* **[Lớp học](http://laptrinhtanbinh.com/course)**
* [**Hướng dẫn sử dụng**](http://laptrinhtanbinh.com/how-to-use)

Top of Form

Bottom of Form

* Photo

3.14. Giao tiếp qua mạng

Haizz, hệ thống mạng !

Với phần lớn các lính mới, có thể khiến cho chương trình mình viết ra có thể giao tiếp với các chương trình khác qua mạng, dù là mạng nội bộ (giữa các máy tính cá nhân nằm cùng 1 nơi) hay Internet, luôn luôn là 1 khát khao cháy bỏng thôi thúc.

Tuy nhiên, đây lại là 1 vấn đề rất phức tạp bởi lẽ để làm được điều này không những chỉ đòi hỏi lập trình viên có các kiến thức về lập trình C++ mà còn yêu cầu có khá nhiều kiến thức về cách vận hành của mạng lưới máy tính. Cái gì mà các lớp trừu tượng, TCP/IP, UPD, socket, vv… Có thể có các bạn đã từng nghe nói đến chúng nhưng ai dám tự tin là có thể định nghĩa rõ ràng những khái niệm này?

*! Ban đầu, tôi hoàn toàn không định thêm bài học này vào giáo trình. Lý do thứ nhất là những kiến thức này không hẳn có liên quan tới GUI, vốn là chủ để chính của chương này. Thêm vào đó, như đã nói, đây là phần kiến thức khá phức tạp, xứng đáng dành cả chương chỉ để nói về nó.*

Thế nhưng do có những yêu cầu từ phía các bạn muốn được hiểu các khái niệm cơ bản về mạng lưới, vậy nên chúng ta sẽ có 1 bài học ngoại lệ trong chương này. Vậy nên, trong bài học này, chúng ta sẽ không nói đến GUI mà đề cập tới mạng lưới.

Cần thông báo trước với các bạn là, như đã nói ở trên, kiến thức này vừa khó vừa dài nên tôi cũng không có đủ thời gian cũng như điều kiện để đề cập đến toàn bộ các khía cạnh và bản thân tôi cũng không hiểu hết về mọi thứ. Vậy nên, tôi đã tìm ra 1 giải pháp thỏa hiệp, đó là thực hiện 1 bài học dưới dạng bài học thực hành, nghĩa là vừa có lý thuyết lại có thực hành.

Trong bài học, chúng ta sẽ không đề cập tới tất cả mọi thứ mà tập trung vào 1 kiểu kiến trúc cơ bản nhất (kiến trúc người dùng - máy chủ hay client/server). Nó sẽ giúp cho các bạn có được những nền tảng cơ bản nhất để hiểu cơ chế hoạt động của mạng lưới. Sau đó sẽ cần chính các bạn tự nỗ lực để áp dụng vào các trường hợp thực tế của bản thân.

***? Vậy chủ đề của bài thực hành lần này là gì ?***

Mục tiêu mà chúng ta muốn thực hiện trong bài này là 1 chương trình dùng để tán gẫu (chat) qua mạng. Chương trình có thể được sử dụng giữa các máy được nối dây với nhau cũng như là các máy liên kết thông qua Internet.

Bắt đầu thôi chứ hả ?

Chúng ta sẽ mở đầu bằng 1 ít kiến thức lý thuyết, rất quan trọng để có thể hiểu được phần tiếp sau của bài học này.

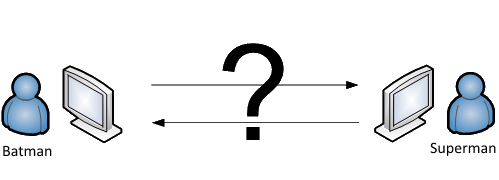
**Làm thế nào để giao tiếp qua mạng lưới ?**

1 câu hỏi rất hay !

Câu trả lời cho câu hỏi này có thể dài như nội dung của chục quyển bách khoa toàn thư cộng lại với nhau. Thậm chí là kể cả thế thì cũng chưa chắc có thể trả lời được hoàn chỉnh về mọi khía cạnh.

Vậy nên ở đây chúng ta sẽ chỉ thảo luận qua về những khái niệm lý thuyết thuộc loại cơ bản nhất. Từ đó chúng ta sẽ xem làm thế nào có thể áp dụng chúng vào thực tế qua việc tạo ra chương trình tán gẫu qua mạng nhờ sự giúp đỡ của Qt.

Để dễ hình dung, hãy tưởng tượng chúng ta có 2 người dung là… Batman và Superman. Mỗi người bọn họ sẽ có 1 chiếc máy tính riêng và họ muốn trao đổi với nhau qua mạng.



Làm sao để họ có thể liên lạc với nhau ? Làm sao để xác định đúng máy tính của người muốn liên lạc biết rằng trong thế giới rộng lớn ngoài kia vẫn còn hàng trăm triệu máy tính khác ngoài họ cũng đang cùng kết nối vào mạng.

