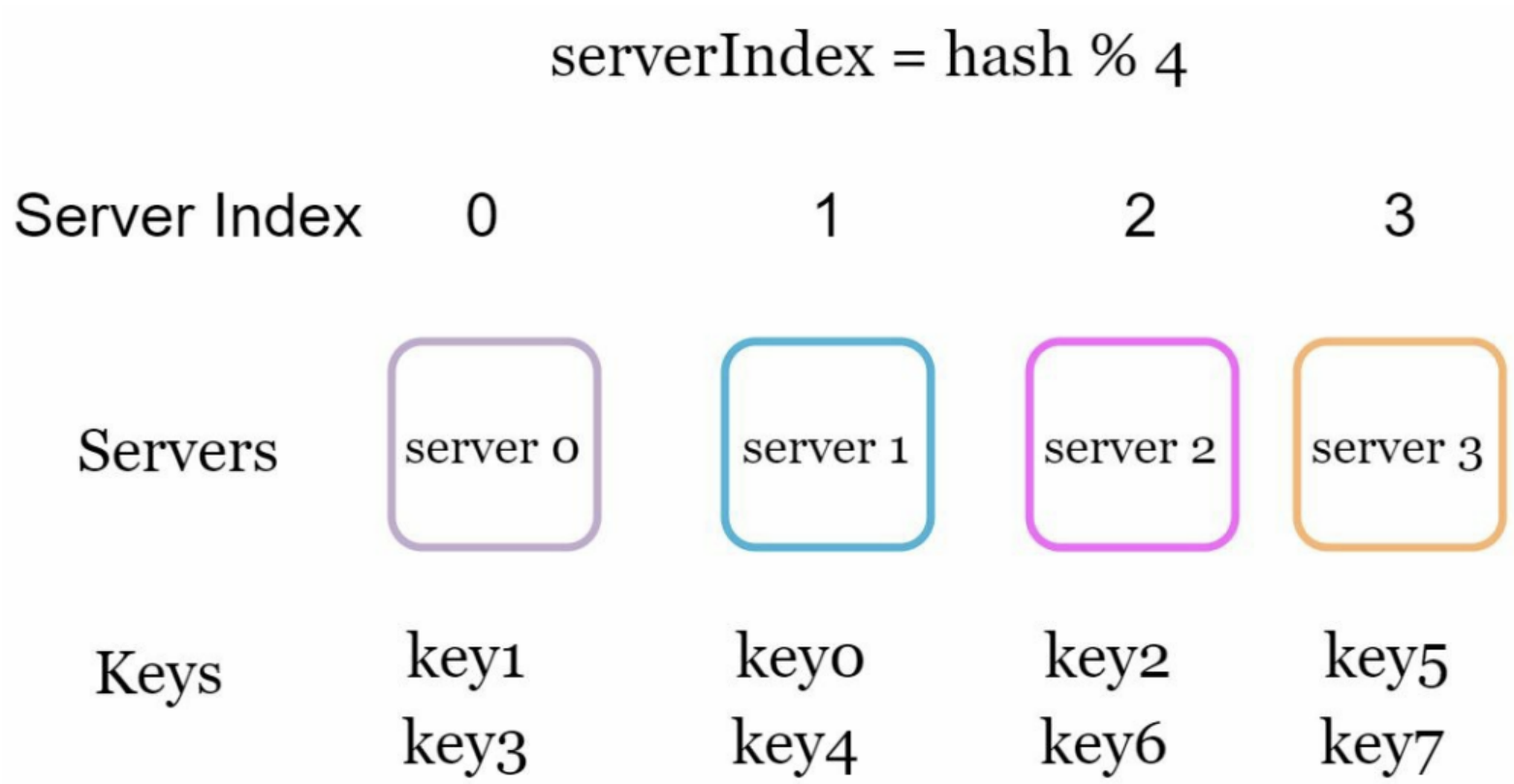
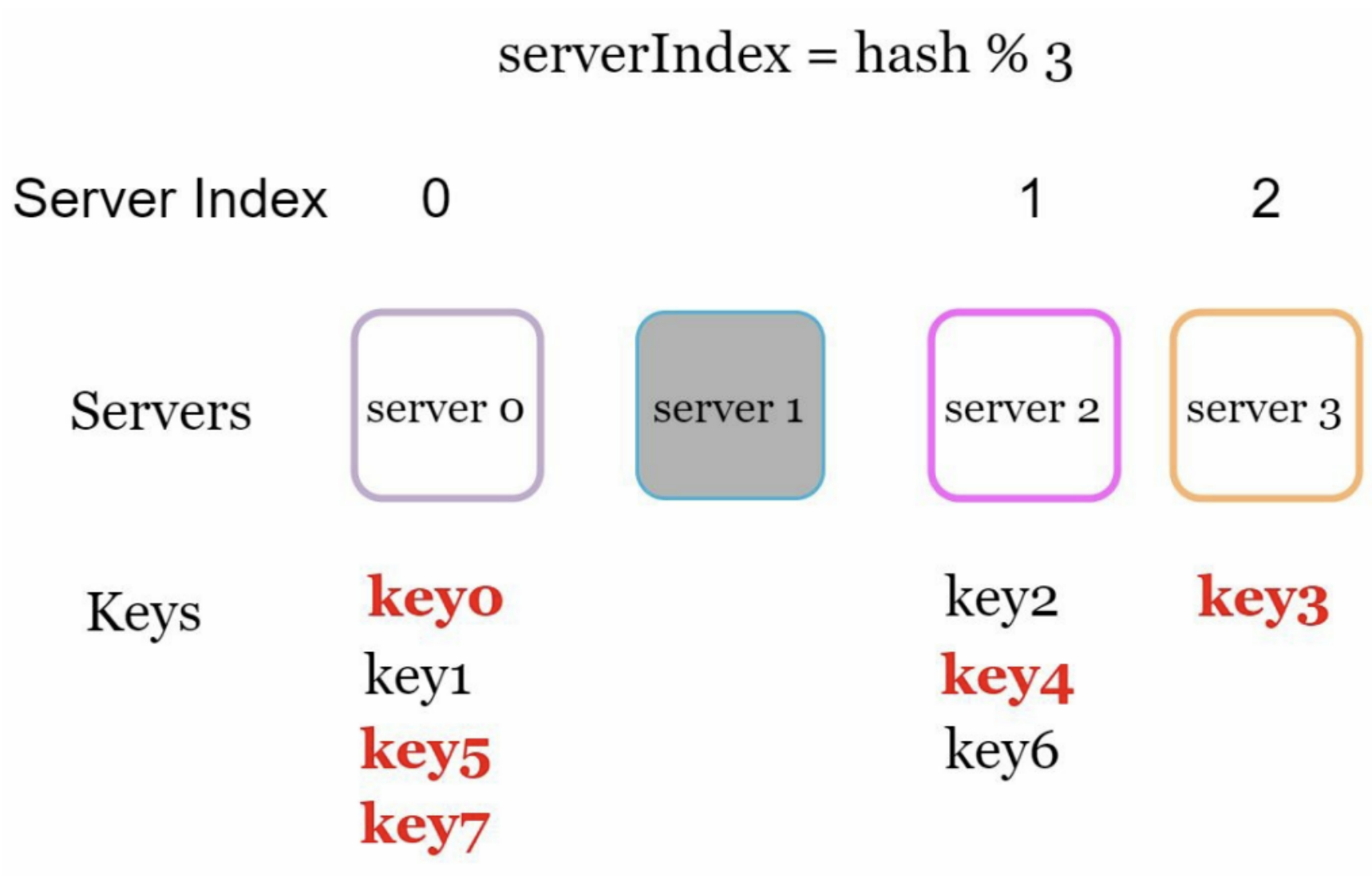
안정 해시 설계

키 재배치 문제

- 아래와 같은 방식으로 키가 배치된 경우를 생각해보자.
  - hash % 4 를 이용하여, 서버에 키를 배치했다.



