

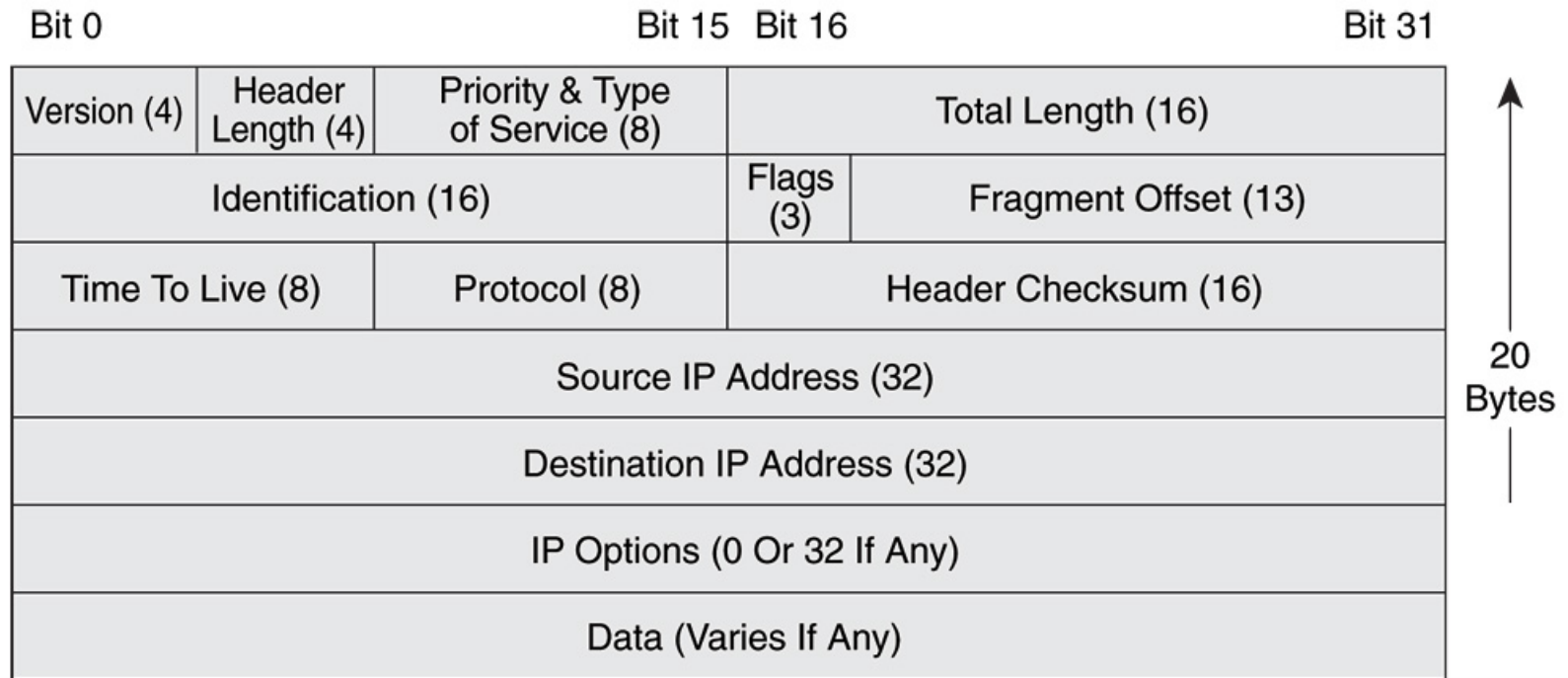
TCP level networking

Стек протоколов

