CHƯƠNG 3. QUẢN LÝ DỮ LIỆU

Với nhiệm vụ nghiên cứu về mạng truy nhập PLC, mục tiêu hướng tới của đồ án là truyền, nhận trực tiếp dữ liệu qua đường tải điện lực và sản phẩm cuối cùng sẽ là một ứng dụng đa lớp thực hiện nhiều kết nối. Sản phẩm này bao gồm một mạch có thể kết nối với đường điện và phần mềm điều khiển quyết định việc truyền nhận thông tin. Mạch được triển khai tương đương với lới vật lý trong mô hình bảy lớp OSI, phần mềm triển khai trên máy tính giả lập một phần lớp liên kết dữ liệu.

Đối với một ứng dụng đa lớp lại có nhiều kết nối giao tiếp thì việc quản lý dữ liệu là việc vô cùng quan trọng mang ý nghĩa quyết định đến sự thành công hay thất bại của dự án. Việc quản lý dữ liệu hiệu quả sẽ là cơ sở để có thể đảm bảo khả năng nhận dữ liệu đúng hay không cao hơn là tạo tiền đề cho việc tối ưu hóa ứng dụng.

Việc quản lý dữ liệu này bao gồm các nội dung sau

Quản lý về cấu trúc dữ liệu

Quản lý về cấu trúc khung dữ liệu

Quản lý lỗi

3.1.Vai trò của cấu trúc dữ liệu

Trong khoa học máy tính, cấu trúc dữ liệu là cách lưu dữ liệu sao cho nó có thể được sử dụng một cách hiệu quả. Thông thường, một cấu trúc dữ liệu được chọn cẩn thận sẽ cho phép thực hiện thuật toán hiệu quả hơn. Việc chọn cấu trúc dữ liệu thường bắt đầu từ chọn một cấu trúc dữ liệu trừu tượng. Một cấu trúc dữ liệu được thiết kế tốt cho phép thực hiện nhiều phép toán, sử dụng càng ít tài nguyên, thời gian xử lý và không gian bộ nhớ càng tốt. Các cấu trúc dữ liệu được triển khai bằng cách sử dụng các kiểu dữ liệu, các tham chiếu và các phép toán trên đó được cung cấp bởi một ngôn ngữ lập trình.

Mỗi loại cấu trúc dữ liệu phù hợp với một vài loại ứng dụng khác nhau, một số cấu trúc dữ liệu dành cho những công việc đặc biệt. Ví dụ, các B-Tree đặc biệt phù hợp trong việc thiết kế cơ sở dữ liệu.

Trong thiết kế nhiều loại chương trình, việc chọn cấu trúc dữ liệu là vấn đề quan trọng. Kinh nghiệm trong việc xây dựng các hệ thóng lớn cho thấy khó khăn của việc triển khai chương trình, chất lượng và hiệu năng của kết quả cuối cùng phụ thuộc rất nhiều vào việc chọn cấu trúc dữ liệu tốt nhất. Sau khi cấu trúc dữ liệu được chọn, người ta thường dễ nhận thấy thuật toán cần sử dụng. Đôi khi trình tự công việc diễn ra theo thứ tự ngược lại: cấu trúc dữ liệu được chọn do những bài toán quan trọng nhất định có thuật toán chạy tốt nhất với một số cấu trúc dữ liệu cụ thể. Trong cả hai trường hợp, việc lựa chọn cấu trúc dữ liệu là rất quan trọng.

Tri thức đó đã dẫn đến sự nổi lên của nhiều ngôn ngữ lập trình và phương pháp thiết kế được hình thức hóa, mà trong đó, nhân tố tổ chức quan trọng là các cấu trúc dữ liệu chứ không phải các thuật toán. Đa số ngôn ngữ có một tính năng thuộc dạng hệ thống module cho phép các cấu trúc dữ liệu được tái sử dụng an toàn trong các ứng dụng khác nhau, bằng cách dùng các giao diện có điều khiển để che các chi tiết cài đặt đã được kiểm thử.