- 이 때, 서버1이 죽는다면 키를 어떻게 배치해야 할까?
  - hash % 3 을 이용하여, 서버에 키를 재배치해야한다.



- 여기서 문제점은?
  - server 1이 죽어 대부분의 키가 재배치 되었음.
  - 이로 인해, 대규모 캐시 미스가 발생하게 될 것임

⇒ 이러한 문제를 해결하는 방법이 안정해시

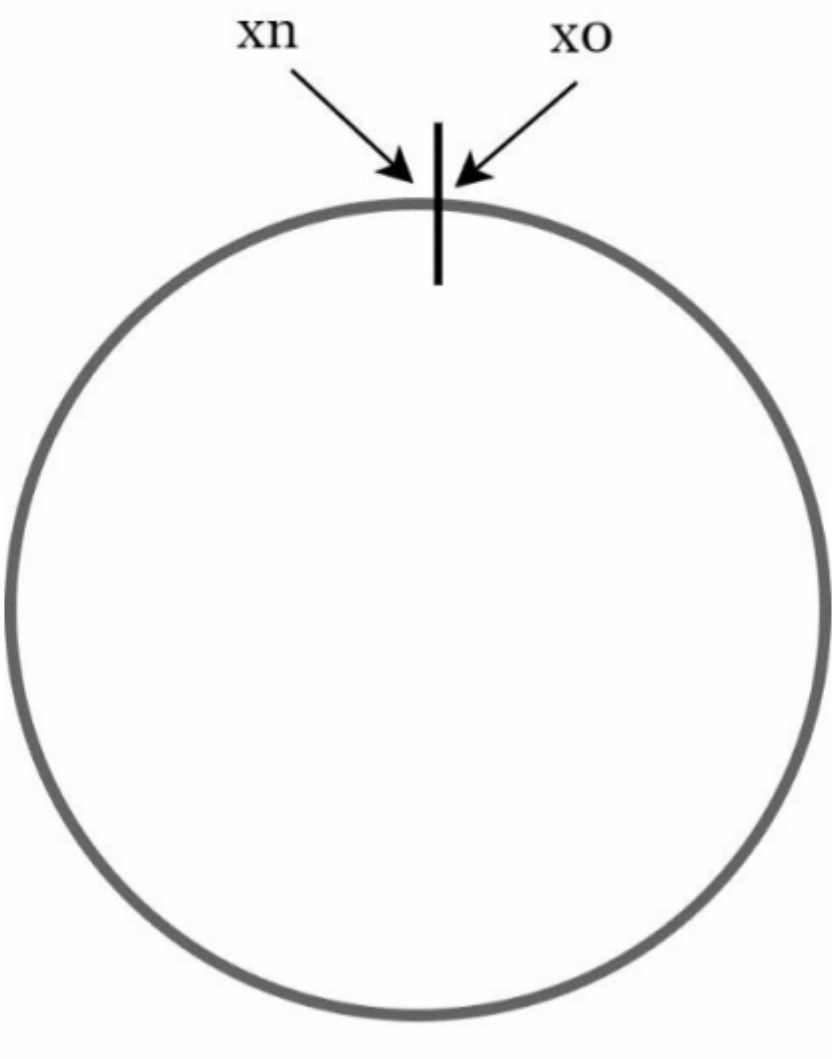
안정 해시 기본 구현

- 해시 테이블의 크기가 조정될 때, 평균적으로 k/n 개의 키만 재배치하는 해시 기술.
  - k : 키의 개수
  - n : 슬롯

⇒ 안정해시가 아닌 경우, 대부분의 키를 재배치하게 된다.

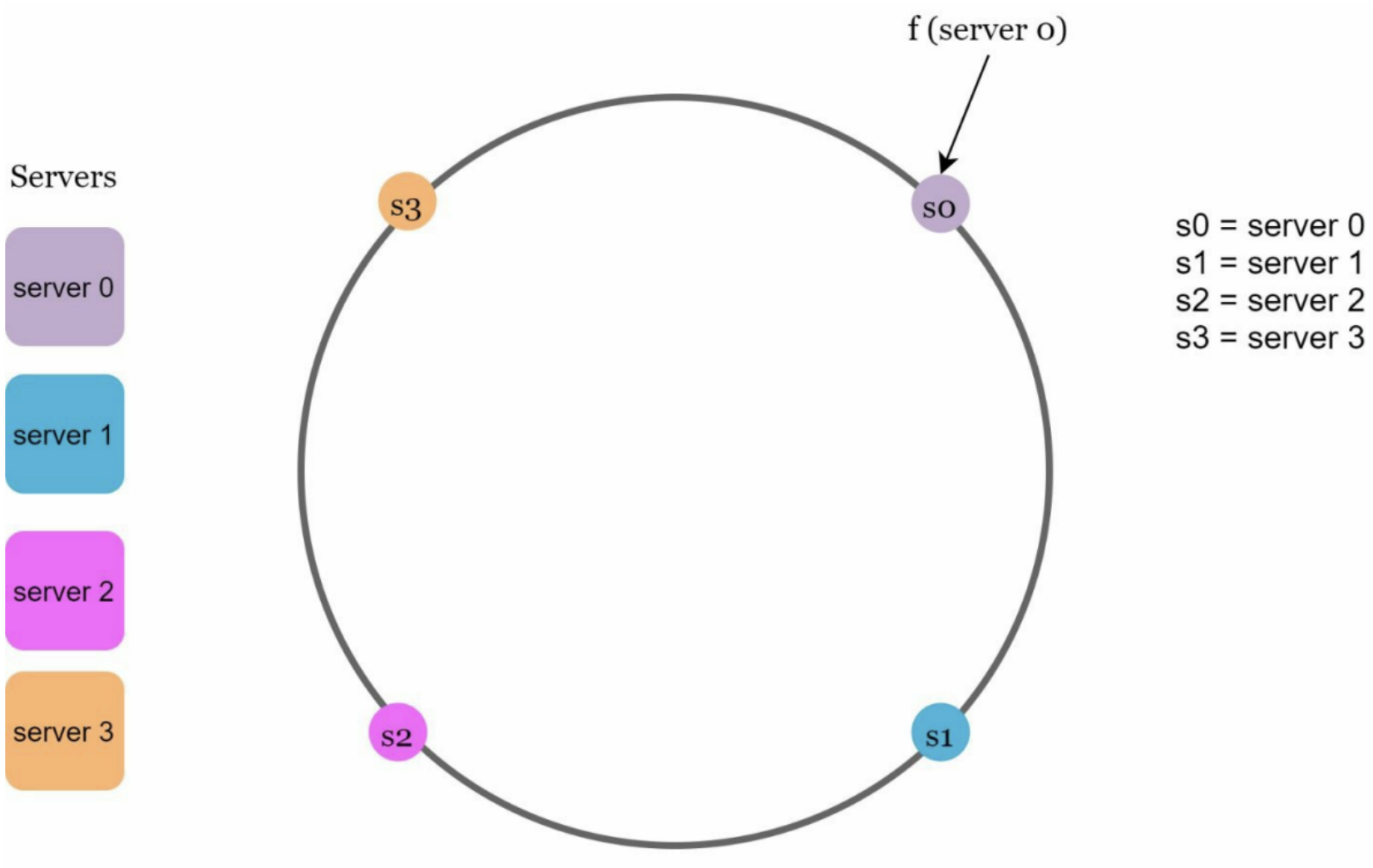
해시 공간과 해시 링

- SHA-1 알고리즘을 사용하는 경우, 해시 함수의 출력 값은 0 ~ 2^160-1 이다.
  - 아래 그림에서 x0 = 0, xn = 2^160-1



