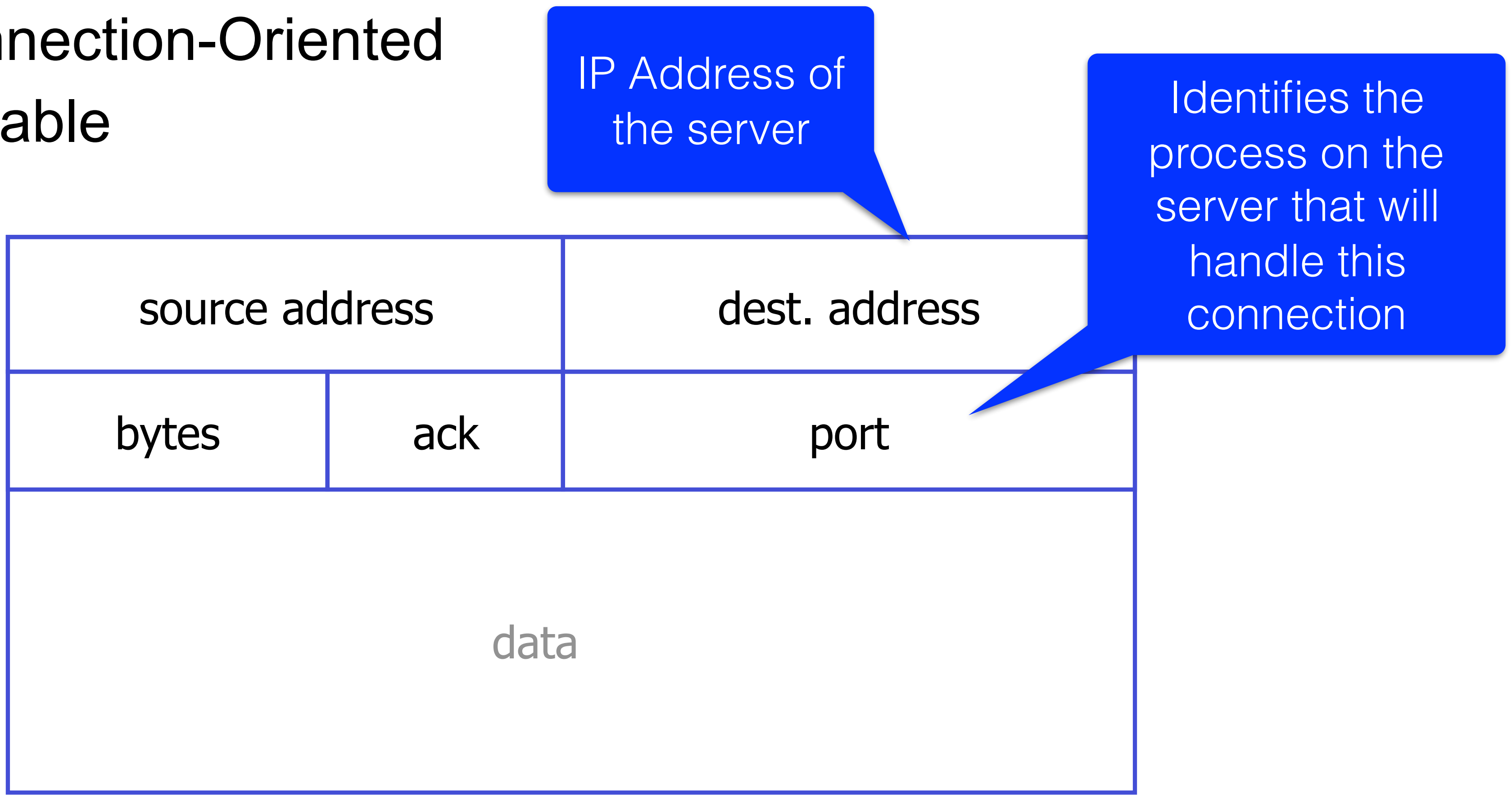
HTTP

Protocols

Application Layer FTP, HTTP, SSH, IMAP
Transport Layer TCP, UDP
Internet Layer IP
Link Layer Ethernet, WiFi