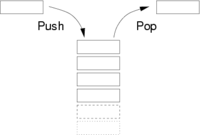
Vì cấu trúc dữ liệu có tính chất quyết định đối với các chương trình chuyên nghiệp nên có rất nhiều hỗ trợ về cấu trúc dữ liệu trong các thư viện chuẩn của các ngôn ngữ lập trình hiện đại, ví dụ thư viện mẫu chuẩn của C, C++, Java API, và Microsoft .NET Framework.

Các cấu trúc xây dựng căn bản của hầu hết các cấu trúc dữ liệu là mảng (array), bản ghi (record), tổ hợp phân biệt (discriminated union), và tham chiếu (reference). Ví dụ tham chiếu khả rỗng (có thể có giá trị null) là một kết hợp của tham chiếu và cấu trúc discriminated union, và cấu trúc dữ liệu liên kết đơn giản nhất, danh sách liên kết, được xây dựng từ các bản ghi và các tham chiếu khả rỗng.

3.2.Lựa chọn cấu trúc dữ liệu cho hệ thống

Các cấu trúc dữ liệu thường dùng

3.2.1.Ngăn xếp

[](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Stack-sv.png)

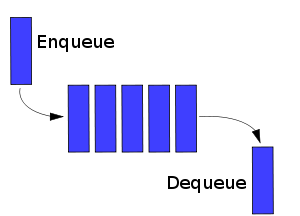
Hình 3.1. Cấu trúc dữ liệu kiểu ngăn xếp.

Trong khoa học máy tính, một ngăn xếp (còn gọi là bộ xếp chồng, tiếng Anh: stack) là một cấu trúc dữ liệu trừu tượng hoạt động theo nguyên lý "vào sau ra trước" (Last In First Out (LIFO).

Một ngăn xếp là một cấu trúc dữ liệu dạng thùng chứa (container) của các phần tử (thường gọi là các nút (node)) và có hai phép toán cơ bản : push and pop. Push bổ sung một phần tử vào đỉnh (top) của ngăn xếp,nghĩa là sau các phần tử đã có trong ngăn xếp. Pop giải phóng và trả về phần tử đang đứng ở đỉnh của ngăn xếp. Trong stack, các đối tượng có thể được thêm vào stack bất kỳ lúc nào nhưng chỉ có đối tượng thêm vào sau cùng mới được phép lấy ra khỏi stack.

Trong ngôn ngữ máy tính hiện nay, một ngăn xếp thường được dùng với các phép toán "push" và "pop". Độ dài (số phần tử) của ngăn xếp cũng là một tham số cần thiết của nó. Đôi khi cần tới một phép toán chỉ cần lấy giá trị của phần từ trên đỉnh ngăn xếp mà không xóa nó khỏi ngăn xếp (peak).

3.2.2.Hàng đợi



Hình 3.2. Cấu trúc dữ liệu kiểu hàng đợi.

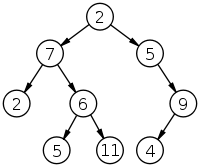
Hàng đợi (tiếng Anh: queue) là một cấu trúc dữ liệu dùng để chứa các đối tượng làm việc theo cơ chế FIFO (viết tắc từ tiếng Anh: First In First Out), nghĩa là "vào trước ra trước.

Trong hàng đợi, các đối tượng có thể được thêm vào hàng đợi bất kỳ lúc nào, nhưng chỉ có đối tượng thêm vào đầu tiên mới được phép lấy ra khỏi hàng đợi. Thao tác thêm vào và lấy một đối tượng ra khỏi hàng đợi được gọi lần lượt là "enqueue" và "dequeue". Việc thêm một đối tượng luôn diễn ra ở cuối hàng đợi và một phần tử luôn được lấy ra từ đầu hàng đợi.

Trong tin học, cấu trúc dữ liệu hàng đợi có nhiều ứng dụng: khử đệ qui, tổ chức lưu vết các quá trình tìm kiếm theo chiều rộng và quay lui, vét cạn, tổ chức quản lý và phân phối tiến trình trong các hệ điều hành, tổ chức bộ đệm bàn phím.

