Network Programming

Ung Văn Giàu
Email: giau.ung@eiu.edu.vn

Optimizing Bandwidth Utilization

Content

- Introduction

- Tricks and tips to increase performance

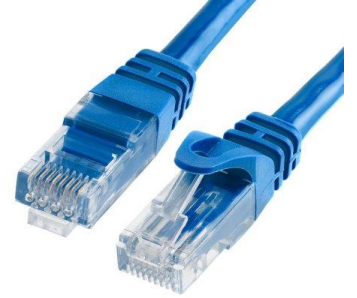- Multicast UDP

- Data compression

1. Introduction

- You can't always expect your customer to have the same bandwidth as your office LAN

- These customers will only buy your software if it works at a speed that is at least usable and does not frustrate them

Performance-enhancing Techniques

Internet Connection Types

1. Introduction

- Two performance-enhancing techniques:

  - **Multicast**

    The ability to send one piece of data to more than one recipient simultaneously

  - **Data compression and decompression**

    The ability to convert a block of data into a smaller block of data and then return this

to either an exact or near copy of the original data

2. Tricks and tips to increase performance

By simple changes to how data is moved between client and server

2.1. Caching

- Stores frequently accessed static data → provides faster data return

- Needs to meet **3** of the following **criteria**:
  - The data must be **frequently accessed**
  - The data must **not often change**
  - The **access time** for cached data must be **faster** than the access time to **receive** the data **directly**

2.1. Caching

- Data can be cached at any point between the client and server:

  - **Server-side caches**
    - ✓ protect against out-of-date data
    - ✓ slower

  - **Client caches**
    - ✓ very fast because reading from disk
    - ✓ prone to out-of-date data

2.1. Caching

- Data can be cached at any point between the client and server:

  - **Proxy caches**

    - ✓ a combination of the two

    - ✓ can refresh their cache regularly

    - ✓ serve data faster (on LAN)

    - ✓ it is awkward to flush the cache of a proxy server manually

2.1. Caching

- **Server caching** can be extremely useful when data needs to be processed before sent to clients

  ASP.NET page, PHP page,…

- When a site consists of mainly **static content**, it is possible to cache a **compressed version**

- When the **content** is **dynamic**, it is possible to utilize on-the-fly compression

  Xcache, APC, eAccelerator, Memcached

- **Hashing** could be used to monitor changes

2.2. Keep-alive connections

- Most Web pages contain many images coming from the same server

- Some older clients create new HTTP connections for each of the images → wasteful
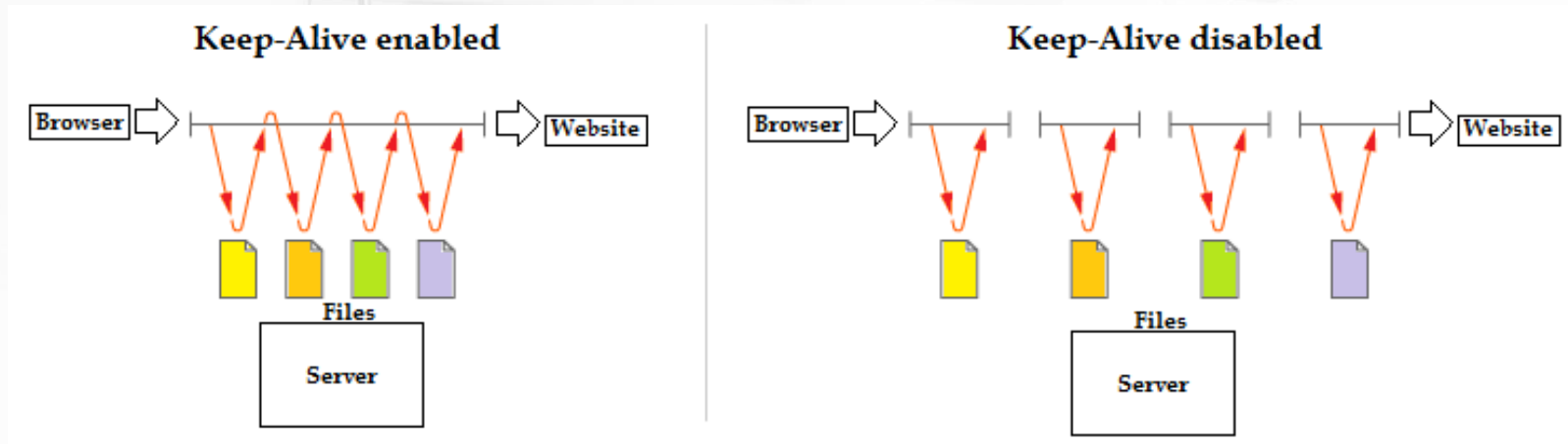
2.2. Keep-alive connections

