PHENIKAA UNIVERSITY

# ỨNG DỤNG PHÂN TÁN

## III. Trao đổi thông tin

**Nhóm chuyên môn UDPT – Trường Công nghệ thông tin Phenikaa**
(Trần Đăng Hoan, Đỗ Quốc Trường, Nguyễn Thành Trung, Phạm Kim Thành, Nguyễn Hữu Đạt)

# NỘI DUNG

1. Trao đổi thông tin giữa các tiến trình
2. Lời gọi thủ tục từ xa
3. Trao đổi thông tin hướng thông điệp
4. Trao đổi thông tin hướng dòng

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

- 1.1 Các giao thức phân tầng
- 1.2 Trao đổi thông tin bằng UDP
- 1.3 Trao đổi thông tin bằng TCP

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

- Giao thức
  - Cấu trúc thông điệp
  - Kích cỡ thông điệp
  - Thứ tự gửi thông điệp
  - Cơ chế phát hiện thông điệp hỏng hay bị mất
  - V.v….
- Phân tầng
- Các loại giao thức:

  Các loại giao thức trong mạng máy tính có thể được phân loại dựa trên hai tiêu chí chính: **hướng kết nối (connection-oriented)** và **tin cậy (reliable)**

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

## Giao thức hướng kết nối (Connection-oriented)

- **Định nghĩa**: Giao thức yêu cầu thiết lập kết nối trước khi trao đổi dữ liệu. Hai bên (máy gửi và máy nhận) cần đồng ý và duy trì kết nối trong suốt quá trình truyền thông.
- **Đặc điểm**:
  - Đảm bảo dữ liệu được truyền theo thứ tự.
  - Có kiểm tra lỗi và cơ chế phục hồi dữ liệu bị mất.
  - Thứ tự gửi thông điệp
  - Chậm hơn vì cần thiết lập và duy trì kết nối.
- **Ví dụ giao thức**:
  - **TCP (Transmission Control Protocol)**: Giao thức hướng kết nối phổ biến nhất, đảm bảo dữ liệu truyền đi đáng tin cậy.
  - **SCTP (Stream Control Transmission Protocol)**: Được thiết kế cho các ứng dụng thời gian thực nhưng vẫn hướng kết nối.

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

## Giao thức không hướng kết nối (Connectionless)

- **Định nghĩa**: Giao thức không yêu cầu thiết lập hoặc duy trì kết nối trước khi gửi dữ liệu. Dữ liệu được gửi đi ngay dưới dạng các gói độc lập.
- **Đặc điểm**:
  - Dữ liệu có thể bị mất hoặc đến không theo thứ tự.
  - Không có cơ chế xác nhận hoặc sửa lỗi.
  - Nhanh hơn vì không cần thiết lập kết nối.
- **Ví dụ giao thức**:
  - **UDP (User Datagram Protocol)**: Giao thức không hướng kết nối phổ biến nhất, được sử dụng cho các ứng dụng yêu cầu tốc độ cao như streaming video, VoIP.
  - **ICMP (Internet Control Message Protocol)**: Giao thức không hướng kết nối dùng để truyền thông tin lỗi hoặc trạng thái mạng, ví dụ lệnh ping.

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

**Giao thức tin cậy (Reliable Protocol)**

- **Định nghĩa**: Giao thức đảm bảo rằng dữ liệu được truyền đi một cách đáng tin cậy, không mất mát, không bị trùng lặp, và đến đích theo đúng thứ tự.
- **Đặc điểm**:
  - Sử dụng cơ chế xác nhận (acknowledgment) và truyền lại (retransmission) khi phát hiện lỗi.
  - Thích hợp cho các ứng dụng yêu cầu độ chính xác cao.
- **Ví dụ giao thức**:
  - **TCP**: Tin cậy, vì nó có cơ chế số thứ tự, xác nhận, và truyền lại dữ liệu bị mất.
  - **SCTP**: Cũng là một giao thức tin cậy, hỗ trợ đa luồng dữ liệu trong một kết nối.

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

**Giao thức không tin cậy (Unreliable Protocol)**

- **Định nghĩa**: Giao thức không đảm bảo dữ liệu sẽ đến đích hoặc đến theo đúng thứ tự. Các gói dữ liệu có thể bị mất mà không có cơ chế khắc phục.
- **Đặc điểm**:
  - Đơn giản, nhanh chóng, nhưng không đảm bảo chất lượng.
  - Phù hợp với các ứng dụng mà mất một phần dữ liệu không gây ảnh hưởng lớn.
- **Ví dụ giao thức**:
  - **UDP**: Không tin cậy, vì nó không có cơ chế xác nhận hoặc truyền lại.
  - **ICMP**: Không tin cậy, vì chỉ dùng để trao đổi thông tin về lỗi hoặc kiểm tra trạng thái mạng.

