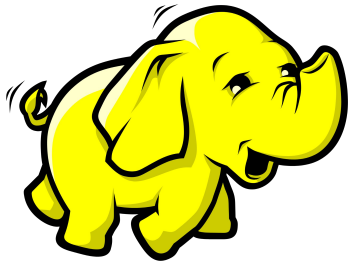
UDP

Илья Кокорин

kokorin.ilya.1998@gmail.com

Формат практик

- Я рассказываю теорию
- Знакомлю с **основами** какой-то системы
- Я пишу код, вы подсказываете
  - Можно травить байки
  - Рассказывать про какие-то gotchas
  - Или про её аналоги



RabbitMQ

protobuf
Protocol Buffers

akka

APACHE ZooKeeper™

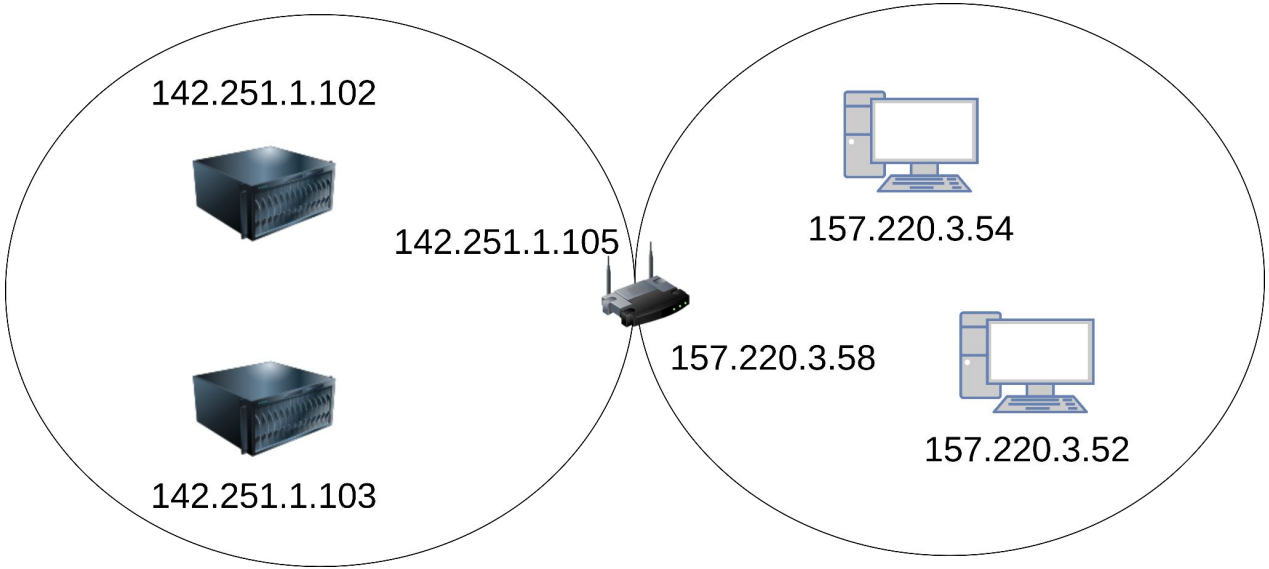
Сокеты

- Они же *Сокеты Беркли*
- Стандартная абстракция для общения процессов по сети
- Появились в 1982 году в ОС 4.1 BSD Unix
- Сокеты бывают разные
  - Мы остановимся на UDP сокетах

