University of St Andrews

CS2003: Internet and the Web

Introduction to client-server with TCP

Questions

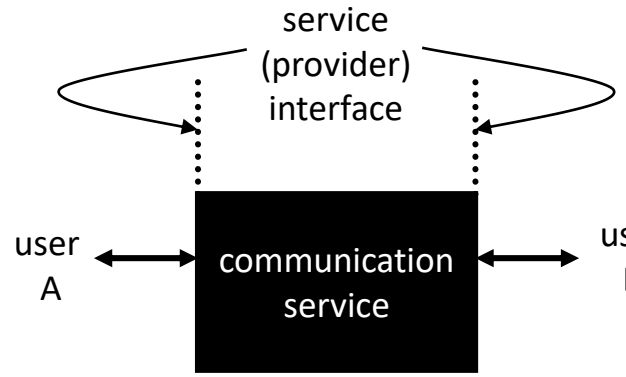
1. What is the client-server model?
2. What is a name, address and port number used for in applications using TCP?

Encapsulation

- Remember encapsulation from yesterday
- Allows us to send and receive payload from different layers in the protocol model without needing to understand what is encapsulated
- Allows us to build "black box" models with well-known **interfaces**

The **black box** model: protocols and services

- Data service interface:
  - communication service
  - **service provider**
  - **service primitives**



- **Unconfirmed** service:
  - **request** sent
  - **indication** received

- **Confirmed** service:
  - **response** from receiver
  - **confirm** at sender

Connection-less (CL)

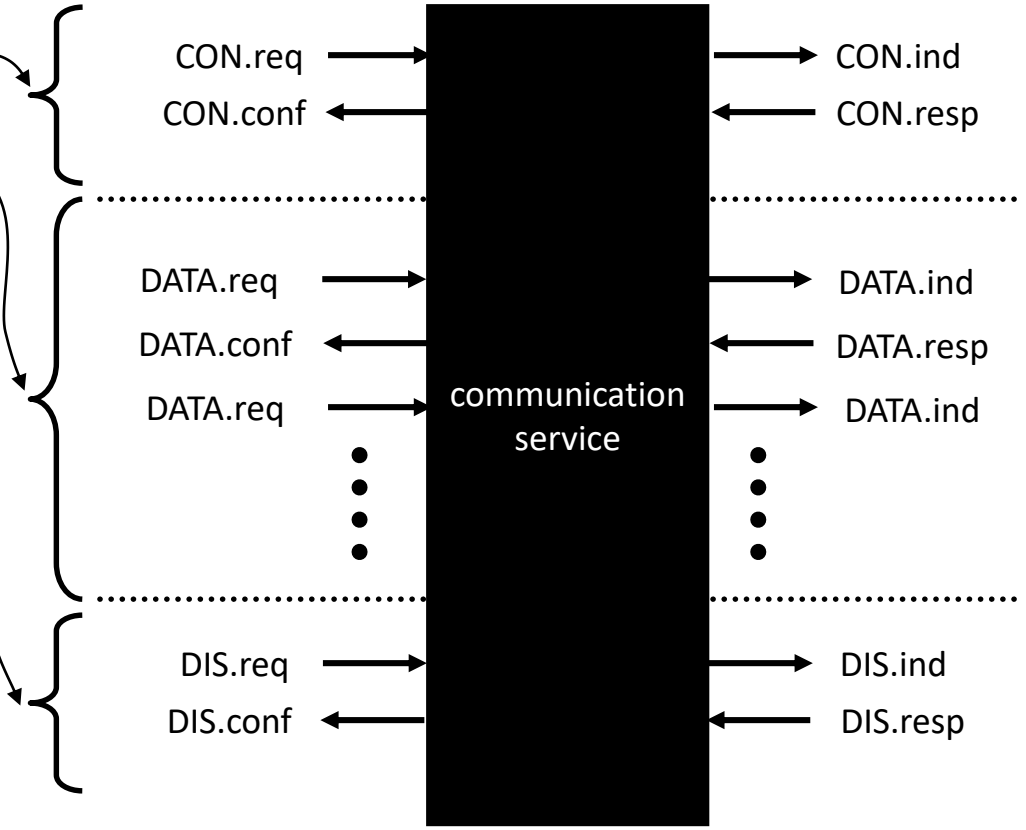
- Send and receive data:
  - **datagrams** (packet).
  - no connection set-up.
- "Send and forget":
  - service is typically **unconfirmed.**
  - (**confirmed** is possible)
- Suitable for "bursty" data.
- Simple interface.
- Resource control (?):
  - packet **flows** may need special treatment.

