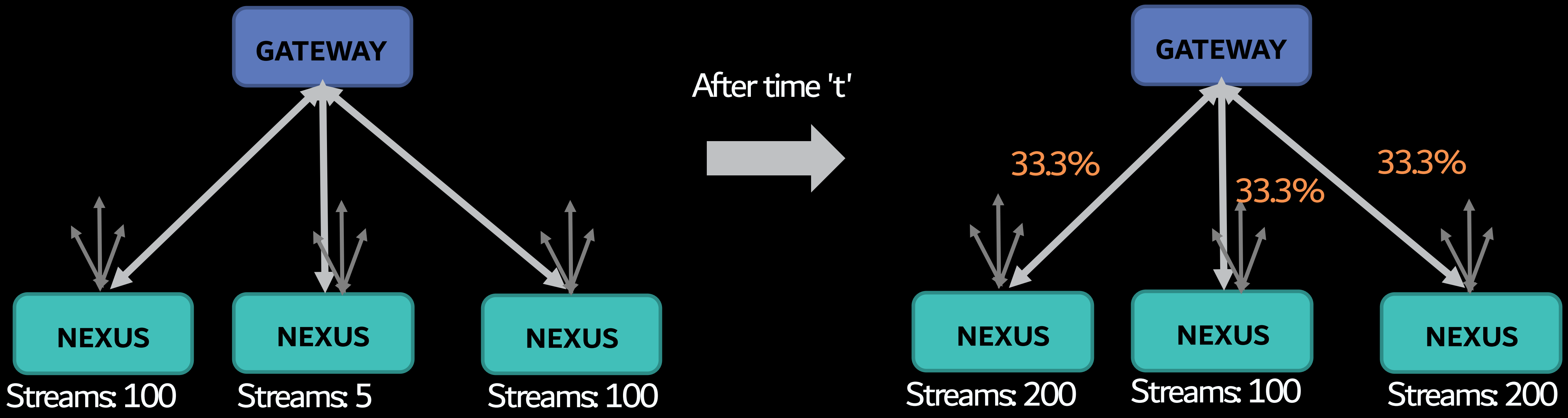
Off-box Architecture

Traffic Routing: Naïve solution – pick lowest loaded



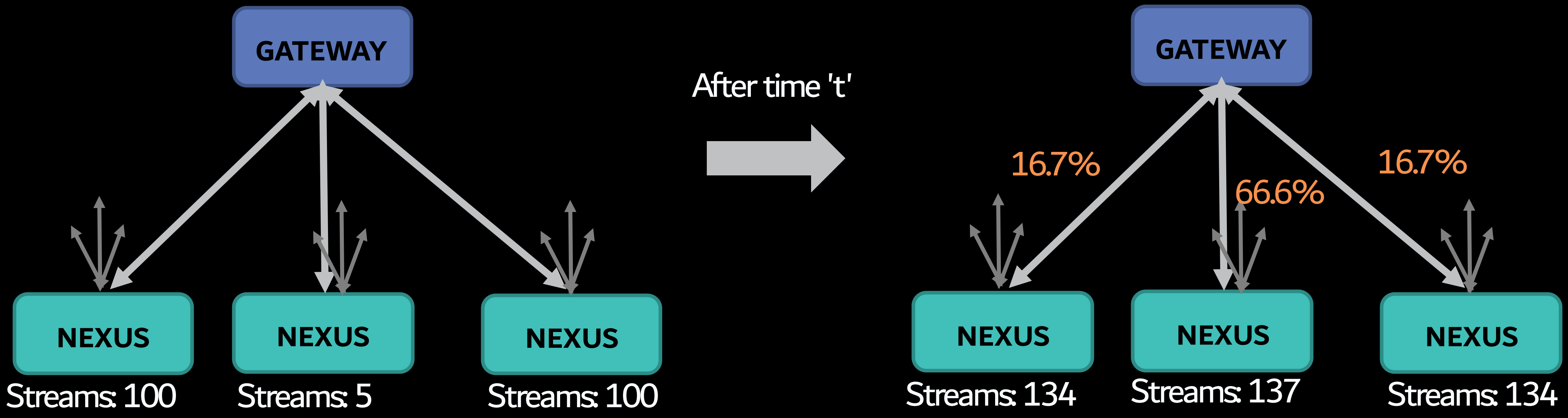
Off-box Architecture

Traffic Routing: Naïve solution – Random host



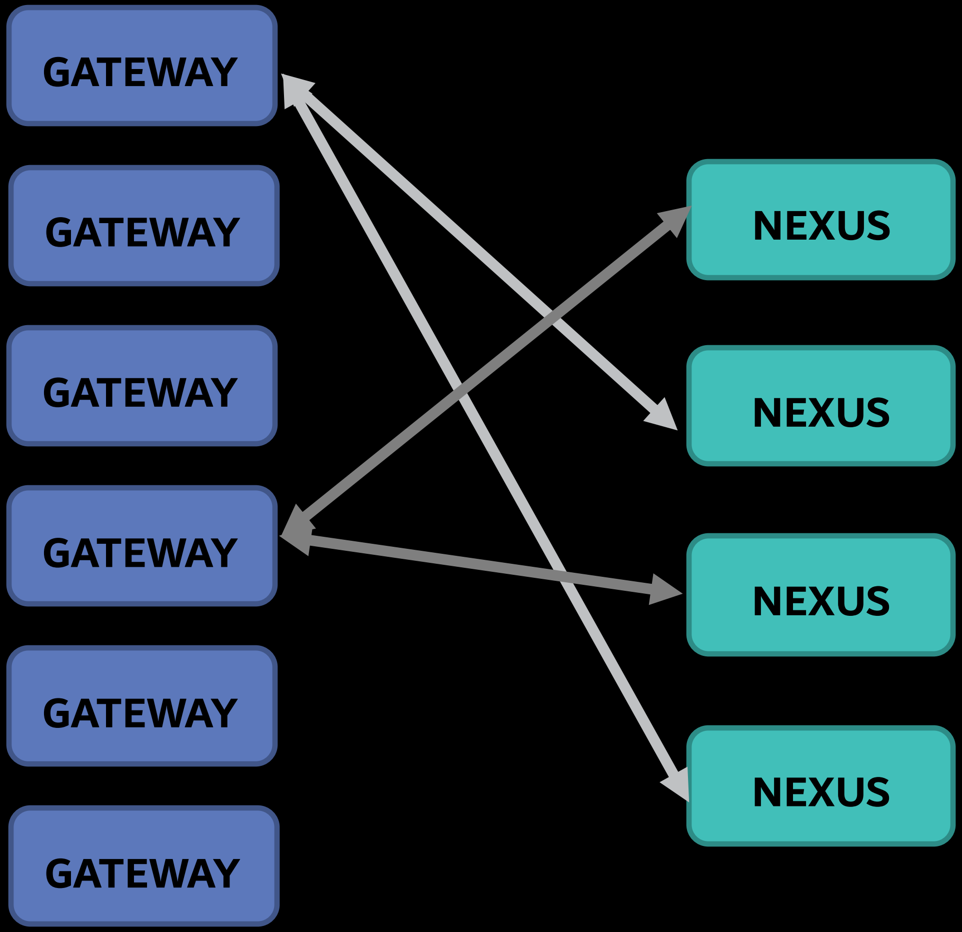
Off-box Architecture

Traffic Routing: Lower of random two hosts



Off-box Architecture

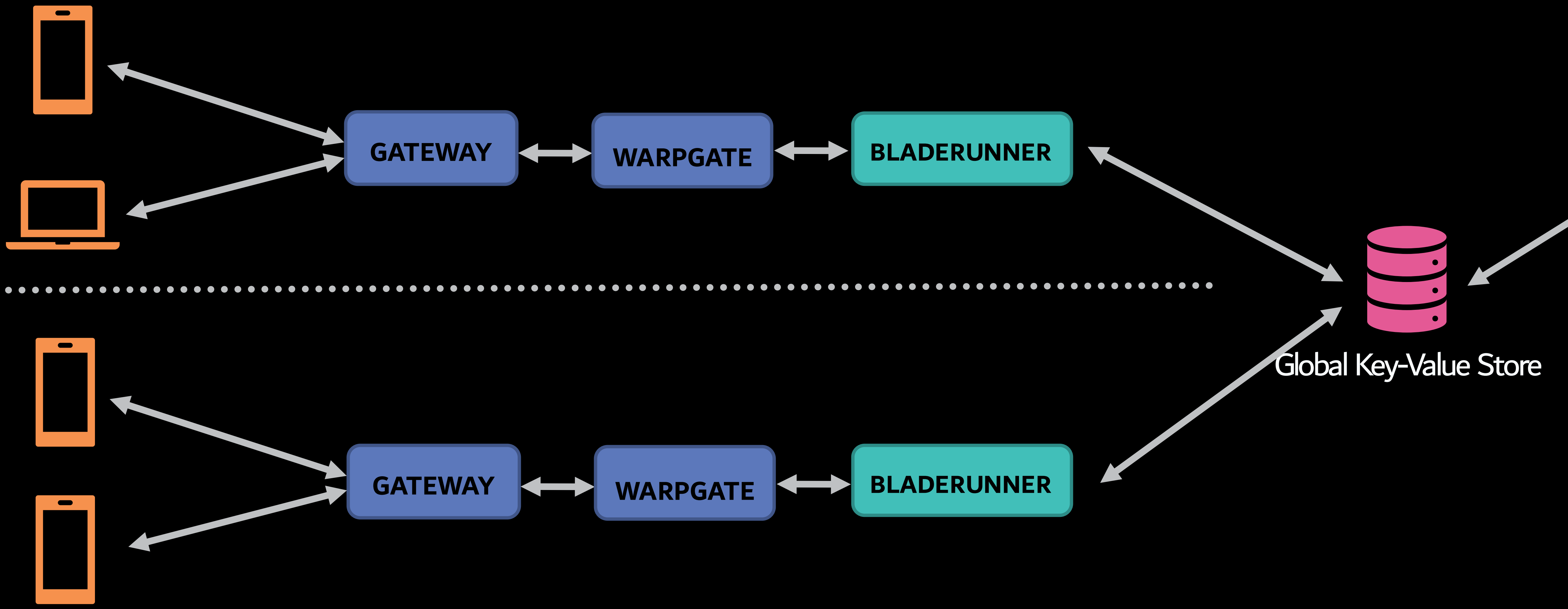
Advantages



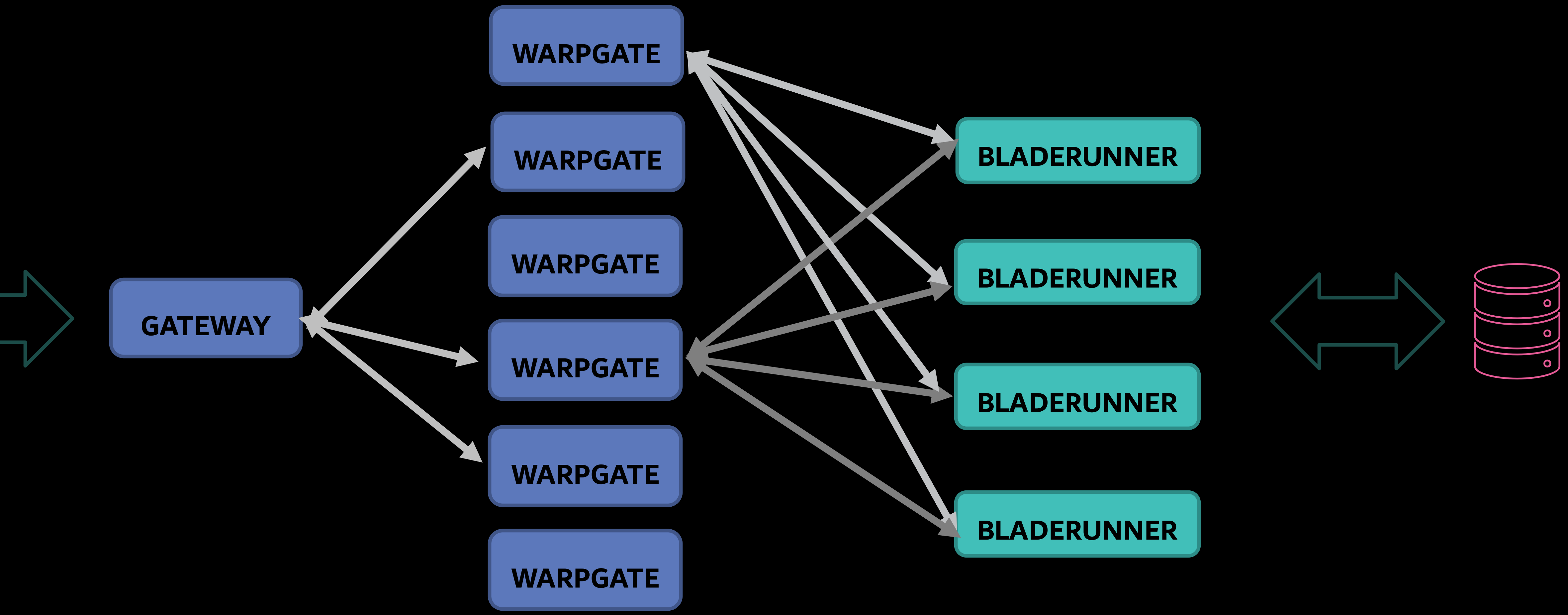
1. Customer Traffic Isolation
2. High Fault Tolerance
3. Dynamic Load Balancing