Các thao tác thêm, trích và huỷ một phần tử phải được thực hiện ở hai phía khác nhau của hàng đợi, do đó hoạt động của hàng đợi được thực hiện theo nguyên tắc FIFO. Cũng như ngăn xếp, cấu trúc mảng một chiều hoặc cấu trúc danh sách liên kết có thể dùng để biểu diễn cấu trúc hàng đợi.

3.2.3.Dạng cây

[](http://vi.wikipedia.org/wiki/T%E1%BA%ADp_tin:Binary_tree.svg)

Hình 3.3. Cấu trúc dữ liệu dạng cây

Trong khoa học máy tính, cây là một cấu trúc dữ liệu được sử dụng rộng rãi gồm một tập hợp các nút (tiếng Anh: node) được liên kết với nhau theo quan hệ cha – con. Cây trong cấu trúc dữ liệu đầu tiên là mô phỏng (hay nói cách khác là sự sao chép) của cây (có gốc) trong lý thuyết đồ thị. Hầu như mọi khái niệm trong cây của lý thuyết đồ thị đều được thể hiện trong cấu trúc dữ liệu. Tuy nhiên cây trong cấu trúc dữ liệu đã tìm được ứng dụng phong phú và hiệu quả trong nhiều giải thuật. Khi phân tích các giải thuật trên cấu trúc dữ liệu cây, người ta vẫn thường vẽ ra các cây tương ứng trong lý thuyết đồ thị.

Một nút có thể chứa một giá trị, một điều kiện, một cấu trúc dữ liệu riêng biệt hoặc chính một cây. Mỗi nút trong một cây có thể không có hoặc có một số nút con, các nút con có mức cao hơn nó (theo quy ước khác với cây tự nhiên, cây trong cấu trúc dữ liệu phát triển từ trên xuống). Một nút có con được gọi là nút cha của các nút con. Một nút có nhiều nhất một nút cha.

Cấu trúc dạng cây là một cấu trúc tối ưu, với cấu trúc dạng cây ta có thể dễ dàng tổ chức dữ liệu với nhưng mức ưu tiên khác nhau, cấu trúc khác nhau mà vẫn đảm bảo khả năng quản lý và tìm kiếm hiệu quả.

Tuy nhiên với mục đích của đồ án là xây dựng lớp vật lý, lớp vật lý là lớp đảm bảo việc truyền, nhận dữ liệu trực tiếp đảm bảo dữ liệu nhận được có ít sai khác nhất so với dữ liệu gốc, nên việc lựa chọn cấu trúc dữ liệu hình cây là không tối ưu.

Cấu trúc dữ liệu dạng hàng đợi và ngăn xếp là phù hợp hơn cả vì chúng dễ triển khai, tốc độ xử lý nhanh. Trong đồ án chúng tôi chọn sử dụng cấu trúc dữ liệu kiểu hàng đợi (FIFO) để triển khai. Chi tiết về nội dung này sẽ được trình bày cụ thể ở những nội dung sau.

3.3.Vai trò khung dữ liệu

Dữ liệu trao đổi giữa các lớp và các module đều ở dưới dạng các bit và byte, chỉ xây dựng cấu trúc dữ liệu là chưa đủ để quản lý các luồng dữ liệu. Một khối dữ liệu mang ý nghĩa cần phải được hệ thống phân tích và nhận dạng được. Chính vì lý do đó, dữ liệu trao đổi dữ các lớp và các module cần được tạo thành các khung, các khung dữ liệu này giúp hệ thống xác định được đó là dữ liệu kiểu gì, độ dài là bao nhiêu cũng như biết khi nào bắt đầu nhận hay bắt đầu xử lý dữ liệu.

