Written by Dr. Thapanapong Rukkanchanunt

05 Client-Server Model

OUTLINE

- Characteristics of Client-Server Model
- Fat Client vs Fat Server
- Two-tier vs Three-tier
- Client-Server Computing
  - Middleware
- Application Programming Interface
- Peer-to-Peer Model

What is Client-Server Model

- Client-Server Model เป็นสภาปัตยกรรมที่แยกส่วน Client จากส่วน Server
- โดยส่วนมากแล้วจะพัฒนาบนระบบเครือข่ายคอมพิวเตอร์
- แบ่งโครงสร้างออกเป็นสองโหนดคือ Client และ Server
  - Client เปรียบเสมือนผู้ใช้บริการโดยส่ง Request ไปยัง Server
  - Server เปรียบเสมือนผู้ให้บริการที่ส่ง Response ตอบกลับ Request ของ Client
  - มักจะถูกเรียกว่า Two-tier
  - สามารถมองเป็นการแชร์ทรัพยากรระหว่าง Client โดยนำเอาทรัพยากรหลักไปฝากไว้ที่ Server

Common Characteristics

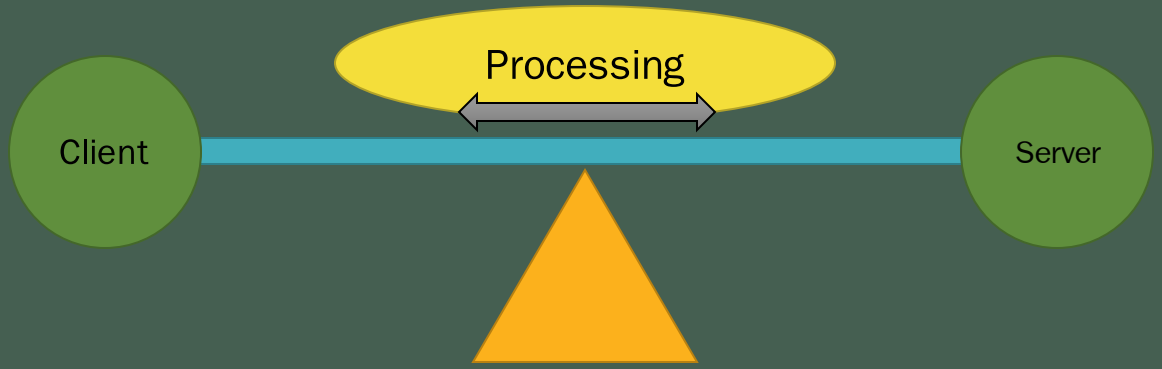
- Client จะต้องเป็นผู้เริ่มติดต่อเวลาใช้บริการ (Initiation)
- ตำแหน่งที่อยู่ของ Server ในระบบเครือข่ายจะต้องเข้าถึงได้จาก Client (Transparency)
- การติดต่อระหว่าง Client และ Server จะเป็นในรูปแบบการส่งข้อความแลกเปลี่ยนกัน (Message-based Transaction)
- Client-Server Model ต้องขยายได้เสมอ (Scalibility)
  - เพิ่มจำนวน Client ได้ (Horizontally scalable)
  - เพิ่มจำนวน Server ได้ (Vertically scalable)

Balancing Tasks

- ใน Client-Server Model เราจะเป็นต้องแบ่งการประมวลผลระหว่าง Client และ Server
- Fat Client คือการนำเอาการประมวลผลส่วนมากมาไว้ที่ Client
  - Client ต้องรู้ว่าข้อมูลถูกจัดเก็บแบบไหน และอยู่ตรงไหนใน Server โดยส่วนมากแล้วจะดาวน์โหลดข้อมูลดังกล่าวมาไว้ที่ตัว Client เอง จึงสามารถทำงานได้แม้จะ Offline
  - Server รองรับการใช้งานจาก Client ได้มาก เพราะมีการประมวลผลน้อย
  - ลดอัตราการเชื่อมต่อในระบบเครือข่าย
  - ถ้ามีการอัพเดตระบบ จะต้องอัพเดต Client ด้วย
  - ตัวอย่างเช่น Online Gaming, Mail Client

Balancing Tasks (2)

