Universiteti i Prishtinës "Hasan Prishtina" Kosovë

Programimi i distribuar

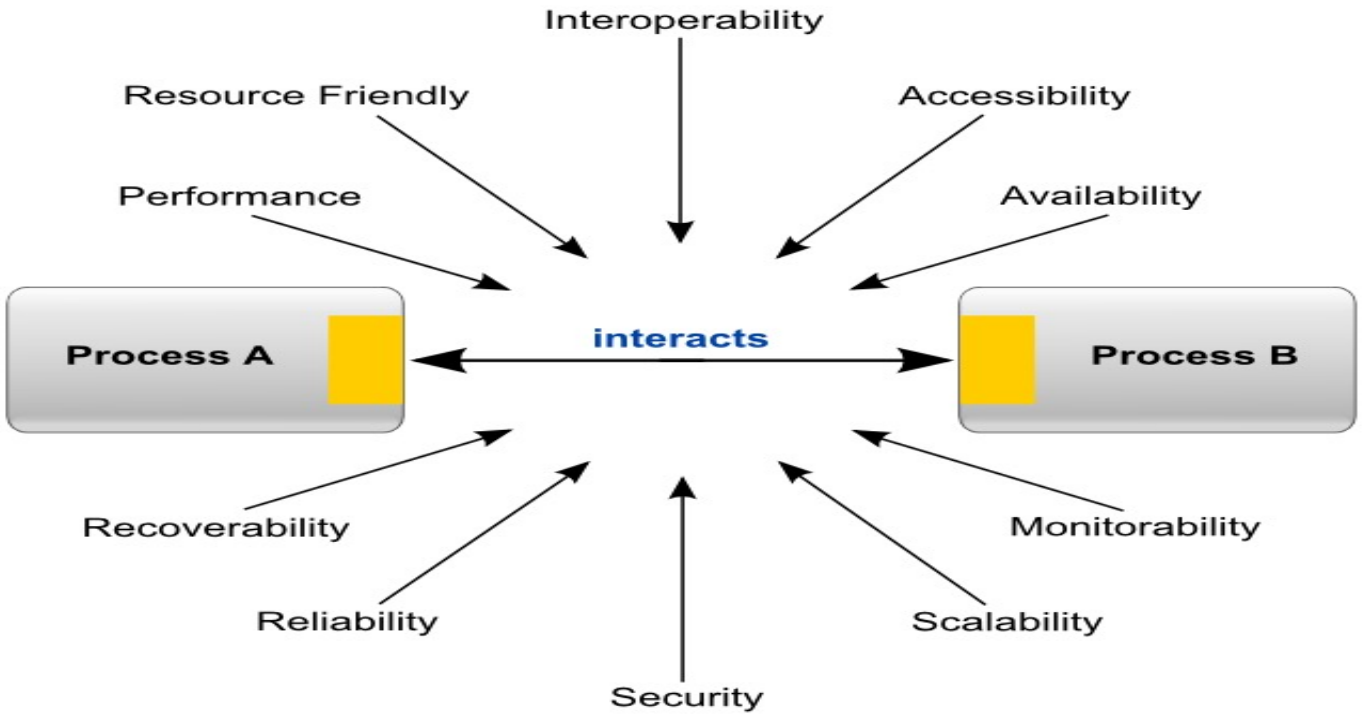
Pjesa 4 – Komunikimi ndërmjet proceseve

Prof. Asoc. Dr. Ermir Rogova

Përmbajtja

- Hyrje
- API për Protokollet e Internetit
- Përfaqësimi i të dhënave të jashtme
- Komunikimi Klient-Server
- Komunikimi në grup

Komunikimi ndërmjet proceseve - (IPC) Inter-process communication



Komunikimi ndërmjet proceseve

- Komunikimi ndërmjet proceseve (IPC) është një grup i ndërfaqeve të programimit që i lejojnë një programuesi të koordinojë aktivitetet në mes të proceseve të ndryshme të programit që mund të punojnë njëkohësisht në një sistem operativ.

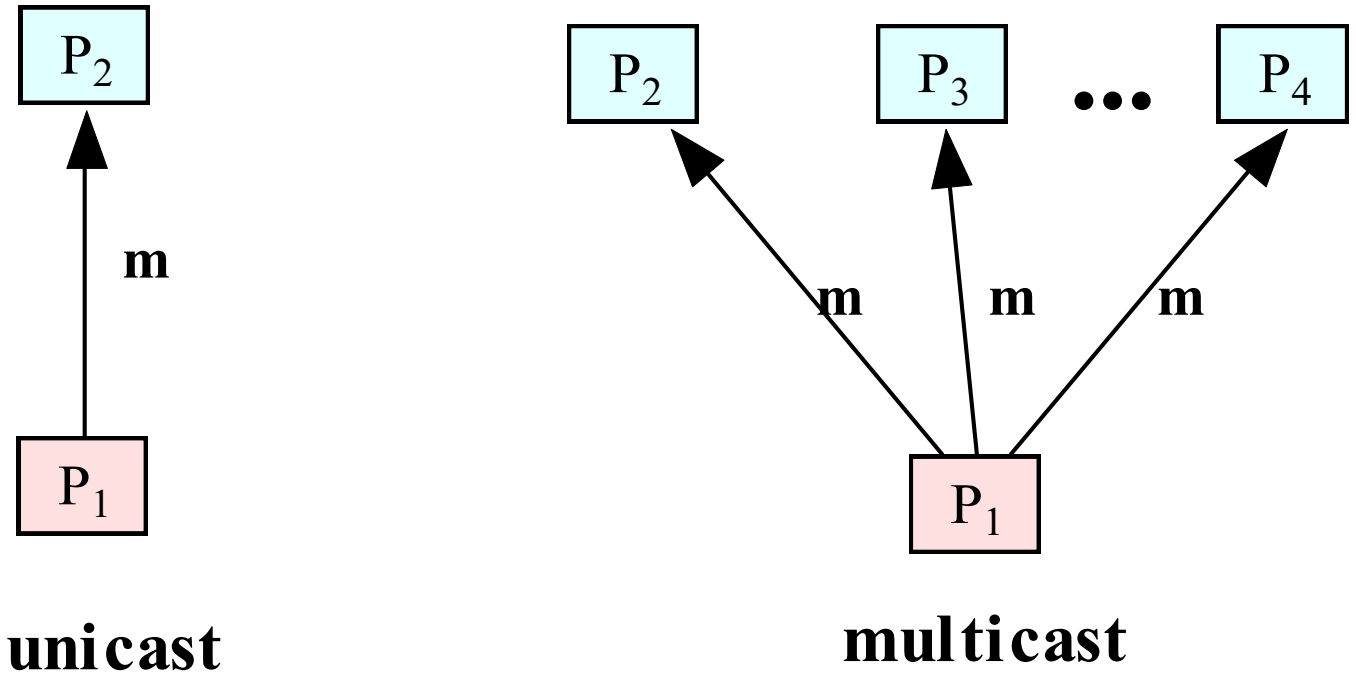
Komunikimi ndërmjet proceseve

- Shkëmbimi i të dhënave bëhet ndërmjet dy ose më shumë proceseve/threads dhe të pavarura.
- Sistemet operative sigurojnë mjedise / burime për IPC, të tilla si rradhët e mesazheve (queues), semaforët, dhe memorja e ndarë (shared).
- Sistemet e shpërndara bëjnë të mundur përdorimin e këtyre objekteve / burimeve që ofrojnë ndërfaqe të aplikacioneve të programuara (**API-application programming interface**) që lejon IPC të jetë i programuar në një nivel më të lartë të abstraksionit. (p.sh. Dërgimi(send) dhe të marrja(receive))

IPC – unicast dhe multicast

- Në sistemet e shpërndara, dy ose më shumë procese angazhohen në IPC duke përdorur një protokoll.
- Një proces mund të jetë një dërgues (sender) në disa pika gjatë një protokolli, dhe marrës (receiver) në pikat e tjera.
- Kur komunikimi është prej një procesi drejt një procesi tjetër të vetëm, IPC është një unicast, p.sh., komunikimi Socket. Kur komunikimi është prej një procesi drejt një grup të proceseve, IPC është një multicast, p.sh., Publikimi (publish)