Адресация в компьютерных сетях: IP-адрес

- Грубо говоря, это адрес устройства, подключённого к сети

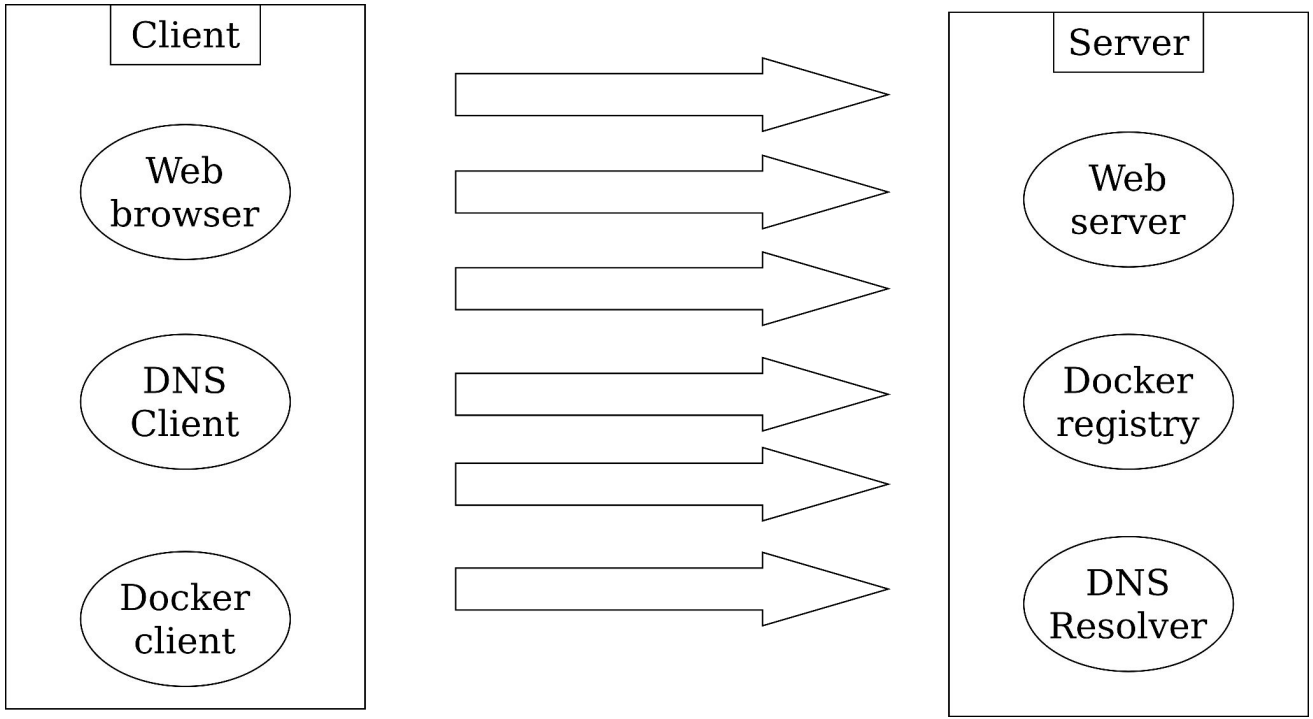
- Бывают IPv4-адреса
  - 4 байта
  - 142.251.1.102