Rồi thì làm sao họ có thể hiểu nhau ? Có cần phải nói chung 1 ngôn ngữ không ?

Câu trả lời là, để 2 chương trình có thể trao đổi với nhau qua mạng, chúng ta cần có 3 điều kiện :

1. Biết địa chỉ IP của máy tính đối phương.
2. Có 1 **cổng mạng** (port) mở và chưa được sử dụng.
3. Sử dụng cùng 1 giao thức truyền dữ liệu.

Nếu hội tụ được tất cả các điều kiện trên, vậy thì chúng ta đã sẵn sàng để giúp 2 chương trình giao tiếp với nhau qua mạng.

Trước hết, chúng ta hãy xem làm sao để chuẩn bị có được những thứ đó.

**1/ Địa chỉ IP, định danh của máy tính trong mạng**

Điều đầu tiên chúng ta cần quan tâm, đó là hiểu cách mà các máy tính nhận ra nhau trong cả mạng lưới.

Làm sao để tin nhắn của Batman chỉ được gửi đến Superman mà không phải là người khác.

**Địa chỉ IP là gì ?**

Cần biết rằng mỗi máy tính trong mạng đều được định danh bằng thứ mà chúng ta gọi là địa chỉ IP. Đó là 1 chuỗi số như 192.168.51.255

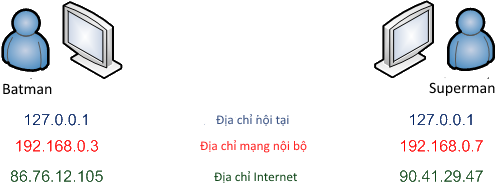
Địa chỉ này đại diện cho 1 máy tính trong mạng. Khi các bạn biết được địa chỉ IP của người muốn liên lạc, vậy thì ít nhất chúng ta có thể biết được chúng ta cần gửi tin nhắn đi đâu.

Thế nhưng, mọi chuyện sẽ không thể đơn giản như thế. Vấn đề ở đây là mỗi máy tính có thể có không chỉ 1 mà đồng thời nhiều địa chỉ IP khác nhau.

Về cơ bản, chúng ta có thể xác định được là mỗi máy tính có trung bình 3 địa chỉ IP :

* Địa chỉ IP nội tại : còn được gọi là địa chỉ *localhost* hay *loopback*. Địa chỉ này được sử dụng bởi bản thân mỗi chiếc máy tính để chỉ chính bản thân mình. Nó không quá hữu dụng do chúng ta không có dính dáng gì đến mạng lưới cả. Thế nhưng nó lại khá có ích khi chúng ta cần dùng để chạy thử chương trình. Ví dụ 127.0.0.1
* Địa chỉ IP mạng nội bộ : Nếu các bạn có đồng thời nhiều máy tính cùng kết nối vào mạng ở nhà riêng thì nhờ địa chỉ IP này, chúng có thể liên lạc với nhau mà không cần thông qua mạng Internet. Ví dụ 192.168.0.51
* Địa chỉ IP Internet : Đây là địa chỉ dùng để liên lạc với tất cả các máy tính khác trên mạng Internet khắp thế giới. Ví dụ 82.39.128.57

Vậy là Batman và Superman mỗi người sẽ có nhiều IP khác nhau tùy loại.



Tại sao chúng ta lại cần biết tất cả những điều này ? Đó là bởi vì tùy thuộc vào khoảng cách giữa 2 người dùng mà chúng ta sẽ cần phải sử dụng các loại địa chỉ IP khác nhau.

Nếu Batman và Superman cùng sống trong chung cư thì chúng ta sẽ chỉ cần dùng địa chỉ mạng nội bộ là đủ.

Thế nhưng nếu họ kết nối với nhau thông qua Internet thì chúng ta sẽ cần phải sử dụng đến địa chỉ Internet.

Về phần địa chỉ nội tại thì nó chỉ có chức năng là mô phỏng hoạt động của mạng. Nếu Batman gửi 1 tin nhắn đến địa chỉ 127.0.0.1 thì anh ta sẽ tự nhận lại được nó. Nó rất hữu ích khi chúng ta không cần làm phiền Superman cứ 5 phút 1 lần để thử nghiệm chương trình chúng ta vừa viết. Dù sao thì anh ta cũng rất bận cứu thế giới.

**Xác định địa chỉ IP của máy tính**

***? Làm sao tôi tìm được những địa chỉ IP của máy mình ? Thế rồi làm sao tôi biết được trong số chúng cái nào là dành cho mạng nội bộ còn cái nào dùng cho Internet ?***

Phương thức tìm kiếm tùy thuộc vào loại IP mà chúng ta muốn tìm.

