ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КУРСОВОЙ ПРОЕКТ

по дисциплине “Сетевое программирование”

на тему

Разработка сетевого приложение. Протокол OSPF.

Выполнил студент Рудцких Владислав Евгеньевич
Ф.И.О.

Группы ИВ-222

Работу принял _____ Г.Н. Третьяков
подпись

Защищена _____ Оценка _____

Новосибирск 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

OSPF (англ. Open Shortest Path First) — протокол динамической маршрутизации, основанный на технологии отслеживания состояния канала (link-state technology) и использующий для нахождения кратчайшего пути алгоритм Дейкстры.

Протокол OSPF, наряду с IS-IS, принадлежит к классу протоколов маршрутизации Link State. Принципы этого класса заключается в том, что в памяти маршрутизатора помимо всех оптимальных маршрутов в удаленные сети должна быть полная карта сети, в том числе с действующими связями между другими маршрутизаторами. OSPF изначально создавался как открытый протокол, что сделало его самым распространенным среди протоколов маршрутизации. Его алгоритм позволяет достаточно легко выстраивать стек протоколов для OSPF.

OSPF имеет следующие преимущества:

1. Высокая скорость сходимости по сравнению с дистанционно-векторными протоколами маршрутизации;
2. Поддержка сетевых масок переменной длины (VLSM);
3. Оптимальное использование пропускной способности с построением дерева кратчайших путей.

ПРИНЦИП РАБОТЫ

Работа протокола OSPF строится по следующему алгоритму:

1. Маршрутизаторы производят обмен малыми пакетами HELLO.
2. После выполнения обмена между ними устанавливаются соседства. Каждый из маршрутизаторов добавляет в специальную локальную таблицу соседей.
3. Маршрутизаторы выполняют сбор состояний своих связей с соседями (линков). Линки включают id самого маршрутизатора и соседа, сеть и префикс, тип сети и метрику (стоимость линка). После сбора состояний маршрутизатор формирует пакет LSA (Link State Advertisement).
4. LSA рассылается каждому соседу, который передает пакет дальше по сети.
5. После получения пакета LSA каждый маршрутизатор добавляет содержащуюся в нем информацию в локальную таблицу LSDB (Link State Database).
6. В таблице LSDB накапливаются данные обо всех парах маршрутизаторов в пределах сети.
7. На основании накопленных данных выстраивается полная карта сети, которая включает все действующие маршрутизаторы и образованные между ними связи.
8. Используя карту, каждый маршрутизатор выполняет поиск самых коротких маршрутов во все сети и формирует из них таблицу маршрутизации.

Учитывая ресурсоемкий и сложный принцип работы OSPF, от каждого маршрутизатора требуется достаточно высокая производительность и значительный объем оперативной памяти.

Типы сетей, поддерживаемые протоколом OSPF:

1. Широковещательные сети со множественным доступом (Ethernet, Token Ring)
2. Point-to-point (T1, E1, коммутируемый доступ)
3. Не широковещательные сети со множественным доступом (NBMA) (Frame relay)

В случае разрыва связи с соседом у одного из маршрутизаторов, он отправляет по сети новые пакеты LCA, и повторяется вся процедура формирования таблицы маршрутизации. Чтобы исключить постоянный пересчет в крупных сетях с большим количеством маршрутизаторов, в них применяют разделение на отдельные зоны. В каждой из них выполняются автономные вычисления с передачей между зонами только итогового результата. В любой конфигурации OSPF должна присутствовать корневая зона с индексом 0. Малые сети обычно помещаются в ее пределах, а для больших — требуется формирование дополнительных зон.

Пакет OSPF помещается в пакет IP с мультикастовым адресом получателя. Отправителю же в нем соответствует адрес маршрутизатора. Пакет помещается в мультикастовый фрейм, например, в Ethernet. При формировании списков контроля доступа нужно учитывать, что OSPF инкапсулируется непосредственно в IP, а не в UDP или TCP.

Учитывая, что при построении карты сети маршрутизаторами в качестве узлов выступают другие маршрутизаторы, большое значение имеет наличие у каждого из них уникального имени. Для определения такого имени используется поле Router ID.

Идентификатор записывается в виде IP-адреса IPv4. Неважно, какое он примет значение, главное, чтобы он был уникальным в пределах этой сети. ID маршрутизатора в OSPF можно задать вручную. Если он не задан, то будет присвоен автоматически.

