# Bài tập Chương 9

## Bài tập 9.1

* Sự khác biệt cơ bản giữa P2P và Client-Server: Kiến trúc Client-Server có sự phân biệt rõ ràng giữa các server (cung cấp dịch vụ, tài nguyên) và các client (yêu cầu dịch vụ). Trong kiến trúc P2P thuần túy, tất cả các peer đều bình đẳng, vừa đóng vai trò là client vừa là server. Một hệ thống P2P có chỉ mục tập trung (centralized index) là một dạng lai (hybrid); nó không phải là Client-Server thuần túy vì sau khi tra cứu thông tin từ server trung tâm, việc truyền dữ liệu vẫn diễn ra trực tiếp giữa các peer.
* Ưu và nhược điểm của P2P file sharing:
  + Người dùng cuối (End-users):
    - Ưu điểm: Tiếp cận được kho tài nguyên khổng lồ, thường miễn phí, khả năng phục hồi tải xuống khi có nhiều nguồn.
    - Nhược điểm: Tốc độ tải không ổn định, tính sẵn sàng của file không được đảm bảo, nguy cơ tải phải phần mềm độc hại.
  + Chủ sở hữu file (File owners):
    - Ưu điểm: Dễ dàng phân phối nội dung với chi phí thấp.
    - Nhược điểm: Mất kiểm soát đối với việc phân phối nội dung, các vấn đề về vi phạm bản quyền.
  + Quản trị viên mạng (Network administrators):
    - Ưu điểm: Hầu như không có.
    - Nhược điểm: Gây tốn băng thông mạng rất lớn, có thể được sử dụng cho các hoạt động bất hợp pháp, khó theo dõi và kiểm soát.

## Bài tập 9.2

* Ưu và nhược điểm của việc phân lớp mạng P2P:
  + Ưu điểm: Trừu tượng hóa khỏi sự phức tạp của mạng vật lý, cho phép tạo ra các topo logic tùy chỉnh (vòng, cây,…) được tối ưu hóa cho các tác vụ cụ thể, và dễ dàng quản lý việc các peer tham gia/rời khỏi mạng một cách linh hoạt.
  + Nhược điểm: Sự gần gũi logic trên mạng P2P không đảm bảo sự gần gũi vật lý. Một “bước nhảy” (hop) logic đến một peer “hàng xóm” có thể tương ứng với một đường truyền vật lý rất dài, dẫn đến định tuyến không hiệu quả và độ trễ cao.
* Tác động đến các loại mạng:
  + Mạng không cấu trúc: Bị ảnh hưởng nặng nề. Cơ chế flooding trở nên rất tốn kém và không hiệu quả nếu các peer logic gần nhau lại ở xa nhau về mặt vật lý.
  + Mạng có cấu trúc (DHT): Bị ảnh hưởng đáng kể. Các đảm bảo về định tuyến (ví dụ: O(log n)) là về số bước nhảy logic, không phải độ trễ vật lý. Các DHT nhận biết sự gần gũi (proximity-aware) là một giải pháp cho vấn đề này.
  + Mạng Superpeer: Ít bị ảnh hưởng hơn. Các superpeer có thể được chọn dựa trên sự ổn định và băng thông cao của mạng vật lý, và các peer thông thường có thể kết nối đến các superpeer gần chúng nhất về mặt địa lý.

## Bài tập 9.3

Sử dụng peer dưới cùng bên trái trong Hình 9.4 làm peer khởi tạo.