* IP nội tại : đừng mất công tìm kiếm nhiều làm gì, chắc chắn đấy là 127.0.0.1 hoặc cụm từ « localhost »
* IP mạng nội bộ : cái này tùy thuộc vào hệ điều hành mà chúng ta sử dụng :
  + Trong Windows , hãy mở cửa sổ dòng lệnh (nơi chúng ta vẫn hay dùng để gõ các dòng lệnh Qt) và gõ câu lệnh dưới đây :

1. ipconfig

Có khả năng là có nhiều kết quả được trả về tùy thuộc vào mạng nội bộ của bạn (cắm dây, wifi, vv…) nhưng chắc chắn là 1 trong số đó là cái chúng ta đang tìm.

* Trong Linux hoặc Mac OS, tương tự nhưng với câu lệnh hơi khác.

1. ifconfig

Địa chỉ này thường có dạng 192.168.XXX.XXX nhưng đôi khi cũng có chút khác biệt.

* IP trên Internet : chắc đơn giản nhất là đi tra cứu trên các trang web cho phép cung cấp thông tin về IP như [whatismyip.com](http://www.whatismyip.com/)!

Bây giờ, sau khi đã xác định được địa chỉ IP của người đối thoại, chúng ta có thể gần như liên lạc với người đó rồi. Chỉ còn thiếu thông tin về cổng liên hệ với máy tính đối phương là được. Vậy nên thuật ngữ chúng ta sẽ tiếp xúc tiếp theo là **cổng**.

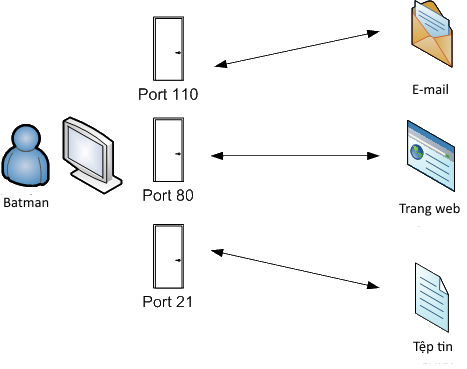
**2/ Các cổng : các cách khác nhau để truy cập vào cùng 1 máy tính**

1 máy tính kết nối vào mạng, ở mỗi thời điểm đều nhận đồng thời vô số tin nhắn khác nhau. Hoàn toàn bình thường khi chúng ta vừa có thể lướt web vừa đồng thời có thể tra cứu thư điện tử phải không ?

Vậy, để quản lý tất cả mớ hỗn độn đó và không nhập nhằng chúng với nhau, người ta đã tạo ra khái niệm cổng.

1 cổng là 1 số nguyên nằm giữa 0 và 65535. Dưới đây là 1 số các cổng nổi tiếng :

* 21 : sử dụng bởi các chương trình FTP (File Transfer Protocol) chuyên gửi nhận tệp.
* 80 : cổng xử lý các yêu của trang web, hay được dùng bởi Firefox hay BanDoX
* 110 : dùng để nhận thư



Nếu chúng ta muốn chương trình viết ra có thể liên lạc được với Superman, vậy thì cần tìm ra 1 cổng không được sử dụng bởi các chương trình khác trên máy tính.

Cần biết là phần lớn các cổng có số nhỏ hơn 1024 đều đã được đặt sẵn trước bởi máy tính. Thế nên chúng ta cần viết chương trình ưu tiên sử dụng các cổng từ 1024 đến 65535.

*! Để tránh việc bất cứ chương trình nào trong mạng cũng có thể liên lạc và truy cập đến các máy tính mà không cần sự cho phép, người ta đã nghĩ ra khái niệm****tường lửa****(firewall). Nhiệm vụ của tường lửa là chặn hầu hết tất các các cổng và chỉ mở 1 số cổng được đánh giá là an toàn. Vậy nên trước khi lập trình thì chúng ta cần xem xét kỹ tùy chỉnh của tường lửa có thể đã được thiết lập trên máy của chúng ta từ trước vì nó có thể chặn cả các kết nối đến máy của chúng ta.*

**3/ Giao thức : trao đổi dữ liệu trong cùng « ngôn ngữ »**

Thế là trong tay chúng ta đã có 2 thứ : địa chỉ IP và các cổng. Địa chỉ IP thì cần tìm ra còn cổng sử thì cần chọn từ các cổng trống có sẵn (trong phần tiếp theo chúng ta sẽ thảo luận cách để chọn ra cổng trống).

Với 2 thứ này, chúng ta đã có thể tạo ra 1 kết nối từ xa tới 1 chiếc máy tính khác trên mạng.

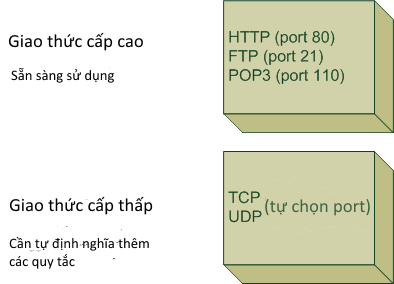
Bây giờ, chúng ta chỉ còn phải giải quyết vấn đề cuối cùng trước khi khiến 2 máy tính có thể trao đổi với nhau, đó là yêu cầu chúng sử dụng cùng 1 loại « ngôn ngữ », thuật ngữ chuyên môn là sử dụng cùng 1 giao thức.

