CHƯƠNG 11
TỐI ƯU BĂNG THÔNG

ThS. Trần Bá Nhiệm
Website:
sites.google.com/site/tranbanhiem
Email: tranbanhiem@gmail.com

Nội dung

- Giới thiệu
- Các thủ thuật tăng cường hiệu suất
- Multicast UDP
- Nén dữ liệu
- Nén không mất mát thông tin
- Nén có mất mát thông tin

Giới thiệu

- Không phải đường truyền nào cũng đạt tốc độ như LAN
- Khách hàng sẽ chọn những phần mềm đòi hỏi tốc độ thấp nhất và không hỏng khi truyền dở
- Chương này sẽ đề cập đến 2 kỹ thuật tăng cường hiệu suất khác nhau.

Giới thiệu

- Thứ nhất, multicast – là khả năng truyền 1 mảnh dữ liệu cho nhiều người nhận khác nhay đồng thời
- Thứ hai, nén và giải nén dữ liệu. Đó là việc chuyển một khối dữ liệu lớn thành khối nhỏ hơn, sau đó trả về chính xác hoặc “gần như” chính xác dữ liệu gốc

Các thủ thuật tăng cường hiệu suất

- Tăng cường hiệu suất thường là nhờ những thay đổi đơn giản phương pháp di chuyển dữ liệu giữa client và server.
- Trong một số trường hợp không thể áp dụng những kỹ thuật này, tuy nhiên khi dùng thích hợp, sẽ giúp dữ liệu của chúng ta di chuyển nhanh chóng hơn

Caching

- Caching có thể tăng hiệu suất mạng nhờ lưu trữ dữ liệu tĩnh được truy cập thường xuyên tại vị trí mà dữ liệu có thể được đáp ứng nhanh hơn thông thường
- Có 3 tiêu chuẩn sau cần đáp ứng:
  - Dữ liệu phải được truy cập thường xuyên
  - Dữ liệu phải không bị thay đổi thường xuyên
  - Thời gian truy xuất với dữ liệu cache phải nhanh hơn truy xuất trực tiếp

Caching

- Dữ liệu có thể được cache tại bất kỳ điểm nào giữa client và server
- Cache phía server có thể ngăn chặn dữ liệu lỗi thời, nhưng chậm hơn cache phía client
- Cache phía client là nhanh nhất bởi vì dữ liệu đọc từ đĩa, không qua mạng, nhưng chúng có khuynh hướng lỗi thời

Caching

- Proxy cache là kết hợp 2 dạng cache trên. Chúng có thể refresh cache khi rảnh rỗi và có thể phục vụ dữ liệu nhanh hơn bởi vì client kết nối với chúng trên đường truyền LAN. Dữ liệu cũ trên proxy có thể gây khó chịu cho người dùng bởi vì khó xử lý làm sạch cache bằng phương pháp thủ công

Caching

- Cache trên server đặc biệt có ích khi dữ liệu trên đó cần phải xử lý trước khi gửi cho client. Ví dụ như trang ASP.NET đã được tải lên server, nó phải được biên dịch trước khi sinh ra nội dung để gửi trả về cho client. Như vậy server sẽ quá lãng phí thời gian để biên dịch lại trang đó mỗi khi có yêu cầu → dịch sẵn, lưu trong cache

Caching

- Khi một site gồm chủ yếu nội dung tĩnh, có thể cache một bản nén của nó bởi vì phần lớn trình duyệt có thể tự động giải nén nội dung với định dạng phù hợp
- Khi nội dung là động, có thể sử dụng công cụ nén on-the-fly như Xcache và Pipeboost
- Một trong những phương pháp đơn giản nhất để xem dữ liệu có lỗi thời hay không là dùng phương pháp băm

Các kết nối keep-alive

- Cho dù phần lớn các trang web chứa nhiều hình ảnh khác nhau đến từ cùng một server, một số giao thức cũ như HTTP 1.0 đều tạo các kết nối HTTP mới tương ứng mỗi hình. Điều đó là lãng phí vì chỉ cần kết nối đầu tiên đã đủ để truyền tất cả hình ảnh cần thiết

Các kết nối keep-alive

- Hiện nay hầu hết các trình duyệt đều có khả năng quản lý các kết nối bền vững dùng HTTP 1.1
- Client có thể yêu cầu server duy trì kết nối TCP được mở với đặc tả “Connection: Keep-Alive” trong phần HTTP header.

