

# Протокол UDP

Сетевые технологии

---

Мишина А. А.

25 ноября 2024

- Протокол UDP
- Создан в 1980
- Быстрый
- Не гарантирует получение отправленных данных
- Протокол без установления соединения (connectionless)



Рис. 1: Дэвид П. Рид

## Протокол UDP

---

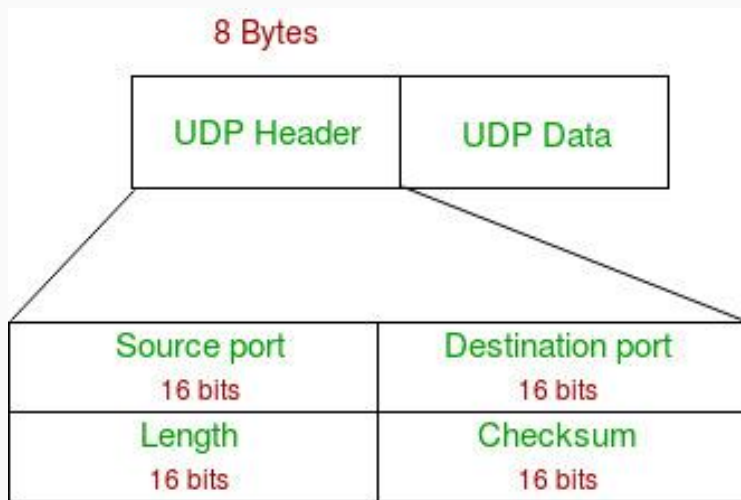


Рис. 2: Структура пакета UDP

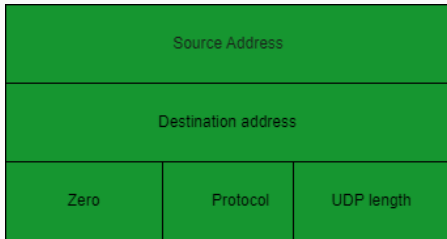


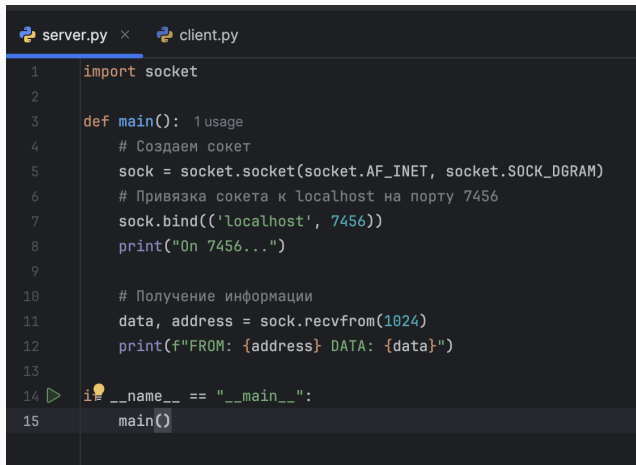
Рис. 3: Структура псевдозаголовка пакета UDP

- IP-адрес источника (длина 32 бит)
- IP-адрес получателя (длина 32 бит)
- Поле нулей (длина 8 бит)
- Протокол (длина 8 бит)
- Длина UDP (длина 16 бит)

## Практическое применение UDP

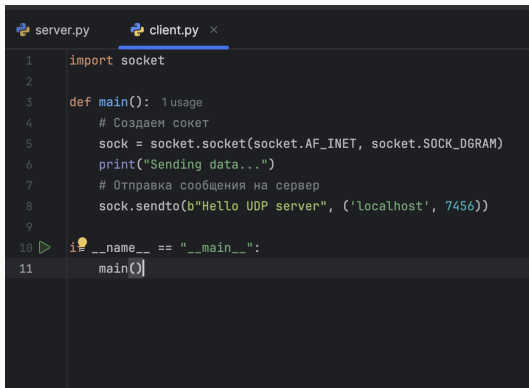
---

# Практическое применение UDP



```
server.py × client.py
1  import socket
2
3  def main(): 1 usage
4      # Создаем сокет
5      sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6      # Привязка сокета к localhost на порту 7456
7      sock.bind(('localhost', 7456))
8      print("On 7456...")
9
10     # Получение информации
11     data, address = sock.recvfrom(1024)
12     print(f"FROM: {address} DATA: {data}")
13
14  if __name__ == "__main__":
15     main()
```

Рис. 4: Сервер



```
server.py  client.py x
1  import socket
2
3  def main(): 1 usage
4      # Создаем сокет
5      sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6      print("Sending data...")
7      # Отправка сообщения на сервер
8      sock.sendto(b"Hello UDP server", ('localhost', 7456))
9
10  if __name__ == "__main__":
11      main()
```

