Web Sessions

It's all an illusion (at the HTTP layer)

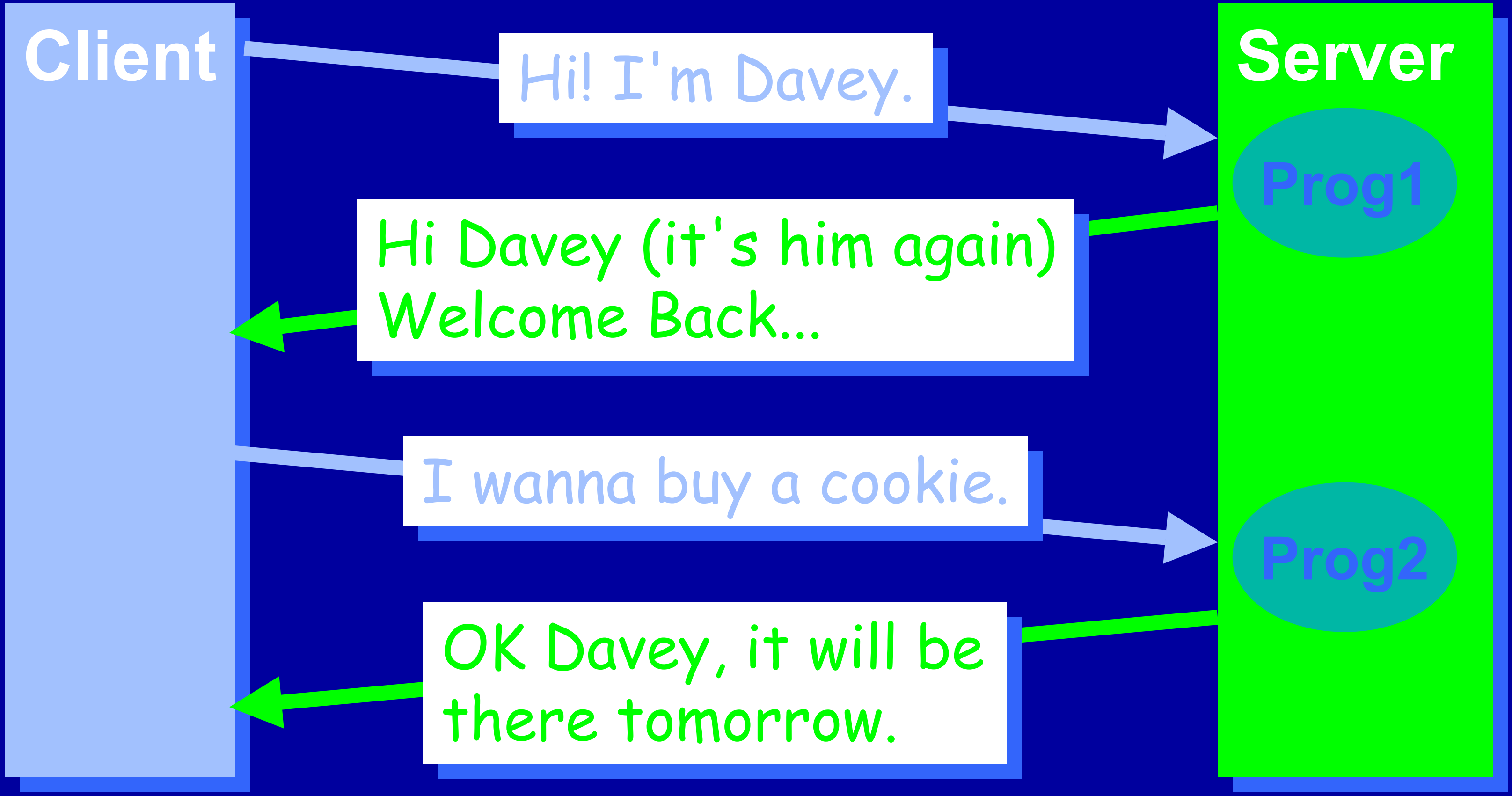
Sessions

- Many web sites allow you to establish a session.
  - you identify yourself to the *system.*
  - now you can visit lots of pages, add stuff to shopping cart, establish preferences, etc.

State Information

- Remember that each HTTP request is unrelated to any other (as far as the Web server is concerned).
- Each new request to a server starts up a brand new copy of the server program (a new thread, or a new process).
- Providing *sessions* requires keeping state information.

Session Conversation



Client

Hi! I'm Davey.

Hi Davey (it's him again) Welcome Back...

I wanna buy a cookie.

OK Davey, it will be there tomorrow.

Server

Hidden Field Usage

- One way to propagate state information is to use hidden fields.
- User identifies themselves to a server program (fills out a form).
- Server sends back a form that contains hidden fields that identify the user or session.

Revised Conversation

Initial form has field for user name.

```
GET /prog1?name=davey HTTP/1.0
```

Prog1 creates order form with hidden field.

```
GET /prog2?name=davey&order=cookie HTTP/1.0
```

Session Keys

- Many Web based systems use hidden fields that identify a *session*.
- When the first request arrives, the system generates a unique *session key* and stores it in a database.
- The session key can be included in all forms/links generated by the system (as a hidden field or embedded in a link).