Các kết nối keep-alive

- Khi các kết nối TCP mở và đóng, một số gói tin bắt tay được gửi qua lại giữa client và server, chúng có thể làm mất thời gian trung bình là 1giây trên đường truyền modem. Nếu chúng ta muốn phát triển một giao thức thích hợp bao gồm nhiều tiến trình tuần tự gửi yêu cầu và nhận đáp ứng giữa client với server thì nên giữ kết nối TCP thay cho mở/đóng liên tục sau mỗi yêu cầu

Các kết nối keep-alive

- Vấn đề trễ do bắt tay có thể tránh được hoàn toàn dùng giao thức non-connection-oriented (connectionless) như UDP
- Tuy nhiên như đã đề cập trong chương 3, UDP có thể gây nguy hiểm cho tính toàn vẹn dữ liệu
- Một số giao thức như real-time streaming protocol (RTSP, RFC 2326) có thể đạt được hiệu suất về tốc độ và tin cậy

Progressive downloads

- Khi phần lớn nội dung file đã được tải thì client có thể dùng được dữ liệu trong đó như ứng dụng video, audio
- Kỹ thuật tương tự được áp dụng trong khá nhiều ngữ cảnh, ví dụ như đang liệt kê danh sách sản phẩm thì người dùng có thể dừng ngang nếu sản phẩm cần tìm đã hiện ra

Progressive downloads

- Các dạng thức hình ảnh như JPEG, GIF có thể được hiển thị dần từng phần cho đến khi nội dung file được tải về đầy đủ
- Các byte đến sau dễ dàng nhận thấy có nhiệm vụ là nhằm nâng cao hơn chất lượng hình ảnh
- Kỹ thuật này được gọi là interlacing

Tinh chỉnh các thiết lập

- Windows được tối ưu mặc định cho việc sử dụng trên Ethernets, vì vậy khi ứng dụng nào đó có truy cập thông qua modems, ISDN, DSL thì cần phải có tinh chỉnh các thiết lập để kết nối hiệu quả hơn, tăng hiệu suất toàn mạng
- Các thiết lập TCP/IP trong registry tại: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters

Tinh chỉnh các thiết lập

- Chỉnh sửa TCP window size trong registry tại: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\GlobalMaxTcpWindowSize
- TCP window size cho biết số byte mà máy tính có thể truyền chưa cần nhận ACK, nên là 256960 (một số giá trị khác 372300, 186880, 93440, 64240, 32120)

Tinh chỉnh các thiết lập

- Vùng giá trị hợp lệ cho TCP window size là từ giá trị maximum segment size (MSS) đến 2^{30}
- Kết quả tốt nhất là bội số của MSS và nhỏ hơn 65535 lần một hệ số (là lũy thừa của 2)
- MSS thông thường gần bằng giá trị maximum transmission unit (MTU)

Tinh chỉnh các thiết lập

- Giá trị time-to-live (TTL) của gói tin có ý nghĩa cực kỳ quan trọng.
- TTL cho biết số router tối đa mà gói tin có thể đi qua trước khi bị hủy. Nếu cao quá sẽ gây trễ, còn thấp quá sẽ làm cho gói tin bị hủy trước khi đến được đích. Giá trị này nên là 64.
- Chỉnh sửa TTL trong registry tại: KLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DefaultTTL

Tinh chỉnh các thiết lập

- MTU là kích thước tối đa mà các gói tin có thể được truyền trên mạng có dây. Nếu giá trị này quá cao, các gói mất sẽ truyền lại lâu hơn và phải phân mảnh. Nếu quá thấp sẽ gây quá tải và tốn thời gian truyền nhiều hơn. Với Ethernet nên là 1500 byte/gói, ADSL 1492 byte/gói, FDDI 8000 byte/gói
- Chỉnh sửa: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\EnablePMTUDiscovery
- Giá trị khuyến cáo là 1

Tinh chỉnh các thiết lập

- Mỗi mảnh datagram được truyền có kích thước bằng MTU. Nếu lớn hơn MTU, nó sẽ phân mảnh, gây ra tốn kém thời gian tính toán, tăng nguy cơ mất dữ liệu.

Tinh chỉnh các thiết lập

- Một router lỗ đen là router bị lỗi khi chuyển gói đi và không thông báo lại cho người gửi với thông điệp ICMP. Nếu nó không phải là vấn đề đối với mạng thì có thể bỏ qua
- Chỉnh sửa: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\EnablePMTUBHDetect
- Giá trị khuyến cáo là 0

Tinh chỉnh các thiết lập

- Selective Acknowledgement (SACK) cho phép cải thiện hiệu suất truyền khi kích thước window nhỏ
- Chỉnh sửa: HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\SackOpts
- Giá trị khuyến cáo là 1

Tinh chỉnh các thiết lập

- Tham số xác định số lượng thông báo trùng lặp có thể nhận trước khi "truyền lại nhanh" được kích hoạt để gửi lại segment nào đã bị bỏ trong quá trình truyền
- Chỉnh sửa: KLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\TcpMaxDupAcks
- Giá trị khuyến cáo là 2
- Thiết lập này đặc biệt quan trọng đối với các đường truyền có nguy cơ mất gói tin cao