- И IPv6
  - 16 байт
  - 2a00:1450:4010:c1e::66

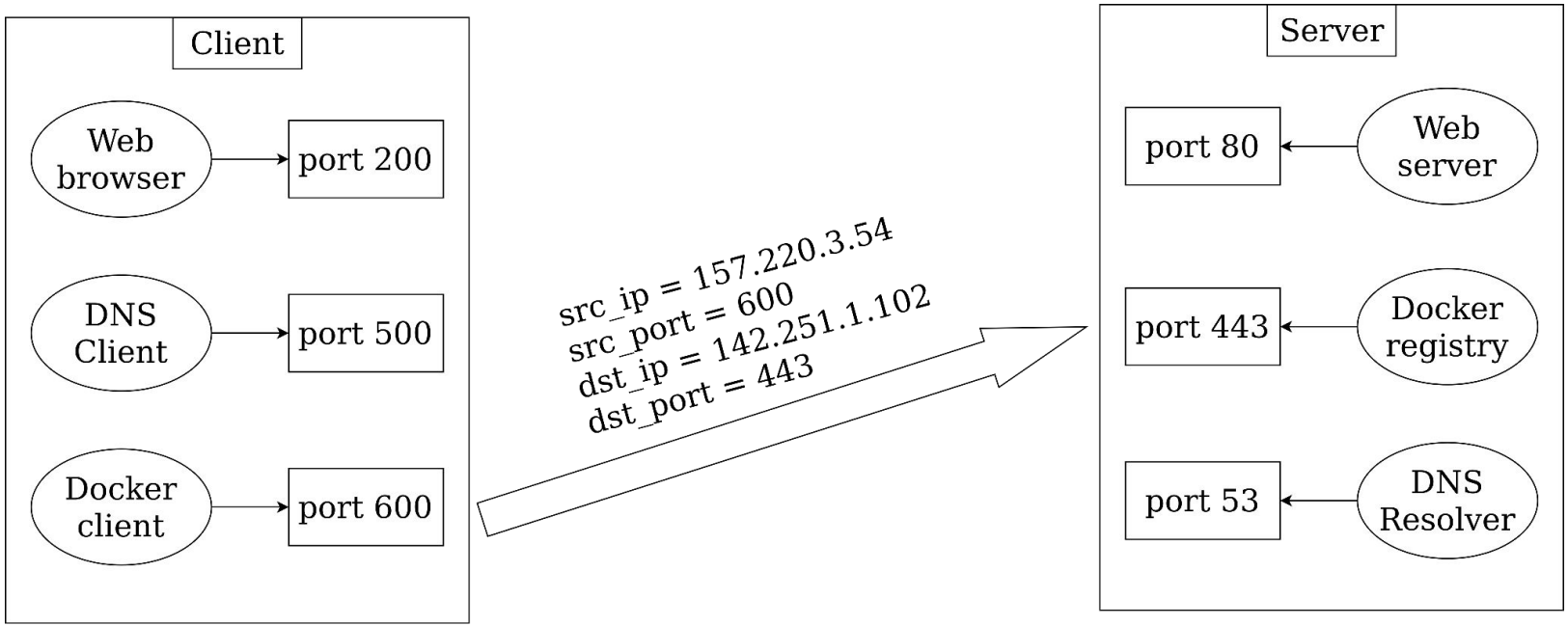
Адресация в компьютерных сетях: порты

- На наш IP-адрес пришло сообщение
- Какому из процессов оно предназначено?



Адресация в компьютерных сетях: порты

- По сообщению можно понять, какому приложению адресовано сообщение
- И куда отправлять ответ

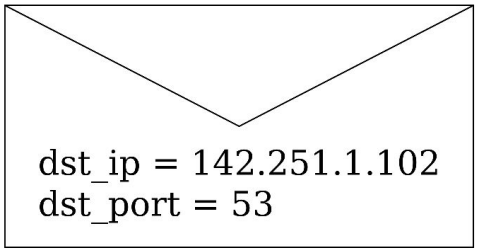


Адресация в компьютерных сетях: порты

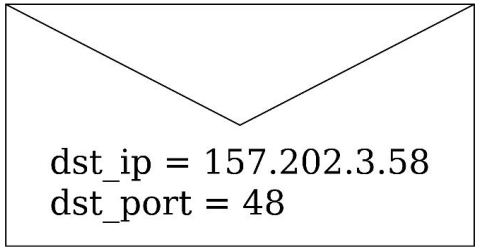
- Номер порта - 2 байта
  - То есть 16 бит
- То есть уникальных портов чуть больше 65 тысяч
  - Иногда порты кончаются!
  - Проблема *Port Exhaustion*
  - Читайте книги по компьютерным сетям чтобы узнать, что делать в таком случае

UDP сокет: что это такое?