Принцип работы протокола OSPF предусматривает следующий алгоритм назначения ID маршрутизатора:

1. В случае явного задания идентификатора командой router-id, используется назначенный вручную ID.

2. В случае если не было ввода router-id, присваивается больший адрес из настроенных на маршрутизаторе loopback интерфейсов.
3. При отсутствии loopback интерфейсов принимается больший адрес из всех включенных на маршрутизаторе интерфейсов.

Для определения большего адреса используется прямое сравнение по октетам слева направо.

Открытый протокол маршрутизации не устанавливает отдельных требований к расчету метрики и оценки маршрутов. Его стандарт определяет стоимость каждого линка. В случае прохождения маршрута через несколько линков их стоимость суммируется. Оптимальным признается маршрут с наименьшей стоимостью. При этом принципы подсчета стоимости линка зависят от принципов, примененных конкретным производителем сетевого

оборудования.

Например, Cisco применяет два варианта расчета стоимости.

В первом случае стоимость линка рассчитывается как обратная величина от его скорости (1000 — для 1 Мбит, 100 — для 10 Мбит, 10 — для 100 Мбит, 1 — для 1 Гбита и т. д.). Этот вариант подойдет при условии, что все маршрутизаторы будут считать стоимость по данному алгоритму, аэто требует использование только устройств Cisco.

Второй способ предусматривает задание стоимости администратором на основе собственного определения качества линка. Этот вариант используют в тех случаях, когда качество линка определяется не одной только его скоростью. В том числе метрика может быть завышена для линка, на котором чаще других появляются ошибки или осуществляется тарификация трафика. Этот способ применим в сетях, где установлены маршрутизаторы разных производителей.

К основным проблемам OSPF можно отнести работу протокола в сетях с множественным доступом. Распространена топология, при которой множество маршрутизаторов объединяются не через последовательное подключение друг к другу, а через общую сеть. Теоретически OSPF должен выстраивать соседства в пределах общей сети на основе принципа «каждый с каждым». Однако это требует формирования огромных таблиц, работа с которыми сильно перегружает процессор и память.

Решение этой проблемы достигается посредством механизма выбора Designated Router (DR) и Backup Designated Router (BDR), которые представляют собой роли маршрутизаторов. В сети с множественным доступом, к которой подключены более 2 маршрутизаторов, один из них назначается на роль DR, а второй — на роль BDR. При отправке любым маршрутизатором какого-либо пакета, он поступает не всем устройства в сети, а подается на отдельный мультикастовый адрес, доступный только DR и BDR. В свою очередь, DR рассылает пакет всем маршрутизаторам в сети. Такое посредничество значительно снижает нагрузку. BDR выполняет резервную функцию и моментально принимает роль DR при его отключении. После этого среди остальных маршрутизаторов сразу выбирается новый BDR.

Выбор DR маршрутизатор осуществляется по следующему принципу:

1. DR: Маршрутизатор с самым высоким приоритетом интерфейса OSPF.
2. BDR: Маршрутизатор со вторым самым высоким приоритетом интерфейса OSPF.
3. Если приоритеты интерфейсов OSPF равны, самый высокий ID маршрутизатора используется, чтобы осуществить выбор.

Важно заметить, что процесс выбора DR и BDR не происходит сразу после получения первых Hello-пакетов от второго маршрутизатора. Для этого существует специальный таймер равный Router Dead Interval — 40 секунд. Если за это время не будет получен Hello-пакет с лучшим ID, то произойдет выбор на основе уже имеющихся Hello-пакетов.

Маршрутизатор, выбранный DR или BDR в одной присоединённой к нему

сети множественного доступа, может не быть DR (BDR) в другой присоединённой сети множественного доступа. Роль DR (BDR) является свойством интерфейса, а не свойством всего маршрутизатора. Иными словами, в каждом сегменте множественного доступа (например, коммутационном сегменте Ethernet), в котором общаются два или более OSPF-маршрутизатора, процесс выбора и распределение ролей DR/BDR происходит независимо от других сегментов множественного доступа.

ВИДЫ OSPF СООБЩЕНИЙ

Протокол маршрутизации OSPF поддерживает 5 типов сообщений:

1. Hello — периодически направляются для поиска соседей.
2. Database Description (DBD) — применяются для контроля синхронизации LSDB у соседей.
3. Link state request (LSR) — запрос у LSA у маршрутизатора, выполняемый принудительно. Применяется в случаях, когда маршрутизатор только включается и ему необходимо определить действующие в сети связи, а также если у него пропала сеть и нужно найти альтернативные маршруты.
4. Link state update (LSU) — содержит данные о состоянии связей маршрутизатора.
5. Link-State Acknowledgment (LSAck) — подтверждает получение других типов пакетов.

