F8

Meddelandesändning med UDP

EDA0965 Nätverksprogrammering

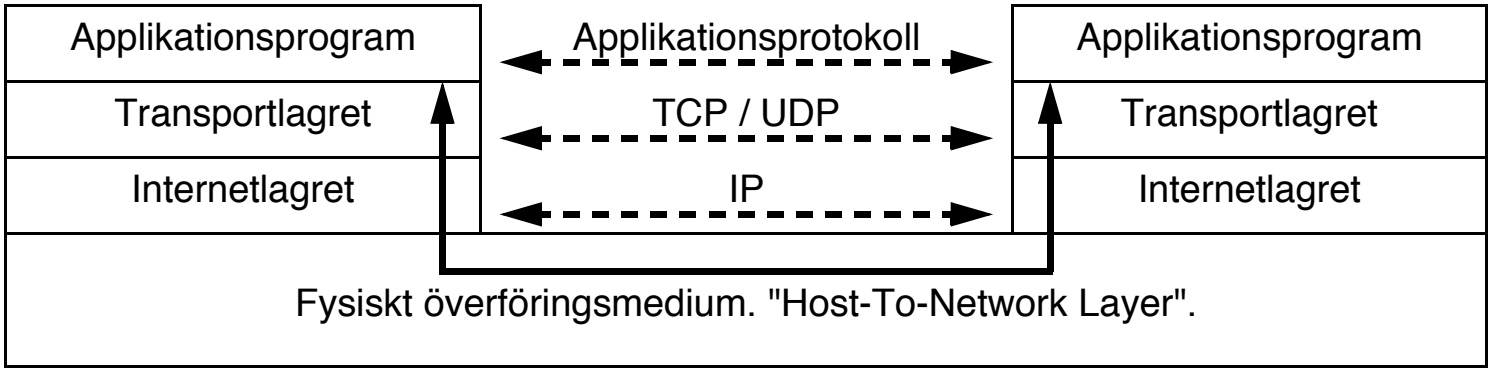
Per Andersson

Datavetenskap

Lunds universitet

Transport Layer

Bygger vidare på “Internet Layer” / IP.



TCP/UDP

- **Transmission Control Protocol (TCP)**
  - Fast (logisk) uppkoppling över nätverket.
  - Dataström – data behöver inte delas upp i paket.
  - Automatisk felkontroll/omsändning. Garanterad leverans och inbördes ordning.
- **User Datagram Protocol (UDP)**
  - Datagram
  - Upp till 65507 byte stora datagram (IPv4)

Java och UDP

I paketen java.net och java.io finns stöd för att sända och ta emot meddelanden (paket) mha UDP.

DatagramPacket

Representerar ett meddelande som kan skickas med UDP.

DatagramSocket

Fungerar som en sändare/mottagare för meddelanden.

“socket” = uttag / ”hål-i-väggen”. Jämför: telefonjack/fax.

DatagramPacket

Konstruktorer

DatagramPacket-objekt som ska användas för att *ta emot* meddelanden:

```
public DatagramPacket(byte[] buffer, int length);
```

DatagramPacket-objekt som ska användas för att sända meddelanden:

```
public DatagramPacket(byte[] buffer, int length,
                      InetAddress destination, int port);
```

Vi måste tillhandahålla en vektor av typen byte[] som är tillräckligt stor för att rymma det meddelande som ska sändas/tas emot.

Ett meddelande består alltså av ett antal *bytes*.