*! Định nghĩa : giao thức là 1 tập hợp các quy tắc cho phép 2 máy tính trao đổi với nhau. 2 máy tính bắt buộc phải sử dụng cùng 1 giao thức thì mới có thể trao đổi được dữ liệu.*

**Các tầng giao thức liên lạc khác nhau**

Có tồn tại hàng trăm giao thức liên lạc khác nhau. Chúng có thể rất phức tạp nhưng cũng có khi rất đơn giản, tùy theo cấp độ trao đổi « cao » hay « thấp ». Chúng ta có thể chia chúng thành 2 loại lớn :

* Giao thức cấp độ cao : ví dụ như giao thức FTP sử dụng cổng 21 để gửi và nhận tệp tin là 1 hệ thống trao đổi dữ liệu ở cấp độ cao. Cách thức hoạt động của nó đã được miêu tả rất rõ ràng. Vậy nên giao thức này khá dễ sử dụng nhưng trái lại, chúng ta khó có thể thêm vào các chỉnh sửa mới.
* Giao thức cấp độ thấp : ví dụ như giao thức TCP. Nó được sử dụng bởi các chương trình mà không giao thức cấp độ cao nào thích hợp với chúng. Khi sử dụng giao thức này, chúng ta có thể sẽ phải thao tác với từng đơn vị dữ liệu nhỏ nhất. Vậy nên đương nhiên là nó khó sử dụng hơn FTP nhưng chúng ta có thể làm tất cả những gì chúng ta muốn với nó.



*! Dành cho những bạn nào có tính tò mò cao thì có thể tham khảo kiến thức về*[*mô hình OSI*](https://vi.wikipedia.org/wiki/M%C3%B4_h%C3%ACnh_OSI)*. Đây là mô hình tổ chức dữ liệu trong mạng lưới trong đó có định nghĩa các cấp độ trao đổi khác nhau (thường được gọi là các lớp). Trong hình bên trên, mô hình đã được đơn giản đi nhiều, nếu không chúng ta sẽ không thể hoàn thành tất cả chỉ trong 1 bài học.*

Các giao thức cấp cao sử dụng các cổng được đặt sẵn từ trước.

Các giao thức cấp thấp thì có thể mượn bất kỳ cổng nào nên linh hoạt hơn nhiều nhưng vấn đề là chúng ta sẽ phải tự định nghĩa cách thức nó sẽ hoạt động.

*! Thực tế thì tất cả các giao thức cấp cao đều sử dụng giao thức cấp thấp hơn để hoạt động. Các giao thức cấp thấp là nên tảng cơ bản để xây dựng giao thức cấp cao.*

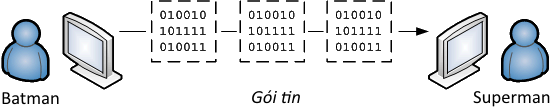
Chúng ta sẽ không tạo ra 1 chương trình quản lý thư điện tử hay trao đổi tệp dữ liệu mà sẽ định nghĩa phương thức trao đổi riêng cho chương trình của chúng ta. Giao thức đó hiển nhiên sẽ dựa trên các giao thức cấp thấp. Vậy nên chúng ta sẽ làm việc với giao thức cấp thấp.

Tin xấu là chúng rất khó sử dụng !

Tin tốt là về mặt kỹ thuật thì chúng sẽ mang lại nhiều điều hấp dẫn hơn.

**Giao thức cấp thấp TCP và UDP**

Dữ liệu trong mạng thường không thể truyền tải 1 lúc quá nhiều mà bắt buộc phải chia ra thành những thành phần nhỏ hơn mà chúng ta gọi là **gói**. Mỗi gói lại có thể được chia thành các gói nhỏ hơn.



Lấy ví dụ Batman muốn gửi Superman tin nhắn với nội dung : Chào Superman, dạo này khỏe chứ ?. Tin nhắn này sẽ không được chuyển trong 1 lần mà được chia ra thành nhiều mảnh nhỏ. Hãy giả dụ là nó được chia thành 4 mảnh.

1. Gói 1 : Chào Superm
2. Gói 2 : an, dạo này k
3. Gói 3 : hỏe c
4. Gói 4 : hứ ?

*! Chúng ta không phải là người sẽ quyết định việc dữ liệu được chia nhỏ như thế nào mà là do giao thức cấp thấp phụ trách. Vậy nên chúng ta không thể biết trước được số lượng cũng như kích thước của từng gói nhỏ. Tuy nhiên biết cách thức hoạt động của chúng khá quan trọng để có thể hiểu được phần tiếp theo.*

Các gói nhỏ có thể được gửi đi theo các cách khác nhau tùy theo giao thức cấp thấp mà chúng ta sử dụng.