Tinh chỉnh các thiết lập

- Tăng tốc trình duyệt web, cải thiện hiệu suất các kết nối HTTP ra bên ngoài bằng cách tăng số lượng kết nối đồng thời
- Chỉnh sửa:

HKEY_USERS\.DEFAULT\Software\Microsoft\Windows\CurrentVersion\Internet Settings\
"MaxConnectionsPerServer"=dword:00000020
"MaxConnectionsPer1_0Server"=dword:00000020
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\
"MaxConnectionsPerServer"=dword:00000020
"MaxConnectionsPer1_0Server"=dword:00000020

Multicast UDP

- Multicasting: thông điệp có thể đi đến nhiều hơn 1 đích tại cùng thời điểm
- Cung cấp cơ chế tăng hiệu suất rõ ràng với trường hợp có nhiều người nhận dữ liệu
- Đặc biệt thích hợp với mạng mà client và server cùng LAN, trường hợp có thể route trên Internet

Multicast UDP

- Một số nhà cung cấp dịch vụ không hỗ trợ Multicasting
- Địa chỉ Multicasting nằm trong khoảng 224.0.0.0 đến 239.255.255.255, nhưng chúng ta không thể tùy ý sử dụng bởi vì có một số hạn chế
- Tổ chức IANA điều hành các địa chỉ IP multicast này

Multicast UDP

- 224.0.0.0 đến 224.0.0.255: Local Network Control Block, không thể route và tìm đến được trên Internet, chúng có những mục đích đặc biệt, ví dụ: DHCP tại 224.0.0.12
- 224.0.1.0 đến 224.0.1.255: Internetwork Control Block, có thể route, nhưng chúng được dùng cho mục đích đặc biệt, ví dụ: Network time protocol (NTP) tại 224.0.1.1, WINS tại 224.0.1.24.

Multicast UDP

- 239.0.0.0 đến 239.255.255.255: gọi là các địa chỉ scope-relative, không thể route, chúng không có những mục đích đặc biệt, có thể được sử dụng tùy ý để thử nghiệm
- Chúng ta có thể yêu cầu một địa chỉ IP multicast duy nhất trên toàn cầu từ IANA. Trước tiên, dùng một địa chỉ trong khoảng cho phép thử nghiệm, ví dụ 234.5.6.11

Multicast UDP

- Hoặc thu được địa chỉ multicast thuê riêng từ multicast address dynamic client allocation protocol (MADCAP), như đã định nghĩa trong RFC 2730
- Nếu có cùng người khác dùng cùng địa chỉ multicast như chúng ta, hiện tượng nhận được các gói tin lang thang có thể làm hỏng dữ liệu chúng ta muốn truyền

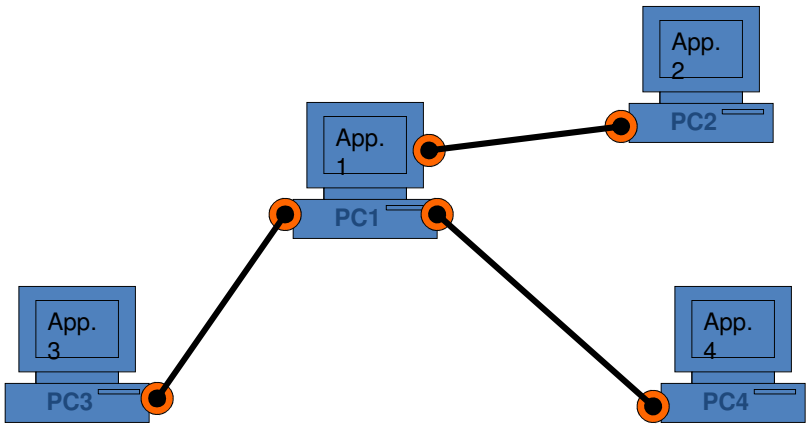
Multicast UDP

- Nếu broadcasting trong LAN, dùng 1 địa chỉ scope-relative
- Khi broadcasting trong WAN (chứ không phải trên Internet), chúng ta có thể giới hạn TTL của gói tin với giá trị < 63. TTL ngăn gói đi lòng vòng không xác định. Mỗi hop giảm TTL xuống 1 đơn vị. Khi TTL = 0, gói tin sẽ bị hủy