Объявление о состоянии канала (Link State Advertisement, LSA) — единица данных, которая описывает локальное состояние маршрутизатора или сети. Множество всех LSA, описывающих маршрутизаторы и сети, образуют базу данных состояния каналов (LSDB). LSDB состоит из нескольких видов LSA:

1. Type 1 LSA — Router LSA — объявление о состоянии каналов маршрутизатора. Эти LSA распространяются всеми маршрутизаторами. В LSA содержится описание всех каналов маршрутизатора и стоимость (cost) каждого канала. Распространяются только в пределах одной зоны.

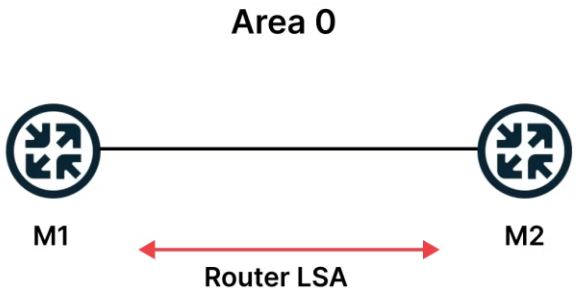

Рисунок 1 – Type 1 LSA

2. Type 2 LSA — Network LSA — объявление о состоянии каналов сети. Распространяется DR в сетях со множественным доступом. В LSA содержится описание всех маршрутизаторов присоединенных к сети, включая DR. Распространяются только в пределах одной зоны.

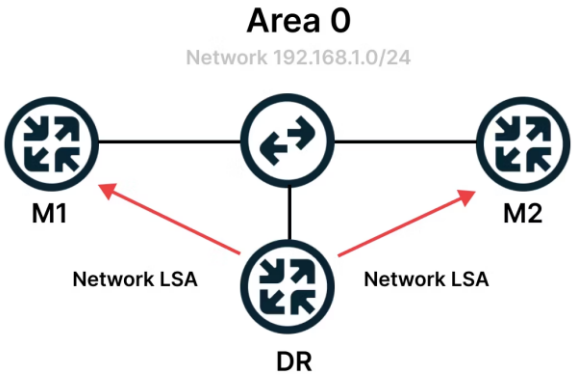

Рисунок 2 - Type 2 LSA

3. Type 3 LSA — Network Summary LSA — суммарное объявление о состоянии каналов сети. Объявление распространяется пограничными маршрутизаторами. Объявление описывает только маршруты к сетям вне зоны и

не описывает маршруты внутри автономной системы. Пограничный маршрутизатор отправляет отдельное объявление для каждой известной ему сети.

Когда маршрутизатор получает Network Summary LSA от пограничного маршрутизатора он не запускает алгоритм вычисления кратчайшего пути. Маршрутизатор просто добавляет к стоимости маршрута указанного в LSA стоимость маршрута к пограничному маршрутизатору. Затем маршрут к сети через пограничный маршрутизатор помещается в таблицу маршрутизации.

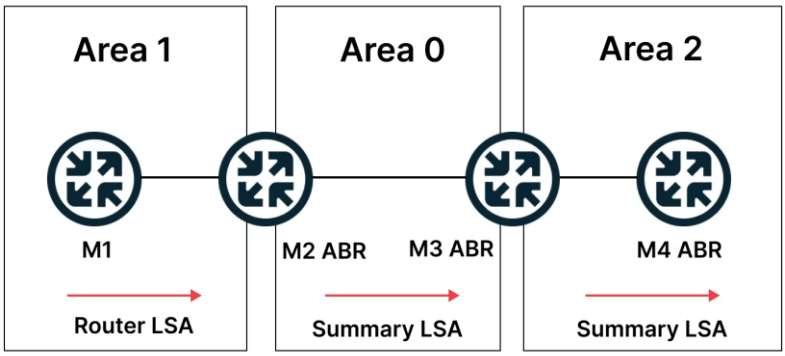

Рисунок 3 - Маршрутизатор M2 ABR создает LSA 3 и отправляет в зону Area 0.

