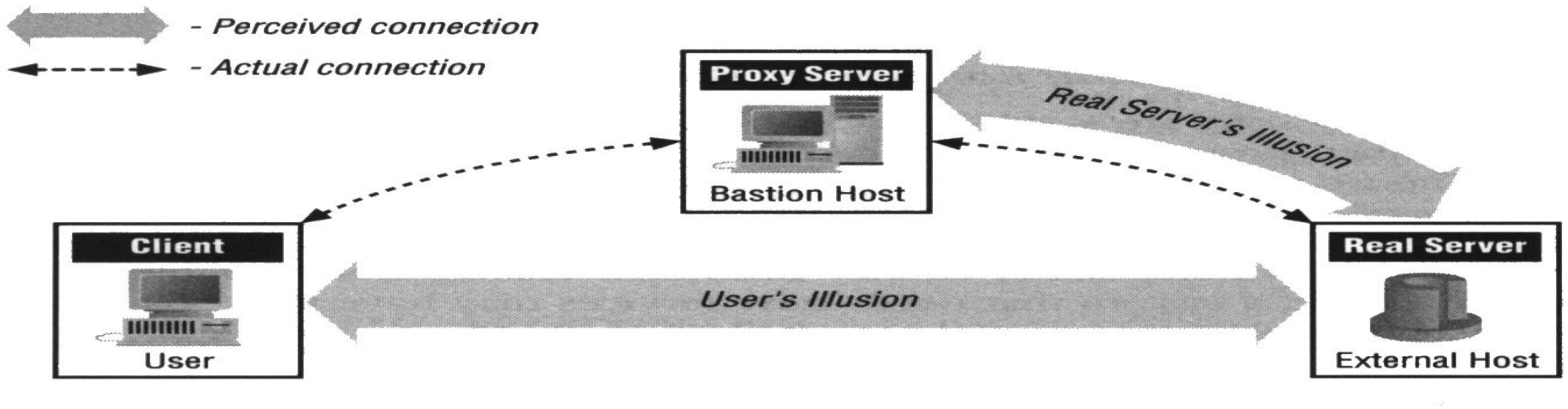
Proxying

- Why?
  - Allow application-level filtering
- Advantages:
  - allow users to access Internet services “directly”
- Disadvantages:
  - Proxy Services lag behind non proxied services (software problems)
  - Proxy services may require different servers for each service
  - Proxy services usually require modifications to clients, procedures, or both
  - Proxy services don’t protect you from all protocol weaknesses


- Perceived connection
- Actual connection
Proxy Server
Bastion Host
Real Server's Illusion
Client
User
User's Illusion
Real Server
External Host

How Proxying Works

- Custom client software
  - the software must know how to contact the proxy server
  - need modified software
  - Problems:
    - the software may be available on some platform only (e.g., SUNOS)
    - it may not be software you wants (not have good interface, etc)
- Custom user procedures
  - use standard client software to talk to the proxy servers
  - tells it to connect to the real server.
  - Main problem:
    - You have to teach your users.

Issues

- Application-level vs. Circuit-level Proxies
  - application-level: knows about the particular application
    - advantage: have control on applications
  - circuit-level: does not know
    - advantage: provide wide variety of different protocols
- Generic vs. Dedicated Proxies
  - Generic: for all protocols
  - for single protocol only.
- Intelligent Proxy Servers
  - E.g., HTTP server caches data.
- No proxying:
  - E.g., SMTP, NNTP, etc.

SOCKS

- Free
- de facto standard proxying package on the Internet.
- Generic
  - So, no intelligent logging or access control
- Only works with TCP
  - For UDP, use UDP Packet Relayer
- Very popular
- Components:
  - SOCKS Server
  - SOCKS client library for UNIX machines
    - e.g., Rconnet() for connect(), change Makefile
  - SOCKS-ified versions of serveral standard programs like FTP and Telnet.