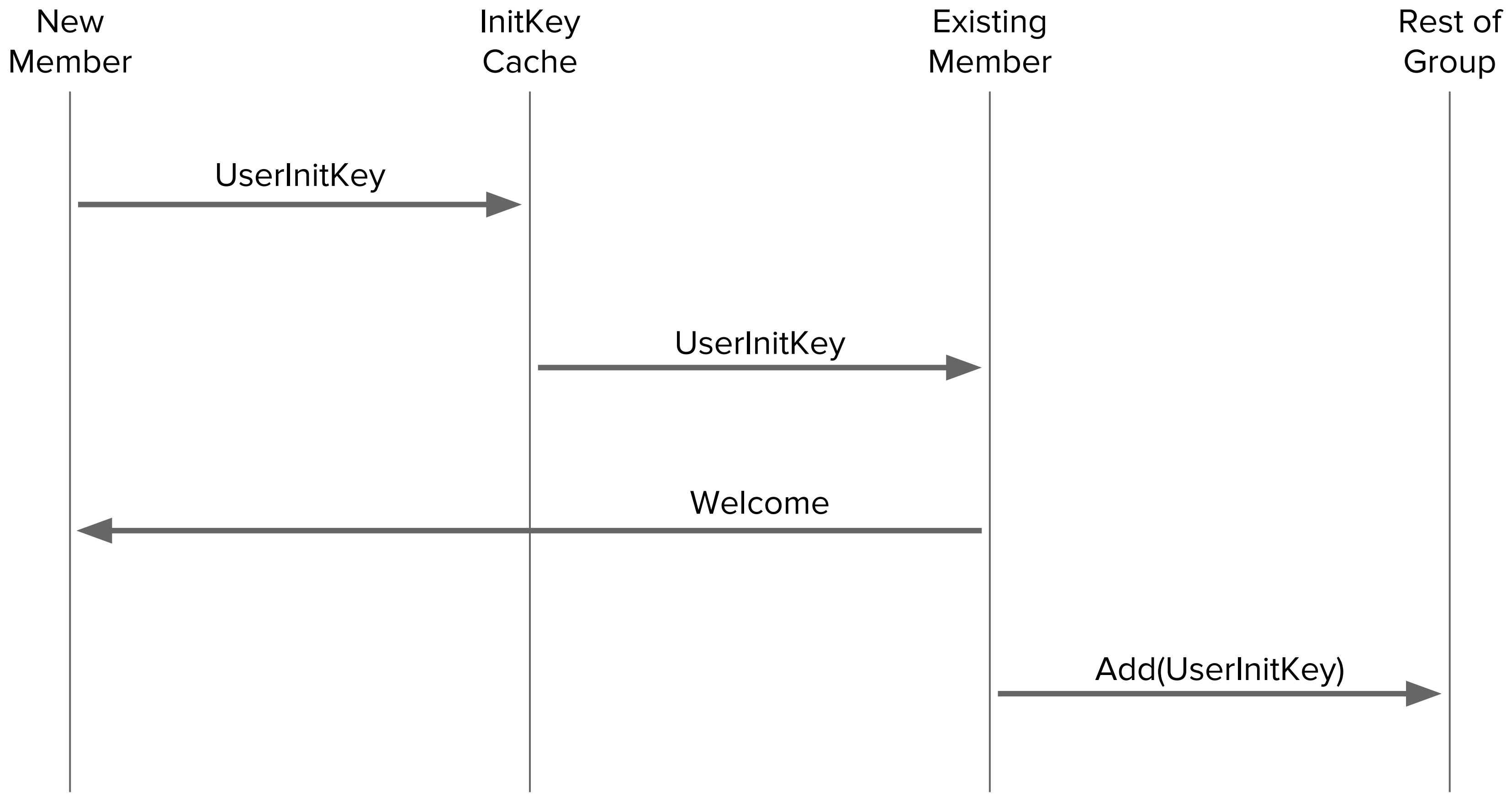
Encryption of Welcome Messages


New Member
InitKey Cache
Existing Member
Rest of Group
UserInitKey
UserInitKey
Welcome
Add(UserInitKey)

```
                                  struct {
                                    opaque group_id<0..255>;
                                    uint32 epoch;
                                    optional<Credential> roster<1..2^32-1>;
                                    optional<PublicKey> tree<1..2^32-1>;
                                    opaque transcript_hash<0..255>;
                                    opaque init_secret<0..255>;
                                  } WelcomeInfo;

struct {                          struct {
    opaque user_init_key_id<0..255>; ──────▶ opaque user_init_key_id<0..255>;
    CipherSuite cipher_suites<0..255>; ────▶ CipherSuite cipher_suite;
    DHPublicKey init_keys<1..2^16-1>; ─────▶ ECIESCiphertext encrypted_welcome_info;
    Credential credential;        } Welcome;
    opaque signature<0..2^16-1>;
} UserInitKey;
```