- Почтовый ящик
- Через один сокет отправляем сообщения кому угодно



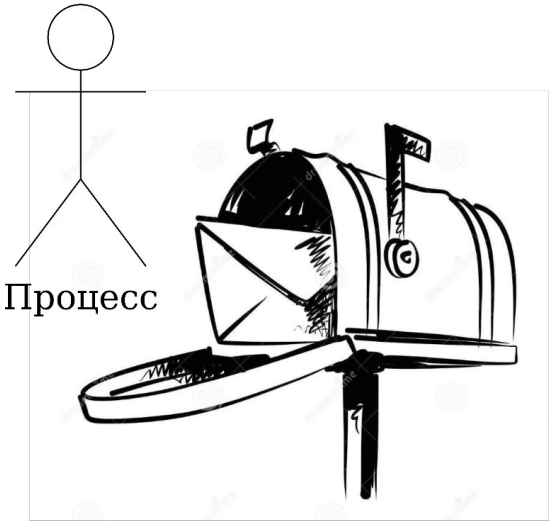
Отправляем на
142.251.1.102:53

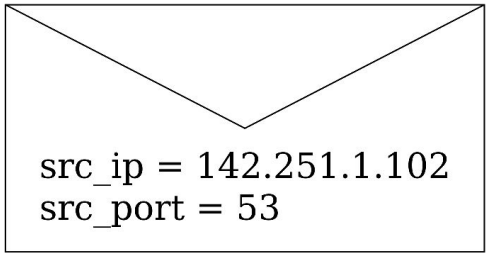


Отправляем на
157.202.3.58:48

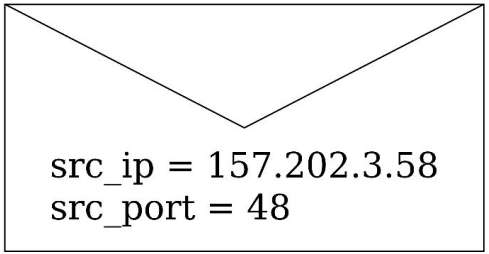
UDP сокет: что это такое?

- Почтовый ящик
- Через один сокет принимаем сообщения от кого угодно





Получили от
142.251.1.102:53



Получили от
157.202.3.58:48

UDP сокет: операции

- Создание сокета
  - `fd := socket(IPPROTO_UDP)`
- Привязка к хосту и порту
  - `bind(fd, host, port)`
- Получение сообщения
  - `msg, s_host, s_port := recvfrom(fd)`
  - Управление не вернётся вызывающему потоку до прихода сообщения
- Отправка сообщения
  - `sendto(fd, msg, d_host, d_port)`
- Закрытие сокета
  - `close(fd)`

UDP: схема процесса

- Всё общение с остальными процессами происходит через единственный сокет

```
1  state := /* Initial state */
2  udp_socket := socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)
3  bind(udp_socket, local_host, local_port)
4  while true:
5      msg, sender_host, sender_port := recvfrom(udp_socket)
6      state ← /* New state */
7      receiver_host, receiver_port := /* Receiver address */
8      message_to_send := /* Some message */
9      sendto(udp_socket, message_to_send, receiver_host, receiver_port)
10     if should_exit:
11         break
12 close(udp_socket)
```

