LẬP TRÌNH MẠNG

NIC & Multicast

hungdn@ptit.edu.vn

Nội dung

- Lập trình với thẻ giao tiếp mạng (NIC)
  - NIC là gì
  - Lớp NetworkInterface
  - Lập trình với giao tiếp mạng
- Lập trình truyền thông multicast
  - Giới thiệu multicast
  - Lớp MulticastSocket
- Một số ví dụ

1. Network Interface Card (NIC)

1.1 Giới thiệu thẻ giao tiếp mạng

- Một giao tiếp mạng (thường là network interface card - NIC) là điểm liên kết giữa máy tính với mạng (private or public network)
- Giao tiếp mạng
  - Giao tiếp mạng vật lý
  - Giao tiếp mạng phần mềm, ảo
- Lớp NetworkInterface hỗ trợ cho cả 2 loại giao tiếp trên
  - Hữu ích trong các hệ thống có nhiều NIC
  - -> Lựa chọn cụ thể NIC nào sẽ thực hiện một hoạt động mạng cụ thể

Lớp NetworkInterface

- Lớp NetworkInterface cung cấp các phương thức để liệt kê tất cả các địa chỉ cục bộ và tạo ra đối tượng InetAddress từ chúng
- Đối tượng NetworkInterface thể hiện phần cứng vật lý hoặc địa chỉ ảo và chúng không thể được xây dựng tuỳ ý
- Tương tự như lớp InetAdddress, nó cũng có một số phương thức tĩnh (static) cho phép trả về đối tượng NetworkInterface gắn kết với bộ giao tiếp mạng cụ thể

Một số phương thức

- Phương thức *getByName()*

public static NetworkInterface getByName(String name) throws SocketException

```
try {
        NetworkInterface ni = NetworkInterface.getByName("eth0");
        if (ni == null) {
                System.err.println("No such interface:  eth0");
        }
} catch (SocketException ex) {
        System.err.println("Could not list sockets.");
}
```

Một số phương thức (Cont.)

- Phương thức *getByInetAddress()*

public static NetworkInterface getByInetAddress(InetAddress address)
throws SocketException

```
try {
        InetAddress local = InetAddress.getByName("127.0.0.1");
        NetworkInterface ni = NetworkInterface.getByInetAddress(local);
        if (ni == null) {
                System.err.println("That's weird. No local loopback address.");
        }
} catch (SocketException ex) {
        System.err.println("Could not list sockets.");
} catch (UnknownHostException ex) {
        System.err.println("That's weird. No local loopback address.");
}
```

Một số phương thức (Cont.)

- Phương thức *getNetworkInterfaces()*

public static Enumeration getNetworkInterfaces( ) throws SocketException

```
try {
      Enumeration interfaces = NetworkInterface.getNetworkInterfaces( );
      while (interfaces.hasMoreElements( )) {
          NetworkInterface ni = (NetworkInterface)interfaces.nextElement( );
          System.out.println(ni);
      }
  } catch (SocketException ex) {
          System.err.println("Error…");
  }
```

Một số phương thức (Cont.)

- *public Enumeration getInetAddresses( )*
  - Trả về đối tượng `java.util.Enumeration` chứa các đối tượng InetAddress
  - *Ex:*
- *public String getName( )*
  - Trả về tên của đối tượng NetworkInterface cụ thể
  - Ex: eth0 hoặc lo
- *public String getDisplayName( )*
  - Trả về tên hiển thị của một giao tiếp mạng cụ thể
  - Ex: Local Area Connection hoặc Local Area Connection 2

1.2 Lập trình với giao tiếp mạng

- Lấy các giao tiếp mạng
  - *getByInetAddress(), getByName()*
  - *getNetworkInterfaces()*
- Lấy danh sách các địa chỉ giao tiếp mạng
  - *getInetAddresses()*
  - *getInterfaceAddresses()*
- Truy cập các tham số giao tiếp mạng
  - `isUp()` các giao tiếp đang hoạt động
  - `isLoopback()` chỉ thị giao tiếp mạng là một giao tiếp loopback.
  - `isPointToPoint()` chỉ thị nếu giao tiếp là giao tiếp point-to-point.
  - `isVirrtual()` chỉ thị nếu giao tiếp là giao tiếp ảo(giao tiếp mềm)

Ex:

- **Tạo socket gắn với NIC cụ thể**

```
NetworkInterface nif = NetworkInterface.getByName("eth0");
Enumeration<InetAddress> nifAddresses = nif.getInetAddresses();

Socket soc = new java.net.Socket();
soc.bind(new InetSocketAddress(nifAddresses.nextElement(), 0));
soc.connect(new InetSocketAddress(address, port));
```