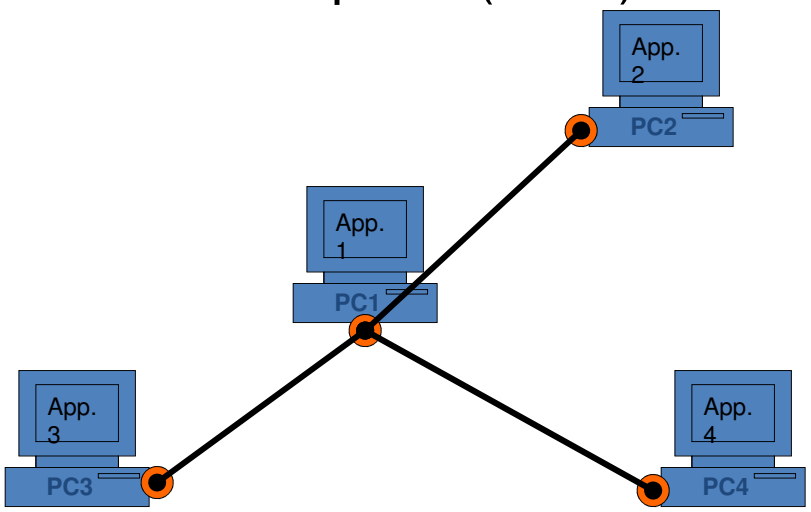
Multicast UDP

- Cần phân biệt broadcasting và multicasting, multipoint và unipoit
- Broadcasting: truyền đến tất cả các client trong phạm vi
- Multicasting: truyền đến một số client trong phạm vi

Point-to-point (TCP / UDP)



Multi-point (UDP)



Multicast UDP

- Multicast UDP có thể là giao thức không-P2P đầu tiên có thể lập trình được
- Giới hạn lớn nhất của network broadcasts là chỉ làm việc trên cùng LAN và không thể route trên Internet
- Để cho phép các người cung cấp dịch vụ chấp nhận dùng multicast để truyền thông, cần có multicast backbone (MBONE)

Multicast UDP

- MBONE hiện tại được sử dụng trên 24 quốc gia, phần lớn trong các mạng thuộc trường đại học
- Multicast hàm ý rằng dữ liệu được truyền trên tất cả các hướng (flood) nhưng trên thực tế không phải tất cả các gói UDP đều dùng flood

Multicast UDP

- Có 3 giao thức multicast routing:
  - distance vector multicast routing (DVMRP)
  - multicast open shortest path first (MOSPF)
  - protocol independent multicast (PIM)
- Không có multicast TCP tương đương vì lý do yêu cầu thiết lập bắt tay 3 bước. Điều đó gây khó khăn cho người phát triển ứng dụng vì dữ liệu gửi bởi UDP có thể hư hỏng do mất, trùng lặp và sắp xếp lại

Multicast UDP

- Vấn đề hư hỏng trên có thể khắc phục bằng cách chèn thêm header vào dữ liệu chứa số tiến trình, giúp cho client tổ chức lại hoặc yêu cầu quá trình truyền lại các gói tin bị thiếu từ server
- Tương tự, khó hiện thực bảo mật khóa chung/riêng thông qua multicast bởi vì mỗi client đều có khóa chung khác nhau → dùng cơ chế bảo mật IETF thay thế

Hiện thực multicast

- Trước khi thử nghiệm, phải bảo đảm là kết nối Internet hỗ trợ multicast traffic và nối với mạng MBONE
- Ví dụ này bao gồm 2 ứng dụng: bên gửi và bên nhận
- Thực hiện ứng dụng phía gửi như trình bày ở sau:

Hiện thực multicast

- Tạo project mới, 1 form, 3 textbox: tbMulticastGroup, tbPort, tbMessage và 1 nút lệnh btnSend
- Xử lý sự kiện Click:

```
private void btnSend_Click(object sender,
System.EventArgs e)
{
  send(tbMulticastGroup.Text, int.Parse(tbPort.Text),
  tbMessage.Text);
}
```


Hiện thực multicast

- Hoạt động multicast được thực thi tại cả mức socket và mức UdpClient. Do vậy, để minh họa cho phong phú, chúng ta cho bên gửi hiện thực socket, còn bên nhận hiện thực UdpClient
- Trước khi gửi và nhận từ nhóm multicast cần phải gia nhập vào nhóm → dùng tùy chọn AddMembership của socket

Hiện thực multicast

- Giống như cách mà socket hoạt động ở chế độ point-to-point (unicast), điểm endpoint từ xa phải được xác định bằng địa chỉ IP + port. Địa chỉ IP trong trường hợp này phải nằm trong vùng 224.0.0.0 đến 239.255.255.255. TTL phải thiết lập giá trị lớn nhất, bằng 255
- Hiện thực hàm send

