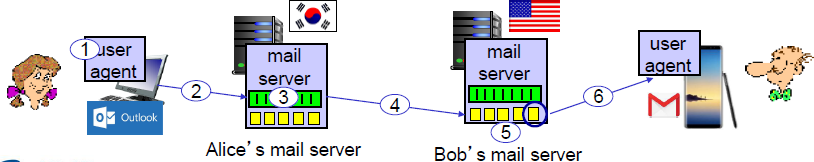
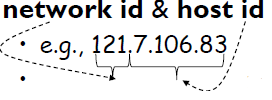
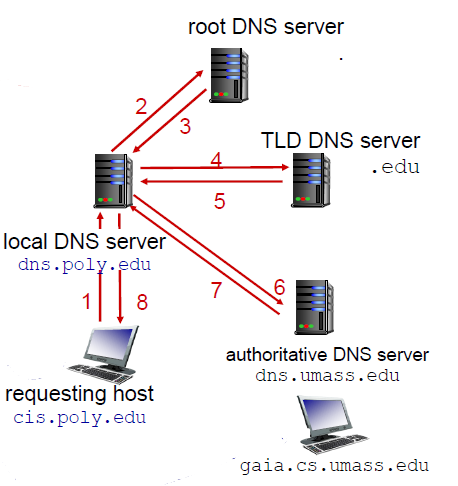
**Computer Network  
Chapter 2: Application Layer– part 2**

**Electronic mail (E-mail):** email의 특징으로는 asynchronous(비동기) communication을 한다 – 이메일은 보낼 때 편하게 보내면 됨. (원래 다른 것은 client가 server에 요청을 주면 바로 응답을 해야한다.)  
 **Three major components:**  
 - User agents: gmail, outlook 같은 이메일을 보고, 작성하고 보내기 등등을 할 수 있는 것.  
 - mail servers: 각 메일은 메일 서버가 존재. (gmail서버, naver 메일 서버)  
 -- mailbox: 우편함 같은 것, 메일은 유저의 메일박스로 들어옴.  
 -- message queue: 메시지가 밖으로 나가기 위한 queue  
 -- SMTP를 이용해서 다른 메일 서버와 연결해서 message queue의 메시지를 보낸다. (같은 메일 서버에 보낼 때는 SMTP를 이용하지 않고 바로 그 유저의 메일박스로 보낸다)  
 - **SMTP** (simple mail transfer protocol): 메일 서버 간의 통신을 하는 프로토콜, **direct transfer**: sending server(SMTP client) to receiving server(SMTP server) – TCP 사용  
  
User A가 outlook (User agent)를 이용한다. 🡪 message queue로 간다. 🡪 SMTP로 다른 메일 서버와 TCP연결한다. 🡪 그 메일은 받은 사람의 메일서버의 mailbox로 들어간다. 🡪 User B가 user agent를 이용해서 읽는다.  
 **Mail access protocols (모두 SMTP이용)**  
 POP: 메일을 받으면 그 유저의 mailbox의 메일을 삭제하고 읽음  
 IMAP: 그 유저의 메일박스 그대로 읽음 –용량 문제 발생  
 Web-based Email: gmail, Hotmail …  
SMTP는 client가 server한테 object를 보내기 때문에 push protocol이라고 한다.  
HTTP는 client가 server로부터 object를 받기 때문에 pull protocol이라고 한다.  
SMTP 메시지는 7bit ASCII 사용 (이미지 보내면 🡪 7bit ASCII변환 🡪 이미지 받기)

**DNS**  
 **IP address**: 4bytes (32bits), hierarchical structure / 1byte.1byte.1byte.1byte 각각 10진수로 표기 (0~255), IP address는 Unique한다.  
IP address는 구분 뿐만 아니라 **network ID** (같은 네트워크를 사용하면 ID가 같음 – 라우터들이 경로를 찾을 때 네트워크 ID로 확인)와 **host ID**를 표기한다.  
  