Рис. 5: Клиент

- FROM: ('127.0.0.1', 53439) DATA: b'Hello UDP server'



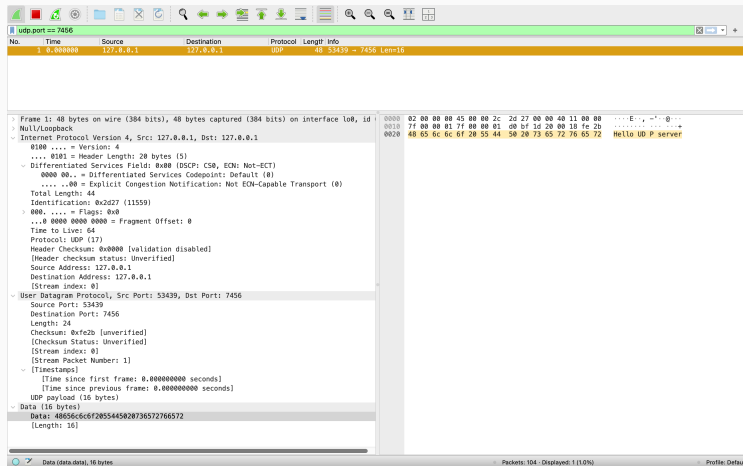


Рис. 6: Wireshark IPv4

```

  ▾ Frame 1: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface lo0, i
    Section number: 1
    ▾ Interface id: 0 (lo0)
      Interface name: lo0
      Interface description: Loopback
      Encapsulation type: NULL/Loopback (15)
      Arrival Time: Nov 24, 2024 22:38:10.770461000 MSK
      UTC Arrival Time: Nov 24, 2024 19:38:10.770461000 UTC
      Epoch Arrival Time: 1732477090.770461000
      [Time shift for this packet: 0.000000000 seconds]
      [Time delta from previous captured frame: 0.000000000 seconds]
      [Time delta from previous displayed frame: 0.000000000 seconds]
      [Time since reference or first frame: 0.000000000 seconds]
      Frame Number: 1
      Frame Length: 48 bytes (384 bits)
      Capture Length: 48 bytes (384 bits)
      [Frame is marked: False]
      [Frame is ignored: False]
      [Protocols in frame: null:ip:udp:data]
      [Coloring Rule Name: UDP]
      [Coloring Rule String: udp]

```

Рис. 7: Физический уровень



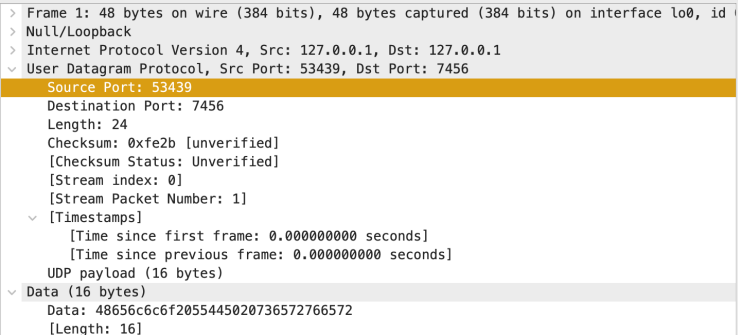
Рис. 8: Канальный уровень

```

  ▾ Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 44
    Identification: 0x2d27 (11559)
  ▾ 000. .... = Flags: 0x0
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 64
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
    [Stream index: 0]

```

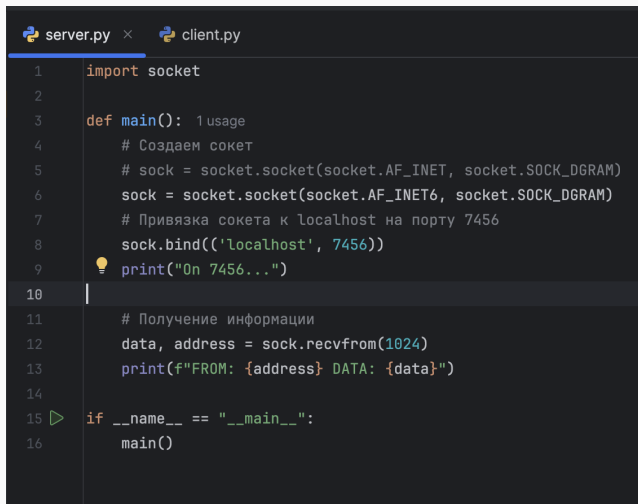
Рис. 9: Сетевой уровень



The image shows a Wireshark packet capture window. The top section, 'Packet List', shows 'Frame 1: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface lo0, id 1'. The bottom section, 'Packet Details', shows the following hierarchy: 'Null/Loopback' (expanded), 'Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1' (expanded), 'User Datagram Protocol, Src Port: 53439, Dst Port: 7456' (expanded), 'Source Port: 53439' (highlighted in orange), 'Destination Port: 7456', 'Length: 24', 'Checksum: 0xfe2b [unverified]' (with '[Checksum Status: Unverified]' and '[Stream index: 0]' below it), '[Stream Packet Number: 1]', '[Timestamps]' (expanded, showing '[Time since first frame: 0.000000000 seconds]' and '[Time since previous frame: 0.000000000 seconds]'), 'UDP payload (16 bytes)', and 'Data (16 bytes)' (expanded, showing 'Data: 48656c6c6f2055445020736572766572' and '[Length: 16]').

