СОКЕТИ

Ненко Табаков
Пламен Танов
Технологическо училище “Електронни системи”
Технически университет – София

21 октомври 2008

ЛИТЕРАТУРА НЕОБХОДИМИ ПРОГРАМИ

- SUN's Java Sockets Tutorial - http://java.sun.com/docs/books/tutorial/networking/sockets/
- Java API документация - http://java.sun.com/javase/6/docs/api/
- Eclipse - www.eclipse.org

СЪДЪРЖАНИЕ

ВЪВЕДЕНИЕ

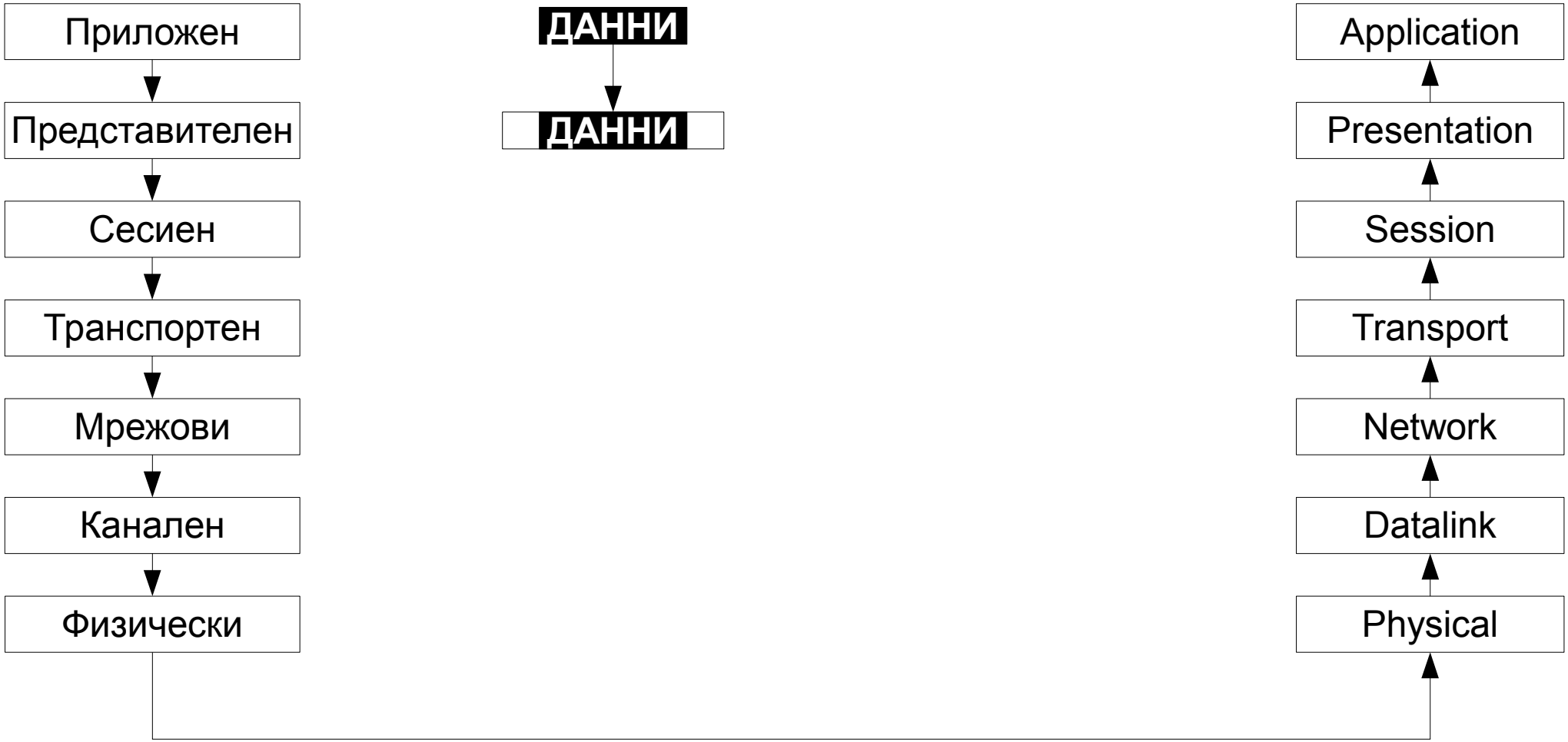
- Служи за връзка между две точки – клиент и сървър
- Сървърът предоставя услуга на клиента
- Връзката трябва да е надеждна
  - загубени пакети, ред на пристигане
- Връзката е двупосочна, едновременно може да се праща от едната и от другата страна
- Предоставя възможност на приложението да изпраща и получава данни
- Съхранява данни за двете страни (IP адрес, порт)
- Съществуват два типа услуги:
  - С установяване на сесия (connection-oriented)
  - Без установяване на сесия (connectionless)

OSI МОДЕЛ

- Абстрактен модел, описващ начина на комуникация в мрежа
- Позволява на еднородни и нееднородни системи да комуникират безпроблемно помежду си
- Описание на различните функции и операции, които трябва да се извършат от участниците при обмен на данни
- Логически са групирани в отделни слоеве, които съчетават близки по същност и замисъл операции, общо представяне на данните и относителна функционална независимост от другите слоеве.
- Броят на тези слоеве е 7 и логически са разположени един над друг
- Всеки слой предоставя интерфейс и услуги на този над него
- Всеки слой получава услуги от слоя под него
- Данните се променят при преминаване през слоевете

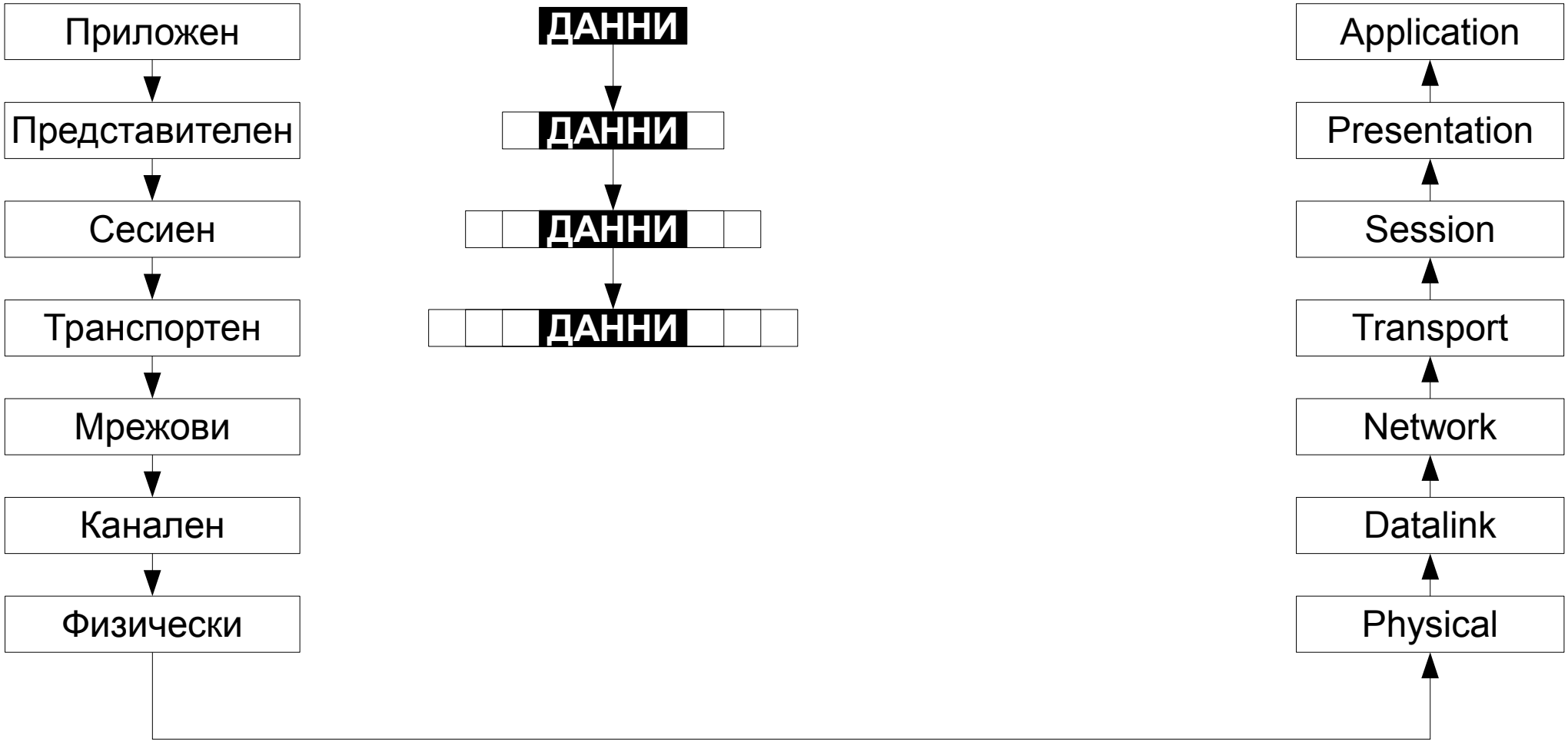
OSI МОДЕЛ СЛОЕВЕ (LAYERS)