Full-Mesh Architecture

5000 ft view

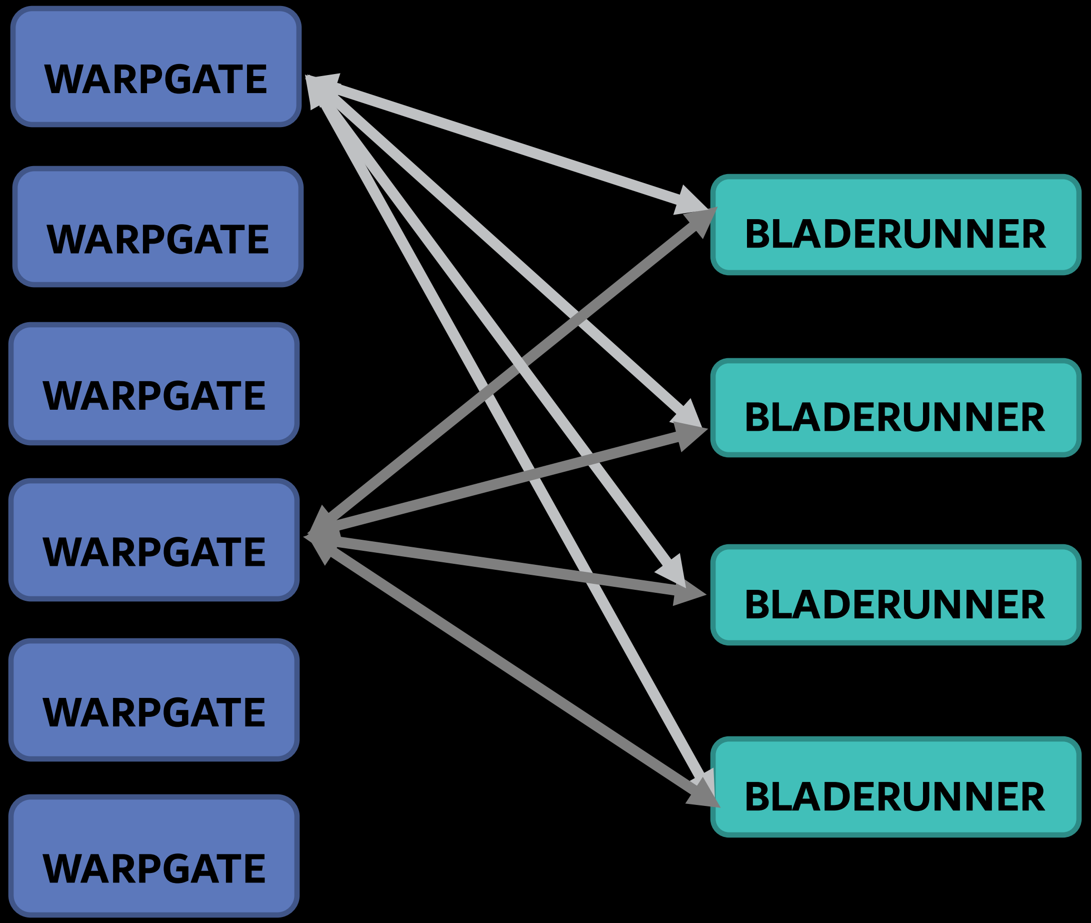

GATEWAY
WARPGATE
BLADERUNNER
GATEWAY
WARPGATE
BLADERUNNER
Global Key-Value Store

Full-Mesh Architecture



Full-Mesh Architecture

New Routing Capabilities

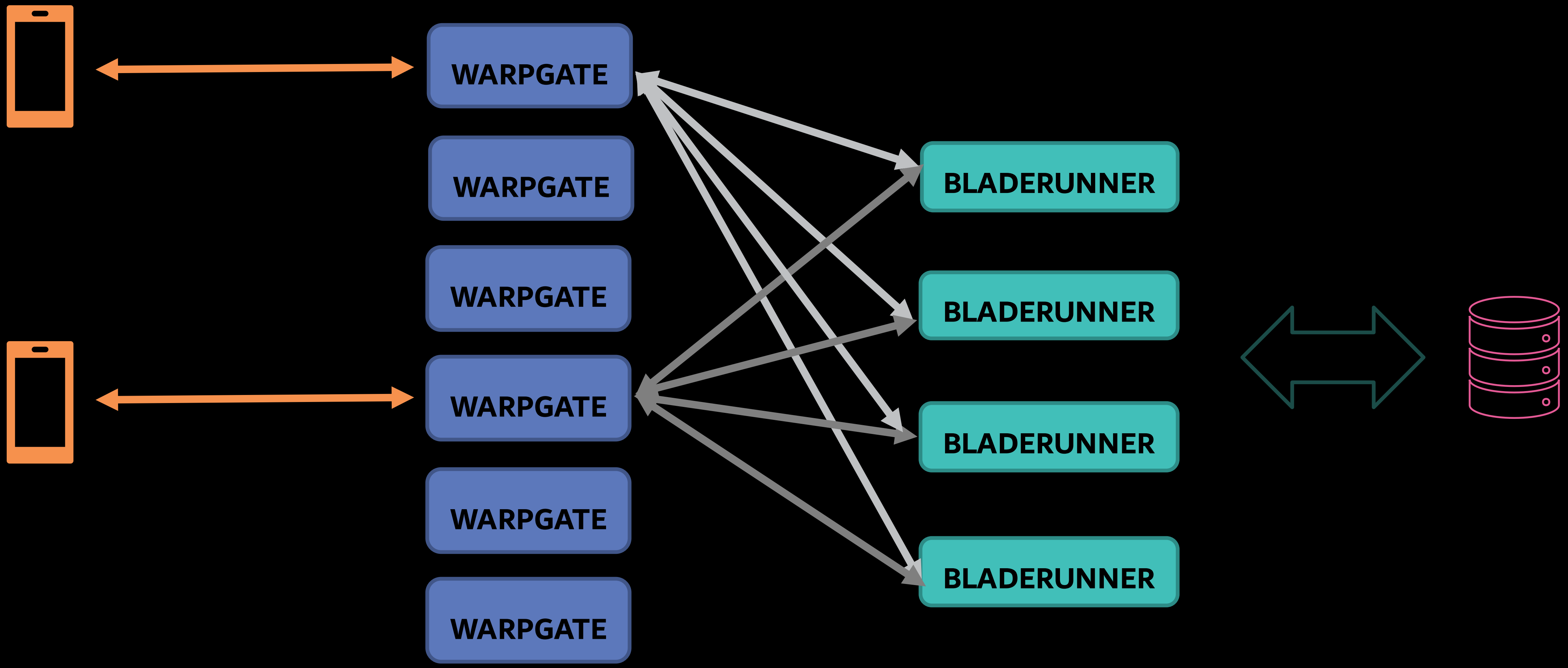


1. Sticky Routing

2. Virtual Isolation

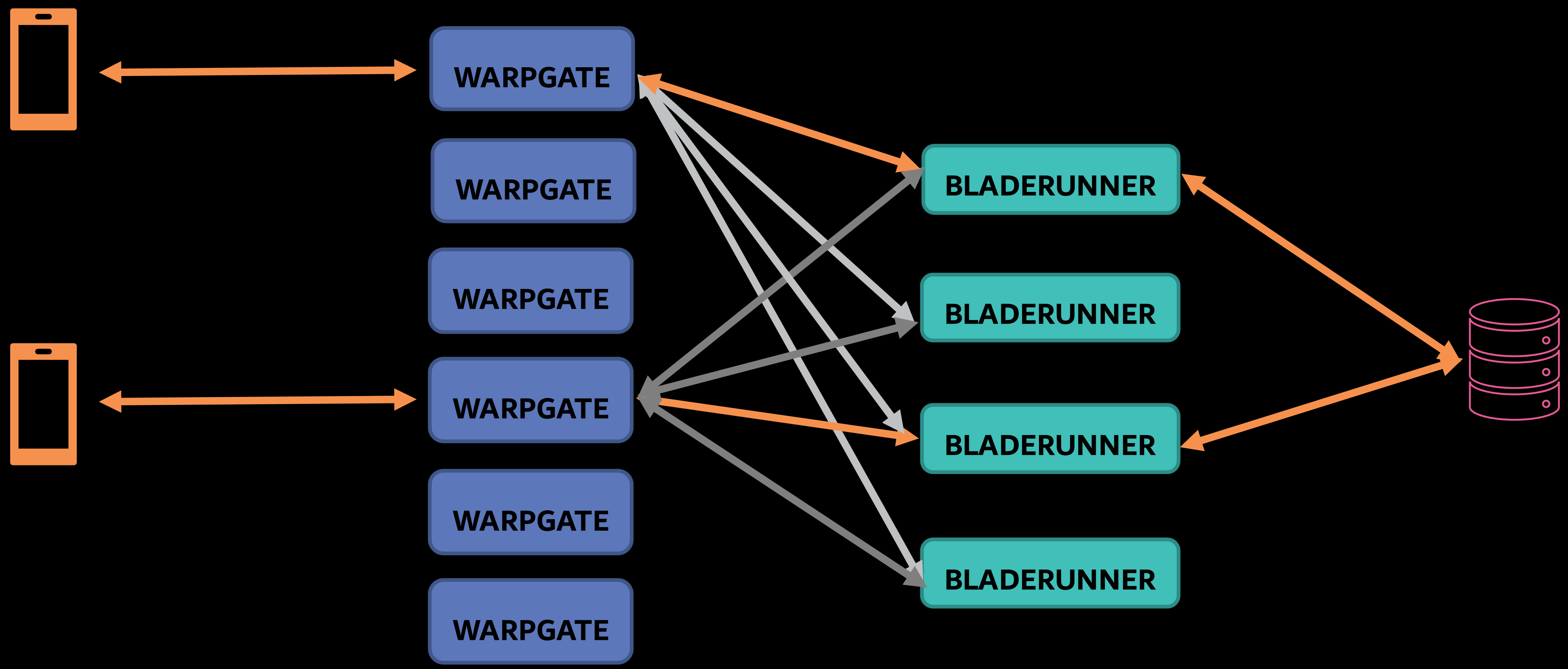
Full-Mesh Architecture

Sticky Routing



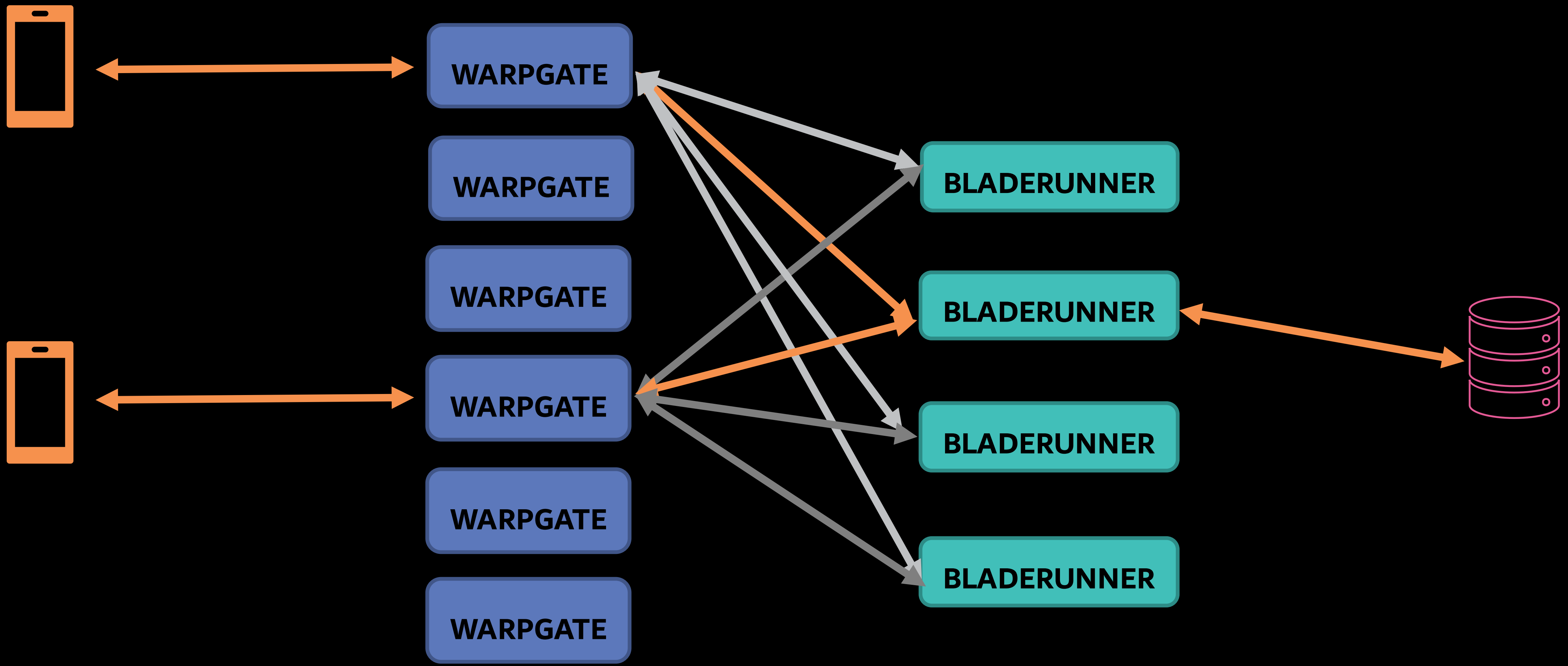
Full-Mesh Architecture

Sticky Routing



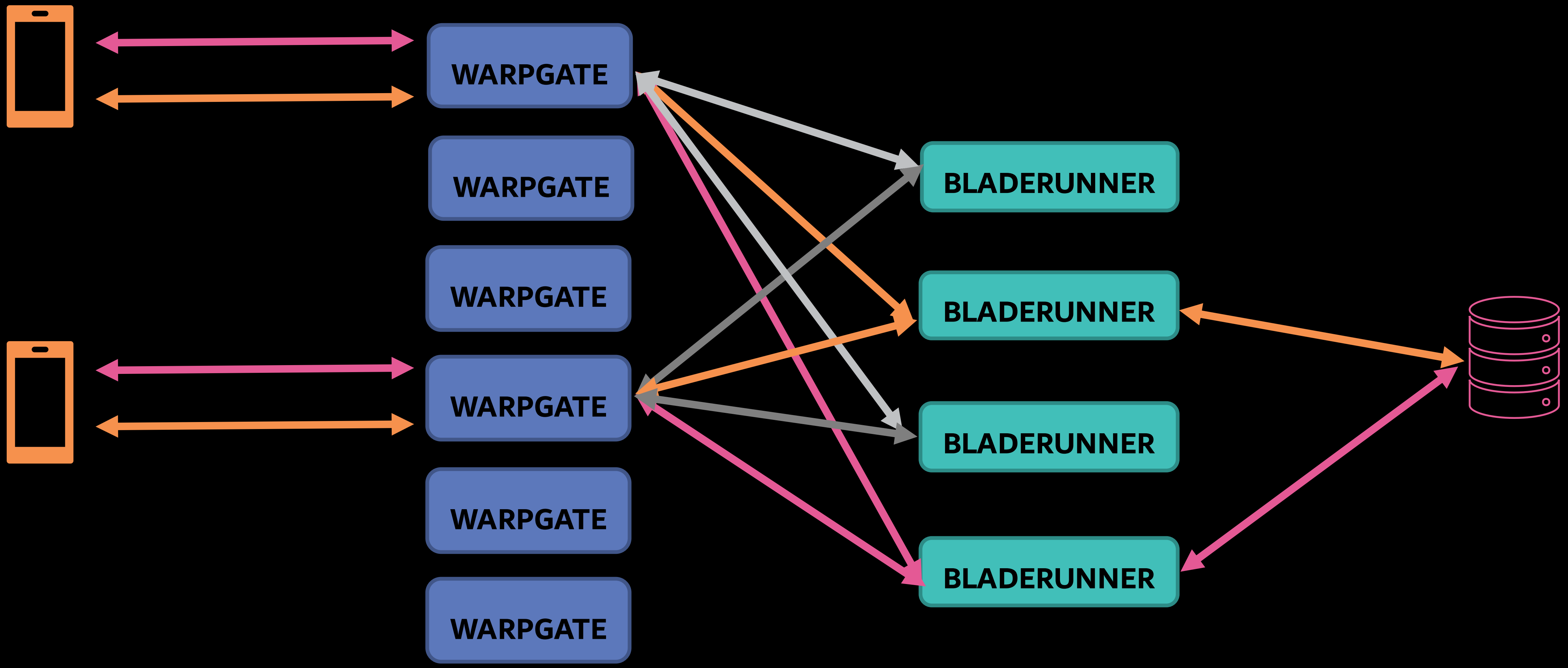
Full-Mesh Architecture

Sticky Routing



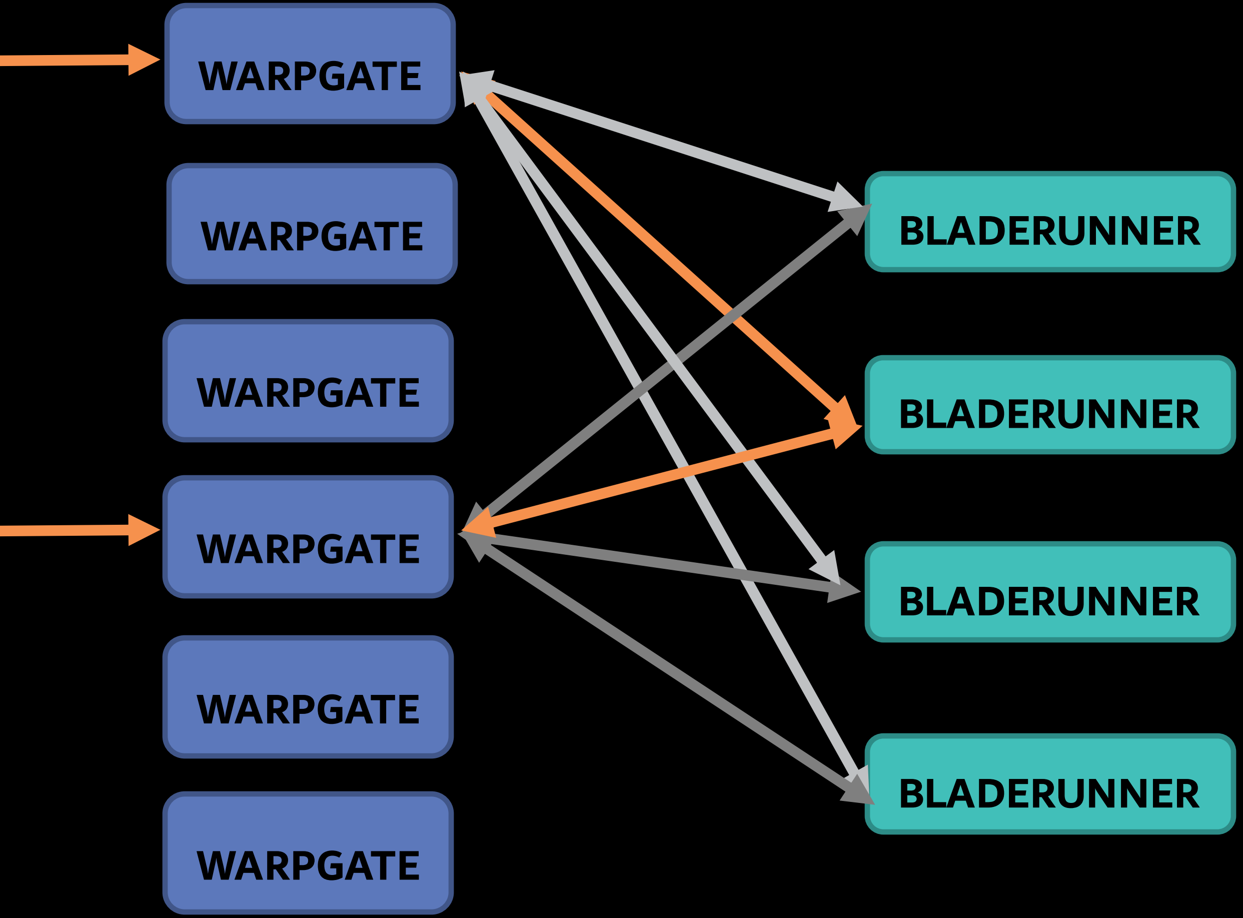
Full-Mesh Architecture

Sticky Routing



Full-Mesh Architecture

Sticky Routing : Wins



15-20% reduction of requests to the Global Key-Value Store

5-10% Memory and CPU Savings on Bladerunner