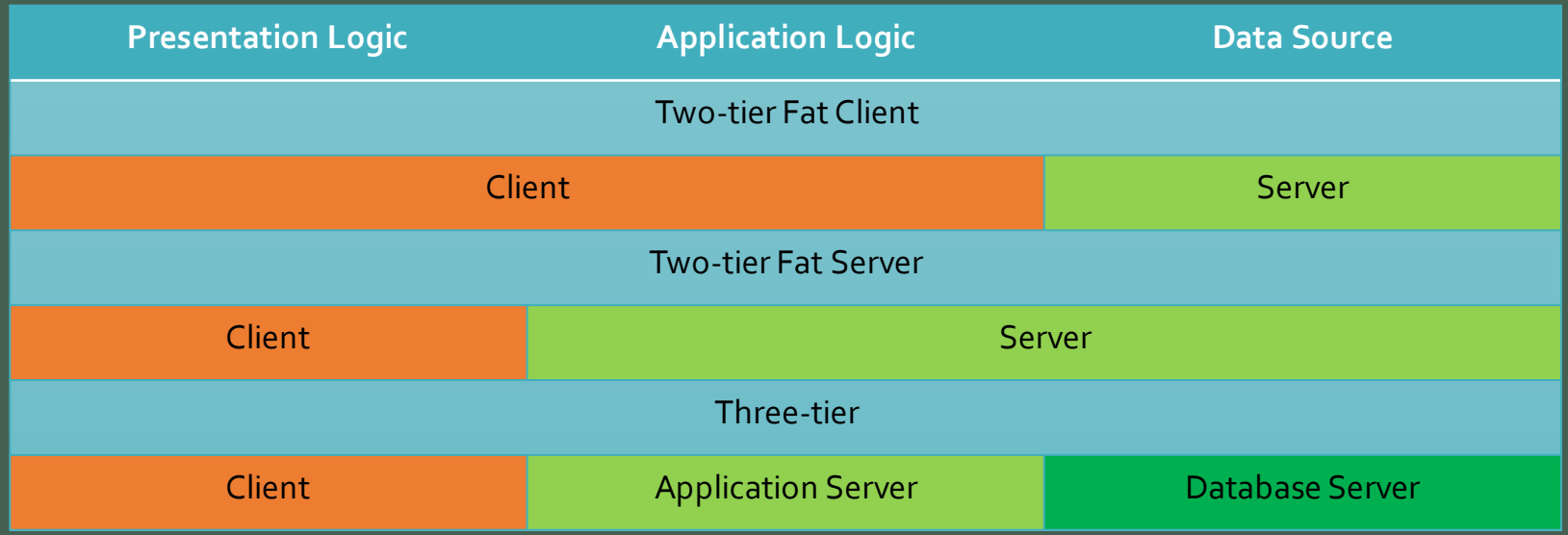
- Fat Server คือการนำเอาการประมวลผลส่วนใหญ่มาไว้ที่ Server
  - โครงสร้างของ Server มีความซับซ้อน จำเป็นต้องมี Server จำนวนมาก
  - Client เชื่อมต่อกับ Server ผ่าน Graphic-User Interface (GUI)
  - ถ้าระบบมีการอัพเดต Client ไม่จำเป็นต้องอัพเดตด้วย
  - ตัวอย่างเช่น Web Application, Cloud Computing