OSI МОДЕЛ СЛОЕВЕ (LAYERS)

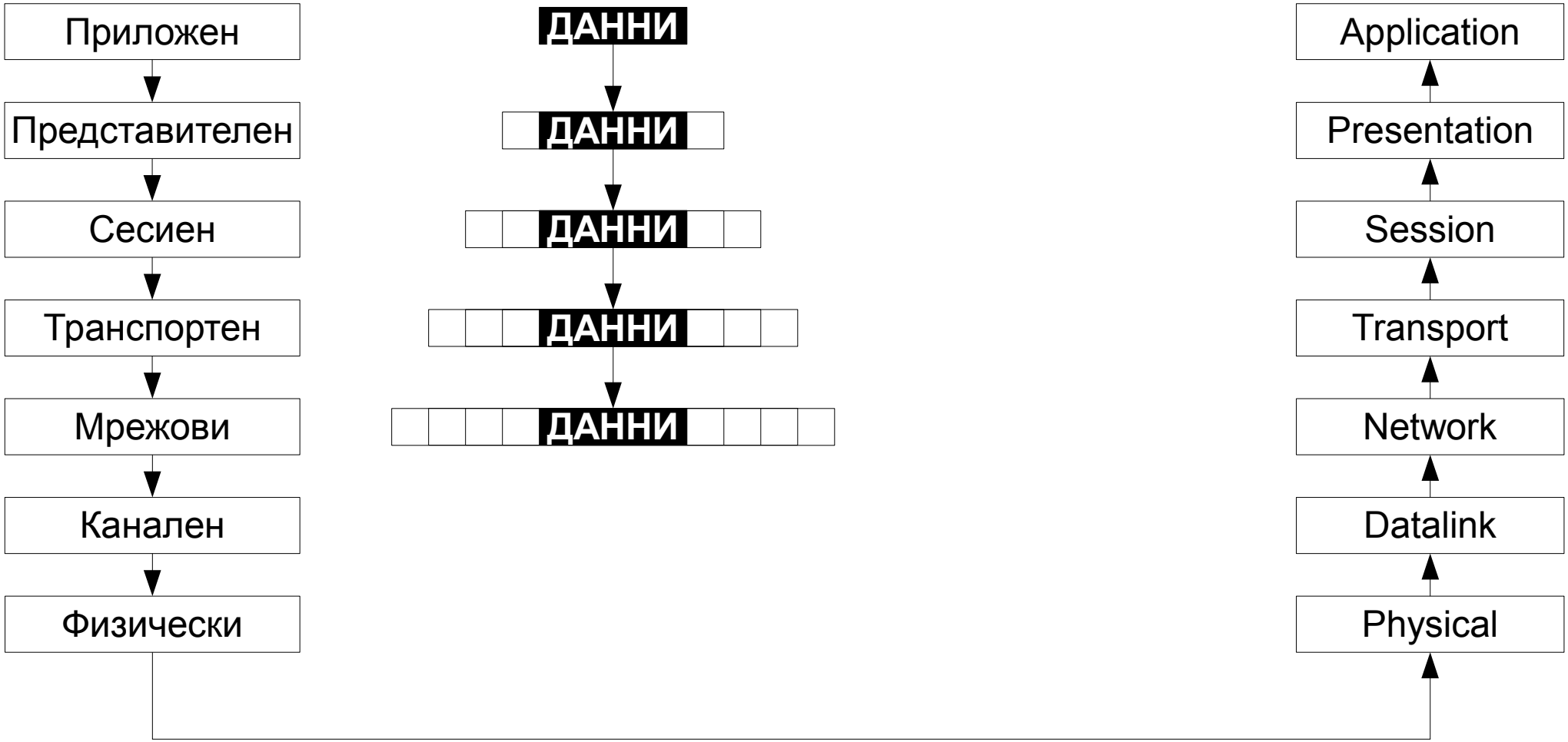


OSI МОДЕЛ СЛОЕВЕ (LAYERS)

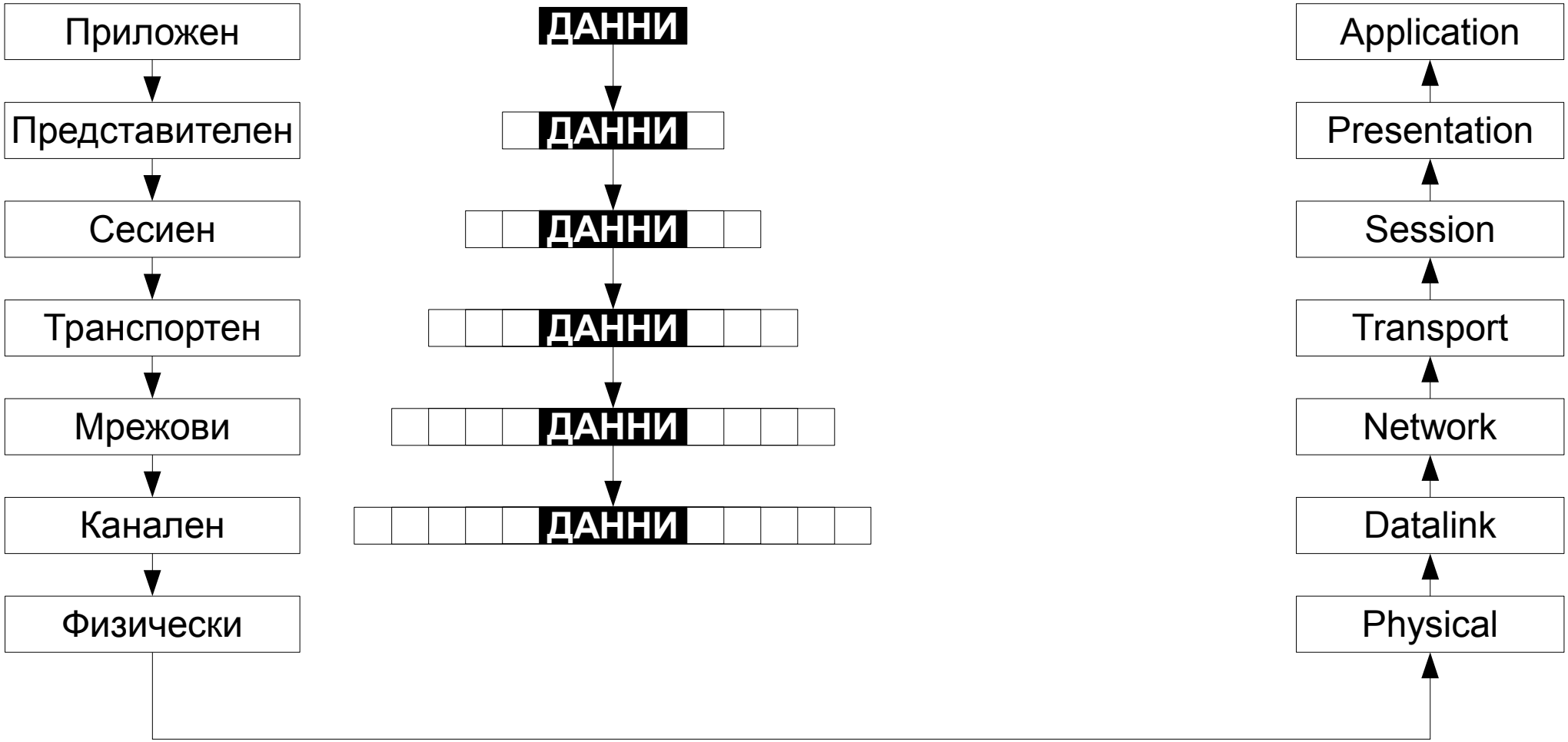
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