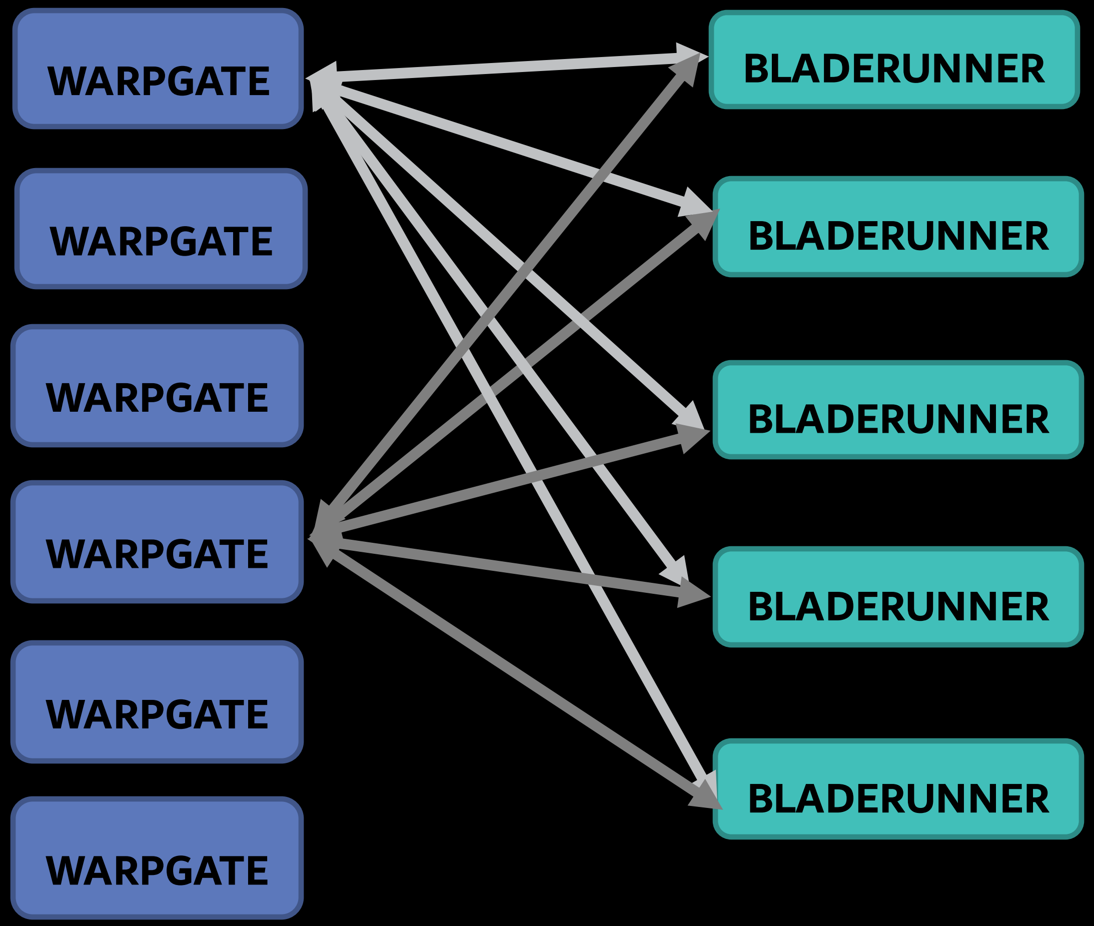
Full-Mesh Architecture

Sticky Routing

How?

Full-Mesh Architecture

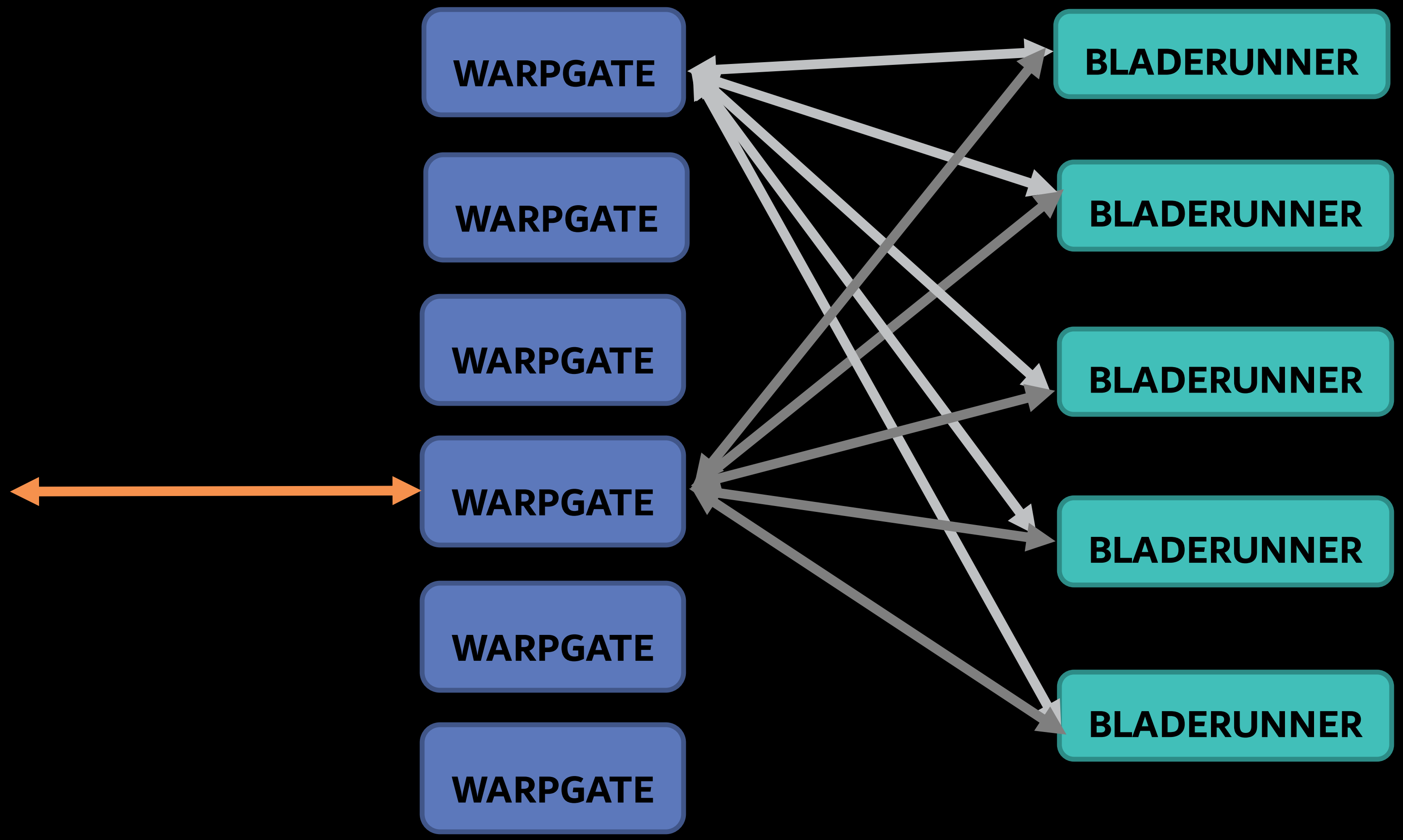
Sticky Routing : Rendezvous Hashing



Full-Mesh Architecture

Sticky Routing : Rendezvous Hashing

On any WarpGate:
Say we are routing feed f_i



Full-Mesh Architecture

Sticky Routing : Rendezvous Hashing

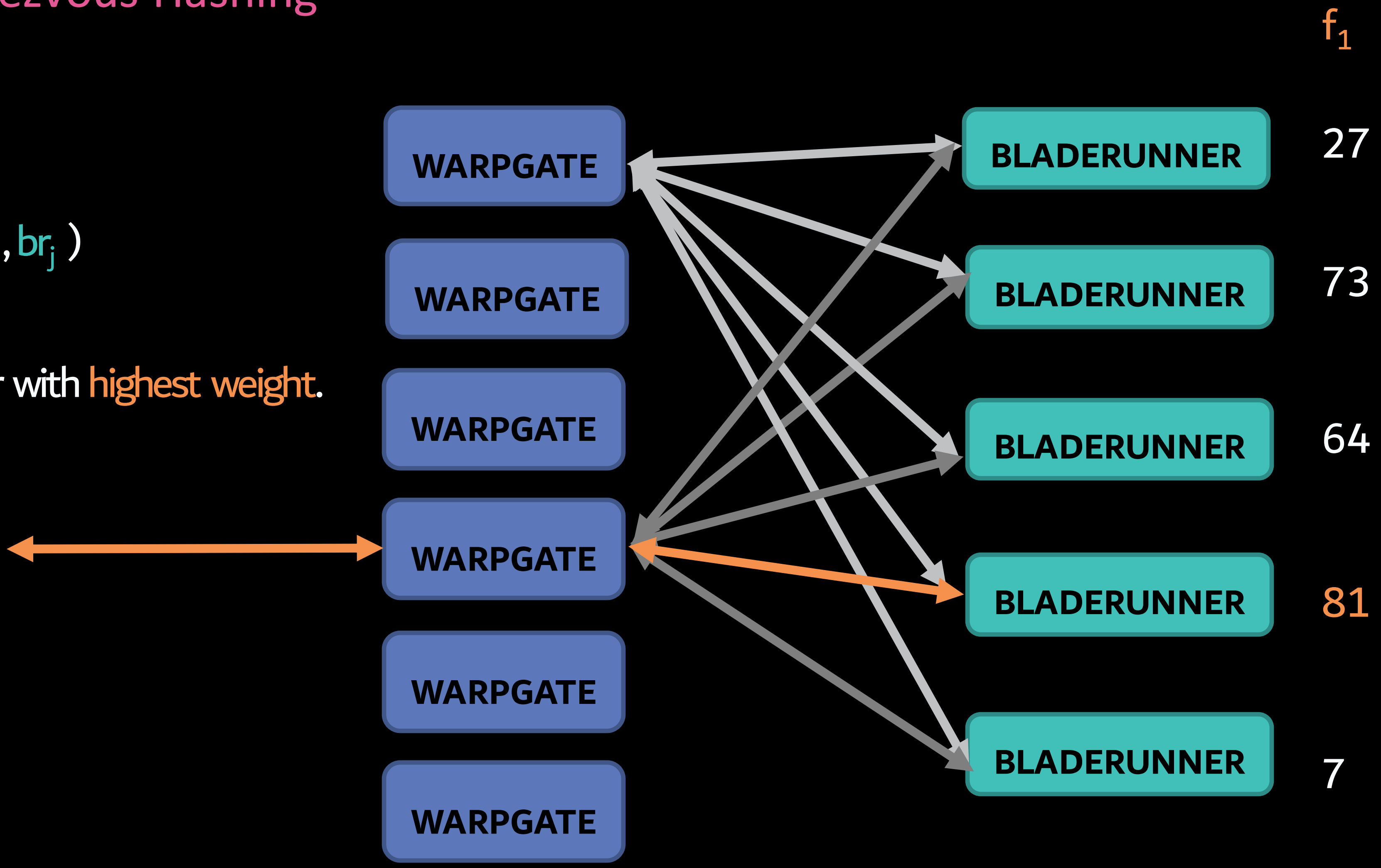
On any WarpGate:

Say we are routing feed f_i

For br_j in Bladerunner:

  $weight\_of\_br_j$ = HASH(f_i , br_j)

Finally, route to Bladerunner with highest weight.



Full-Mesh Architecture

Sticky Routing : Rendezvous Hashing

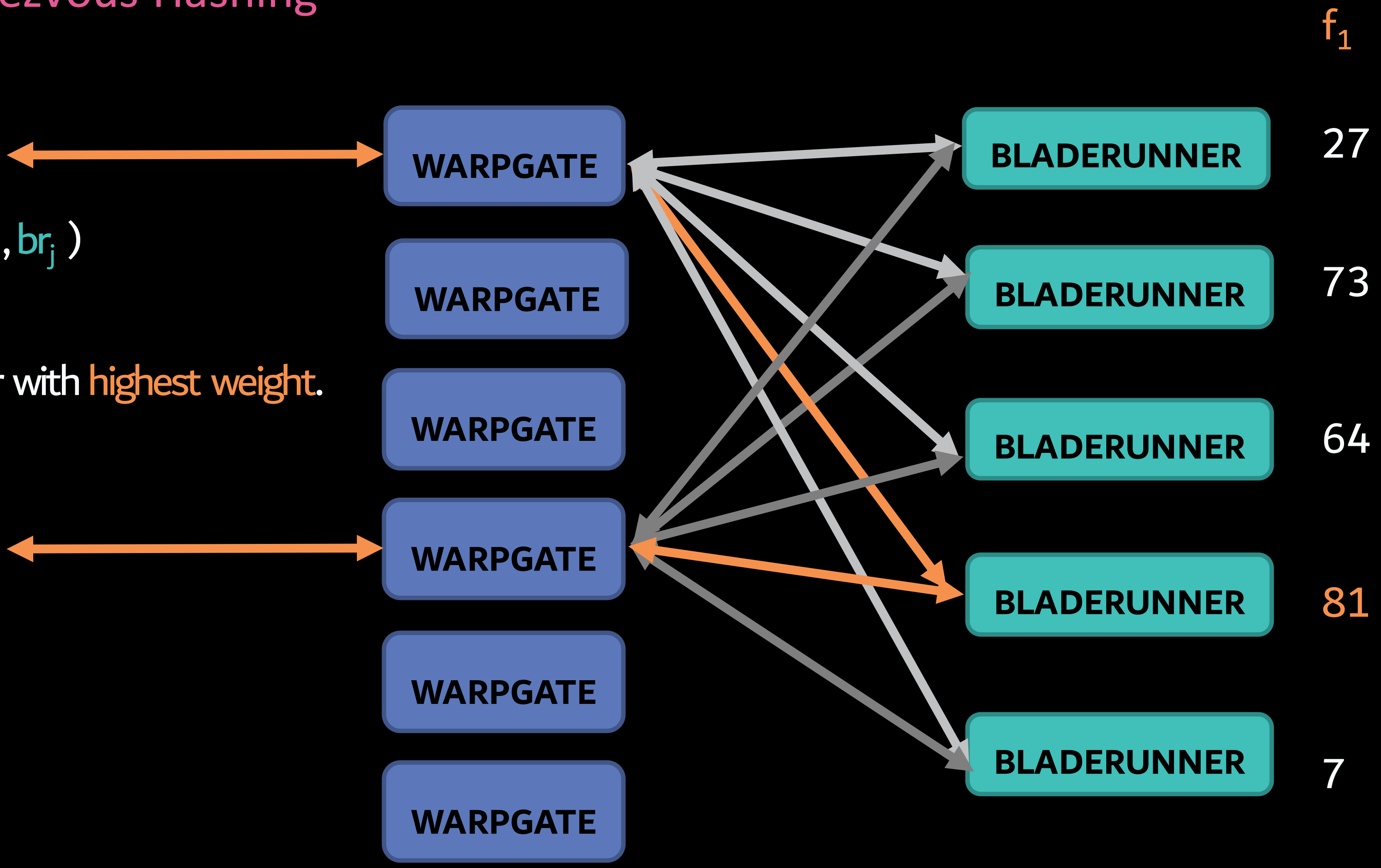
On any WarpGate:

Say we are routing feed f_i

For br_j in Bladerunner:

  $weight\_of\_br_j$ = HASH(f_i , br_j)

Finally, route to Bladerunner with highest weight.