* Giao thức TCP : giao thức thuộc loại cổ điển. Nó cần phải thiết lập 1 đường truyền giữa 2 máy trước khi dữ liệu được chuyển đi. Hệ thống quản lý gói tin được thiết lập để phòng trường hợp 1 trong số các gói tin bị thất lạc và hệ thống sẽ gửi lại 1 gói khác thay thế. Vậy nên, với TCP chúng ta có thể chắc chắn là mọi gói tin đều được gửi tới nơi và theo đúng thứ tự. Trái lại, hệ quả là của việc này là khiến cho dữ liệu được truyền đi chậm hơn so với sử dụng UDP.
* Giao thức UDP : giao thức này không cần thiết lập trước đường truyền và nó truyền dữ liệu đi nhanh hơn. Trái lại, do các gói tin không được quản lý, việc các gói tin bị thất lạc có thể xảy ra và chúng ta không thể xác định được. 1 khả năng xấu khác là các gói tin có thể đều đến nơi nhưng theo thứ tự lộn xộn.

Chúng ta sẽ phải chọn giữa 2 giao thức này.

Tôi thì đã đưa ra lựa chọn từ trước rồi, đó là sử dụng giao thức TCP. Trong thực tế, chúng ta đang viết 1 chương trình tán gẫu và không thể để xảy ra việc tin nhắn hoặc 1 phần của tin nhắn bị thiếu hụt, nếu không cuộc hội thoại sẽ trở nên khó hiểu. Ai lại hiểu được tin nhắn kiểu này chứ : Chào Superm hỏe chứ ?.

***? Nói vậy thì ai chả chọn dùng TCP để chắc chắn các gói thông tin được chuyển tới nơi. Ai lại dám mạo hiểm sử dụng UDP chứ ?***

1 vài chương trình phức tạp sử dụng rất nhiều việc trao đổi thông tin trên mạng có thể chọn sử dụng UDP, lấy ví dụ như các trò chơi chẳng hạn.

Hãy lấy ví dụ các trò chơi chiến thuật như Starcraft hay bắn súng góc nhìn thứ nhất như CounterStrike. Đồng thời có thể tồn tại hang chục nhân vật trong cùng 1 bản đồ. Sự thay đổi vị trí của từng nhân vật sẽ phải không ngừng được chuyển tới máy của từng người chơi. Vậy nên chúng đòi hỏi phải sử dụng 1 giao thức truyền tải nhanh và dù có mất 1 hay vài gói tin thì cũng không quá quan trọng bới vị trí của từng nhân vật đều được cập nhật lại vài lần mỗi giây. Trong trường hợp đó, lập trình viên sẽ chọn dung UDP.

**Kiến trúc dự án Aloo! với Qt**

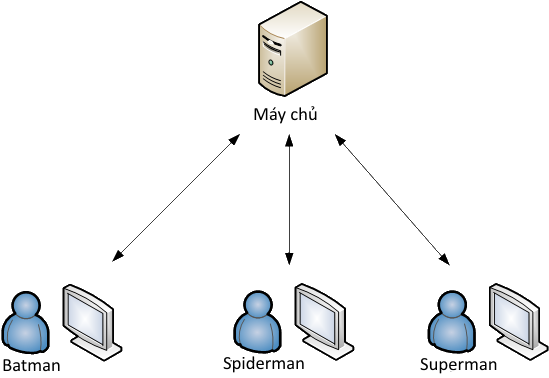
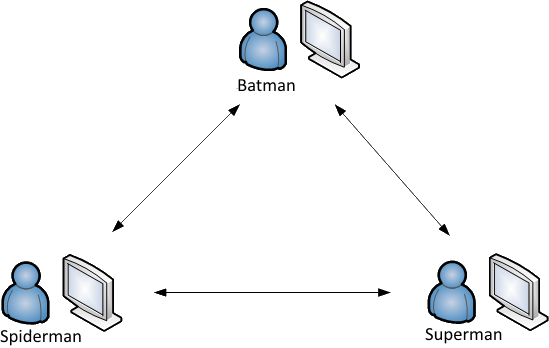
Chúng ta vừa cùng nhau điểm qua 1 vài khái niệm lý thuyết về mạng lưới. Công việc bây giờ là xác định kiến trúc mạng của chương trình tán gẫu Aloo!.

***? Kiến trúc mạng là cái thứ gì vậy ?***

Từ nãy tới giờ, chúng ta đang giả định 1 trường hợp sử dụng đơn giản nhất, đó là chỉ có 2 máy tính nối với nhau nói chuyện. Vấn đề là chương trình của chúng ta cần phải cho phép nhiều hơn 2 người cùng tán gẫu 1 lúc. Hãy tưởng tượng Spiderman cũng muốn tham gia tán gẫu với các đàn anh thì chúng ta phải đặt anh ta ở đâu ? Chẳng lẽ lại đặt giữa 2 người trước ?