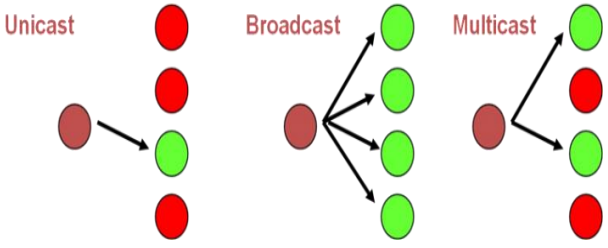
- **Lựa chọn NIC cho multicast group**

```
NetworkInterface nif = NetworkInterface.getByName("eth0");
MulticastSocket ms = new MulticastSocket();
ms.joinGroup(new InetSocketAddress(hostname, port), nif);
```

2. Lập trình truyền thông multicast

2.1 Giới thiệu bài toán multicast

- **Một số thuật ngữ**
  - **Unicast** truyền tin từ 1 điểm tới 1 điểm
  - **Broadcast** truyền tin từ 1 điểm tới tất cả các điểm khác
  - **Multicast** truyền tin từ 1 hoặc nhiều điểm tới 1 tập hợp/ nhóm các điểm khác
- **IP Multicast**
  - Một điểm (sender) gửi gói tin giống nhau tới nhiều điểm khác (Receivers)
    - Chỉ cần gửi gói tin đến một địa chỉ chung của nhóm (multicast group)
    - Các điểm trong nhóm có thể nhận được gói tin này
  - Ưu điểm:
    - Tối ưu băng thông
    - Giảm độ trễ giữa điểm gửi và nhận



Multicast

- **Trong Java, truyền thông multicast tương tự như UDP nhưng dùng không gian địa chỉ lớp D từ 224.0.0.0 đến 239.255.255.255 làm địa chỉ IP multicast group**
- **Xác định các "multicast group"**
  - Tương tự địa chỉ (đích) của một host
  - Phân biệt một đích đến logic hay một group
- **Một vài ứng dụng**
  - Videoconferencing
  - Usenet news
  - Computer configuration
  - Game multiplayer
- **Một số địa chỉ multicast**
  - 224.0.0.1: Tất cả các hệ thống ở trên mạng con cục bộ
  - 224.0.0.2 : Tất cả các router trên mạng con cục bộ.
  - 224.0.0.11: Các tác tử di động(agent) trên mạng con cục bộ
  - 224.0.1.1 : Giao thức định thời mạng
  - 224.0.1.20: Thử nghiệm mà không cho vượt ra khỏi mạng con cục bộ
  - 224.2.X.X (Multicast Backbone on the Internet (MBONE)): Được sử dụng cho audio và video quảng bá trên mạng Internet

2.2 Lập trình multicast

- **Thao tác cài đặt multicasting**
  - Truyền tin theo giao thức UDP
  - Sử dụng giao diện lập trình socket (`MulticastSocket`)
  - Gửi các multicast packet từ một UDP Socket (unicast)
  - Nhận các multicast packet bằng các MulticastSocket
    - Tạo một socket và thực hiện phương thức join()
    - Đóng socket và thực hiện phương thức leave()
- **Vấn đề lưu lượng tăng nhanh khi sử dụng multicast**
- **-> Điều khiển phạm vi của multicast**
  - Dựa trên TTL (trong IP Header)
  - Chỉ những packet có TTL > ngưỡng cho phép mới được tồn tại

Lập trình multicast với Java

- Java hỗ trợ lớp `MulticastSocket` cho phép tạo ra socket thực hiện truyền thông kiểu này. Lớp `MulticastSocket` được kế thừa từ lớp `DatagramSocket`

```
public class MulticastSocket extends DatagramSocket
```

- `MuticastSocket` là một `DatagramSocket` mà thêm khả năng ghép nối gộp nhóm các máy trạm multicast trên mạng Internet
- Một nhóm multicast được chỉ ra bởi địa chỉ lớp D và một địa chỉ cổng UDP chuẩn.
- Lớp MulticastSocket được sử dụng phía bên nhận

MulticastSocket

- Khởi tạo
  - `MulticastSocket()` được gán với cổng bất kỳ
  - `MulticastSocket(int port)` Tạo socket muticast và gắn với socket đó một địa chỉ cổng cụ thể
- Nhiều multicast socket có thể được gắn cùng một port tại cùng một thời điểm (khác với TCP và UDP)
- Có các phương thức kế thừa (send, receive) và 3 phương thức
  - `joinGroup(InetAddress group)`
  - `leaveGroup(InetAddress group)`
  - `setTimeToLive(int ttl)`

Làm việc với lớp MulticastSocket

- Khi một thể hiện (đối tượng) MulticastSocket được tạo, nó có thể thực hiện các hoạt động
  - Tham gia vào một multicast group
    - `public void joinGroup(InetAddress address) throws IOException`
    - `public void joinGroup(SocketAddress address, NetworkInterface interface) throws IOException`
  - Gửi/ nhận dữ liệu đến các thành viên trong group
    - `public void send(DatagramPacket p) throws IOException`
    - `public void send(DatagramPacket packet, byte ttl) throws IOException`
  - Rời khỏi group
    - `public void leaveGroup(InetAddress address) throws IOException`
    - `public void leaveGroup(SocketAddress multicastAddress, NetworkInterface interface) throws IOException`

