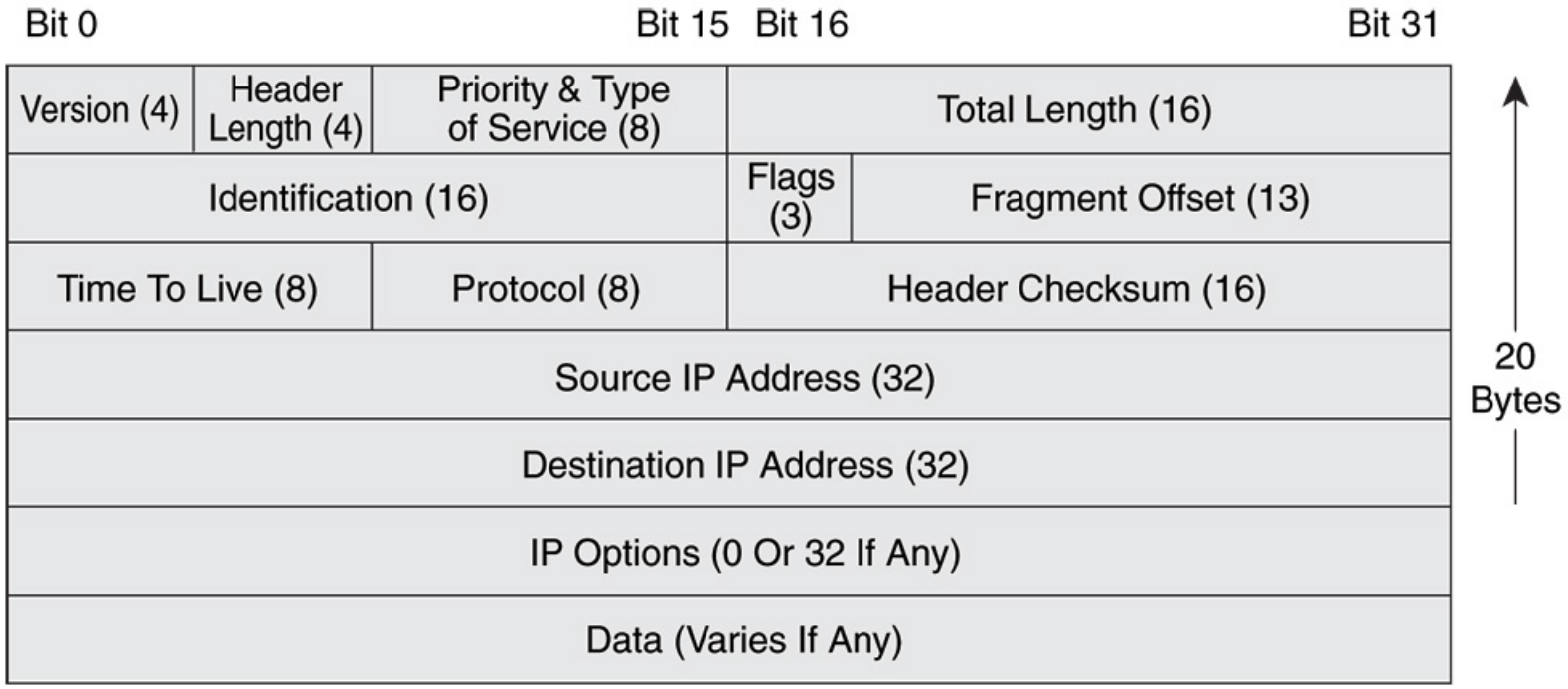
TCP level networking

Стек протоколов

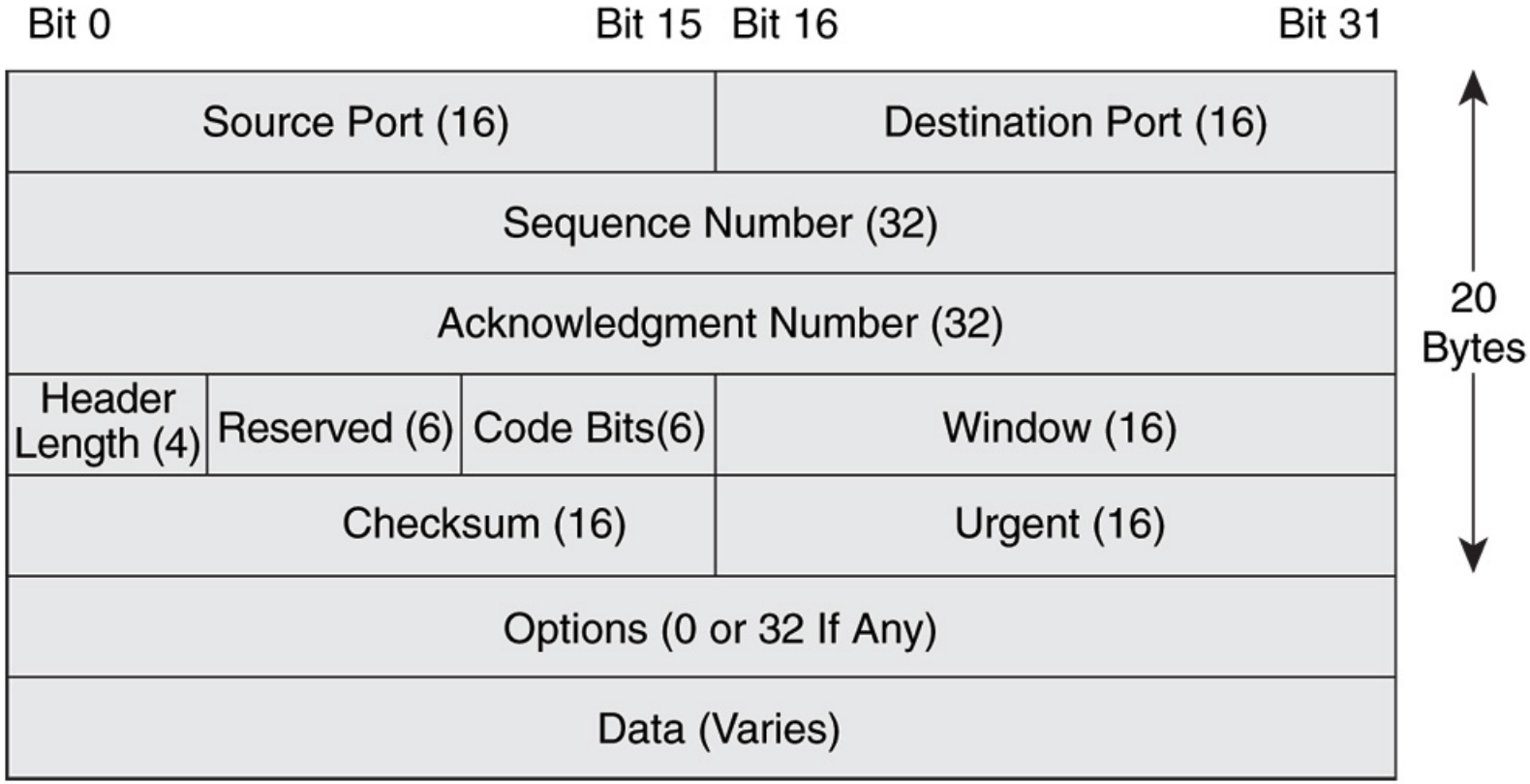
- Ethernet
- IP
- TCP/UDP
- HTTP/FTP/SSH/...

ISO vs. ANSI

IP


Bit 0
Bit 15
Bit 16
Bit 31
Version (4)
Header Length (4)
Priority & Type of Service (8)
Total Length (16)
Identification (16)
Flags (3)
Fragment Offset (13)
Time To Live (8)
Protocol (8)
Header Checksum (16)
Source IP Address (32)
Destination IP Address (32)
IP Options (0 Or 32 If Any)
Data (Varies If Any)
20 Bytes

TCP/UDP


Bit 0
Bit 15
Bit 16
Bit 31
Source Port (16)
Destination Port (16)
Sequence Number (32)
Acknowledgment Number (32)
Header Length (4)
Reserved (6)
Code Bits(6)
Window (16)
Checksum (16)
Urgent (16)
Options (0 or 32 If Any)
Data (Varies)
20 Bytes

Python TCP client

```python
import socket

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:  # tcp by default
    s.connect(('127.0.0.1', 12345))  # ip or hostname
    s.send("Hello, World!".encode('utf8'))

    # 1
    print("Recv", s.recv(1024).decode('utf8'))

    # 2
    # fd = s.makefile()
    # 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

```
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")
```

```
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())
```

```
import contextlib

@contextlib.contextmanager
def transaction(conn):
    cr = conn.cursor()
    try:
        yield cr
    except:
        conn.rollback()
        raise
    conn.commit()
```

```
with transaction(conn) as cr:
    cr.execute("insert into messages values (?, ?)", ("5", "this

with transaction(conn) as cr:
    show_db(cr)

with transaction(conn) as cr:
    cr.execute("insert into messages values (?, ?)", ("6", "this
    show_db(cr)
    raise ValueError("")

with transaction(conn) as cr:
    show_db(cr)
```

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

```
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)
```

```
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
```