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

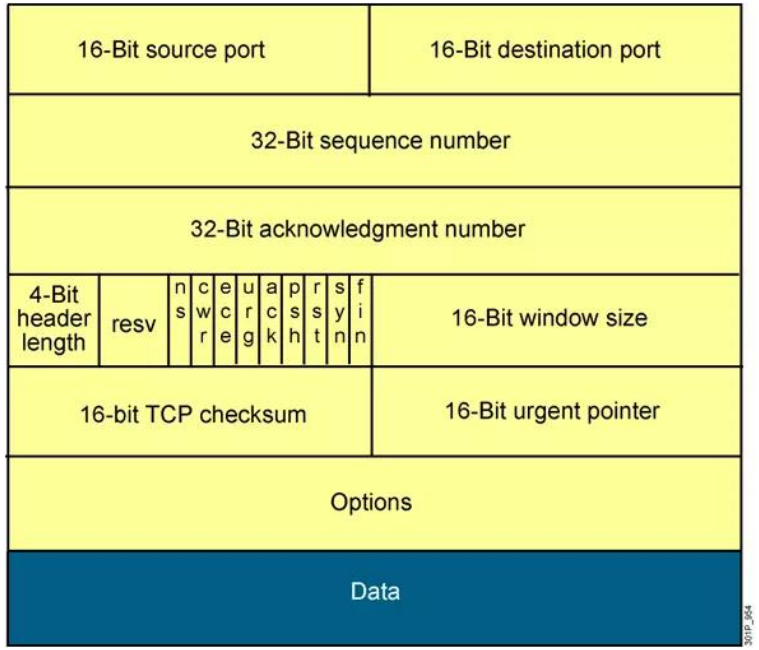
## Tổng hợp

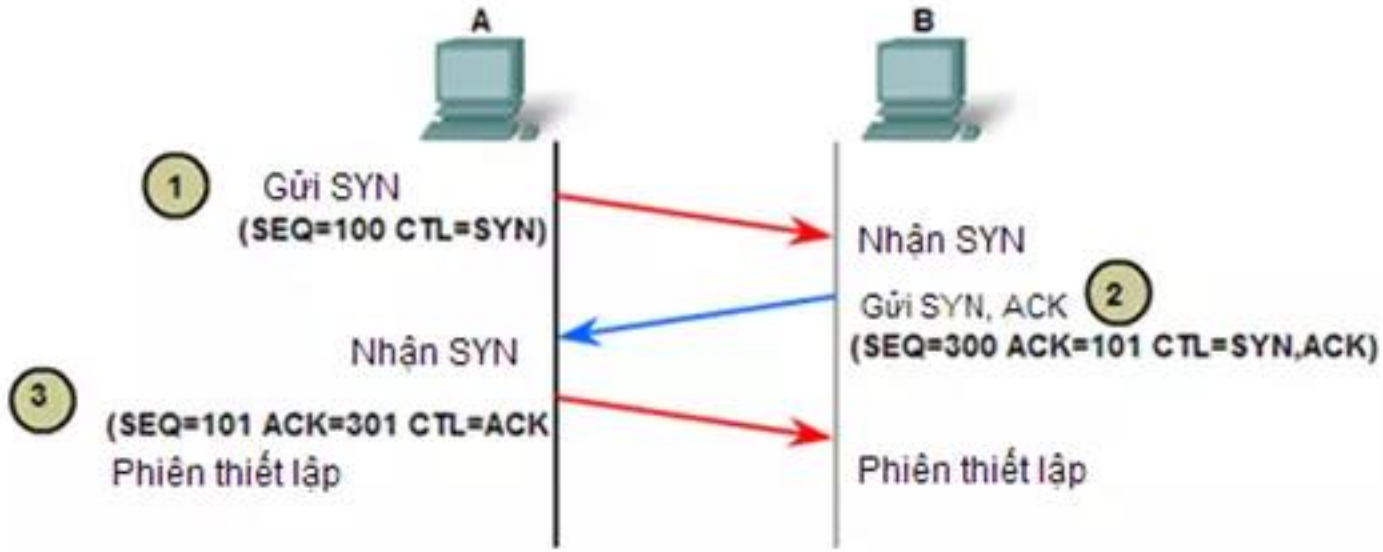
| Loại giao thức | Hướng kết nối | Không hướng kết nối |
|---|---|---|
| Tin cậy | TCP, SCTP | Không phổ biến |
| Không tin cậy | Không phổ biến | UDP, ICMP |

★ ***Các tiến trình trao đổi thông tin với nhau thông qua chủ yếu 2 giao thức TCP và UDP***

# TRAO ĐỔI THÔNG TIN BẰNG TCP

- Các tiến trình trao đổi thông tin với nhau thông qua chủ yếu 2 giao thức TCP và UDP
  - TCP