Connection-oriented (CO)

- Three phases:
  - connection establishment
  - data transfer
  - connection release

- Connection management:
  - signalling (latency)
  - connection state
  - typically confirmed service

- Resources maintained(?):
  - Resources at set-up?
- Multipoint / multiparty?
  - point-to-multipoint
  - multipoint-to-multipoint



CL vs. CO

University of St Andrews

CL, e.g. UDP (courier)

- ☺ Low latency.
- ☺ Low traffic overhead.
- ☹ No reliability.
  - Dealing with loss?
    - ☹ No ordering.
  - Multipath effects?
    - ☹ Possible duplication:
  - Errors in network?
    - ? Control of packet flow:
  - Congestion (network overload)?
  - Flow (receiver overload)?

CO, e.g. TCP (phone call)

- ☺ Reliable delivery.
- ☺ Ordered delivery.
- ☺ No duplicates.
- ☺ Control of packet flow:
  - ☺ Congestion control.
  - ☺ Flow control.
- ☹ Connection overhead:
  - ☹ latency.
  - ☹ traffic overhead.
  - ☹ no flow rate control.

TCP: client-server

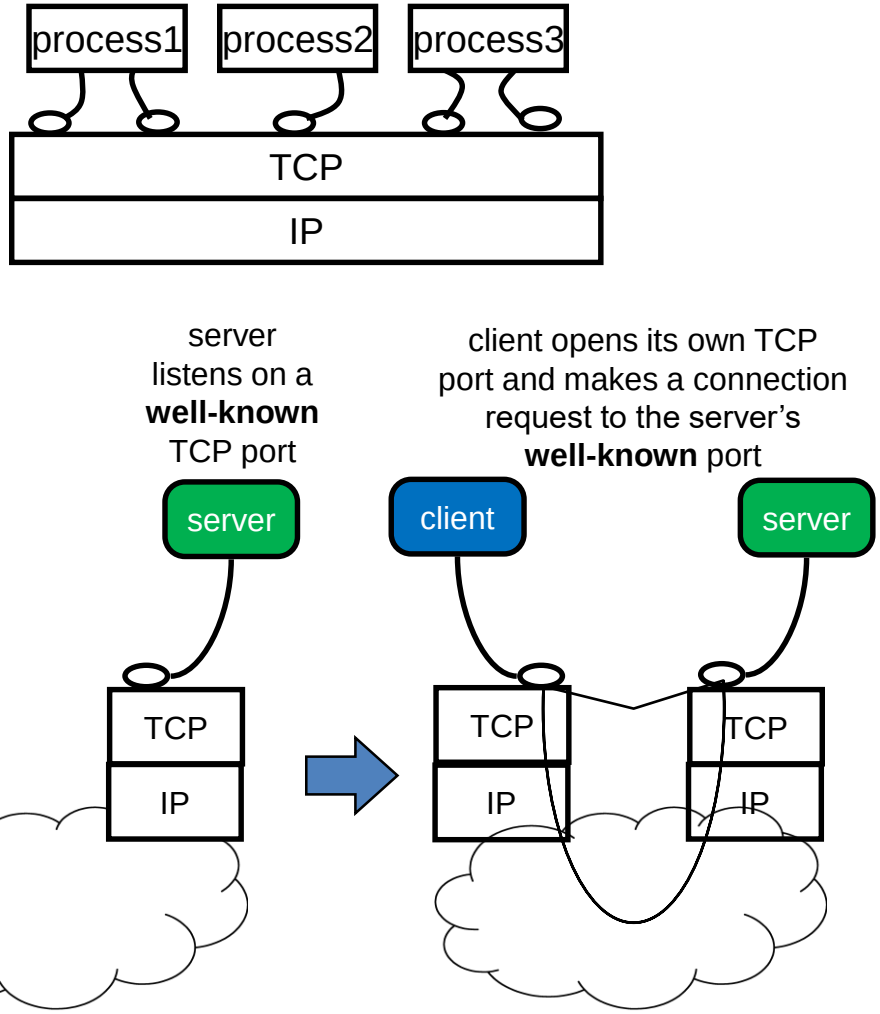
- **Server**: a host / program that provides a service:
  - Waits for clients to **connect** to make use of service.
  - Application specific.
- **Client**: a program that makes use of a service:
  - Must identify a server and **connect** to it.
  - Typically has interface for a human user, e.g. GUI.
- Examples: email, WWW, SSH.
- Server/service usually identified by a name:
  - e.g. www.cs.st-andrews.ac.uk