Khung dữ liệu được định nghĩa theo các nhiệm vụ khác nhau của hệ thống, nhưng có thể chia ra làm hai nhóm chính sau:

Thứ nhất, là các khung dữ liệu dùng cho việc trao đổi dữ liệu giữa các lớp và các module của một phần tử. Trong đề tài này, khai niệm phần tử được hiểu là toàn bộ một mạch giao tiếp cũng như máy tính điều khiển nó, mạch giao tiếp và máy tính cùng mạch giao tiếp khác là một phần tử khác.

Thứ hai, là các khung dữ liệu dùng trong các tác vụ trao đổi dữ liệu giữa các phần tử khác nhau.

3.4.Khung dữ liệu giao tiếp giữa các lớp và module

Cấu trúc chung



Cấu trúc khung dữ liệu trao đổi giữa các lớp và các module

Bảng mô tả khung dữ liệu trao đổi giữa hai module

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Độ lớn | Mô tả |
| 1 | HEADER | 1 byte | Báo hiệu bắt đầu một khung (0xAA) |
| 2 | CODE | 1 byte | Lệnh kèm theo khung, lệnh này giúp hệ thống xác định xem dữ liệu là kiểu gì |
| 3 | LENGTH | 1 byte | Độ dài dữ liệu kèm theo, giá trị lớn nhất của một khung này là 255 |
| 4 | DATA | N bytes | Dữ liệu kèm theo có độ dài như đã định nghĩa từ byte mang giá trị LENGTH phía trên. Dữ liệu này không phân biệt kiểu hay độ dài. |

Hệ thống sẽ đọc nếu như trong hàng đợi đang có dữ liệu, khi đọc được giá trị báo hiệu HEADER hệ thống sẽ biết là bắt đầu một khung mới, tiếp đến việc đọc giá trị CODE sẽ giúp hệ thống biết sẽ phải xử lý tiếp theo, giá trị LENGTH quyết định hệ thống sẽ ngừng đọc khi nào để xử lý dữ liệu.

Việc sử dụng khung dữ liệu như trên còn tạo tính mở trong việc mở rộng thêm các tính năng mới cho hệ thống. Bên cạnh đó cũng cho phép xây dựng các khung có độ dài bất kỳ mà không bị ràng buộc bởi những giá trị mặc định từ trước.

Bảng mô tả các lệnh sử dụng trong giao tiếp

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Mã lệnh | Giá trị | Vai trò | Ghi chú |
| 1 | COM\_GET\_CRT | 0x00 | Đọc thanh ghi ST7538/7540 |  |
| 2 | COM\_SET\_CRT | 0x01 | Ghi thanh ghi ST7538/7540 |  |
| 3 | COM\_GET\_PLM | 0x02 | Nhận dữ liệu |  |
| 4 | COM\_SET\_PLM | 0x03 | Truyền dữ liệu |  |
| 5 | COM\_GET\_MAX LENGTH | 0x04 | Lấy độ dài khung giao tiếp qua tải | Không còn được sử dụng, trong phiên bản mới dữ liệu có thể được truyền với độ dài bất kỳ |
| 6 | COM\_SET\_MAX LENGTH | 0x05 | Cấu hình độ dài khung giao tiếp qua tải |
| 7 | COM\_BROAD CAST | 0xFF | Bản tin broad cast IP |  |
| 8 | COM\_HEADER | 0xAA | Báo header | Được sử dụng để tránh lỗi trong các trường hợp mạch reset |

Bảng mô tả trên được triển khai ở tất cả các lớp cũng như các module tạo sự thống nhất trong thiết kế. Dưới đây sẽ là các khung dữ liệu cụ thể dùng trong hệ thống.