DatagramPacket, fortsättning

Get-metoder

```
public InetAddress getAddress();

public int getPort();

public byte[] getData();
public int getLength();
```

Set-metoder

```
public void setData(byte[] data);

public void setAddress(InetAddress remote);

public void setPort(int port);
public void setLength(int length);
```

DatagramPacket, exempel

Skapa ett DatagramPacket-objekt avsett att skickas till port 2000 på login.cs.lth.se och som innehåller texten “Network Programming”:

```
// Create an InetAddress object

InetAddress dest = null;
try {
   dest = InetAddress.getByName("login.cs.lth.se");

} catch(UnknownHostException e) { System.exit(1); }

// Create message buffer
String s = "Network Programming";

byte[] data = s.getBytes(); // Default character encoding

// Create the DatagramPacket object
DatagramPacket packet =

    new DatagramPacket(data,data.length,dest,2000);
```

DatagramSocket

Klassen DatagramSocket ansluts till en port och kan:

- Sända UDP-paket från denna port till en annan port på en annan dator.
- Ta emot meddelanden som sänds till denna port på denna dator.

Jämför med en fax (DatagramSocket) som ansluts till ett telefonjack (porten).

Datagramsocket, fortsättning

Konstruktorer

Skapa en DatagramSocket och anslut den till angiven port:

```
public DatagramSocket(int port) throws SocketException;
```

Skapa en DatagramSocket på en för tillfället ledig port:

```
public DatagramSocket() throws SocketException;
```

Konstruktorerna genererar ett SocketException om det inte gick att skapa socketen, t.ex. därför att angiven port var upptagen.

DatagramSocket, fortsättning

Skicka ett datagram

```
public void send(DatagramPacket dp) throws IOException;
```

Glöm inte att skapa ett DatagramPacket med tillhörande byte-vektor först!

Glöm inte heller att fylla i byte-vektorn med ditt meddelande!

Ta emot ett datagram

```
public void receive(DatagramPacket dp) throws IOException;
```

Anrop av receive blockerar tills ett meddelande anländer.

Glöm inte att skapa ett tomt DatagramPacket med en tillhörande byte-vektor som är tillräckligt stor för att rymma meddelandet!

Vid både mottagning och sändning genereras ett IOException om det uppstod

DatagramSocket, fortsättning

Frigöra portar

När man inte har behov av en port längre skall man frigöra den så att den kan återanvändas till andra ändamål. Detta görs genom att man anropar "close" på motsvarande DatagramSocket-objekt:

