rhashtable

Concurrent, lockless hashtable with deferred resizing

One hashtable to rule them all

Lesson Learned

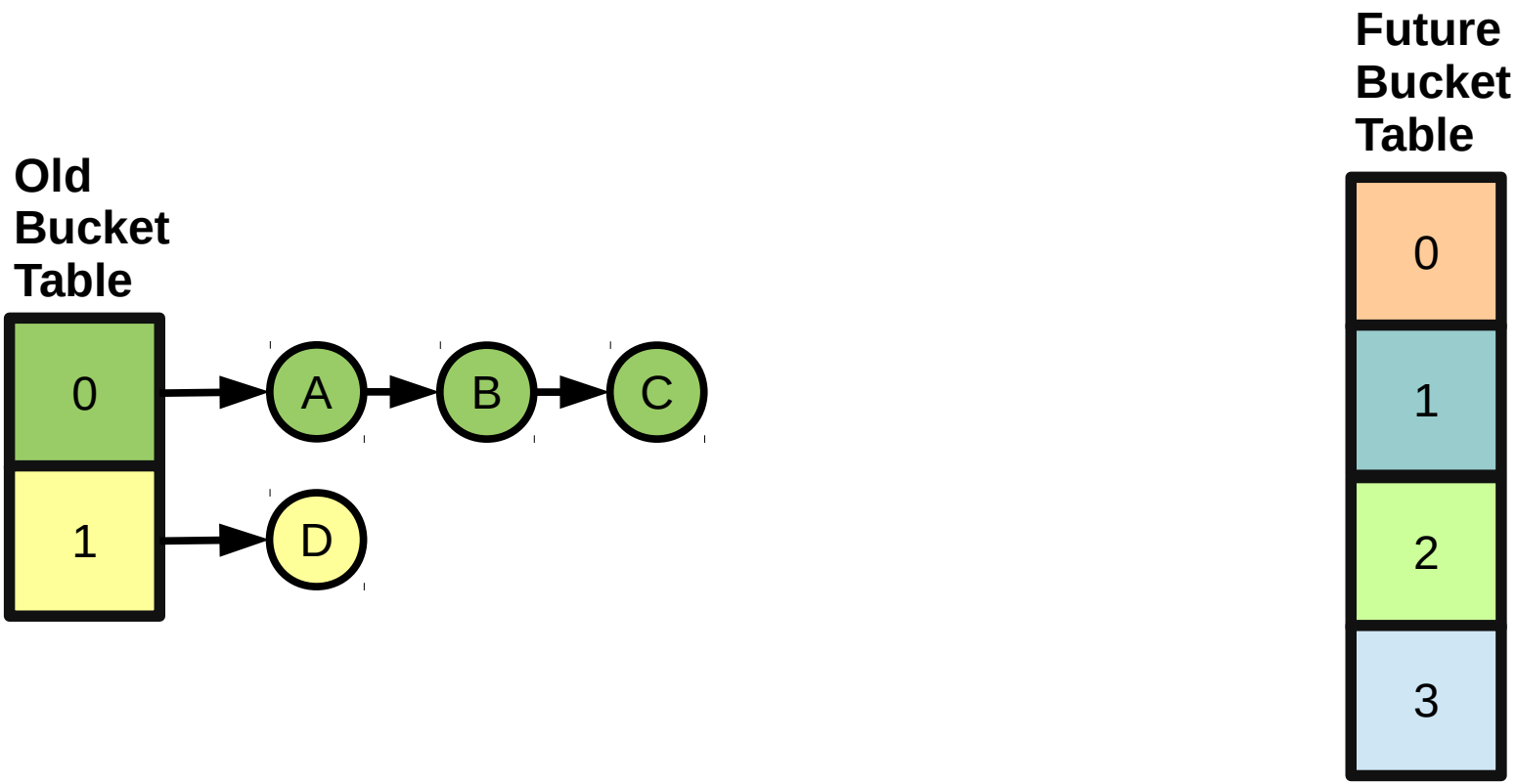
RCU is not always pretty

Characteristics

- Lockless lookup and traversal
- Parallel insertion and removal in non-conflicting buckets
- O(1) insert
- O(n) removal
- Deferred resizing with parallel lookups, insertion and removals
- Limited traversal order guarantee