3.4.1.Khung dữ liệu đọc thanh ghi ST7538/7540

Yêu cầu đọc



Hình . Cấu trúc khung dữ liệu yêu cầu đọc thanh ghi

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Trường thông tin | Giá trị | Mô tả |
| 1 | HEADER | 0xAA |  |
| 2 | CODE | 0x00 | COM\_GET\_CRT |
| 3 | LENGTH | 0x00 | Không có dữ liệu đi kèm |

Giá trị nhận lại

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Trường thông tin | Giá trị | Mô tả |
| 1 | HEADER | 0xAA |  |
| 2 | CODE | 0x00 | COM\_GET\_CRT |
| 3 | LENGTH | 0x04 | Nhận lại 4 byte dữ liệu là giá trị của thanh ghi ST75xx |
| 4 | DATA |  | 4 byte dữ liệu |

3.4.2.Khung dữ liệu ghi thanh ghi ST7538/7540

Yêu cầu ghi

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Trường thông tin | Giá trị | Mô tả |
| 1 | HEADER | 0xAA |  |
| 2 | CODE | 0x01 | COM\_SET\_CRT |
| 3 | LENGTH | 0x04 | Dữ liệu gồm 4 byte trong đó có 3 byte đầu được dùng để cấu hình thanh ghi ST75xx |
| 4 | DATA |  | 4 byte dữ liệu |

Giá trị nhận lại

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Trường thông tin | Giá trị | Mô tả |
| 1 | HEADER | 0xAA |  |
| 2 | CODE | 0x01 | COM\_SET\_CRT |
| 3 | LENGTH | 0x01 | 1 byte thông tin báo quá trình ghi thanh ghi có thành công hay không |
| 4 | DATA |  | DATA = 0x00 ghi thành công  DATA = 0x01 ghi không thành công |

3.4.3.Khung dữ liệu gửi thông tin

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Trường thông tin | Giá trị | Mô tả |
| 1 | HEADER | 0xAA |  |
| 2 | CODE | 0x03 | COM\_SET\_PLM |
| 3 | LENGTH |  | Giá trị bằng độ dài dữ liệu cần gửi (giá trị này tối đa là 255) |
| 4 | DATA |  | Đoạn dữ liệu cần gửi |

3.4.4.Khung dữ liệu nhận thông tin

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Trường thông tin | Giá trị | Mô tả |
| 1 | HEADER | 0xAA |  |
| 2 | CODE | 0x02 | COM\_GET\_PLM |
| 3 | LENGTH |  | Độ dài khung dữ liệu nhận được |
| 4 | DATA |  | Đoạn dữ liệu nhận được |

3.5.Khung dữ liệu giao tiếp giữa các phần tử mạng

Cấu trúc khung dữ liệu đảm bảo việc gửi đúng địa chỉ các bản tin



|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Độ lớn | Mô tả |
| 1 | Domain Address | 1 byte | Địa chỉ domain của phần tử (các phần tử có thể nằm ở các domain khác nhau) |
| 2 | Transmitter Address | 1 byte | Địa chỉ phần tử gửi dữ liệu |
| 3 | Receiver Address | 1 byte | Địa chỉ phần tử nhận dữ liệu |
| 4 | Control Byte | 1 byte | Byte điều khiển, byte này quyết định mức ưu tiên của bản tin hay việc có gửi ACK phản hồi không, v…v |
| 5 | Total Messages | 1 byte | Tổng số bản tin cần gửi trong một phiên (khi đoạn dữ liệu cần gửi quá dài sẽ được chia nhỏ để gửi làm nhiều lần) |
| 6 | Current Message | 1 byte | Số thứ tự bản tin hiện tại |
| 7 | Repitition | 1 byte | Số lần gửi lại của bản tin đó (trong trường hợp sử dụng ACK) |
| 8 | Data | N byte | Dữ liệu dùng trong trao đổi |

Do đồ án tập trung xử lý các vấn đề của lớp vật lý, trong những phần trước có nhấn mạnh lớp vật lý có khả năng truyền nhận dữ liệu có độ dài bất kỳ nên cấu trúc khung nêu trên có thể dễ dàng thay đổi. Thực tế đồ án này cũng chưa sử dụng hết tất cả các trường như bản mô tả trên. Các trường total messages, current messages, repitition được gán các giá trị mặc định.