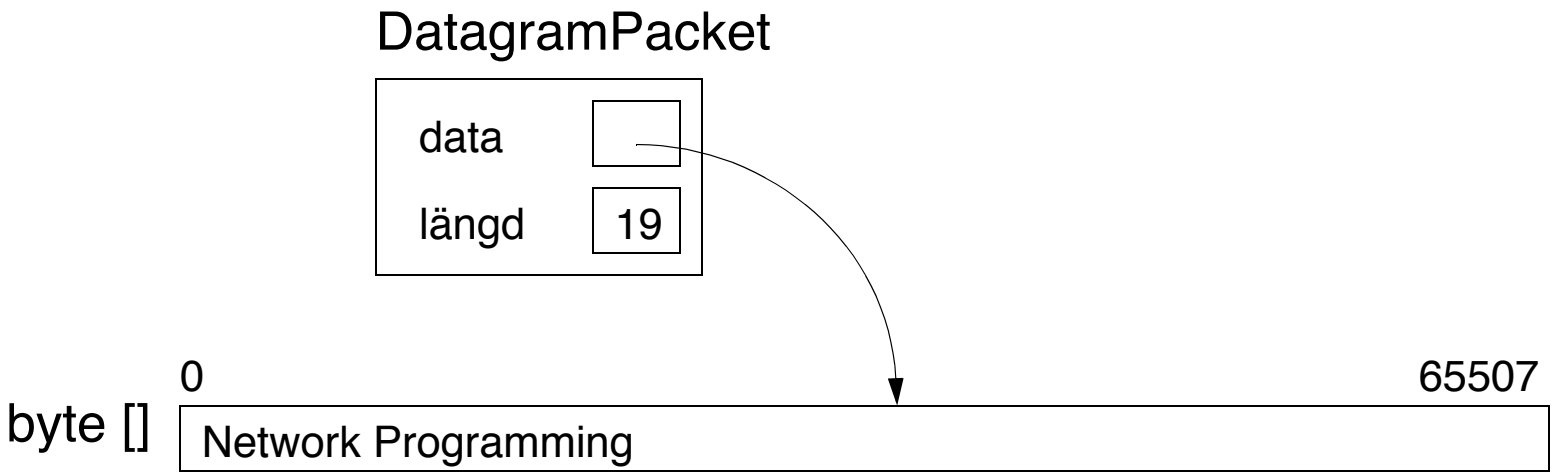
```
public void close();
```

Extra inställningar

Anrop av receive blockerar ända tills ett meddelande anländer. Vill man att receive ska vänta på ett meddelande högst en viss tid kan man sätta en timeout för förbindelsen. Har inget meddelande mottagits inom angiven tid (i millisekunder) genereras ett InterruptedIOException.

```
public void setSoTimeout(int timeout)
                                    throws SocketException;
public int getSoTimeout() throws IOException;
```

Längd för DatagramPacket



Omvandla mottaget meddelande till en sträng:

```
String s = new String(dp.getData(),0,dp.getLength());
```

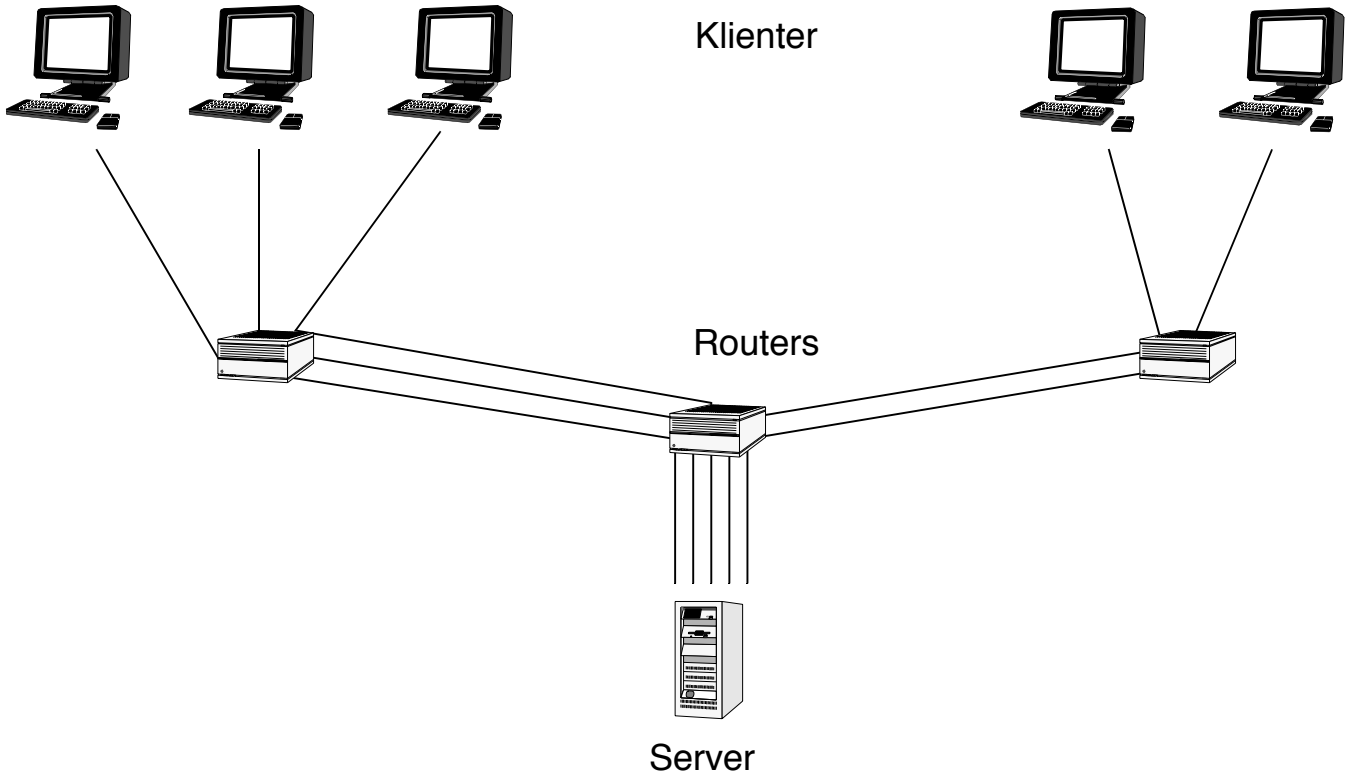
Längden i ett DatagramPacket styr hur stort meddelande som kan tas emot.
Kom ihåg att återställa om du vill återanvända ett DatagramPacket:

```
dp.setLength(dp.getData().length);
```