Full-Mesh Architecture

Sticky Routing

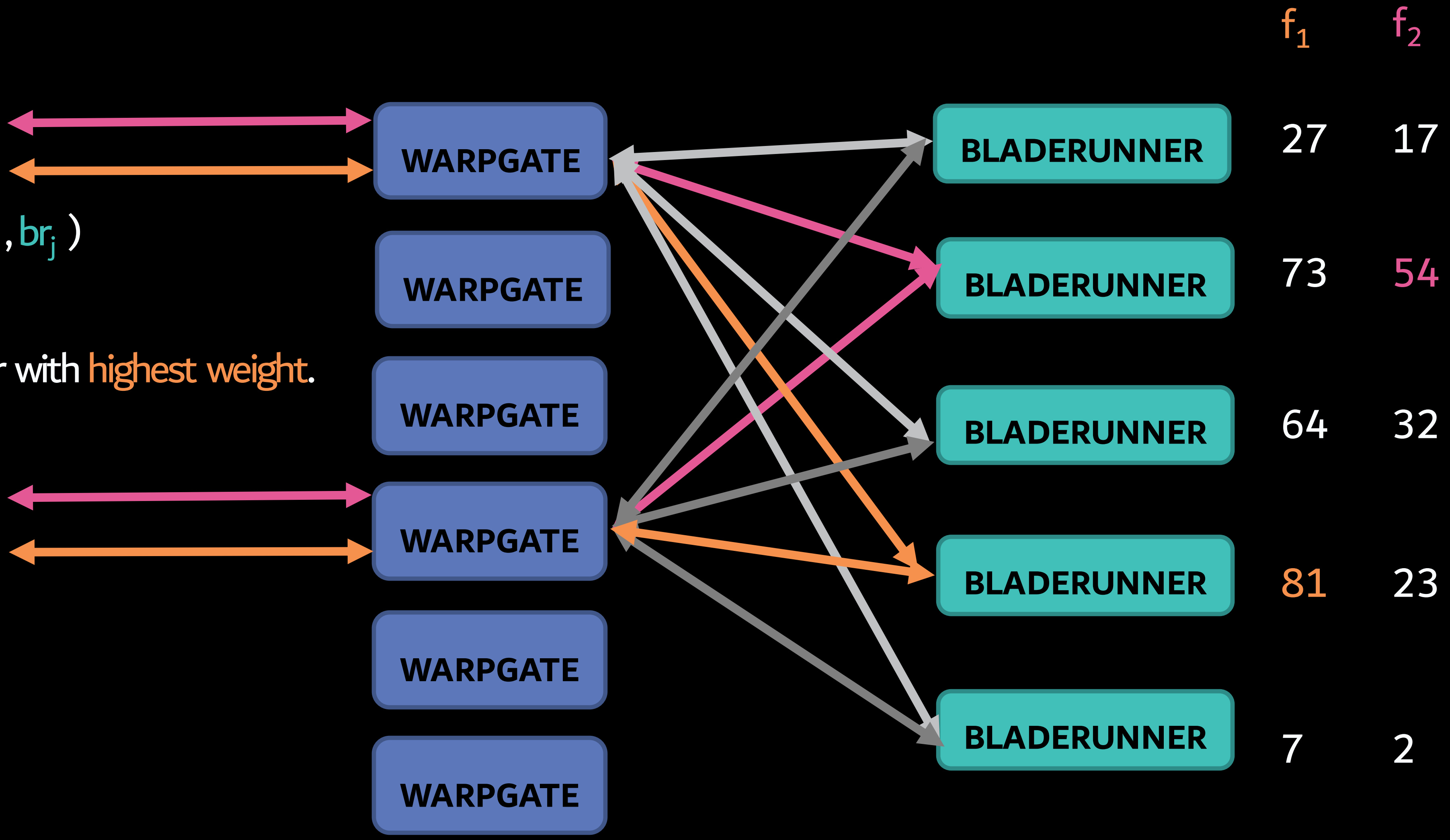
On any WarpGate:

Say we are routing feed f_i

For br_j in Bladerunner:

$\quad weight\_of\_br_j = HASH(f_i , br_j)$

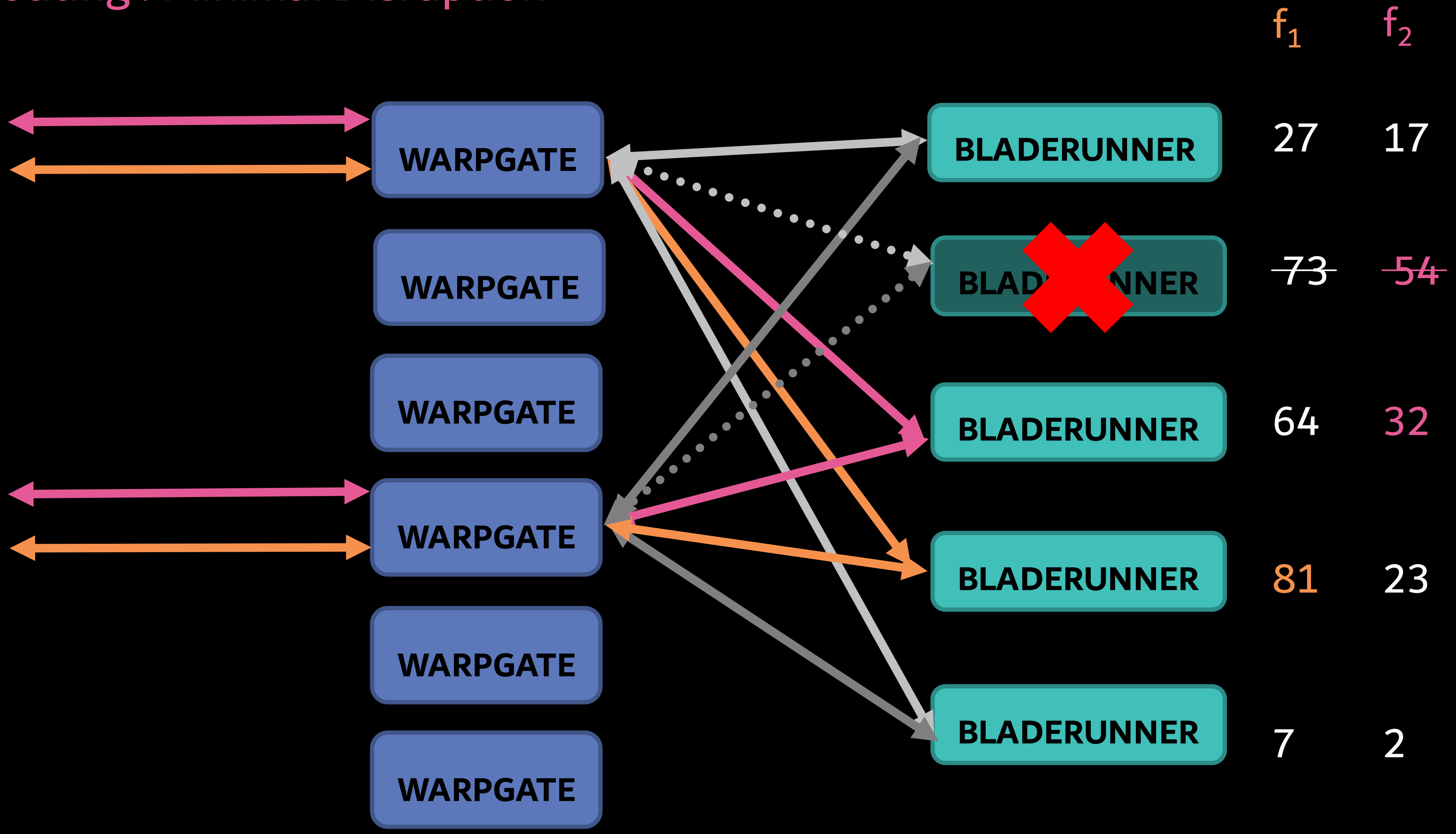
Finally, route to Bladerunner with highest weight.



