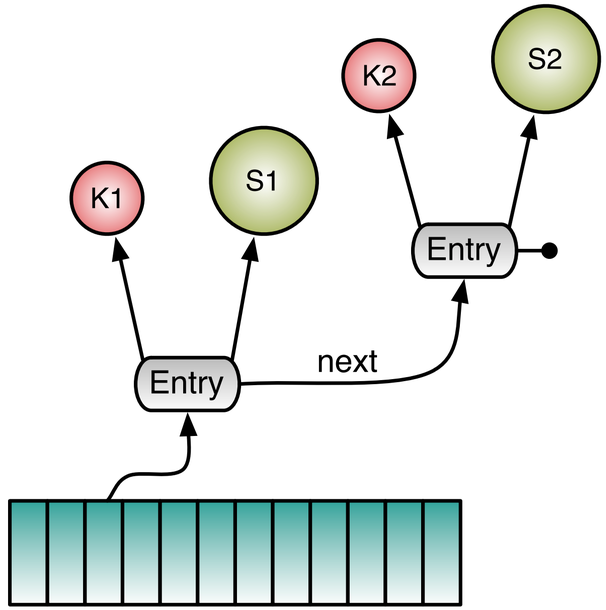
Copy-on-Write Hash Map

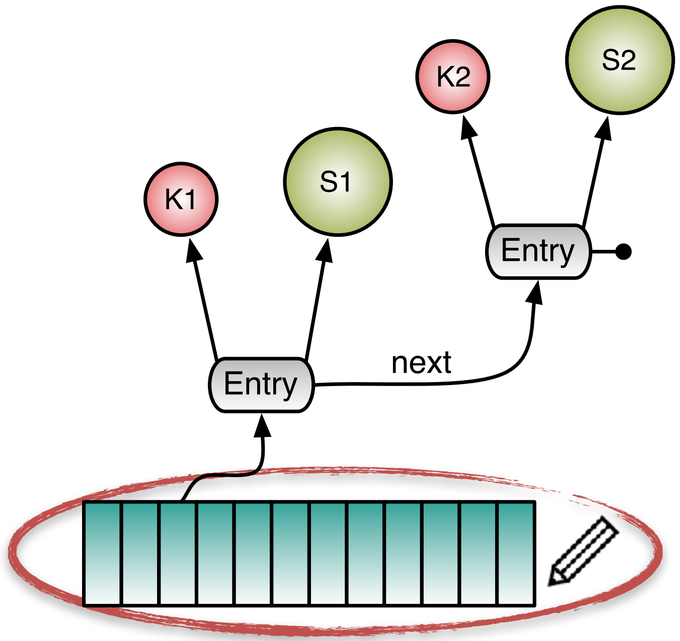
```
Map<K, S>
```




```
Map<K, S> {

    Entry<K, S>[] table;
}

Entry<K, S> {

    final K key;
    S state;
    Entry next;
}
```

Copy-on-Write Hash Map

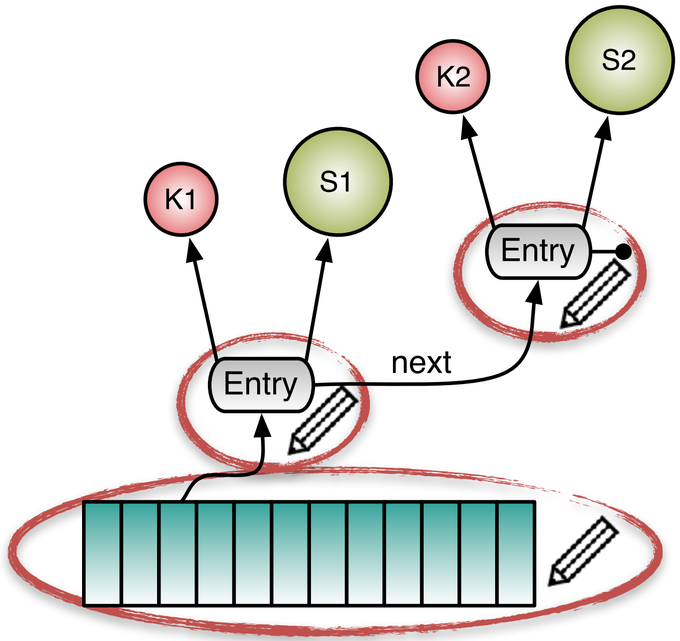
```
Map<K, S>
```




```
Map<K, S> {

    Entry<K, S>[] table;
}

Entry<K, S> {

    final K key;
    S state;
    Entry next;
}
```

Copy-on-Write Hash Map

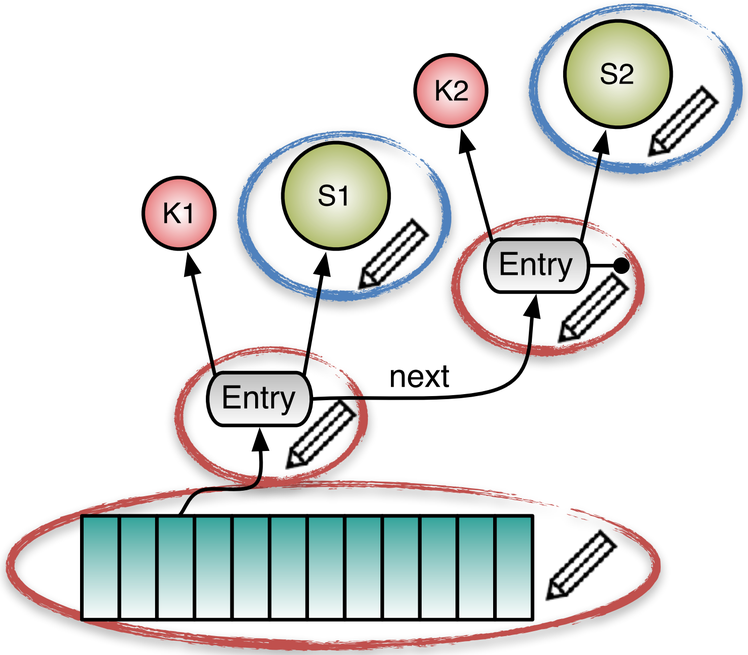
```
Map<K, S>
```




```
Map<K, S> {

    Entry<K, S>[] table;
}

Entry<K, S> {

    final K key;
    S state;
    Entry next;
}
```

Copy-on-Write Hash Map

```
Map<K, S>
```

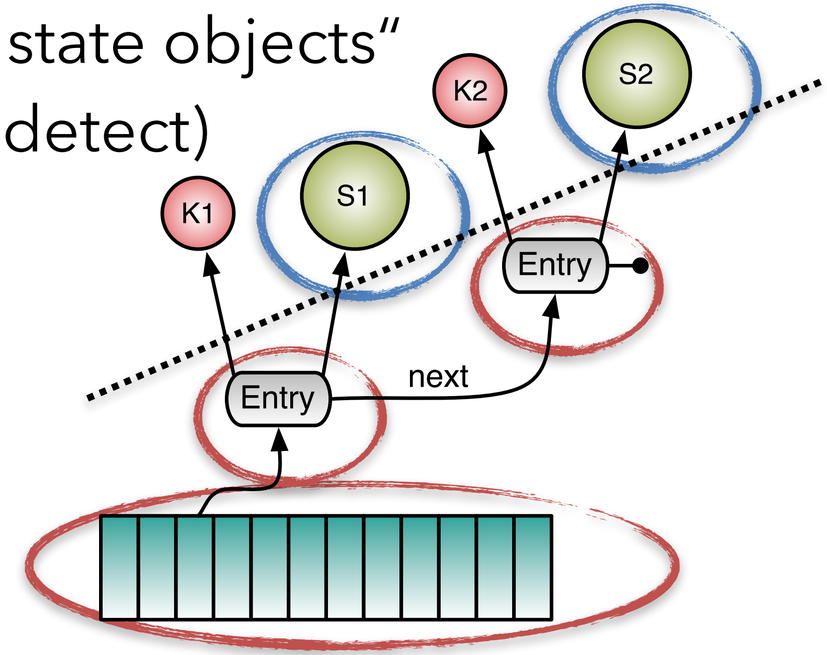



```
Map<K, S> {

    Entry<K, S>[] table;
}

Entry<K, S> {

    final K key;
    S state;
    Entry next;
}
```

