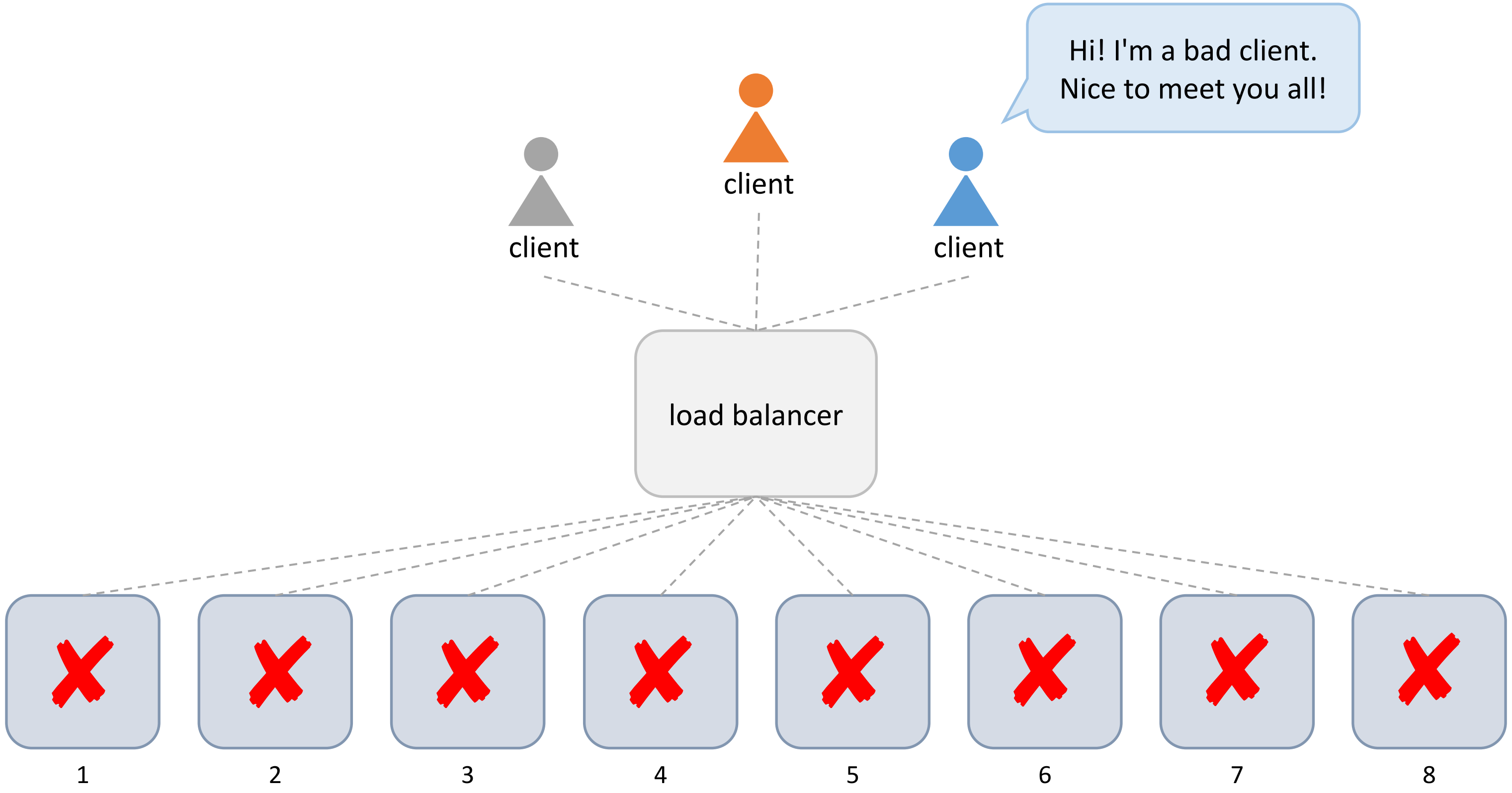