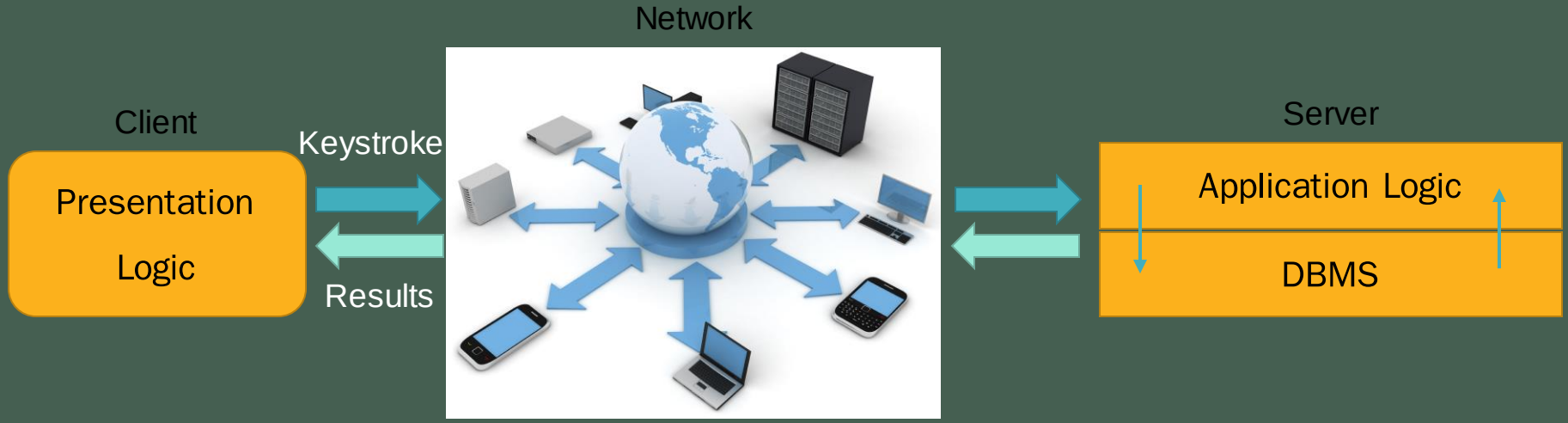
Three Layers of Application

- นอกจาก Fat Client / Fat Server แล้วเรายังสามารถจำแนกสถาปัตยกรรมตามส่วนประกอบของ Application
- ส่วนประกอบของ Application มีด้วยกัน 3 ชั้นได้แก่
  - Presentation Logic ชั้นแสดงผลลัพธ์
  - Aplication Logic ชั้นประมวลผล
  - Data Source ชั้นข้อมูล
- โดยทั่วไปแล้ว Presentation Logic จะอยู่ใน Client ในขณะที่ Data Source จะอยู่ใน Server ดังนั้น Fat Client / Fat Server จะระบุตามที่อยู่ของ Aplication Logic

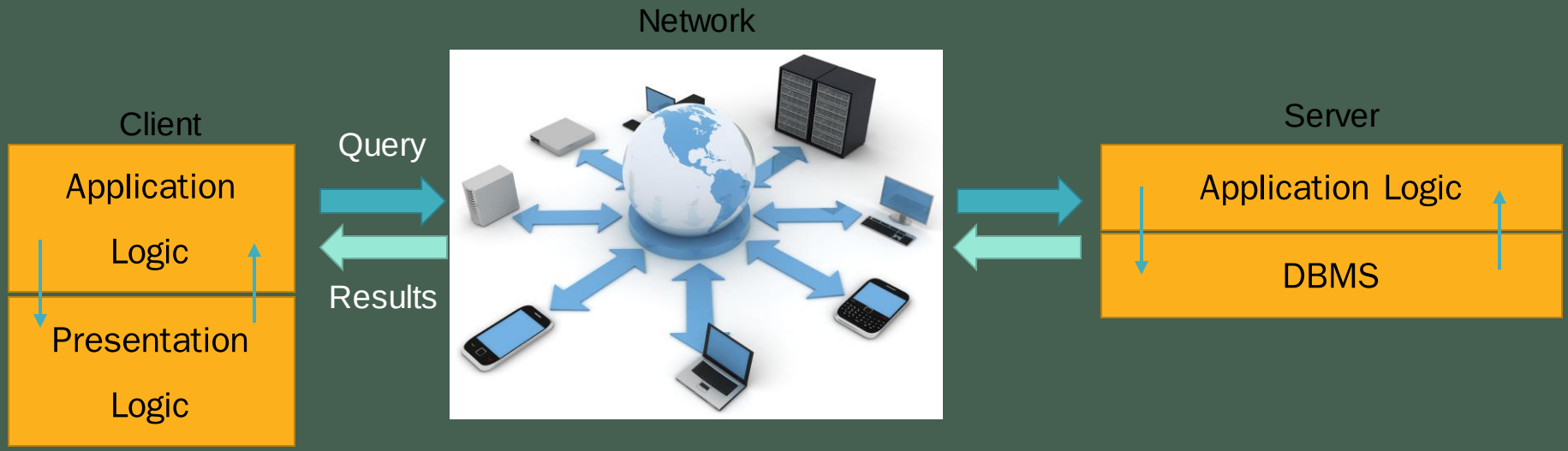
Two-tier vs Three-tier



Example of Client–Server Interaction



Distributed Approach



Client–Server Computing

- Client–Server Computing หรือ Distributed Computing เป็นส่วนต่อเติมจากเขียนโปรแกรมแบบ Standalone ที่ใช้อุปกรณ์หลายเครื่องในการประมวลผลร่วมกัน
- แบ่งการทำงานของโปรแกรมทั้งหมดให้อยู่ในรูปของ Module
- แต่ละ Module อาจจะไม่ได้ทำงานเสร็จในพื้นที่ที่เดียวกัน
  - Calling Module กลายมาเป็น Client (ส่ง Request)
  - Called Module กลายมาเป็น Server (ตอบ Response)
- ส่วนประกอบของ Client–Server Computing ได้แก่
  - Client
  - Server
  - Middleware