**Kiến trúc mạng**

Chúng ta có 1 giải pháp khá đơn giản cho vấn đề trên.

* Kiến trúc **người dùng - máy chủ** (client/server) : đây là 1 trong những kiến trúc loại cổ điển nhất của thế giới mạng, rất dễ xây dựng và đưa vào sử dụng. Thiết bị của người dùng (Batman, Superman, Spiderman, vv…) được gọi là « người dùng » hay « client ». Ngoài những thiết bị này, chúng ta có thêm 1 máy tính gọi là « máy chủ » hay « server » chịu trách nhiệm quản lý dòng thông tin truyền tới các người dùng.
* Kiến trúc **P2P** (Cá nhân tới cá nhân hay Peer-To-Peer) : là kiến trúc phức tạp hơn, được xếp vào loại kiến trúc phân tán bởi không có thiết bị nào là máy chủ. Mỗi người dùng có thể trao đổi trực tiếp với người dùng khác. Kiến trúc này trực tiếp hơn, không cần đến sự có mặt của máy chủ nhưng trái lại việc thiết lập nó lại đòi hỏi tinh tế hơn nhiều. 

Chúng ta sẽ sử dụng kiến trúc người dùng – máy chủ vì nó đơn giản hơn.

Thế nên, chúng ta sẽ cần tạo 2 chứ không phải 1 dự án.

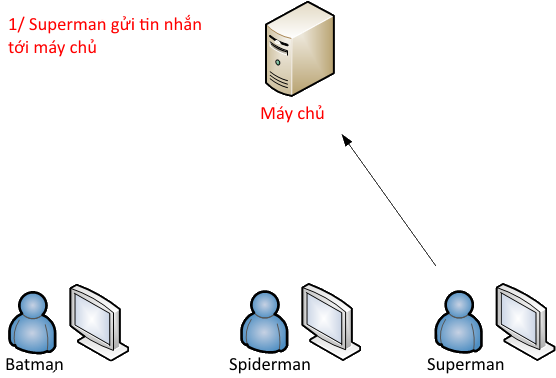
* **Dự án « máy chủ »**: tạo ra chương trình chịu trách nhiệm phân phối tin nhắn cho từng người dùng.
* **Dự án « người dùng »** : dành cho người muốn sử dụng chương trình tán gẫu.

*! Chúng ta không bắt buộc phải sử dụng một chiếc máy đặc biệt để làm máy chủ. Có thể đơn giản là chọn trong số các máy người dùng làm máy chủ. Trong trường hợp đó, trên máy người dùng đó sẽ đồng thời chạy 2 chương trình « máy chủ » và « người dùng ». Thế là trong trường hợp thử nghiệm thực tiễn, chúng ta sẽ cho 1 máy chạy đồng thời 2 chương trình và các máy còn lại chỉ chạy chương trình « người dùng ».*

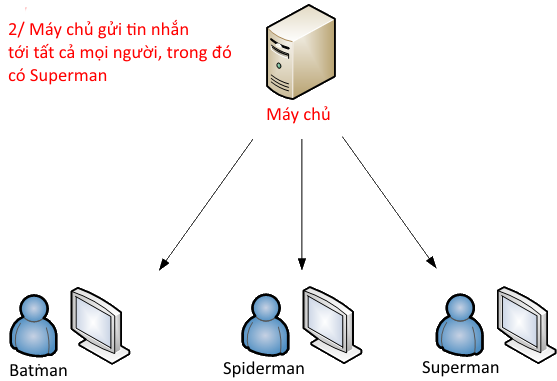
**Nguyên lý hoạt động của Aloo!**

Nguyên lý rất đơn giản : 1 người dùng gửi đi tin nhắn, tất cả những người dùng nhận được tin nhắn đó sẽ hiển thị nó trên màn hình.

Hãy chia ra thành 2 bước :



1 người dùng gửi tin nhắn tới máy chủ.



Máy chủ gửi lại tin nhắn cho tất cả mọi người để hiển thị nó lên màn hình.

***? Tại sao máy chủ gửi lại tin nhắn cho Superman dù anh ta là người đã gửi tin đó đi ?***

Đây đơn giản là 1 lựa chọn. Chúng ta hoàn toàn có thể viết chương trình yêu cầu máy chủ không gửi lại tin nhắn cho chủ nhân của nó để tránh lãng phí băng thông mạng, nhưng làm thế sẽ khiến chương trình chạy trên máy chủ của chúng ta trở nên rắc rối hơn.