- Most browsers and servers are capable of handling HTTP 1.1 persistent connections

- A client can request a server to keep a TCP connection open by specifying **Connection: Keep-Alive** in the HTTP header

2.2. Keep-alive connections

- When a TCP connection opens and closes, several handshake packets are sent back and forth → waste time → keep the TCP connection open for as long as possible

- The whole handshake **latency issue** can be **avoided** completely by using a connectionless protocol

- **Real-time streaming protocol** (RTSP) uses a combination of TCP and UDP to achieve a compromise between speed and reliability
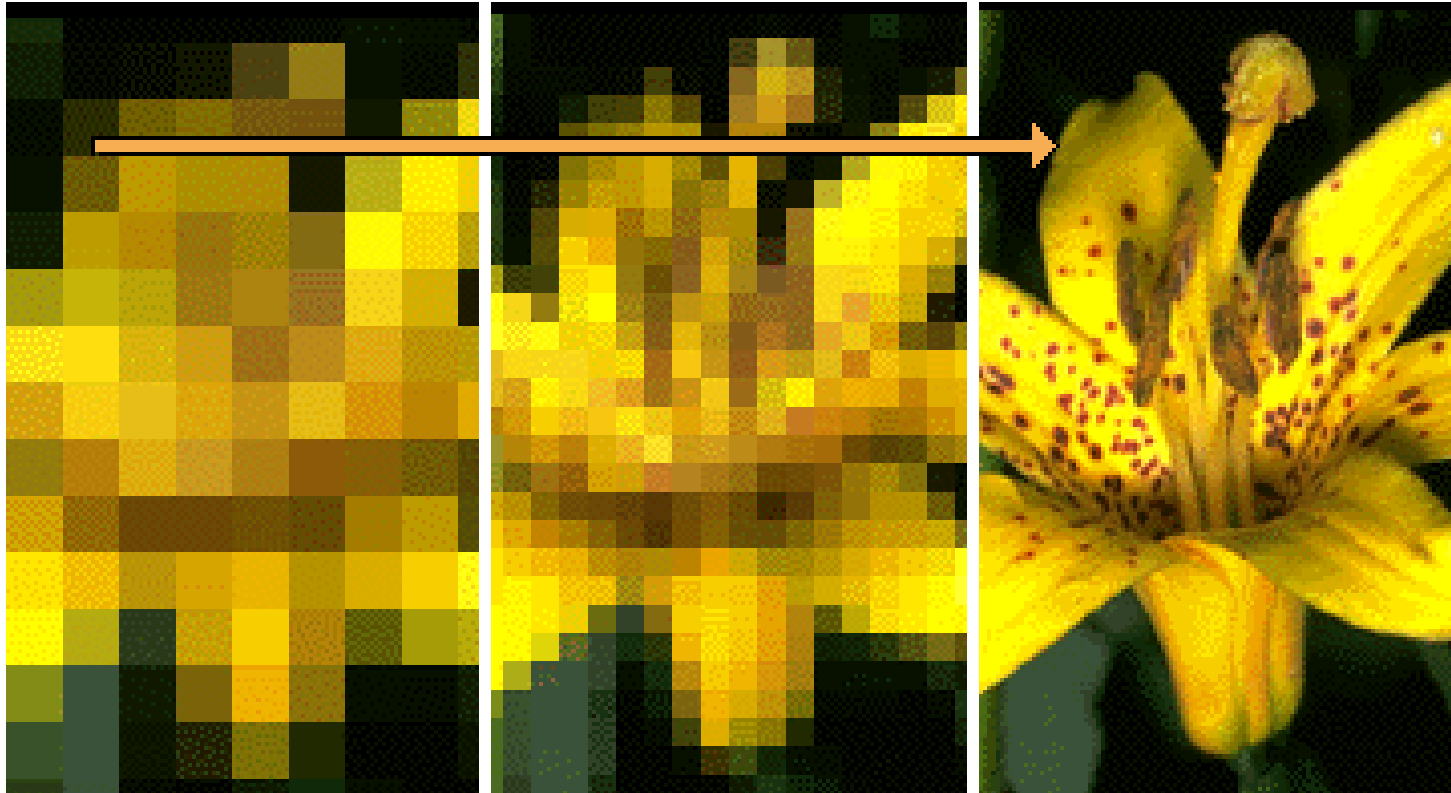
2.3. Progressive downloads