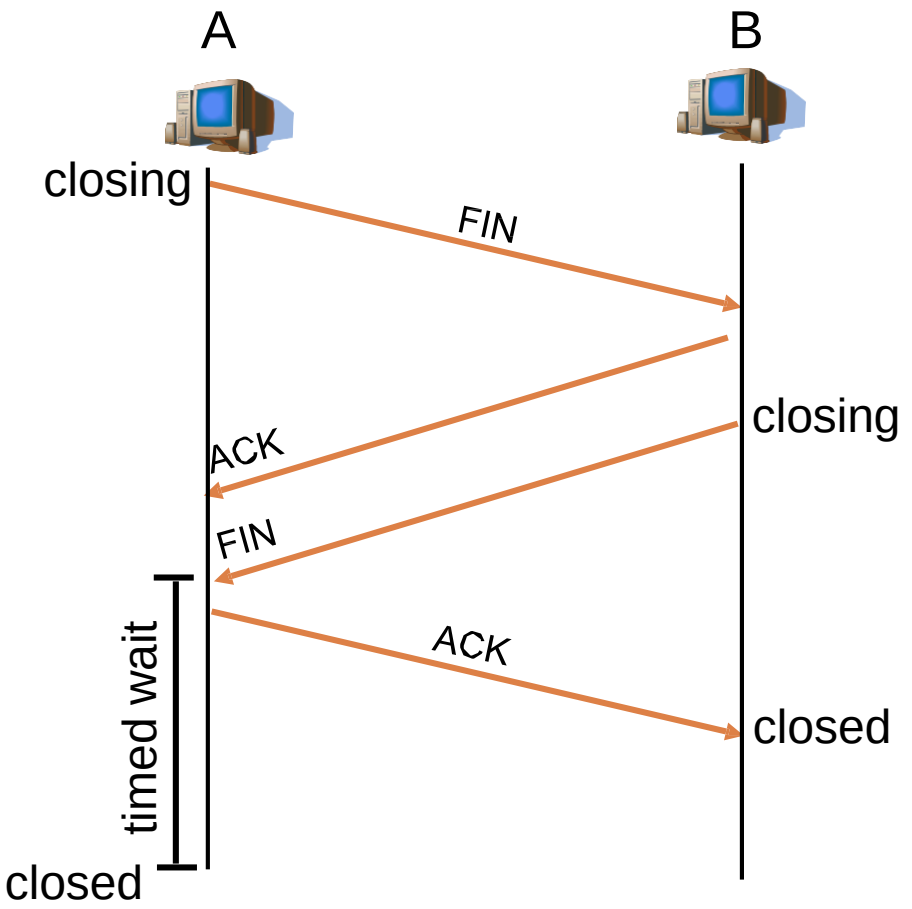


# VÍ DỤ VỀ VIỆC ĐÓNG LIÊN KẾT

- Bước 1: Gửi FIN cho B
- Bước 2: B nhận được FIN, trả lời ACK, đồng thời đóng liên kết và gửi FIN.
- Bước 3: A nhận FIN, trả lời ACK, vào trạng thái chờ
- Bước 4: B nhận ACK. đóng liên kết.

Lưu ý: Cả hai bên đều có thể chủ động đóng liên kết



# TRAO ĐỔI THÔNG TIN BẰNG UDP

- Các tiến trình trao đổi thông tin với nhau thông qua chủ yếu 2 giao thức TCP và UDP
  - UDP

# 1. TRAO ĐỔI THÔNG TIN GIỮA CÁC TIẾN TRÌNH

**Khi nào nên chọn TCP hoặc UDP trong trao đổi thông tin giữa tiến trình?**

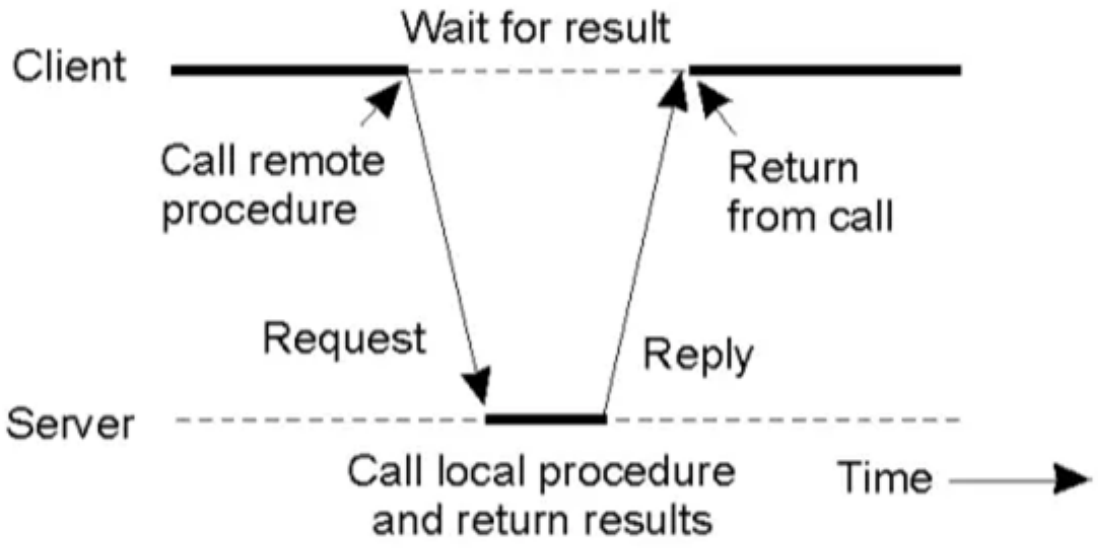
- **Chọn TCP**: Khi cần độ tin cậy, dữ liệu phải đến đúng thứ tự, ví dụ:
  - Gửi file, email. Dịch vụ web.
  - Ứng dụng ngân hàng.
- **Chọn UDP**: Khi cần tốc độ và chấp nhận mất mát dữ liệu nhỏ, ví dụ:
  - Truyền thông trực tuyến (video, âm thanh).
  - Ứng dụng IoT cần nhẹ và nhanh.
  - Trò chơi trực tuyến.

# 2. LỜI GỌI THỦ TỤC TỪ XA

- Lời gọi thủ tục từ xa
  - Cơ chế Remote Procedure Call

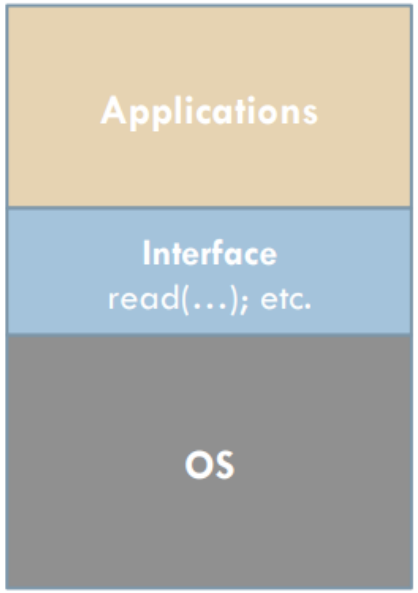
Khi một tiến trình trên client muốn thực hiện một thủ tục nào đó nằm trên server:

