Lập trình mạng (Network Programming)

Chương 6. Client/server applications

1

Echo protocol

- Server
  - Chờ kết nối từ phía client
  - Nhận một chuỗi bytes
  - Gửi trả client
- Steps
  - Khởi tạo một phiên – tạo socket kết nối đến server
  - Server chấp nhận kết nối và chờ nhận dữ liệu
  - Client gửi chuỗi bytes đến server
  - Client chờ nhận lại chuỗi byte từ server
  - Server nhận chuỗi bytes từ client – gửi trả lại client
  - Client nhận từ server, đóng kết nối

Khi nào server và client biết toàn bộ dữ liệu đã gửi?

2

Framing

- Frame: đơn vị truyền dữ liệu trên mạng
- Giao thức: một message gửi qua kết nối, gửi xong thì bên gửi đóng socket
- Sử dụng độ dài message cố định. Bên nhận sẽ đọc số byte và biết khi nào nhận đủ dữ liệu
- Đặt độ dài message ở đầu message (prefix)
- Sử dụng ký tự đặc biệt để báo hiệu kết thúc message

3

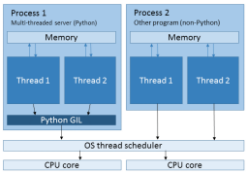
Xây dựng chương trình

- Base module (dùng cho cả server và client)
  - Tạo socket, listen, accept
  - Tạo message theo giao thức (thêm \0 vào cuối message)
  - Nhận dữ liệu
    - Lặp, nhận cho đến khi gặp null (\0)
  - Gửi dữ liệu
- Server
- Client

4

Multithreading

- Copy code và chạy trên các thread/process khác
- Hệ điều hành lập lịch trên CPU



5

Global Interpreter Lock (GIL)

- GIL chỉ cho phép 1 thread hoạt động
  - Không tận dụng được hết ưu thế của máy tính đa nhân, đa luồng
  - Các thread đang blocking cũng có thể được ưu tiên lại
  - Thời gian chờ I/O lâu hơn thời gian xử lý dữ liệu
    - → multithread (nhanh hơn dùng multiprocess)

6

Multithreading

- class threading.Thread(group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None)
  - Group = none
  - Target = đối tượng đc gọi (gửi/nhận dữ liệu)
  - Name: thread name
  - Args = tuple cần gọi (khai báo socket)
  - Kwangs = dictionary/keyword cần gọi
  - Daemon = sets whether thread is daemonic (True)
- thread.start()
- Ví dụ thread

7

Thiết kế giao thức

- Client khởi tạo session – tạo socket kết nối với server
- Server chấp nhận kết nối, chờ dữ liệu từ client
- Client chờ dữ liệu từ server
- Server gửi message đến tất cả các kết nối
- Message kết thúc bằng byte null

- Server gửi message tới tất cả các client (broadcast) → multiple threads

8

Thiết kế server/client

- Server
  - Nhận message từ client → broadcast message tới các clients
  - Gửi message: kiểm tra trong hàng đợi có message mới → send
  - Đóng kết nối, xóa hàng đợi
- Client
  - Nhận chuỗi ký tự (q để kết thúc) → gửi

9

Hàng đợi

- Hoạt động: FIFO (First in, First out)
- Lấy phần tử: get()
- Nạp phần tử: put()
- Lock
- Socket.fileno(): duplicate file descriptor fd – tạo socket với các tham số tương tự trước đó

10