* Flooding với TTL=3:
  + Minh họa: Peer khởi tạo gửi yêu cầu đến 2 hàng xóm (TTL=1). 2 hàng xóm này gửi tiếp đến các hàng xóm chưa nhận của chúng (TTL=2). Các peer nhận được ở TTL=2 lại gửi tiếp (TTL=3). Quá trình sẽ lan tỏa ra một phần lớn của mạng.
  + Tính đầy đủ của kết quả: Không được đảm bảo. Kết quả chỉ được tìm thấy nếu peer chứa tài nguyên nằm trong phạm vi 3 bước nhảy logic từ peer khởi tạo.
* Gossiping (với partial view ≤ 3):
  + Minh họa: Peer khởi tạo chọn ngẫu nhiên 1 trong số các hàng xóm của nó và gửi yêu cầu. Peer đó lại chọn ngẫu nhiên 1 hàng xóm của nó để gửi tiếp, và cứ thế.
  + Tính đầy đủ của kết quả: Không được đảm bảo một cách chắc chắn. Tuy nhiên, nó có xác suất cao sẽ tìm thấy tài nguyên theo thời gian nếu có nhiều vòng gossiping được khởi tạo. Đây là cơ chế dựa trên xác suất.

## Bài tập 9.4

So sánh các loại DHT trên thang điểm 1-5 (1: Kém, 5: Tốt):

| Tiêu chí | Trie (Cây) | Hypercube | Ring (Vòng) |
| --- | --- | --- | --- |
| Autonomy | 2 | 2 | 2 |
| Query Expressiveness | 3 (Tốt hơn cho range query) | 1 | 1 |
| Efficiency (Lookup) | 5 | 5 | 5 |
| QoS (Lookup) | 5 | 5 | 5 |
| Fault-tolerance | 4 | 3 | 5 |
| Security | 2 | 2 | 2 |

## Bài tập 9.5

Các cặp (key, data) cho một ứng dụng mạng xã hội trên DHT:

* Hồ sơ người dùng (User Profile): (hash(UserID), UserProfile\_Object)
* Danh sách bạn bè (Friend List): (hash("friends\_of\_" + UserID), List\_of\_FriendIDs)
* Bài đăng (Posts):
  + (hash(PostID), Post\_Object)
  + (hash("posts\_by\_" + UserID), List\_of\_PostIDs)
* Bình luận (Comments):
  + (hash("comments\_on\_" + PostID), List\_of\_Comment\_Objects)

## Bài tập 9.6

Mô tả các thao tác dựa trên thiết kế ở bài 9.5:

* Tạo người dùng NewUser: put(hash(NewUserID), NewUserProfile\_Object)
* Xóa người dùng OldUser: delete(hash(OldUserID)). (Khó khăn: phải cập nhật danh sách bạn bè của tất cả những người đã kết bạn với OldUser).
* Kết bạn (UserA kết bạn với UserB):
  1. listA = get(hash("friends\_of\_" + UserA\_ID))
  2. listA.add(UserB\_ID)
  3. put(hash("friends\_of\_" + UserA\_ID), listA)
  4. Lặp lại các bước 1-3 cho UserB.
* Đọc bài đăng của bạn bè (cho MyUser):
  1. friend\_list = get(hash("friends\_of\_" + MyUserID))
  2. Lặp qua từng friendID trong friend\_list:
     + post\_id\_list = get(hash("posts\_by\_" + friendID))
     + Lặp qua từng postID trong post\_id\_list:
       - post = get(hash(postID))

Nhược điểm: Thao tác đọc tin tức (news feed) yêu cầu rất nhiều lệnh get, gây độ trễ cao. Việc xóa người dùng/hủy kết bạn không phải là một thao tác nguyên tử và rất phức tạp.

## Bài tập 9.7

Thiết kế mới để dữ liệu riêng tư được lưu tại peer của người dùng:

* (key, data) trong DHT: DHT chỉ lưu trữ con trỏ đến peer, không lưu dữ liệu. (hash(UserID), Peer\_IP\_Address)
* Thao tác đọc hồ sơ của bạn bè:
  1. Thực hiện get(hash(FriendID)) trên DHT để lấy địa chỉ IP của người bạn.
  2. Kết nối trực tiếp đến địa chỉ IP đó và yêu cầu dữ liệu hồ sơ.