Vậy nên lựa chọn đơn giản hơn là để máy chủ phân phối tin nhắn tới tất cả mọi người không trừ 1 ai. Superman sẽ nhận lại được tin nhắn của anh ta và hiển thị nó lên màn hình. Lợi ích của việc này là anh ta có thể chắc chắn là chương trình thực hiện tốt việc truyền dữ liệu qua mạng.

**Cấu trúc gói tin**

Trong mạng, các tin nhắn được gửi đi trong các gói tin. Chính chúng ta là người sẽ quyết định cấu trúc của các gói tin mà chúng ta muốn gửi.

Ví dụ, khi Batman muốn gửi đi 1 tin nhắn, 1 gói tin sẽ được tạo ra trong mạng. Dưới đây là cấu trúc gói tin mà tôi đề nghị chúng ta sẽ sử dụng trong Aloo!.



Gói tin được chia thành 2 phần :

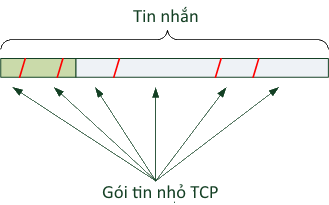
* kichThuoc : số nguyên chỉ ra kích thước của đoạn tin phía sau. Thông tin này cho phép máy chủ biết được kích thước của đoạn tin nhắn sẽ được gửi đi để có thể xác định được khi nào thì nó nhận được tin nhắn hoàn chỉnh.

*! Đại lượng này không mang kiểu int như chúng ta đã biết mà sẽ là kiểu quint16. Vì sao lại sử dụng kiểu dữ liệu này ? Vấn đề nằm ở việc kiểu int sẽ mang các kích thước khác nhau (8 bit hoặc 16 bit) tùy theo từng loại máy tính. Vậy nên chúng ta sẽ sử dụng 1 kiểu dữ liệu tạo trước bới Qt là quint16 luôn mang kích thước là 16 bit trên tất cả các máy tính. quint16 chính là viết tắt của “Qt unsigned int 16”.*

* tinNhan : chứa nội dung tin nhắn gửi đi bởi người dùng. Ở đây, chúng ta dùng kiểu QString quen thuộc.

***? Tại sao lại đặt kích thước của tin nhắn ở trước ? Chỉ bản thân tin nhắn thôi không đủ à ?***

Cần biết là giao thức TCP sẽ thực hiện việc chia nhỏ lại gói tin của chúng ta thành những gói tin nhỏ hơn trước khi truyền đi. Chúng có thể không được gửi đi ngay toàn bộ mà bị chia ra như dưới đây.



Chúng ta không thể làm chủ được kích thước của những gói tin nhỏ đó và không thể biết được gói tin của chúng ta sẽ được cắt thế nào.

Vấn đề là máy chủ sẽ chỉ nhận được từng gói tin nhỏ mỗi lần và không biết lúc nào mới nhận được toàn bộ tin nhắn.

*! Giao thức TCP khong cho phép chúng ta can thiệp vào việc chia nhỏ gói tin nhưng lại đảm bảo cho chúng ta việc các gói tin sẽ đến theo thứ tự. Thế là đã đơn giản công việc đi rất nhiều vì chúng ta không cần mất công đi sắp xếp lại thứ tự các gói tin. Thông tin thêm cho bạn nào muốn biết thì giao thức UDP không thực hiện việc đảm bảo thứ tự các gói tin.*

Thế là chúng ta sẽ cần biết khi nào thì chúng ta nhận được đầy đủ tin nhắn và không cần chờ thêm các gói tin nhỏ nữa.

Để giải quyết việc này, chúng ta thêm vào gói tin đầu tiên kích thước của tin nhắn cần gửi đi. Sau khi nhận được kích thước này, chúng ta sẽ dựa vào nó để tính toán đến khi tin nhắn được nhận đủ.

Khi nhận đủ lượng dữ liệu tương đương với kích thước nhận được từ trước chính là lúc tin nhắn được chuyển đi toàn bộ.

Không phải quá dễ dàng đúng không ? Dù sao thì tôi cũng đã báo trước cho các bạn rồi đấy.

**Dự án chạy trên máy chủ**

Như đã nói bên trên, chúng ta tạo ra 2 dự án, chạy trên máy chủ và máy người dùng thường. Chúng ta sẽ bắt đầu với dự án trên máy chủ.

**Tạo dự án**

Dự án mới được tạo ra sẽ bao gồm 3 tệp :

* main.cpp
* CuaSoMayChu.cpp
* CuaSoMayChu.h

Chúng ta cũng cần thay đổi tệp .pro để Qt biết là chúng ta muốn làm việc với mạng lưới.

1. QT += widgets network
2. # Input
3. HEADERS += CuaSoMayChu.h
4. SOURCES += CuaSoMayChu.cpp main.cpp

Với dòng lệnh QT += widgets network, Qt biết là chúng ta sẽ làm việc với hệ thống mạng vào tạo ra 1 tệp*makefile* thích hợp.