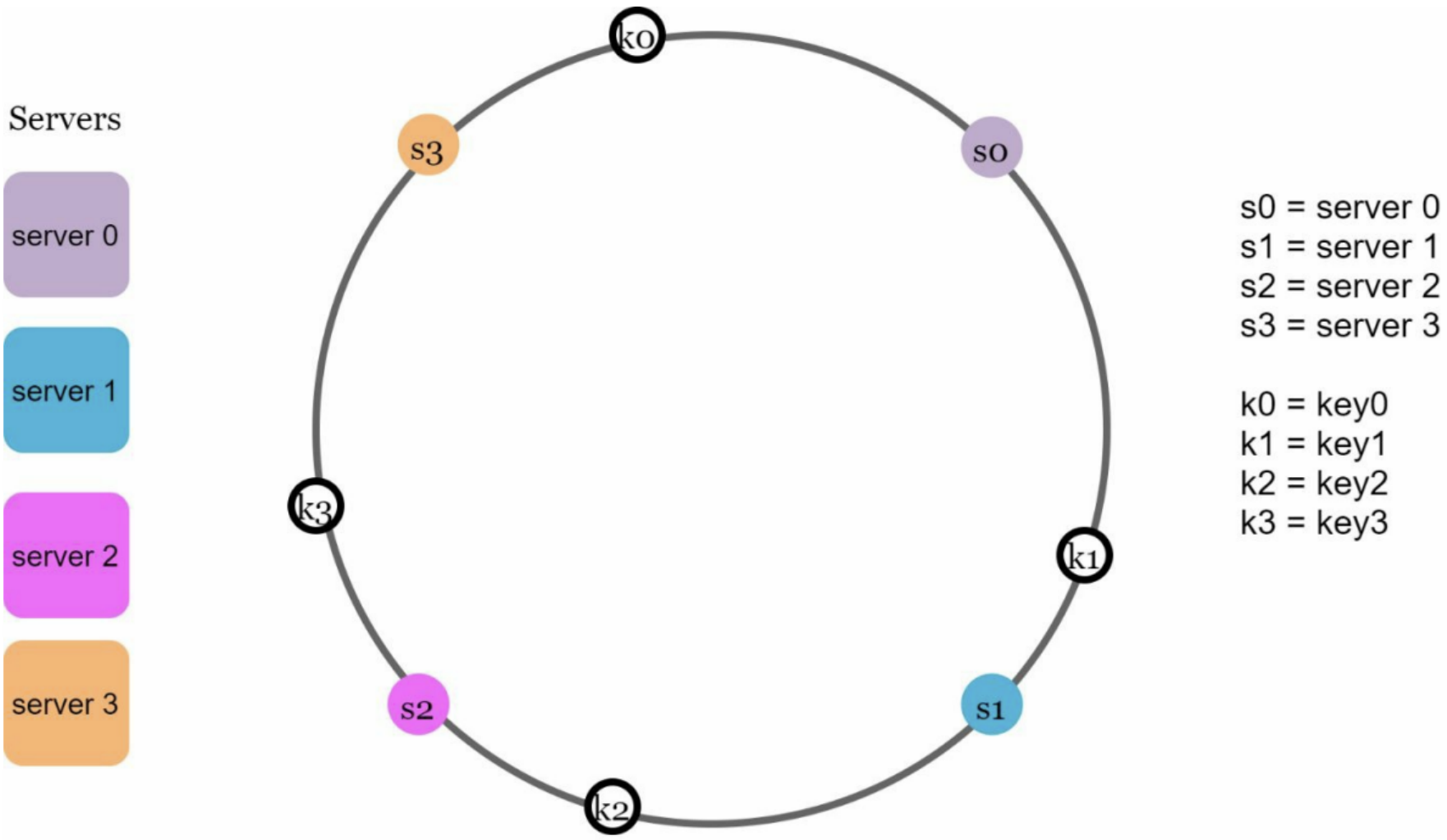
해시 서버

- 해시 함수를 이용하면, 아래와 같이 서버 IP나 이름을 해시 링에 대응 시킬 수 있다.