	f_1	f_2
BLADERUNNER	27	17
BLADERUNNER	73	54
BLADERUNNER	64	32
BLADERUNNER	81	23
BLADERUNNER	7	2

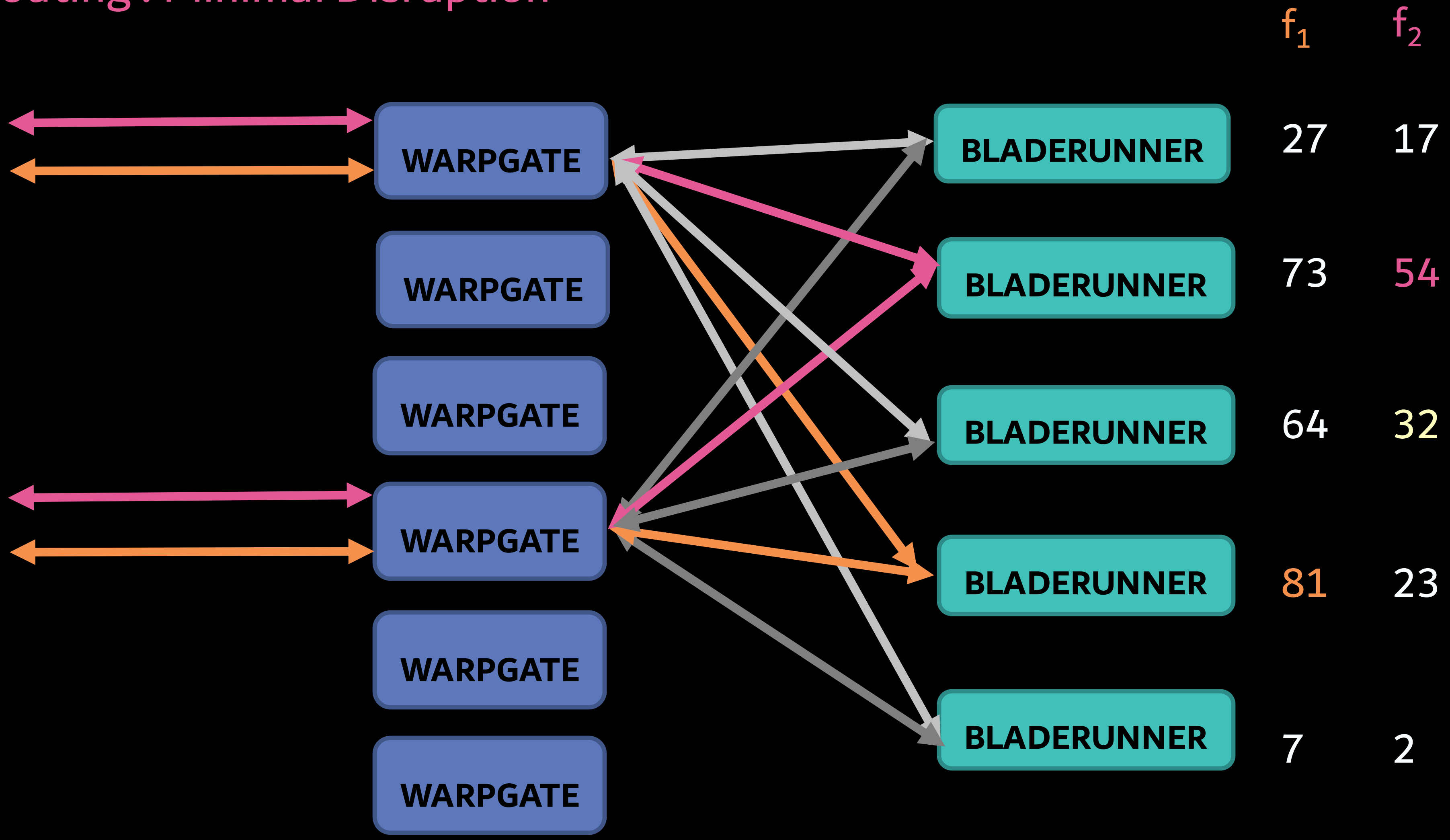
Full-Mesh Architecture

Sticky Routing : Minimal Disruption



Full-Mesh Architecture

Sticky Routing : Minimal Disruption



WARPGATE

WARPGATE

WARPGATE

WARPGATE

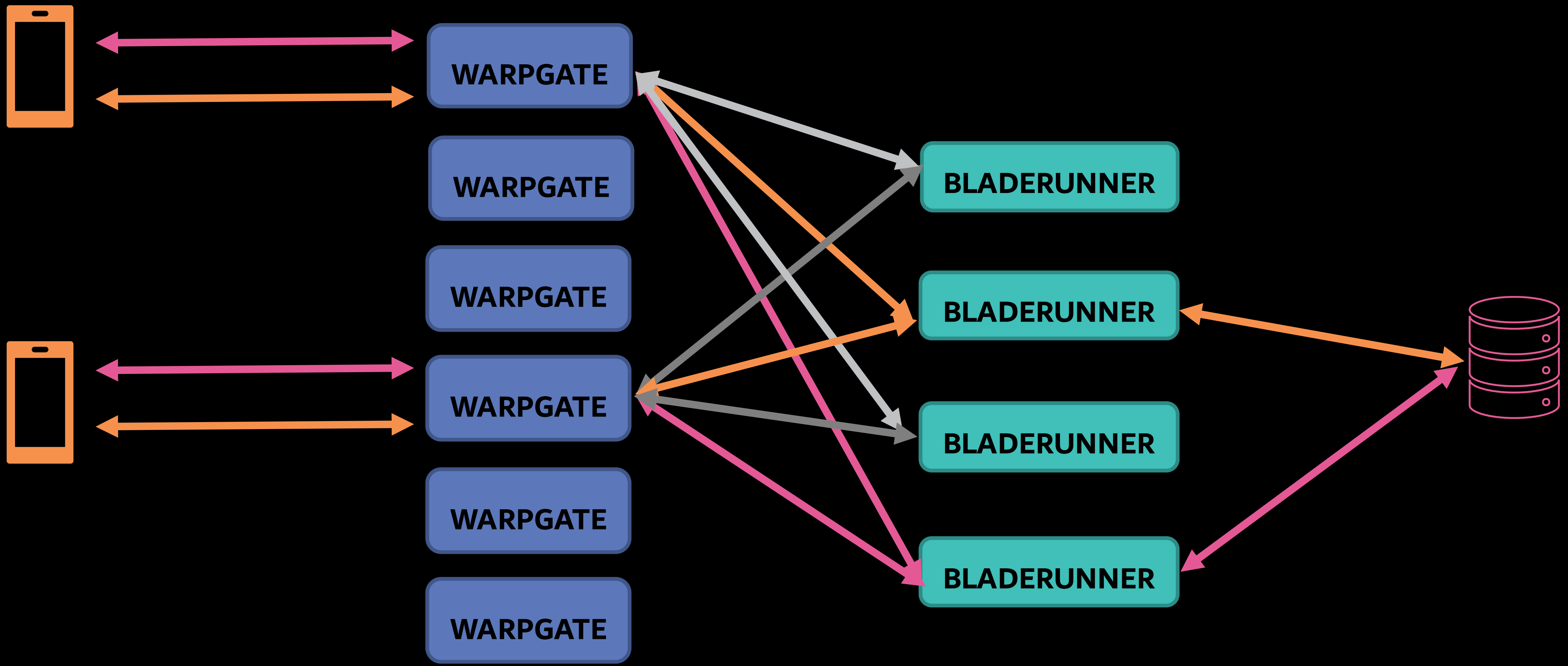
WARPGATE

WARPGATE

	f_1	f_2
BLADERUNNER	27	17
BLADERUNNER	73	54
BLADERUNNER	64	32
BLADERUNNER	81	23
BLADERUNNER	7	2

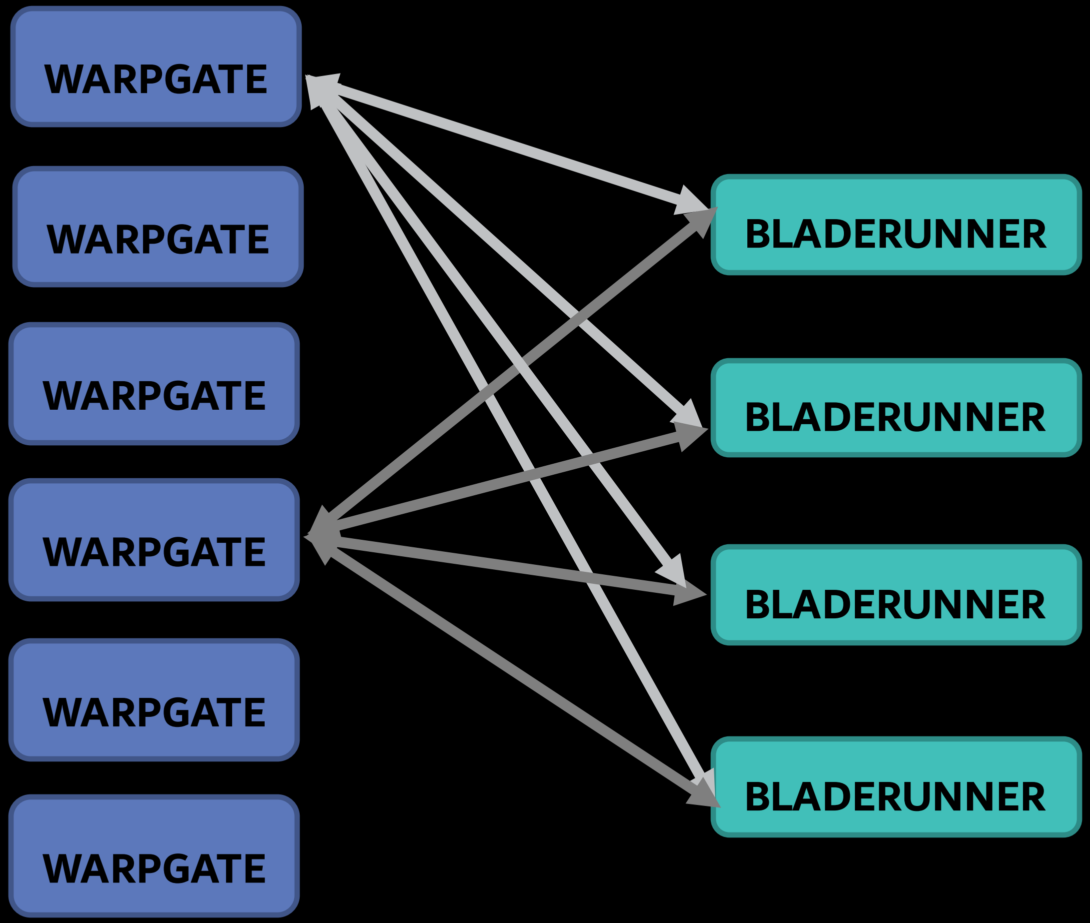
Full-Mesh Architecture

Sticky Routing



Full-Mesh Architecture

New Routing Capabilities

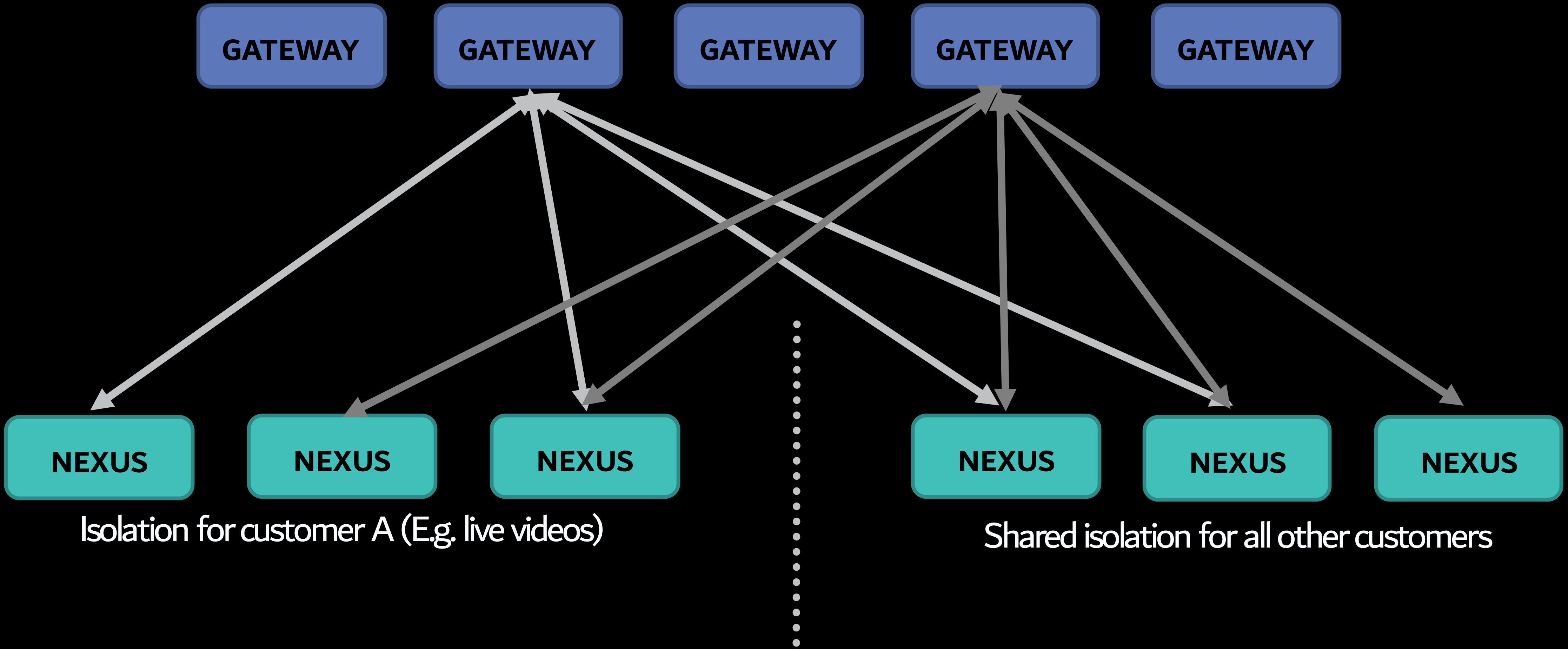


1. Sticky Routing

2. Virtual Isolation

Looking back

Customer traffic Isolation in the off-box architecture



Isolation for customer A (E.g. live videos)

Shared isolation for all other customers

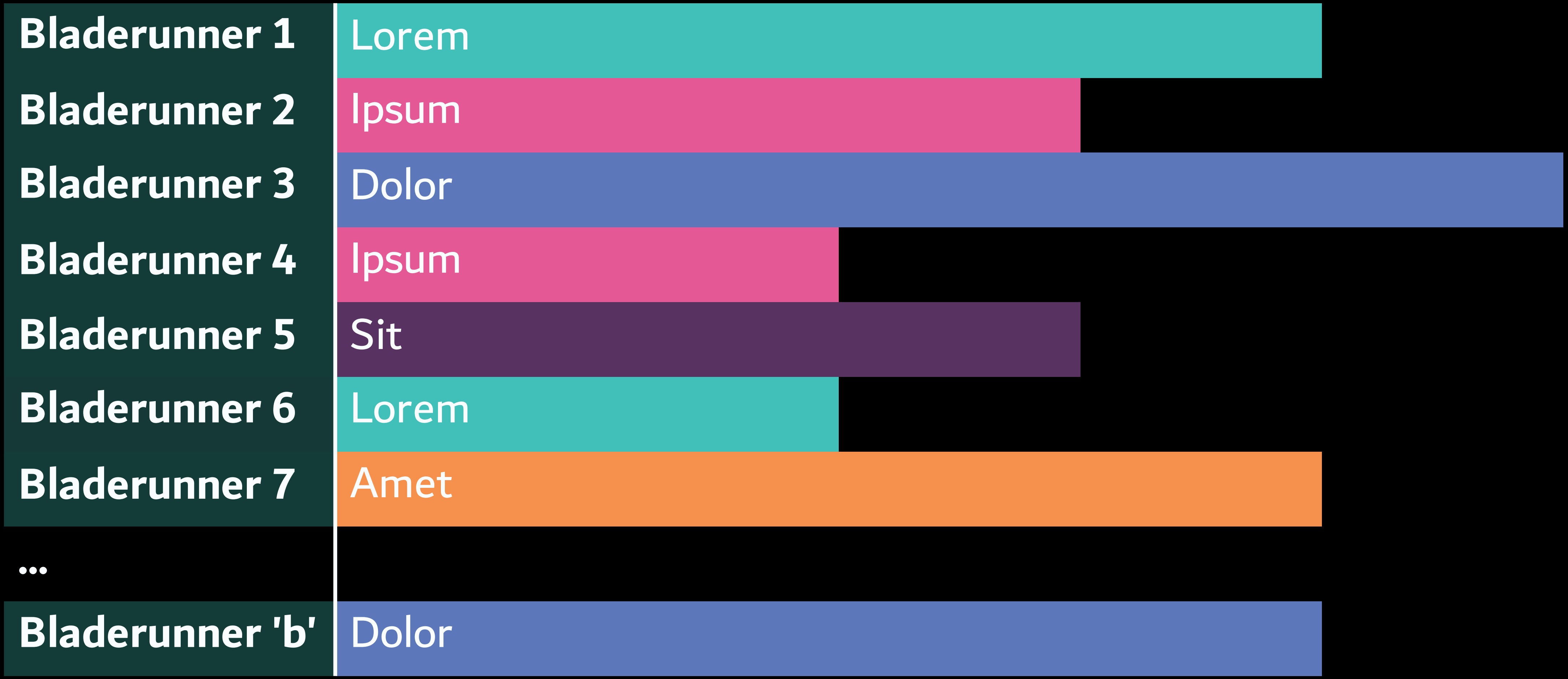
Full-Mesh Architecture

Virtual Isolation

Bladerunner 1	Lorem	Ipsum	Dolor	Sit	Amet
Bladerunner 2	Lorem	Ipsum	Dolor	Sit	Amet
Bladerunner 3	Lorem	Ipsum	Dolor	Sit	Amet
Bladerunner 4	Lorem	Ipsum	Dolor	Sit	Amet
Bladerunner 5	Lorem	Ipsum	Dolor	Sit	Amet
Bladerunner 6	Lorem	Ipsum	Dolor	Sit	Amet
Bladerunner 7	Lorem	Ipsum	Dolor	Sit	Amet
...					
Bladerunner 'b'	Lorem	Ipsum	Dolor	Sit	Amet

Full-Mesh Architecture

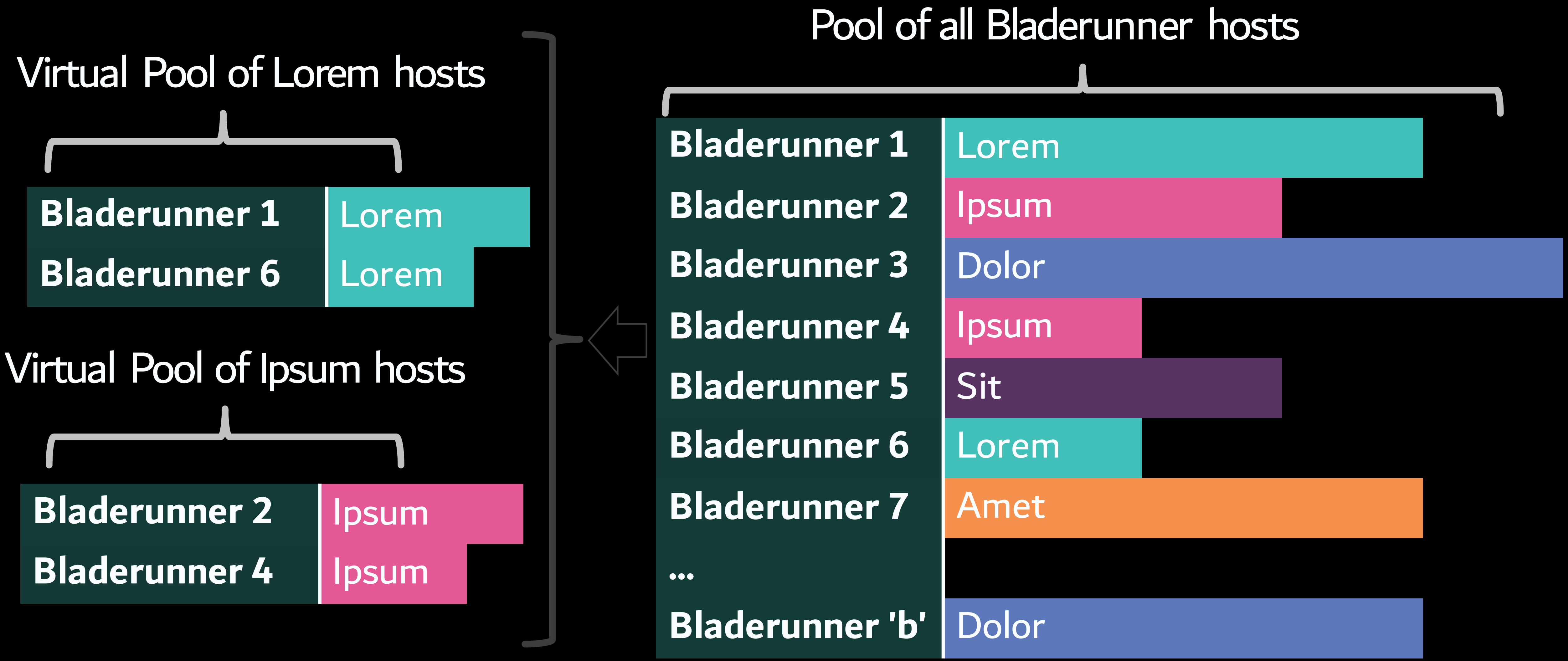
Virtual Isolation



Bladerunner 1	Lorem
Bladerunner 2	Ipsum
Bladerunner 3	Dolor
Bladerunner 4	Ipsum
Bladerunner 5	Sit
Bladerunner 6	Lorem
Bladerunner 7	Amet
...	
Bladerunner 'b'	Dolor

Full-Mesh Architecture

Virtual Pools for Isolation



Pool of all Bladerunner hosts

Bladerunner 1	Lorem
Bladerunner 2	Ipsum
Bladerunner 3	Dolor
Bladerunner 4	Ipsum
Bladerunner 5	Sit
Bladerunner 6	Lorem
Bladerunner 7	Amet
...	
Bladerunner 'b'	Dolor