Copy-on-Write Hash Map

`Map<K, S>`

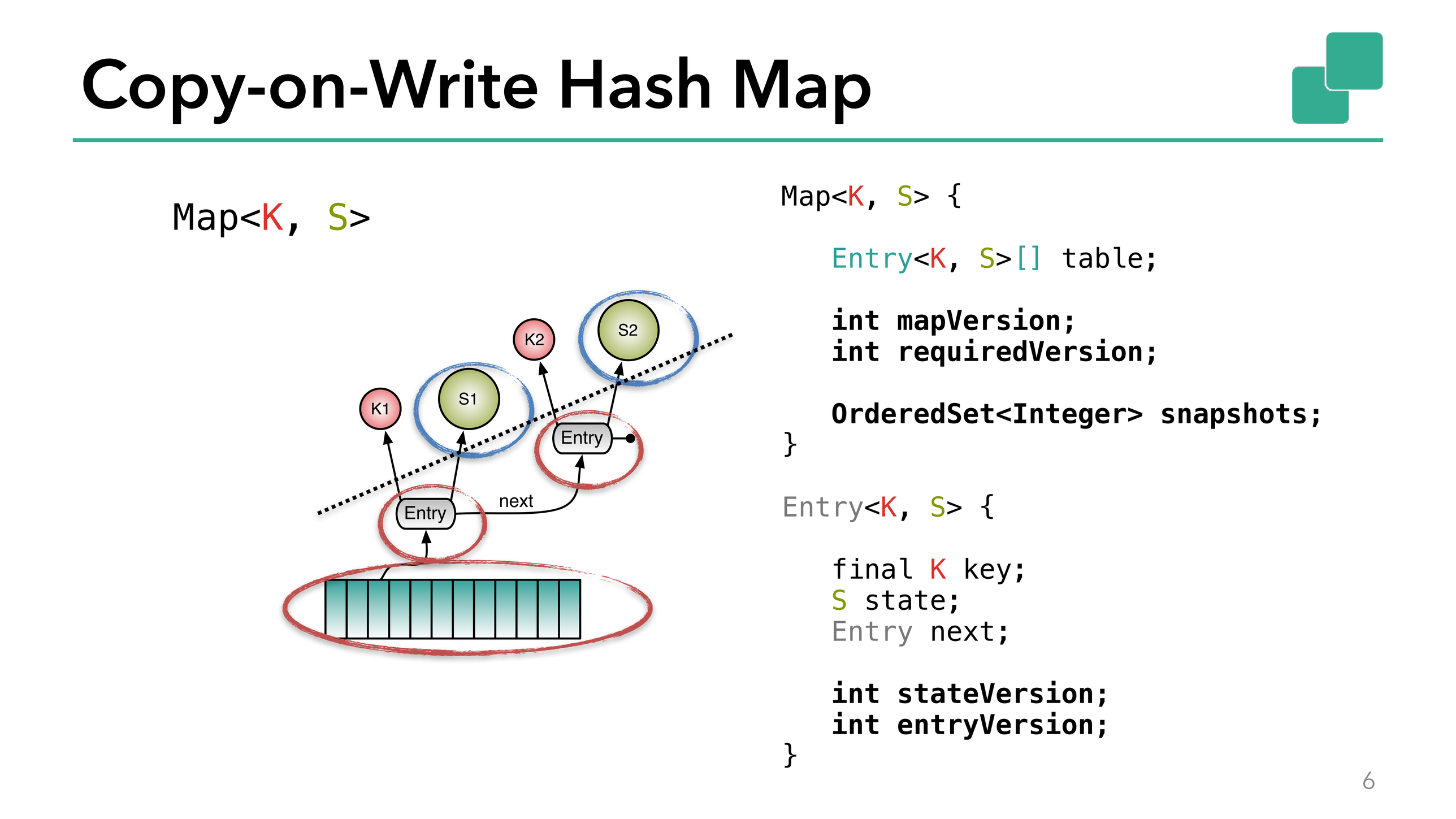
„User modifies state objects“
(hard to detect)




```
Map<K, S> {

    Entry<K, S>[] table;
}

Entry<K, S> {

    final K key;
    S state;
    Entry next;
}
```

„Structural changes to map“
(easy to detect)

Copy-on-Write Hash Map



Map<K, S>


```
Map<K, S> {

    Entry<K, S>[] table;

    int mapVersion;
    int requiredVersion;

    OrderedSet<Integer> snapshots;
}

Entry<K, S> {

    final K key;
    S state;
    Entry next;

    int stateVersion;
    int entryVersion;
}
```

Copy-on-Write Hash Map

Create Snapshot:

1. Flat array-copy the table array
2. snapshots.add(mapVersion);
3. ++mapVersion;
4. requiredVersion = mapVersion;

Release Snapshot:

1. snapshots.remove(releaseVersion);
2. requiredVersion = snapshots.getMax();

```
Map<K, S> {

   Entry<K, S>[] table;

   int mapVersion;
   int requiredVersion;

   OrderedSet<Integer> snapshots;
}

Entry<K, S> {

   final K key;
   S state;
   Entry next;

   int stateVersion;
   int entryVersion;
}
```

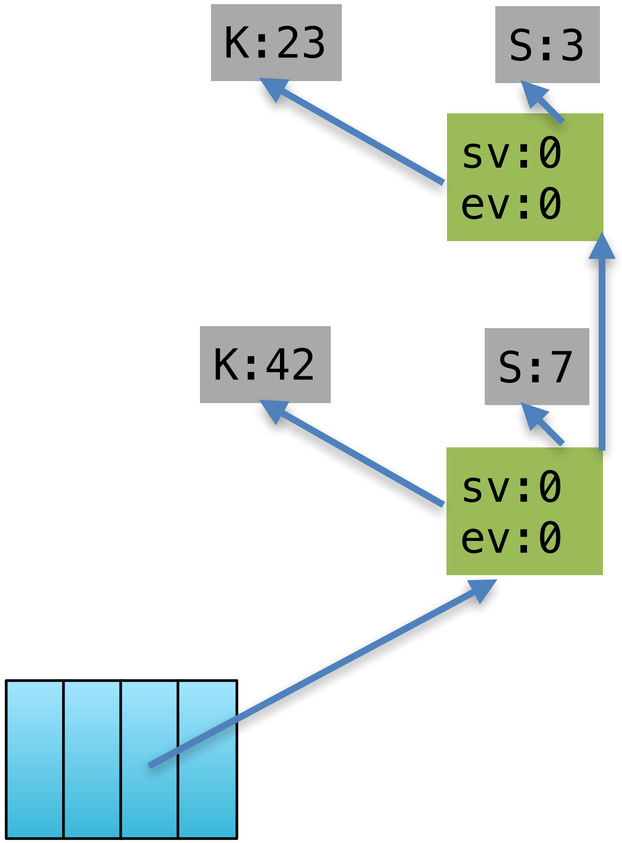
Copy-on-Write Hash Map - 2 Golden Rules

1. *Whenever a map entry* `e` *is modified and* `e.entryVersion < map.requiredVersion`*, first copy the entry and redirect pointers to the copy. Set* `e.entryVersion = map.mapVersion`*. Pointer redirection can trigger recursive application of rule 1 to other entries.*