Middleware

- Middleware คือซอฟต์แวร์กลางที่เชื่อม Client และ Server เข้าด้วยกัน
  - อาจจะเชื่อม Server กับ Server ด้วยกันเองได้ แต่จะ Client จะไม่เชื่อมต่อกันเอง
- Middleware ช่วยให้ Client และ Server สามารถเรียกใช้งาน Service นอกเหนือจากที่ Operating System มีให้ เช่น Authentication, API Management
- Middleware แบ่งออกเป็น 6 ประเภทได้แก่
  - Asynchronous Remote Procedure Calls (RPC) – Client ส่ง Request แต่ไม่ต้องรอ Response ก่อนจะส่ง Request ถัดไป
  - Synchronous RPC – หลังจาก Client ส่ง Request จะถูกบล็อกจนกว่าจะได้รับ Response

Middleware (2)

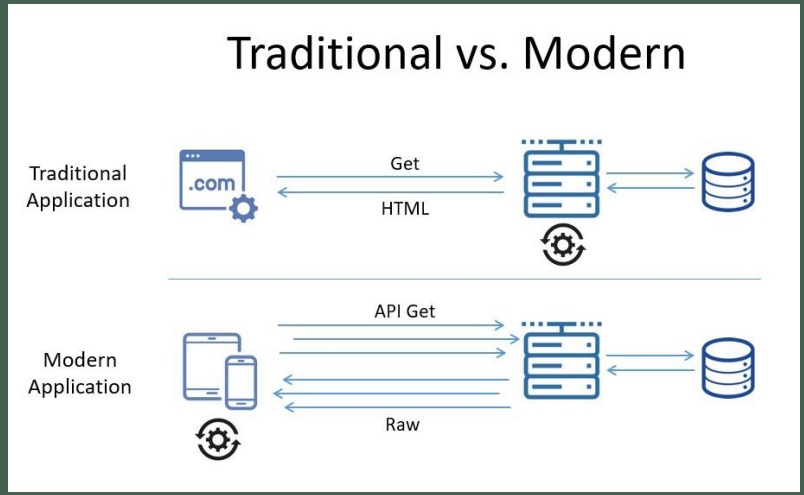
- Middleware แบ่งออกเป็น 6 ประเภทได้แก่
  - Publish/Subscribe (Push) – Server เฝ้าดูการทำงานและส่ง Response เมื่อ Client ว่าง เหมาะกับระบบที่ต้องทำงานเมื่อมีเหตุการณ์เฉพาะเกิดขึ้น
  - Message-Oriented Middleware (MOM) – Client ส่ง Request ไปเก็บไว้ที่ Server จนกว่า Server จะมีเวลามาตอบ Response
  - Object Request Broker (ORB) – การจัดการข้อมูลในรูปแบบ Object โดย ORB จะเก็บที่อยู่ของแต่ละ Object แล้วรับส่ง Request/Response ไปยัง Object ที่ต้องการ
  - SQL-Oriented Data Access – Middleware ระหว่าง Application server และ Database server สามารถแปลงภาษา SQL ให้เหมาะกับฐานข้อมูลปลายทาง

Application Programming Interface (API)