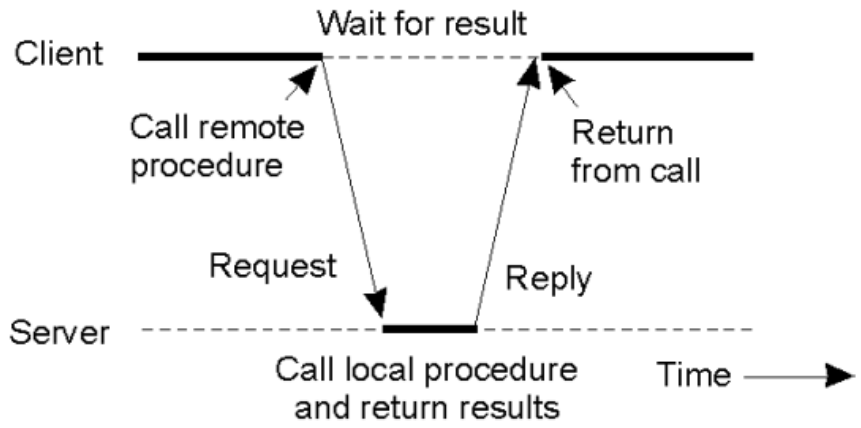
- Thực hiện một lời gọi thủ tục từ xa tới server.
- Tiến trình gọi trên client sẽ vào trạng thái chờ (treo), và quá trình thực thi thủ tục được gọi diễn ra trên server dựa trên các tham số được truyền đến từ client và kết quả sẽ được truyền trở lại cho client tương ứng.



# 2. LỜI GỌI THỦ TỤC TỪ XA

**Lời gọi thủ tục từ xa (Remote Procedure Call - RPC)** là một cơ chế trong lập trình phân tán, cho phép một chương trình máy tính thực hiện một **thủ tục (hàm, phương thức)** trên một máy tính khác (máy chủ từ xa) như thể thủ tục đó đang được thực hiện cục bộ trên cùng một máy.

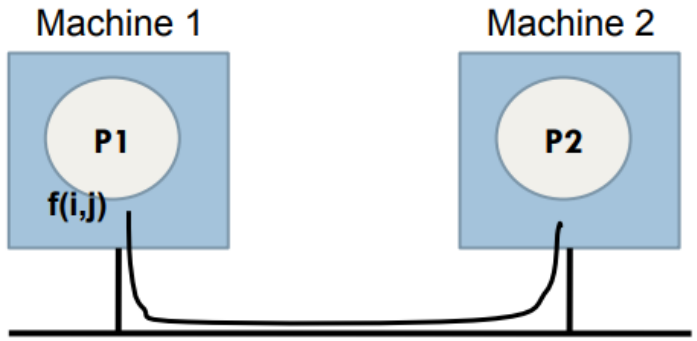


# 2. LỜI GỌI THỦ TỤC TỪ XA

**Đặc điểm của RPC**

- **Minh bạch**: RPC giúp lập trình viên gọi thủ tục từ xa mà không cần quan tâm đến chi tiết truyền thông mạng. Nó giống như gọi một hàm cục bộ => Cơ chế truy cập trong suốt với người dùng
- **Đồng bộ hóa**: Gọi RPC thường là đồng bộ, tức là chương trình máy khách sẽ chờ kết quả từ máy chủ.
- **Tầng giao thức**: RPC thường sử dụng các giao thức mạng như TCP hoặc UDP để truyền tải dữ liệu.



# 2. LỜI GỌI THỦ TỤC TỪ XA

**Thành phần của RPC**

- **Client Stub**: Là mã chương trình phía máy khách, chịu trách nhiệm đóng gói tham số và gửi yêu cầu đến máy chủ.
- **Server Stub**: Là mã chương trình phía máy chủ, nhận yêu cầu từ Client Stub, giải mã tham số, thực hiện thủ tục, và gửi trả kết quả.
- **Protocol**: Là giao thức truyền thông giữa Client Stub và Server Stub (thường là TCP/IP).

# 2. LỜI GỌI THỦ TỤC TỪ XA

## Ưu điểm của RPC

- **Đơn giản hóa việc lập trình**: Lập trình viên có thể gọi các thủ tục từ xa một cách tự nhiên như các hàm cục bộ.
- **Trong suốt**: Che giấu các chi tiết phức tạp của giao tiếp mạng.
- **Tính linh hoạt**: RPC có thể được sử dụng trong các hệ thống phân tán với nhiều ngôn ngữ và nền tảng khác nhau.

## Nhược điểm của RPC:

- **Phụ thuộc vào mạng**: Nếu mạng chậm hoặc bị ngắt, hiệu suất hoặc tính sẵn sàng của RPC sẽ bị ảnh hưởng.
- **Độ trễ cao hơn so với gọi cục bộ**: Do phải truyền dữ liệu qua mạng và thực hiện quá trình đóng gói/giải mã.
- **Khả năng xử lý lỗi phức tạp**:RPC phải xử lý các lỗi liên quan đến mạng, máy chủ, hoặc tham số.

# 2. LỜI GỌI THỦ TỤC TỪ XA

**Biến thể và công nghệ liên quan**

- **gRPC**: Một framework RPC hiện đại từ Google, sử dụng giao thức HTTP/2 để truyền dữ liệu và Protobuf để mã hóa.
- **XML-RPC**: RPC sử dụng XML để mã hóa dữ liệu và HTTP để truyền dữ liệu.
- **JSON-RPC**: RPC sử dụng JSON làm định dạng trao đổi dữ liệu.

# 2. LỜI GỌI THỦ TỤC TỪ XA

**Ví dụ minh họa**: cách triển khai **RPC (Remote Procedure Call)** cơ bản sử dụng Python. Ví dụ này sử dụng **xmlrpc** module, một thư viện có sẵn trong Python để xây dựng RPC dựa trên giao thức **XML-RPC**.