Немного специфики: Порядок байт

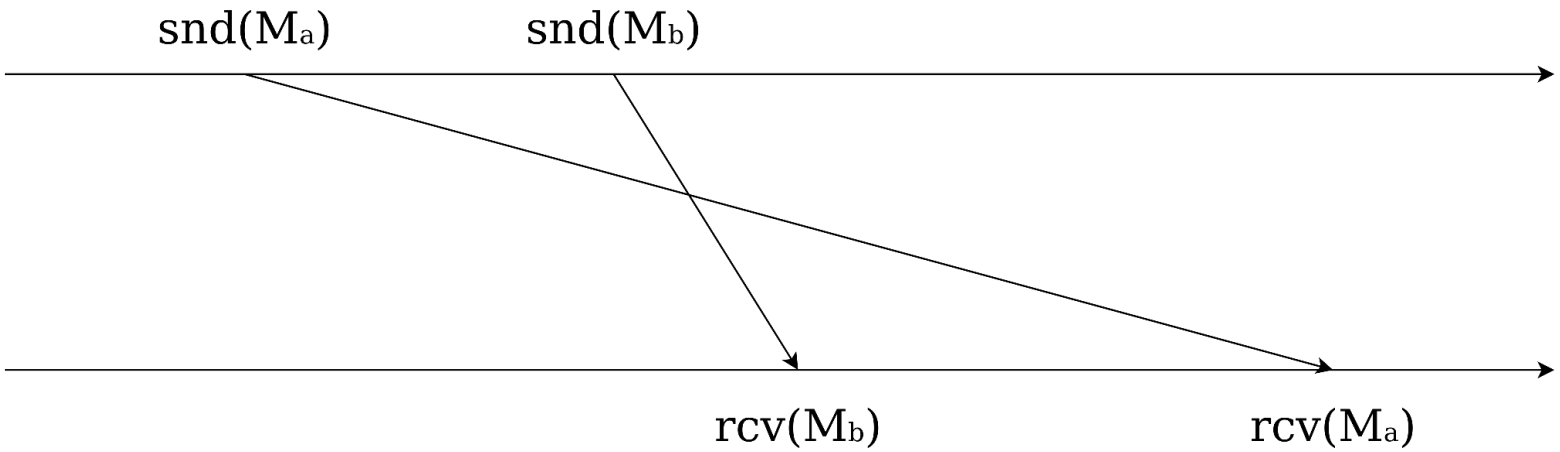
- Пусть у нас есть число (номер порта)
  - Например, 65000_{10} = 0xFDE8
  - На самом деле это два байта
  - 0xFD и 0xE8
- Можно хранить байты в порядке big-endian
  - От старшего байта к младшему
  - 0xFD; 0xE8;
- Можно хранить байты в порядке little-endian
  - От младшего байта к младшему
  - 0xE8; 0xFD;

Немного специфики: Порядок байт

- API требует чтобы байты порта были записаны в big-endian
- В памяти вашей машины они записаны в little-endian
  - Если у вас x86 или ARM
- `uint32_t htonl(uint32_t hostlong);`
  - host to network long
- `uint16_t htons(uint16_t hostshort);`
  - host to network short
- `uint32_t ntohl(uint32_t netlong);`
  - network to host long
- `uint16_t ntohs(uint16_t netshort);`
  - network to host short

Гарантии UDP: нарушение порядка

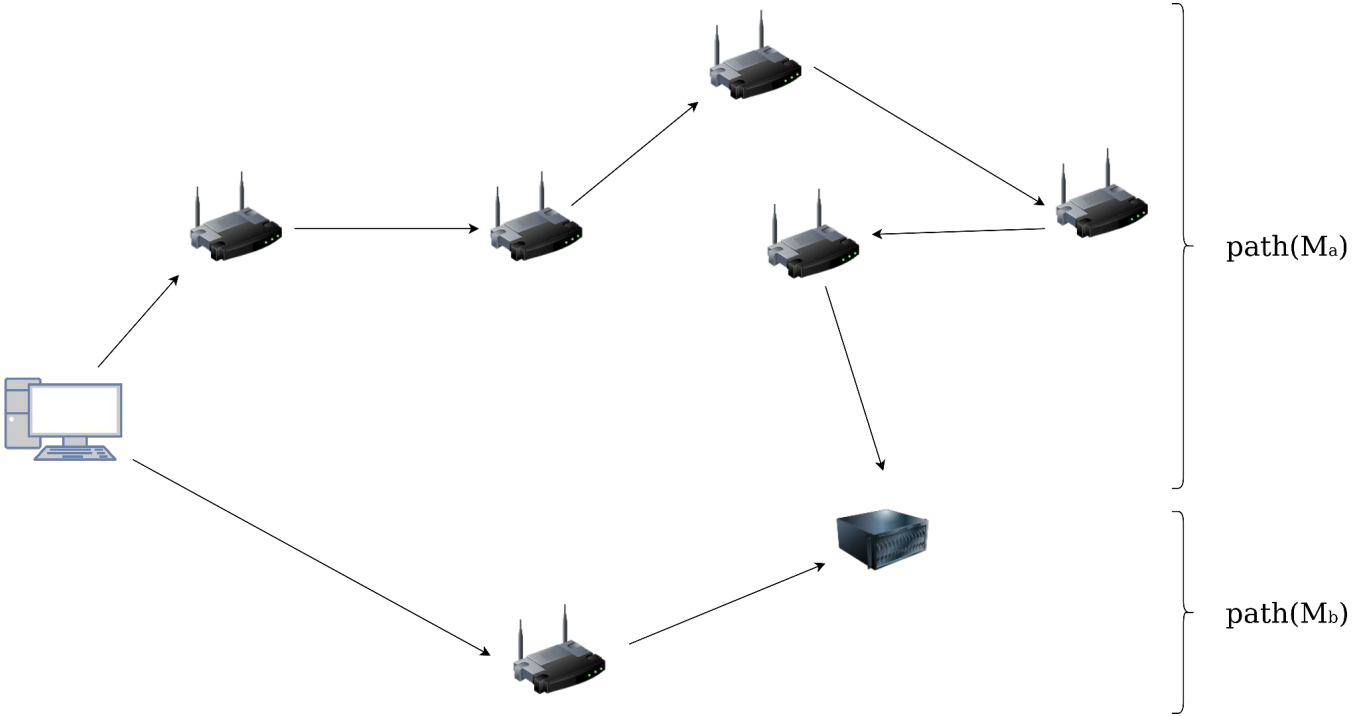
- $snd(M_a) < snd(M_b)$
- $rcv(M_b) < rcv(M_a)$