Hiện thực multicast

```
private void btnSend_Click(object sender, EventArgs e)
{
    string mcastGroup = tbMulticastGroup.Text;
    int port = int.Parse(tbPort.Text);
    string message = tbMessage.Text;
    IPAddress ip = IPAddress.Parse(mcastGroup);
    Socket s = new Socket(AddressFamily.InterNetwork,
     SocketType.Dgram, ProtocolType.Udp);
     s.SetSocketOption(SocketOptionLevel.IP,
SocketOptionName.AddMembership,
new MulticastOption(ip));
```

Hiện thực multicast

```
    s.SetSocketOption(SocketOptionLevel.IP,
      SocketOptionName.MulticastTimeToLive, 255);
    byte[] b;
    b = Encoding.ASCII.GetBytes(message);
    IPEndPoint ipep = new
IPEndPoint(IPAddress.Parse(mcastGroup), port);
    s.Connect(ipep);
    s.Send(b, b.Length, SocketFlags.None);
    s.Close();
}
```


Hiện thực multicast

- Trong phần code trên, chúng ta dùng socket để truyền dữ liệu multicast chứ không phải stream
- SocketOptionName.AddMembership chỉ ra socket gắn vào nhóm multicast
- Message ở dạng string phải chuyển sang mảng byte
- Endpoint được cài địa chỉ multicast trên port xác định

Hiện thực multicast

- Socket sau đó được kết nối với endpoint, gửi mảng byte, sau đó đóng kết nối
- Để hoàn thành phần code cho server, tất nhiên phải khai báo các namespace cần thiết sau:

```
using System.Net;
using System.Net.Sockets;
using System.Text;
```


Hiện thực multicast

- Bước tiếp theo là hiện thực project phần bên nhận.
- Tạo project mới, gồm: 1 form, 1 textbox tên tbMessages với thuộc tính multiline = true
- Xử lý sự kiện Form_Load():

Hiện thực multicast

```
private void Form1_Load(object sender,
System.EventArgs e)
{
    Thread thdReceiver = new Thread(new
ThreadStart(receiverThread));
    thdReceiver.Start();
}
```


Hiện thực multicast

```
public void receiverThread()
{
    UdpClient client = new UdpClient(5000);
    IPAddress group =
IPAddress.Parse("224.5.6.7");
    int timeToLive = 255;
    int port = 5000;
    client.JoinMulticastGroup(group, timeToLive);
    IPEndPoint remoteEP = new
IPEndPoint(group, port);
```

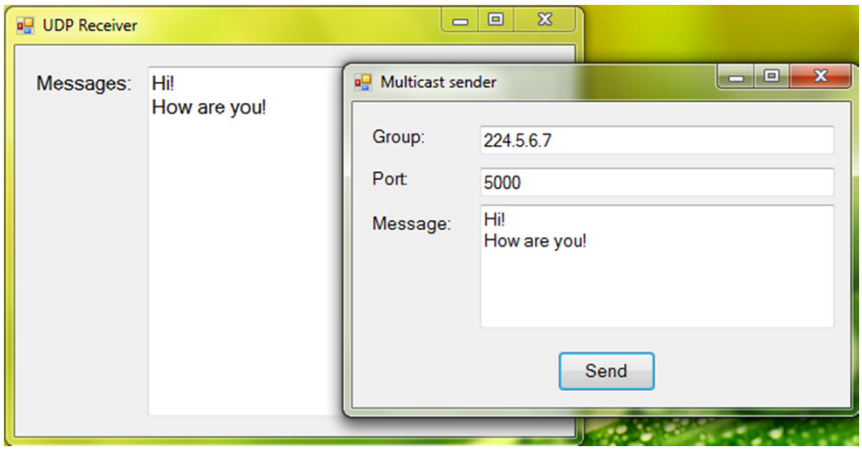
Hiện thực multicast

```
    while (true)
    {
        IPEndPoint ep = null;
        byte[] buffer = client.Receive(ref ep);
        string message =
Encoding.ASCII.GetString(buffer);
        string s = message + "\n";
        InfoMessage(s);
    }
}
```


Hiện thực multicast

```
public void InfoMessage(String info)
{
    if (tbMessages.InvokeRequired)    {
        InfoMessageDel method = new
InfoMessageDel(InfoMessage);
        tbMessages.Invoke(method, new object[] { info
});
        return;
    }
    tbMessages.Text += info;
}
```

Hiện thực multicast



Nén dữ liệu

- Phương pháp hiệu quả để gửi dữ liệu nhanh hơn giữa các máy tính là gửi ít, điều này không có nghĩa là gửi thiếu thông tin mà thực tế là nén lại
- Tiến trình nén và giải nén dữ liệu về như ban đầu được gọi là nén không mất mát - lossless compression (đã được dùng trong dạng nén ZIP)