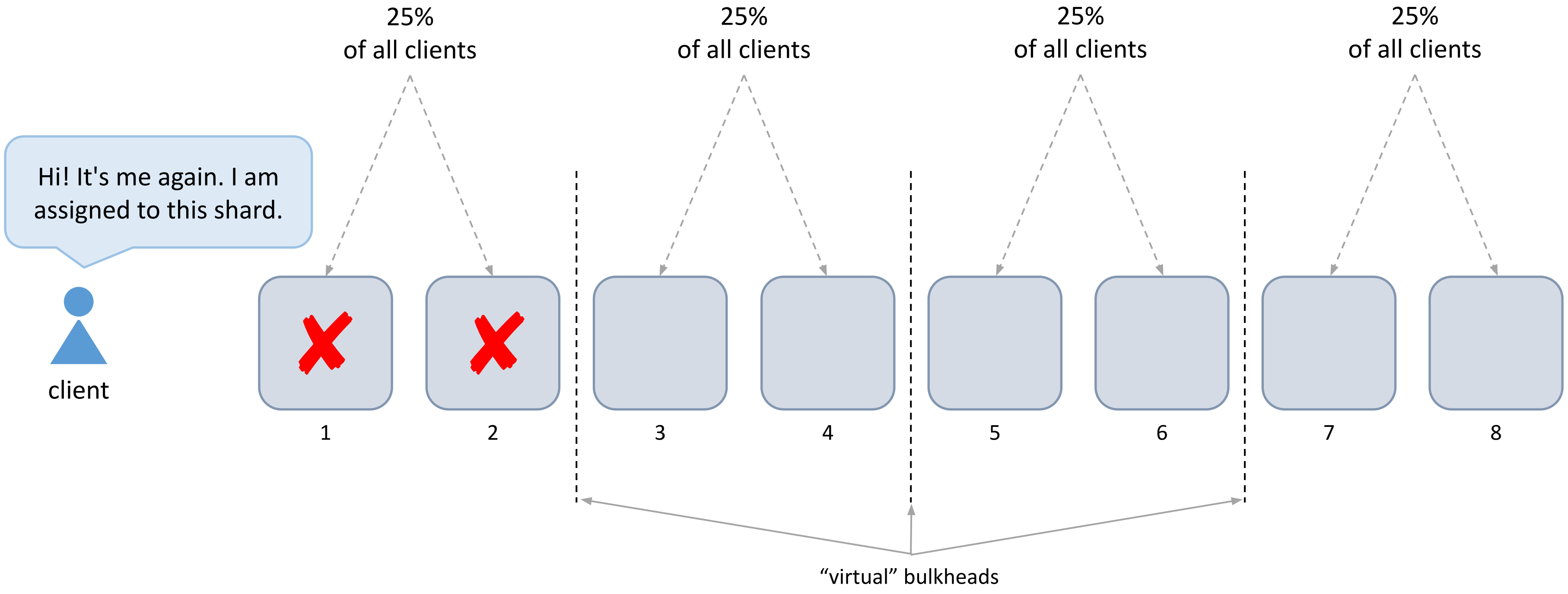
Shuffle sharding