Walkthrough: Growing

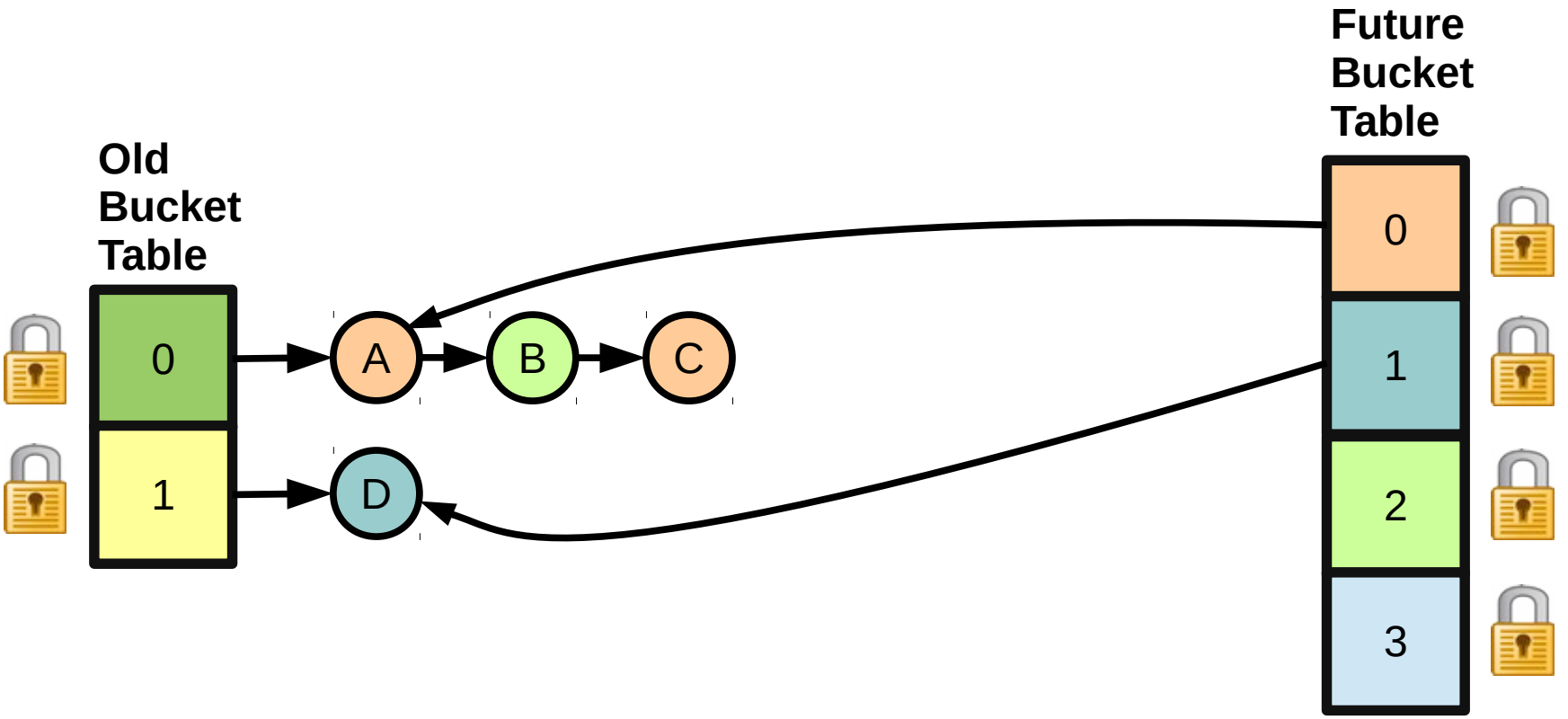
Growing



1. Allocate future table of 2*size
2. Announce future table