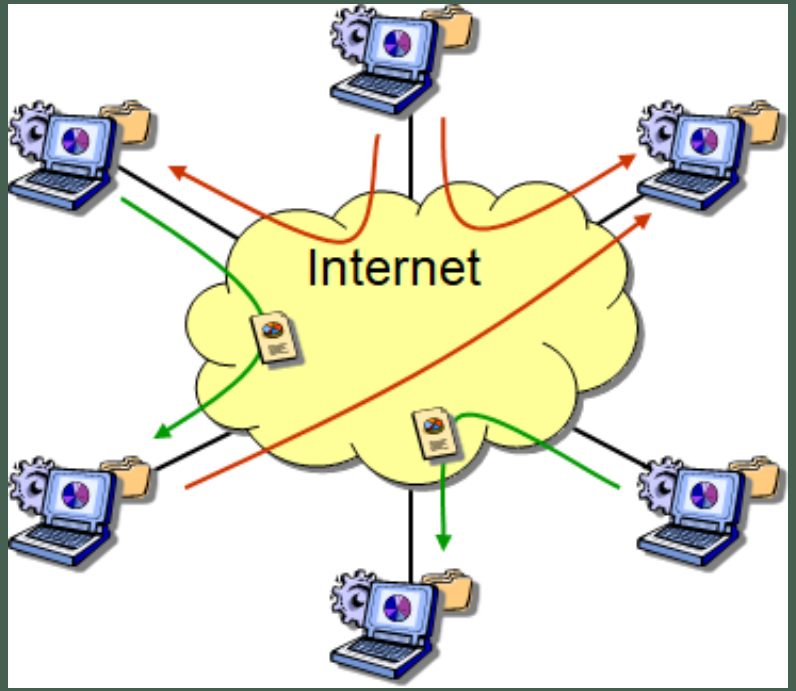
- Application Programming Interface หรือ API คือชุดคำสั่งที่สามารถเรียกใช้งานได้ใน Client เพื่อติดต่อสื่อสารกับ Server (ลักษณะคล้าย Library ที่เรียกใช้งานใน Application เพื่อติดต่อกับ Operating System)
- API ที่ดีต้องใช้งานง่าย ล้มเหลวได้ยาก มีการกำหนดรายละเอียดข้อมูลที่ใช้รับส่งอย่างเป็นระบบ และสามารถรองรับการใช้งานในหลายภาษาได้
- ในปัจจุบันมีหลายบริการที่เปิดให้ใช้งาน API (Public API) เช่น
  - Twitter API – ค้นหา Tweet และ Trend ตามเงื่อนไขที่กำหนด
  - Google Map API – แสดงข้อมูลแผนที่จาก Google Map

API-based App vs Traditional App

- ในอดีตเว็บแอพส่วนมากเข้าผ่าน Web Browser และการประมวลส่วนมากจะอยู่ในส่วน Server
- ในปัจจุบัน Client มีมากกว่าแค่ Web Browser บวกกับประสิทธิภาพที่เพิ่ม ทำให้ผลลัพธ์ที่ส่งออกมาจาก Server จะยังเป็นข้อมูลดิบที่เอาไปประมวลผลต่อใน Client



Peer-to-peer Model



- Peer-to-peer (P2P) Model เป็นสถาปัตยกรรมที่แบ่งการประมวลผลให้กับอุปกรณ์ในเครือข่ายแบบเท่าเทียมกัน ไม่มีการแยกการทำงานระหว่าง Client และ Server หรืออาจจะกล่าวได้ว่าอุปกรณ์ทุกชิ้นทำงานเป็นทั้ง Client และ Server
- ไม่มีส่วนควบคุมกลาง ทำงานในรูปแบบ Dynamic มีการเข้าออกอยู่ตลอดเวลา

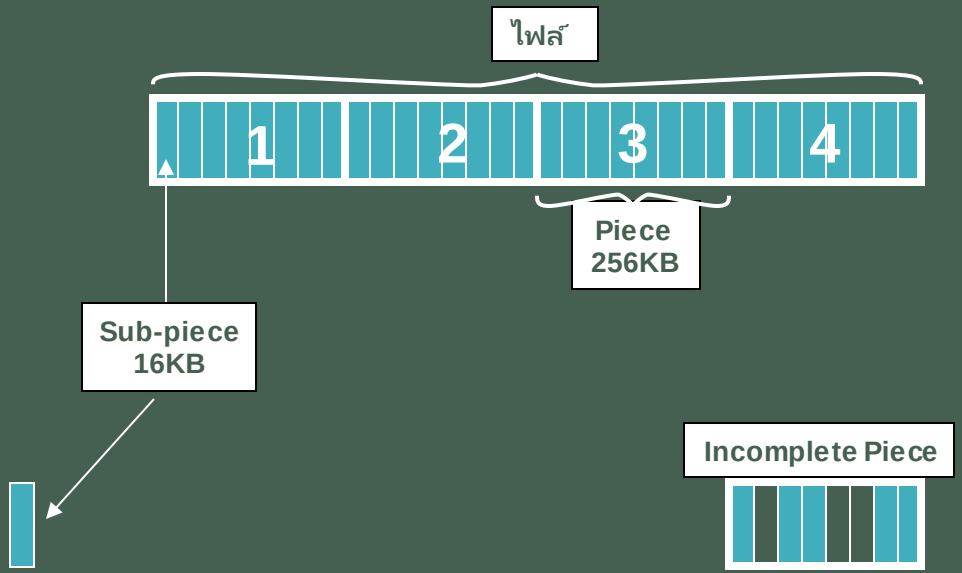
Type of P2P

- P2P สามารถแบ่งออกเป็นสองประเภทใหญ่ ๆ คือ
  - Unstructured ไม่มีการกำหนดโครงสร้างในการเชื่อมต่อและโครงสร้างของการเก็บบันทึกข้อมูล
  - Strctured มี Distributed Hash Table สำหรับการจัดระเบียบโหนดในเครือข่าย
- ในปัจจุบัน Unstructured P2P เป็นที่นิยมมากที่สุด ตัวอย่างที่มักจะใช้อธิบาย P2P ประเภทนี้คือ BitTorrent โปรแกรมแชร์ไฟล์ยอดนิยม