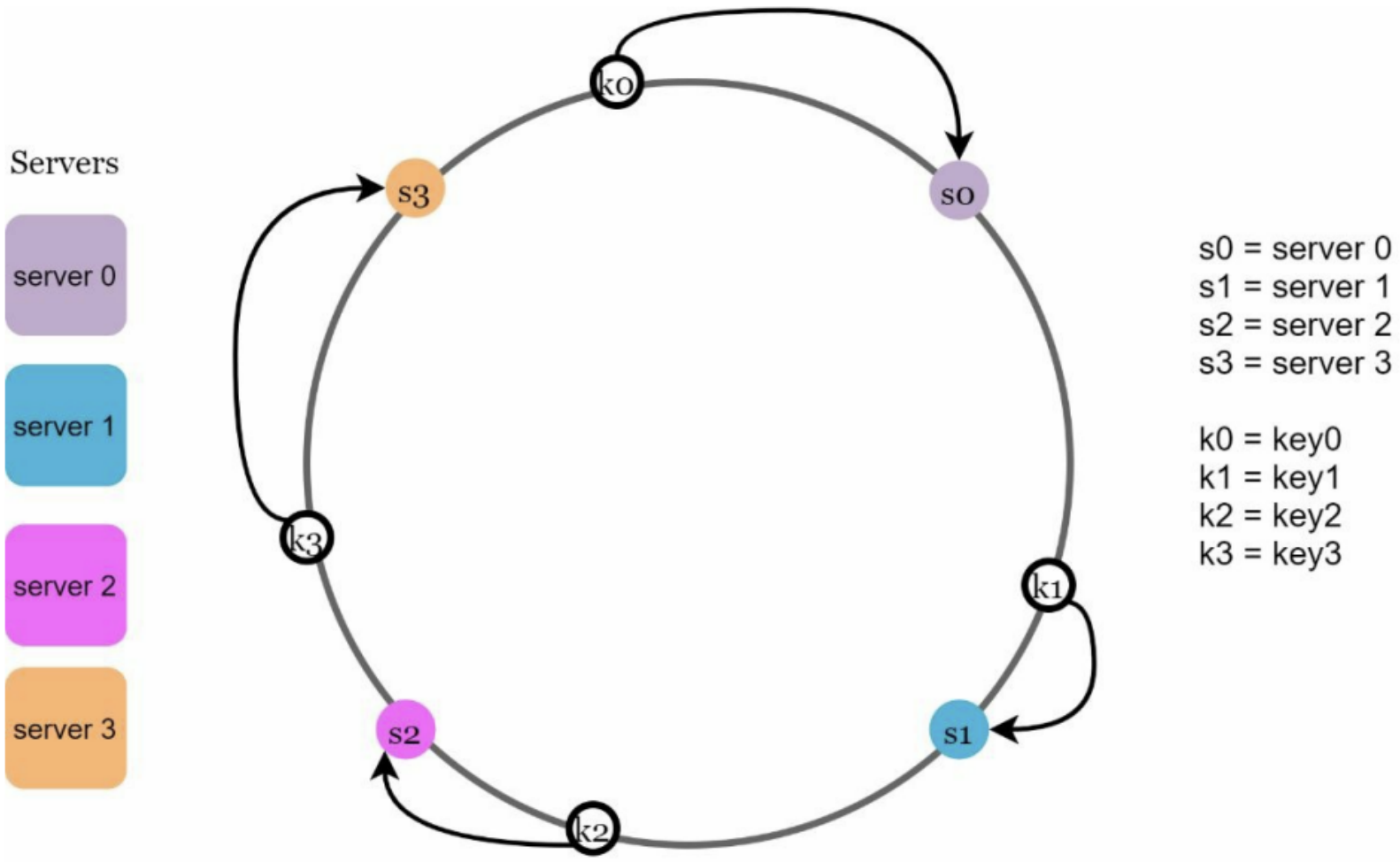
해시 키

- 키들도 아래 그림과 같이 해시 링 위의 지점에 배치할 수 있다.



서버 조회

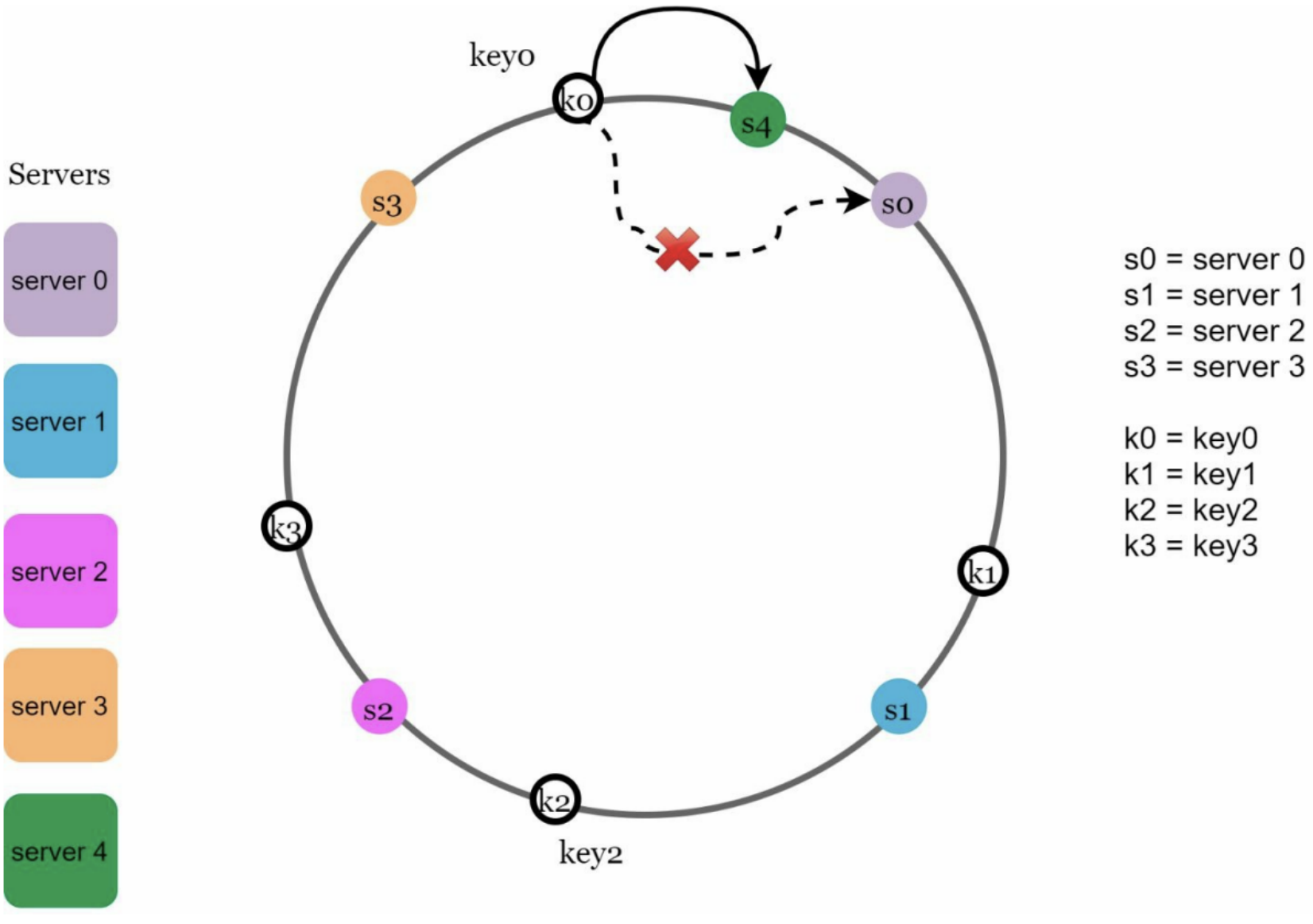
- 키가 저장되는 서버는, 키를 기준으로 시계 방향으로 탐색하면서 만나는 첫 번째 서버.



- 위 구림에서 k0 → s0, k1 → s1, k2 → s2, k3 → s3 인 것을 알 수 있다.

