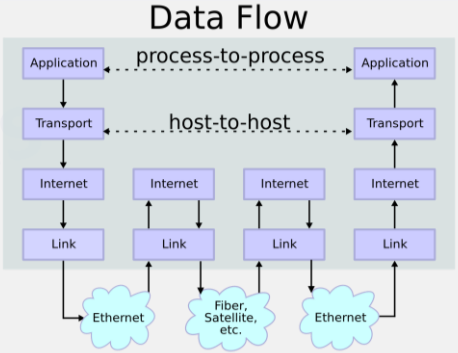
CLOUD COMPUTING APPLICATIONS

Cloud Computing Glue: Internet Protocol, HTTP and RPC on HTTP

Prof. Reza Farivar

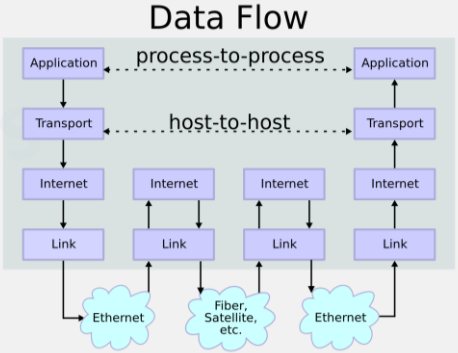
Internet Protocol Stack: Link and Internet Layers

- The second level, “internet level”, is where internet switches and routers operate
  - *Gets your message from one machine to another*
  - Responsible for addressing **host interfaces**
  - Encapsulating data into datagrams (including fragmentation and reassembly)
  - Routing datagrams across one or more IP networks
  - *IP address, IPV4, IPV6*
- EE and Network engineers live at the bottom level, the Link level
  - Out of scope our discussion



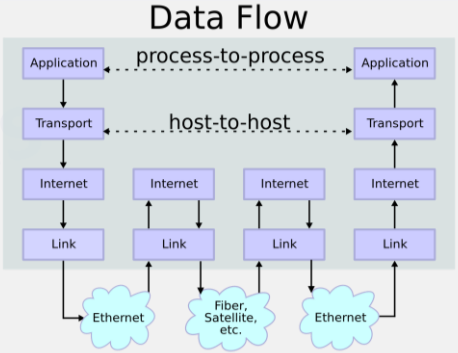
Internet Protocol Stack: Transport Layer

- The transport level handles packetizing of data
- Gets your message from a process in one machine to another process in another machine
- Typically implemented in the Operating System
- Provide port numbers (similar to Unix sockets)
- TCP
  - Keeps track of data segments, retransmission, acknowledgement
    - Handle network congestion
    - Traffic load balancing
    - Unpredictable network behavior
    - Lost, duplicated, or delivered out of order IP packets
  - Guarantees that all bytes received will be identical and in the same order as those sent
- UDP
  - “User Datagram Protocol”, aka. “Unreliable Datagram Protocol”
  - Very simple



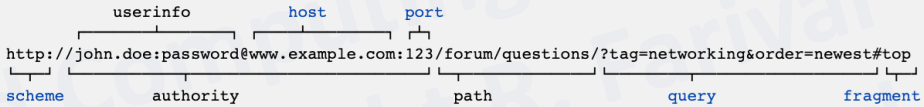
Internet Protocol Stack: Application Layer

- Application level handles “what to send”
  - HTTP, HTTPS
    - RESTful APIs
  - FTP
  - WebSocket
  - SMTP
  - IMAP
  - SSH
  - DHCP
  - DNS
  - Bit Torrent



HTTP Protocol

- Originally targeted static web pages: *Client Requests, Server Responds, Connection is closed*
- Works on top of TCP for reliable transport
- Client (user agent) can be a web browser, or any other software
- HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), using the Uniform Resource Identifiers (URI's) schemes http and https



- In HTTP/1.0 a separate connection to the same server is made for every resource request
  - Establishment of TCP connections has overhead
- HTTP/1.1 can reuse a connection multiple times to download images, scripts, stylesheets, etc. after the page has been delivered
  - Persistent Sessions
  - Lower latency

HTTP Message Format

- Request “verbs”
  - GET: retrieve data
  - POST: server should accept the call parameter as a new value for the resource specified in the URL
  - PUT: server should store the enclose entity under the supplied URL
  - DELETE: server should delete the specified resource by the URL
  - PATCH: server should apply partial modification to the resource
  - …
- Request message
  - a request line (e.g., GET /dataset/inventory.htm HTTP/1.1, which requests /dataset/inventory.htm resource from the server)
  - request header fields
  - an empty line
  - an optional message body
- Response message
  - a status line which includes the status code and reason message (e.g., HTTP/1.1 200 OK, which indicates that the client's request succeeded)
    - *Informational 1XX | Successful 2XX | Redirection 3XX | Client Error 4XX | Server Error 5XX*
  - response header fields (e.g., Content-Type: text/html)
  - an empty line
  - an optional message body

Client Request

```
GET / HTTP/1.1
Host: www.example.com
```

Session

Server Response

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>Hello World, this is a very simple HTML document.</p>
  </body>
</html>
```

RPC on HTTP

- Remote Procedure Calls built on HTTP
  - For many types of RPC, the client/server conversation model of HTTP works just fine
  - Just replace the HTML markup with an XML or JSON representation of the data
  - XML-RPC
  - JSON-RPC
    - E.g. Bitcoin server
    - Commands encoded as JSON, sent over HTTP