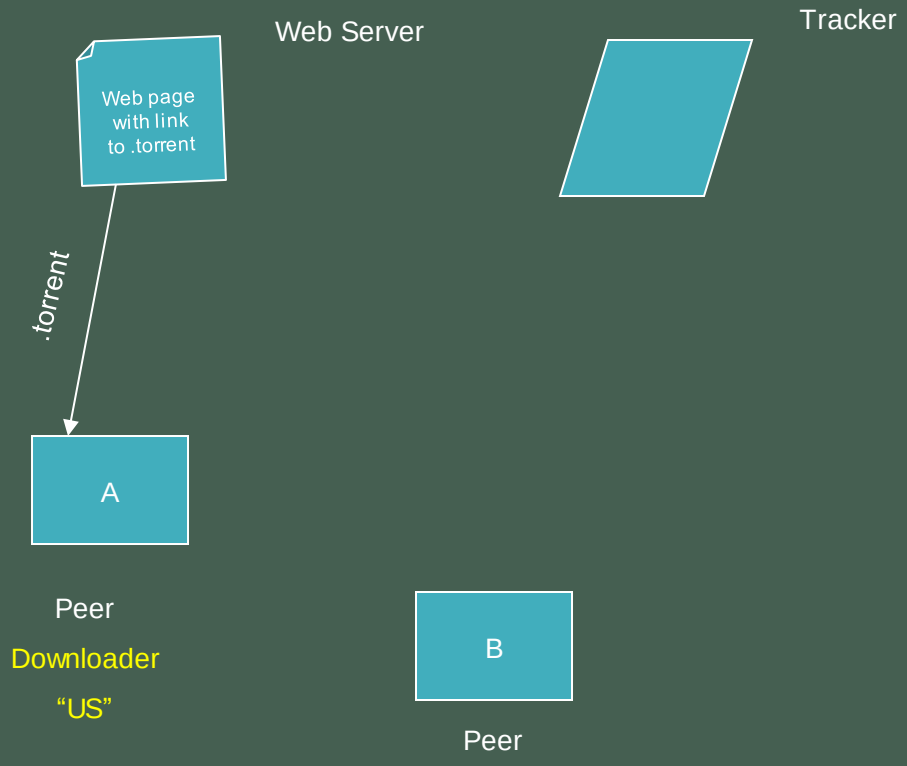
BitTorrent

- BitTorrent เป็นโปรแกรมแชร์ไฟล์บนอินเตอร์เน็ต โดยไฟล์จะถูกเก็บไว้ในโหนดเริ่มต้นก่อนที่จะค่อย ๆ กระจายไปยังโหนดต่าง ๆ ในเครือข่าย
- โหลดในเครือข่ายจะถูกเรียกว่า Peer แต่ถ้ามีไฟล์ครบถ้วนจะถูกเรียกว่า Seed
- ไฟล์จะถูกแตกออกเป็น Piece ย่อย ในแต่ละ Piece ก็จะแตกย่อยลงไปอีกเป็น Sub-piece ขนาด 16KB เป็นส่วนใหญ่
- จนกว่า Piece จะประกอบเสร็จ จะดาวน์โหลดเฉพาะ Sub-piece ของ Piece นั้น
- หากเป็นไปได้แต่ละ Peer จะโหลด Piece ที่ไม่เหมือนกัน จะทำให้การประกอบไฟล์กลับเป็นไปได้อย่างรวดเร็ว