서버 추가

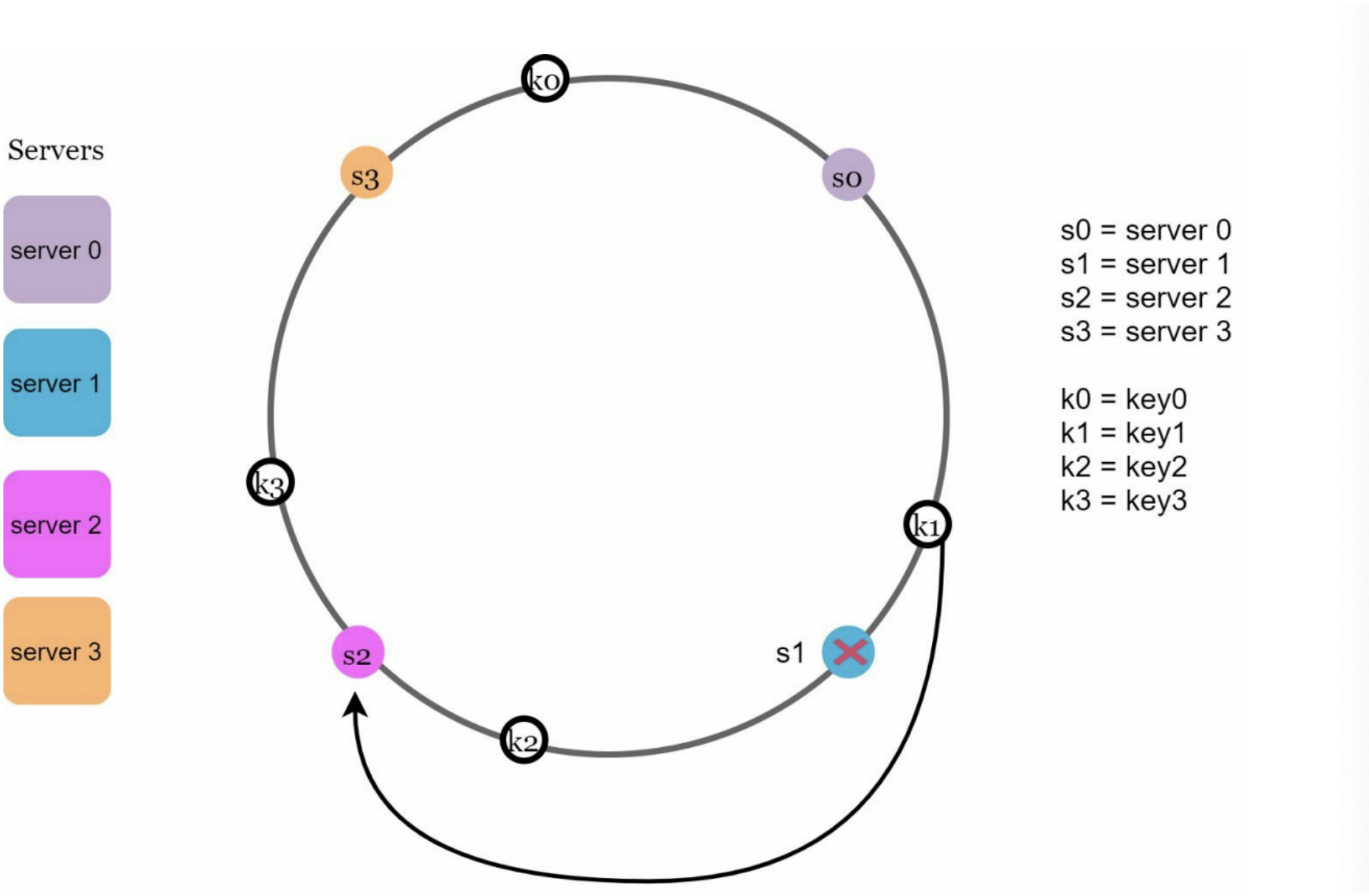
- 서버 조회의 내용을 보면 알 수 있듯이, 서버를 추가하게 되는 경우에는 모든 키를 재배치 할 필요성가 없다.



- 위와 같이 s4가 추가 되었을 때, k0 만 s4 로 재배치 되는 것을 알 수 있다.

서버 제거

- 추가와 같이, 하나의 서버가 제거 되면 키 가운데 일부만 재배치된다.



- k1만 s2로 재배치 되었음을 알 수 있다.

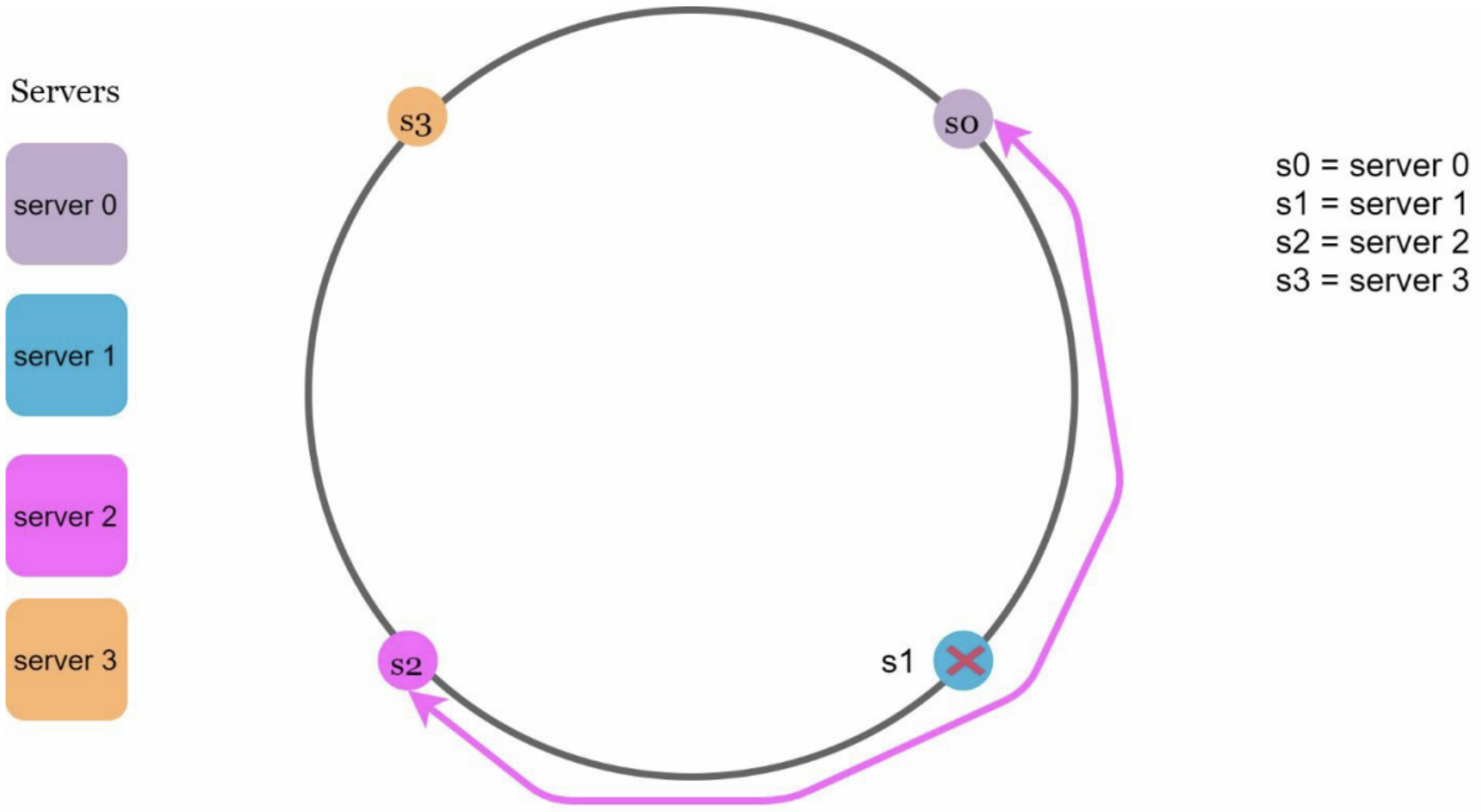
기본 구현법의 두 가지 문제

안정 해시 알고리즘은 다음과 같은 두 가지 절차를 거친다.