Garbage Collection

Trees get Ragged

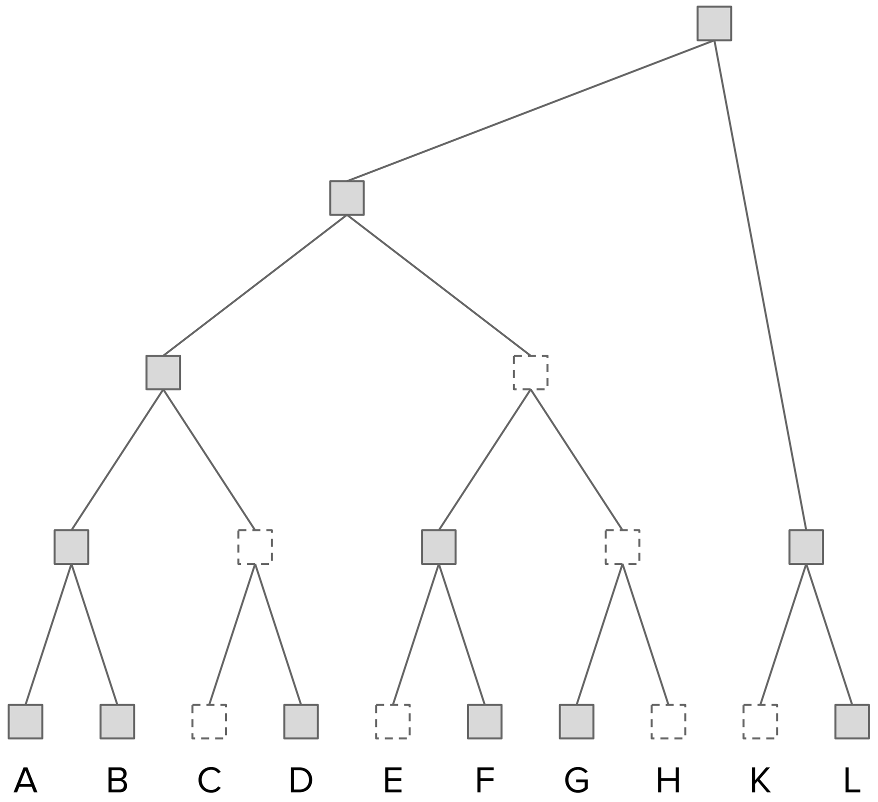
Suppose we start with a full tree...

