```
#include <stdio.h>
#include <stdlib.h>

// Structure for a node in the broadcast tree
typedef struct TreeNode {
    int host_id;
    struct TreeNode* children[10]; // Assuming a maximum of 10 children for simplicity
    int child_count;
} TreeNode;

// Function to create a new tree node (representing a host)
TreeNode* createNode(int host_id) {
    TreeNode* newNode = (TreeNode*)malloc(sizeof(TreeNode));
    if (!newNode) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newNode->host_id = host_id;
    newNode->child_count = 0;
    for (int i = 0; i < 10; i++) {
        newNode->children[i] = NULL;
    }
    return newNode;
}
```

```
// Function to add a child node to a parent node
void addChild(TreeNode* parent, TreeNode* child) {
    if (parent->child_count < 10) {
        parent->children[parent->child_count++] = child;
    } else {
        printf("Warning: Maximum children reached for host %d\n", parent->host_id);
    }
}

// Function to print the broadcast tree (depth-first traversal)
void printTree(TreeNode* root, int level) {
    if (root == NULL) {
        return;
    }
    for (int i = 0; i < level; i++) {
        printf("  "); // Indentation to show levels
    }
    printf("Host %d\n", root->host_id);
    for (int i = 0; i < root->child_count; i++) {
        printTree(root->children[i], level + 1);
    }
}

// Function to free the memory allocated for the tree
void freeTree(TreeNode* root) {
```

```
    if (root == NULL) {
        return;
    }
    for (int i = 0; i < root->child_count; i++) {
        freeTree(root->children[i]);
    }
    free(root);
}

int main() {
    // Create the root of the broadcast tree (could be a central server or router)
    TreeNode* root = createNode(0);

    // Add some child nodes representing hosts in the subnet
    addChild(root, createNode(1));
    addChild(root, createNode(2));

    // Add more levels to the tree
    addChild(root->children[0], createNode(3));
    addChild(root->children[0], createNode(4));
    addChild(root->children[1], createNode(5));

    addChild(root->children[0]->children[0], createNode(6));

    printf("Broadcast Tree:\n");
```

```
    printTree(root, 0);

    // Remember to free the allocated memory
    freeTree(root);

    return 0;
}
```

# output

```
Broadcast Tree:
Host 0
  Host 1
    Host 3
      Host 6
    Host 4
  Host 2
    Host 5
```