Một số phương thức quan trọng

- Lấy / thiết lập giao tiếp mạng để gửi nhận dữ liệu
  - `public void setInterface(InetAddress address) throws SocketException`
  - `public InetAddress getInterface() throws SocketException`
  - `public void setNetworkInterface(NetworkInterface interface) throws SocketException`
  - `public NetworkInterface getNetworkInterface() throws SocketException`
- Lấy / thiết lập TTL
  - `getTimeToLive()`
  - `getTTL()`
  - `setTimeToLive(int ttl)`
  - `setTTL(byte ttl)`

Ex: Chat multicast

MulticastServer

```
public class MulticastServer {
  public static void main(String args[]) {
      DatagramSocket socket = null;
      BufferedReader in = null;
      boolean moreQuotes = true;
      try {
          socket = new DatagramSocket();
          while (true) {
              InetAddress group = InetAddress.getByName("224.2.2.3");
              for (int i = 1; i < 1000; i++) {
                  String dString = i + "--" + (InetAddress.getLocalHost());
                  byte[] buf = dString.getBytes();
                  DatagramPacket packet
                          = new DatagramPacket(buf, buf.length, group, 1107);
                  socket.send(packet);
                  Thread.sleep(1000);
              }
          }
      } catch (Exception e) {
      }
  }
}
```

Ex: Chat multicast

MulticastClient

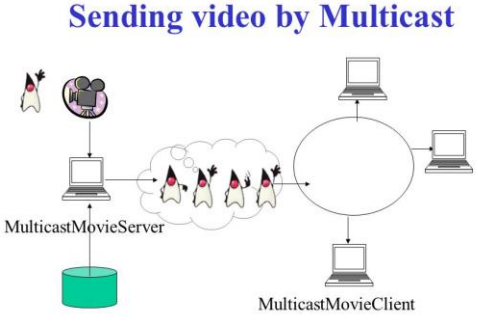
```
public class MulticastClient {
  public static void main(String[] args) throws IOException {
      MulticastSocket socket = new MulticastSocket(1107);
      InetAddress address = InetAddress.getByName("224.2.2.3");
      socket.joinGroup(address);
      byte[] buf = new byte[256];
      DatagramPacket packet;

      while (true) {
          packet = new DatagramPacket(buf, buf.length);
          socket.receive(packet);

          String received = new String(packet.getData());
          System.out.println("Received: " + received);
          try {
              Thread.currentThread().sleep(0);
          } catch (InterruptedException e) {
          }
      }
  }
}
```

Một số ví dụ

- **Gửi video thông qua multicast**
  - MulticastMovieServer.java
  - MulticastMovieClient.java



Một số ví dụ (Cont.)

- **Ứng dụng chat multicast**
  - Không có server ?
  - Mỗi thành viên chạy cùng một ứng dụng và tham gia vào nhóm multicast
  - Thông điệp gửi từ một máy trạm bất kỳ sẽ được gửi cho tất cả các thành viên trong nhóm
  - Không có đảm bảo rằng thông điệp sẽ được gửi tới hoặc không, và chúng được gửi tới đúng thứ tự
- **Ex**
  - MulticastChat.java
- ***Spontaneous Networking***
  - Multicast phù hợp các hệ thống mà thành viên tham gia trong một phiên thường đến và đi ngẫu nhiên
  - Những hệ thống như vậy thường được gọi là *Spontaneous Networking*
- **Ý tưởng**
  - Người mới gia nhập sẽ thông báo cho các thành viên khác biết sự có mặt của mình
  - Người rời khỏi mạng sẽ được ghi nhận bởi các thành viên khác trong mạng

Demo

- **NIC**
  - Enumeration NIC
- **MulticastServer**
  - Create DatagramSocket
  - Create DatagramPacket
  - Send
- **MulticastClient**
  - Create MulticastSocket
  - joinGroup
  - Receive

Exercise

- **Sử dụng lớp NetworkInterface và InetAddress**
  - Liệt kê danh sách các NIC
  - Thông tin chi tiết trên mỗi NetworkInterface (name, InetAddress…)
  - Cấu hình một NIC thực hiện một hành vi mạng nào đó
- **Cài đặt thêm server cho ví dụ ứng dụng chat multicast**
- **Giám sát và hiển thị được trạng thái các client vào ra group**
  - SendThread
  - ReceiveThread
  - CheckThread
  - RefreshThread

Tổng kết

- **NIC**
- **Unicast/Multicast/Broadcast/Anycast**
- **Chủ đề**
  - Video multicast
  - Chat multicast
  - *Spontaneous Networking*

Q & A