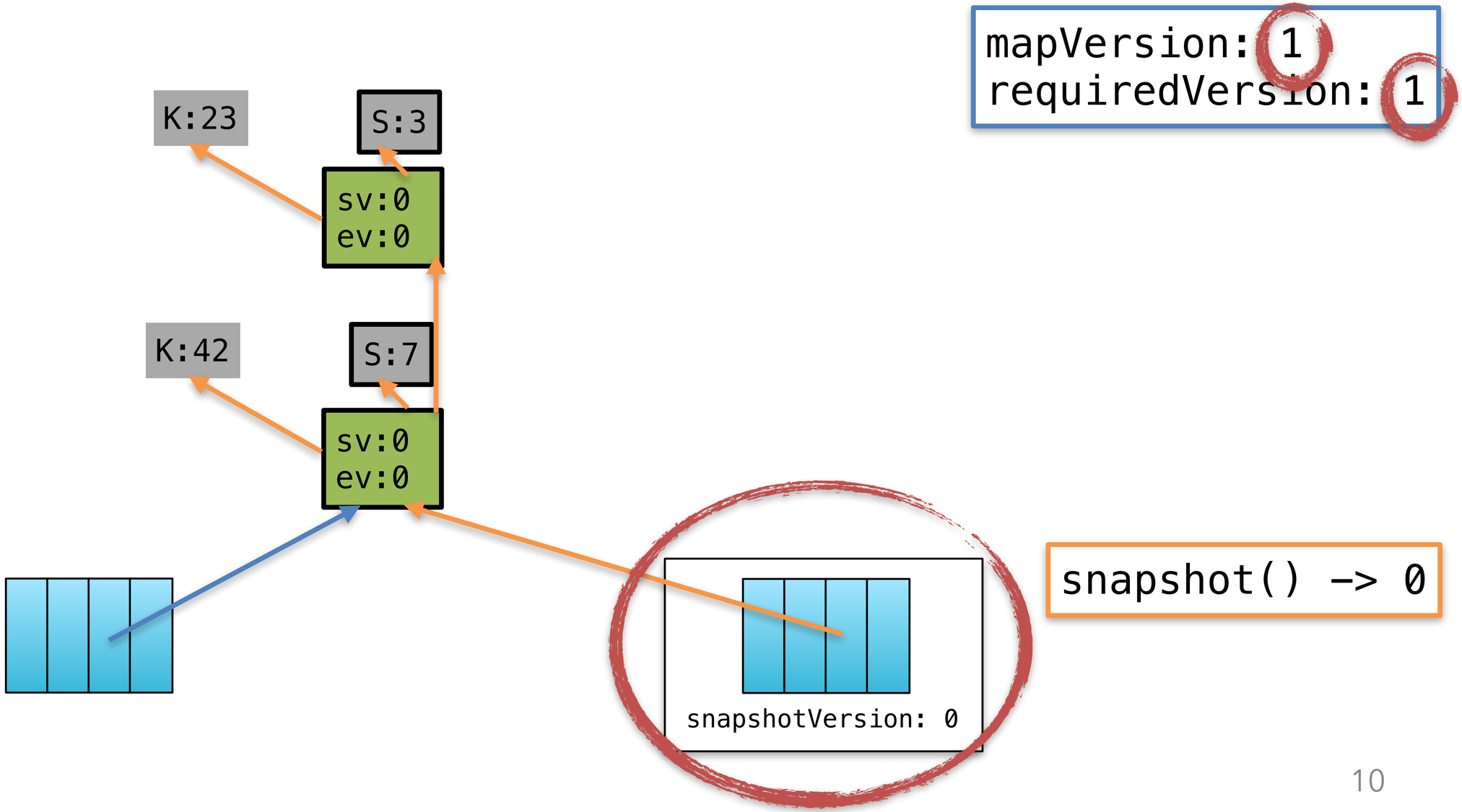
2. *Before handing the state object* `s` *of entry* `e` *to a caller, if* `e.stateVersion < map.requiredVersion`*, create a deep copy of* `s` *and redirect the pointer to* `s` *in* `e` *to the copy. Set* `e.stateVersion = map.mapVersion`. Then return the copy to the caller. *Applying rule 2 can trigger rule 1.*

Copy-on-Write Hash Map - Example

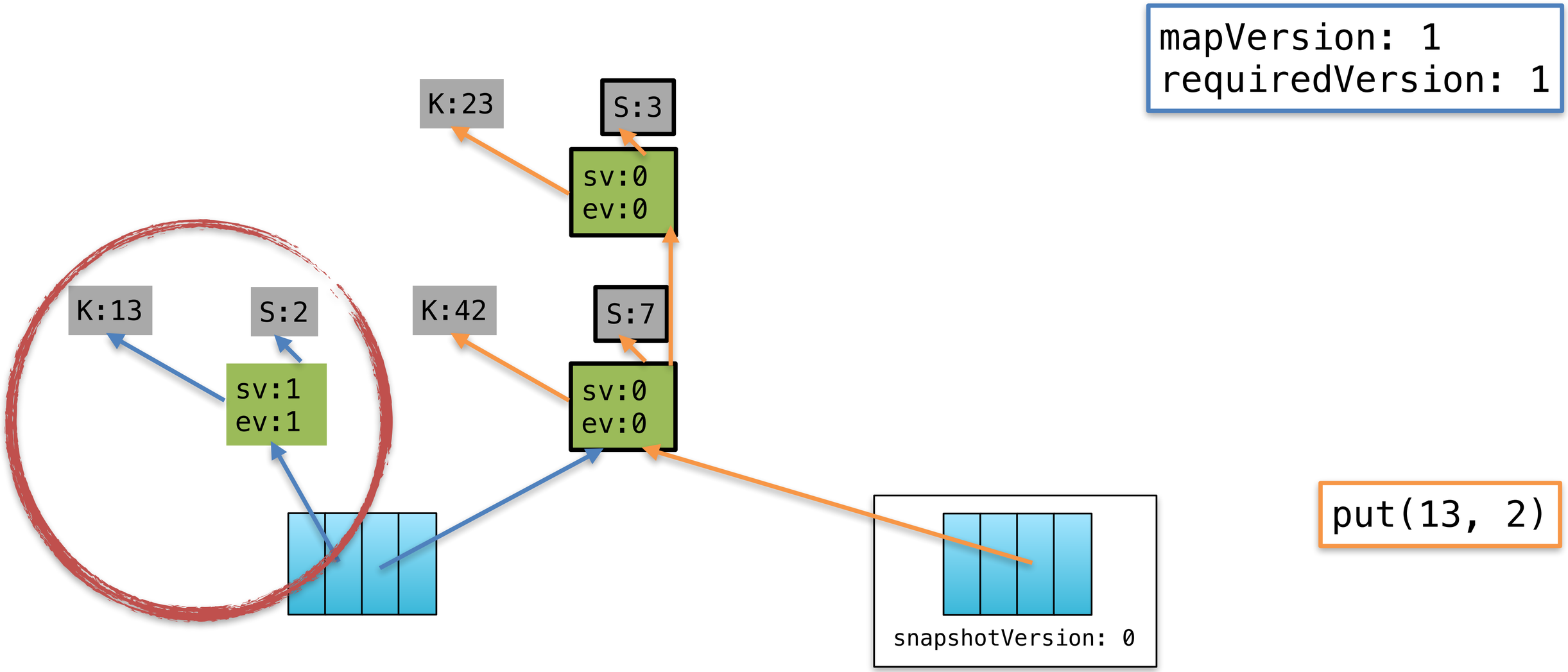
```
mapVersion: 0
requiredVersion: 0
```