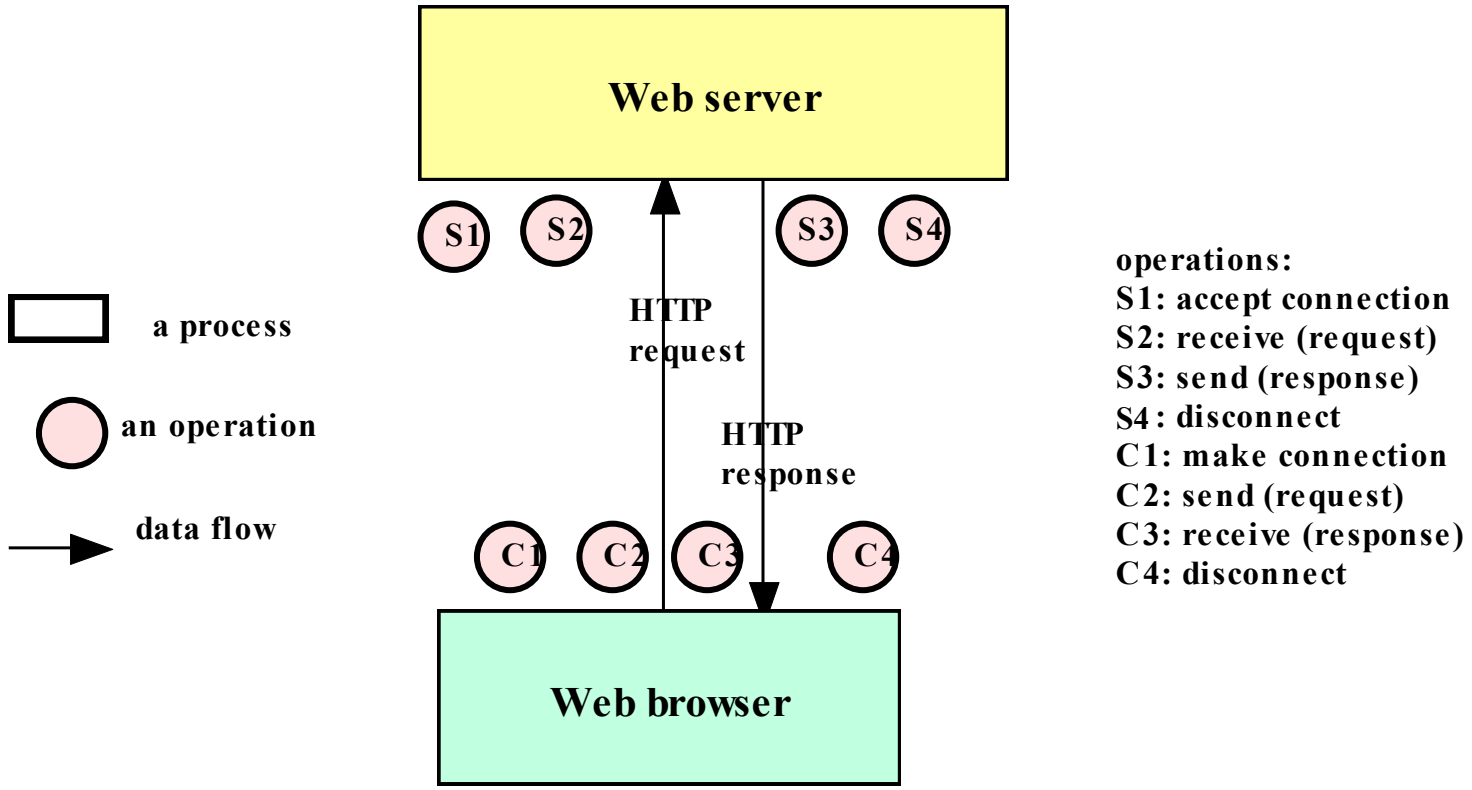
Unicast vs Multicast



Operacionet e parashikuara në një API tipik

- **Connect** (lidhja) (sender address, receiver address), për komunikim connection-oriented
- **Send** – dërgimi ([receiver], message)
- **Receive** ([sender], message storage object)
- **Disconnect** - shkëputja (connection identifier), për komunikim connection-oriented

Komunikimi Interprocess në HTTP



Rënditja e procesimit: C1, S1, C2, S2, S3, C3, C4, S4

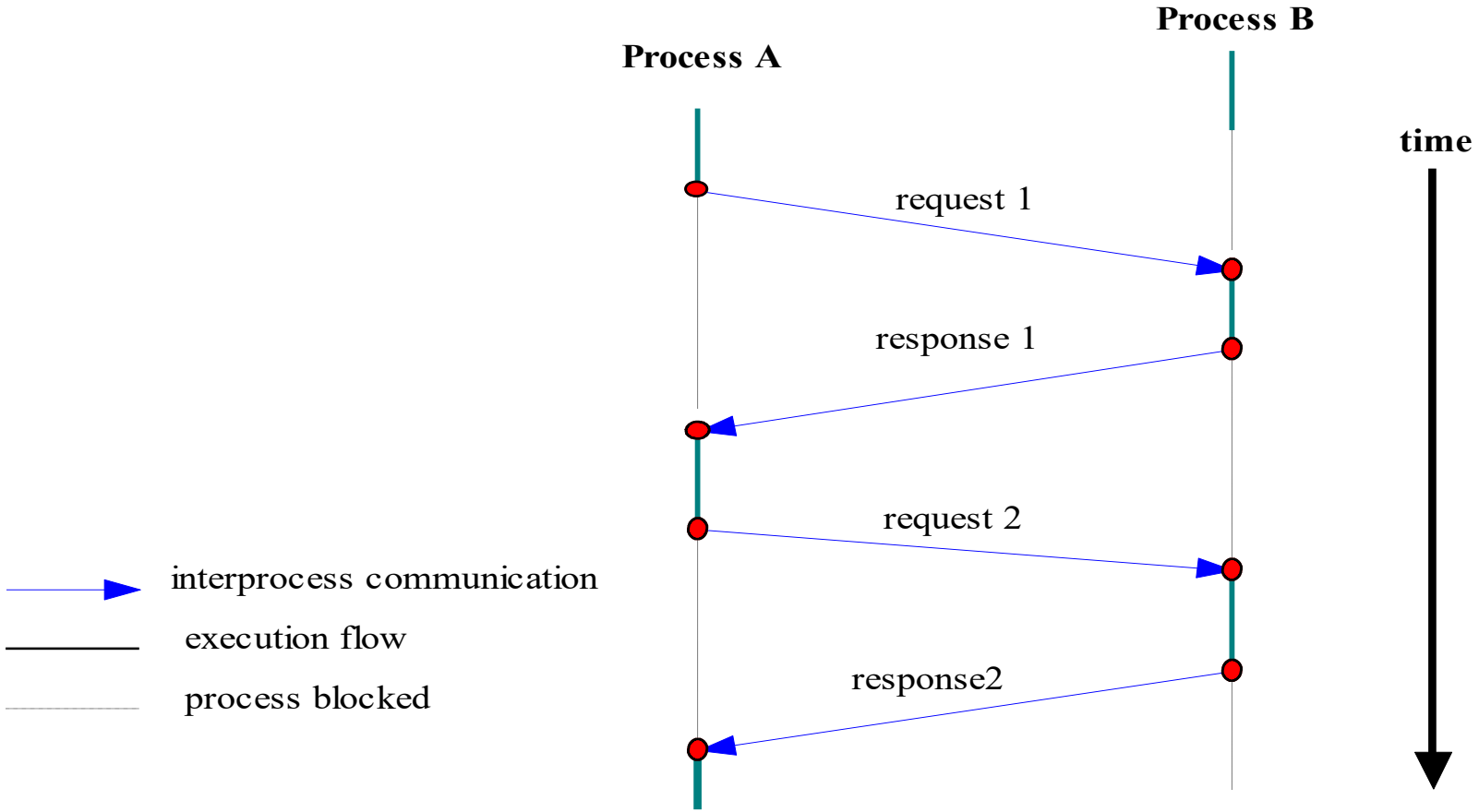
Sinkronizimi i ngjarjeve

- Komunikimi Interprocess mund të kërkojë që të dy proceset të sinkronizojnë operacionet e tyre: njëra anë dërgon, atëherë tjetri merr deri sa të gjitha të dhënat janë dërguar dhe marrë.
- Në mënyrë ideale, operacioni i dërgimit fillon para se operacioni marrës të fillojë.
- Në praktikë, sinkronizimi kërkon mbështetje të sistemit.