 **DNS: domain name system**사람들이 보기 쉬운 Host name과 router들이 경로를 찾기 쉬운 IP address를 맵핑  
host name 🡪 IP address를 DNS  
**application-layer protocol에서 사용한다** (end host가 host name을 입력하면 DNS가 IP address로 변환해준다.)  
 **DNS: a distributed, hierarchical database** 하나의 서버로 넓은 지역과 많은 유저와 정보를 관리하기에는 힘들기 때문에 서버를 분산하여 계층적 구조로 관리한다.  
**Root DNS server**에 먼저 물어봄 (자신의 하위단계 TLD server가 무엇인지 찾아봐서 TLD서버 주소를 알려줌) 🡪 **TLD 서버**로 가서 원하는 사이트가 있는 authoritative server를 물어보고 알아온다 🡪 **Authoritative DNS server**에서는 처음에 Root DNS server에 물어봤던 host name의 IP address를 알아온다.  
 **Root DNS server** (전 세계 유저들이 IP address를 찾아가기 위해 꼭 걸치는 중요한 곳) 전세계 13개  
 **TLD server** (Top-Level-Domain) – com, org, net 혹은 country domain(kr, jp) 등등을 담당한다.  
 **Authoritative DNS server**: organization’s own DNS server – authoritative DNS server는 1단계로 구성 되어있을 수 있고, 그 외 여러 단계로 구성될 수 있다. (amazon.com – 1단계, gachon.ac.kr - 2단계(gachon, ac))  
 **Local DNS name server** 각 ISP 내에 존재  
 **Iterated query**: client에서 hostname을 입력해서 검색하면**, Local DNS server대신해서** Root DNS server으로부터 TLD주소를 알아오고, Local DNS server가 다시 TLD server로 가서 authoritative DNS server를 알아온 후 Local이 authoritative DNS server로 가서 IP address를 알아온다. **그 정보를 마지막으로 client한테 알려준다**.  
 recursive query는 부화가 커서 잘 안쓴다.  
 **Local DNS server를 이용하는 이유는** 많은 유저가 특정사이트를 요청하는데, 그 사이트 주소 정보를 미리 local DNS server에 캐싱해둔다. Client는 무조건 local DNS server를 활용하기 때문에 그 **캐싱 된 정보를 바로바로 가져올 수** 있다 – **빠르다(response time이 크게 줄어듦**) 그리고 매번 root DNS – TLD – authoritative DNS server를 갈 필요가 없기 때문에 이 세개의 DNS server의 부화를 줄여준다. **또한 TLD server와 Authoritative DNS server를 알기 때문에 그 연관 된 주소를 빠르게 찾을 수 있다**. 🡺 원래는 총 8번의 이동이 필요한데 local DNS server를 통해서 6개, 4개 2개까지 줄어들 수 있다.  
  
  
 **DNS: caching, updating records**  
DNS 캐시는 시간이 지나면 자동으로 삭제되게 만든다 (TTL) – 캐시로 인해서 out of date: 잘못 된 데이터를 줄 수 있다.. 만약에 TTL이 되기전에 IP address가 바뀌면 캐시는 계속 남아있기 때문에 잘못된 주소를 보내 줄 수 있다. 🡺 캐시의 단점  
 **DNS: services**   
 **Translate hostname to IP address  
 Load distribution:** 분산된 서버를 로드 할 수 있다. – naver.com은 여러 서버를 가지고 있는데, DNS server가 client에게 네이버의 분산 된 여러 서버의 주소를 요청에 대한 응답으로 골고루 보내준다.

**P2P applications**  
 **Pure P2P architecture**: no always-on server, end systems끼리 직접 communicate File distribution (Bit Torrent), VoIP (Skype), Streaming (KanKan)  
 **File distribution: client-server vs P2P  
 client-server**: server의 file을 n명의 client가 다운받을 때, cllient의 다운로드 속도와 server의 업로드 속도 두개를 고려해야한다. – file size: F, upload 시간: Us 🡪 file하나를 보내는 데, F/Us… 1Mbit/1Mbps = 0.01s 🡺 총 시간은 server의 업로드 시간과 client의 다운로드 시간을 비교해서 오래 걸리는 시간을 따른다. 🡺 NF/us  
서버의 파일을 N명이 다운받을 때 서버는 업로드를 n개 해야하므로 server의 시간이 오래걸리게 됨. 그래서 client-server system에서는 client가 많아질 수 록 속도가 느려진다.  
 **P2P**: P2P는 client가 다운받을 때 업로드도 같이 하므로 결국 업로더도 같이 증가하여 점점 빨라진다. – self scalability: 마치 서버의 개수가 늘어나 업로더 수가 늘어나는 것과 같은 현상 🡺 NF/(us + ∑ui)  
 **P2P file distribution: Bit Torrent**torrent: 여러 chunk(조각)으로 이뤄져 있는 한 파일을 chunk를 다른 유저랑 주고 받는 그룹, track server가 존재 – 이것으로 한 그룹에 어떤 유저가 있고, 각 유가 어떤 조각을 가지고 있는지 report 해준다.  
 - file chunk를 주고 받을 때는 **골고루 받는 것이** 좋다. 그래서 골고루 업로드, 다운로드 속도가 높아진다.  
 - **rarest first**: 레어한 것부터 공유해야 그 레어한 조각이 흔해진다.  
 - **tit-for-tat**: 업로드할 때 4명한데 동시에 업로드한다. 그 중 자신한테 업로드를 잘 해주는 얘한테만 동시에 업로드를 잘 해준다. 주지 않는 나머지는 요청이 들어와도 보내지 않는다. 🡪 10초에 4명을 다시 고른다., 30초마다 랜덤으로 한 명 재 선택 🡪 더 좋은 애를 찾기 위해, optimistically unchoke: 30초마다 랜덤으로 한 명이 선택 되므로 처음에 들어온 애도 다운 가능