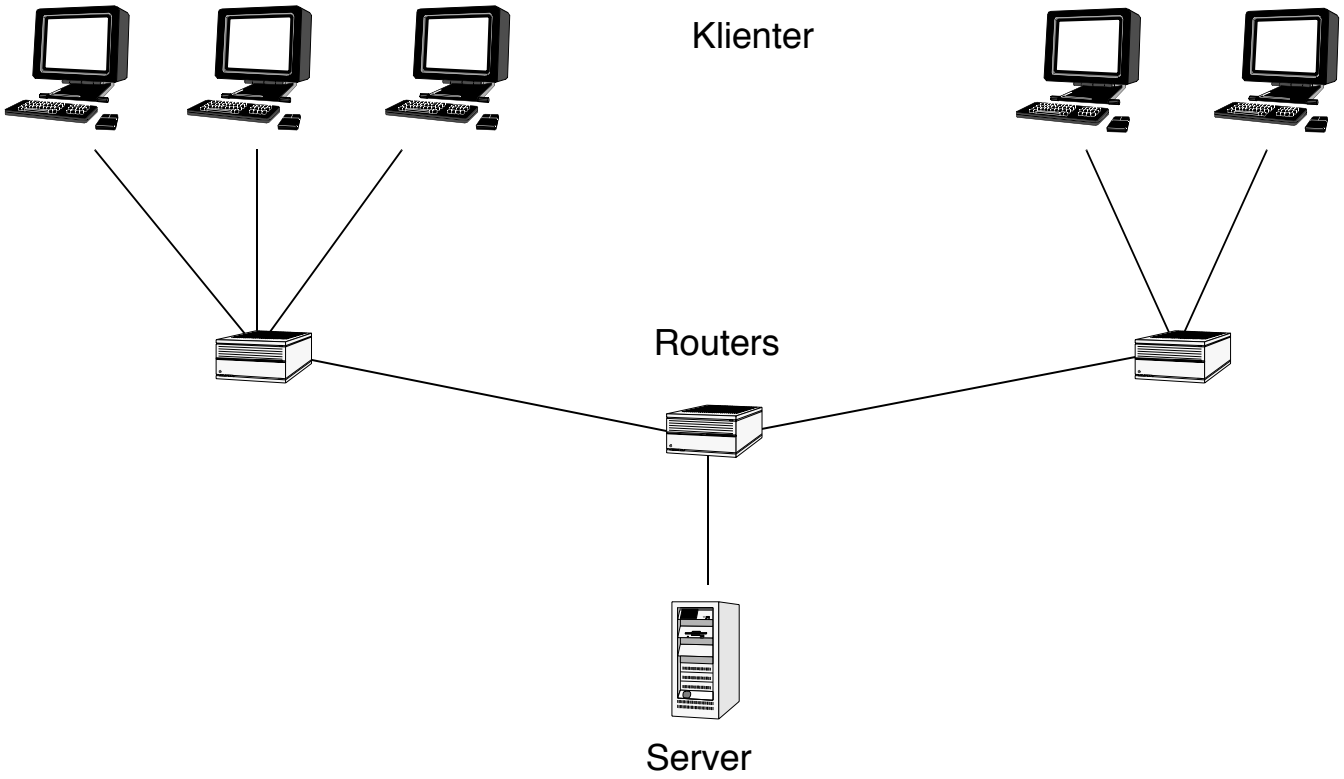
UDP-server: skelett

```
while (true) {
  receive(client, command, parameters);
  switch (command) {
    case commandA:
      result = doCommandA(parameters);
      break;
    case commandB:
      result = doCommandB(parameters);
      break;
    case commandC:
      result = doCommandC(parameters);
      break;
    ...
    default: ...
  }
  send(client, result);
}
```