Ưu điểm: Người dùng có toàn quyền kiểm soát và bảo mật dữ liệu của mình.

Nhược điểm: Tính sẵn sàng rất thấp. Nếu một người dùng ngoại tuyến, tất cả dữ liệu của họ (hồ sơ, bài đăng) sẽ hoàn toàn không thể truy cập được bởi bạn bè của họ.

## Bài tập 9.8

* Điểm chung: Cả hai đều giải quyết vấn đề không đồng nhất về lược đồ.
* Điểm khác biệt:
  + Hệ đa CSDL (MDBS): Thường có một lược đồ toàn cục (GCS) được định nghĩa trước hoặc được tích hợp một cách có kiểm soát. Kiến trúc thường là tập trung (mediator-wrapper). Quá trình tích hợp diễn ra một lần và tương đối tĩnh.
  + Hệ thống P2P: Không có GCS, hoàn toàn phi tập trung. Các ánh xạ được định nghĩa theo cặp (pairwise) giữa các peer. Mạng lưới và các ánh xạ có tính động rất cao, các peer có thể tham gia/rời đi bất cứ lúc nào.

## Bài tập 9.9

* FD với Random Walk: Thay vì gửi truy vấn đến tất cả hàng xóm, peer chỉ gửi đến một hàng xóm được chọn ngẫu nhiên.
  + Ưu điểm: Giảm đáng kể lưu lượng mạng so với flooding.
  + Nhược điểm: Tăng độ trễ, và có nguy cơ cao không tìm thấy kết quả nếu đường đi ngẫu nhiên không đến được peer chứa dữ liệu.
* FD với Gossiping: Peer gửi truy vấn đến một tập con các hàng xóm được chọn ngẫu nhiên.
  + Ưu điểm: Cân bằng tốt hơn giữa việc giảm lưu lượng mạng và khả năng tìm thấy kết quả so với random walk.
  + Nhược điểm: Vẫn có tính xác suất, không đảm bảo tìm thấy tất cả kết quả.

## Bài tập 9.10

Áp dụng thuật toán TPUT trên 3 danh sách trong Hình 9.10 với k=3:

1. Phase 1: Mỗi list holder gửi 3 mục đầu tiên. Query originator nhận được {d1, d4, d9} từ L1, {d2, d6, d7} từ L2, {d3, d5, d8} từ L3.
   * Z (3 mục có tổng điểm riêng phần cao nhất) sẽ là {d3, d5, d8}.
   * λ1 (tổng điểm riêng phần của mục thứ 3) là 28 (của d8).
2. Phase 2: Gửi ngưỡng τ = λ1/m = 28/3 ≈ 9.3 đến tất cả các list holder.
   * Các holder gửi lại tất cả các mục có điểm cục bộ ≥ 9.3.
   * Dựa trên các điểm số mới nhận được, λ2 được tính lại. Các mục ứng cử viên có điểm giới hạn trên u(d) < λ2 sẽ bị loại bỏ.
3. Phase 3: Query originator yêu cầu điểm số còn thiếu cho các mục ứng cử viên còn lại và tính toán 3 mục có điểm tổng cao nhất cuối cùng. Kết quả cuối cùng là {d3, d5, d8}.

## Bài tập 9.11

Áp dụng thuật toán DHTop:

1. Phase 1 (Chuẩn bị danh sách subdomain): Với mỗi thuộc tính trong hàm tính điểm, thuật toán tạo một danh sách các subdomain của thuộc tính đó, được sắp xếp theo thứ tự giảm dần về tác động tích cực đến điểm số. Các subdomain không thỏa mãn điều kiện của truy vấn sẽ bị loại bỏ.
2. Phase 2 (Truy xuất và tính toán):
   * Thuật toán truy xuất song song các giá trị thuộc tính từ các peer chịu trách nhiệm cho subdomain đầu tiên trong mỗi danh sách.
   * Với mỗi giá trị thuộc tính nhận được, nó truy xuất tuple tương ứng, tính điểm tổng thể.
   * Nó duy trì một danh sách k tuple có điểm cao nhất cho đến nay và một giá trị ngưỡng (threshold).
   * Ngưỡng được tính dựa trên các giá trị thuộc tính cuối cùng được nhìn thấy từ mỗi danh sách.
   * Quá trình dừng lại khi điểm số của k tuple cao nhất đều lớn hơn hoặc bằng ngưỡng. Nếu chưa dừng, nó tiếp tục với subdomain tiếp theo.