Nén dữ liệu

- Tiến trình nén và giải nén dữ liệu về gần chính xác như ban đầu nhưng không hoặc ít cảm thấy khác biệt gọi là nén chấp nhận mất mát - lossy compression (đã được dùng trong dạng JPEG hoặc MP3)

Nén không mất mát

- Có 2 cách nén không mất mát:
  - Entropy encoding
  - Source encoding
- Entropy encoding là phương pháp thống kê độ tương tự giữa các byte hoặc các dãy byte chứ không phải bản thân các byte dữ liệu
- Source encoding: xem xét tỷ lệ thay đổi giữa các byte hoặc các dãy byte chứ không phải bản thân các byte dữ liệu

Nén không mất mát

- Entropy encoding dùng trong ZIP
- Source encoding dùng trong delta pulse code modulation (ADPCM) – một kỹ thuật nén audio
- Dạng cơ bản nhất của Entropy encoding là run length encoding (RLE), trong đó dãy byte gồm toàn bộ các byte giống nhau được chuyển thành dữ liệu số và byte đó

Nén không mất mát

- Ví dụ RLE: chuỗi dưới dạng thập lục phân 00 00 00 00 00 được chuyển thành 05 00
- Phương pháp tiếp cận này đạt hiệu quả chỉ khi các file có entropy rất cao
- Một phương pháp khá hiệu quả khác của ZIP là nén Huffman. Các byte ít phổ biến được mã hóa thành các dãy bit dài hơn 1 byte, nhưng vì chúng có số lượng nhỏ nên phương pháp này vẫn hiệu quả

Nén không mất mát

- Bảng chuyển đổi bit-code-to-byte được gọi là codebook, chúng có thể ở 2 dạng là static và dynamic.
- Vì codebook được thêm vào file truyền nên nó phải nhỏ hoặc không có đối với static codebook. Không cần truyền codebook với dữ liệu vì bên nhận đã có nó

Nén không mất mát

- Các static codebook đã có trong một vài năm gần đây, cũng có thể có từ khi xuất hiện máy tính
- Sơ đồ nén đầu tiên là mã Morse
- Có thể những người thiết kế mã Morse đã nghĩ đến việc giảm bớt entropy
- Mã Morse không áp dụng cho nén dữ liệu trên máy tính vì nó dùng ký hiệu dừng để phân tách, ký tự này không thể hiện được dưới dạng nhị phân

Nén không mất mát

- Dynamic codebook được xây dựng trong suốt quá trình nén, khi đó các ký tự phổ biến được xác định và sau đó được gán chuỗi bit
- Codebook được dùng để nén các byte dữ liệu vào các chuỗi bit ngắn hơn – từ đó hình thành dòng byte ngắn hơn dữ liệu nguyên thủy

Nén không mất mát

- Codebook có thể xây dựng tùy ý.
- Chúng phải phản ánh tần số của mỗi ký tự trong dữ liệu và dễ dàng có thể phân tách.
- Dạng đơn giản nhất là gán chuỗi 2-bit cho ký tự phổ biến (ví dụ: 01)
- Mỗi byte đi sau ký tự này được thể hiện bởi thêm bit 1 hoặc 00. Ví dụ với tiếng Anh thì khoảng trắng có tần số cao nhất, được gán 01, ‘e’ là 011, ‘t’ là 0100,…

Nén không mất mát

- Với phương pháp như vậy, chuỗi “e et” (gồm 6 byte) được biểu diễn thành 0110101010110100 (chỉ chiếm 2 byte).
- Tiến trình xây dựng Huffman codebook (hoặc “Huffman tree”) xem thêm tài liệu môn học Lý thuyết thông tin của cùng tác giả http://sites.google.com/site/tranbanhiem

Hiện thực nén với ZIP

- Tạo project mới, 1 form, 2 textbox tên tbInput, tbOutput và 3 button với tên btnCompress, btnBrowseInput, btnBrowseOutput,1 NumericUpdown với tên numericUpDown
- Chọn Projects→Add References→Browse và chọn SharpZipLib.dll từ thư mục #ZipLib đã được cài đặt sẵn (tải về từ www.icsharpcode.net)

Hiện thực nén với ZIP

```
private void btnBrowseInput_Click(object sender,
EventArgs e)
{
    openFileDialog.ShowDialog();
    tbInput.Text = openFileDialog.FileName;
}
private void btnBrowseOutput_Click(object sender,
EventArgs e)
{
    saveFileDialog.ShowDialog();
    tbOutput.Text = saveFileDialog.FileName;
}
```


Hiện thực nén với ZIP

- Giải thuật chính nằm ở phần code xử lý cho nút lệnh Compress.
- Các file ZIP có thể chứa hơn 1 file nguồn, CRC và thông tin ngày tháng với mỗi file để giúp bảo vệ tính toàn vẹn
- Checksum (hoặc CRC) tương tự giá trị băm nhưng để kiểm tra tính toàn vẹn chứ không phải nhãn bảo mật