Komunikimi Sinkron vs Asinkron

- Operacionet IPC mund të sigurojnë sinkronizimin e nevojshëm duke përdorur bllokimin.
- Një operacion bllokimi i lëshuar nga një proces do të bllokojë përpunimin apo procesimin e mëtejshëm të derisa operacioni të përmbushet apo plotësohet.
- Nga ana tjetër, operacionet IPC mund të jenë asinkron ose nonblocking.
- Një operacion asinkron i lëshuar nga një proces nuk do të bllokojë përpunimin e mëtejshëm të procesit.
- Në vend të kësaj, procesi është i lirë të vazhdojë punën dhe në mënyrë opcionale mund të njoftohet nga sistemi kur operacioni është përmbushur apo plotësuar.

Diagrami i ngjarjes

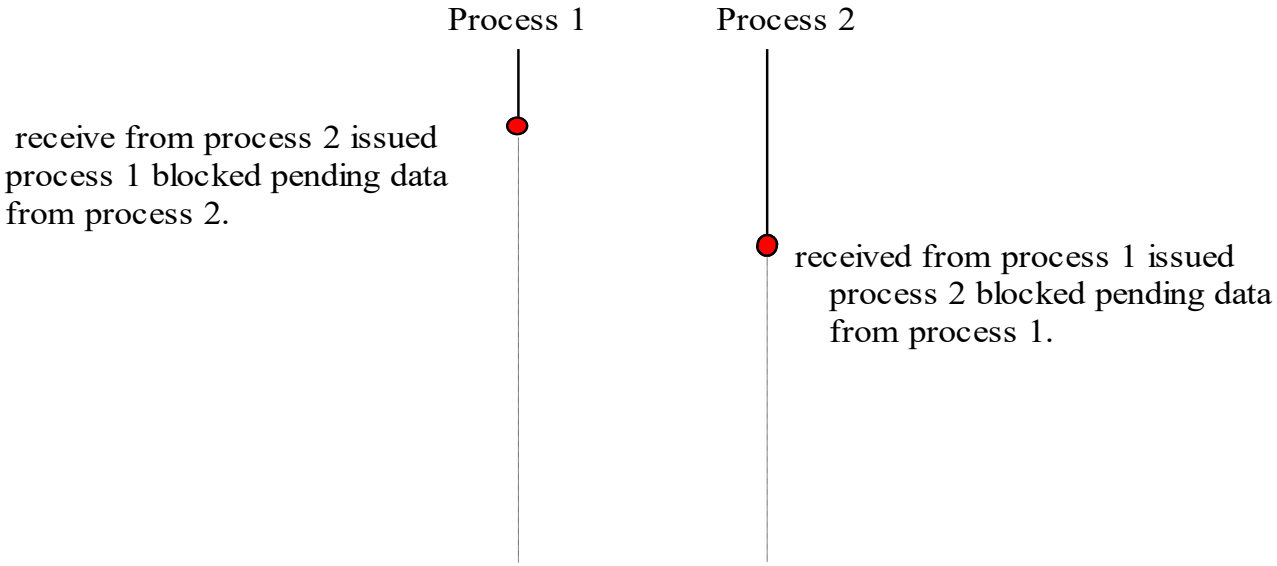


Event diagram for a protocol

Synchronous send and receive

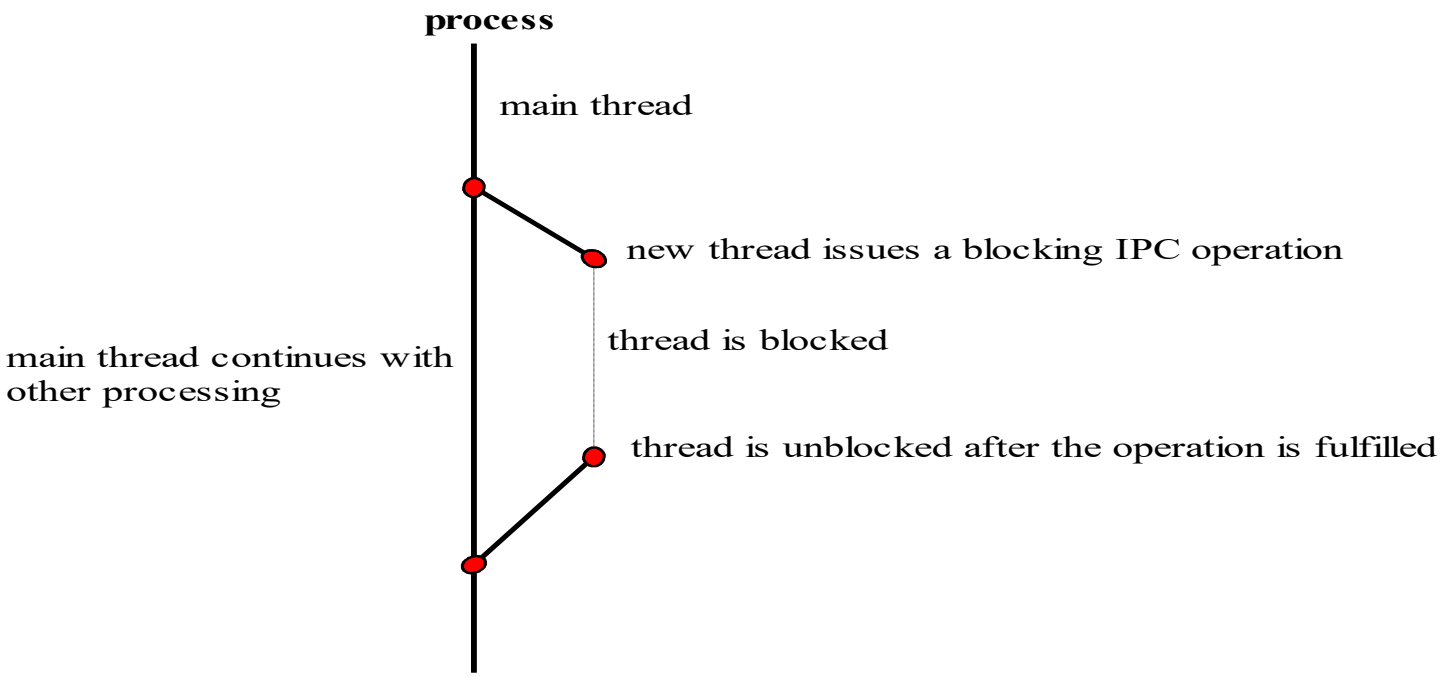
Blocking, deadlock, timeouts

- Operacionet e bllokuara të lëshuara në sekuencë apo rradhitje të gabuar mund të shkaktojnë deadlocks.
- Deadlock-et duhet të shmangen. Përndryshe, timeout mund të përdoren për të zbuluar apo detektuar deadlocks.



P1 pret të dhëna nga P2; P2 pret të dhëna nga P1.