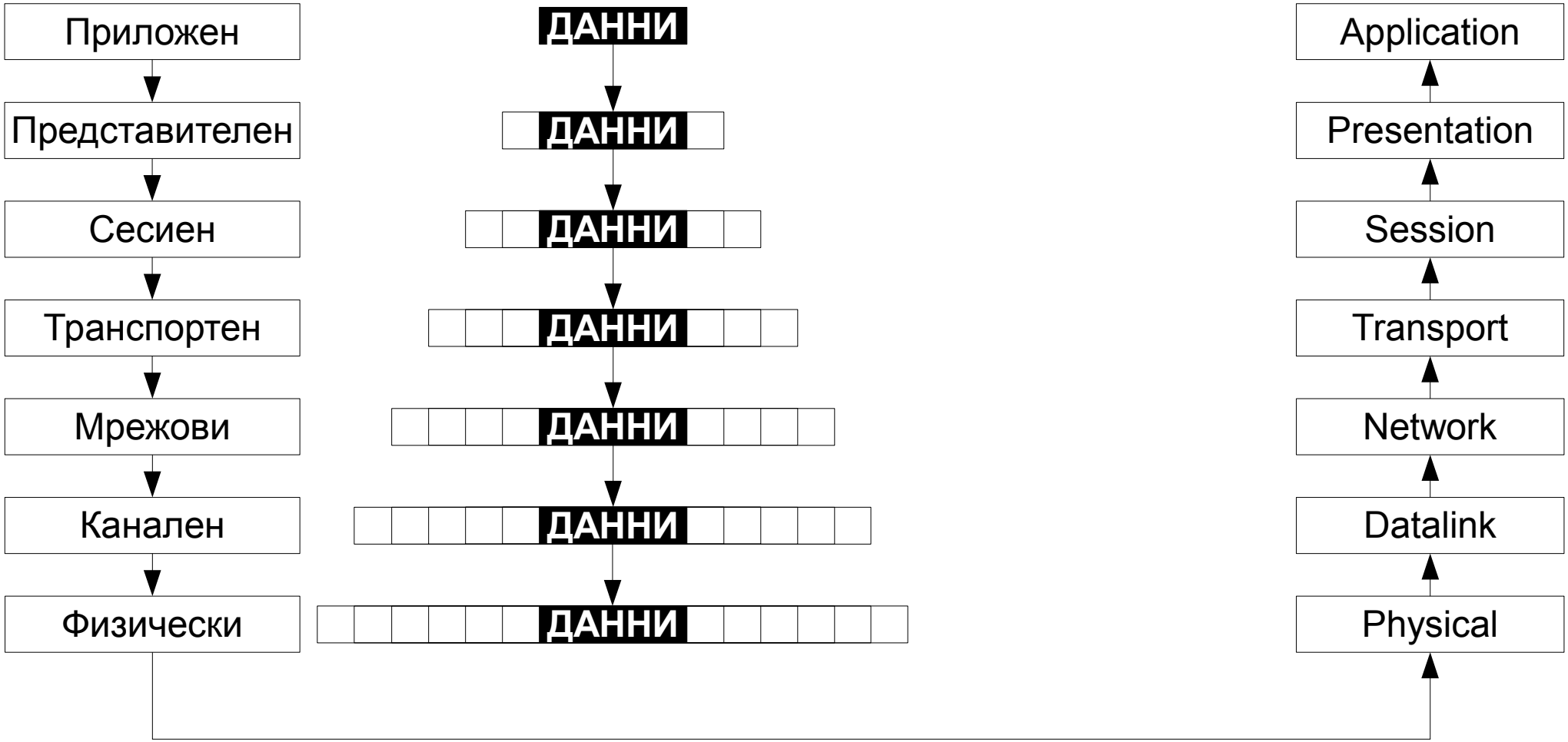
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



OSI МОДЕЛ СЛОЕВЕ (LAYERS)



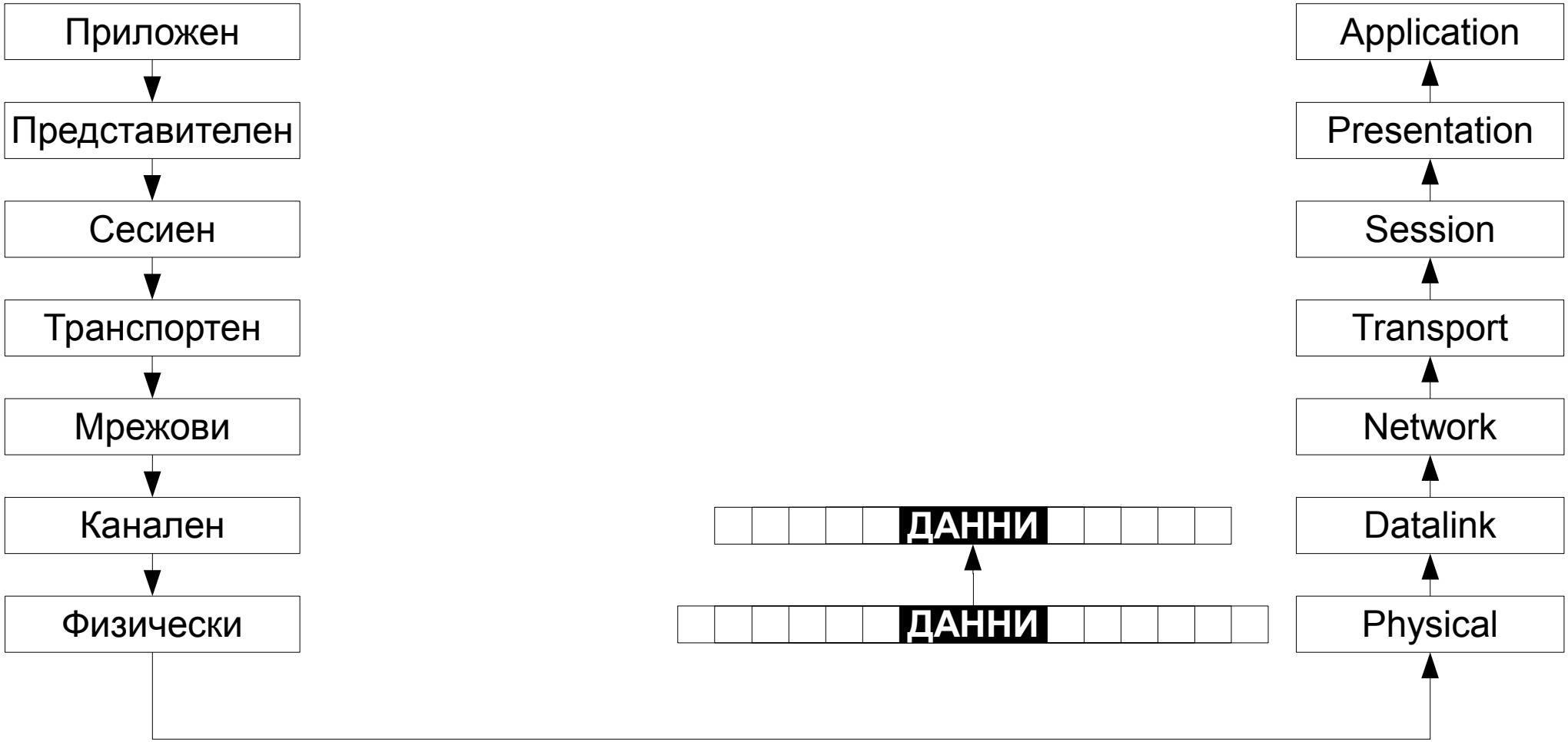
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