Session Key Properties

- Must be unique.
- Should *expire* after a while.
- Should be difficult to predict.
  - typically use a pseudo-random number generator seeded carefully.

Server
Session Keys

- A server using session keys:

```
<INPUT TYPE=HIDDEN
  NAME=sessionkey
  VALUE=HungryStudent971890237>
```

Pizza Order

A request to order a pizza might now look like this (all on one line):

```
GET /pizza?sessionkey=
  HungryStudent971890237&pizza=
  cheese&size=large  HTTP/1.0
```

HTTP Cookies

- A “cookie” is a *name,value* pair that a server program can ask the client to remember.
- The client sends this name,value pair along with every request to the server.
- We can also use "cookies" to propagate state information.

Cookies are HTTP

- Cookies are HTTP headers.
- A server can *give* the browser a cookie by sending a **`Set-Cookie`** header line with the response.
- A client can send back a cookie by sending a **`Cookie`** header line with the request.

Setting a cookie

```
HTTP/1.0 200 OK
Content-Type: text/html
Set-Cookie: customerid=0192825
Content-Length: 12345
Favorite-Company: IBM
Nap-Time: 12-2
...
```

`Set-Cookie` Header Options

The general form of the Set-Cookie header is:

`Set-Cookie: name=value; options`

The options include:

```
expires=...
domain=...
path=...
```

`expires` Option

```
expires=Friday 29-Feb-2000 00:00:00 GMT
```

- This tells the browser how long to hang on to the cookie.
- The time/date format is very specific!

`expires`
Time Format

```
Weekday, Day-Month-Year
  Hour:Minute:Second GMT
```

- This all must be on one line!
- Weekday is spelled out.
- Month is 3 letter abbreviation
- Year is 4 digits

Default expiration

- If there is no expires option on the `Set-Cookie` header line, the browser does not save the cookie to disk.
- In this case, when the browser is closed it will forget about the cookie.

`domain` Option

```
domain=.rpi.edu
```

- The domain option tells the browser the *domain(s)* to which it should send the cookie.
- *Domains* as in DNS.
- The domain must start with "." and contain at least one additional "."

`domain` option rules

- The server that sends the Set-Cookie header must be in the domain specified.

- If no domain option is in the header, the cookie will only be sent to the same server.

Default Behavior

`path` Option

```
path=/
```

or

```
path=/~hollingd/netprog
```

- The path option tells the browser what URLs the cookie should be sent to.

`path` default

- If no path is specified in the header, the cookie is sent to only those URLs that have the same *path* as the URL that set the cookie.
- A *path* is the leading part of the URL (does not include the filename).

Default Path Example

If the cookie is sent from:

```
    /~hollingd/netprog/pizza/pizza
```

it would also be sent to

```
     /~hollingd/netprog/pizza/blah
```

but not to

```
/~hollingd/netprog/soda/coke
```

Set-Cookie Fields

- Many options can be specified.
- Things are separated by ";":

```
Set-Cookie: a=blah; path=/;
  domain=.cs.rpi.edu;
  expires=Thursday, 21-Feb-2002
  12:41:07 2002
```

All must be on one line!

Cookie creation

- A server program can send back any number of HTTP headers.
  - can set multiple cookies
- Content-Type is required!
- Blank line ends the headers!

Example

```
Content-Type: text/html
Set-Cookie: prefs=nofrms
Set-Cookie: Java=yes
```

… *now sends document content*

Getting Cookies

Drop by Dave's office anytime!

If you want cookies, you might consider bringing some with you...

Getting HTTP Cookies

- The browser sends each cookie as a header:

```
Cookie: prefs=nofrms
Cookie: Java=OK
```

- The Web server reads the cookies from the headers. CGI and servlets pass the cookies via environment variables or via the `HttpServletRequest getCookies()` method (which returns an array of `Cookie`s).

Multiple Cookies

- There can be more than one cookie.
- Using CGI, the Web Server puts them all together like this:

```
prefs=nofrms; Java=OK
```

maybe a space, maybe not!

and puts this string in the environment variable: **HTTP_COOKIE**

Using Servlets, you can use:

```
…cookies[i].getName();…
…cookies[i].getValue();…
…cookies[i].getVersion();…
```

Cookie Limits

- Each cookie can be up to 4k bytes.
- One "site" can store up to 20 cookies on a user's machine.

Cookie Usage

- Create a *session*.
- Track user browsing behavior.
- Keep track of user preferences.
- Avoid logins.

Cookies and Privacy

- Cookies can't be used to:
  - send personal information to a web server without the user knowing about it.
  - be used to send viruses to a browser.
  - find out what other web sites a user has visited.
  - access a user's hard disk

Some Issues

- Persistent cookies take up space on user's hard disk.
- Can be used to track your behavior within a web site.
  - This information can be sold or shared.
- Cookies can be shared by cooperating sites (advertising agencies do this).