Përdorimi i threads për IPC asinkrone



Përfaqësimi i të dhënave

- Të dhënat që transmetohen në rrjet janë binare apo binary stream.
- Një sistem i komunikimit interprocess mund të sigurojë aftësinë për të lejuar përfaqësim të të dhënave që do të vendosen apo imponohen në të dhënat në radhë (row data).
- Ngaqë kompjuterët e ndryshëm mund të kenë format të ndryshëm të magazinimit(storage) të brendshëm për të njëjtin lloj të të dhënave, atëherë një përfaqësim i jashtëm i të dhënave mund të jetë i nevojshëm – formati standard.
- Marshallimi i të dhënave është procesi i
  - rrafshimit apo shkatërimit të një strukture të të dhënave, dhe
  - konvertimin e të dhënave për një përfaqësim të jashtëm.
- Disa skema të njohura të përfaqësimit të të dhënave të jashtme janë:
  - Sun XDR (External Data Representation)
  - ASN.1 (Abstract Syntax Notation One)
  - XML (Extensible Markup Language)

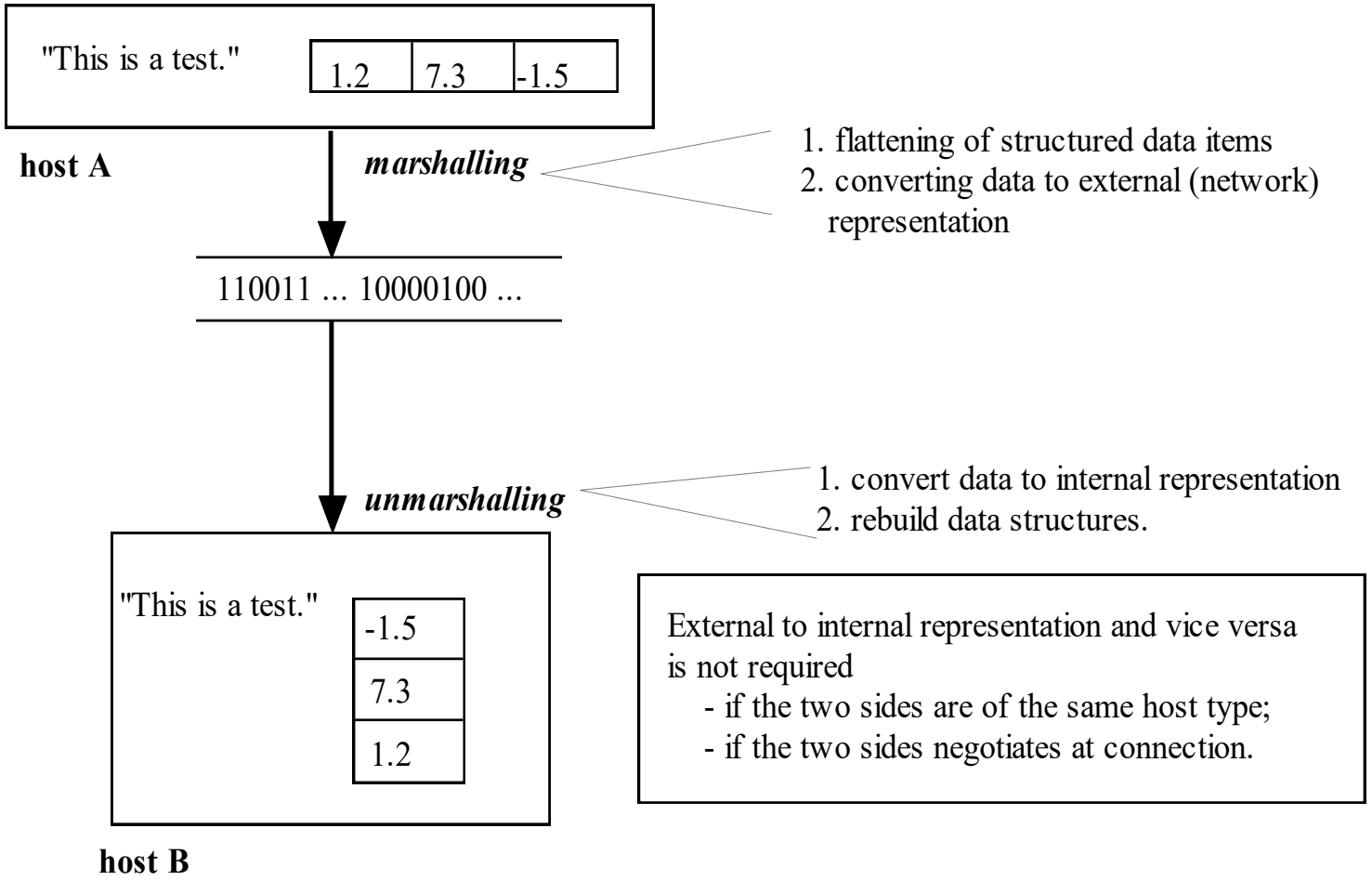
Shembull i një XML file-i

- XML është text-based markup language për shkëmbimin e të dhënave në Web.
- XML ka sintaks analoge në HTML.
- Ndryshe nga HTML, XML tags tregojnë se çka nënkuptojnë të dhënat (data), në vend se si shfaqen ato.
- Shembull:
  - <message>
    - <to>you@yourAddress.com</to> <from>me@myAddress.com</from>
    - <subject>XML Is Really Cool</subject>
    - <text> How many ways is XML cool? Let me count the ways... </text>
  - </message>

Përfaqsimi i të dhënave

- Marshalling është procesi i marrjes së një koleksioni të artikujve të të dhënave dhe grumbullimi i tyre në një formë të përshtatshme për transmetimin e një mesazhi.
- Unmarshalling është procesi i çmontimit të një koleksioni të të dhënave në mbërritje për të prodhuar një koleksion ekuivalent të artikujve të të dhënave në burim.

Data Marshalling / Rradhitja e të dhënave



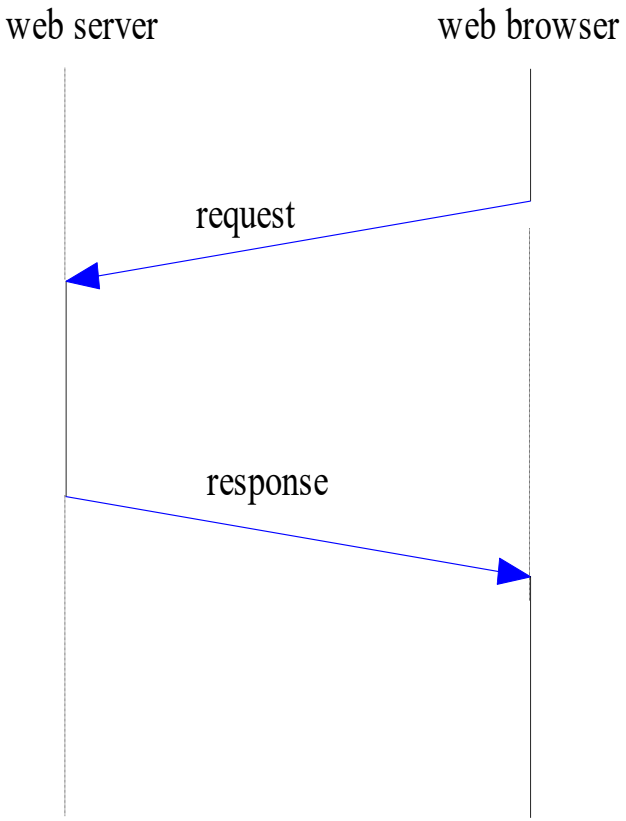
Protokolli

- Në një aplikacion të shpërndarë, dy procese performojnë komunikimin interprocess në një protokoll të përbashkët për të cilin kanë rënë dakord.
- Specifikimi i protokollit duhet të përfshijë
  - sekuencën e shkëmbimit të të dhënave, të cilat mund të përshkruhen duke përdorur kohën e diagramit të ngjarjes.
  - formatin e shkëmbimit të të dhënave në çdo hap.

HTTP: Një protokoll i thjeshtë

- HyperText Transfer Protocol është protokoll për procesin (browser-in) për të marrë një dokument nga web server process-i.
- Është një protokoll i kërkesës/përgjigjes apo request/response protocol:
  - browser-i i dërgon një kërkesë procesit të web server-it, i cili i kthen një përgjigje.

HTTP protokoll-i bazik



request is a message in 3 parts:
- <command> <document adddress> <HTTP version>
- an optional header
- optional data for CGI data using post method

response is a message consisting of 3 parts:
- a status line of the format <protocol><status code><description>
- header information, which may span several lines;
- the document itself.