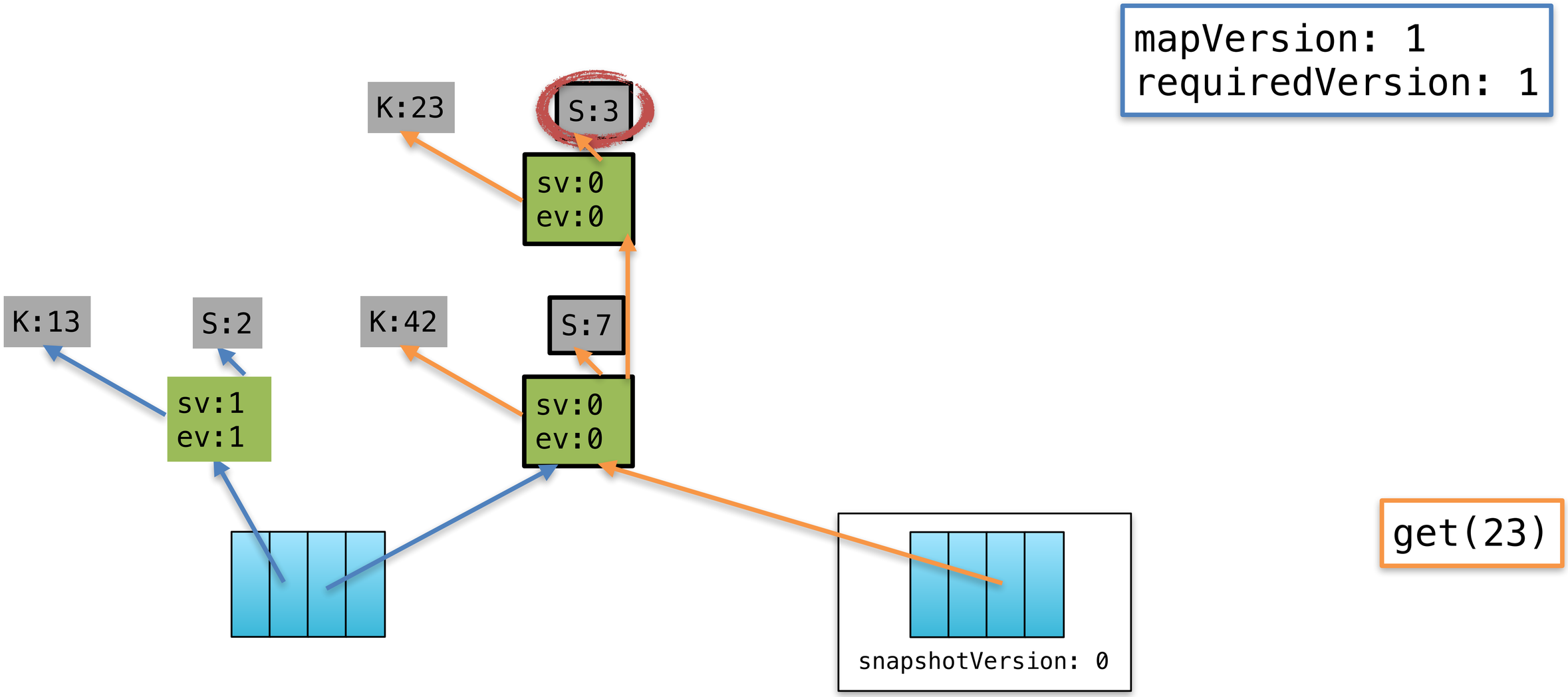
Copy-on-Write Hash Map - Example



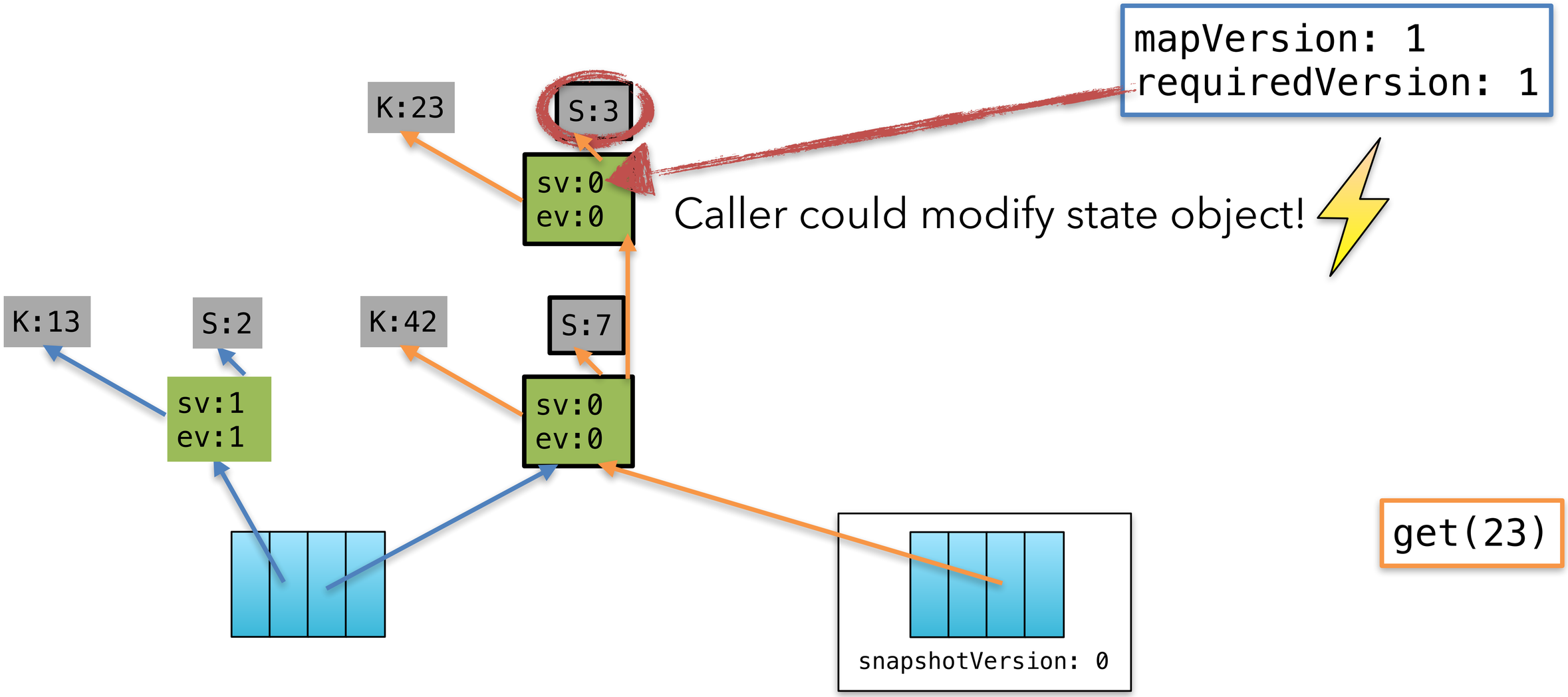
Copy-on-Write Hash Map - Example



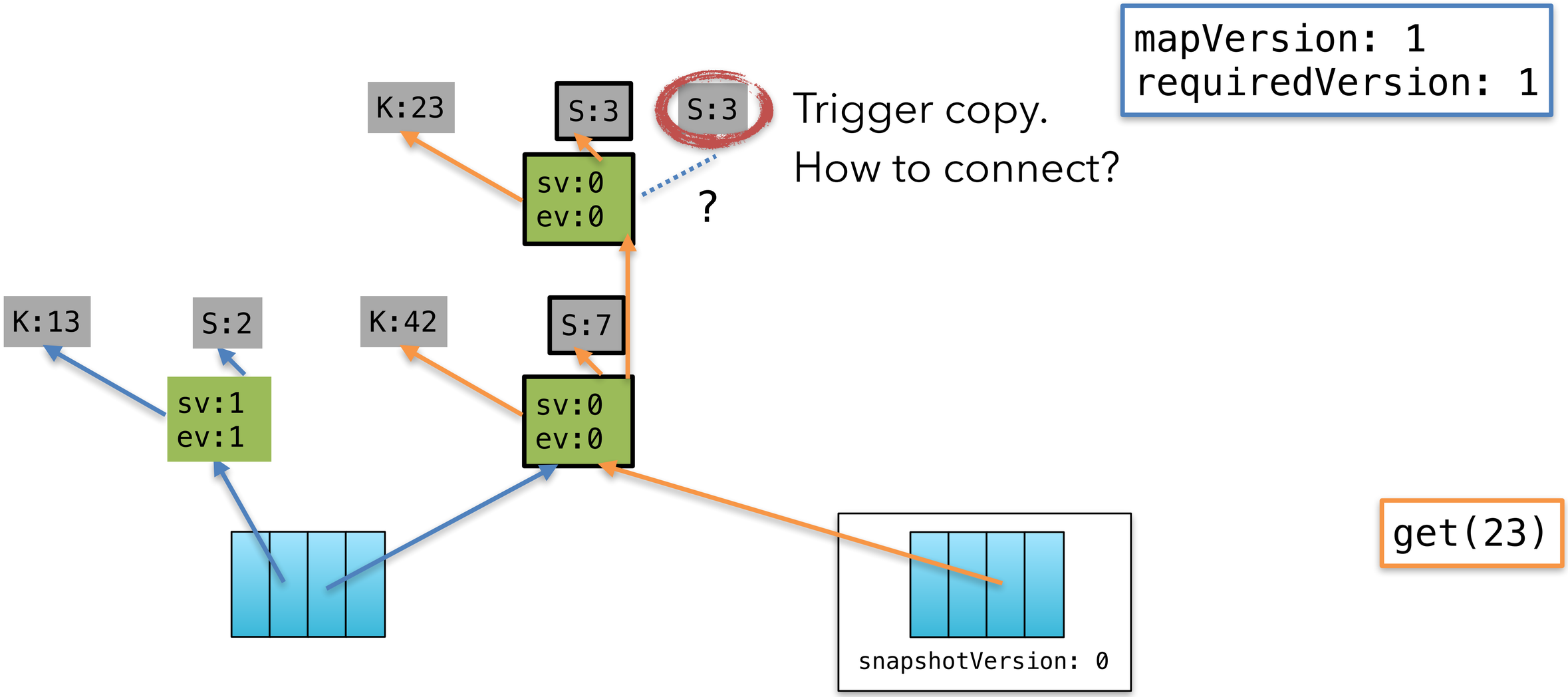
Copy-on-Write Hash Map - Example