Utan multicast



Med multicast



Multicast

En variant av UDP.

Unicast

Ett meddelande sänds från en avsändare till EN mottagare.

Flera mottagare - flera kopior av meddelandet sänds.

Multicast

Ett meddelande sänds från en avsändare till FLERA mottagare – alla som är intresserade av att mottaga det.

Endast EN kopia av meddelandet så långt som möjligt.

Kräver stöd av routrar.

Exempel: Live-utsändning av videodata.

Multicastgrupper

En multicastgrupp omfattar alla datorer som är intresserade av att mottaga en viss typ av meddelanden.

Exempel: En videoutsändning av en live-konsert.

Multicastadresser

Varje multicastgrupp motsvaras av ett speciellt IP-nummer i serien 224.0.0.0 - 239.255.255.255 (IPv4).

Man kan säga att alla datorer i en multicastgrupp "delar på” detta IP-nummer.

Datagram som sänds till IP-numret går ut till alla datorer i multicastgruppen – avsändaren behöver inte veta vilka dessa är.

Att välja multicastadress

Permanenta multicastadresser

IANA - Internet Assigned Numbers Authority - delar ut fasta multicastadresser. Börjar med 224.0, 224.1, 224.2 eller 239.

Domännamn	IP-adress	Syfte
all-systems.mcast.net	224.0.0.1	Alla datorer på det lokala subnätet.
experiment.mcast.net	224.0.1.20	Experiment som inte går utanför det lokala subnätet.
ntp.mcast.net	224.0.1.1	Network Time Protocol
ietf-1-video.mcast.net	224.0.1.12	Video från IETF-möten. Kanal 1.

Tillfälliga multicastadresser

Vem som helst kan välja vilken adress som helst som inte är reserverad av IANA.

Time To Live – TTL

Till för att undvika överdrivna trafikvolymer och begränsa spridningen av multicastpaket.

Varje paket förses med ett "bäst-före-datum", TTL, i form av en räknare som räknas ned varje gång paketet passerar en router.

När paketets räknare blir noll dör paketet.

Java och multicast

Meddelanden är av typen DatagramPacket - som för UDP.

MulticastSocket

Ersätter DatagramSocket, men är likartad.

Subklass till DatagramSocket.

En MulticastSocket kan:

- Ansluta sig till en multicastgrupp.
- Skicka meddelanden till andra datorer i gruppen.
- Mottaga meddelanden från andra datorer i gruppen.
- Lämna en multicastgrupp.

MulticastSocket

Konstruktorer

Som för DatagramSocket:

```
public MulticastSocket() throws SocketException;
public MulticastSocket(int port) throws SocketException;
```

Ansluta till en multicastgrupp

Behövs bara för att ta emot meddelanden.

```
public void joinGroup(InetAddress address)
                                    throws IOException;
```

Lämna en multicastgrupp

```
public void leaveGroup(InetAddress address)
                                    throws IOException;
```

MulticastSocket, fortsättning

Ange Time-To-Live

```
public void setTimeToLive(int ttl) throws IOException;
```

Sända paket

```
public void send(DatagramPacket packet) throws IOException;
```

Ta emot paket

```
public void receive(DatagramPacket dp) throws IOException;
```

Frigöra portar

```
public void close();
```