**Link code**:
https://drive.google.com/drive/folders/1vOUGrEEk9Z2g0KTKxLEmHEzBNBycJZ6S?usp=sharing

# 2. LỜI GỌI THỦ TỤC TỪ XA

**Câu hỏi**: API HTTP có phải là Lời gọi thủ tục từ xa **RPC (Remote Procedure Call)** không?

# 2. LỜI GỌI THỦ TỤC TỪ XA

**Câu trả lời**: **Tùy thuộc vào ngữ cảnh và cách thức sử dụng**, gọi API HTTP có thể được coi là một dạng của RPC, nhưng không phải tất cả các lời gọi HTTP API đều là RPC. Dưới đây là phân tích chi tiết:

- Nếu bạn sử dụng HTTP để gửi yêu cầu thực hiện một thủ tục (thường là qua POST hoặc GET) và nhận kết quả trả về, thì đó có thể được coi là một dạng của RPC.

  VD: **POST /api/calculate** Body: { "operation": "add", "a": 5, "b": 10 }

- Nếu API tuân theo REST (Representational State Transfer)

  VD: **GET /api/users/123** (Trả về thông tin của người dùng có ID 123. Đây không phải là RPC vì bạn không gọi một thủ tục mà đang truy xuất một tài nguyên.)

# Tính mở của RPC

- Client và Server được cài đặt bởi các NSX khác nhau
- Giao diện thống nhất client và server
  - Không phụ thuộc công cụ và ngôn ngữ lập trình
  - Mô tả đầy đủ và trung lập
  - Thường dùng ngôn ngữ định nghĩa giao diện

# Ví dụ RPC

- Distributed Computing Environment (DCE) được phát  triển bởi Open Group: http://www.opengroup.org/dce/
- Tầng middleware
- Mô hình client-server
- Giao tiếp được thực hiện thông qua RPCs
- Các dịch vụ:
  - Distributed file service
  - Directory service
  - Distributed time service

# Thực hành: Xây dựng RPC trên Windows

- Link: https://docs.microsoft.com/en-us/windows/win32/rpc/tutorial
- Create interface definition and application configuration files.
- Use the MIDL compiler to generate C-language client and server stubs and headers from those files.
- Write a client application that manages its connection to the server.
- Write a server application that contains the actual remote procedures.
- Compile and link these files to the RPC run-time library to produce the distributed application.

# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

**3.1 Trao đổi thông tin hướng thông điệp tạm thời**

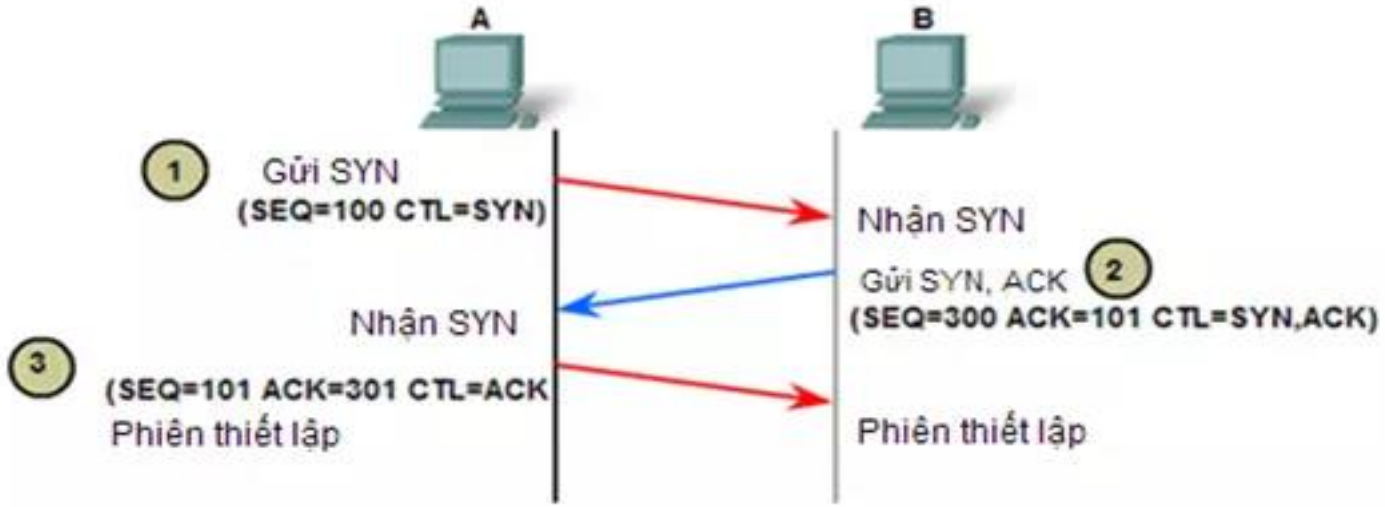
**3.2 Trao đổi thông tin hướng thông điệp bền vững**

# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

**Trao đổi thông tin hướng thông điệp tạm thời**

- 2 tiến trình cùng chạy, gửi và nhận trực tiếp cho nhau.
- Phải đảm bảo rằng hai bên (client và server) đồng ý kết nối trước khi trao đổi dữ liệu