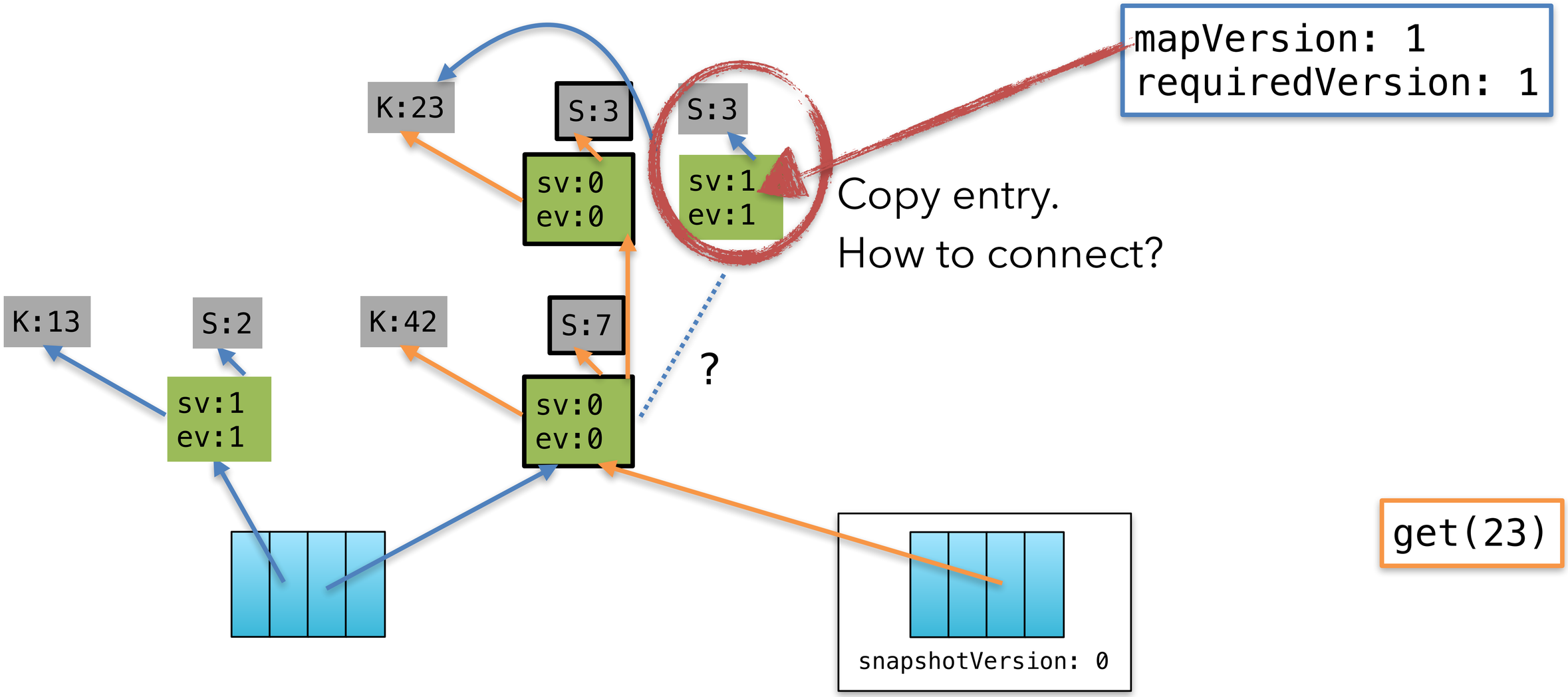
Copy-on-Write Hash Map - Example



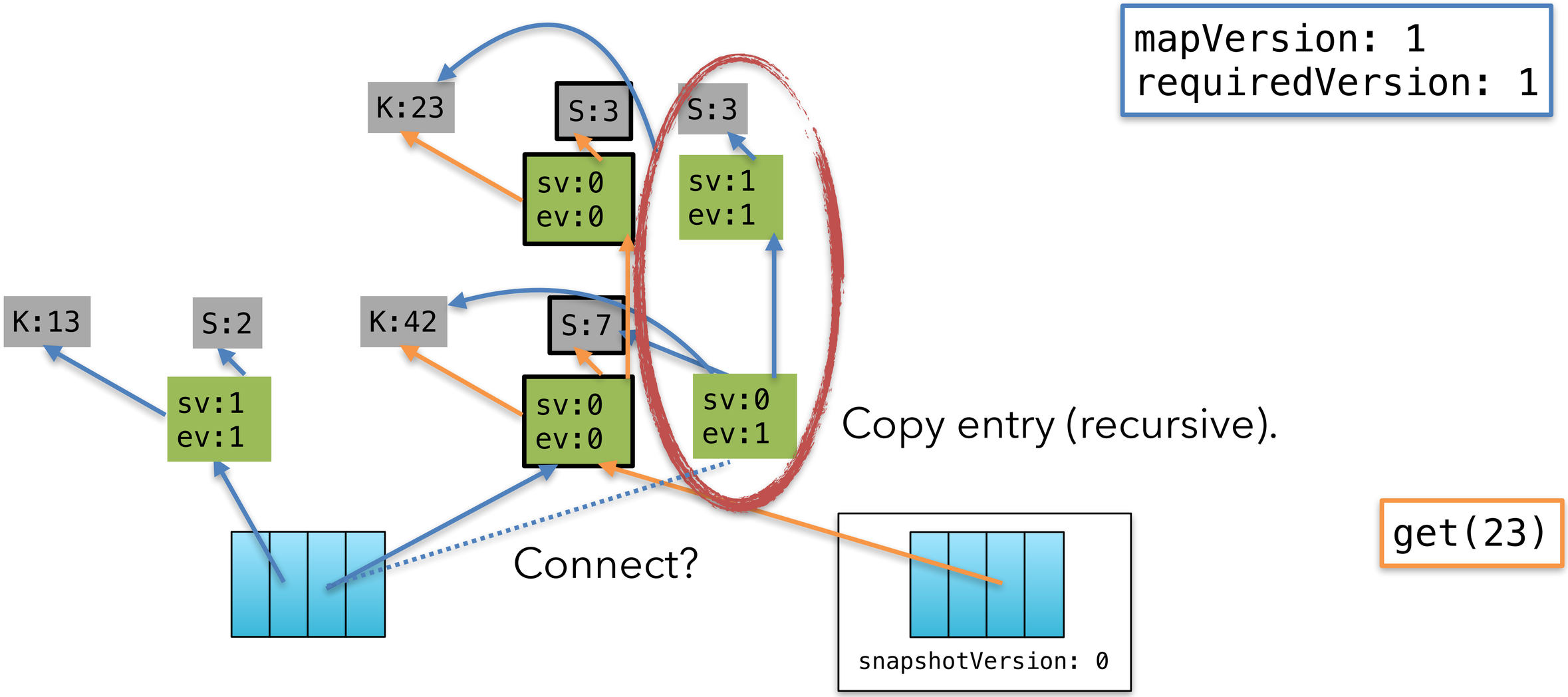
Copy-on-Write Hash Map - Example



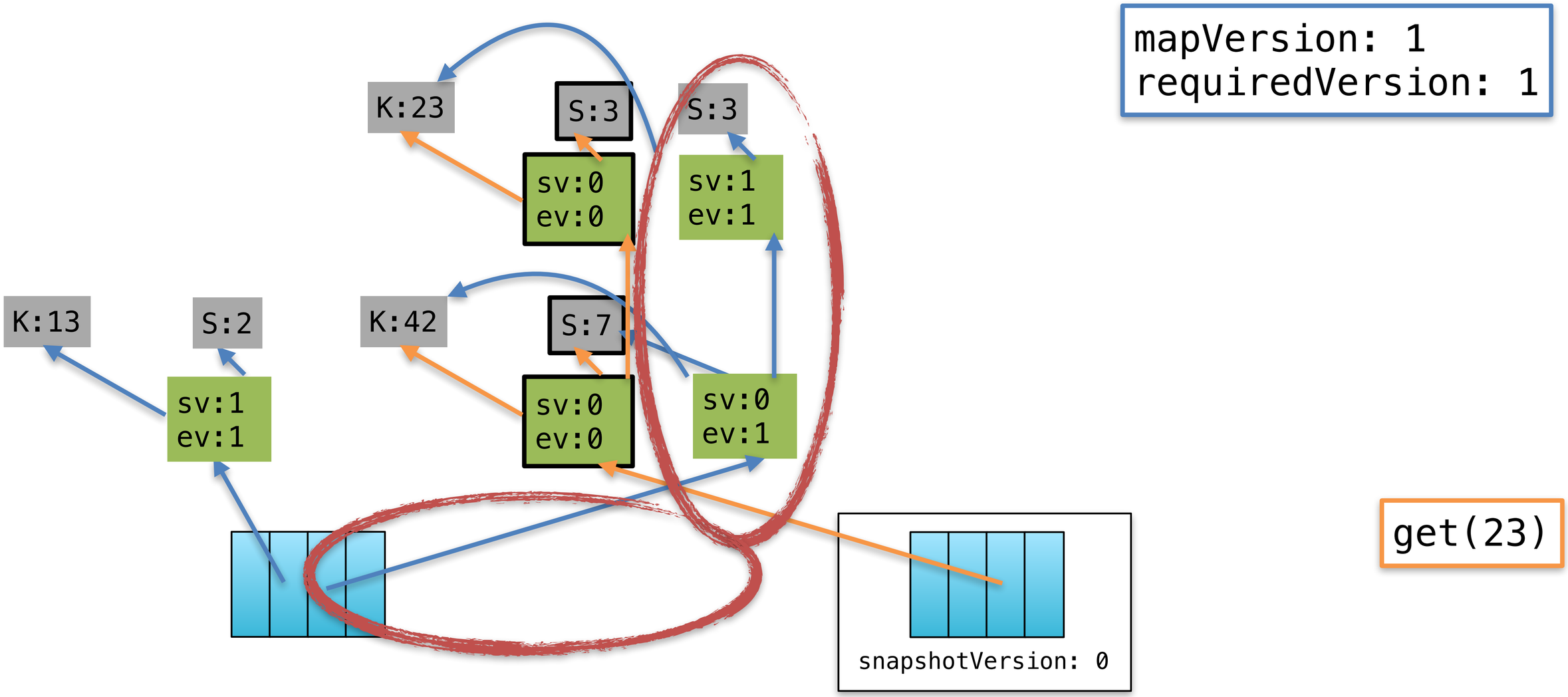