4. Type 4 LSA — ASBR Summary LSA — суммарное объявление о состоянии каналов пограничного маршрутизатора автономной системы. Объявление распространяется пограничными маршрутизаторами. ASBR Summary LSA отличается от Network Summary LSA тем, что распространяется информация не о сети, а о пограничном маршрутизаторе автономной системы.

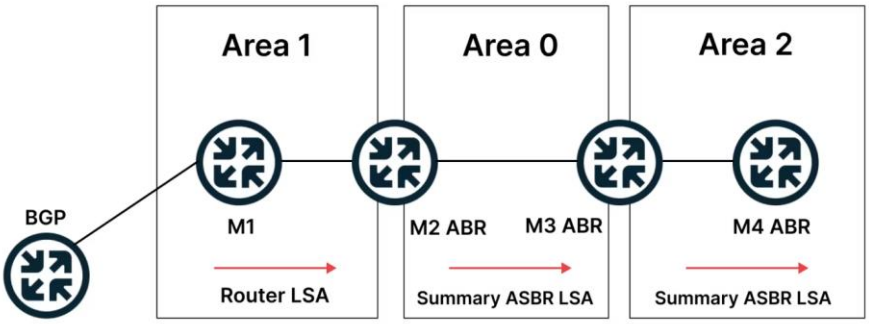

Рисунок 4 - Маршрутизатор M2 ABR принимает пакет LSA 1 от M1 и создает пакет LSA 4, который передает маршрут ASBR (из Area 1) и вводит его в Area 0.

5. Type 5 LSA — AS External LSA — объявление о состоянии внешних каналов автономной системы. Объявление распространяется пограничным маршрутизатором автономной системы в пределах всей автономной системы. Объявление описывает маршруты, внешние для автономной системы OSPF, или маршруты по умолчанию (default route), внешние для автономной системы OSPF.
6. Type 6 LSA — Multicast OSPF LSA — специализированный LSA, который используют мультикаст OSPF приложения (не осуществлено Cisco).
7. Type 7 LSA — AS External LSA for NSSA — объявления о состоянии внешних каналов автономной системы в NSSA зоне. Это объявление может передаваться только в NSSA зоне. На границе зоны пограничный маршрутизатор преобразует type 7 LSA в type 5 LSA.
8. Type 8 LSA — Link LSA — анонсирует link-local адрес и префикс(ы) маршрутизатора всем маршрутизаторам

разделяющим канал (link). Отправляется только если на канале присутствует более чем один маршрутизатор. Распространяются только в пределах канала (link).

9. Type 9 LSA — Intra-Area-Prefix LSA — ставит в соответствие: список префиксов IPv6 и маршрутизатор, указывая на Router LSA, список префиксов IPv6 и транзитную сеть, указывая на Network LSA. Распространяются только в пределах одной зоны.

Hello пакеты отправляются с установленной периодичностью. По умолчанию она составляет 1 раз в 10 секунд для сетей BMA (Broadband Multi-Access) и point-to-point и 1 раз в 40 секунд для сетей NBMA (Non-Broadcast Multi-Access). Также существует понятие Dead-интервала, который по умолчанию равняется 4 Hello-интервалам. Если в течение этого периода маршрутизатор не получает ни одного пакета, то он считает, что сосед отключился. За этим следует пересчет и обновление таблицы маршрутизаторы.

ПРИМЕР РАБОТЫ

Обеспечение работы протокола OSPF между двумя маршрутизаторами:

No.	Time	Source	Destination	Protocol	Length	Info
2	1.707699	192.168.1.1	224.0.0.5	OSPF	90	Hello Packet
11	11.171751	192.168.1.1	224.0.0.5	OSPF	90	Hello Packet
17	19.712072	192.168.1.2	224.0.0.5	OSPF	90	Hello Packet
22	20.391727	192.168.1.1	224.0.0.5	OSPF	94	Hello Packet
23	20.394355	192.168.1.2	192.168.1.1	OSPF	78	DB Description
24	20.395812	192.168.1.2	192.168.1.1	OSPF	94	Hello Packet
25	20.416755	192.168.1.1	192.168.1.2	OSPF	78	DB Description
26	20.422136	192.168.1.1	192.168.1.2	OSPF	98	DB Description
27	20.426201	192.168.1.2	192.168.1.1	OSPF	98	DB Description
28	20.436773	192.168.1.1	192.168.1.2	OSPF	70	LS Request
29	20.440436	192.168.1.1	192.168.1.2	OSPF	78	DB Description
30	20.443697	192.168.1.2	192.168.1.1	OSPF	122	LS Update
31	20.453707	192.168.1.2	192.168.1.1	OSPF	70	LS Request
32	20.459486	192.168.1.1	192.168.1.2	OSPF	122	LS Update
33	20.914637	192.168.1.1	224.0.0.5	OSPF	122	LS Update
34	20.958876	192.168.1.1	224.0.0.5	OSPF	94	LS Update
36	22.963538	192.168.1.1	224.0.0.5	OSPF	78	LS Acknowledge
37	22.979772	192.168.1.2	224.0.0.5	OSPF	118	LS Acknowledge
40	25.273057	192.168.1.2	224.0.0.5	OSPF	122	LS Update
43	27.813947	192.168.1.1	224.0.0.5	OSPF	78	LS Acknowledge
45	28.848207	192.168.1.2	224.0.0.5	OSPF	94	Hello Packet
46	30.158025	192.168.1.1	224.0.0.5	OSPF	94	Hello Packet
54	38.367012	192.168.1.2	224.0.0.5	OSPF	94	Hello Packet
56	39.669315	192.168.1.1	224.0.0.5	OSPF	94	Hello Packet
63	47.934789	192.168.1.2	224.0.0.5	OSPF	94	Hello Packet
65	49.205721	192.168.1.1	224.0.0.5	OSPF	94	Hello Packet
71	57.355533	192.168.1.2	224.0.0.5	OSPF	94	Hello Packet

Рисунок 5 - Сетевой трафик после установки OSPF

Сначала Hello отправляет только маршрутизатор 1.1.1.1 (192.168.1.1), как только мы включим OSPF на 192.168.1.2, то отправится Hello пакет. Первый и второй маршрутизатор получат Hello пакеты друг у друга и для того, чтоб соседство состоялось важно, чтобы в конфигурации OSPF на обоих маршрутизаторах были одинаковыми следующие параметры:

- Hello Interval — частота отправки сообщений Hello
- Router Dead Interval — период времени, по прохождению которого, сосед считается недоступным, если не было Hello.

- Area ID — соседство может установится только посредством интерфейсов в одной зоне.
- Authentication — пароль использующийся для аутентификации и тип аутентификации, если он есть.
- Stub area flag — необязательный флаг, который устанавливается на всех маршрутизаторах, которые принадлежат тупиковой зоне (stub area)

Как только один из маршрутизаторов получит новый Hello-пакет и проверит все условия, он сразу отправит Hello-пакет, где укажет в поле Active Neighbor адрес нового маршрутизатора, второй маршрутизатор, получив и увидев себя в поле соседей себя, добавит первый в соседи и отправит уже юникастовый пакет на 192.168.1.1 (первый маршрутизатор), где укажет его соседом.

После установки соседства начинается этап построения базы LSBD.

LSDB состоит из нескольких видов LSA. В сообщениях DBD используется достаточно много флагов для определения состояния синхронизации, а также данные сообщения содержат информацию о собственной базе данных. То есть, первый маршрутизатор сообщает в данных сообщениях, что в его базе есть информация о таких сетях, как 192.168.0/24, 1.1.1.0/24 (LSA Type 1), а второй маршрутизатор в свою очередь сообщает, что у него есть записи о сетях: 192.168.2.0/24, 2.2.2.0/24 (LSA Type 1). После получения сообщений DBD, каждый маршрутизатор отправляет LSAck в подтверждение о полученном сообщении, и, далее, сравнивает информацию в базе соседа со своей. Если найдено, что не достает какой-либо информации, то маршрутизатор отправляет LS Request, где запрашивает полную информацию о каком-либо LSA. Например, первый маршрутизатор запросил LS Request у второго, а второй отправляет в ответ LS Update, в которой уже содержится подробная

информация о каждом маршруте.