Hiện thực nén với ZIP

- ZipOutputStream được gắn vào mỗi đối tượng ZipEntry
- SetLevel được dùng để định nghĩa mức độ của nén dữ liệu, trong đó 0 là không nén và 9 là nén tối đa

Hiện thực nén với ZIP

```
private void btnCompress_Click(object sender, EventArgs e)
{
   int ziplevel = int.Parse(numericUpDown.Value.ToString());
   Crc32 crc = new Crc32();
   using (ZipOutputStream ZipStream = new
ZipOutputStream(File.Create(tbOutput.Text)))
   {
      ZipStream.SetLevel(ziplevel);
      string file = tbInput.Text;
      FileStream fs = File.OpenRead(file);
      byte[] buffer = new byte[fs.Length];
```

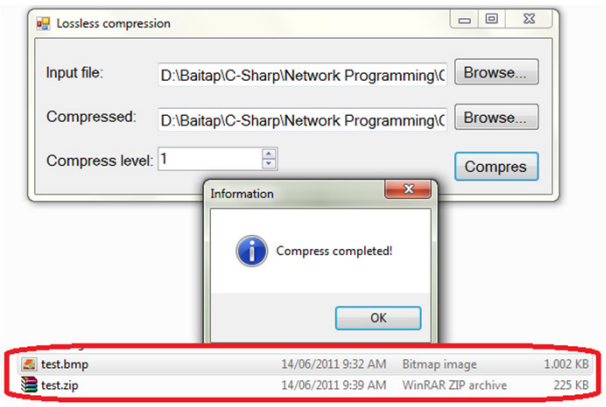
Hiện thực nén với ZIP

```
ZipEntry entry = new ZipEntry(file);
 entry.DateTime = DateTime.Now;
 entry.Size = fs.Length;
 fs.Close();
 crc.Reset();
 crc.Update(buffer);
 entry.Crc = crc.Value;
 ZipStream.PutNextEntry(entry);
```


Hiện thực nén với ZIP

```
        ZipStream.Write(buffer, 0, buffer.Length);
        ZipStream.Finish();
        ZipStream.Close();
        MessageBox.Show("Compress
completed!", "Information",
MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
}
```

Hiện thực nén với ZIP



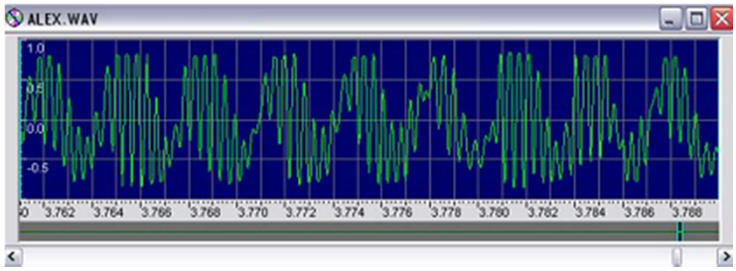
Nén có mất mát thông tin

- Trong trường hợp không cần yêu cầu toàn vẹn dữ liệu thì nén có mất mát là lựa chọn hợp lý
- Đây là lựa chọn cho nén dữ liệu audio, video

Nén dữ liệu audio

- File dữ liệu audio có đặc tính byte-to-byte entropy rất thấp, nên nén kiểu ZIP hoặc Huffman sẽ cho hiệu suất thấp
- Dữ liệu audio được tạo bởi các sóng âm. Mỗi mẫu sóng âm rất giống với mẫu trước đó. Tốc độ thay đổi mẫu theo hướng tăng/giảm là điều hòa, do đó thay vì ghi giá trị của mỗi mẫu thì ghi tốc độ thay đổi

Nén dữ liệu audio



Nén dữ liệu audio

- Trong giải thuật DPCM, sự tăng/giảm giá trị mẫu được thể hiện bởi bit 1/0. Khi giải nén, giá trị mẫu cũng được tăng/giảm bởi 1, phụ thuộc vào bit hiện tại trong dòng bit
- Cách trên cũng gây ra 2 thiệt hại:
  - Slope overload
  - Nhiễu granular

Nén dữ liệu hình ảnh

- Nén hình ảnh khá tương tự với nén audio ngoại trừ xử lý trên cả 2 chiều so với 1 chiều của audio
- Trong quá trình nén, hình ảnh được chia thành các khối macroblock hoặc khối 8x8 cho mỗi pixel. Mỗi macroblock dùng một DCT 2 chiều để cô lập và giảm số lượng màu trong vùng