We will explore HTTP in details later this quarter.

HTTP session i thjeshtë

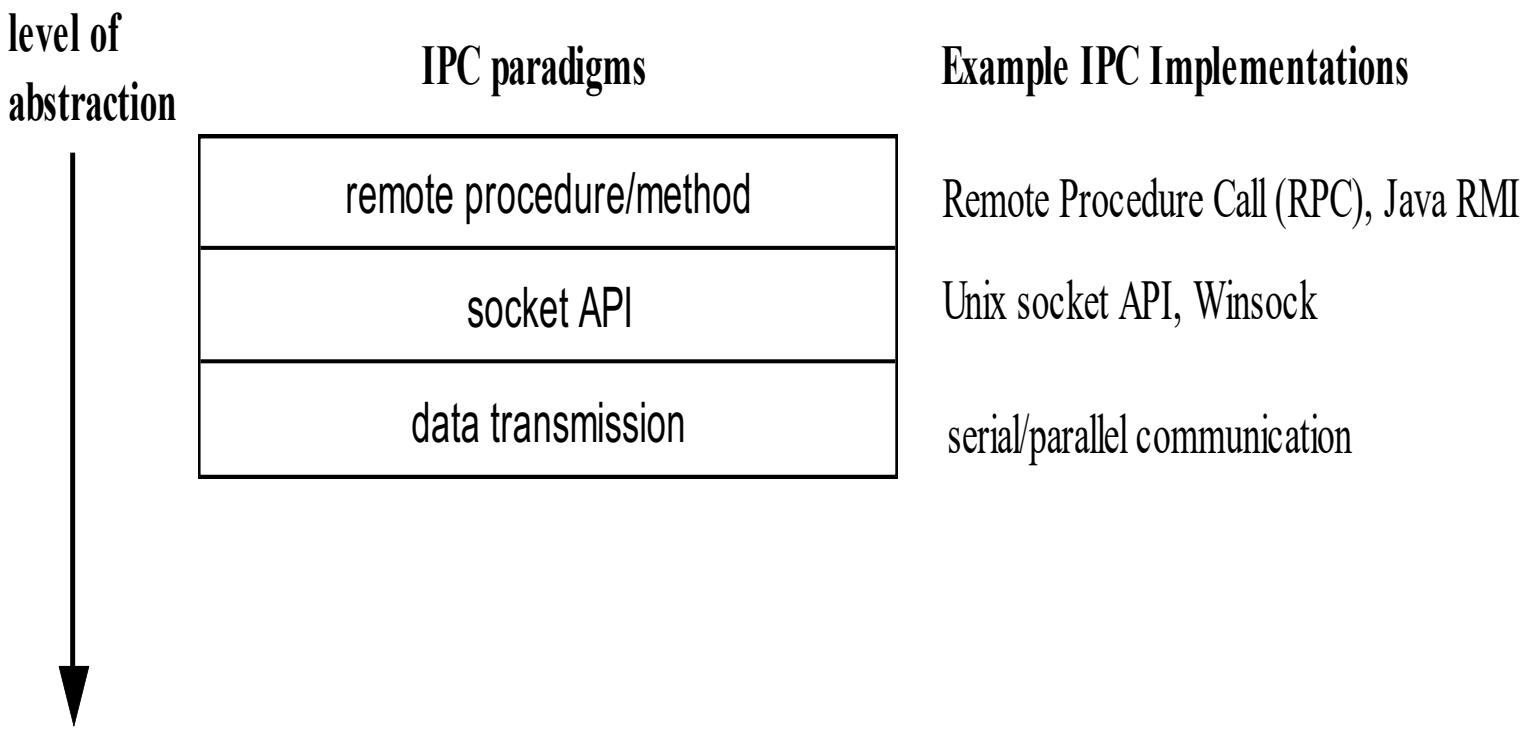
```
Script started on Tue Oct 10 21:49:28 2000
9:49pm telnet www.csc.calpoly.edu 80
Trying 129.65.241.20...
Connected to tiedye2-srv.csc.calpoly.edu.
Escape character is '^]'.
GET   /~mliu/   HTTP/1.0                          <-- HTTP  Request

HTTP/1.1 200 OK                                   <-- HTTP response status line
Date: Wed, 11 Oct 2000 04:51:18 GMT               <-- HTTP response header
Server: Apache/1.3.9 (Unix) ApacheJServ/1.0
Last-Modified: Tue, 10 Oct 2000 16:51:54 GMT
ETag: "1dd1e-e27-39e3492a"
Accept-Ranges: bytes
Content-Length: 3623
Connection: close
Content-Type: text/html

<HTML>                                            <-- document content
<HEAD>
<TITLE> Mei-Ling L. Liu's  Home Page
</TITLE>
</HEAD>
<BODY bgcolor=#ffffff>
 ...
```

Paradigmat IPC dhe implementimi

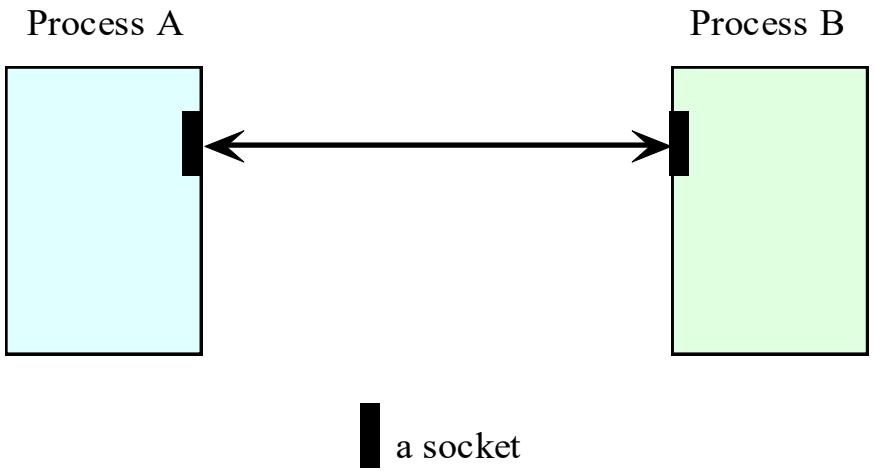
- Paradigmat e IPC niveleve të ndryshme të abstraksionit kanë evoluar, me zbatimet apo implementimet përkatëse.



The socket API – Priza API

- Priza API është një ndërfaqe e (IPC) të dhënë fillimisht si pjesë e sistemit operativ UNIX Berkeley.
- Ajo është bartur në të gjitha sistemet operative moderne, duke përfshirë Sun Solaris dhe Windows sistemet.
- Kjo është një standard për programimin IPC, dhe është baza e ndërfaqes më të sofistikuar IPC siç është remote procedure call (RPC) dhe remote method invocation (RMI).

Modeli konceptual i prizës API



Logical port numbers: **1,024** -- **65,535** ($2^{16} - 1$)

The socket API

- Një prize API siguron një konstrukt programues të socket apo prizë.
- Një proces që dëshiron të komunikojë me një tjetër proces duhet të krijojë një instance të këtij konstrukti (socket)
- Të dy proceset pastaj përdorin operacionet e ofruara nga API për të dërguar dhe marrë të dhëna (p.sh., një mesazh)

