using UnityEngine;

using UnityEngine.Events;

using System.Collections.Generic;

using System.Linq;

public class DHCPServer : MonoBehaviour

{

[SerializeField] private string \_subnet = "192.168.1.0/24"; //диапозон айпишников

[SerializeField] private float \_leaseTime = 86400; //время аренды айпишника в секундах

[SerializeField] private bool \_enableLogging = true; // флаг для вкл/выкл логов

private HashSet<string> \_availableIPs = new HashSet<string>(); //множество свободных айпи

private Dictionary<string, DHCPLease> \_leasedIPs = new Dictionary<string, DHCPLease>(); //словарь занятых айпи (ключ - айпи, значение - данные аренды)

private List<string> \_reservedIPs = new List<string>(); //список зарезервированных айпи

public UnityEvent<string> OnIPLeased; //событие которое вызывается при выдаче айпишника

public UnityEvent<string> OnIPReleased; //событие которок вызывается при освобождении айпишника

private class DHCPLease

{

public string MacAddress; //MAC-адрес устройства

public float ExpiryTime; //время окончания аренды (в сек с момента запуска)

}

void Start()

{

\_availableIPs = new HashSet<string>();

InitPool(); // Явная инициализация пула

Debug.Log($"Пул инициализирован. Доступно IP: {\_availableIPs.Count}");

}

//инициализация пула ip

public void InitPool()

{

\_availableIPs.Clear(); //очистка списка свободных ip

\_leasedIPs.Clear(); //Очистка списка занятых Ip

string baseIP = \_subnet.Split('/')[0]; // получение базового списка ip (до /)

string[] octets = baseIP.Split('.'); // разбиение на октеты (192, 168, 1, 0)

for (int i = 1; i <= 254; i++)

{

string ip = $"{octets[0]}.{octets[1]}.{octets[2]}.{i}";

\_availableIPs.Add(ip); //добавление в пул свободных адресов

}

if (\_enableLogging) // если логировние включено

Debug.Log($"Пул IP инициализирован. Доступно: {\_availableIPs.Count} адресов");

}

public string RequestIP(string macAddress)

{

if (!ValidateMac(macAddress))

{

Debug.LogError($"Некорректный MAC: {macAddress}");

return null;

}

if (\_availableIPs.Count == 0)

{

CleanExpiredLeases();

if (\_availableIPs.Count == 0)

{

Debug.LogError("DHCP Pool is empty!");

return null;

}

}

// Используем First() с LINQ

string ip = \_availableIPs.First();

\_leasedIPs[ip] = new DHCPLease

{

MacAddress = macAddress,

ExpiryTime = Time.time + \_leaseTime

};

\_availableIPs.Remove(ip);

if (\_enableLogging)

Debug.Log($"Assigned IP {ip} to MAC {macAddress}");

OnIPLeased?.Invoke(ip);

return ip;

}

private void CleanExpiredLeases()

{

var leasesToCheck = new List<string>(\_leasedIPs.Keys);

foreach (var ip in leasesToCheck)

{

if (!string.IsNullOrEmpty(ip) && \_leasedIPs.TryGetValue(ip, out var lease))

{

if (lease.ExpiryTime <= Time.time)

{

\_availableIPs.Add(ip);

\_leasedIPs.Remove(ip);

}

}

}

}

public bool ReleaseIP(string ip)

{

// Добавьте проверку на null в начале метода

if (string.IsNullOrEmpty(ip))

{

Debug.LogError("Передан пустой IP-адрес!");

return false;

}

if (!IsValidIP(ip))

{

Debug.LogError($"Неверный формат IP: {ip}");

return false;

}

\_availableIPs.Add(ip);

\_leasedIPs.Remove(ip);

if (\_enableLogging)

Debug.Log($"Освобожден IP: {ip}");

OnIPReleased?.Invoke(ip);

return true;

}

private bool IsValidIP(string ip)

{

return System.Text.RegularExpressions.Regex.IsMatch(ip,

@"^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$");

}

public bool RenewLease(string ip, string macAddress)

{

if (!\_leasedIPs.TryGetValue(ip, out DHCPLease lease))

{

Debug.LogWarning($"IP {ip} is not leased!"); // Выводим предупреждение

return false; // Возвращаем false

}

if (lease.MacAddress != macAddress) // Если MAC не совпадает

{

Debug.LogWarning($"MAC {macAddress} doesen't match leased IP {ip}");

return false;

}

lease.ExpiryTime = Time.time + \_leaseTime;

if (\_enableLogging)

Debug.Log($"Lease renewed for IP {ip}");

return true;

}

private bool ValidateMac(string mac)

{

return System.Text.RegularExpressions.Regex.IsMatch(

mac,

"^([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa-f]{2})$"

);

}

public List<string> GetLeasedIPs()

{

return new List<string>(\_leasedIPs.Keys);

}

}