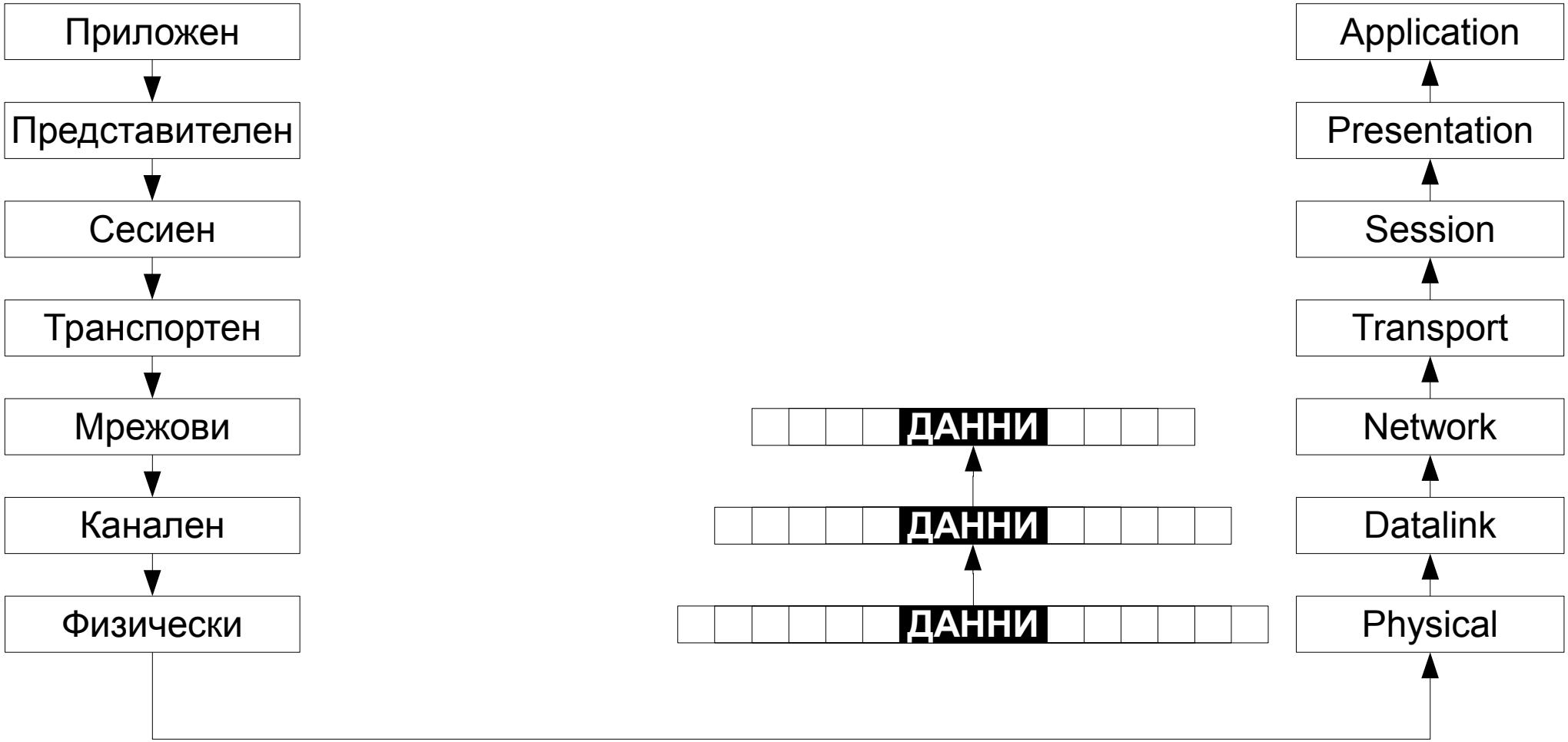
OSI МОДЕЛ СЛОЕВЕ (LAYERS)

OSI МОДЕЛ СЛОЕВЕ (LAYERS)

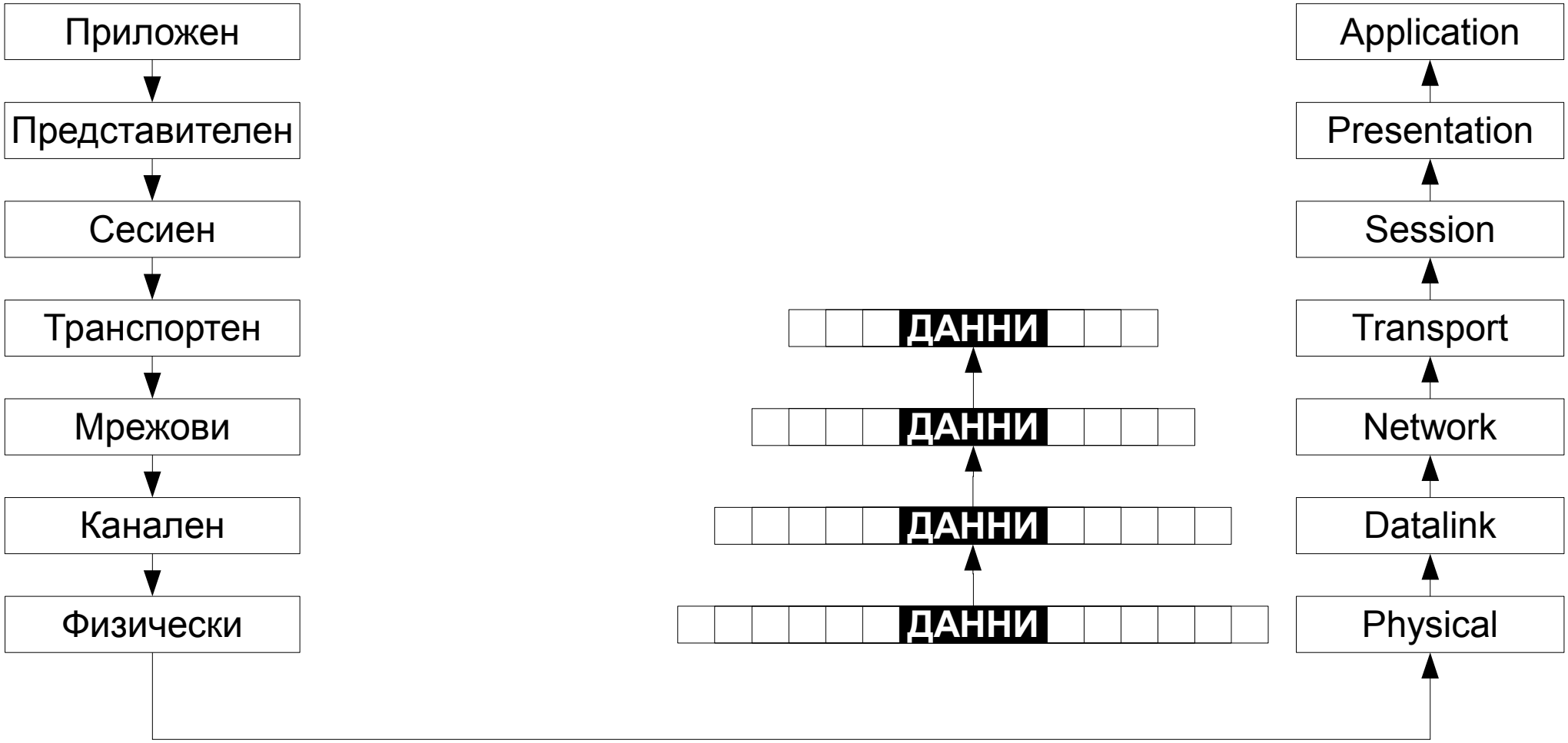
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



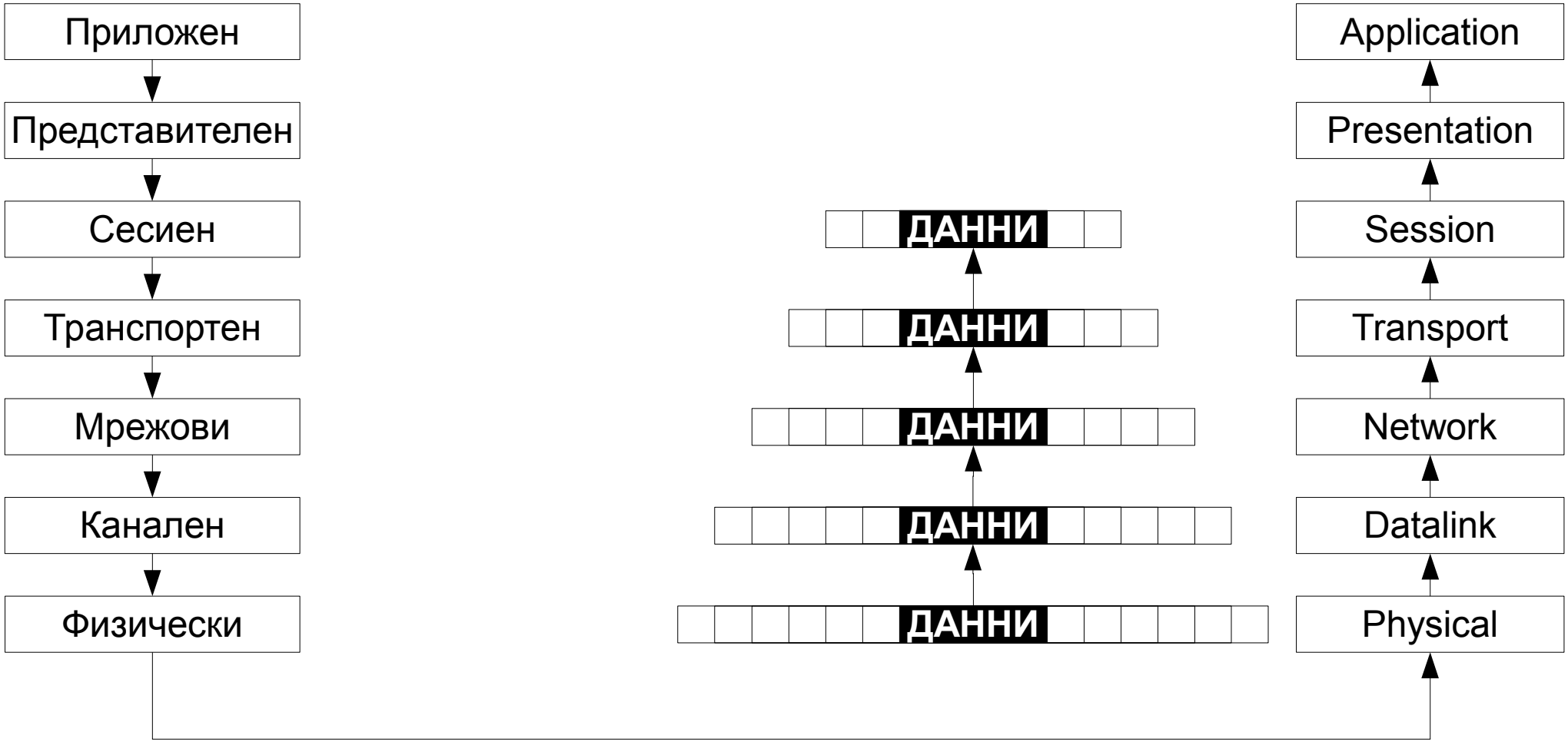
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