Copy-on-Write Hash Map - Example



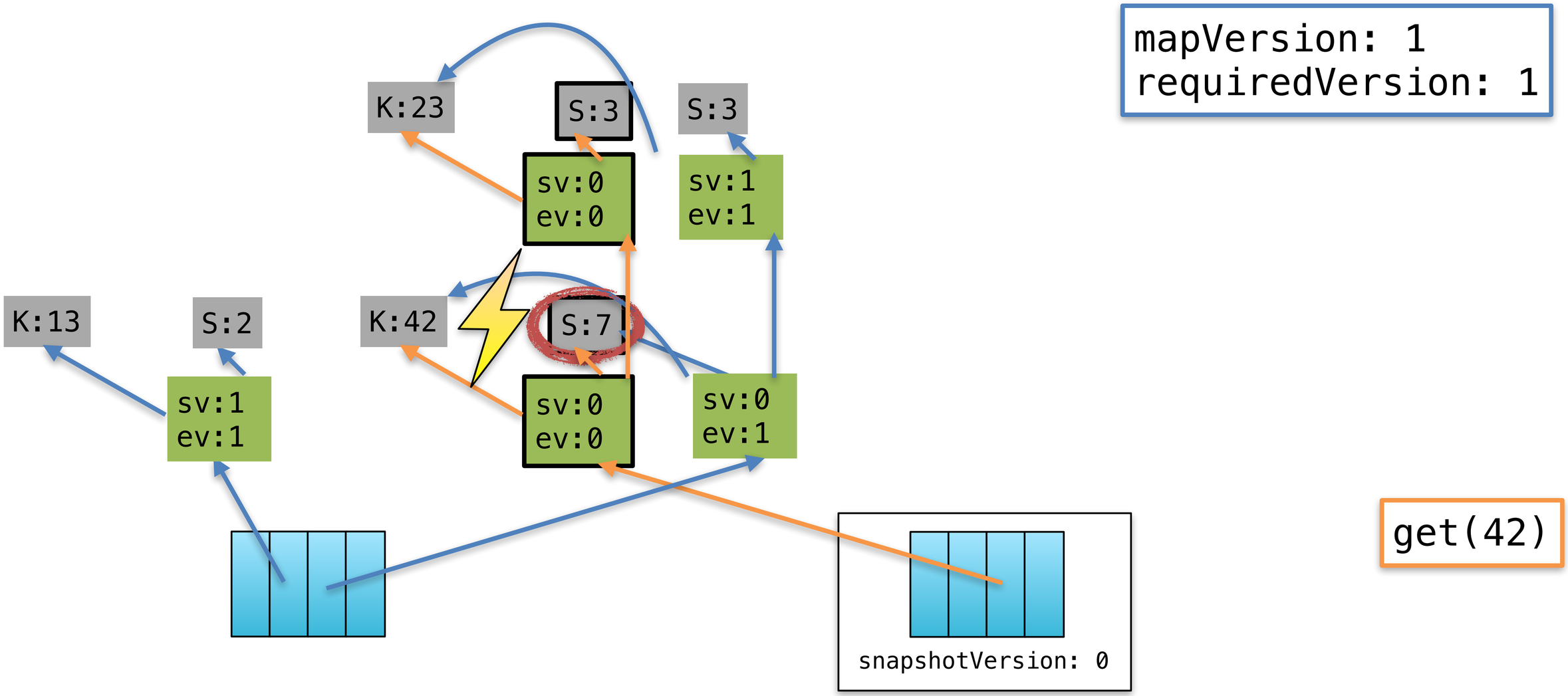
Copy-on-Write Hash Map - Example



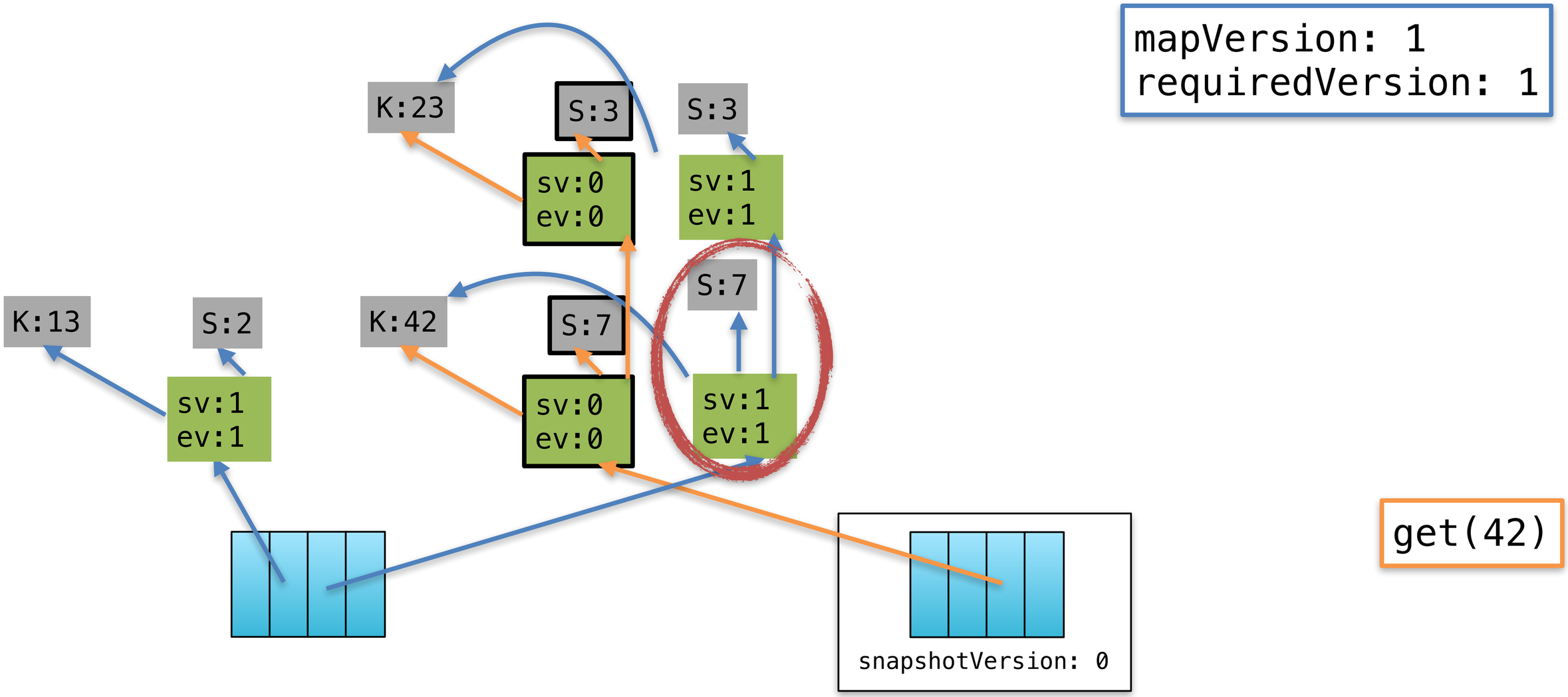
Copy-on-Write Hash Map - Example



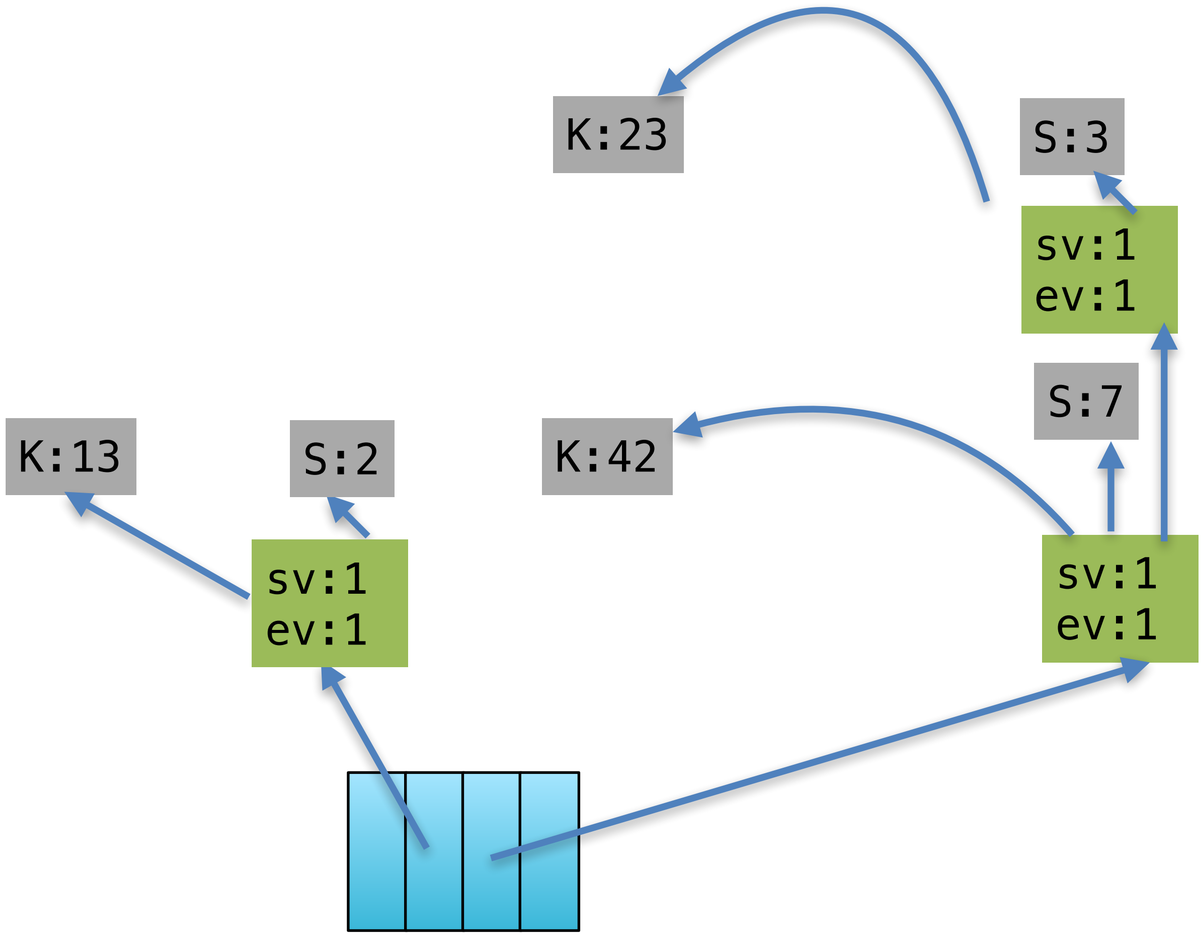