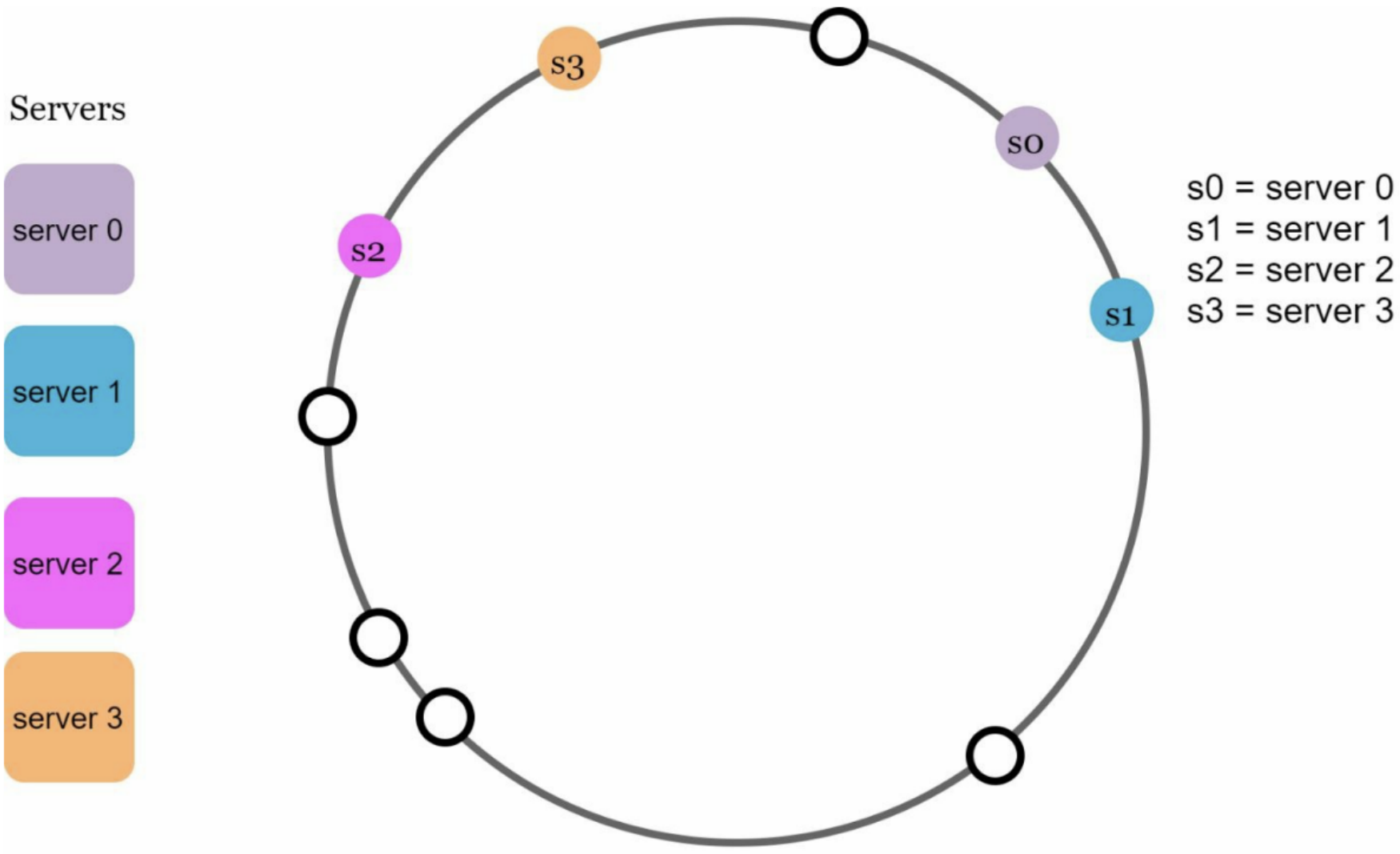
- 서버와 키를 균등 분포 해시 함수를 사용해 해시 링에 배치한다.
- 키의 위치에서 링을 시계 방향으로 탐색하다 만나는 최초의 서버가 키가 저장될 서버이다.

여기서 두 가지 문제가 발생한다.

- 추가 혹은 삭제 상황에서, **파티션 크기를 균등하게 유지하는 것이 불가능**하다.
  - 파티션 = 서버와 서버 사이의 해시 공간
  - 아래 그림과 같이 s1이 제거 되면, s0-s2 사이의 해시 공간이 다른 파티션보다 커지게 된다.



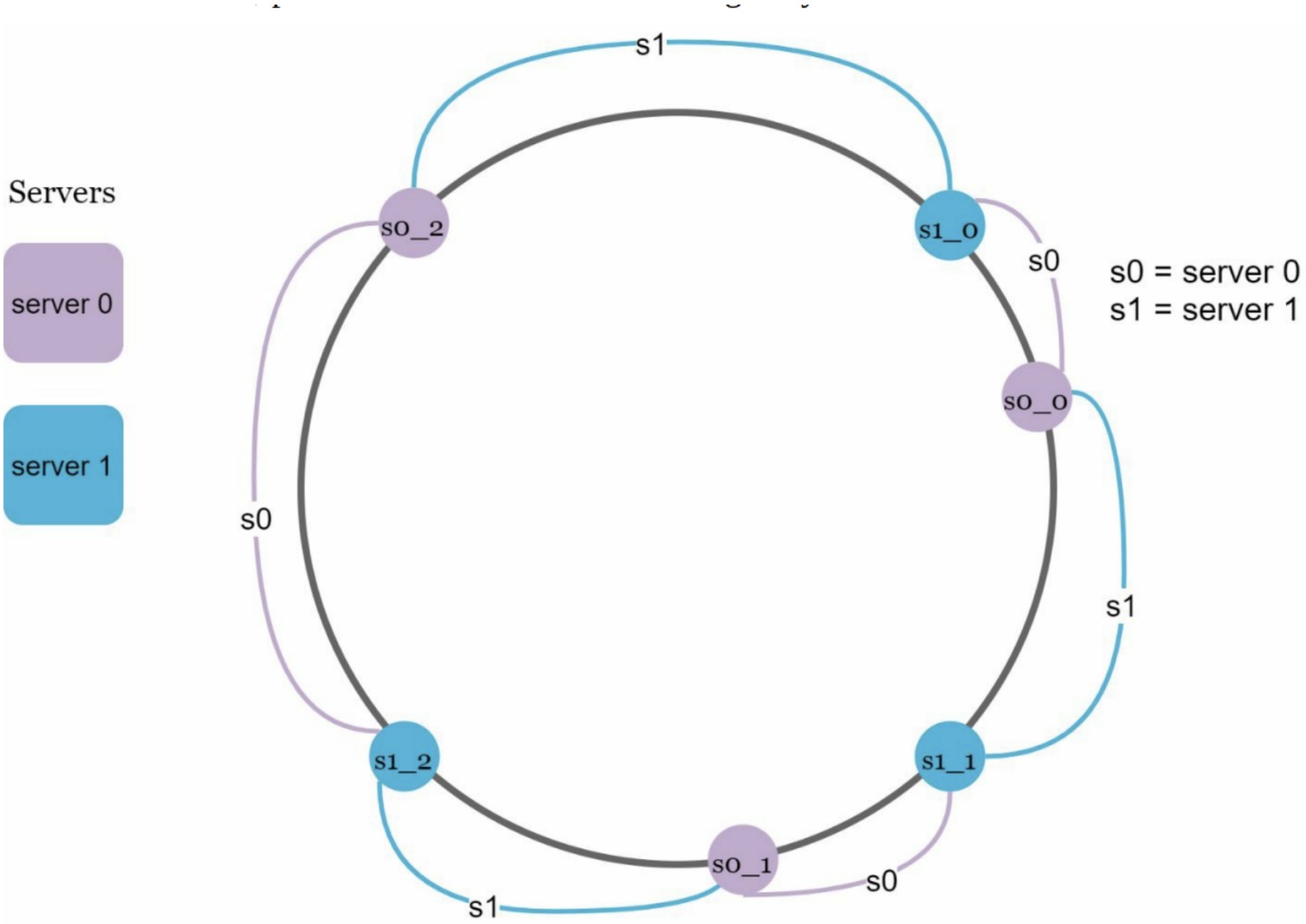
- **키의 균등 분포를 달성하기 어렵다**는 문제도 있다.
  - 아래와 같이 키가 분포되어 있다면, s1과 s3는 아무 데이터를 갖지 않고 s2는 4개의 데이터를 갖게 된다.