3. rcu_synchronize()

Growing

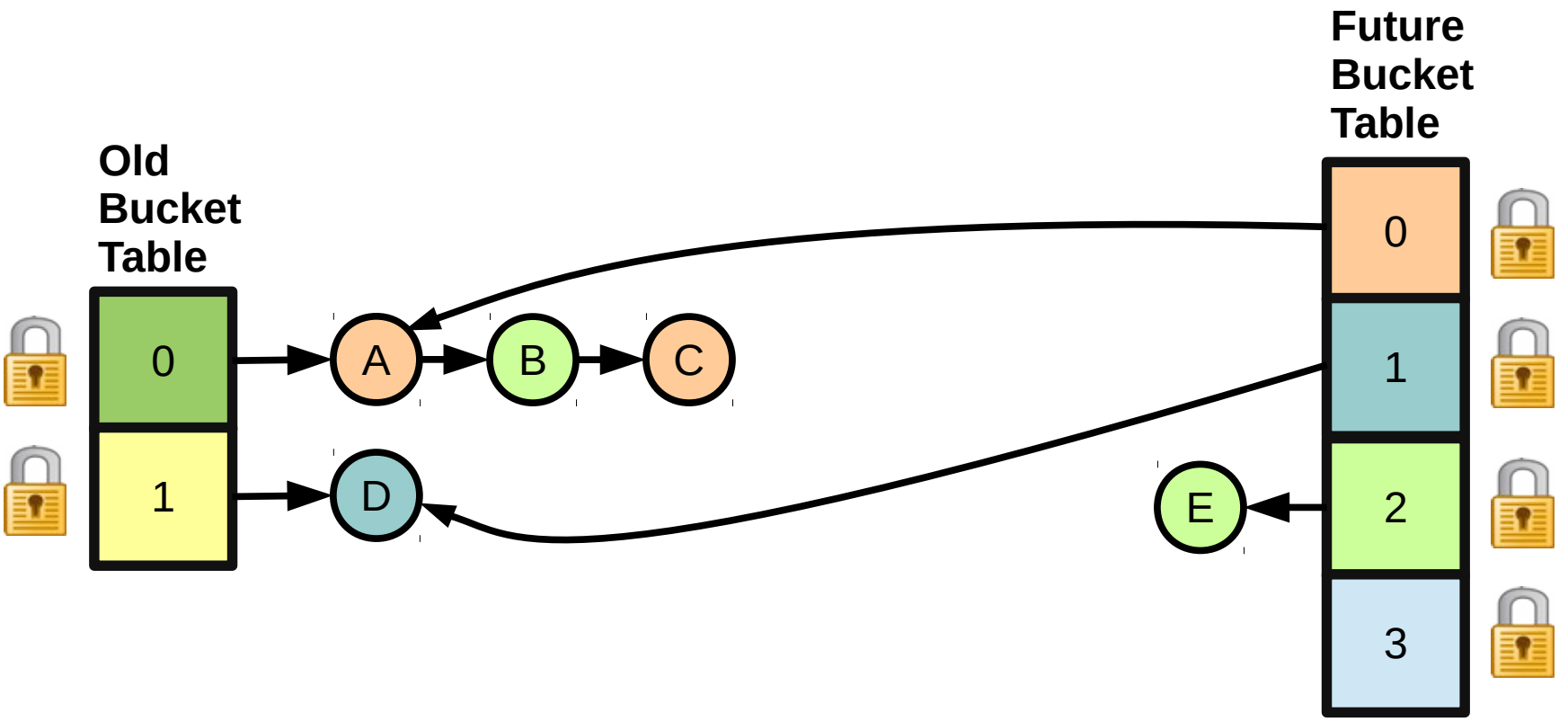
Recolored entries based on new hash function