Copy-on-Write Hash Map - Example



Copy-on-Write Hash Map - Example



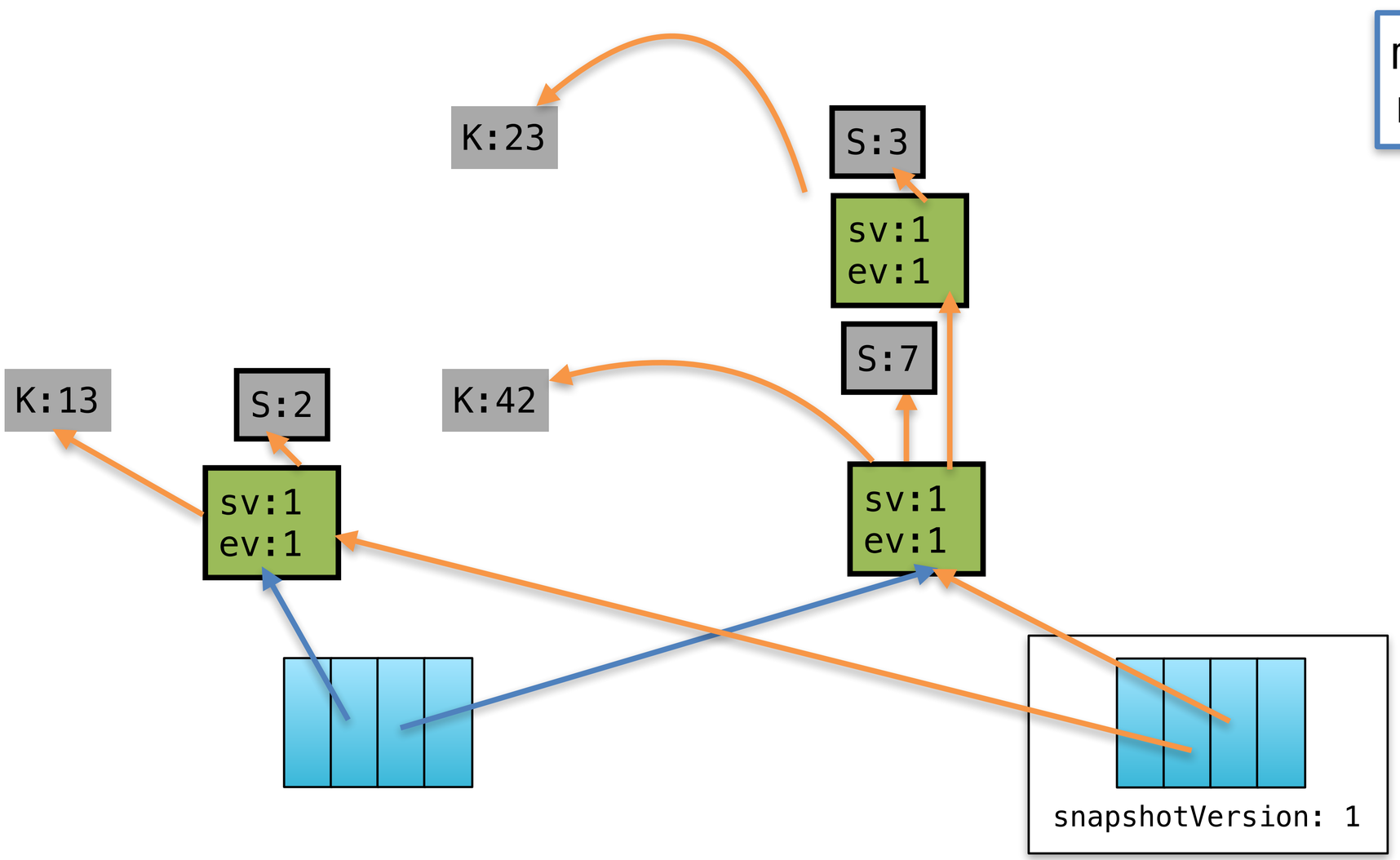
Copy-on-Write Hash Map - Example




```
mapVersion: 1
requiredVersion: 0
```

```
release(0)
```