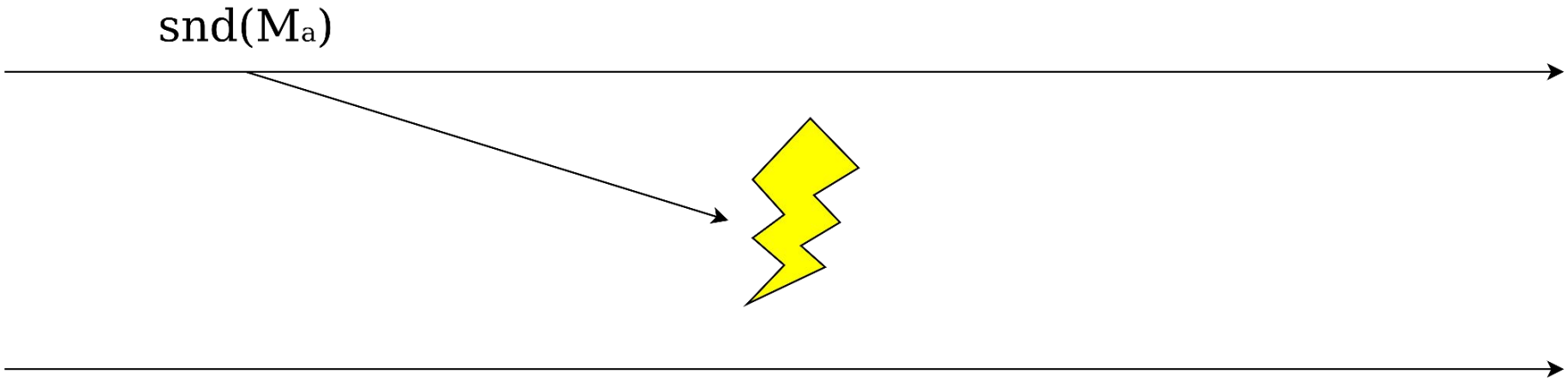
- Какой алгоритм распределённой блокировки нельзя написать?

