Lập trình mạng

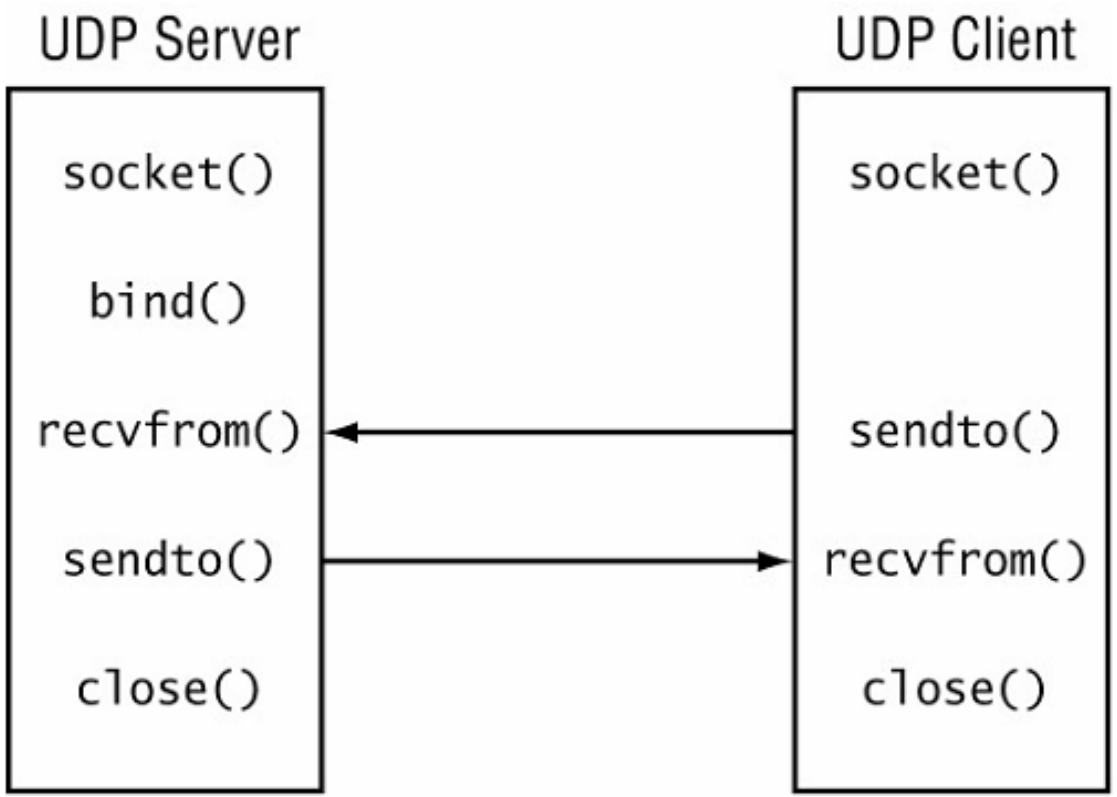
Chương 3:
Lập trình Socket phi kết nối

TS. TRẦN NGÔ NHƯ KHÁNH

Tổng quan

- Truyền thông điệp không cần tạo kết nối (Không dùng phương thức Connect())
- Không đảm bảo truyền dữ liệu đến đích (mạng bận, bị đứt,…)

Mô hình lập trình UDP



Gửi và nhận dữ liệu

- Gửi dữ liệu sử dụng phương thức SendTo()
  - SendTo(byte[] *data*, EndPoint *Remote*)
  - SendTo(byte[] *data*, SocketFlags Flags, EndPoint *Remote*)
  - SendTo(byte[] *data*, int Size, SocketFlags Flags, EndPoint *Remote*)
  - SendTo(byte[] *data*, int Offset, int Size, SocketFlags Flags, EndPoint *Remote*)
- Nhận dữ liệu sử dụng phương thức ReceiveFrom()
  - ReceiveFrom(byte[] data, ref EndPoint Remote)
- *Ví dụ: SimpleUDPServer & Client*

Sử dụng phương thức Connect() trong UDP Client

- Sử dụng khi chỉ gửi và nhận dữ liệu với một host
- Sau khi dùng phương thức Connect(), dùng phương thức Send() và Receive() để gửi và nhận dữ liệu
- *Ví dụ: OddUDPClient*