1. Link old table to new table bucket by bucket

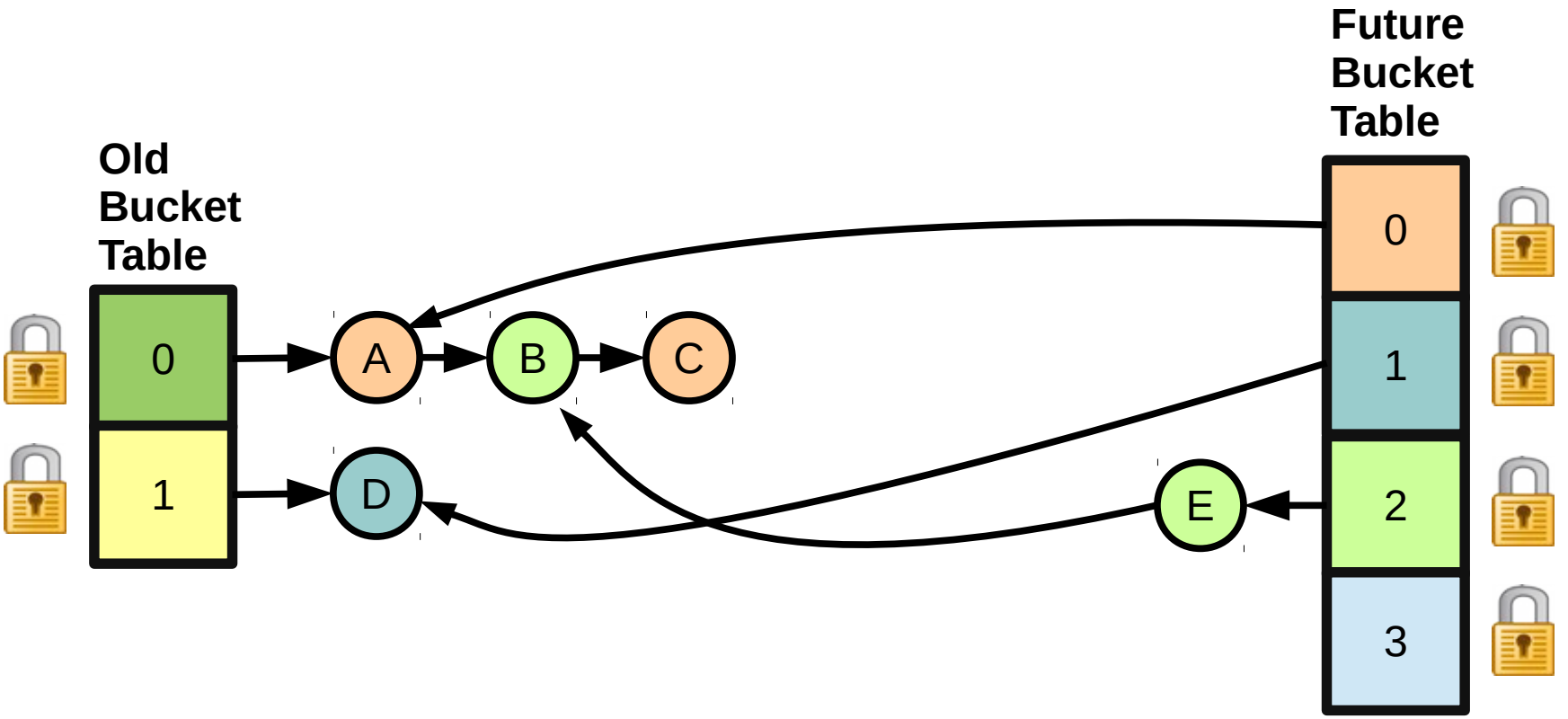
A wild insertion appears!