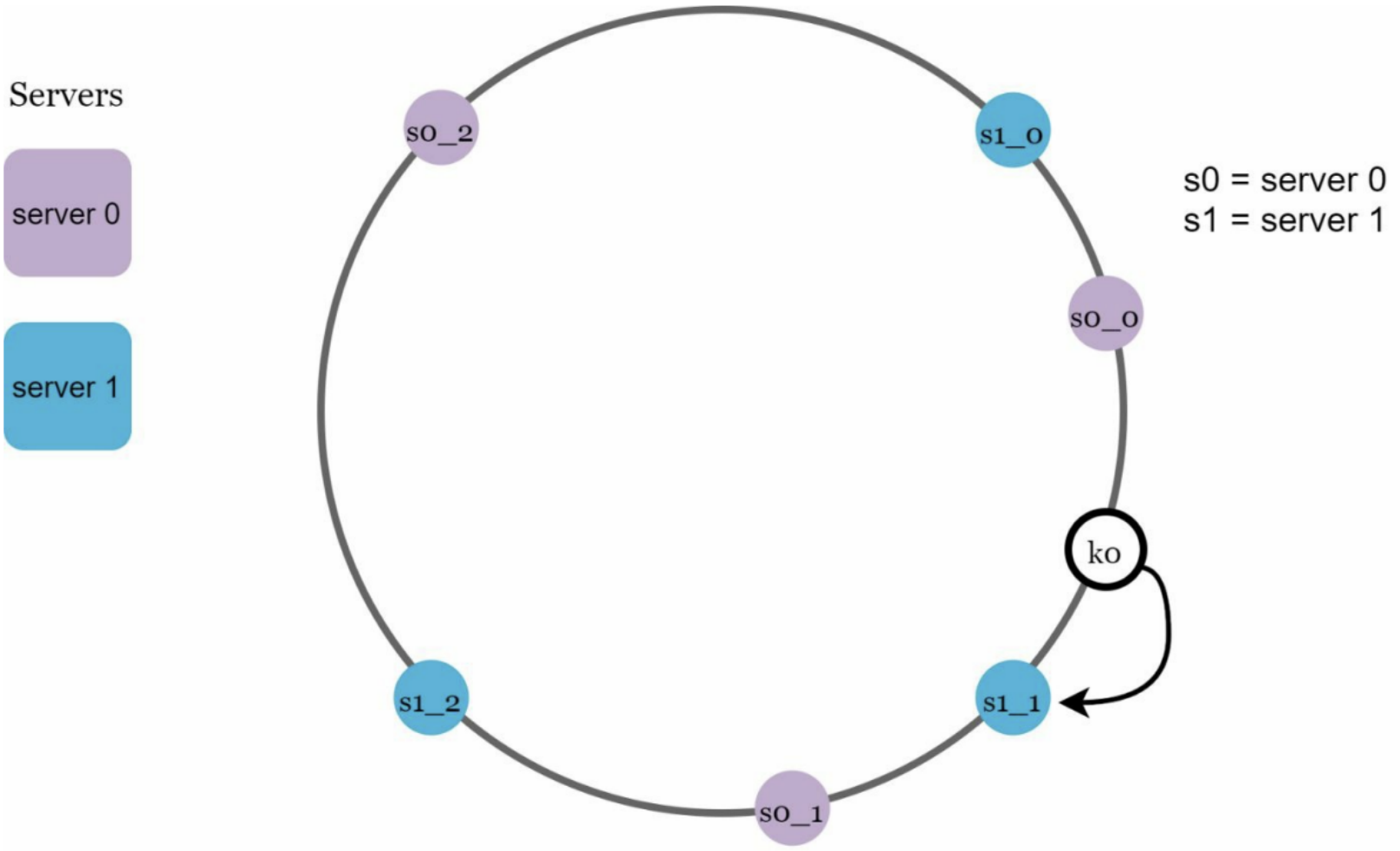
⇒ 위 두개의 문제를 해결하기 위해 **가상 노드**라는 기법을 알아 볼 것이다.

가상 노드 (virtual node)

- 가상 노드는 실제 노드 또는 서버를 가리키는 노드이다.
- 하나의 서버는 여러 개의 가상 노드를 가질 수 있다.
  - 아래 그림을 보면, s0과 s1은 3개의 가상 노드를 가진다.
    - 실제 시스템은 3개보다 훨씬 큰 가상 노드를 가진다.
- 이렇게 각 서버가 여러 개의 노드로 나뉘어, 각 서버는 여러 파티션을 관리하게 되었다.
  - 아래 그림에서 각각 s1, s0로 이어진 선의 범위의 합이 관리하게 되는 총 파티션이다.



- 가상 노드가 없었던 예시와 같이, 키가 가장 먼저 만나는 가상 노드가 가리키는 서버에 키가 저장된다.