ЗАКЛЮЧЕНИЕ

Протокол OSPF (Open Shortest Path First) является одним из наиболее распространённых и эффективных внутренних протоколов маршрутизации, используемых в современных компьютерных сетях. Благодаря своей способности быстро адаптироваться к изменениям топологии, поддержке масштабируемости и использованию алгоритма Дейкстры для определения кратчайших путей, OSPF обеспечивает стабильную и оптимальную маршрутизацию внутри автономных систем. Его гибкая структура, возможность разделения сети на области и поддержка различных типов интерфейсов делают его предпочтительным выбором для крупных корпоративных и провайдерских сетей. В целом, протокол OSPF играет важную роль в обеспечении надежной и эффективной работы современных сетевых инфраструктур, способствуя быстрому обмену маршрутной информацией и повышая устойчивость сети к сбоям.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Протокол OSPF // https://ru.wikipedia.org/wiki/OSPF
2. Документация к протоколу OSPF // https://datatracker.ietf.org/doc/html/rfc2328
3. Дополнения к протоколу OSPF (часть 1) // https://habr.com/ru/articles/418391
4. Дополнение к протоколу OSPF (часть 2) // https://selectel.ru/blog/ospf

ПРИЛОЖЕНИЕ

client.cpp

```
#include <iostream>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <unistd.h>


int main(int argc, char **argv)
{
    int sockMain, sockServ;
    struct sockaddr_in serv;
    sockMain = socket(AF_INET, SOCK_STREAM, 0);
    if (sockMain < 0)
    {
        std::cerr << "SOCKET OPEN ERROR" << '\n';
        close(sockMain);
        return 1;
    }
    serv.sin_family = AF_INET;
    serv.sin_addr.s_addr = INADDR_ANY;
    int time = atoi(argv[2]);
    serv.sin_port = htons(atoi(argv[1]));
    socklen_t s_size = sizeof(serv);
    if ((sockServ = connect(sockMain, (struct sockaddr*) &serv, s_size)) == -1)
    {
        std::cerr << "CONNECT ERROR" << '\n';
        close(sockServ);
        close(sockMain);
        return 1;
    }
    std::cout << "CONNECTED TO SERVER" << '\n';
    for(int i = 0; i <= time ; i++)
    {
        std::cout << "send: " << time << '\n';
        if (send(sockMain, &time, 4, 0) < 0)
        {
            std::cerr << "SENDING RROR\n";
            exit(1);
        }
        sleep(time);
```

```
        recv(sockMain, &time, sizeof(time), 0);
        std::cout << "received: " << time << '\n';
    }
    close(sockServ);
    close(sockMain);
    return 0;
}
```

server.cpp

```
#include <iostream>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <signal.h>


void killzom(int sign)
{   int stat;

while(wait3(&stat, WNOHANG, (struct rusage *)0) >= 0);
}


int main (int argc, char **argv)
{
    int sockMain, sockBind, check, sockClient, stat, childs;
    struct sockaddr_in serv;
    sockMain = socket(AF_INET, SOCK_STREAM, 0);
    if (sockMain < 0)
    {
        std::cerr << "SOCKET ERROR" << '\n';
        return 1;
    }
    childs = 0;
    socklen_t size = sizeof(struct sockaddr);
    serv.sin_family = AF_INET;
    serv.sin_addr.s_addr = INADDR_ANY;
    serv.sin_port = 0;
    signal(SIGCHLD, killzom);
    sockBind = bind(sockMain, (struct sockaddr *) &serv, sizeof(serv));
    if (sockBind == -1)
    {
```

```
    std::cerr << "BIND ERROR" << '\n';
    return 1;
}
if (getsockname(sockMain, (struct sockaddr *) &serv, &size))
{
    std::cerr << "GET SOCKET NAME ERROR" << '\n';
    return 1;
}
std::cout << "PORT: " << ntohs(serv.sin_port) << '\n';
while (1)
{
    if ((check = listen(sockMain, 5)) == -1)
    {
        std::cerr << "LISTEN ERROR" << '\n';
        return 1;
    }
    if ((sockClient = accept(sockMain, (struct sockaddr *) 0, 0)) == -1)
    {
        std::cerr << "ACCEPT ERROR" << '\n';
        return 1;
    }
    std::cout << "| connected |" << '\n';
    childs++;
    pid_t im = fork();
    switch (im)
    {
        case -1:
            std::cerr << "FORK ERROR" << '\n';
            wait(&stat);
            return -1;
            break;
        case 0:
            while (1) {
                int buf = 0;
                ssize_t tmp = 0;
                tmp = recv(sockClient, &buf, 4, 0);
                std::cout << "received: " << buf << '\n';
                if (tmp == 0)
                {
                    close(sockClient);
                    exit(0);
                }
                send(sockClient, &buf, 4, 0);
            }
            break;
        default:
```

```
            if(childs >= 5)
            {
                wait(&stat);
                close(sockMain);
                close(sockClient);
                exit(0);
            }
            break;
        }
    }
    return 0;
}
```