Một số khung dữ liệu khác

Bản tin ACK



|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Độ lớn | Mô tả |
| 1 | Domain Address | 1 byte | Địa chỉ domain của phần tử (các phần tử có thể nằm ở các domain khác nhau) |
| 2 | Transmitter Address | 1 byte | Địa chỉ phần tử gửi dữ liệu |
| 3 | Receiver Address | 1 byte | Địa chỉ phần tử nhận dữ liệu |
| 4 | Control Byte | 1 byte | Byte điều khiển, byte này quyết định mức ưu tiên của bản tin hay việc có gửi ACK phản hồi không, v…v |
| 5 | Total Messages | 1 byte | Tổng số bản tin cần gửi trong một phiên (khi đoạn dữ liệu cần gửi quá dài sẽ được chia nhỏ để gửi làm nhiều lần) |
| 6 | Current Message | 1 byte | Số thứ tự bản tin hiện tại |
| 7 | Repitition | 1 byte | Số lần gửi lại của bản tin đó (trong trường hợp sử dụng ACK) |

Bản tin quảng bá



|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên | Độ lớn | Mô tả |
| 1 | Domain Address | 1 byte | Địa chỉ domain của phần tử (các phần tử có thể nằm ở các domain khác nhau) |
| 2 | Transmitter Address | 1 byte | Địa chỉ phần tử gửi dữ liệu |
| 3 | Receiver Address | 1 byte | Địa chỉ phần tử nhận dữ liệu |
| 4 | Control Byte | 1 byte | Byte điều khiển, byte này quyết định mức ưu tiên của bản tin hay việc có gửi ACK phản hồi không, v…v |
| 5 | Total Messages | 1 byte | Tổng số bản tin cần gửi trong một phiên (khi đoạn dữ liệu cần gửi quá dài sẽ được chia nhỏ để gửi làm nhiều lần) |
| 6 | Current Message | 1 byte | Số thứ tự bản tin hiện tại |
| 7 | Repitition | 1 byte | Số lần gửi lại của bản tin đó (trong trường hợp sử dụng ACK) |

Quản lý lỗi

Một trong những nội dung quan trọng trong quản lý dữ liệu đó là quản lý lỗi. Đường tải điện như những phân tích trong nhwung nội dung trước không được thiết kế phục vụ truyền dữ liệu nên nhiễu trên đường truyền gây lỗi rất lớn. Chính vì vậy việc sử dụng các cơ chế phát hiện lỗi và sửa lỗi là cần thiết. Trong đồ án này chúng tôi triển khai hai biện pháp phát hiện lỗi và sửa lỗi trên hai module hệ thống.

Cơ chế phát hiện lỗi trên phần mềm máy tính

Cơ chế sửa lỗi tích hợp trên vi xử lý

Cả hai cơ chế này đều được triển khai bằng mã vòng

Phát hiện lỗi

Để phát hiện lỗi ta sử dụng một chuỗi kiểm tra khung (FCS) đưa thêm các byte checksum vào cuối khung. FCS được thêm vào trước khi chuyển khung dữ liệu từ lớp điều khiển xuống lớp vật lý và được tách ra khi máy thu chuyển từ lớp vật lý lên lớp điều khiển. Hệ thống sẽ kiểm tra các byte FCS này để xác định xem dữ liệu nhận được có đúng hay không.

Đa thức sinh FCS là x16+x15+x2+1

Đa thức trên sẽ tạo thêm hai byte FCS ghép thêm vào cuối khung dữ liệu





Với đa chuối kiểm tra khung này ta có thể phát hiện lỗi của khung có độ dài bất kỳ tuy nhiên không cho ta khả năng sửa các lỗi này.

Sửa lỗi

Trong đồ án chúng tôi sử dụng mã tự sửa lỗi (FEC) để sửa các lỗi xuất hiện trên đường truyền. Để tối ưu tốc độ tính toán thì giải pháp chúng tôi đưa ra là sửa lỗi cho từng byte một. Việc triển khai sửa lỗi cho từng byte một hay một cụm byte là hoàn toàn tương đương.

Đa thức sinh cho mã FEC x6+x5+x4+x3+1