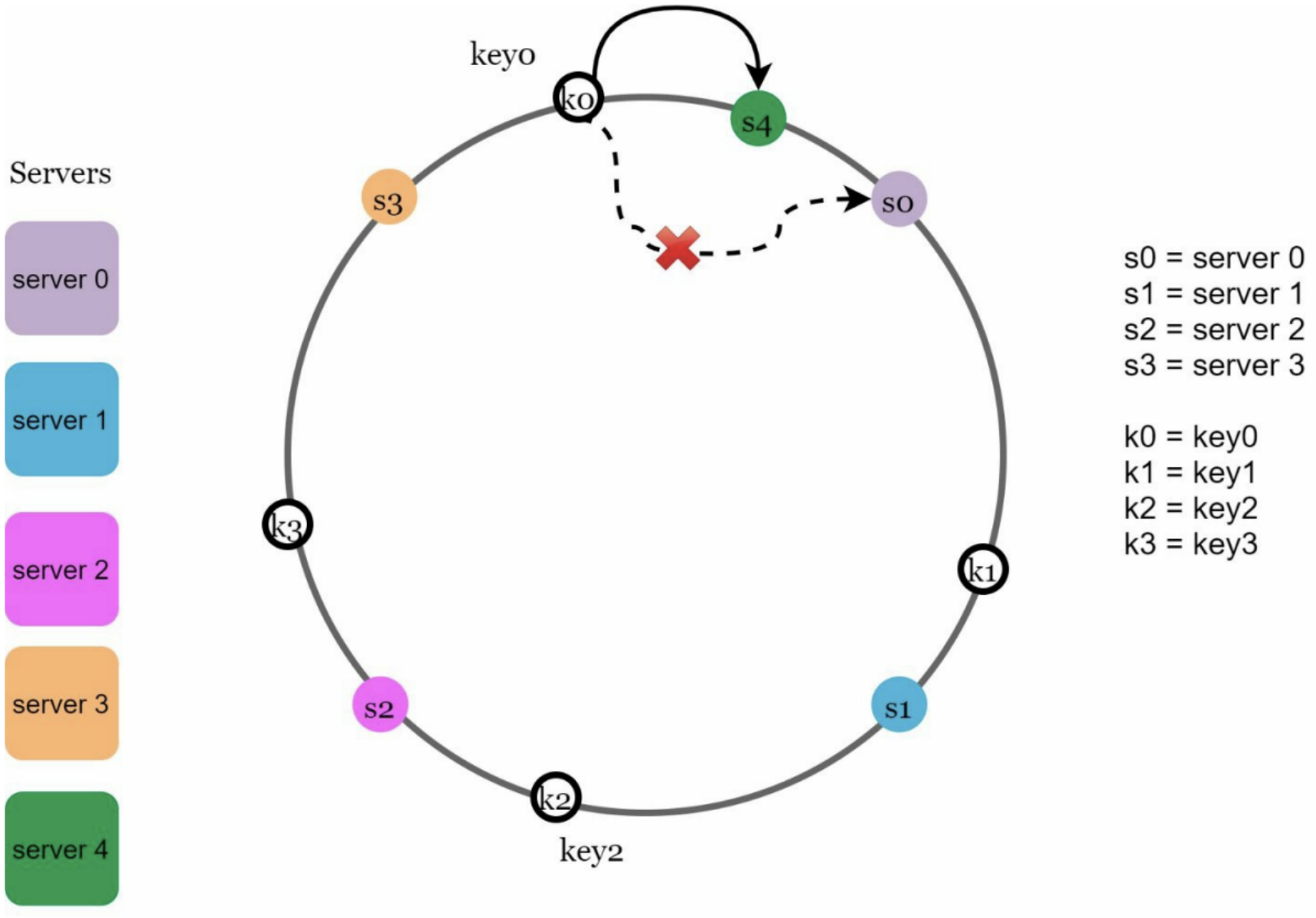


이렇게 가상 노드의 개수를 늘리면, 키의 분포는 균등해진다.

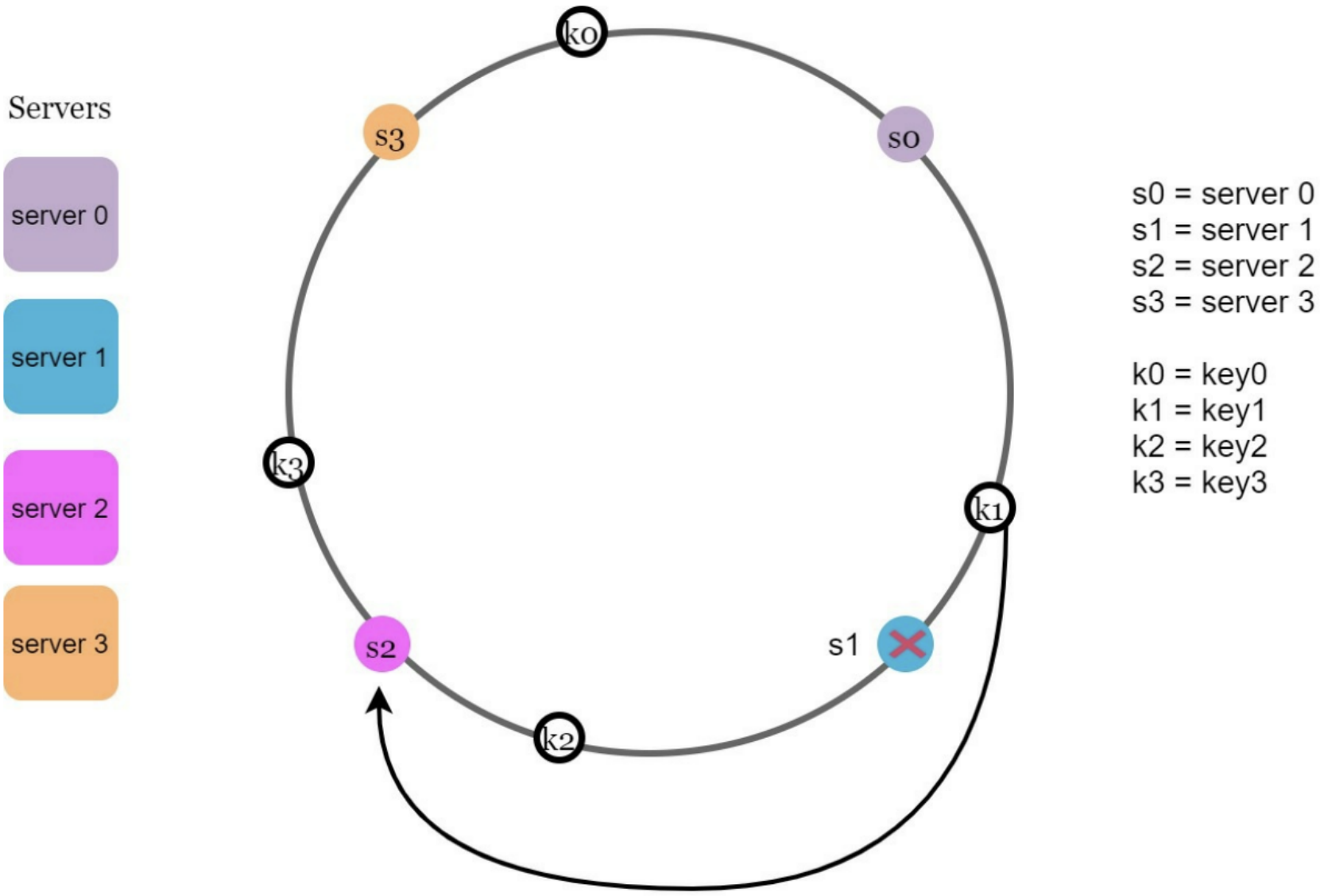
⇒ 표준편차가 작아져 데이터가 고르게 분포되기 때문.

⇒ 가상 노드의 개수가 많아질수록 표준편차는 작아지는데, 가상 노드 데이터 저장에 필요한 공간이 많아지기 때문에 항상 trade-off를 생각하자.

재배치할 키를 결정



- 위 그림과 같이 s4가 추가 될 경우, 시계 방향으로 s3 ~ s4 범위의 키 들을 재배치해야한다.



- 위 그림과 같이 s1이 삭제 된다면, 시계 방향으로 s0 ~ s2 사이의 키들이 재배치 되어야한다.

마치며

안정 해시의 이점

- 서버가 추가되거나 삭제될 때, 재배치되는 키의 수를 최소화.
- 데이터가 균등하게 분포되어, 스케일 아웃을 달성하기 쉬움.
- 핫스팟 키 문제를 줄인다.
    - 특정한 샤드에 대한 접근이 지나치게 빈버하면 서버 과부화.
    - 안정해시는 데이터를 더 균등하게 분배하므로 이런 문제가 생길 가능성을 줄인다.