**Nhắc lại : Giao thức bắt tay 3 bước**



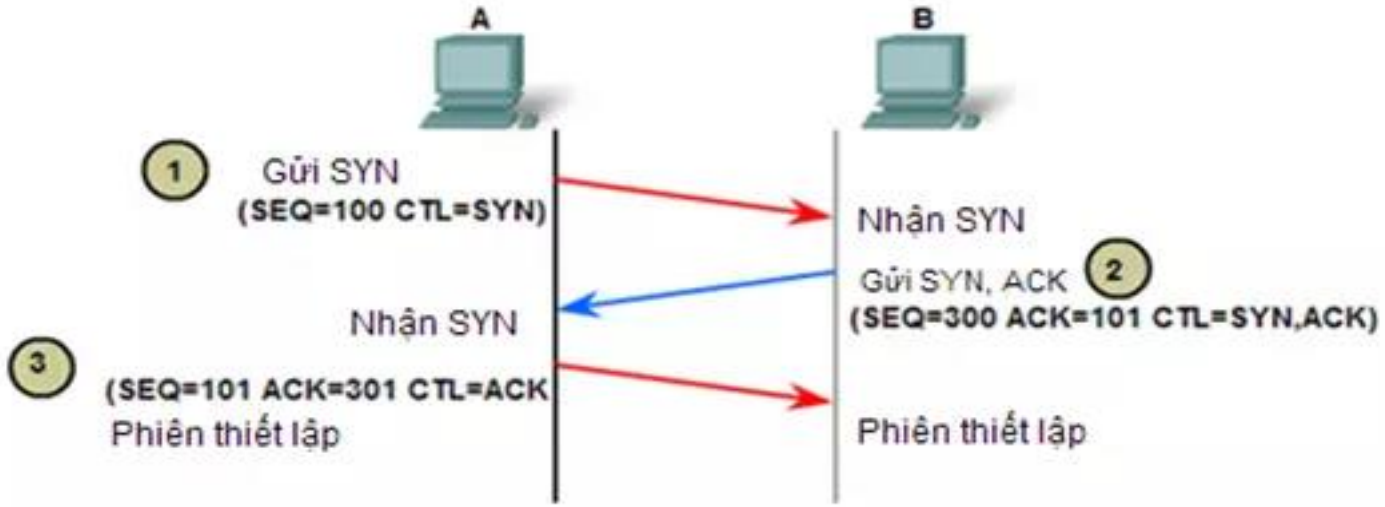

- Bước 1: A gửi SYN cho B
  - chỉ ra giá trị khởi tạo seq # của A
  - không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYNACK
  - B khởi tạo vùng đệm
  - chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYNACK, trả lời ACK, có thể kèm theo dữ liệu

# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

**Trao đổi thông tin hướng thông điệp tạm thời**

- 2 tiến trình cùng chạy, gửi và nhận trực tiếp cho nhau.
- Phải đảm bảo rằng hai bên (client và server) đồng ý kết nối trước khi trao đổi dữ liệu

**Nhắc lại : Giao thức bắt tay 3 bước**




- Bước 1: A gửi SYN cho B
  - chỉ ra giá trị khởi tạo seq # của A
  - không có dữ liệu
- Bước 2: B nhận SYN, trả lời bằng SYNACK
  - B khởi tạo vùng đệm
  - chỉ ra giá trị khởi tạo seq. # của B
- Bước 3: A nhận SYNACK, trả lời ACK, có thể kèm theo dữ liệu

# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

**Trao đổi thông tin hướng thông điệp bền vững**

- MOM (Messeage-Oriented Middleware)
- Hệ thống hàng đợi thông điệp hỗ trợ trao đổi thông tin không đồng bộ bền vững
- Hỗ trợ khả năng lưu trữ trung gian cho thông điệp
- Chấp nhận độ trễ thời gian cao
- Ví dụ: hệ thống email

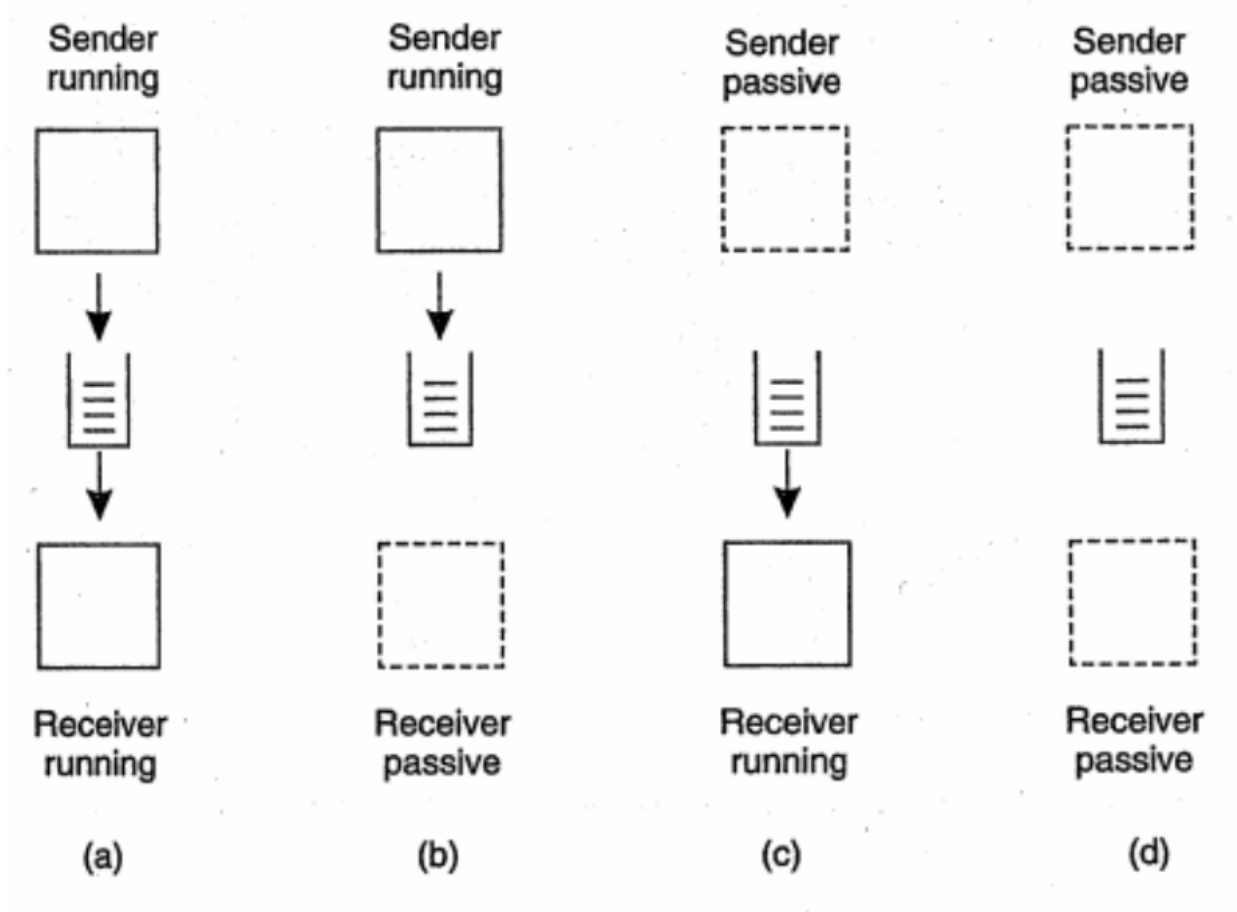
# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