Гарантии UDP: нарушение порядка

- Более раннее сообщение могло пойти более длинным путём



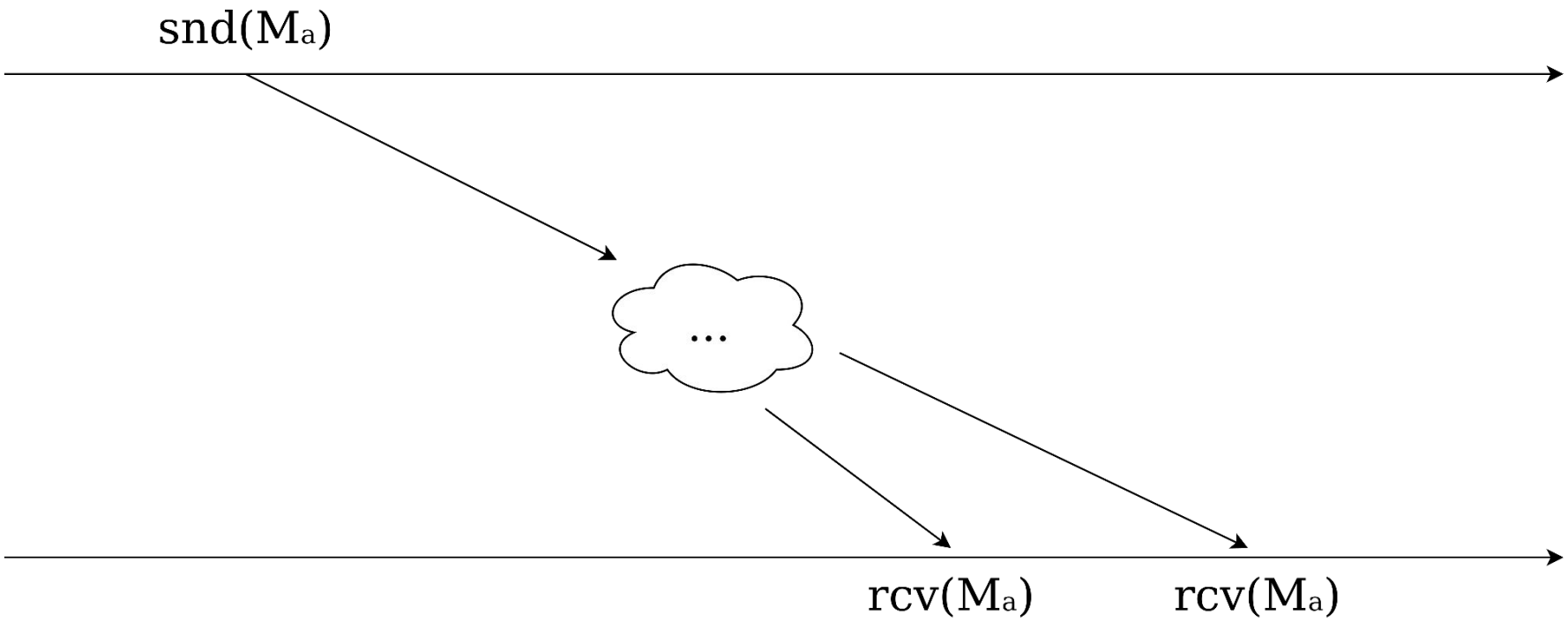
Гарантии UDP: потеря сообщений



- Провод порвался
- Маршрутизатор перегружен
  - Выкинул лишние сообщения из очереди
- На том конце никого нет
- И ещё тысяча причин