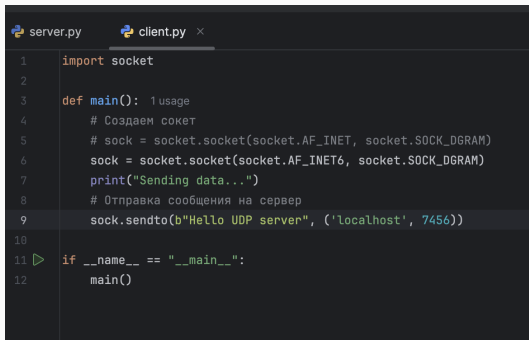
- > Frame 1: 48 bytes on wire (384 bits), 48 bytes captured (384 bits) on interface lo0, id 1
- > Null/Loopback
- > Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- ✓ User Datagram Protocol, Src Port: 53439, Dst Port: 7456
  - Source Port: 53439
  - Destination Port: 7456
  - Length: 24
  - Checksum: 0xfe2b [unverified]  
[Checksum Status: Unverified]
  - [Stream index: 0]
  - [Stream Packet Number: 1]
  - ✓ [Timestamps]
    - [Time since first frame: 0.000000000 seconds]
    - [Time since previous frame: 0.000000000 seconds]
  - UDP payload (16 bytes)
  - ✓ Data (16 bytes)
    - Data: 48656c6c6f2055445020736572766572
    - [Length: 16]

Рис. 10: Транспортный уровень



```
server.py x client.py
1  import socket
2
3  def main(): 1 usage
4      # Создаем сокет
5      # sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6      sock = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
7      # Привязка сокета к localhost на порту 7456
8      sock.bind(('localhost', 7456))
9      💡 print("On 7456...")
10
11     # Получение информации
12     data, address = sock.recvfrom(1024)
13     print(f"FROM: {address} DATA: {data}")
14
15  ▶ if __name__ == "__main__":
16     main()
```

Рис. 11: Сервер



```
server.py  client.py x
1  import socket
2
3  def main(): 1 usage
4      # Создаем сокет
5      # sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
6      sock = socket.socket(socket.AF_INET6, socket.SOCK_DGRAM)
7      print("Sending data...")
8      # Отправка сообщения на сервер
9      sock.sendto(b"Hello UDP server", ('localhost', 7456))
10
11  if __name__ == "__main__":
12      main()
```

Рис. 12: Клиент

- FROM: ('::1', 49389, 0, 0) DATA: b'Hello UDP server'

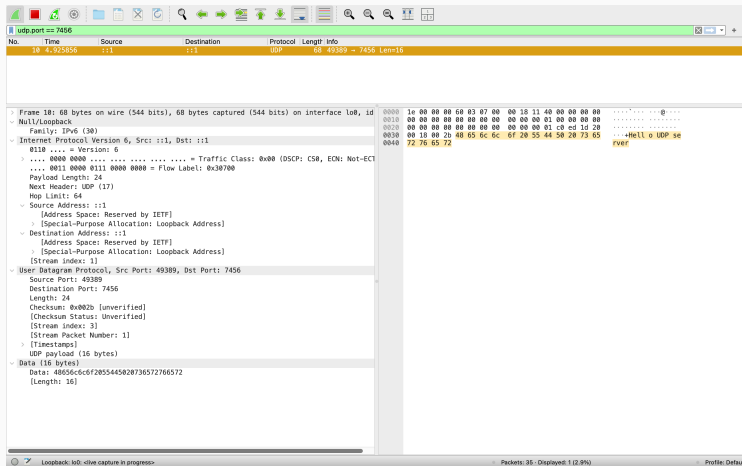


Рис. 13: Wireshark IPv6



## Преимущества и недостатки UDP

---

- Скорость
- Меньшая задержка
- Простота реализации
- Поддержка широковещательной и многоадресной передачи
- Меньшая нагрузка на сеть

- Отсутствие надёжности
- Отсутствие контроля ошибок
- Отсутствие управления потоком
- Делает систему уязвимой к атакам
- Ограниченная поддержка в некоторых сетях

- Важный протокол транспортного уровня
- Быстрый, но не надёжный
- Пользуясь UDP, приложение само несёт ответственность за коррекцию ошибок