Nén dữ liệu hình ảnh

- Biểu diễn toán học của DCT 2 chiều:

$$S(v,u)=\frac{Cv}{2}\frac{Cu}{2}\sum_{y=0}^{7}\sum_{x=0}^{7}f(y,x)\cos\frac{(2x+1)\pi u}{16}\cos\frac{(2y+1)\pi v}{16}$$

- Trong đó: Cu = 0.7071
- Công thức này sinh ra một mảng 2 chiều, sau đó được nén bằng cách cho các giá trị gần bằng 0 về 0, tiếp đó dùng nén RLE, nén Huffman

Hiện thực nén dữ liệu hình ảnh

- May mắn là chúng ta không phải hiện thực giải thuật nén vì .NET đã hỗ trợ cho JPEG, cũng như hàng chục định dạng khác nữa như: PNG, TIFF, GIF
- Tạo project mới, gồm 1 form, 1 picture box tên pictureBox, 2 textbox tên tbInput, tbOutput, 3button tên btnBrowseInput, btnBrowseOutput, btnCompress

Hiện thực nén dữ liệu hình ảnh

- Để thực hiện việc nén, đơn giản chỉ gọi phương thức Save của đối tượng Image chúng ta tạo ra:

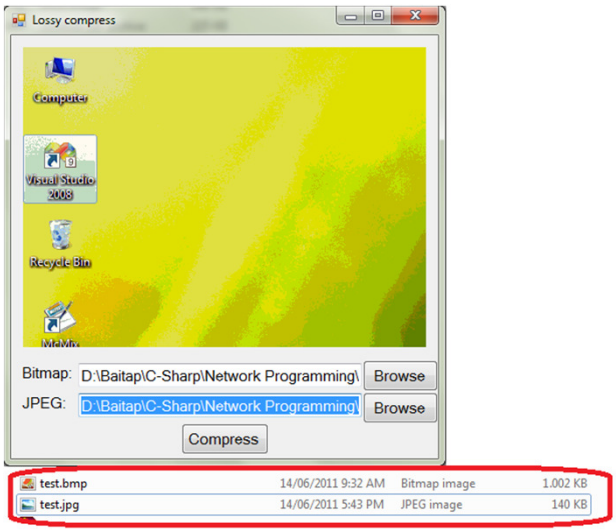
```
private void btnCompress_Click(object sender,
EventArgs e)
{
    FileStream fs = new FileStream(tbOutput.Text,
FileMode.CreateNew);
    pictureBox.Image.Save(fs,
System.Drawing.Imaging.ImageFormat.Jpeg);
    fs.Close();
}
```


Hiện thực nén dữ liệu hình ảnh

```
private void btnBrowseInput_Click(object sender, EventArgs e)
{
   openFileDialog.ShowDialog();
   tbInput.Text = openFileDialog.FileName;
   pictureBox.Image =
Image.FromFile(openFileDialog.FileName);
}

private void btnBrowseOutput_Click(object sender, EventArgs e)
{
   saveFileDialog.ShowDialog();
   tbOutput.Text = saveFileDialog.FileName;
}
```

Hiện thực nén dữ liệu hình ảnh



Nén dữ liệu video

- Nếu không thực hiện nén dữ liệu thì băng thông đường truyền video hiện tại không thể đáp ứng được
- Một trong những chuẩn nén thành công nhất là Motion Pictures Expert Group (MPEG).
- Chuẩn nén tương đối tốt, xếp sau, là audio-video interleaved (AVI)

Nén dữ liệu video

- AVI của công nghệ của Microsoft nên được tích hợp vào Windows API
- Một nguồn tài nguyên tham khảo rất tốt cho lập trình với AVI file là: www.shrinkwrapvb.com
- Nén video tương tự nén audio, ngoại trừ video có 3 kênh dữ liệu hình ảnh (mỗi pixel ảnh là tổ hợp của 3 màu: đỏ, xanh lục, xanh lá RGB), 1 kênh cho audio.

Nén dữ liệu video

- Một trong những kỹ thuật nén quan trọng là subsampling
- Subsampling: chuyển từ định dạng RGB sang YUV. YUV định nghĩa mỗi màu là tổ hợp của luminance và chrominance. Chrominance định nghĩa màu từ đỏ sang xanh lá. Luminance định nghĩa độ xám của màu

Nén dữ liệu video

- Vì luminance thay đổi thường xuyên hơn chrominance nên ít dữ liệu màu được gửi
- Khi hiện tượng này áp dụng cho nén, các mức chrominance được cập nhật mỗi frame, trái lại các mức bão hòa chỉ cập nhật ở một số frame.
- Với chuẩn H.261 tỷ số của lấy mẫu chrominance:luminance là 4:1

Bài tập

- Cài đặt các chương trình đã minh họa trong bài giảng của chương bằng ngôn ngữ C# hoặc VB.NET