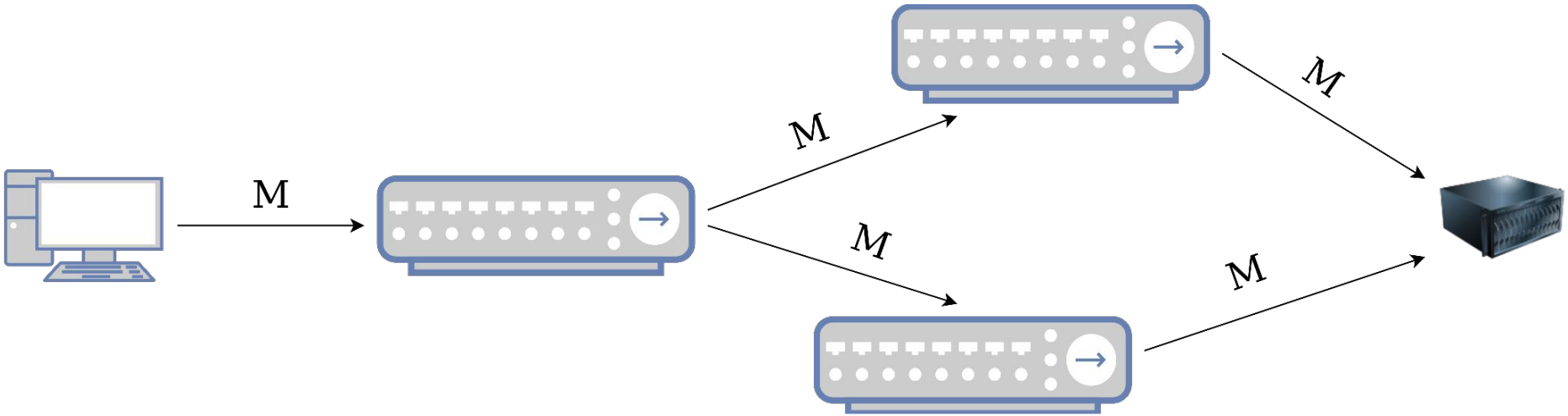
Гарантии UDP: дублирование сообщений

- Редко, но возможно



Гарантии UDP: дублирование сообщений

- Сетевой концентратор (Ethernet hub)
  - Получает сообщение на аппаратный порт
  - Ретранслирует его на все остальные аппаратные порты



Гарантии UDP: контрольная сумма

```
1 fun sendto(fd, msg, host, port):
2     emsg := ⟨ msg, hash(msg) ⟩
3     __sendto_impl(fd, emsg, host, port)
```

```
1 fun recvfrom(fd):
2     while true:
3         ⟨ msg, ch ⟩, host, port := __recvfrom_impl(fd)
4         if hash(msg) = ch:
5             return msg, host, port
```

- Если контрольная сумма сошлась - скорее всего содержимое сообщения не изменилось

Гарантии UDP: контрольная сумма

- Стандартная контрольная сумма в UDP достаточно слабая
  - $\texttt{udp\_hash}(M_a) = \texttt{udp\_hash}(M_b)$
  - $M_a \neq M_b$
- Пользуйтесь более сильными контрольными суммами
- SHA-256
  - Не найдено ни одной коллизии
- xxHash, MurmurHash, MD5
  - Коллизии существуют, но достаточно редки
  - **Не являются криптографически безопасными!**

Что почитать: компьютерные сети

- *Tannenbaum A. S.* Computer Networks.
- *Stevens W. R.* TCP/IP illustrated, Volumes 1-3.
- *Seth S., Venkatesulu M. A.* TCP/IP Architecture, Design and Implementation in Linux.
- *Rosen R.* Linux kernel networking: Implementation and theory.
- *Kerrisk M.* The Linux programming interface: a Linux and UNIX system programming handbook.

Что почитать & посмотреть: сокеты

- How to receive a million packets per second
- How to achieve low latency with 10Gbps Ethernet
- Ephemeral port exhaustion and how to avoid it
- How to stop running out of ephemeral ports and start to love long-lived connections
- *Александр Тоболь.* Пишем свой протокол поверх UDP
- *Александр Тоболь.* UDP против TCP, или Будущее сетевого стека

Что почитать: маны

- `man udp`
- `man socket`
- `man bind`
- `man sendto`
- `man recvfrom`
- `man man`

Что почитать: контрольные суммы

- How both TCP and Ethernet checksums fail
- *Stone J., Partridge C.* When the CRC and TCP checksum disagree
- AWS S3 Outage, July 2008
- Отъявленные баги и как их избежать на примере ClickHouse

Thanks for your attention