Overall Architecture

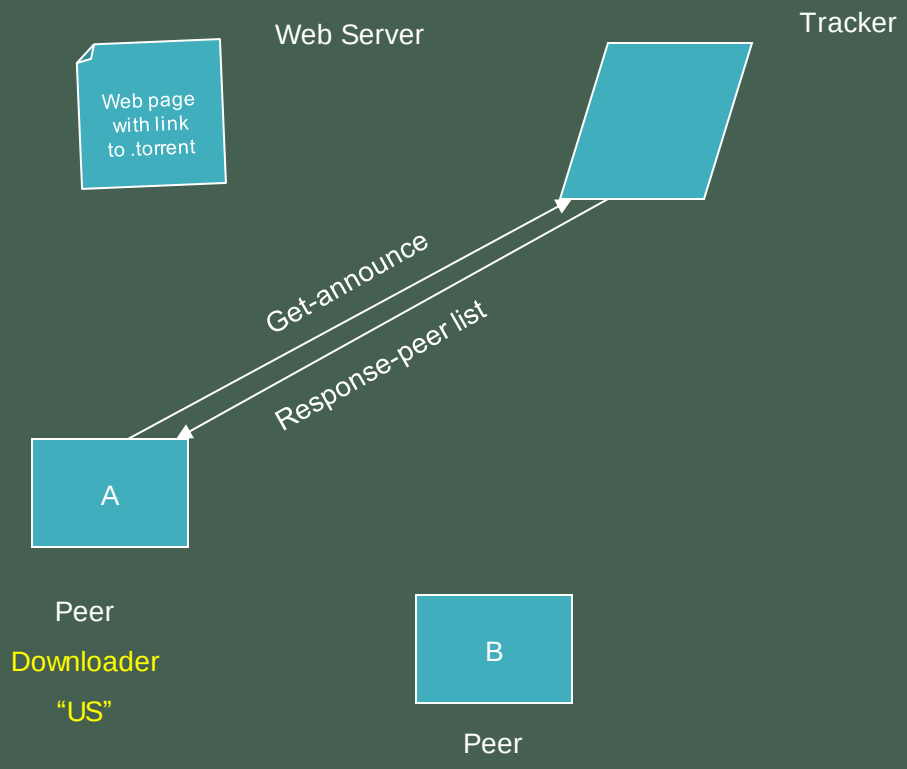


Overall Architecture



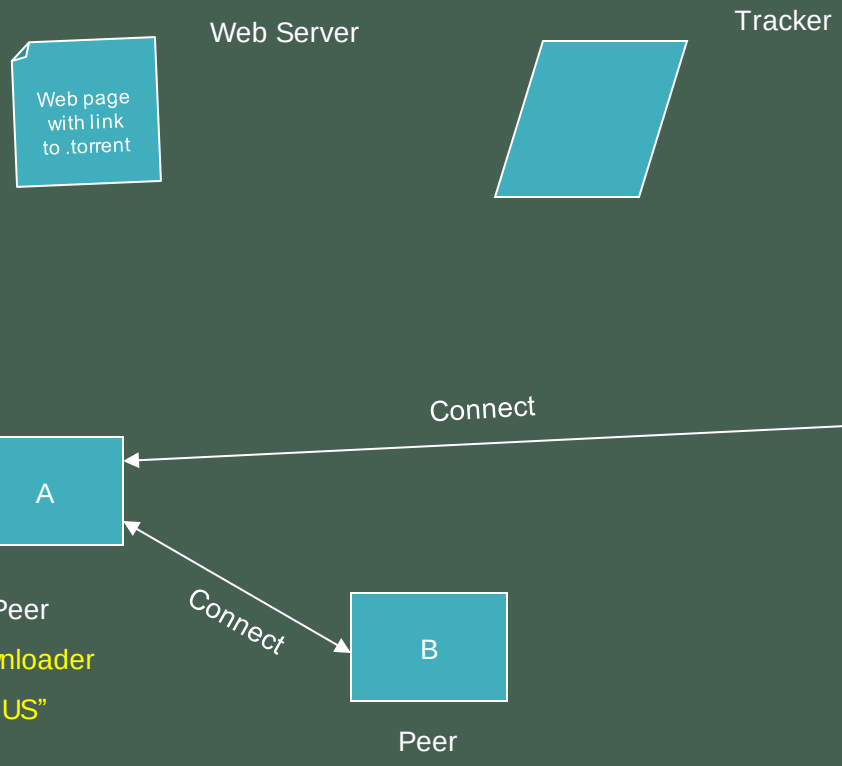
เริ่มต้นด้วยการโหลดไฟล์ .torrent จาก Web Server โดยในไฟล์จะมีที่อยู่ของ Tracker