Virtual Pool of Lorem hosts

Bladerunner 1	Lorem
Bladerunner 6	Lorem

Virtual Pool of Ipsum hosts

Bladerunner 2	Ipsum
Bladerunner 4	Ipsum

Full-Mesh Architecture

Virtual Pools for Isolation

Noisy Neighbour

Resource Attribution

Maintenance Costs

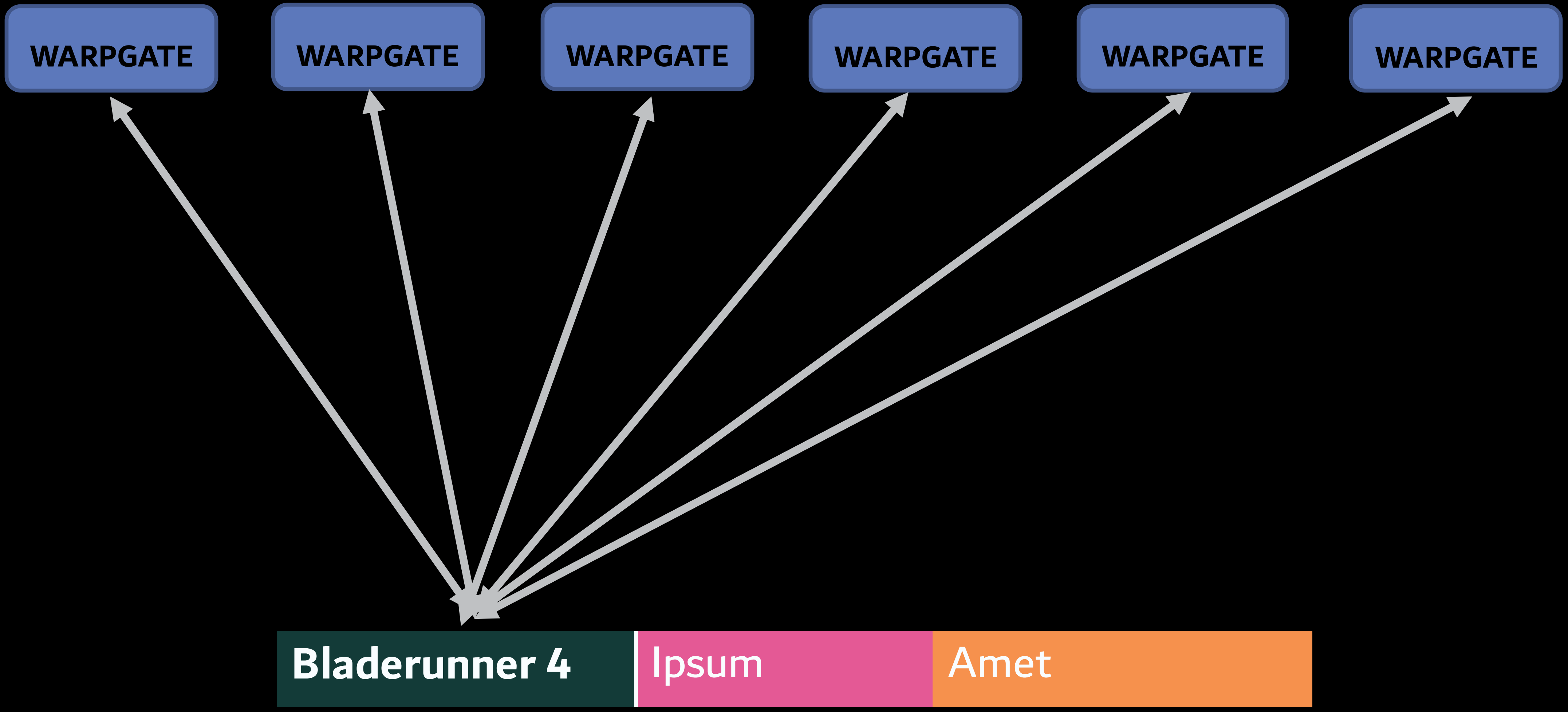
Full-Mesh Architecture

Virtual Pools for Isolation

1. WarpGate Visibility

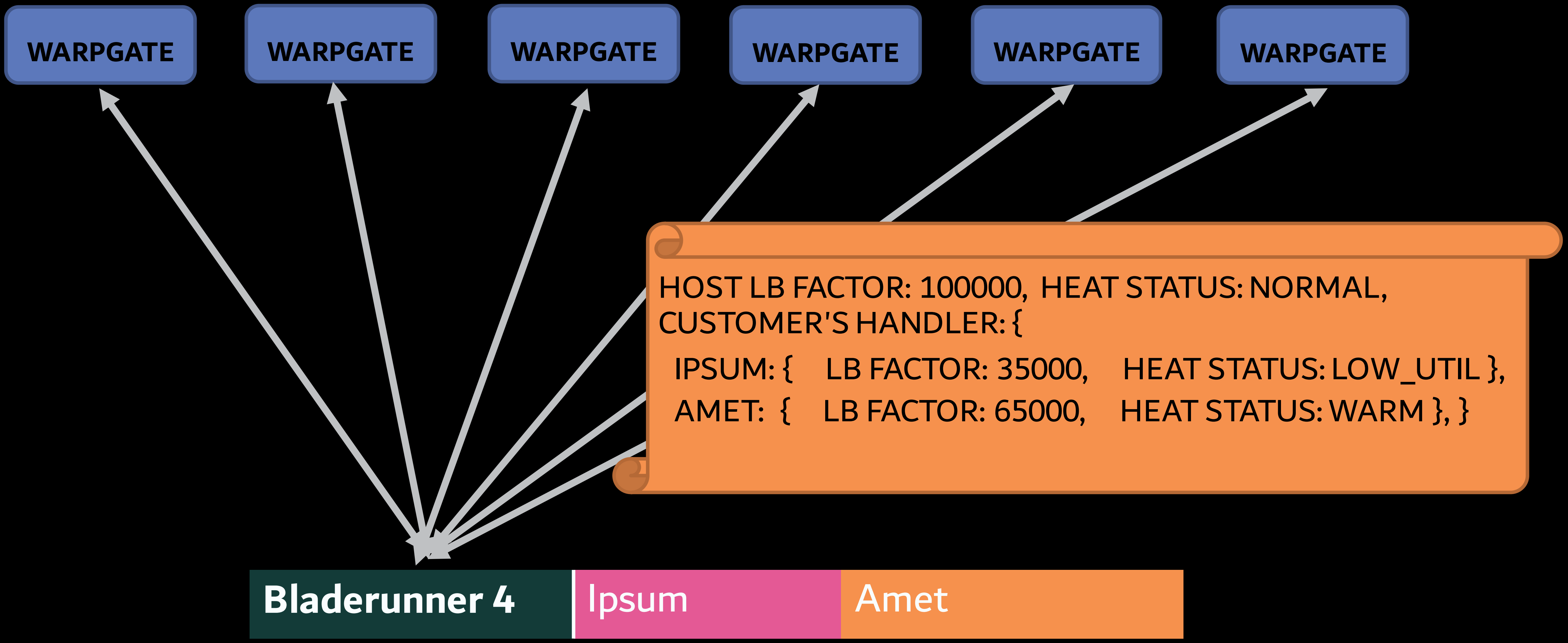
Full-Mesh Architecture

Virtual Pools for Isolation: WarpGate Visibility



Full-Mesh Architecture

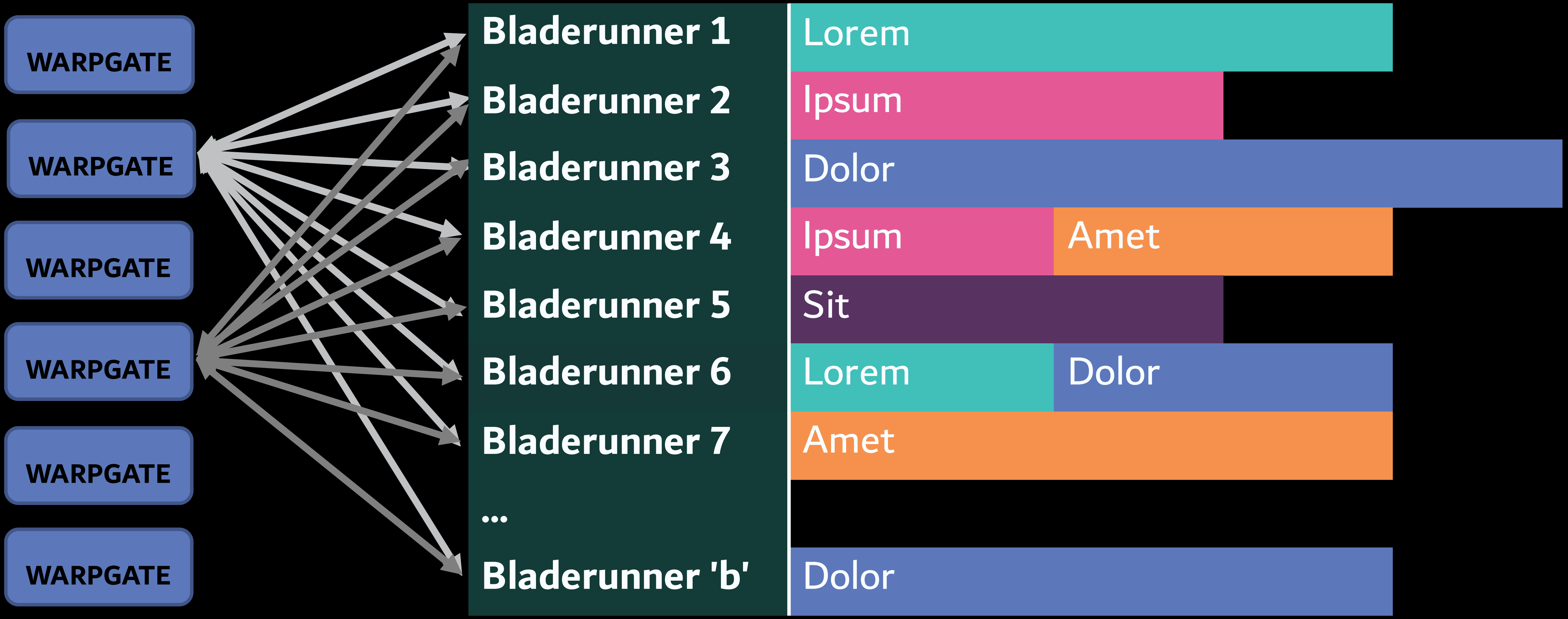
Virtual Pools for Isolation: WarpGate Visibility



HOST LB FACTOR: 100000, HEAT STATUS: NORMAL,
CUSTOMER'S HANDLER: {
IPSUM: { LB FACTOR: 35000, HEAT STATUS: LOW_UTIL },
AMET: { LB FACTOR: 65000, HEAT STATUS: WARM }, }

Full-Mesh Architecture

Virtual Pools for Isolation: WarpGate Visibility



Bladerunner 1	Lorem	
Bladerunner 2	Ipsum	
Bladerunner 3	Dolor	
Bladerunner 4	Ipsum	Amet
Bladerunner 5	Sit	
Bladerunner 6	Lorem	Dolor
Bladerunner 7	Amet	
...		
Bladerunner 'b'	Dolor	

Full-Mesh Architecture

Virtual Pools for Isolation

1. WarpGate Visibility

2. Virtual Pool placement

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:

Say we are to scale up vpool_i

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

```
On each WarpGate:
Say we are to scale up $vpool_i$
For $br_j$ in Bladerunner:
    $weight\_of\_br_j = HASH( vpool_i , br_j )$
```

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

```
On each WarpGate:
Say we are to scale up $vpool_i$
For $br_j$ in Bladerunner:
    $weight\_of\_br_j$ = HASH( $vpool_i$ , $br_j$ )
```

Ignore weights of bladerunners that we don't want to grow into.

Finally, upsize into the bladerunner host with the highest weight.

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:

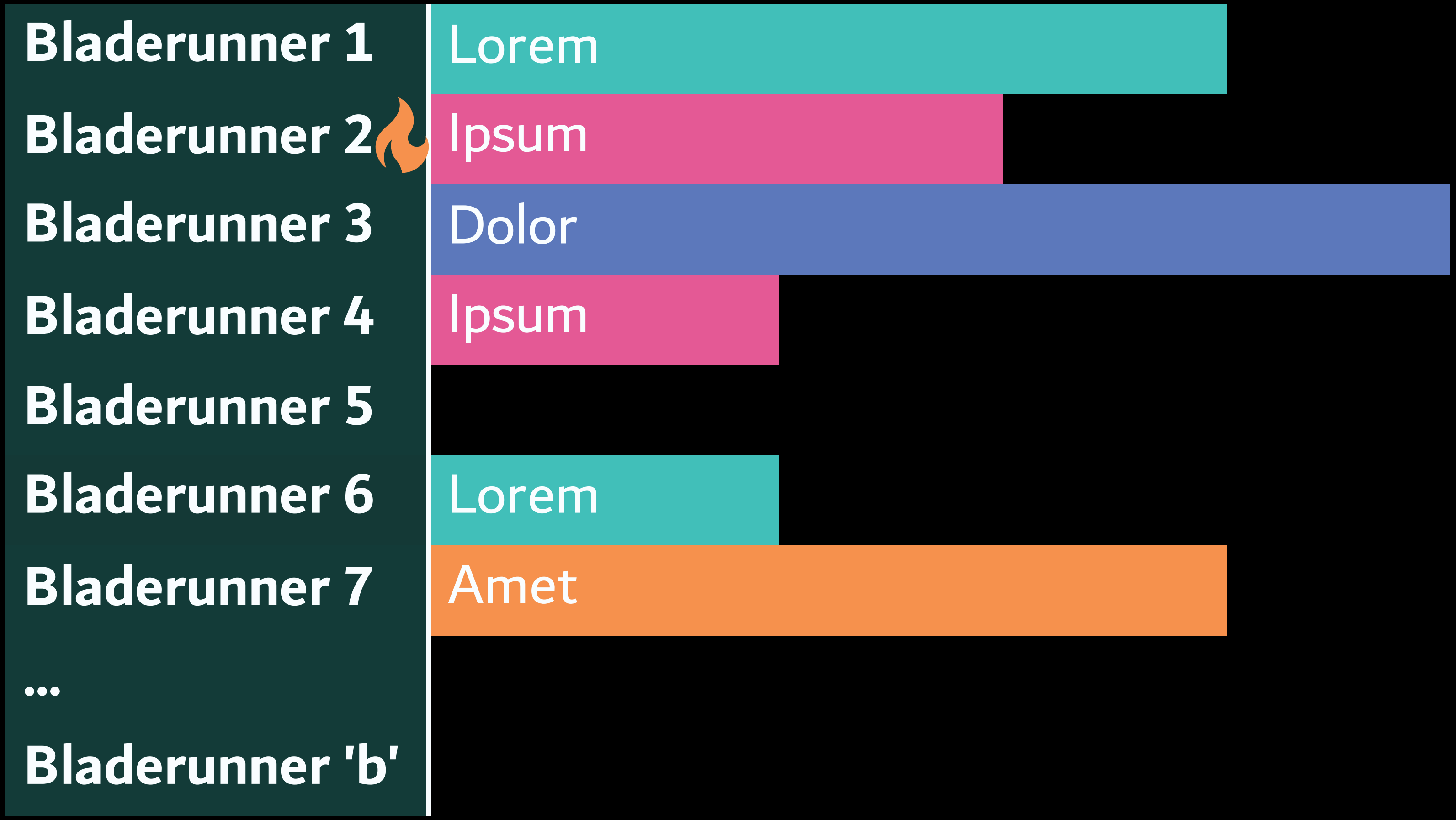
Say we are to scale up vpool $_i$

For br$_j$ in Bladerunner:

  weight_of_br$_j$ = HASH(vpool$_i$, br$_j$)

Ignore weights of bladerunners that we don't want to grow into.

Finally, upsize into the bladerunner host with the highest weight.



Bladerunner 1	Lorem
Bladerunner 2 🔥	Ipsum
Bladerunner 3	Dolor
Bladerunner 4	Ipsum
Bladerunner 5	
Bladerunner 6	Lorem
Bladerunner 7	Amet
...	
Bladerunner 'b'	

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:

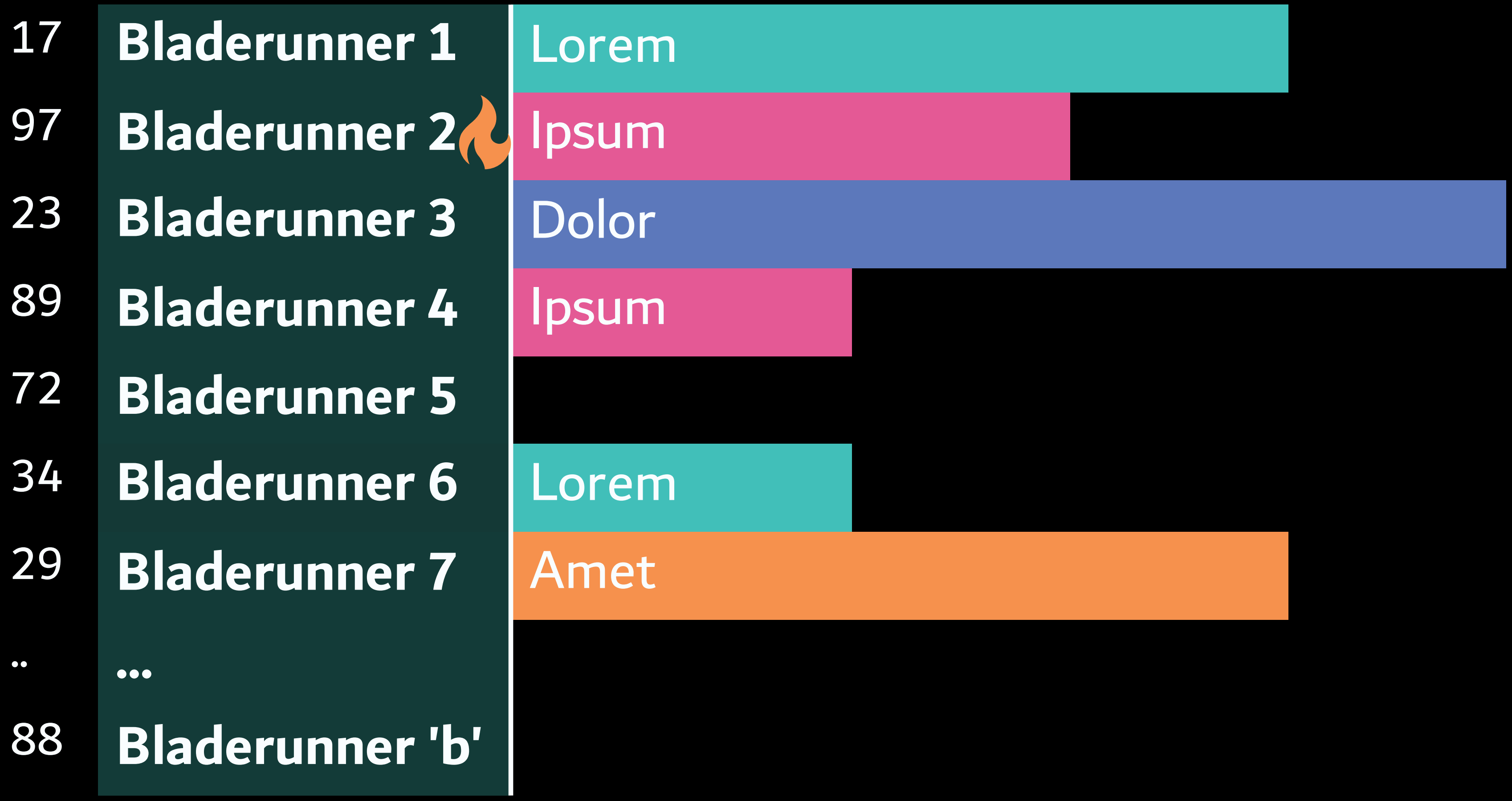
Say we are to scale up $vpool_i$

For br_j in Bladerunner:

$weight\_of\_br_j = HASH(vpool_i, br_j)$

Ignore weights of bladerunners that we don't want to grow into.