Phân biệt thông điệp UDP

- UDP phân biệt được biên các thông điệp đã gửi.
- Mỗi thông điệp gửi đi được đóng gói trong một gói tin riêng cùng với thông tin IP thiết bị gửi.
- Mỗi lần gọi phương thức ReceiveFrom() chỉ đọc dữ liệu được gởi từ một phương thức SendTo().
- *Ví dụ: TestUdpServer & Client*

Các vấn đề UDP

- Một chương trình UDP cần quan tâm đến các vấn đề sau:
  - Mất dữ liệu do cách làm việc của phương thức ReceiveFrom ()
  - Phát hiện và cho phép mất gói tin

Ngăn ngừa mất dữ liệu

- Vấn đề
  - UDP không quan tâm việc gửi lại gói tin nên không dùng bộ đệm.
  - Vấn đề có thể phát sinh kích thước bộ đệm của phương thức ReceiveFrom() không đủ lớn
  - *Ví dụ: BadUDPClient*
- Giải quyết:
  - Bắt ngoại lệ phương thức ReceiveFrom().
  - Tăng kích thước bộ đệm lần nhận dữ liệu kế tiếp.
  - *Ví dụ: BetterUDPClient*

Ngăn ngừa mất gói tin

- UDP là giao thức phi kết nối
- Khắc phục: dùng cơ chế hồi báo tương tự TCP
- Các kỹ thuật truyền lại gói tin UDP
  - Socket bất đồng bộ (Asynchronous sockets) và đối tương Timer
  - Socket đồng bộ (Synchronous sockets) và thiết lập giá trị Socket Time-out

Socket Time-out

- Phương thức ReceiveFrom() mặc định block chương trình cho tới khi nhận được dữ liệu.
- Sử dụng phương thức SetSocketOption() thiết đặt thời gian chờ nhận dữ liệu (ReceiveTimeout).

```
SetSocketOption(SocketOptionLevel so, SocketOptionName sn, int value)
```

- *Ví dụ: TimeOutUDPClient*

Xử lý biệt lệ

- Hết thời gian Time-out, nếu phương thức ReceiveFrom() không nhận được dữ liệu sẽ phát sinh biệt lệ.
- Đặt phương thức vào khối try-catch để xử lý biệt lệ

Socket Error Codes

- Một vài mã lỗi
  - 10040: Message too long
  - 10054: Connection reset by peer
  - 10060: Connection timed out
- Ví dụ Xử lý lỗi

```
} catch (SocketException e)
{       if (e.ErrorCode == 10054)
        {
                // xử lý lỗi do ReceiveFrom() timeout
        }
        else if (e.ErrorCode == 10040)
        {
                //xử lý lỗi bộ đệm
        }
 }
```

Điều khiển việc truyền lại gói tin

- Tạo một phương thức riêng điều khiển việc gửi và nhận thông điệp:
  1) Gửi thông điệp đến máy ở xa
  2) Chờ trả lời từ máy ở xa
  3) Nếu nhận được trả lời, thoát khỏi phương thức với dữ liệu và kích thước dữ liệu nhận được.
  4) Nếu không nhận được trả lời sau thời gian time-out, thực hiện gửi lại và tăng biến đếm gửi lại.
  5) Kiểm tra biến đếm gửi lại. Nếu nhỏ hơn giá trị qui đinh, lặp lại bước 1, ngược lại thoát khỏi phương thức và thông báo lỗi.

Phương thức gửi, nhận UDP

```
private int SndRcvData(Socket s, byte[] message, EndPoint rmtdevice)
        {
            int recv;
            int retry = 0;
            while (true)
            {
                Console.WriteLine("Attempt #{0}", retry);
                try
                {
                    s.SendTo(message, message.Length, SocketFlags.None,
                     rmtdevice);
                    data = new byte[1024];
                    recv = s.ReceiveFrom(data, ref Remote);
                }
                catch (SocketException)
                {
                    recv = 0;
                }
                if (recv > 0)
                {
                    return recv;
                }
                else
                {
                    retry++;
                    if (retry > 4)
                    {
                        return 0;
                    }
                }
            }
        }
```