Overall Architecture



ส่ง Request ไปยัง Tracker เพื่อขอรายชื่อ Peer ที่มีไฟล์ตรงตาม .torrent

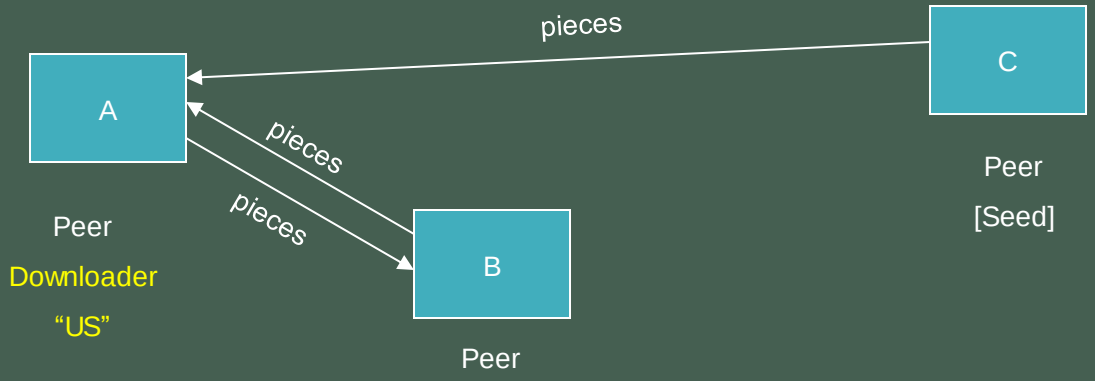
Overall Architecture



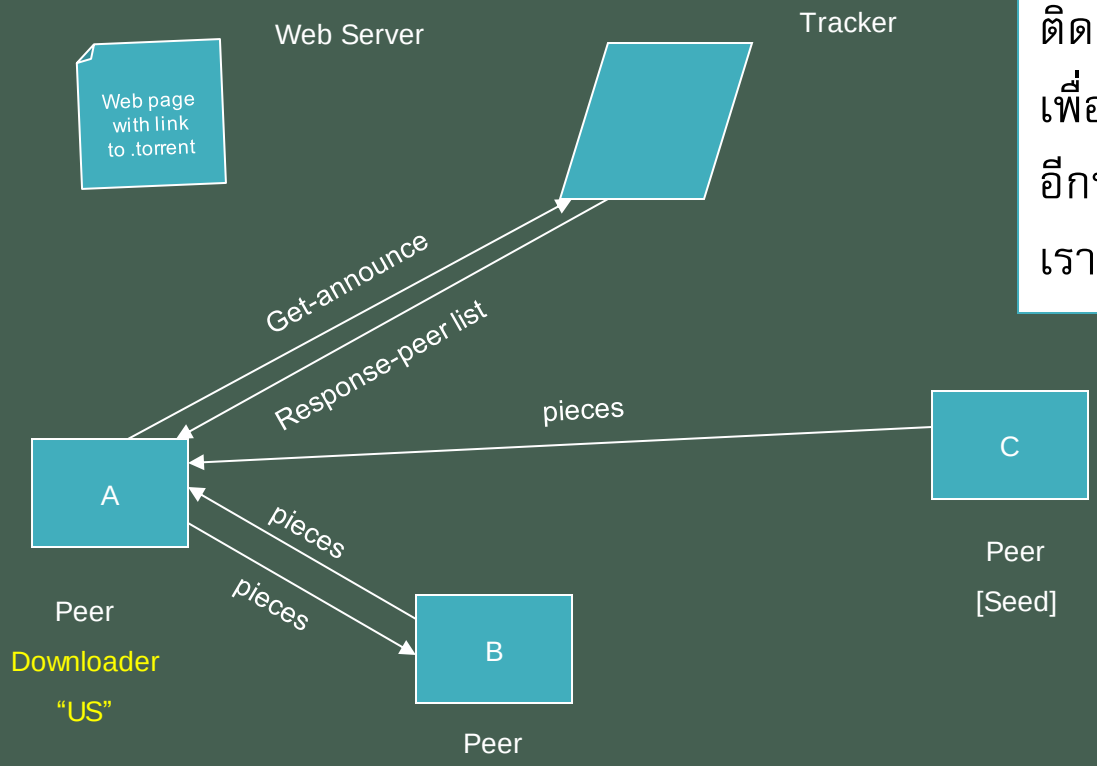
เชื่อมต่อกับ Peer จากรายชื่อที่ได้รับ เพื่อขอรับไฟล์

Overall Architecture

แต่ละ Peer ส่ง Piece ของไฟล์นั้นมาให้ ถ้าหากเรามี Piece ที่ Peer อื่นไม่มี เราสามารถเป็นคนส่งได้



Overall Architecture



ในระหว่างนั้น เราจะติดต่อกลับไปยัง Tracker เพื่อค้นหา Peer เพิ่มเติม อีกทั้งยังบอก Tracker ว่าเรามี Piece ใดบ้างแล้ว