Finally, upsize into the bladerunner host with the highest weight.



17	Bladerunner 1	Lorem
97	Bladerunner 2 🔥	Ipsum
23	Bladerunner 3	Dolor
89	Bladerunner 4	Ipsum
72	Bladerunner 5	
34	Bladerunner 6	Lorem
29	Bladerunner 7	Amet
..	...	
88	Bladerunner 'b'	

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:

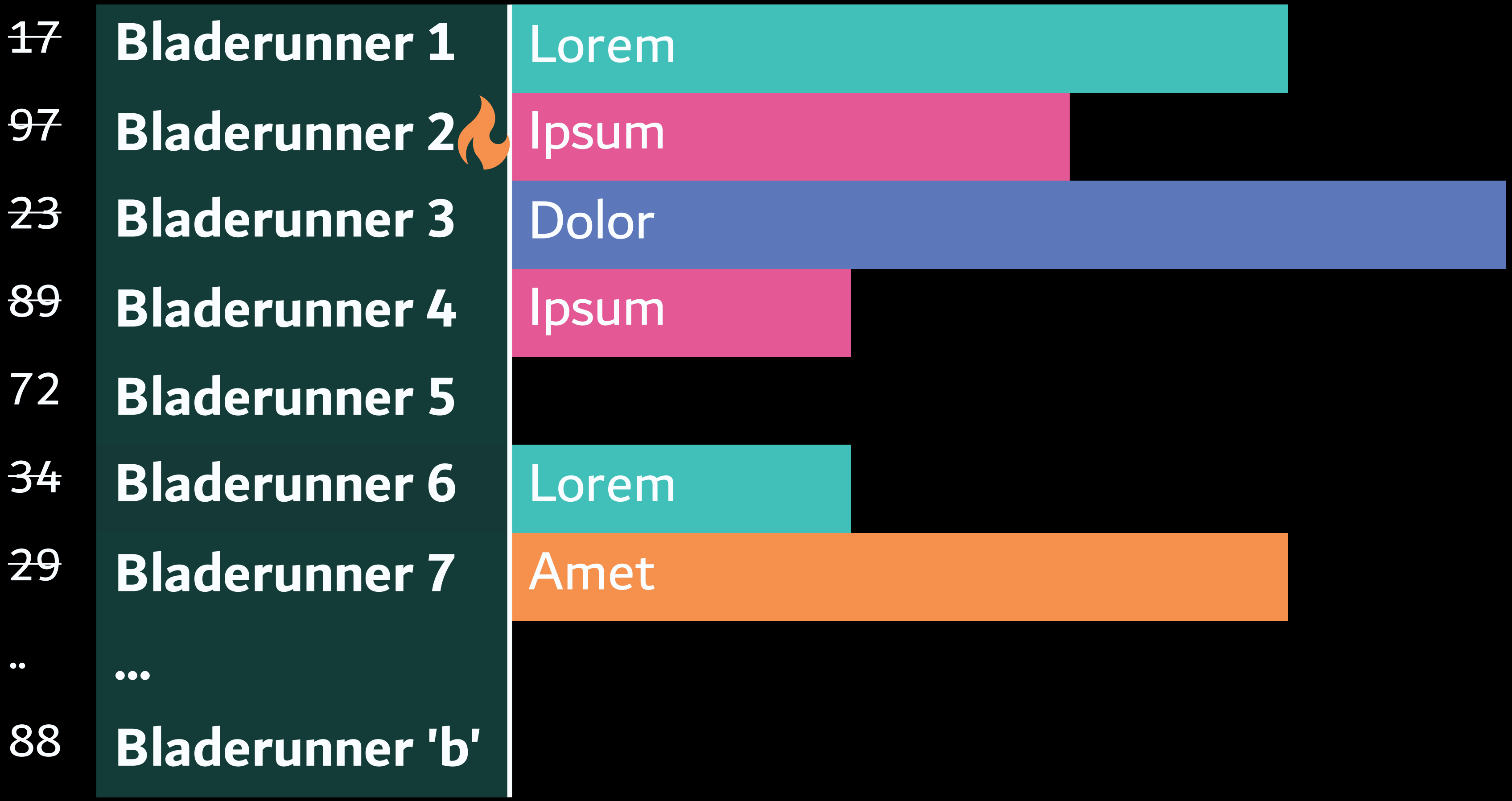
Say we are to scale up vpool $_i$

For br$_j$ in Bladerunner:

  weight_of_br$_j$ = HASH(vpool$_i$, br$_j$)

Ignore weights of bladerunners that we don't want to grow into.

Finally, upsize into the bladerunner host with the highest weight.



~~17~~	Bladerunner 1	Lorem
~~97~~	Bladerunner 2 🔥	Ipsum
~~23~~	Bladerunner 3	Dolor
~~89~~	Bladerunner 4	Ipsum
72	Bladerunner 5	
~~34~~	Bladerunner 6	Lorem
~~29~~	Bladerunner 7	Amet
..	...	
88	Bladerunner 'b'	

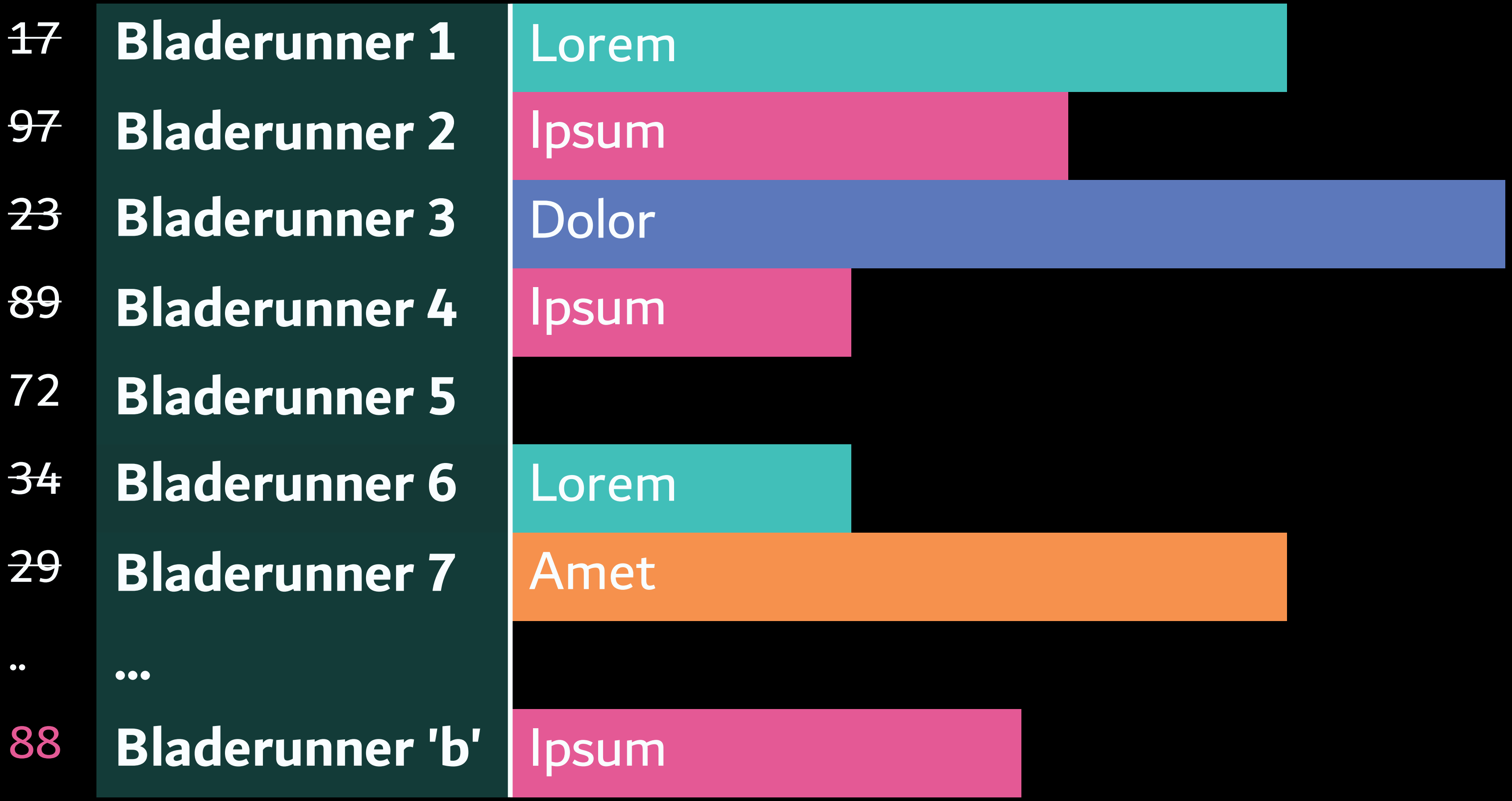
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale up vpool$_i$
For br$_j$ in Bladerunner:
    $\text{weight\_of\_br}_j = \text{HASH}(\text{vpool}_i, \text{br}_j)$

Ignore weights of bladerunners that we don't want to grow into.

Finally, upsize into the bladerunner host with the highest weight.



~~17~~	Bladerunner 1	Lorem
~~97~~	Bladerunner 2	Ipsum
~~23~~	Bladerunner 3	Dolor
~~89~~	Bladerunner 4	Ipsum
72	Bladerunner 5	
~~34~~	Bladerunner 6	Lorem
~~29~~	Bladerunner 7	Amet
..	...	
88	Bladerunner 'b'	Ipsum

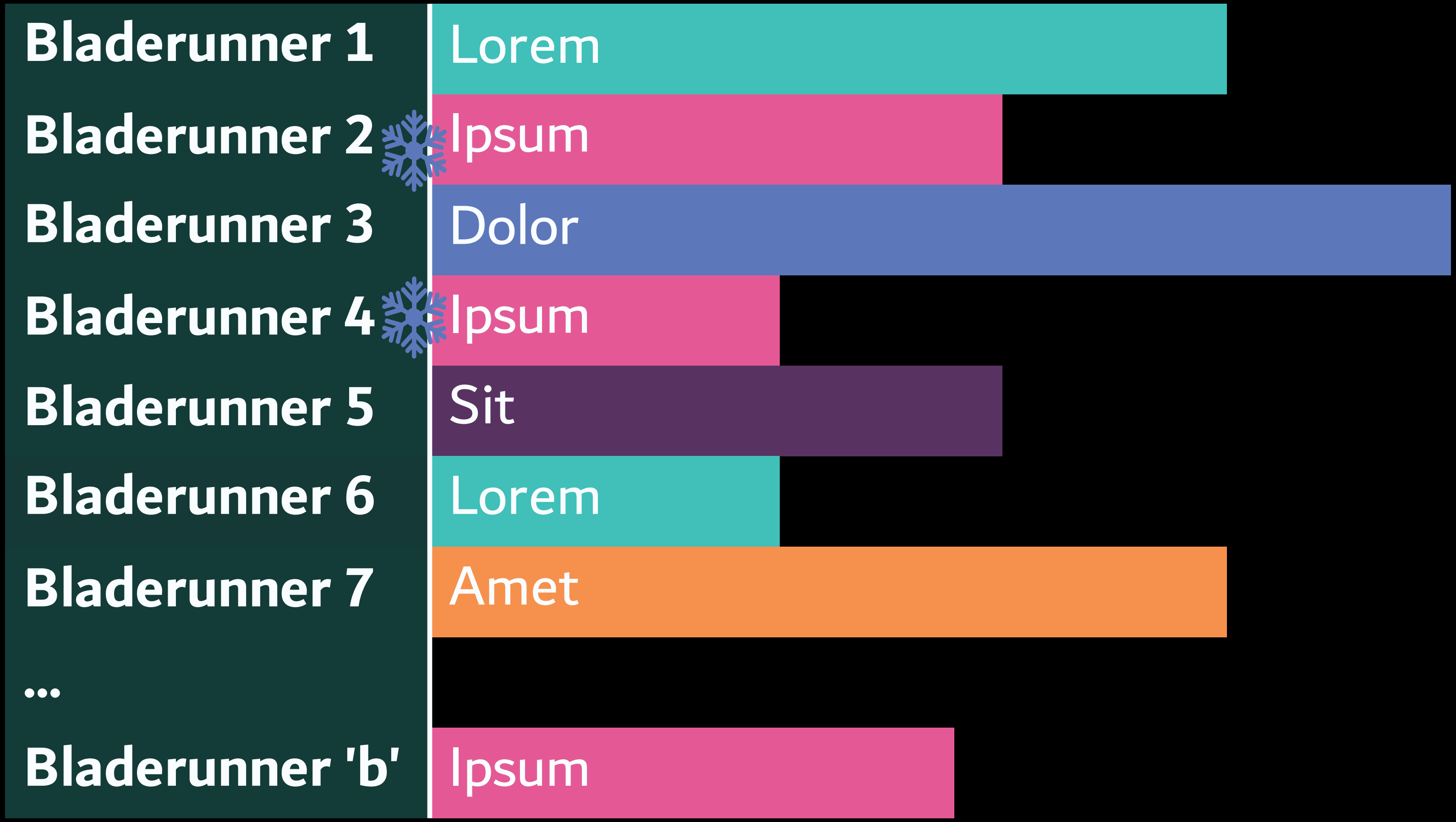
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:
Say we are to scale down $vpool_i$
For br_j in Bladerunner:
    $weight\_of\_br_j$ = HASH($vpool_i$, br_j)

Ignore weights of bladerunners that don't have desired $vpool_i$

Finally, downsize the bladerunner host with the LOWEST weight in the virtual pool



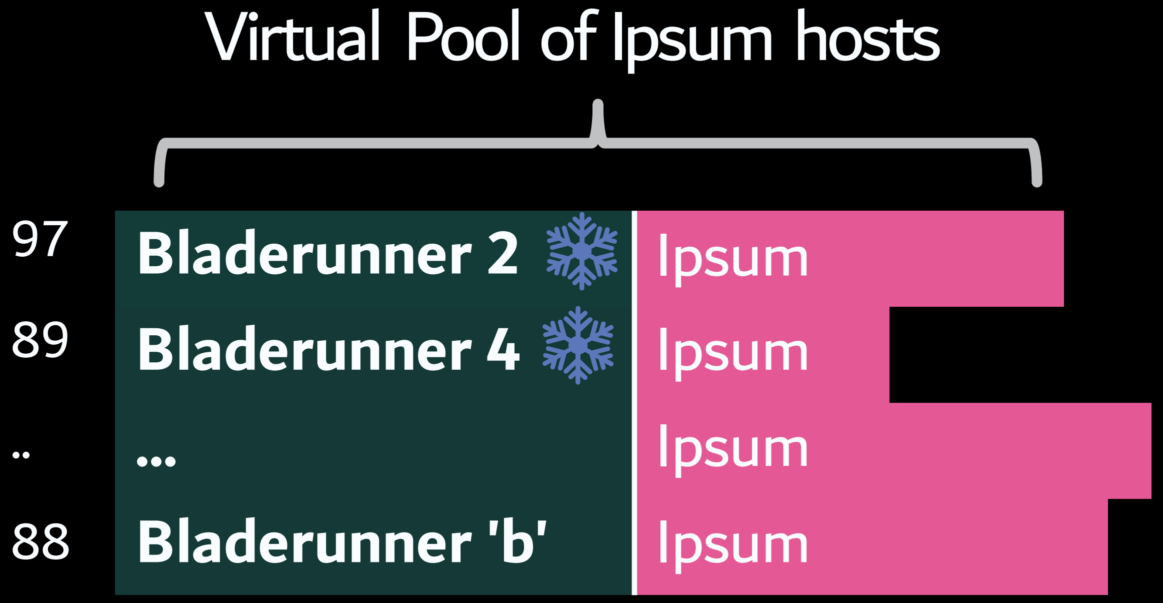
Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

```
On each WarpGate:
Say we are to scale down $vpool_i$
For $br_j$ in Bladerunner:
    $weight\_of\_br_j$ = HASH( $vpool_i$ , $br_j$ )
```