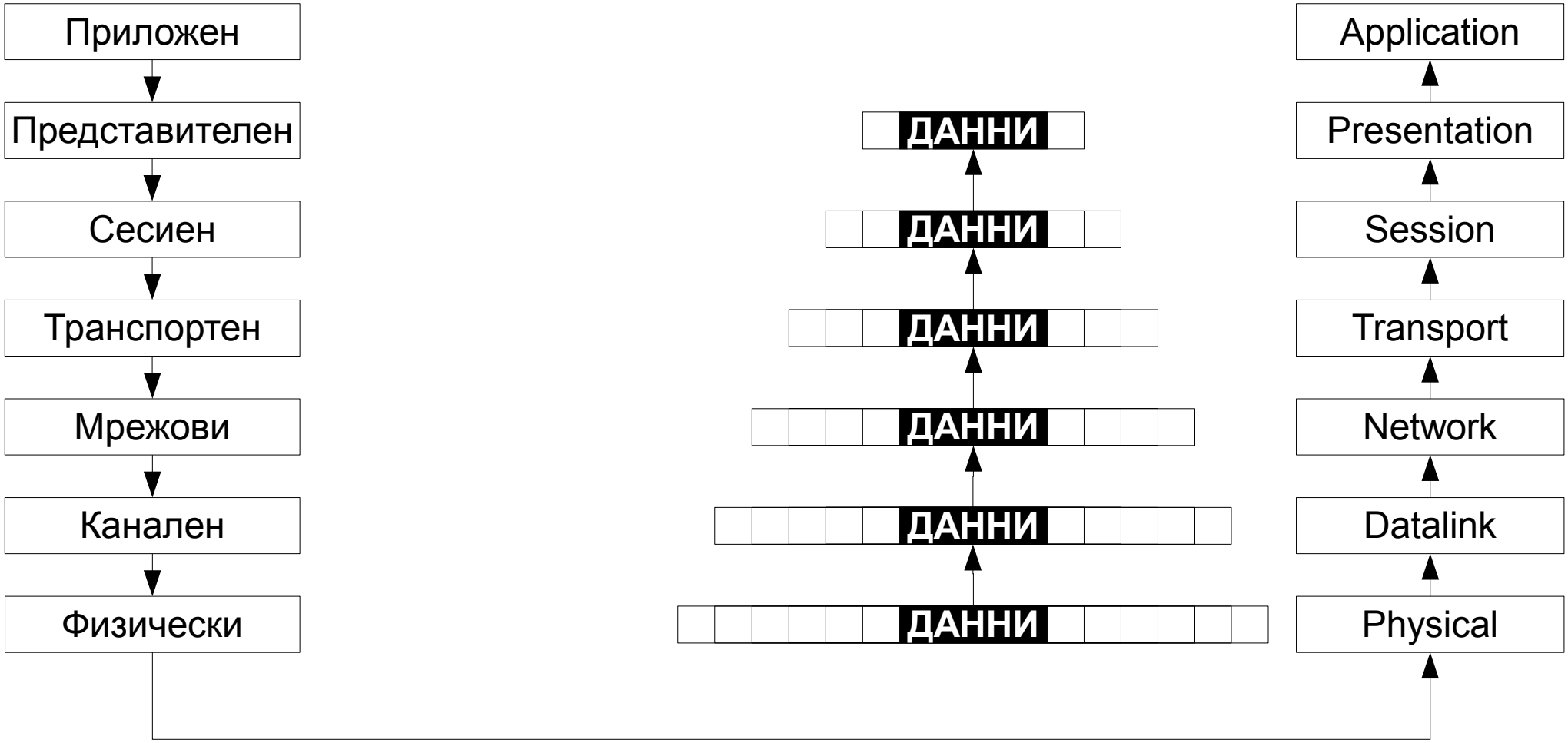
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



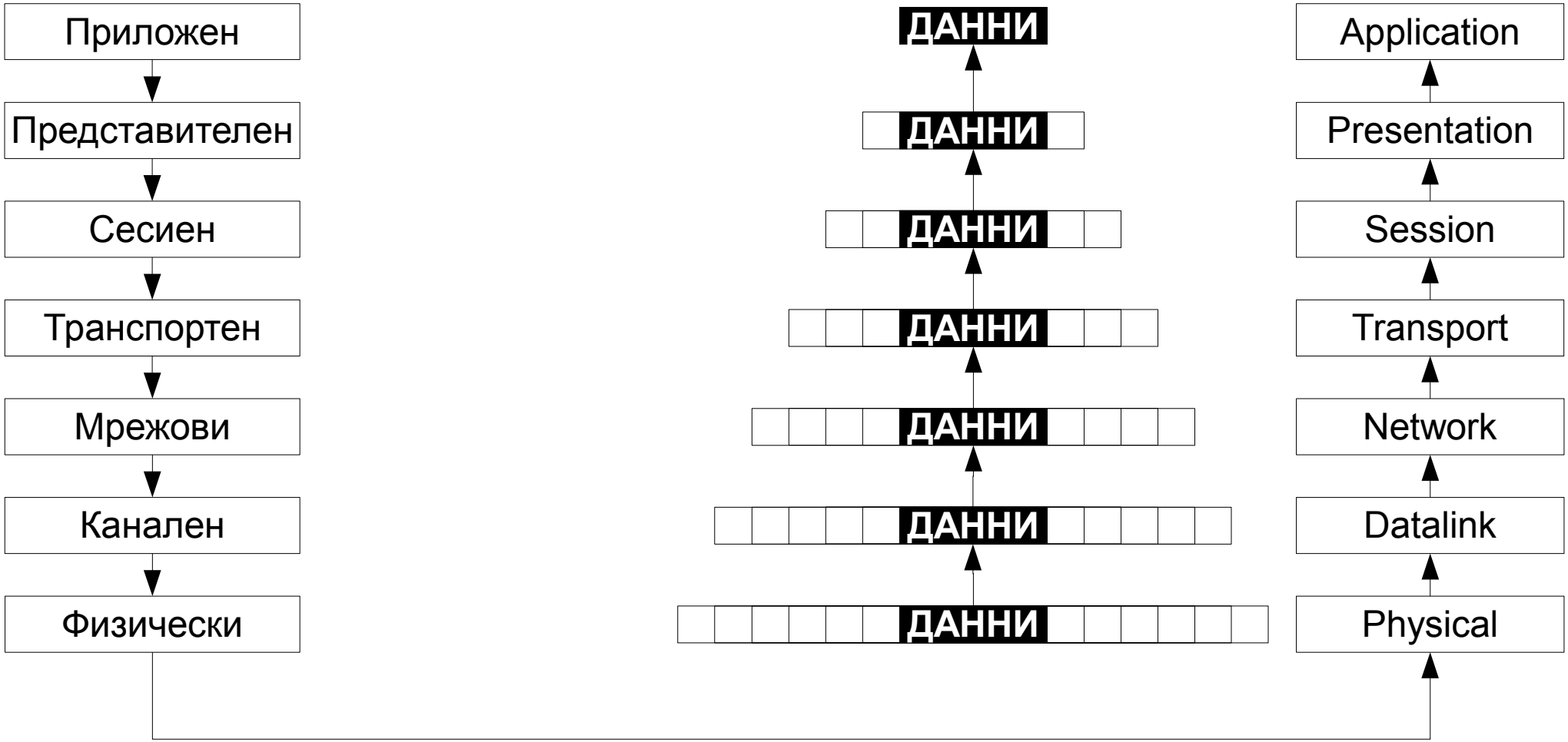
OSI МОДЕЛ СЛОЕВЕ (LAYERS)



