

E-Parcel

Overview

The *e-parcel* is an e-mail-like system capable of sending attachments of arbitrarily large size. A semester-long project will ask you to design and implement an e-parcel system. The main purpose of this document is to give a high level overview of e-parcel and of the design and implementation process.

User Interface

Alice wishes to send a 700MB MPEG file to Bob. Here's how e-parcel works:

- Alice launches e-parcel, attaches the file, fills in Bob's regular e-mail address, and clicks *send*. That's it: same as plain old e-mail.
- Bob's e-parcel client beeps and shows a short notice asking Bob whether he wishes to accept delivery of Alice's 700MB video (and a warning Bob not to accept delivery if he does not have enough disk space). Bob accepts delivery. (If Bob declined delivery, Alice would be notified by e-parcel, and no attachment is delivered.)
- After the transmission is complete, Bob receives a second notification with a link to a file on the disk of Bob's own machine. When Bob clicks the link, the video starts playing.

Variations of this basic user interface are possible and can be considered as refinement of the e-parcel system as the project progresses.

Internals

Easy enough? Well, but how would you implement this?

- The two main component of the e-parcel system are the *client* and the *server*. Both Alice and Bob run an e-parcel client. The user interface to an e-parcel client is called the *user agent*. A third host runs the server.
- When Alice clicks send, the e-parcel user agent interfaces to Alice's e-parcel client, which contacts Alice's e-parcel server with a short description of the file to be sent.
- When Bob starts his e-parcel user agent, Bob's client checks for new messages on the server and finds Alice short description. The user agents asks Bob for permission to receive the e-parcel.

- After Bob accepts, Alice and Bob's clients establish a direct connection to the server. Alice's client sends the file to the server, which then continuously relays it to Bob's client.

In summary, the *user agent* is responsible for:

- Interfacing the user with local client.

The *client* is responsible for:

- Sending an e-parcel description to the server.
- Checking for new e-parcels at the server and notifying the user agent.
- Relaying approval or refusal messages to the server.
- Relaying approval refusal messages from the server to the client.
- Establishing a rendezvous at the server side between the sender and the receiver.
- Sending, receiving, storing data, and notifying the user agent of completion.

The *server* is responsible for:

- Receiving, storing, and forwarding e-parcel description tabs.
- Receiving, storing, and forwarding delivery approvals or refusal.
- Aiding in the rendezvous between the two end-hosts.
- Siphoning the connections between sender and receivers.

Additional functionality will emerge during the course of the project as the specifications are developed.

Approach

E-parcel is a fairly sophisticated system and you will iteratively implement it throughout the course of the semester. The design process involves a sequence of phases in which the operations of e-parcel are defined, documented, and implemented. Each phase involves the following steps:

- Students are asked to define a protocol for one of the operations of e-parcel and to explain their design choices.
- The protocol designs are graded individually and returned to students.
- The designs from the various students are merged by the instructor into one document, which forms the specification for the implementation.
- Students are asked to individually implement the merged specifications.