- When most of a file is downloaded, the client should be able to begin to use the data
   audio and video

- The same technique is applicable in many scenarios
   if product listings are being displayed, a user could interrupt the process once the
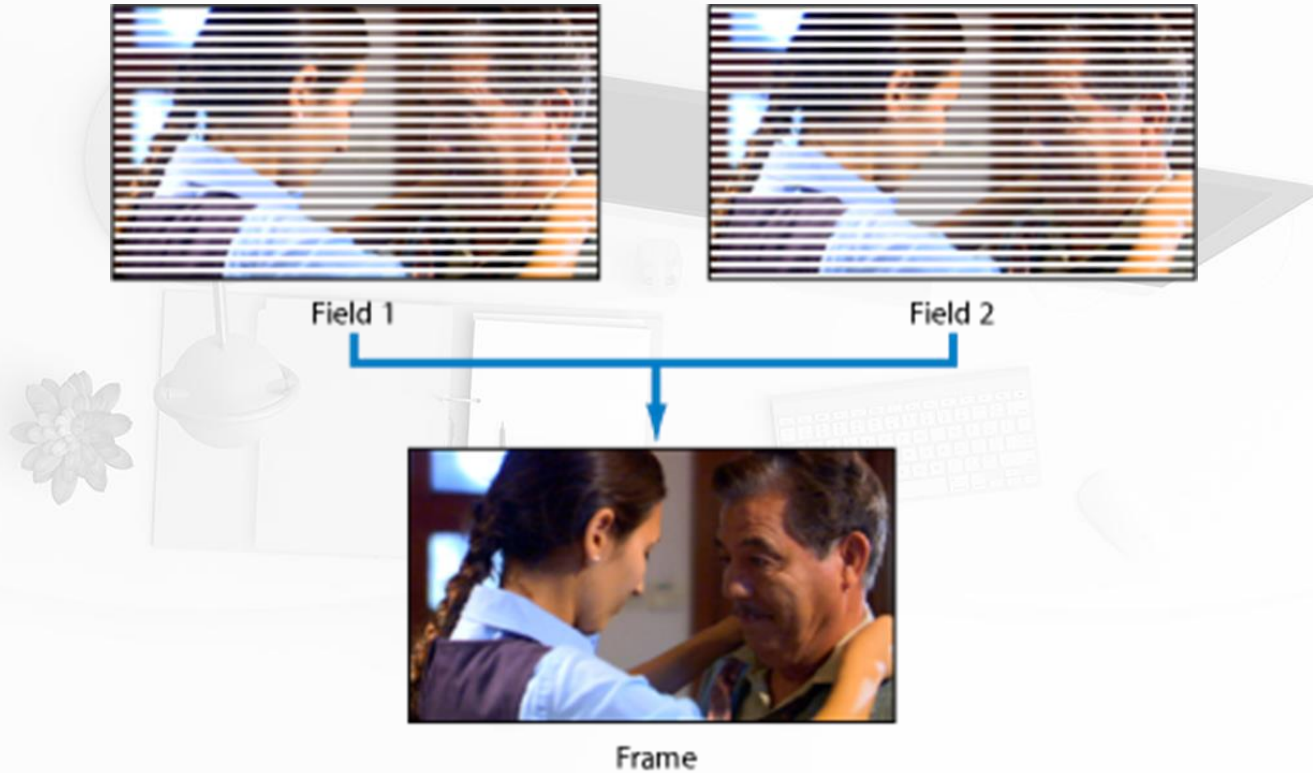desired product is shown