OSI МОДЕЛ СЛОЕВЕ (LAYERS)



OSI МОДЕЛ СЛОЕВЕ (LAYERS)



OSI МОДЕЛ СЛОЕВЕ (LAYERS)

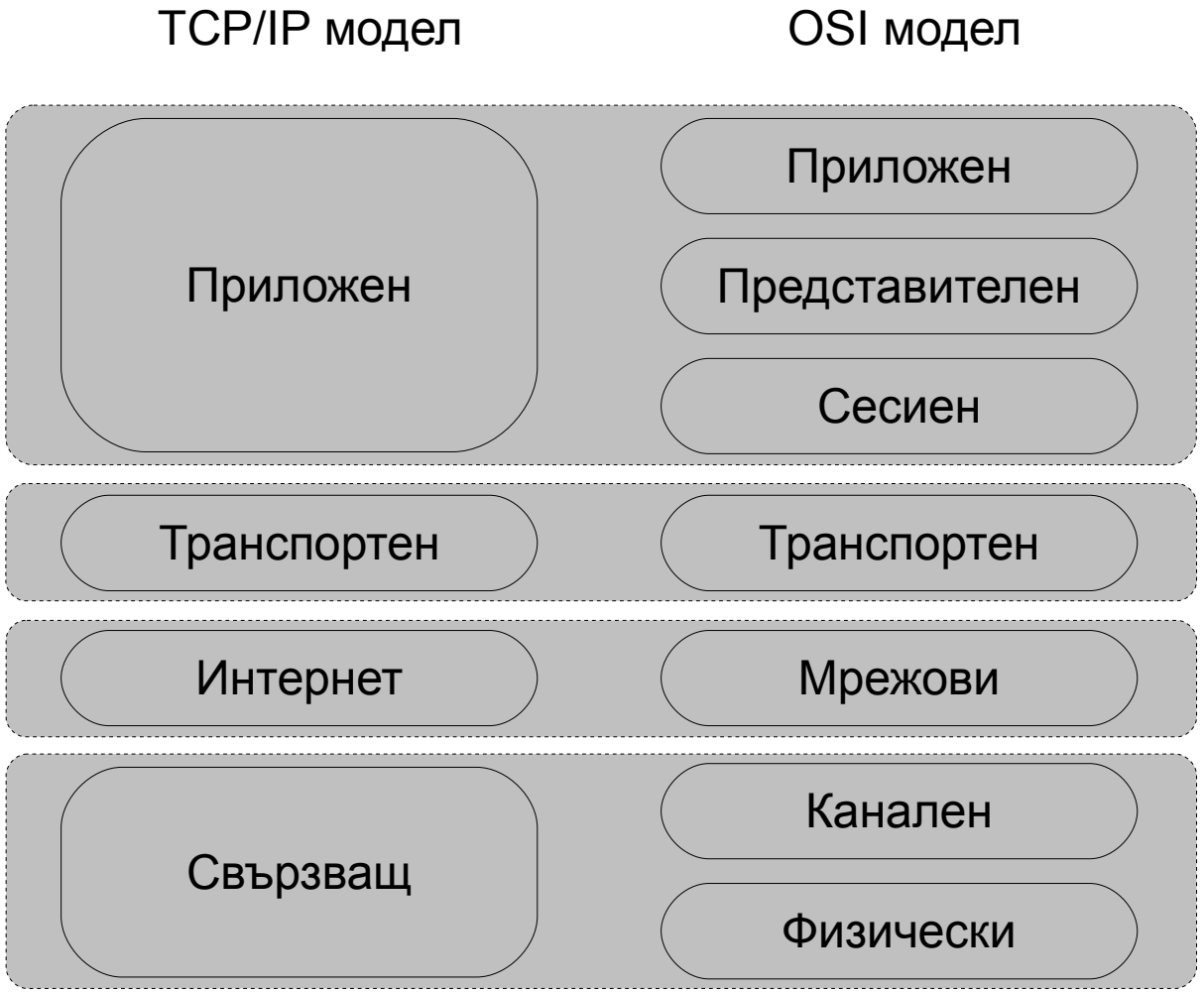
OSI МОДЕЛ СЛОЕВЕ (LAYERS)

- Приложен (Application) (приложения и услуги)
- Представителен (Presentation)
  - Представяне на данните, компресиране и криптиране
- Сесиен (Session)
  - Поддържане на сесия и точки на синхронизация
- Транспортен (Transport)
  - Комуникация от край до край между отдалечени процеси
- Мрежови (Network)
  - Определяне на оптимален път и логическо адресиране
- Канален (Datalink)
  - Физическо адресиране и контрол на достъп до средата
- Физически (Physical)
  - Предаване на битове до отдалечена система

TCP/IP (DARPA) МОДЕЛ

- Приложен слой (Application)
  - Директно достъпни услуги за потребителя
  - Telnet, FTP, SMTP, DNS, SNMP, NFS
- Транспортен слой (Transport)
  - Осигурява логическа връзка между отдалечените програми
  - TCP, UDP
- Интернет слой (Internet)
  - Базова комуникация, адресиране и намиране на път
  - IP, IGMP, ICMP, ARP
- Свързващ слой (Link, Network Interface Layer)
  - Физическа връзка и организация на мрежата
  - Ethernet, Token Ring, Frame Relay, ATM