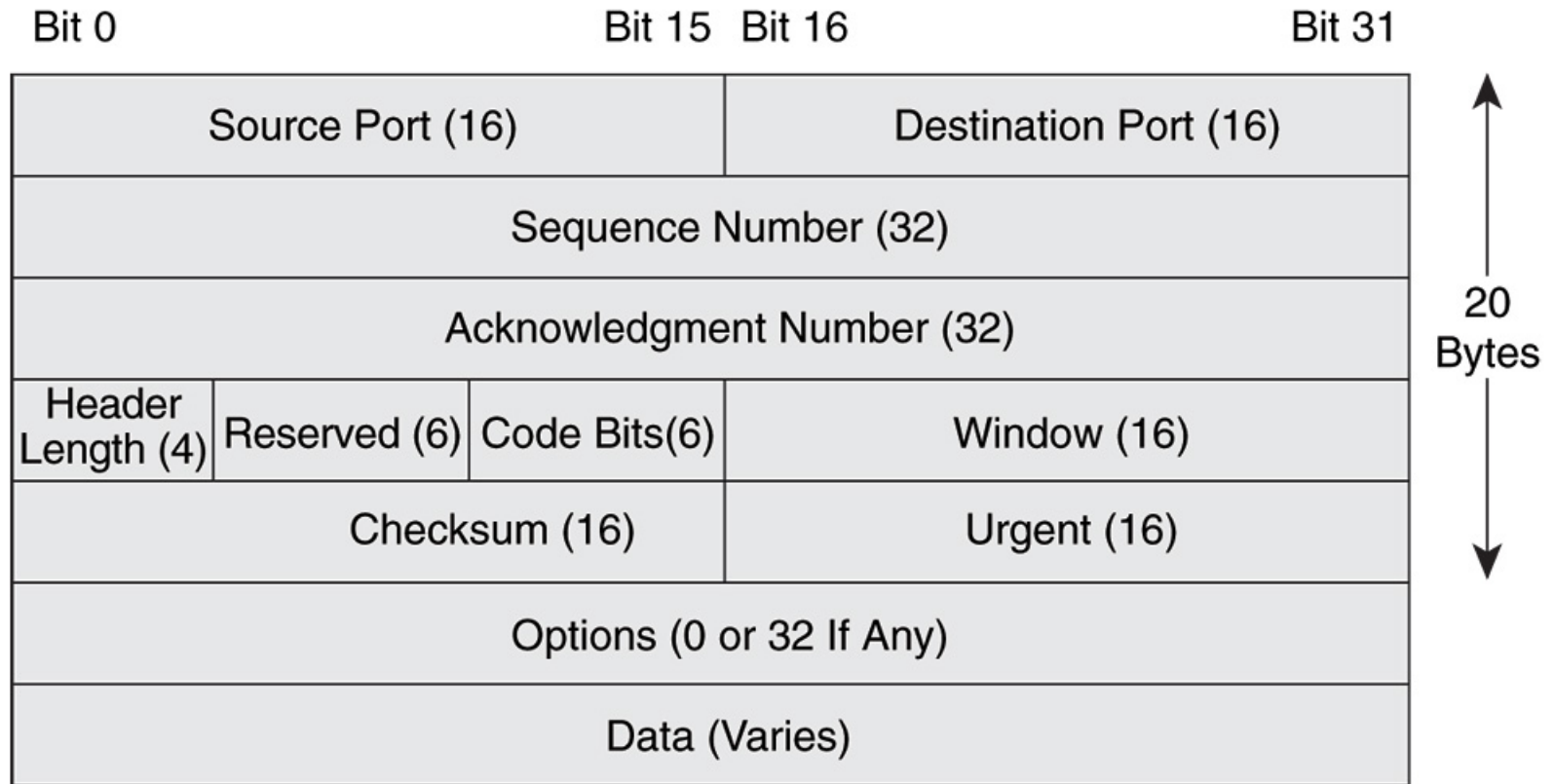
- Ethernet
- IP
- TCP/UDP
- HTTP/FTP/SSH/...