Growing



Growing

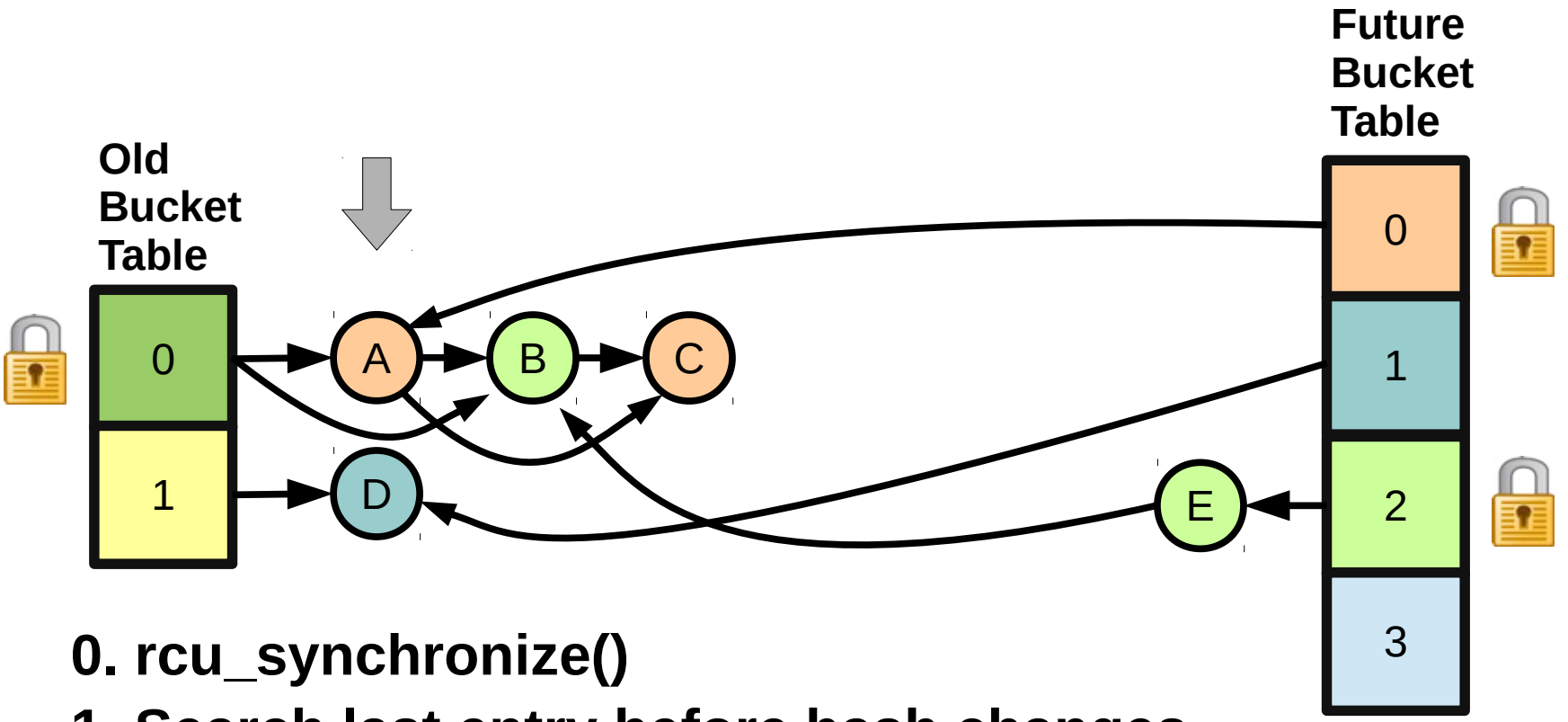
Linking continues....



All entries can now be found through future table but we traverse unnecessary entries.

Growing

Next: Optimize the chains by unzipping them.