2.3. Progressive downloads

- Image formats renders them as full-size images very soon after the first few hundred bytes are received

- Subsequent bytes form a more distinct and higher-quality image

- This technique is known as **interlacing**

Interlacing

Interlacing

Field 1

Field 2

Frame

2.3. Progressive downloads

- In an online catalog application:

  - product names and prices download first,

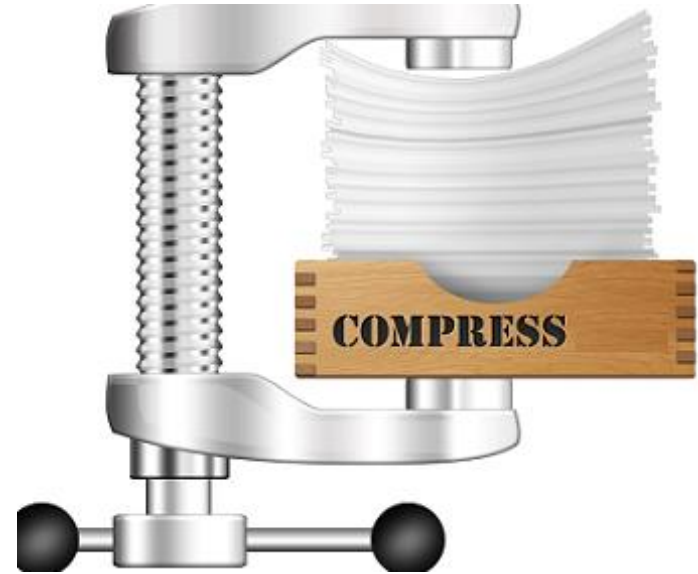 - followed by the images of the various products

3. Multicast UDP

- **Multicasting** is where a message can travel to more than one destination at the same time

- It is ideally suited to networks where all clients and servers are on the same LAN

  only supported by some service providers

4. Data compression

4. Data compression

- The most effective way to send data between computers faster is to send less data