TCP/IP

- Transmission Control Protocol.
  - Connection-Oriented
  - Reliable


IP Address of the server
Identifies the process on the server that will handle this connection
source address
dest. address
bytes
ack
port
data

Application Layer Protocols

Application: communicating distributed processes

- Running on network hosts in user space
- Exchange messages to implement app
- E.g., email, file transfer, bit torrent, web

Application-layer protocols:

- One piece of an app
- Defines messages exchanged by apps
- Uses services provided by lower layer protocols

Application Layer Protocols

API: application programming interface

- Defines interface between application and transport layer

socket: Internet API

- send, receive

HTTP

- Sits on top of TCP - data payload
- Goal: transfer objects between client (browser) and server (web application)
- Separate from other Web concepts:
    - HTML: page layout
    - URLs : object naming

http in operation

Suppose user enters: http://www.tkf.toronto.on.ca

http client initiates TCP connection to http server at www.tkf.toronto.on.ca on port 80

http server at host www.tkf.toronto.on.ca accepts the connection notifying client

http client sends http request message into TCP socket

http server receives request message, forms response message and sends it into socket

HTTP is stateless

- Server does not maintain status information across client requests
- No way to correlate multiple request from some user
- Protocols that maintain “state” are complex
- past history must be maintained
- if server or client crashes, their views of “state” may be inconsistent and must be reconciled.

HTTP Request and Response

GET

```
GET /~cs209hf/cgi-bin/remark-submit.cgi?
course=csc209h&first_name=Karen&last_name=R
eid&cdf_account=reid&student_number=1112223
33&email_address=reid%40cdf.toronto.edu&ass
ignment=a1a&request=The+TA+ought+to+be+shot
+for+doing+such+a+terrible+job. HTTP/1.1
User-Agent: curl/7.18.2 (i486-pc-linux-gnu)
libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3
libidn/1.8 libssh2/0.18
Host: wwwcgi.cdf.toronto.edu
Accept: */*
```

POST

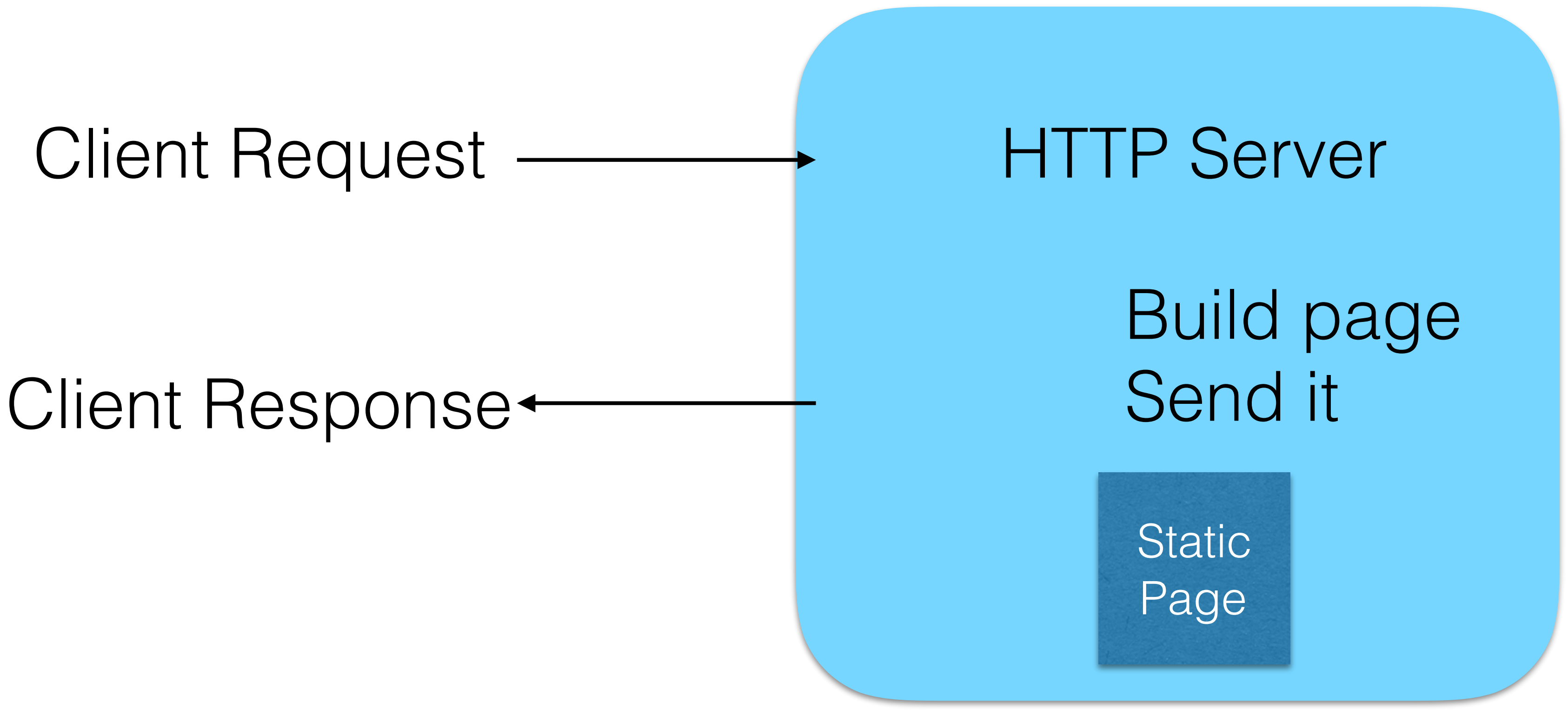
```
POST /~cs209hw/cgi-bin/processa1a.cgi HTTP/1.1
User-Agent: curl/7.18.2 (i486-pc-linux-gnu)
libcurl/7.18.2 OpenSSL/0.9.8g zlib/1.2.3.3
libidn/1.8 libssh2/0.18
Host: wwwcgi.cdf.toronto.edu
Accept: */*
Content-Length: 293
Content-Type: multipart/form-data;
boundary=----------------------------46916b928ffe

cdf_account=reid
data=1,1,412,Success
```