**Massage Queues (Hàng đợi thông điệp)**

- Khái niệm: Một hàng đợi chứa các tin nhắn được gửi từ một nhà cung cấp đến một hoặc nhiều người dùng. Và được xử lý theo thứ tự
- Chức năng:
  - Cung cấp cơ chế lưu trữ tin nhắn tạm thời
  - Cho phép xử lý bất đồng bộ và phân tán
- Ví dụ: RabbitMQ, Apache ActiveMQ

# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

## Mô hình hàng đợi thông điệp

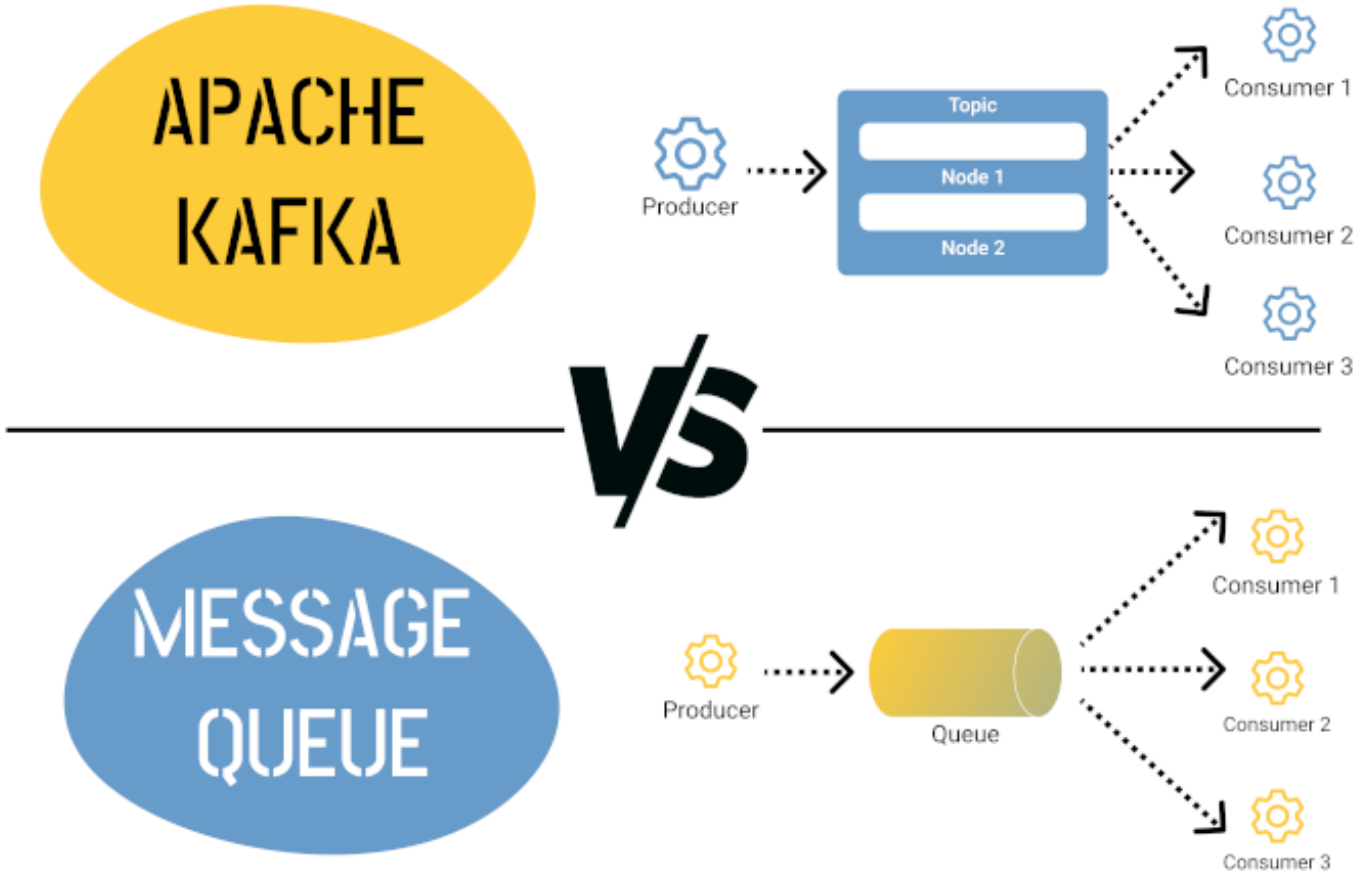


# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP

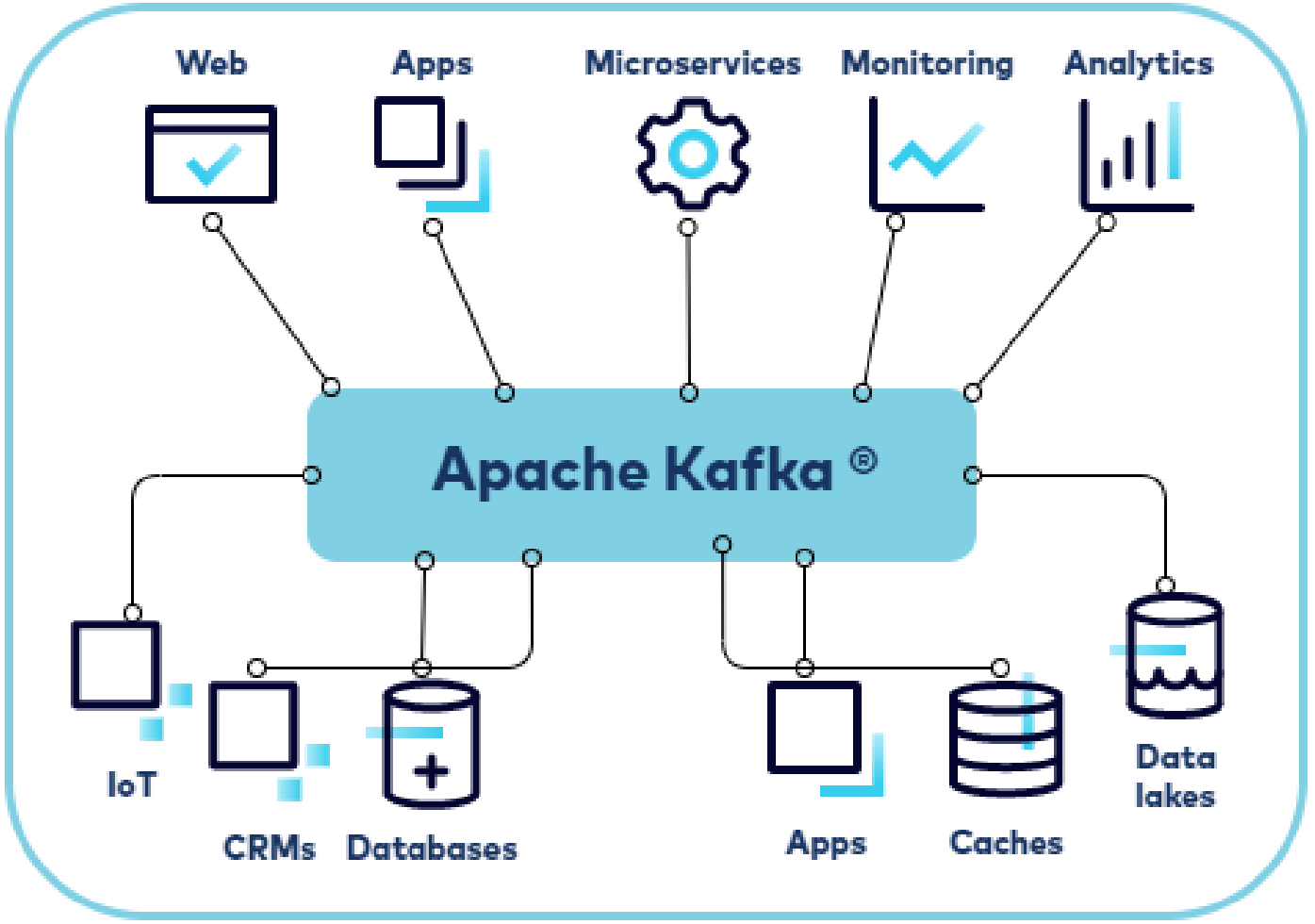
**Publish-Subscribe Model**

- Khái niệm: cho phép các thành phần gửi tin nhắn (publishers) và nhận tin nhắn (subscribers) mà không cần biết đến nhau.
- **Message broker** là công cụ thực hiện mô hình.
- Chức năng:
  - Publishers phát tán các sự kiện hoặc tin nhắn đến chủ đề (topics).
  - Subscribers đăng ký nhận thông báo từ các chủ đề mà họ quan tâm
- Ví dụ: RabbitMQ, Kafka, ActiveMQ, Redis Pub/Sub, Google Pub/Sub.

# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP



# 3. TRAO ĐỔI THÔNG TIN HƯỚNG THÔNG ĐIỆP



# 4. TRAO ĐỔI THÔNG TIN HƯỚNG DÒNG

**Hỗ trợ cho phương tiện truyền thông liên tục**

- Phương tiện truyền đạt thông tin
  - Lưu trữ
  - Truyền tin
  - Biểu diễn (màn hình, v.v…)
- Phương tiện truyền thông liên tục/rời rạc
- Yếu tố thời gian là quan trọng trong việc trao đổi thông tin
- Ví dụ: RTSP Streaming Video,…

# 4. TRAO ĐỔI THÔNG TIN HƯỚNG DÒNG

**Vấn đề:**

- Nén dữ liệu
- Kiểm soát chất lượng đường truyền
- Đồng bộ hóa

# 5. Bài tập

- Bài tập 1: Tìm hiểu cơ chế, chức năng và cài đặt một trong cách dịch vụ truyền thông điệp như RabbitMQ, Kafka, ActiveMQ, Redis Pub/Sub, Google Pub/Sub,.... (Viết thành báo cáo)
- Bài tập 2: Code một hệ thống đơn giản có sử dụng một trong các dịch vụ đó.
- Bài tập 3: Với RPC, ngoài thư viện xmlrpc demo trước. còn các thư viện nào khác. Viết báo cáo tìm hiểu và demo một thư viện có sử dụng định dạng JSON.