## Bài tập 9.12

Cải tiến thuật toán PIERjoin khi một trong hai quan hệ (ví dụ R) đã được băm trên thuộc tính nối:

* Giai đoạn Multicast: Tương tự, gửi truy vấn Q đến các peer trong home của R và S.
* Giai đoạn Hash: Giai đoạn này được bỏ qua hoàn toàn cho quan hệ R. Các peer trong home của R không cần làm gì. Các peer trong home của S vẫn băm các tuple của S và gửi chúng đến home của R (vì R đã ở đúng vị trí).
* Giai đoạn Probe/Join: Các peer trong home của R nhận các tuple từ S. Thay vì xây dựng 2 bảng băm, chúng chỉ cần xây dựng bảng băm cho các tuple S đến và dò tìm trên các tuple R cục bộ của chúng (hoặc ngược lại).

Sự cải tiến này giúp giảm một nửa chi phí truyền thông và xử lý của giai đoạn hash.

## Bài tập 9.13

Sử dụng sao chép tại các peer hàng xóm thay vì dùng nhiều hàm băm:

* Tác động đến kịch bản trong Ví dụ 9.7: Khi put(k, d1) được thực hiện tại p1, p1 sẽ sao chép bản cập nhật này đến các hàng xóm của nó. Nếu p2 bị lỗi, các hàng xóm của p1 vẫn có bản sao mới nhất. Khi p2 kết nối lại, nó có thể thực hiện đối chiếu (reconcile) với các hàng xóm của mình để lấy trạng thái mới nhất.
* Ưu điểm: Tăng tính cục bộ của dữ liệu sao chép (bản sao nằm gần bản chính về mặt logic), có thể nhanh hơn.
* Nhược điểm: Tính sẵn sàng của dữ liệu phụ thuộc vào tính sẵn sàng của một nhóm các peer lân cận. Nếu cả một “cụm” hàng xóm cùng bị lỗi, dữ liệu có thể bị mất. Tải sao chép tập trung vào các hàng xóm của peer gốc.

## Bài tập 9.14

* Điểm chung: Cả hai đều là sổ cái phân tán, dựa trên mạng P2P, sử dụng mã hóa và cơ chế đồng thuận.
* Điểm khác biệt và Giao thức xác thực:

| * Tiêu chí | * Public Blockchain | * Private Blockchain |
| --- | --- | --- |
| * Tham gia | * Không cần cấp phép (Permissionless). Bất kỳ ai cũng có thể tham gia. | * Cần cấp phép (Permissioned). Chỉ các thành viên được xác định mới có thể tham gia. |
| * Danh tính | * Ẩn danh/Bút danh (Anonymous/Pseudonymous). | * Đã biết (Known identities). |
| * Giao thức đồng thuận | * Cần chống lại các tấn công Byzantine. Thường dùng các cơ chế tốn kém như Proof-of-Work (PoW). | * Các thành viên được tin tưởng hơn. Có thể dùng các giao thức hiệu quả hơn như Paxos hoặc Raft (chịu lỗi crash, không phải Byzantine). |
| * Hiệu năng | * Chậm, thông lượng thấp. | * Nhanh, thông lượng cao hơn nhiều. |
| * Bảo mật dữ liệu | * Tất cả các giao dịch đều công khai. | * Có thể hạn chế quyền truy cập vào các giao dịch. |