C and E are removed

F updates

H and K are removed

L is added

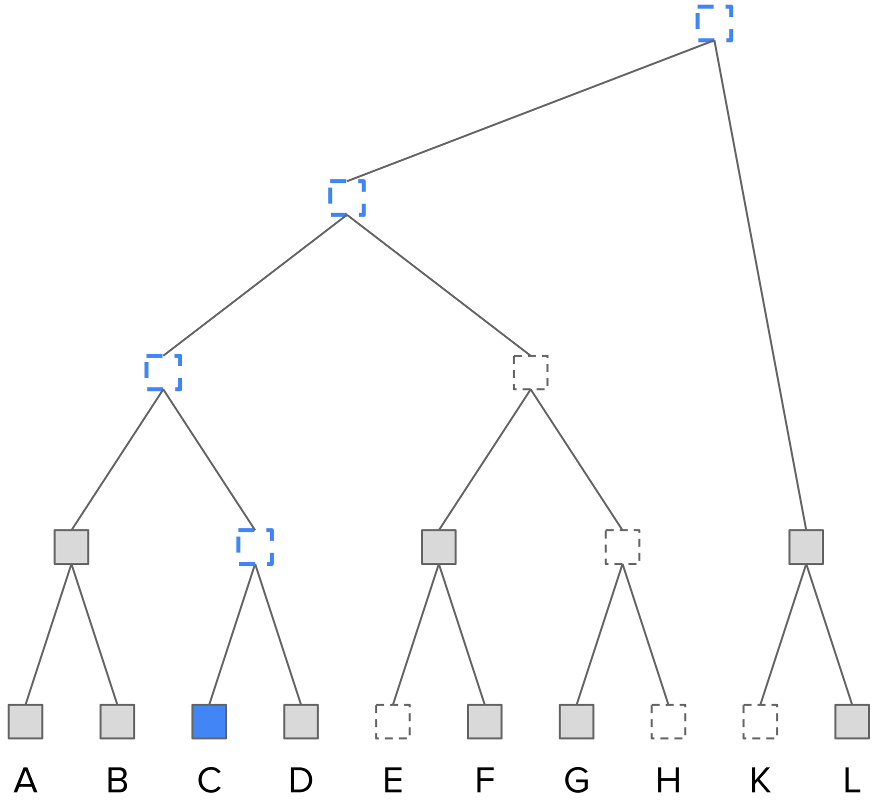


Add-in-Place

```
struct {
  uint32 index;
  UserInitKey init_key;
} Add;
```

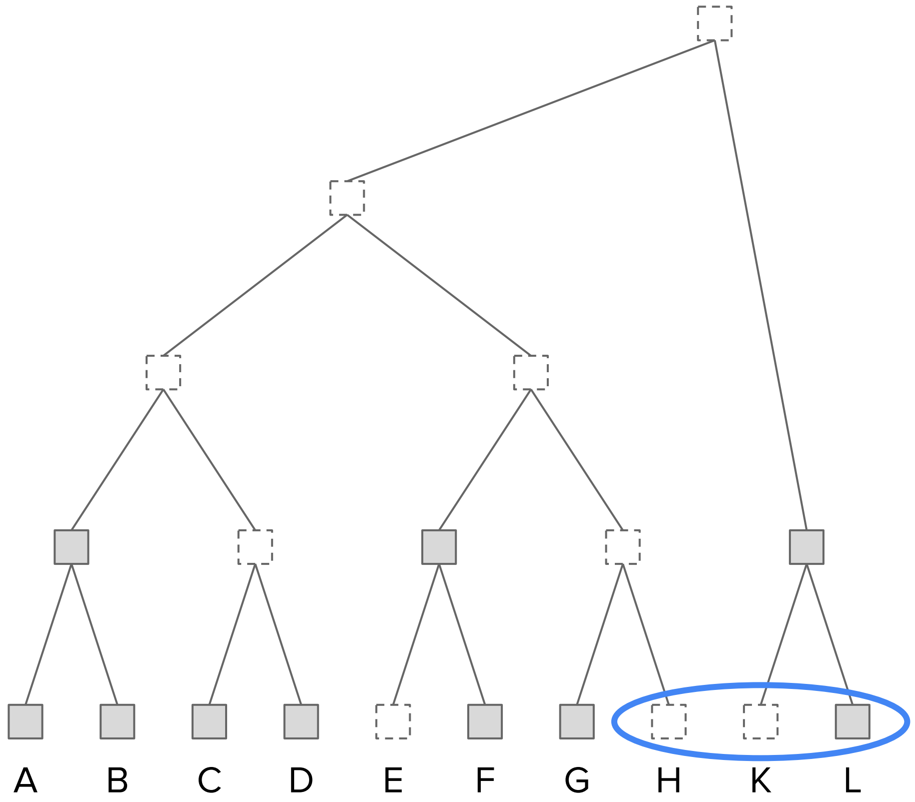
Reclaim a leaf

Blank out its direct path



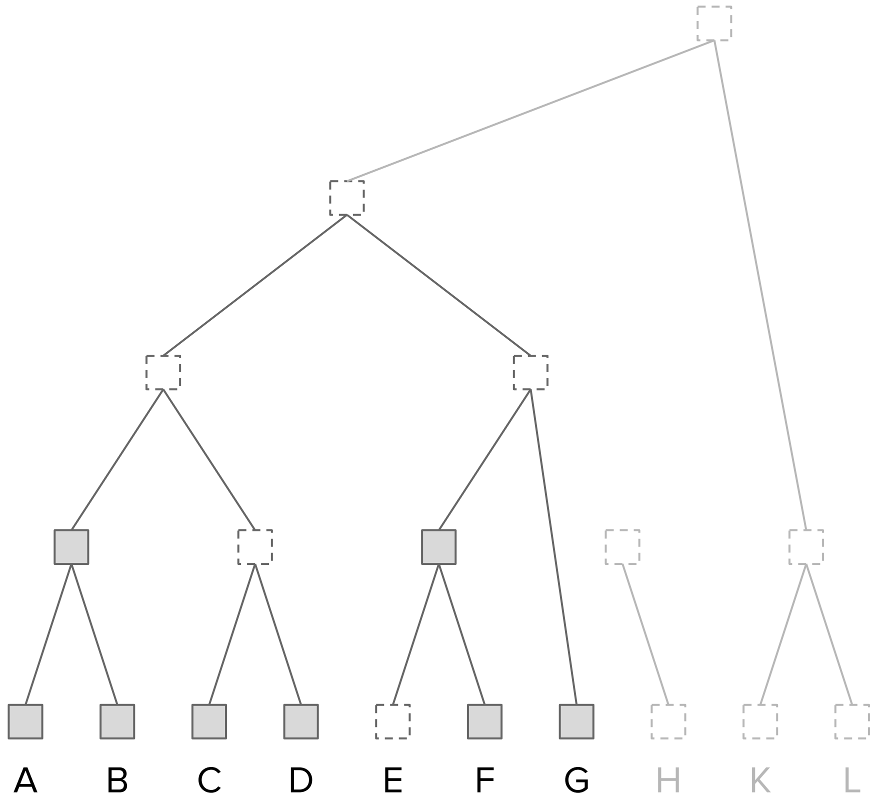
Cleanup-on-Remove

When you remove the right-most node, also remove any blanks

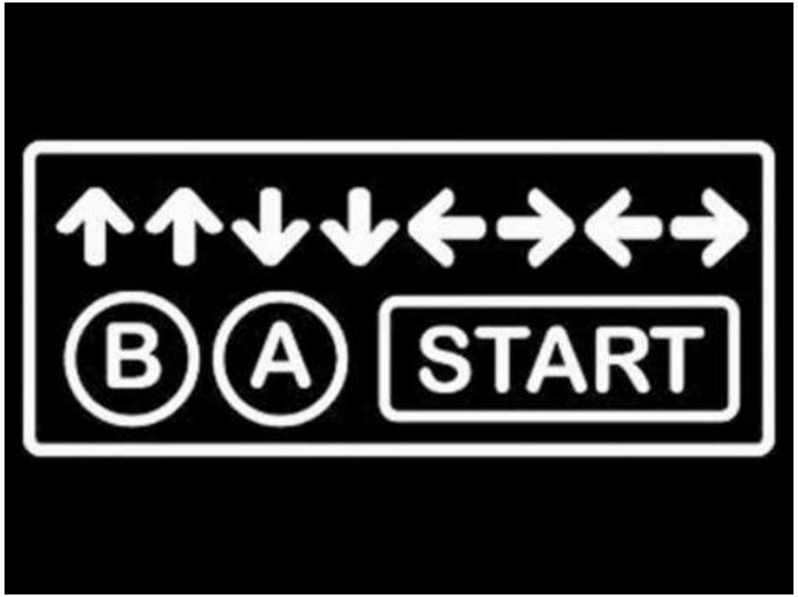


Cleanup-on-Remove

When you remove the right-most node, also remove any blanks



Combo moves



Resync = Remove + Add-in-place

Move = Remove + Add-in-place + Update

```
struct {
    uint32 prior_epoch;
    GroupOperation operation;
    uint32 signer_index;
    opaque signature<1..2^16-1>;
    opaque confirmation<1..2^8-1>;
} Handshake;
```

Vector?

Efficiency

Two problems

Size of State

Right now: Every member caches the whole roster and whole tree. Welcome and GroupState structs carry these objects by value

Approach: Cache state on the server

- Commitment in Welcome and GroupState
- Get actual objects from server …
- … and maybe update from Handshake

Warm-up Time

Right now: On group creation, operations start linear, converge to log as members update

Approach: Defer inefficiency to remove time

- Creator populates some nodes
- … which are thus double-joined
- Other members track double-join
- … and resolve as members update

https://ipv.sx/treekem/blanking/