**Peer-to-peer Protocols and Applications**

**Homework #3 P2P檔案下載模擬程式**

**請繳交程式及數據圖，Due on 2018-6-26**

請撰寫模擬程式來評估類似Bit Torrent系統中，對於決定下一個下載片段時，比較使用random selection和local rarest first兩種策略之效能差異。評估之metric為peer平均下載完成時間，而系統參數假設如下：

* 只有一個seed，假設為peer 0
* n = peer 個數 (不含seed)
* d = 每個peer鄰居個數
* fp = 檔案piece個數，假設fp = 50，每個piece大小為2Mb。每個peer的上傳及下載頻寬各為2Mbps,，seed的上傳頻寬也是2 Mbps。
* 假設每個peer只能發出一個下載request，要等到該request完成後，才能再發出下一個request
* 假設peer的服務策略是first come first server (不採用Tit-for-Tat)
* avg\_time = peer平均下載完成時間 =  peer i所需時間 / n

請繪製下列點線圖，同一張圖上並列兩種策略的數據以便比較

1. avg\_time vs. n (= 50, 100, 150, 200, 250) 假設 d = 10
2. avg\_time vs. d (= 10, 20, 30, 40, 50) 假設 N = 50

模擬程式結構流程大致如下(使用single thread即可，不需要multi-thread):

main ( ) {

for ( n = 50; n <= 250; n = n+50) {

//建立鄰居關係並記錄，peer 0為seed

for ( i = 1; i <= n; i++) {

將peer i和seed設為鄰居;

if (peer i的鄰居個數 bi少於d) 再隨機挑選(d – bi)個peer當鄰居;

}

simulate (rarest first);

simulate (random selection);

}

}

simulate (strategy) {

初始化 buffer map, request queue等

**int** currTime = -1;

while (not all peers finished) {

currTime++;

// 處理此時會服務完之request

for ( i = 0; i <= n; i++)

if ( peer i 在此時會完成上傳某個片段給peer j ) {

peer j 更新buffer map，若此時檔案全部下載完成，則記錄下載完成時間;

peer i 刪除該項來自peer j的request (因已完成);

}

}

// 鄰居交換 buffer map

for ( i = 1; i <= n; i++) peer i收集其鄰居的buffer map;

// 選擇下一個片段，發出下載request

for ( i = 1; i <= n; i++) {

if (peer i 沒有未完成的request在外)

根據strategy 挑選下一個要下載的片段，並對擁有該片段的peer中隨機挑選一個發出request，若鄰居peer皆沒有該片段，則對seed發出request;

}

// 從queue中選擇下一個要服務的request

for ( i = 0; i <= n; i++)

if ( peer i 在此時並沒有在服務任何request ) {

peer i 挑選queue裡面下一個request來服務，並記錄預計完成時間

}

}

}

return 平均下載完成時間;

}