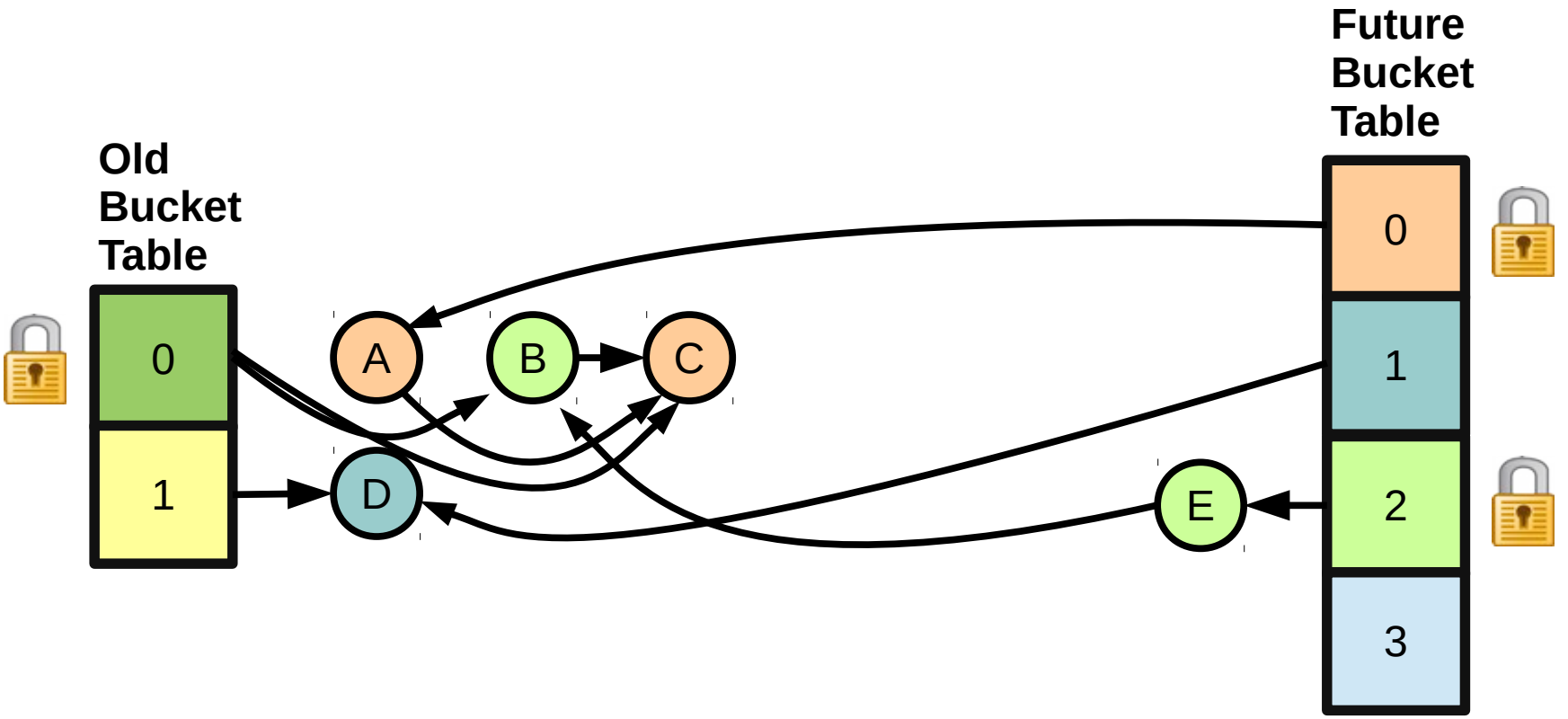


0. rcu_synchronize()
1. Search last entry before hash changes
2. Search for next hash match and link it
3. Link old bucket to where hash first changes
4. Repeat

A wild removal appears!