СЪПОСТАВЯНЕ НА МОДЕЛИТЕ

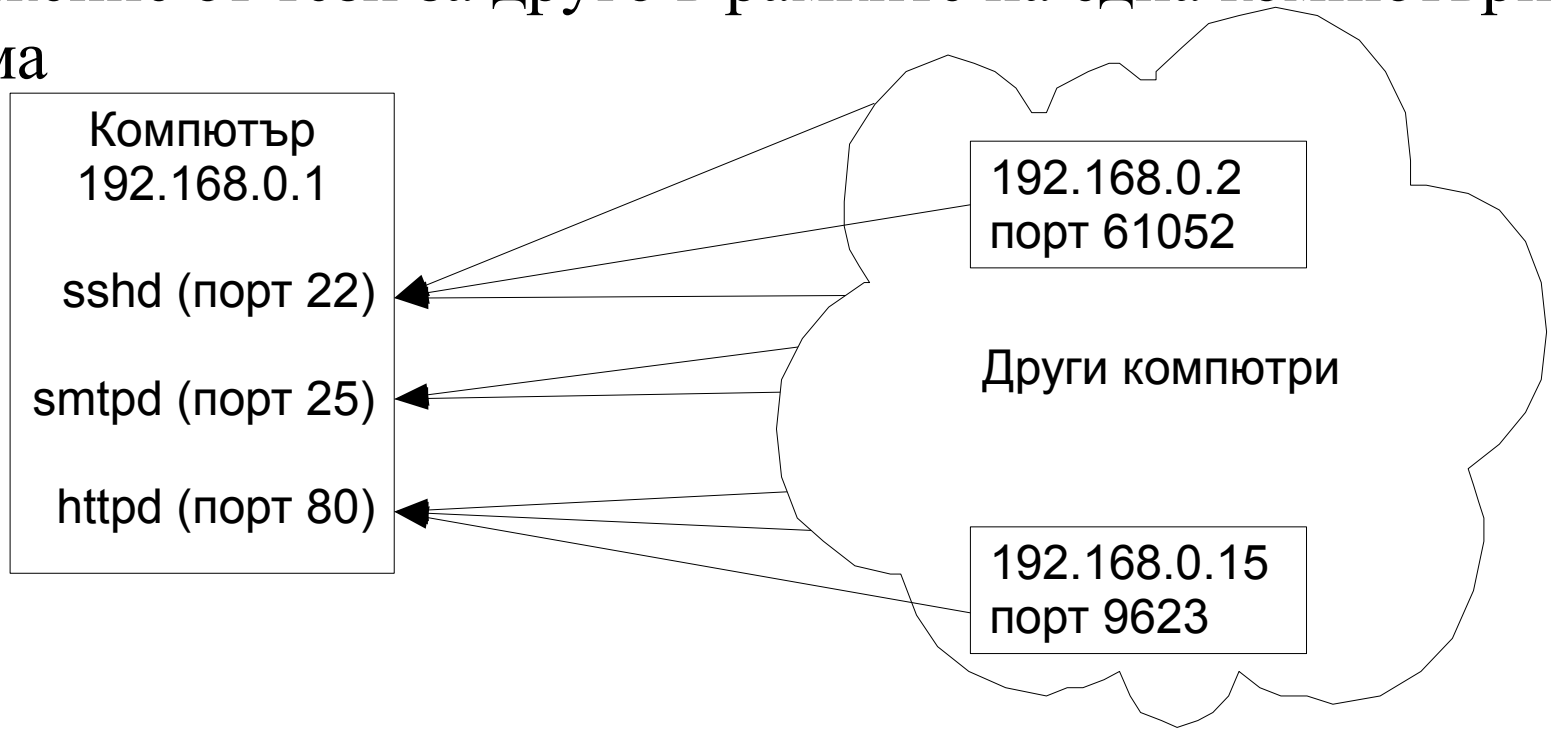


IP АДРЕС

- Уникален номер на компютър в мрежа (мрежови слой)
- Позволява на компютрите, които предават информация да укажат къде да бъде пратена
- Позволява на компютрите, които получават информация да знаят от къде е била изпратена
- Използват се две версии на IP адреса:
  - IPv4
    - 32 битово число
    - Изчерпване
    - Пример: 72.5.124.61
  - IPv6
    - 128 битово число
    - 2001:0db8:85a3:0000:0000:8a2e:0370:7334

ПОРТ

- Множество приложения на един компютър могат едновременно да пращат и получават данни през TCP/UDP
- Портът служи за да може да се различават пакетите за дадено приложение от тези за друго в рамките на една компютърна система



ПОРТ

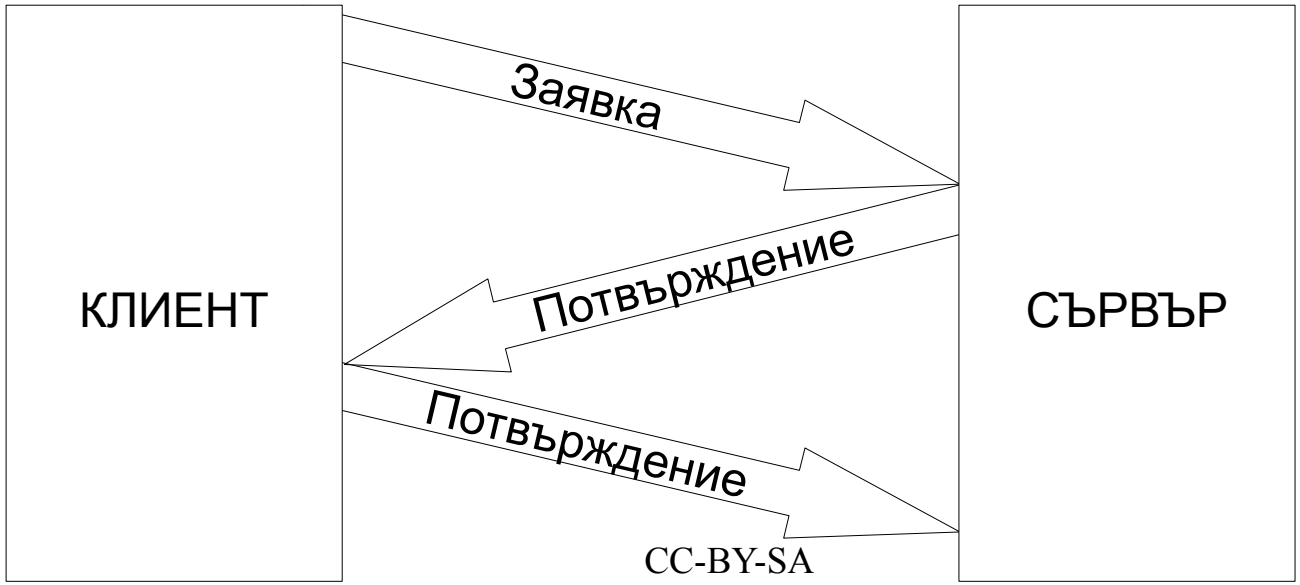
- Множество приложения на един компютър могат едновременно да пращат и получават данни през TCP/UDP
- Портът служи за да може да се различават пакетите за дадено приложение от тези за друго в рамките на една компютърна система
- Различават се посредством 16 битово число, наречено номер на порт (от 0 до 65535):
  - от 0 до 1023 – добре известни портове (well known ports)
    - супер-потребителски (*root*) права за отварянето им
  - от 1024 до 49151 – регистрирани портове (registered ports)
  - от 49152 до 65535 – динамични портове (dynamic ports)
- Едно приложение може да ползва повече от един порт за изпращане и получаване на данни

УСЛУГА БЕЗ УСТАНОВЯВАНЕ НА СЕСИЯ

- Реализира се от ***UDP*** (**U**ser **D**atagram **P**rotocol)
- По-бърза комуникация от TCP, но несигурна
- Не е необходимо да се установи връзка преди пращане/получаване на данни (връзка не се установява!)
- Не се гарантира (приложението само трябва да се грижи):
  - получаването на данните (цели пакети могат да бъдат изгубени)
  - реда на получаване на данните (пакетите могат да бъдат получавани в произволен ред)

УСЛУГА С УСТАНОВЯВАНЕ НА СЕСИЯ

- Реализира се от ***TCP*** (**T**ransmission **C**ontrol **P**rotocol)
- Преди да могат да се предават данни е необходимо да бъде установена връзка (т. нар. three-way handshake):
  - Заявка за осъществяване на връзка (***SYN***)
  - Потвърждение на заявката (***SYN-ACK***)
  - Потвърждение за осъществяване на връзка (***ACK***)



УСЛУГА С УСТАНОВЯВАНЕ НА СЕСИЯ

- Реализира се от ***TCP*** (**T**ransmission **C**ontrol **P**rotocol)
- Преди да могат да се предават данни е необходимо да бъде установена връзка (three-way handshake):
  - Заявка за осъществяване на връзка (***SYN***)
  - Потвърждение на заявката (***SYN-ACK***)
  - Потвърждение за осъществяване на връзка (***ACK***)
- Гарантира се:
  - получаването на данните (не се губят пакети)
  - реда на получаване на данните
- При приключване на работа връзката се прекратява