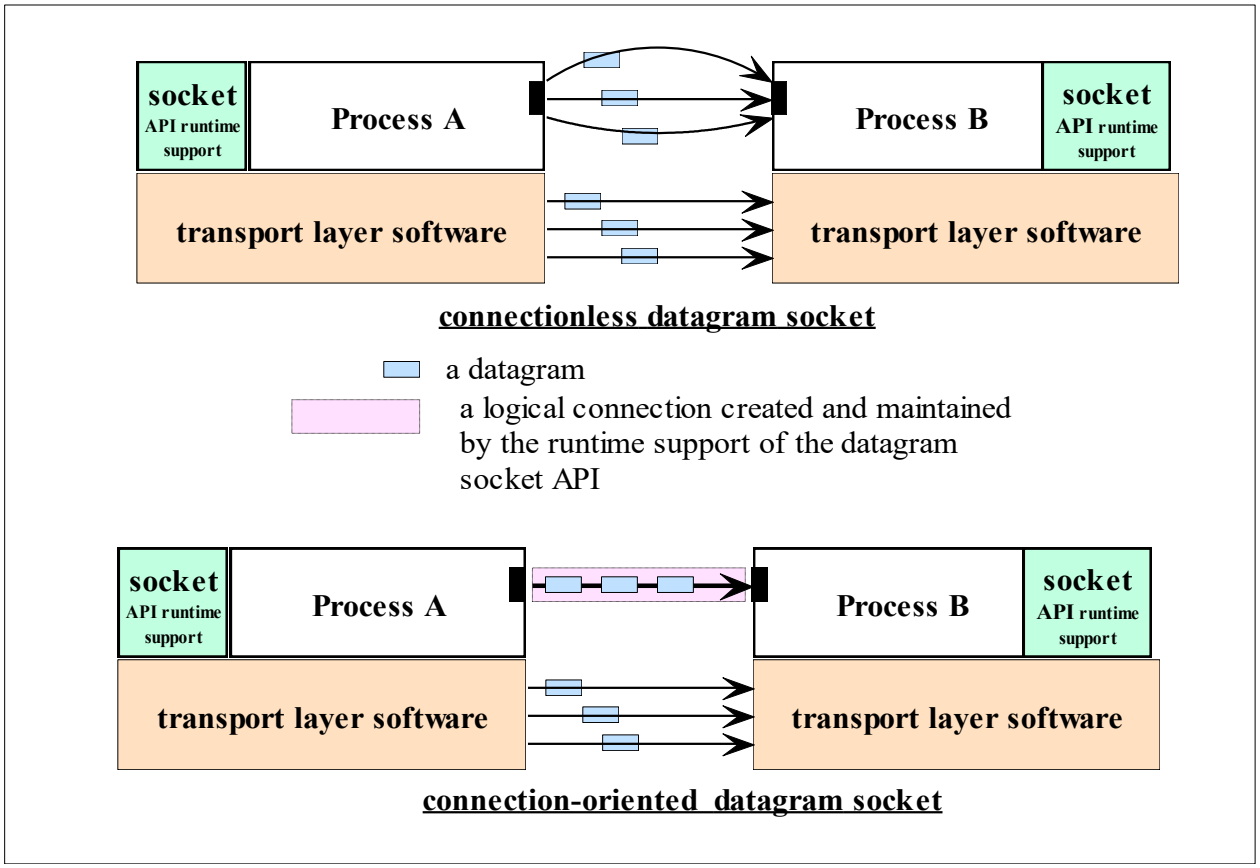
Datagram Socket vs. Stream Socket

- Konstrukti i socket programimit mund të përdoret për UDP (User Datagram Protocol) ose TCP (Transmission Control Protocol).
- Një socket është një përgjithësim i mekanizmit të qasjes së UNIX file-it që ofron një endpoint (pikë e fundme) për komunikim.
- Një Datagram përbëhet nga një datagram header, që përmban IP adresat e burimit dhe destinacionit, dhe një zonë e të dhënave Datagram (datagram data area).
- Sockets që përdorin UDP për transport janë të njohur si datagram sockets, ndërsa ata që përdorin TCP janë quajtur stream sockets.

Datagram socket

- Datagram sockets mund të mbështesin të dy komunikimet connection-less dhe connection-oriented në shtresën e aplikacionit.
- Kjo është kështu, sepse edhe pse datagramet janë dërguar apo pranuar pa nocionin e lidhjeve në shtresën e transportit, biblioteka Runtime e socket-it API mund të krijojë dhe të mbajë lidhje logjike për datagramet e shkëmbyera ndërmjet dy proceseve.
- Biblioteka Runtime e një API është një grup i softuerit që është i lidhur me programin gjatë ekzekutimit në mbështetje të API.

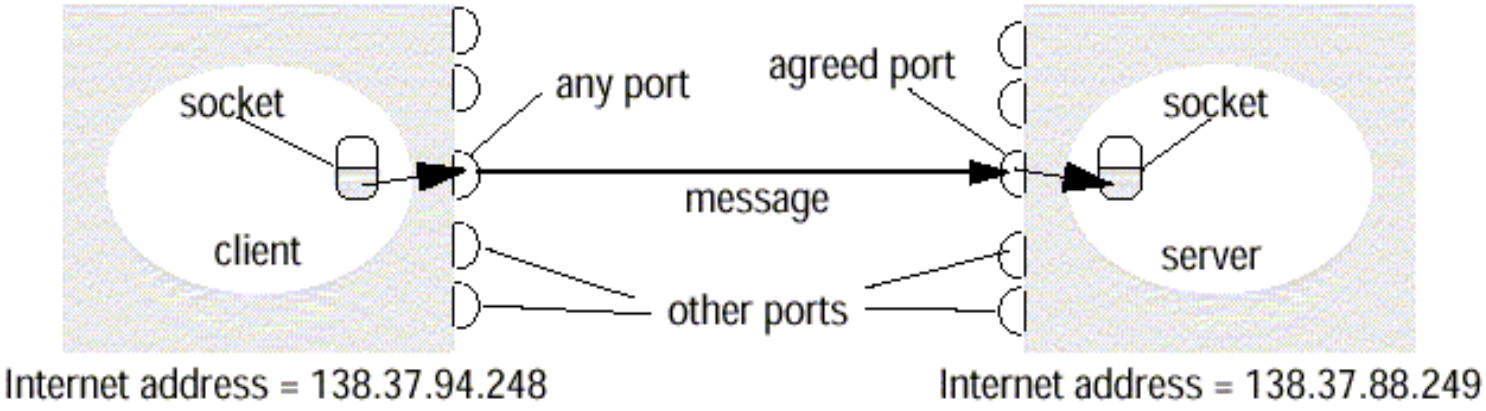
Connection-oriented & connectionless Datagram socket



The Java Datagram Socket API

- Ka dy klasa në Java për API socket Datagramin :
  - Klasa DatagramSocket për sockets.
  - Klasa DatagramPacket për datagrams.
- Një proces që dëshiron të dërgojë ose të marrë të dhëna duke përdorur këtë API duhet të krijojë instance të një
  - Objekt DatagramSocket - një socket
  - Objekt DatagramPacket - një datagram
- Secili socket në një proces marrës thuhet të jetë i lidhur në një port UDP të makinës lokale të procesit.

Sockets



The Java Datagram Socket API

- Për ti dërguar një datagram një procesi tjetër, një proces:
  - krijon një objekt DatagramSocket (socket), dhe një objekt që përfaqëson vetë datagramin. Ky datagram objekti mund të krijohet nga instanca e objektit DatagramPacket, e cila bartë një referencë për një vargu të bajtëve dhe adresën e destinacionit – ID-në e hostit dhe numri i portit, për të cilin socketi i marrësit është i lidhur.
  - lëshon një thirrje për metodën send në objektin DatagramSocket, duke specifikuar një referencë për objektin DatagramPacket si një argument.