Growing

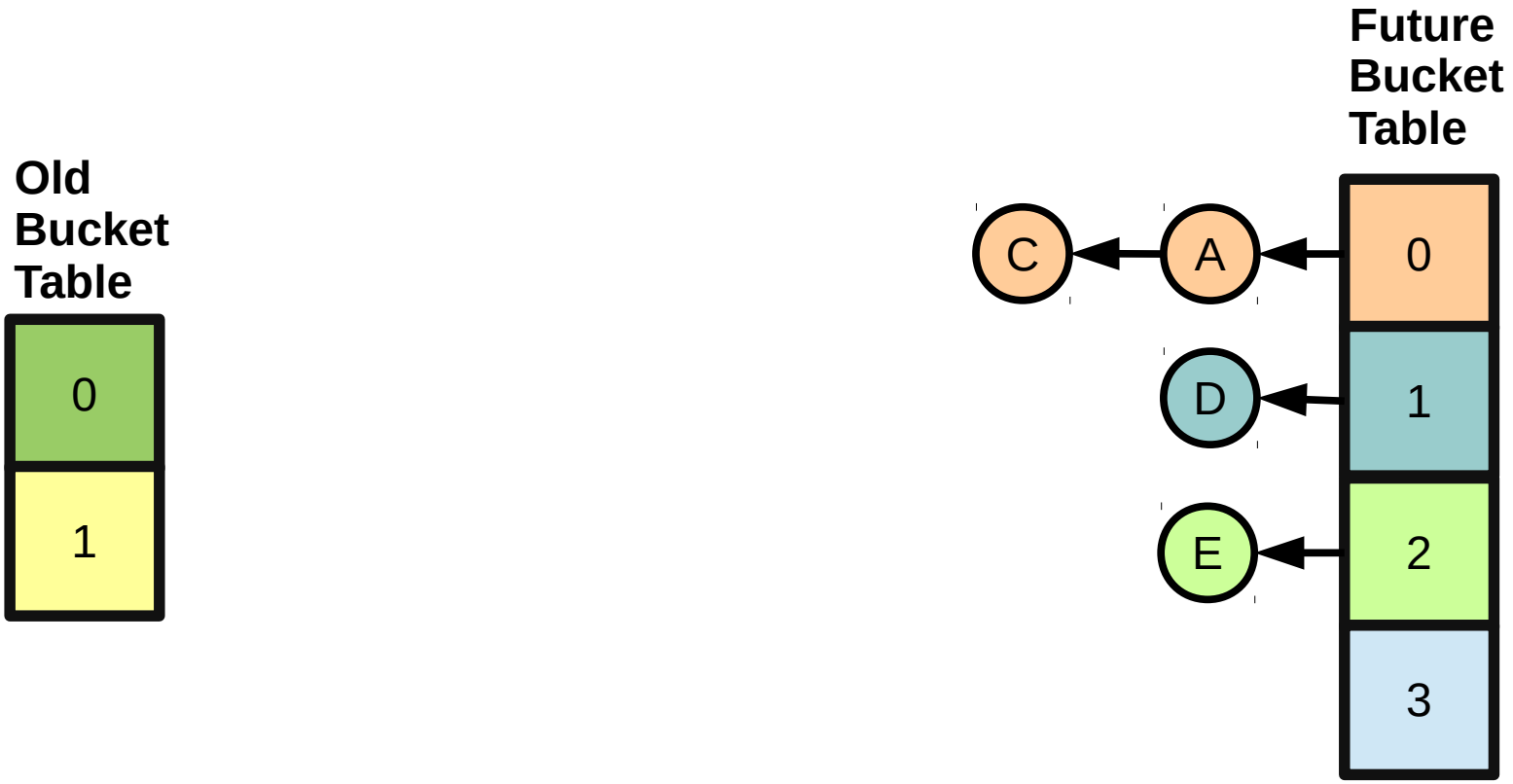
Removal of B



1. Find removal candidate through future table
2. Find removal candidate through old table

Growing

Unzip iterations continues until old bucket is empty



Final Step: Free old bucket table

Converted so far

- Netlink sockets table
- nft_hash

Candidates?

- Obvious:
  - TCP hash tables
- Others?