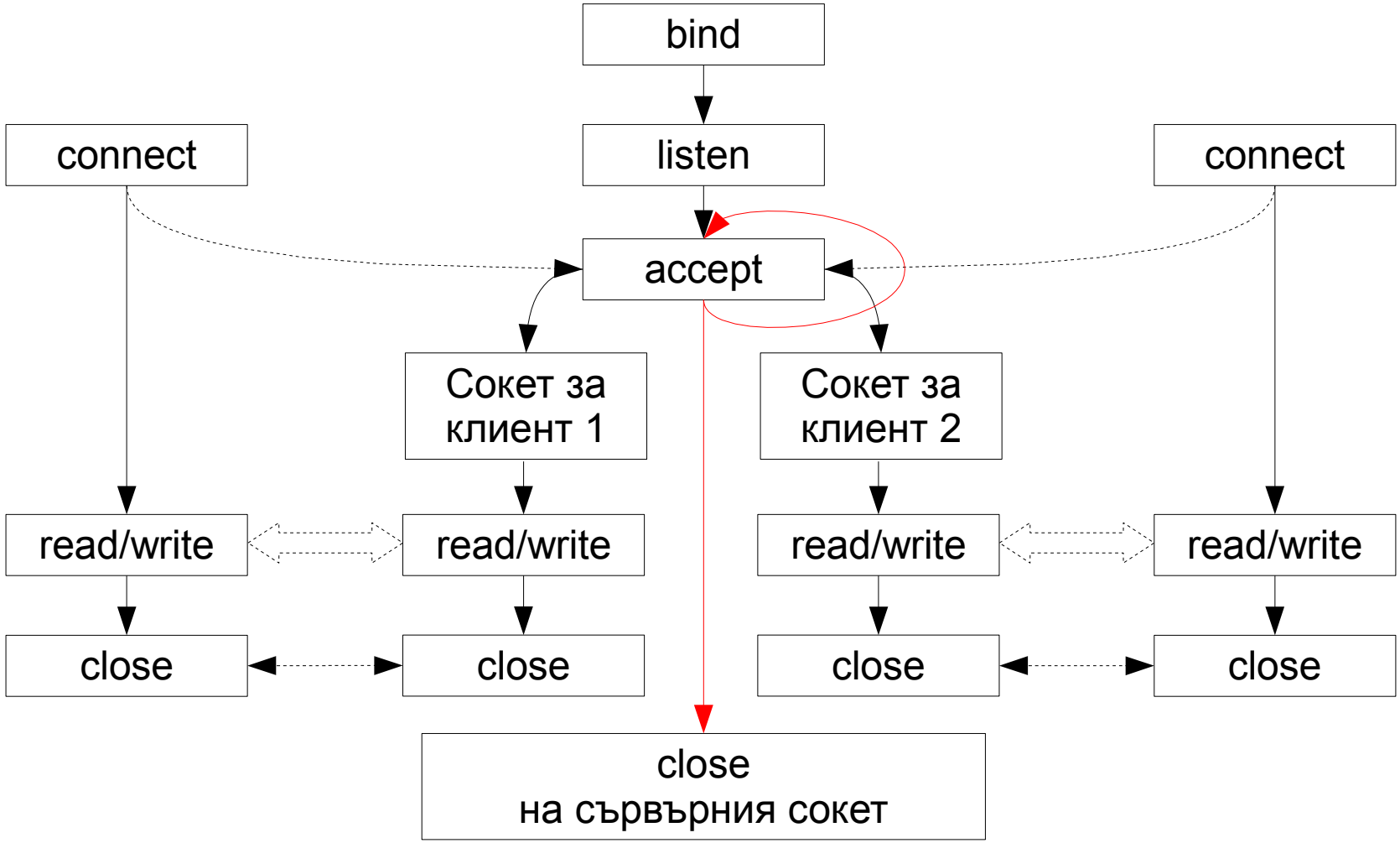
ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА КЛИЕНТ

- Създаване на ***TCP*** сокет
- Установяване на връзка (connect)
- Обмен на данни (read/write)
- Затваряне на връзка (close)

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Създаване на ***TCP*** сървърен сокет
- Свързване на сървърния сокета с порт (bind)
- Отваряне на сървърния сокета (listen)
- За всеки опит за връзка:
  - Приемане на връзката – създаване на нов сокет за нея (accept)
  - Обмен на данни, чрез него (read/write)
  - Затваряне на конкретната връзка/сокет (close)
- Затваряне на сървърния сокет

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА



ПОТОЦИ (*STREAM*) в *JAVA*

- http://java.sun.com/javase/6/docs/api/java/io/InputStream.html
- http://java.sun.com/javase/6/docs/api/java/io/OutputStream.html
- И техните наследници

- Предоставят методи за четене и запис на байтове:
  - `int read();`
  - `int read(byte[] b);`
  - `int read(byte[] b, int off, int len);`
  - `void write(byte[] b);`
  - `void write(byte[] b, int off, int len);`
  - `void write(int b);`
  - и други
  - могат да бъдат буферирани (`flush()`)
- Изключение от тип `IOException` при входно-изходна грешка

ПОТОЦИ (*STREAM*) в *JAVA*

- http://java.sun.com/javase/6/docs/api/java/io/DataInputStream.html
- http://java.sun.com/javase/6/docs/api/java/io/DataOutputStream.html
- И техните наследници

- Предоставят методи за четене и запис и на примитивни типове:
  - `char readChar();`
  - `int readInt();`
  - `double readDouble();`
  - `void writeChar(int v);`
  - `void writeInt(int v);`
  - `void writeDouble(double v);`
  - и други
  - могат да бъдат буферирани (`flush()`)
- Изключение от тип `IOException` при входно-изходна грешка

РАБОТА С ПОТОЦИ

- http://java.sun.com/javase/6/docs/api/java/io/Reader.html
- http://java.sun.com/javase/6/docs/api/java/io/Writer.html
- И техните наследници

- Предоставят методи за четене и запис на форматирани данни:
  - Преобразуване от байтове към символи и обратно по зададена кодова таблица (charset) (InputStreamReader, OutputStreamWriter)
  - Буфериране (BufferedReader)
    - Подобрява ефективността
    - Възможност за прочитане на цели редове
  - Форматиран текстов изход (PrintWriter)
  - и други
- Изключение от тип `IOException` при входно-изходна грешка

ПРИМЕР
Работа със стандартния вход и изход

```
import  java.io.BufferedReader;
import  java.io.IOException;
import  java.io.InputStreamReader;
import  java.io.PrintWriter;
public  class Streams {
   public static void main(String[] args) throws IOException {
      BufferedReader stdIn = new BufferedReader(new
                                 InputStreamReader(System.in));

      System.out.print("Please enter some text: ");
      String line = stdIn.readLine();

      PrintWriter stdOut = new PrintWriter(System.out, true);
      stdOut.printf("You have entered %s,\nsome formated
                           output: %15.2f %10d\n", line, 1.1, 5);
   }
}
```

СОКЕТИ в *JAVA*

- Услуга без установяване на връзка (***UDP***)
  - http://java.sun.com/javase/6/docs/api/java/net/DatagramSocket.html
  - http://java.sun.com/javase/6/docs/api/java/net/DatagramPacket.html
- Услуга с установяване на връзка (***TCP***)
  - http://java.sun.com/javase/6/docs/api/java/net/Socket.html
  - http://java.sun.com/javase/6/docs/api/java/net/ServerSocket.html

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА КЛИЕНТ

- Създаване на ***TCP*** сокет
- Установяване на връзка

```
Socket echoSocket = new Socket(HOST, PORT);
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА КЛИЕНТ

- Създаване на ***TCP*** сокет
- Установяване на връзка
- Обмен на данни

```
Socket echoSocket = new Socket(HOST, PORT);
OutputStream outStream = echoSocket.getOutputStream();
InputStream inStream = echoSocket.getInputStream();
// String TEXT = "Hello world!\n";
outStream.write(TEXT.getBytes());
outStream.flush();

byte[] buffer = new byte[TEXT.length()];
inStream.read(buffer);
// do something...
System.out.println("echo: " + new String(buffer));
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА КЛИЕНТ

- Създаване на ***TCP*** сокет
- Установяване на връзка
- Обмен на данни
- Затваряне на връзката

```
Socket echoSocket = new Socket(HOST, PORT);
OutputStream outStream = echoSocket.getOutputStream();
InputStream inStream = echoSocket.getInputStream();
// String TEXT = "Hello world!\n";
outStream.write(TEXT.getBytes());
outStream.flush();

byte[] buffer = new byte[TEXT.length()];
inStream.read(buffer);
// do something...
System.out.println("echo: " + new String(buffer));

echoSocket.close();
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Създаване на ***TCP*** сървърен сокет
- Свързване на сървърния сокета с порт (bind)
- Отваряне на сървърния сокета (listen)
- За всеки опит за връзка:
  - Приемане на връзката – създаване на нов сокет за нея (accept)
  - Обмен на данни, чрез него (read/write)
  - Затваряне на конкретната връзка/сокет (close)
- Затваряне на сървърния сокет

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Създаване на ***TCP*** сървърен сокет
- Свързване на сървърния сокета с порт (bind)
- Отваряне на сървърния сокета (listen)

```
// create socket
ServerSocket serverSocket = new ServerSocket(PORT);
System.out.println("Started server on port " + PORT);
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Приемане на връзката (accept)

```
// a "blocking" call which waits until a connection is requested
Socket clientSocket = serverSocket.accept();
System.out.println("Accepted connection from client: "
                                                + clientSocket);
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Обмен на данни (read/write)

```
// open up IO streams
InputStream inStream = clientSocket.getInputStream();
OutputStream outStream = clientSocket.getOutputStream();

BufferedReader in = new BufferedReader(new InputStreamReader(
                                                    inStream));
PrintWriter out = new PrintWriter(outStream, true);

// waits for data and reads it in until connection dies
// readLine() blocks until the server receives a new line from
// client
String s;
while ((s = in.readLine()) != null) {
    out.println(s);
}
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Затваряне на конкретната връзка (close)

```
// close IO streams, then socket
System.out.println("Closing connection with client");
out.close();
in.close();
clientSocket.close();
```

ПОСЛЕДОВАТЕЛНОСТ ПРИ РАБОТА НА СЪРВЪР

- Затваряне на сървърния сокет

```
serverSocket.close();
```