Ignore weights of bladerunners that don't have desired $vpool_i$

Finally, downsize the bladerunner host with the LOWEST weight in the virtual pool



Virtual Pool of Ipsum hosts

97	Bladerunner 2 ❄	Ipsum
89	Bladerunner 4 ❄	Ipsum
..	...	Ipsum
88	Bladerunner 'b'	Ipsum

Full-Mesh Architecture

Virtual Pools for Isolation: Virtual Pool Placement

On each WarpGate:

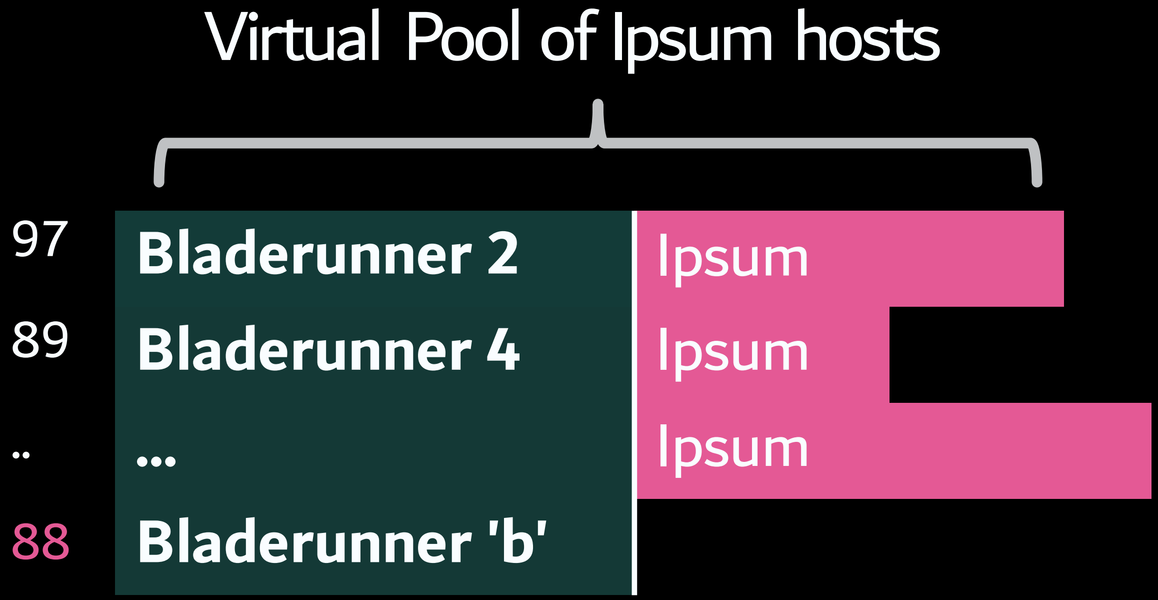
Say we are to scale down vpool $_i$

For br_j in Bladerunner:

    $weight\_of\_br_j$ = HASH($vpool_i$, br_j)

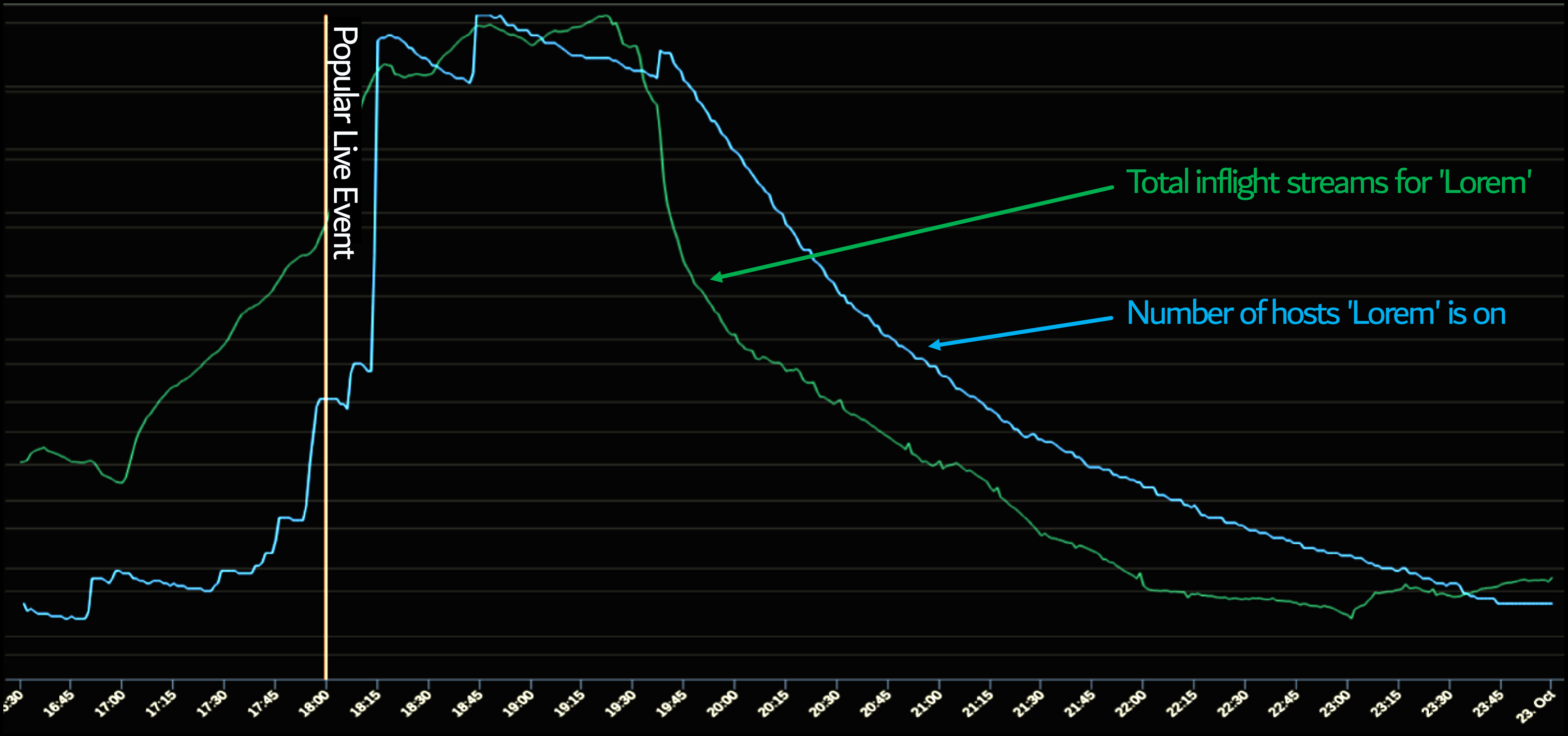
Ignore weights of bladerunners that don't have desired $vpool_i$

Finally, downsize the bladerunner host with the LOWEST weight in the virtual pool



Full-Mesh Architecture

Virtual Pools for Isolation: Case Study of a Recent Event



Full-Mesh Architecture

Virtual Pools for Isolation

Noisy Neighbour

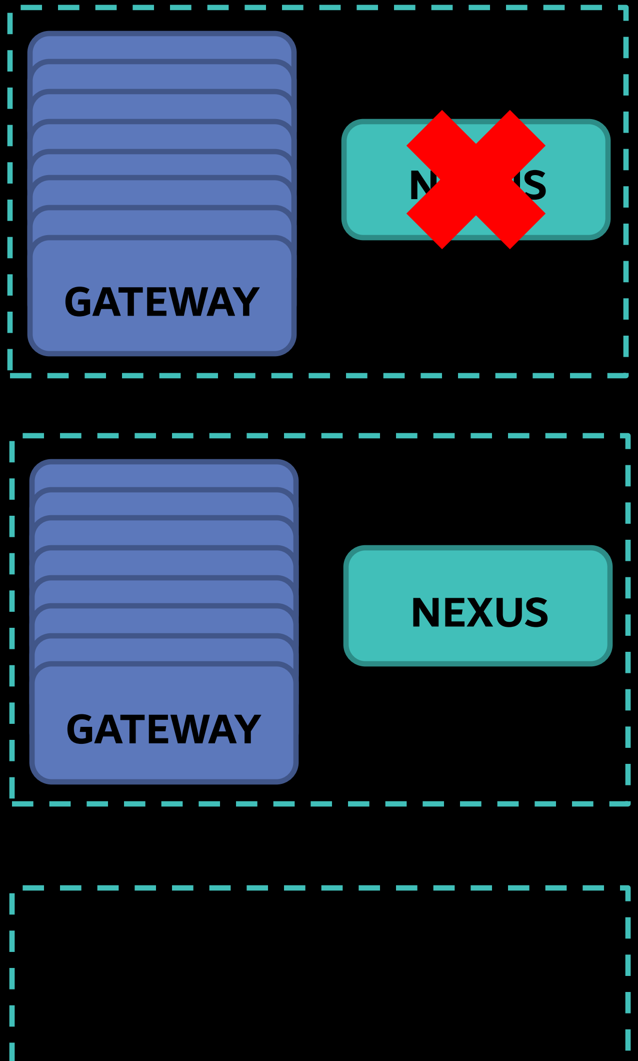
Resource Attribution

Maintenance Costs

The Evolution of Traffic Routing in a Streaming World

In Conclusion

1. Moving away from Collocation

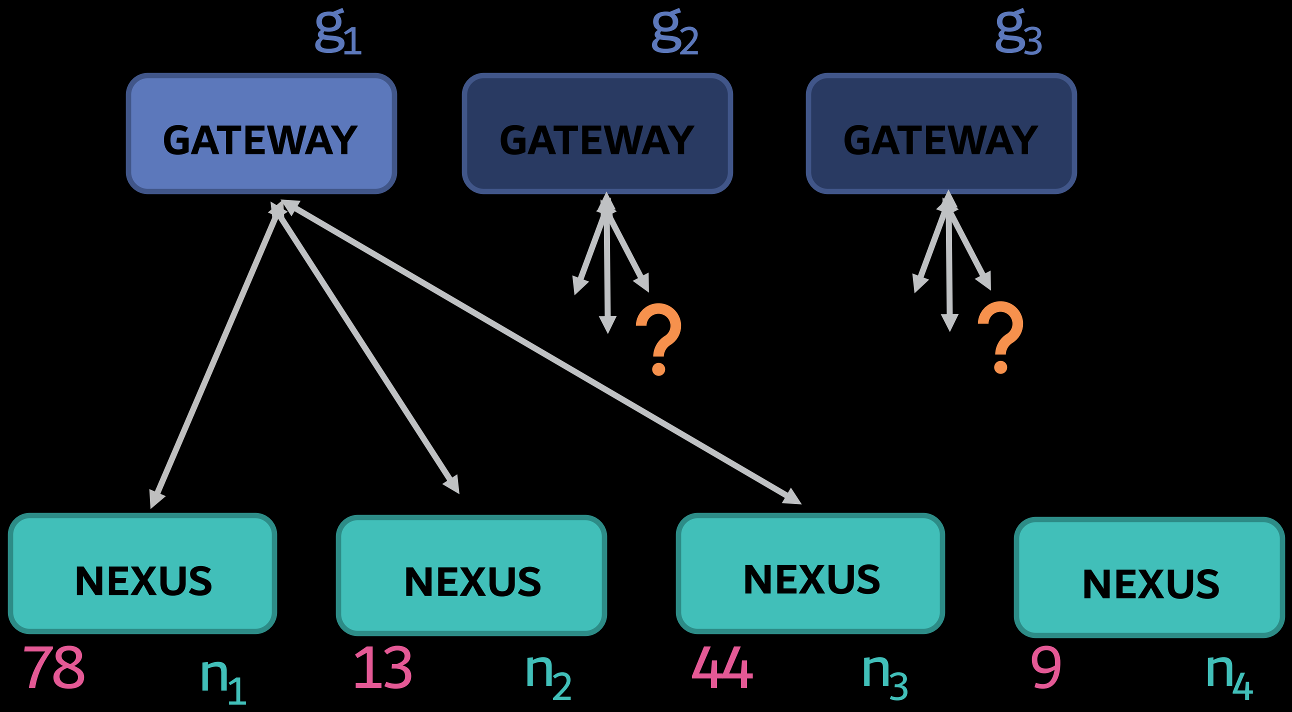


The Evolution of Traffic Routing in a Streaming World

In Conclusion

1. Moving away from Collocation

2. Rendezvous Hashing



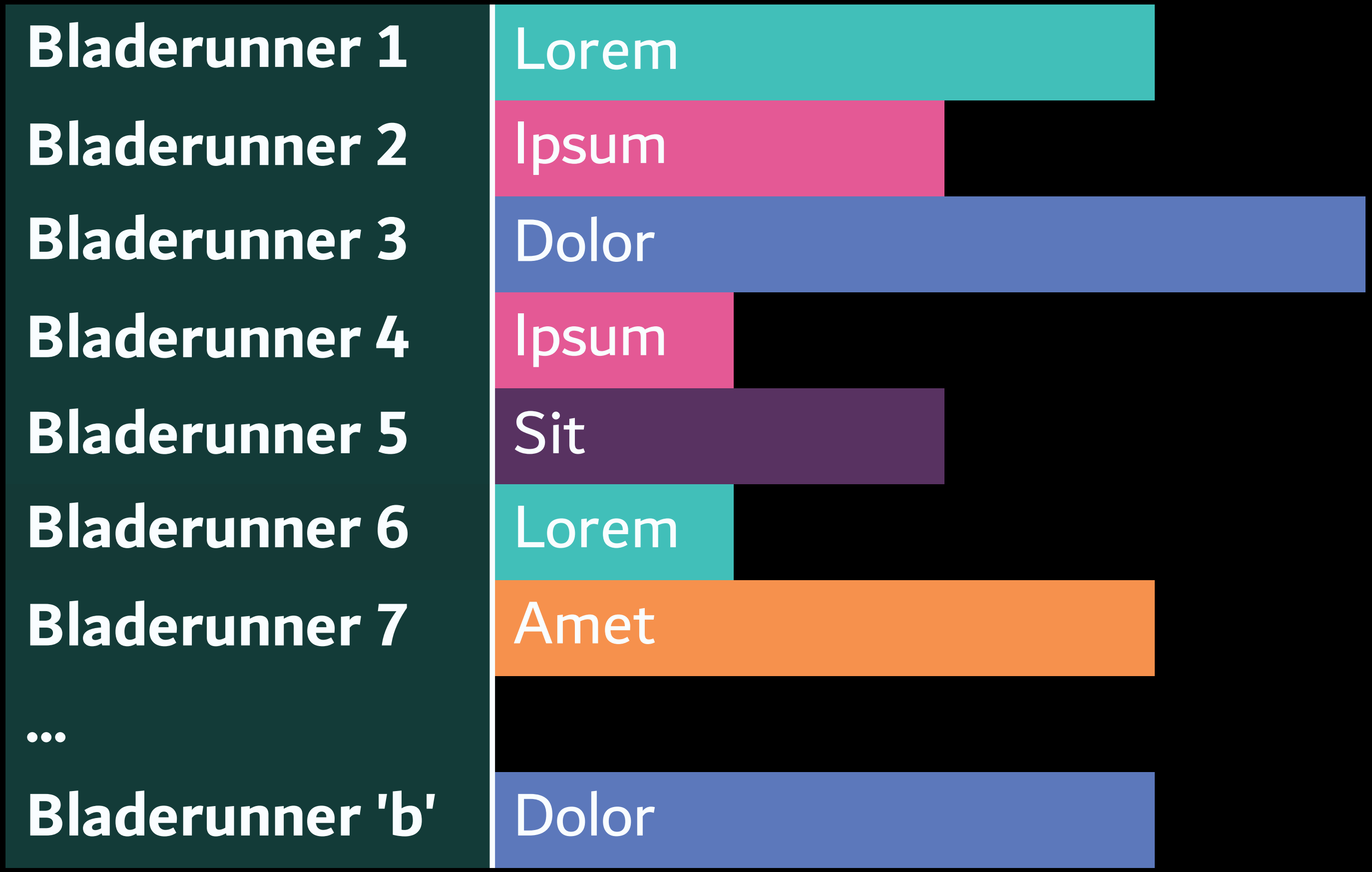
The Evolution of Traffic Routing in a Streaming World

In Conclusion

1. Moving away from Collocation

2. Rendezvous Hashing

3. Full Mesh Architecture



Bladerunner 1	Lorem
Bladerunner 2	Ipsum
Bladerunner 3	Dolor
Bladerunner 4	Ipsum
Bladerunner 5	Sit
Bladerunner 6	Lorem
Bladerunner 7	Amet
...	
Bladerunner 'b'	Dolor

Thank you