The Java Datagram Socket API

```
DatagramSocket    mySocket = new DatagramSocket();
                  // any available port number
byte[ ] byteMsg = message.getBytes( );
DatagramPacket datagram = new DatagramPacket (byteMsg ,
byteMsg.length, receiverHost, receiverPort);
mySocket.send(datagram);
mySocket.close( );
```

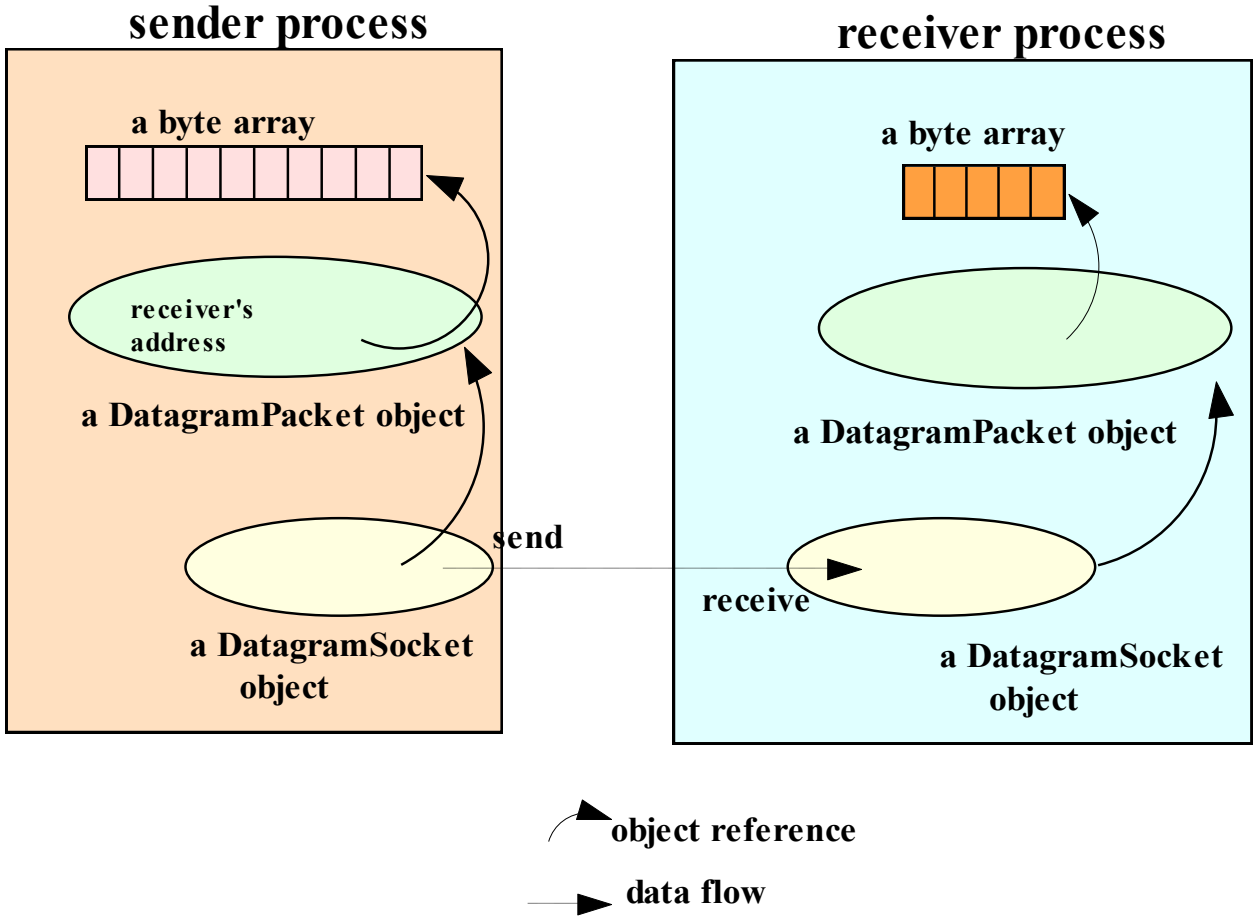
The Java Datagram Socket API

- Në procesin e pranimit (receiving process), objekti DatagramSocket (socket) duhet të jetë gjithashtu i instancuar dhe i lidhur në një port lokal, numri i portit duhet të jetë ai që është specifikuar në paketën e Datagramit të dërguesit.
- Për të marrë apo pranuar datagrame (receive datagrams) të dërguar në socket, procesi krijon një objekt datagramPacket që i adresohet një vargu të bajtave (array byte), dhe thërret metodën receive (receive method) në objektin e DatagramSocket, duke specifikuar si argument një referencë në objektin DatagramPacket.

The Java Datagram Socket API

```
DatagramSocket    mySocket = new DatagramSocket(port);
byte[ ] recMsg = new byte[MAX_LEN];
DatagramPacket datagram =  new DatagramPacket(recMsg, MAX_LEN);
mySocket.receive(datagram);  // blocking and waiting
mySocket.close( );
```

Strukturat e të dhënave në programet e dërguesit dhe pranuesit



Rrjedha e programit në programet e dërguesit dhe pranuesit

sender program

create a datagram socket and
bind it to any local port;
place data in a byte array;
create a datagram packet, specifying
the data array and the receiver's
address;
invoke the send method of the
socket with a reference to the
datagram packet;

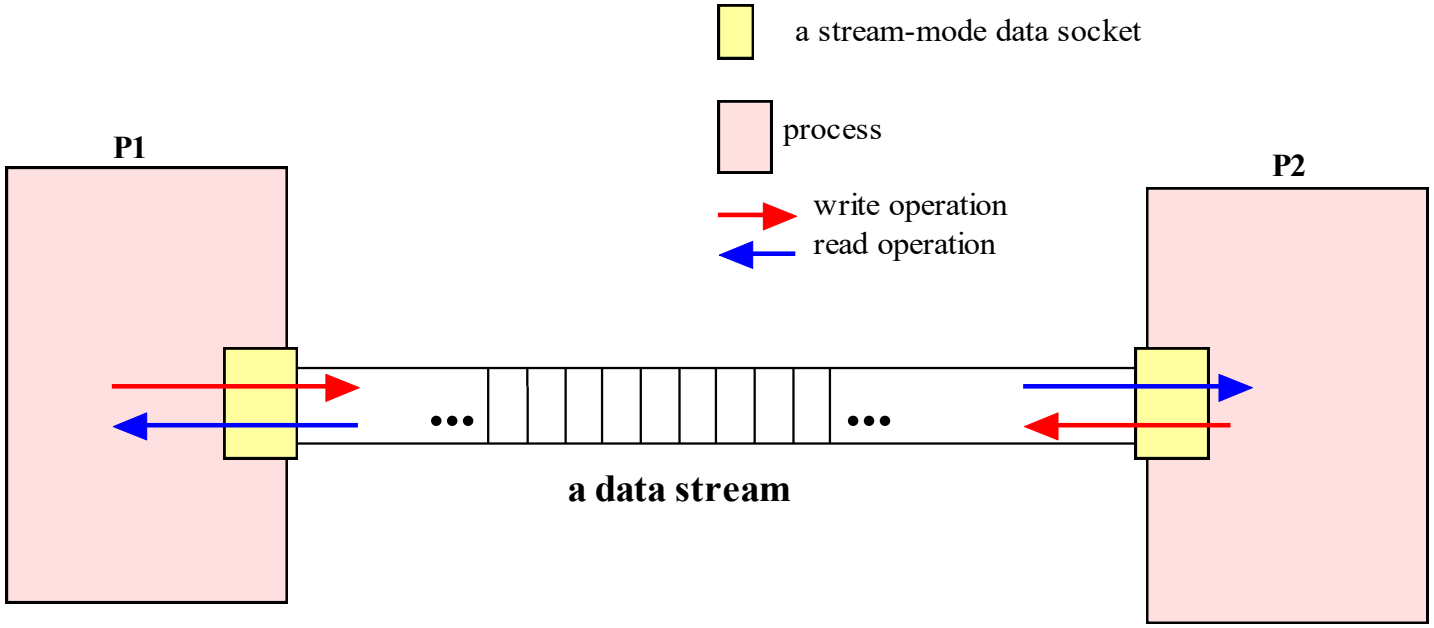
receiver program

create a datagram socket and
bind it to a specific local port;
create a byte array for receiving the data;
create a datagram packet, specifying
the data array;
invoke the receive method of the
socket with a reference to the
datagram packet;

The Stream-Mode Socket API

- Datagram socket-i mbështet shkëmbimin e njësive diskrete të të dhënave.
- Stream socket API ofron një model të transferimit të të dhënave bazuar në stream-mode I/O të sistemeve operative Unix.
- Sipas përkufizimit, një stream-mode socket mbështet komunikim të orientuar në lidhje (connection-oriented communication).

