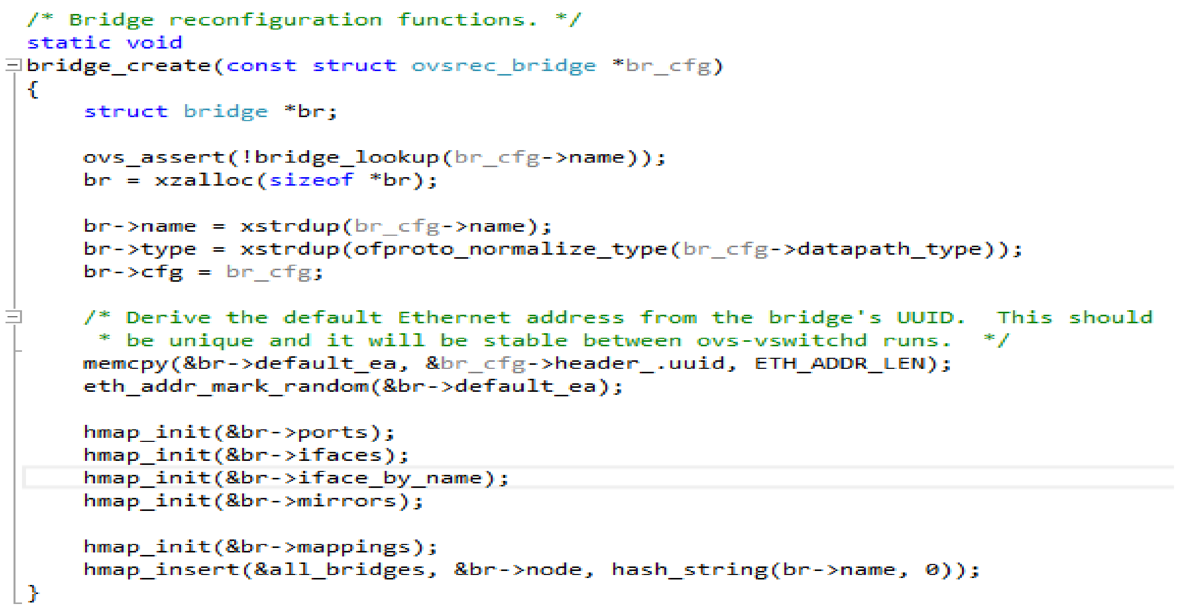
Hashmap implementation

Friday, January 13, 2017 6:28 PM

D:\Original6\monitoringagent\ovs\vswitchd\bridge.c

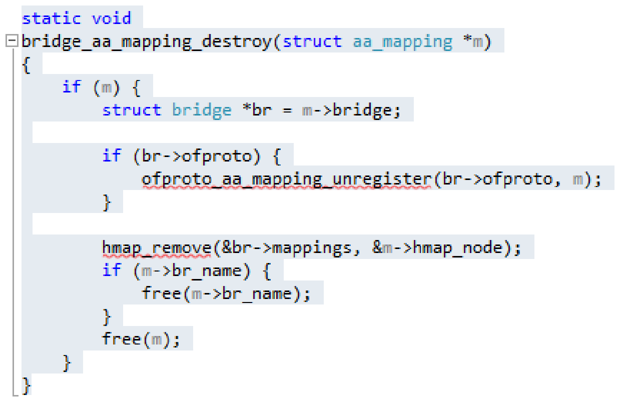


```c
/* Bridge reconfiguration functions. */
static void
bridge_create(const struct ovsrec_bridge *br_cfg)
{
    struct bridge *br;

    ovs_assert(!bridge_lookup(br_cfg->name));
    br = xzalloc(sizeof *br);

    br->name = xstrdup(br_cfg->name);
    br->type = xstrdup(ofproto_normalize_type(br_cfg->datapath_type));
    br->cfg = br_cfg;

    /* Derive the default Ethernet address from the bridge's UUID.  This should
     * be unique and it will be stable between ovs-vswitchd runs.  */
    memcpy(&br->default_ea, &br_cfg->header_.uuid, ETH_ADDR_LEN);
    eth_addr_mark_random(&br->default_ea);

    hmap_init(&br->ports);
    hmap_init(&br->ifaces);
    hmap_init(&br->iface_by_name);
    hmap_init(&br->mirrors);

    hmap_init(&br->mappings);
    hmap_insert(&all_bridges, &br->node, hash_string(br->name, 0));
}
```

Bridge has hash-maps in it.

Making code changes on these lines : D:\Original6\monitoringagent\ovs\vswitchd\bridge.c
/* All bridges, indexed by name. */
static struct hmap all_bridges = HMAP_INITIALIZER(&all_bridges);

Use this code to delete the monitoring agent Node and the corresponding HashMap node.



```c
static void
bridge_aa_mapping_destroy(struct aa_mapping *m)
{
    if (m) {
        struct bridge *br = m->bridge;

        if (br->ofproto) {
            ofproto_aa_mapping_unregister(br->ofproto, m);
        }

        hmap_remove(&br->mappings, &m->hmap_node);
        if (m->br_name) {
            free(m->br_name);
        }
        free(m);
    }
}
```