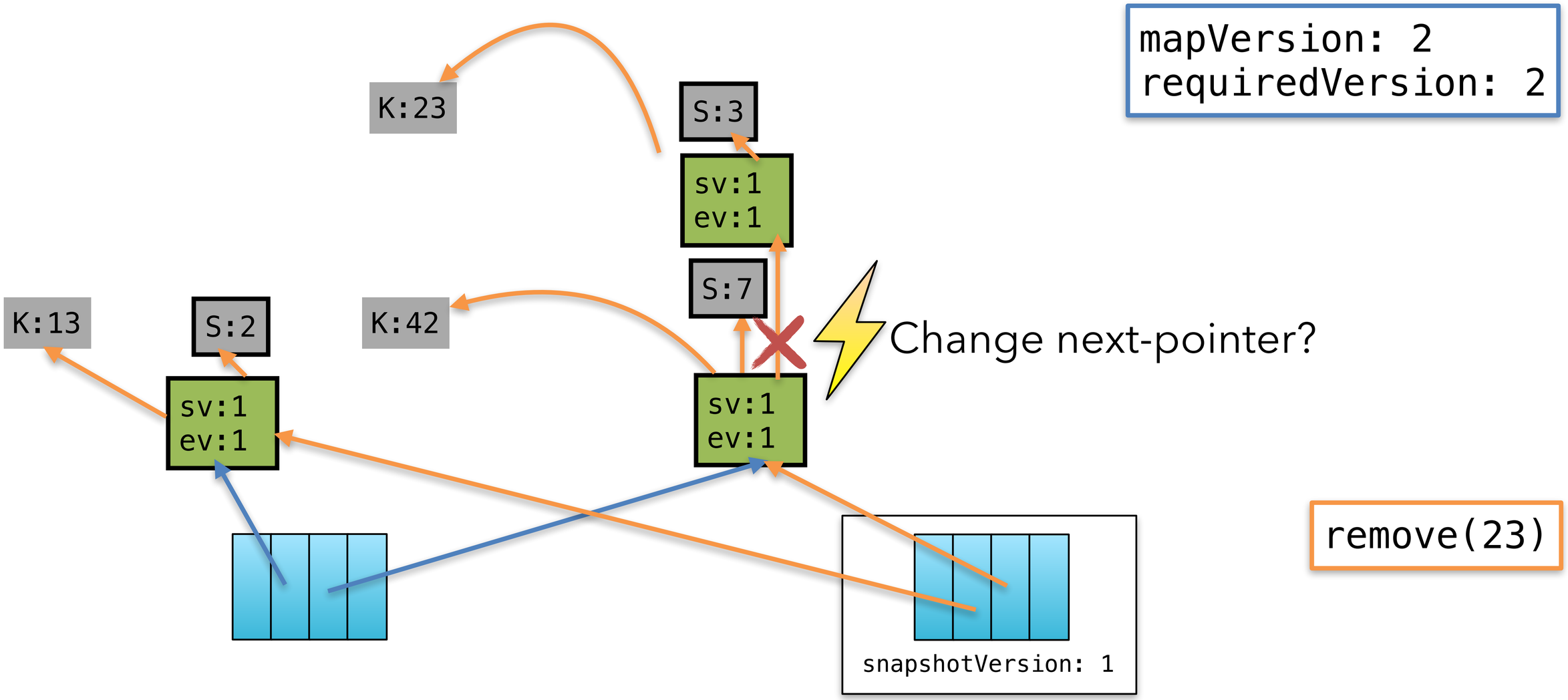
Copy-on-Write Hash Map - Example



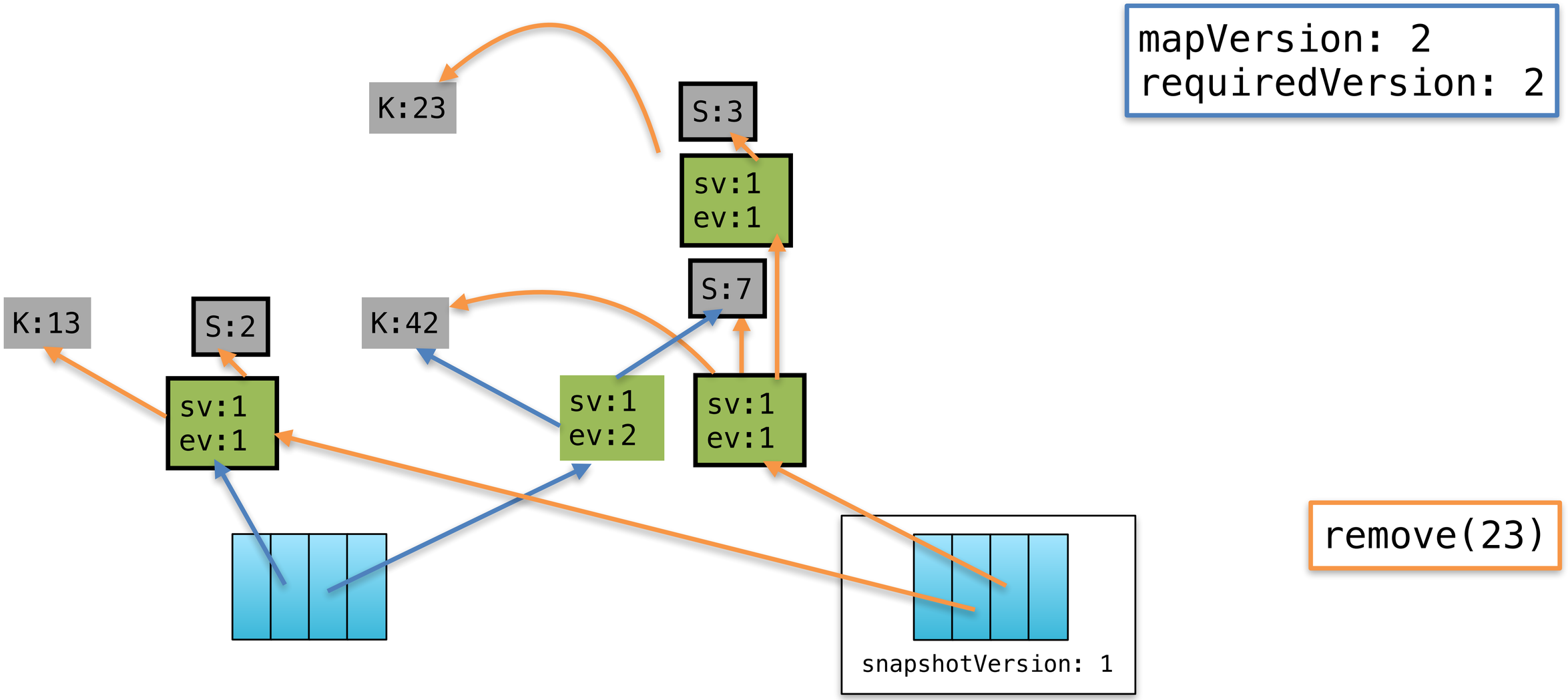

```
mapVersion: 2
requiredVersion: 2
```

```
snapshot() -> 1
```

Copy-on-Write Hash Map - Example

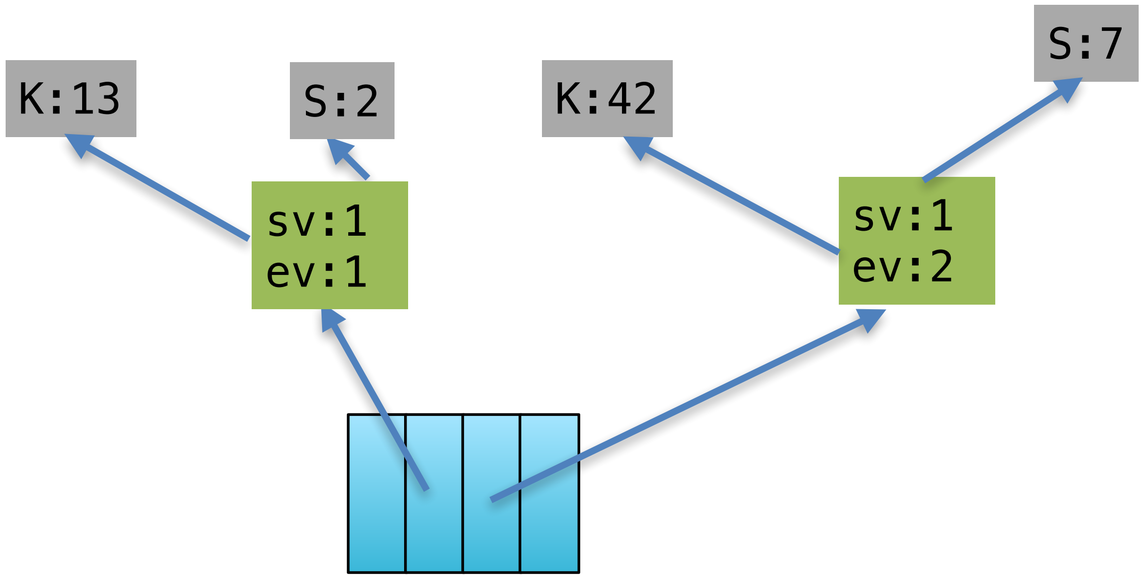


Copy-on-Write Hash Map - Example



Copy-on-Write Hash Map - Example

```
mapVersion: 2
requiredVersion: 0
```




```
release(1)
```