Stream-Mode Socket API (connection-oriented socket API)



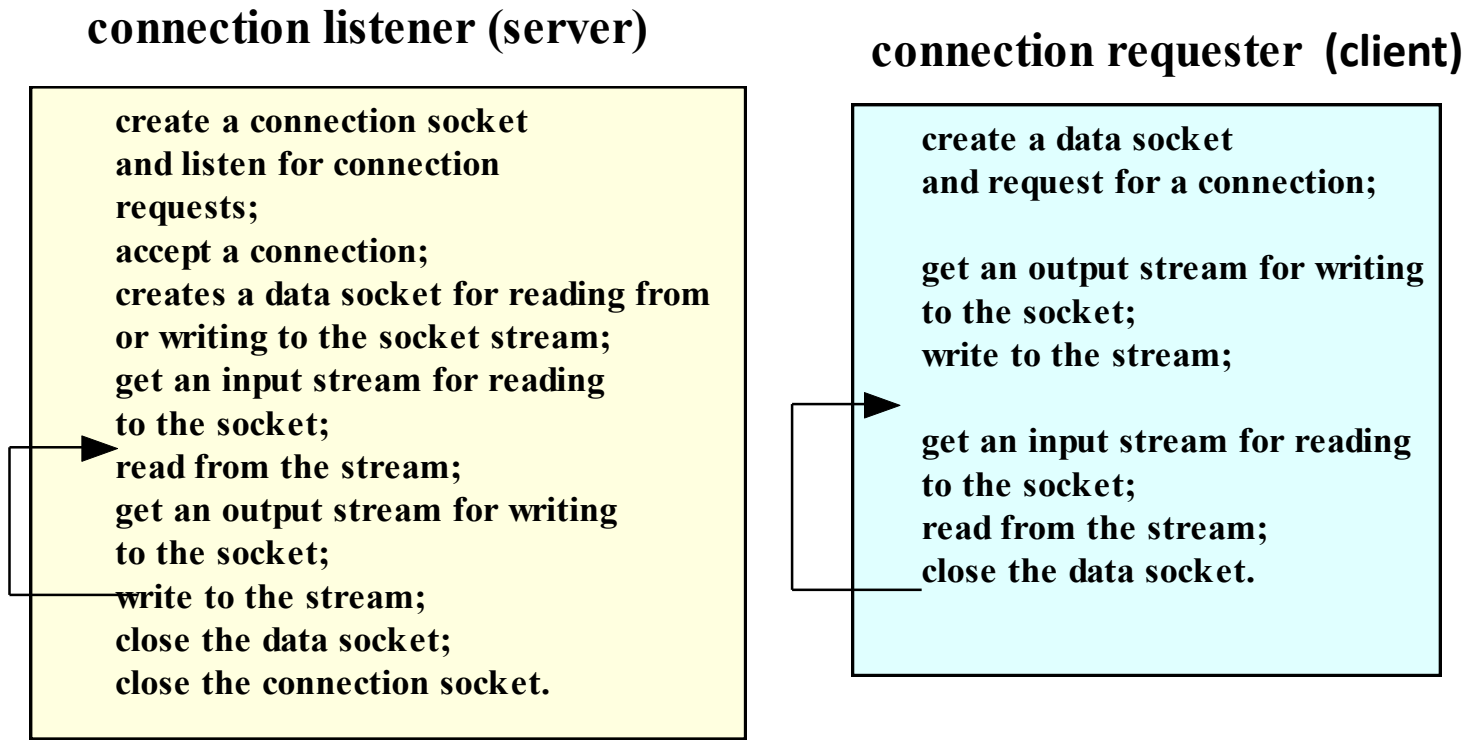
Stream-Mode Socket API

- Një stream-mode socket është krijuar për shkëmbimin e të dhënave ndërmjet dy proceseve të veçanta apo specifike.
- Rrëkeu i të dhënave (Data stream) shkruhet në socket në njërin anë, dhe lexohet nga ana tjetër.
- Një rrëke i të dhënave nuk mund të përdoret për të komunikuar me më shumë se një proces.

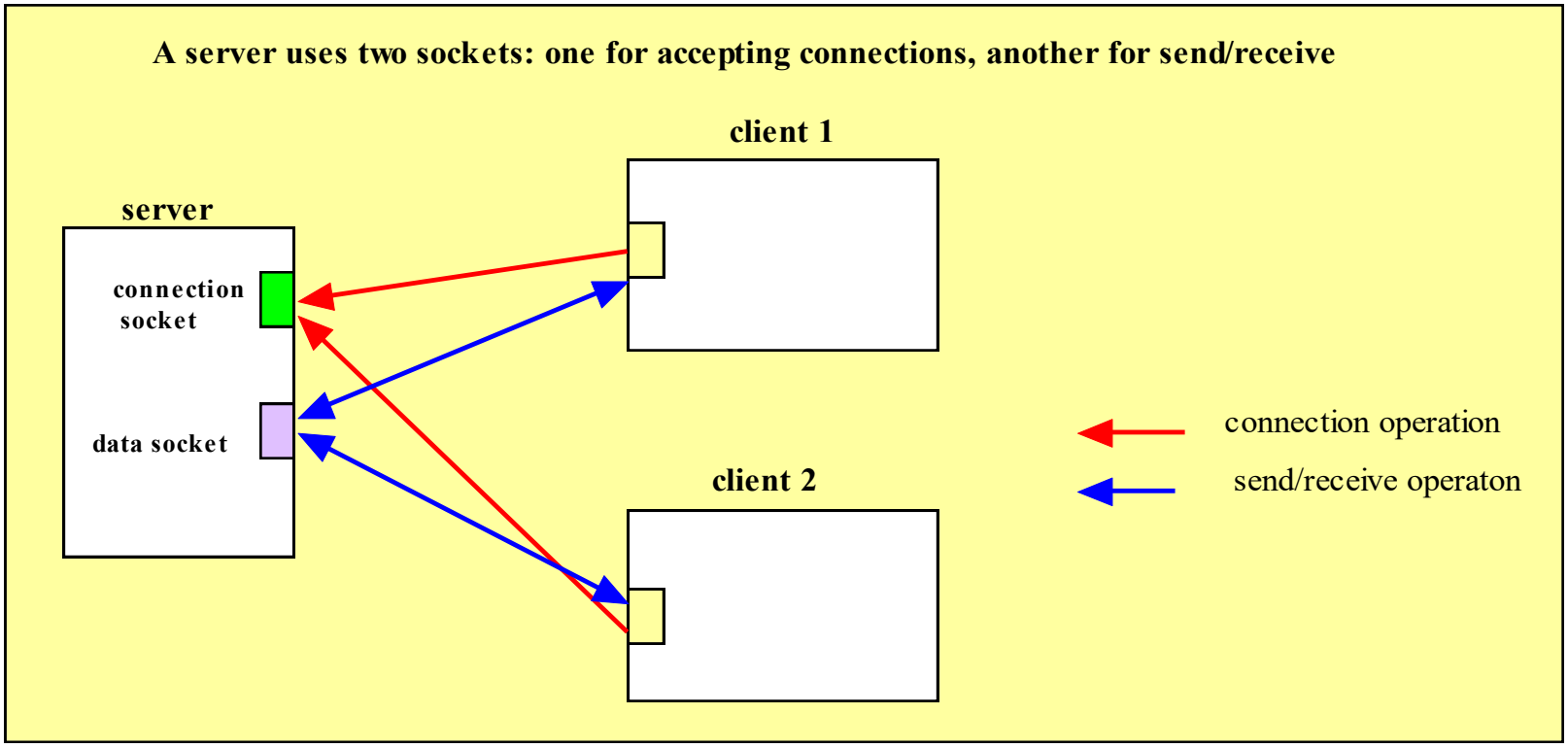
Stream-Mode Socket API

- Në Java, stream-mode socket API është i pajisur me dy klasa:
  - ServerSocket: për pranimin e lidhjeve (accepting connections); ne do të thërrasim një objekt të kësaj klase, një socket për lidhje (connection socket).
  - Socket: për shkëmbimin e të dhënave (data exchange); ne do të thërrasim një objekt të kësaj klase një socket të të dhënave(data socket).

Stream-Mode Socket API rrjedha e programit



Serveri (dëgjuesi i lidhjes)



Pyetje???