Response

```
HTTP/1.1 200 OK
Server: Apache/2.4.7 (Ubuntu) SVN/1.8.8
PHP/5.5.9-1ubuntu4.22 OpenSSL/1.0.1f
Vary: Host
Accept-Ranges: bytes
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html
```

Simplest Server



REST

- HTTP messages are really just an interface to a web server
- REST (Representation State Transfer)
  - interface definition
  - constrains how HTTP messages are used for good design

Interface Design

- Define a good Domain model
  - What are the objects in your application? How do they interact?
- Abstract core functions
  - What are your verbs?
- From DB: CRUD
- From OO: interfaces

Good Design

- URL contains nouns
- URL specifies resource (or nested resources)
- Verbs are “GET”, “POST”, “PUT”, “DELETE”

HTTP Methods

GET	collection	List resources along with attributes in a collection
POST	collection	Create a new entry in the collection
GET	resources	Retrieve a single resource
PUT	resource	Replace a resource, or update parts of a resource
DELETE	resource	Delete a resource

What do these do?

- GET /api/assignments
- POST /api/assignments
- GET /api/assignments/id
- PUT /api/assignments/id
- GET /api/assignments/id/groups
- GET /api/assignments/id/groups/id

- GET /api/users.xml?filter=type:Ta

Response

- HTTP Status Code
- Data

HTTP Status code examples

Success codes:

- 200 - OK (the default)
- 201 - Created
- 202 - Accepted (often used for delete requests)

User error codes:

- 400 - Bad Request (generic user error/bad data)
- 401 - Unauthorized (this area requires you to log in)
- 404 - Not Found (bad URL)
- 405 - Method Not Allowed (wrong HTTP method)
- 409 - Conflict (i.e. trying to create the same resource with a PUT request)

Data

- The program sending the request needs to know how to interpret the response.
    - Standard, structured text
- XML
- JSON

JSON

Object

- `key : value`
- List
- Collection

Ordered list

- `[object, object, object]`

Collection

- `{object, object, object }`

```
[
  {
    id: 123,
    name: ‘Assignment 1'
  },
  {
    id: 456,
    name: 'Aliya'
  }
]
```