ISO vs. ANSI

IP



TCP/UDP



Python TCP client

```
1 import socket
2
3 with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
4     # tcp by default
5     s.connect(('127.0.0.1', 12345)) # ip or hostname
6     s.send("Hello , World!".encode('utf8'))
7
8     # 1
9     print("Recv", s.recv(1024).decode('utf8'))
10
11    # 2
12    # fd = s.makefile()
13    # fd.read()
```

Python TCP server

```
1 import socket
2
3 sz = 32
4
5 with socket.socket() as s:
6     s.bind((' ', 12345))
7     s.listen(1)
8
9     conn, addr = s.accept()
10    print('Connection address:', addr)
11    while True:
12        data = conn.recv(sz)
13        if not data:
14            break
15        print("received data:", data.decode('utf8'))
16        conn.send(data)
```

БД/DB2 API

- Connection (commit/rollback)
- Cursor (execute/fetchone/fetchall/executemany)
- Транзакция

sqlite3

```
1 import sqlite3
2
3 conn = sqlite3.connect( '/tmp/data.db ' )
4 cursor = conn.cursor()
5 conn.close()
```

sqlite3

```
1 import sqlite3
2
3 def prepare(cr):
4     try:
5         cr.execute("select key from messages limit 1")
6     except sqlite3.OperationalError:
7         cr.execute("create table messages (key text primary key,
8
9
10 def show_db(cr):
11     cr.execute("select key, msg from messages")
12     print("\n#-----")
13     for key, msg in cr.fetchall():
14         print(">>>", key, msg)
15     print("#-----\n")
```

sqlite3

```
1 conn = sqlite3.connect("/tmp/data.db")
2 cr = conn.cursor()
3 show_db(cr)
4 prepare(cr)
5 conn.commit()
6
7 cr = conn.cursor()
8 cr.execute("insert into messages values (?, ?)", ("1", "this wil
9 show_db(cr)
10 conn.rollback()
11 show_db(conn.cursor())
12
13
14 cr = conn.cursor()
15 cr.execute("insert into messages values (?, ?)", ("2", "this wil
16 show_db(cr)
17 conn.commit()
18 show_db(conn.cursor())
```

sqlite3

```
1 import contextlib
2
3 @contextlib.contextmanager
4 def transaction(conn):
5     cr = conn.cursor()
6     try:
7         yield cr
8     except:
9         conn.rollback()
10        raise
11    conn.commit()
```

sqlite3

```
1 with transaction(conn) as cr:
2     cr.execute("insert into messages values (?, ?)", ("5", "this
3
4 with transaction(conn) as cr:
5     show_db(cr)
6
7 with transaction(conn) as cr:
8     cr.execute("insert into messages values (?, ?)", ("6", "this
9     show_db(cr)
10    raise ValueError("")
11
12 with transaction(conn) as cr:
13     show_db(cr)
```

sqlite3

```
1 data = [(str(x), "msg {}".format(x)) for x in range(10, 20)]
2 with transaction(conn) as cr:
3     cr.executemany("insert into messages values (?, ?)", data)
```

sqlite3

```
1 from sqlalchemy import Column, String
2 from sqlalchemy.ext.declarative import declarative_base
3
4 Base = declarative_base()
5
6 class Message(Base):
7     __tablename__ = 'messages'
8     key = Column(String, primary_key=True)
9     msg = Column(String)
10
11     def __str__(self):
12         return "Message({0.key!r}, {0.msg!r})".format(self)
13
14 # Base.metadata.create_all(engine)
```

sqlite3

```
1 from sqlalchemy.orm import sessionmaker
2 from sqlalchemy import create_engine
3
4 engine = create_engine('sqlite:///tmp/data.db', echo=True)
5 Session = sessionmaker(bind=engine)
6 session = Session()
7
8 msg1 = Message(key="101", msg="test1")
9 msg2 = Message(key="102", msg="test2")
10 session.add_all([msg1, msg2])
11 session.commit()
12
13 for msg in session.query(Message).all():
14     print(msg)
```


pymssql

```
1 import pymssql
```