What is a bad client? (from the server's point of view)

- creates a flood of requests (a way more than a typical client)
- sends very expensive requests (computationally intensive, scan large volumes of data, result in heavy responses)
- generates poisonous requests (expose high-severity server bugs)



Shuffle sharding

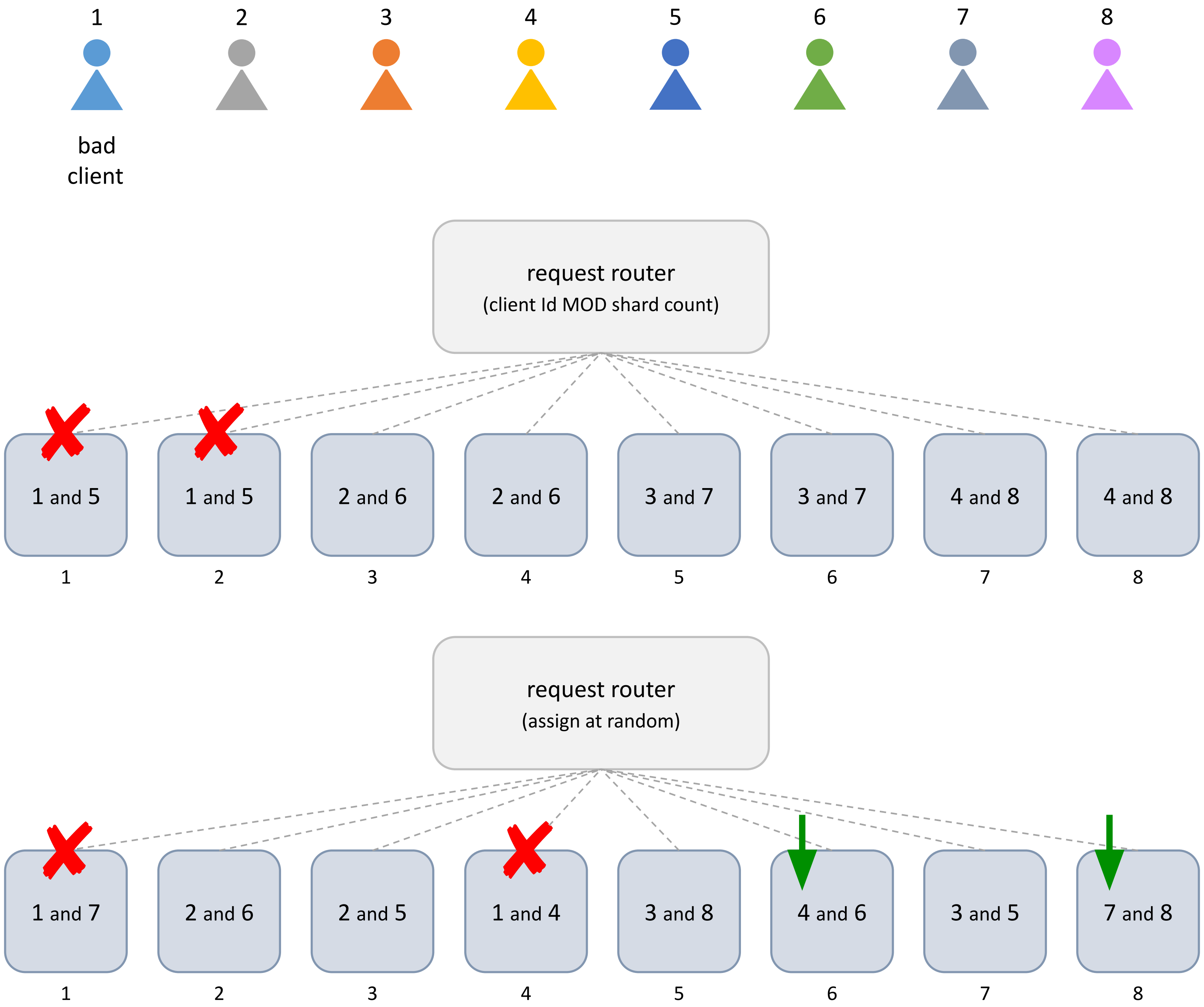


$$\text{blast radius} = \frac{\textit{number of clients}}{\textit{number of shards}} = 25\%$$

much better than 100% we had without sharding

but it's still a lot

Shuffle sharding



1 2 3 4 5 6 7 8

bad client

request router
(client Id MOD shard count)

two clients (1 and 5)
are impacted

1	2	3	4	5	6	7	8
1 and 5	1 and 5	2 and 6	2 and 6	3 and 7	3 and 7	4 and 8	4 and 8

request router
(assign at random)

only client 1 is impacted
client 1 neighbors (4 and 7)
are served by other servers

1	2	3	4	5	6	7	8
1 and 7	2 and 6	2 and 5	1 and 4	3 and 8	4 and 6	3 and 5	7 and 8

Shuffle sharding

- there will be complete overlaps (when two clients share the same set of servers)

servers in a cluster
8

servers in each shard
2

overlap	%	interpretation
0	53.6	There is a 53.6% chance you do not overlap with a bad client at all.
1	42.8	There is a 42.8% chance you share 1 server with a bad client.
2	3.6	There is a 3.6% chance you share 2 (all) servers with a bad client. This is a chance of having a complete overlap. Much better than 25% chance we get with regular sharding.

https://twitter.com/colmmacc/status/1034500109445165056

servers in a cluster
100

servers in each shard
5

overlap	%
0	77
1	21
2	1.8
3	0.06
4	0.0006
5	0.0000013

https://twitter.com/colmmacc/status/1034500800020537344

- clients need to know how to handle server failures (specifically, set short timeouts and retry failed requests)
- requires an intelligent routing component
- we can assign clients to shuffle shards in either a stateless (we do not look back at existing assignments) or stateful (we look at all existing shuffle shards) manner