- **Lossless compression**

    The process of compressing data so that the decompressed data is identical to the original
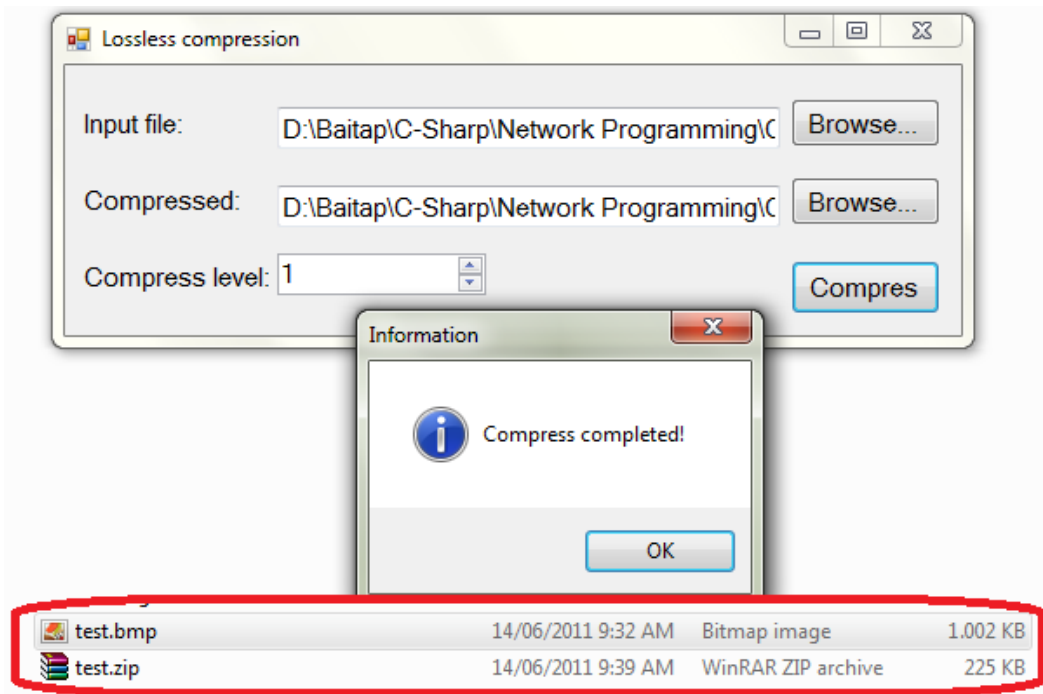
- **Lossy compression**

    The process of compressing data in a way that is not identical, but is not perceived as different from the original

4.1. Lossless compression

- Used when the integrity of data is paramount

- There are 2 ways:

  - Entropy encoding: ZIP

  - Source encoding: Audio compression technique
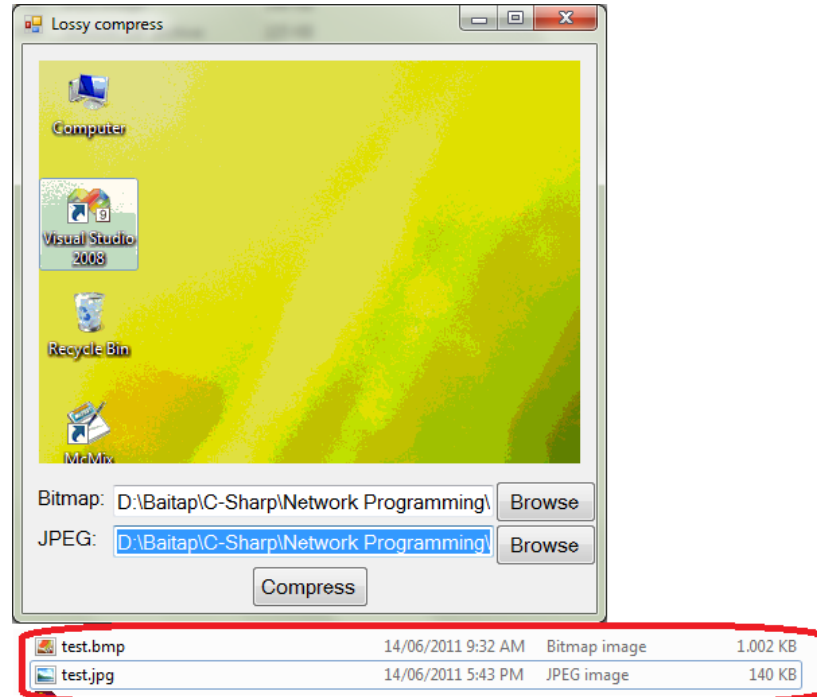
Exercise

Use ZIP to compress and decompress data.

4.2. Lossy compression

- In cases where data integrity is not as important

- This is particularly pertinent to audio and visual data

- Users accept a little muffling or blurring without having to wait too long

Exercise

Image compression.

Q&A