FQDNs and IPv4 addresses

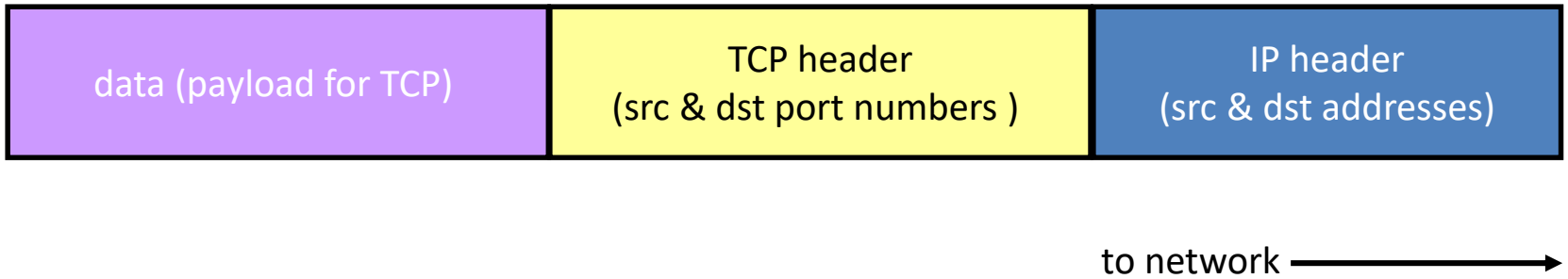
- Fully Qualified Domain Name (FQDN):
  - trenco.cs.st-andrews.ac.uk
- IPv4 address, 32 bit unsigned number, normally written in “dotted decimal” notation:
  - 138.251.22.79
- Domain Name System (DNS):
  - Mappings of FQDN to address
  - Reverse lookup possible: IP address to FQDN.
- Tools:
  - *host*, *nslookup*, *dig*

TCP port numbers

- Port numbers:
  - Layer 4 (transport) de-multiplexers.
  - **Assigned**, IANA
  - **Ephemeral**, dynamic
- **Assigned** numbers:
  - "well-known"
  - /etc/services.
- **Ephemeral**:
  - allocated by OS.



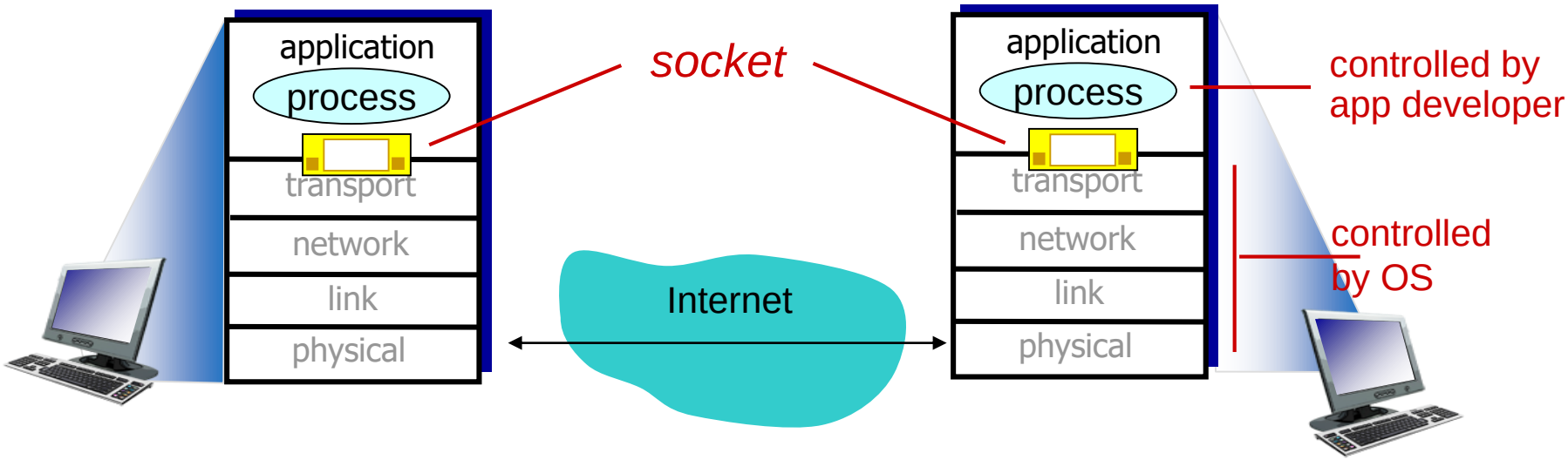
A simple application ”packet”



Socket programming

how to build client/server applications that communicate using *sockets*

socket: door between application process and end-end-transport protocol



Socket programming *with TCP*

University of St Andrews

client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

client contacts server by:

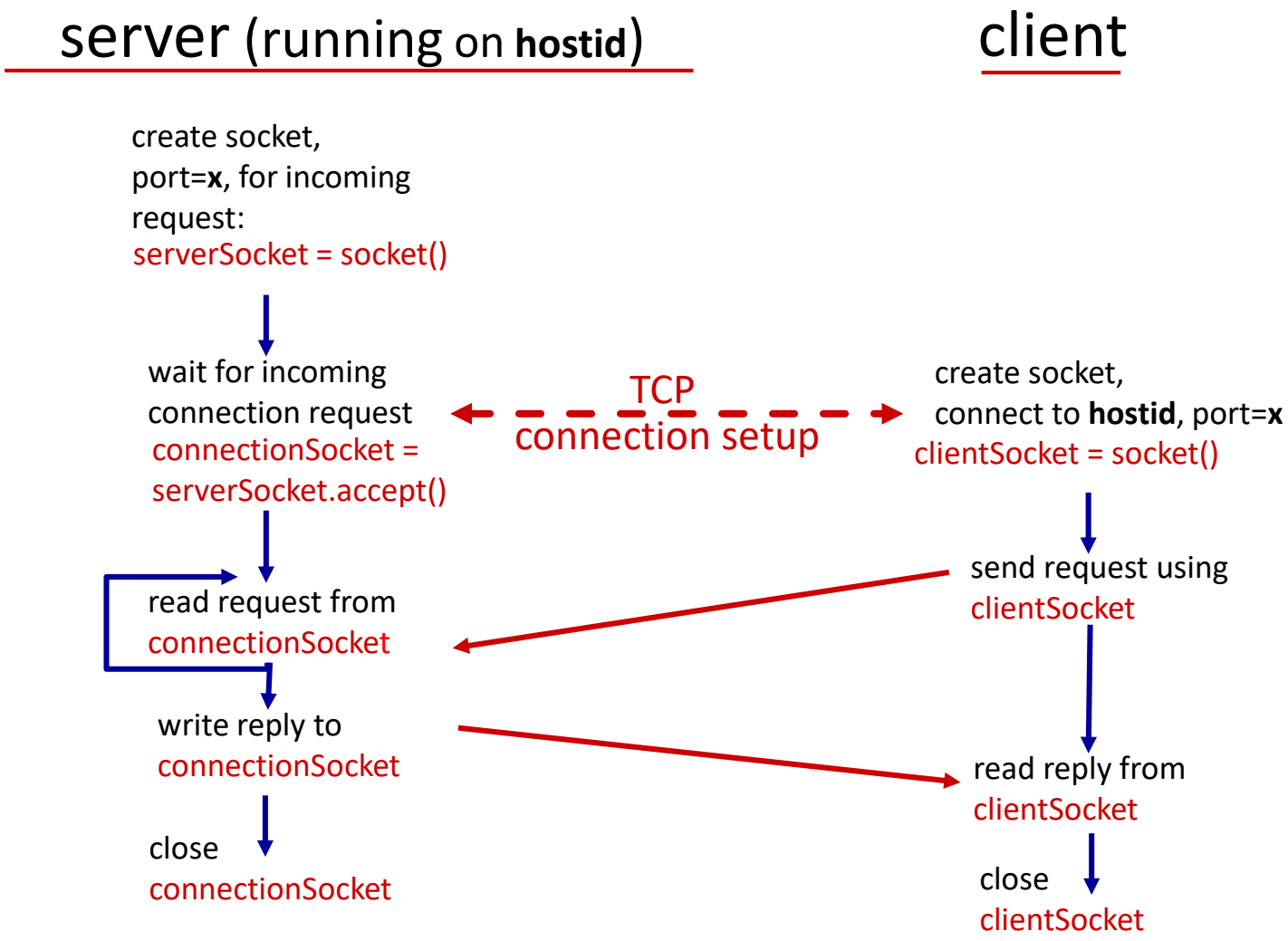
- creating TCP socket, specifying IP address, port number of server process
- *when client creates socket:* client TCP establishes connection to server TCP

- when contacted by client, *server TCP creates new socket* for server process to communicate with that particular client
  - allows server to talk with multiple clients
  - source port numbers used to distinguish clients (more later)

application viewpoint:

TCP provides reliable, in-order byte-stream transfer ("pipe") between client and server

Client/server socket interaction: TCP



Summary

- Brief introduction to client-server model with TCP.
  - more later
- Names and addresses.
- Port numbers.
- Application messages.
- Sockets (doors) – an API for clients and servers to send data
  - see code in the exercise class.
- Reading: Chapter 1 of Peterson & Davie or Kurose & Ross