**Chương trình trên máy chủ**

Chương trình chạy trên máy chủ là 1 chương trình chạy ở trạng thái ẩn nghĩa là không cần phải có giao diện tương tác với người dùng. Chính vì thế chúng ta không bắt buộc phải tạo ra 1 cửa sổ cho chương trình này. Tuy nhiên trong bài thực hành này, chúng ta vẫn sẽ tạo ra 1 cửa sổ cho nó để người dùng có thể dừng chương trình khi đóng cửa sổ.

Cửa sổ của chúng ta sẽ rất đơn giản với dòng chữ “Máy chủ đang chạy trên cổng XXXX” và nút “Thoát”.

Việc tạo ra cửa sổ này khá dễ dàng. Phức tạp là việc quản lý giao tiếp mạng phía sau.

**main.cpp**

Tệp nguồn này vẫn đơn giản như mọi khi.

1. #include <QApplication>
2. #include "CuaSoMayChu.h"
4. int main(int argc, char\* argv[]) {
5. QApplication app(argc, argv);
7. CuaSoMayChu cuaSo;
8. cuaSo.show();
10. return app.exec();
11. }

**CuaSoMayChu.h**

1. #ifndef CUASOMAYCHU\_H
2. #define CUASOMAYCHU\_H
4. #include <QtWidgets>
5. #include <QtNetwork>
7. class CuaSoMayChu : public QWidget {
8. Q\_OBJECT
10. public:
11. CuaSoMayChu();
12. void guiTinNhanChoMoiNguoi(const QString &tinNhan);
14. private slots:
15. void ketNoiMoi();
16. void nhanDuLieu();
17. void ngatKetNoi();
19. private:
20. QLabel \*trangThai;
21. QPushButton \*thoat;
23. QTcpServer \*mayChu;
24. QList<QTcpSocket \*> nguoiDung;
25. quint16 kichThuoc;
26. };
28. #endif

Cửa sổ này thừa kế từ lớp *QWidget*, giúp chúng ta tạo ra 1 cửa sổ đơn giản nhất. Nó chỉ bao gồm 1 nhãn QLabel để viết văn bản và 1 nút bấm *QPushButton*.

Thêm vào đó chúng ta thêm 1 số thuộc tính đặc biệt để thực hiện giao tiếp mạng.

* QTcpServer \*mayChu : là đối tượng đại diện máy chủ trong mạng.
* QList<QTcpSocket \*> nguoiDung : là mảng chứa danh sách người dùng đang kết nối. Chúng ta cũng có thể sử dụng kiểu mảng cổ điển nhưng ở đây chúng ta dùng *QList* để thao tác với mảng động. Thực tế là chúng ta hoàn toàn không biết trước số người sẽ kết nối tham gia tán gẫu. Mỗi *QTcpSocket* của mảng chính là đại diện của 1 kết nối của người dùng.

*! Chúng ta sẽ không thảo luận nhiều về QList biết rằng đây là 1 lớp cho phép chúng ta thao tác với các mảng mà kích thước thay đổi. Điều này rất có ích khi chúng ta không biết trước số người sẽ dùng chương trình. Để biết làm sao để sử dụng lớp này, hãy tham khảo tài liệu Qt.*

* quint16 kichThuoc : dùng để ghi nhớ kích thước dữ liệu mà máy chủ cần nhận. Chúng ta đã nhắc đến nó bên trên và sẽ còn quay lại thảo luận sau.

Bên cạnh các thuộc tính, lớp này còn chứa các phương thức hay slot khác.

* Phương thức khởi tạo : tạo ra các widget và khởi động đối tượng máy chủ *QTcpServer*.
* guiTinNhanChoMoiNguoi() : gửi tin nhắn trong tham số cho tất cả những người đang kết nối.
* Slot ketNoiMoi() : kích hoạt khi người dùng mới kết nối
* Slot nhanDuLieu() : kích hoạt khi máy chủ nhận dữ liệu. Chú ý là ở đây, slot này được kích hoạt mỗi khi gói tin nhỏ TCP được nhận. Để xác định lúc tin nhắn hoàn chỉnh được nhận, chúng ta cần chờ khi máy chủ nhận đủ lượng dữ liệu được ghi trong thuộc tính *kichThuoc*.
* Slot ngatKetNoi() : kích hoạt khi người dùng thoát kết nối.

Bây giờ sẽ là phần xử lý thú vị của các phương thức này.

**CuaSoMayChu.cpp**

**Phương thức khởi tạo**

Phương thức khởi tạo chiu trách nhiệm sắp xếp các widget vào cửa sổ của như khởi động máy chủ nhờ*QTcpSocket*.

1. CuaSoMayChu::CuaSoMayChu() {
2. // Tao va sap xep widget
3. trangThai = new QLabel();
4. thoat = new QPushButton(tr("Thoát"));
6. connect(thoat, SIGNAL(clicked()), qApp, SLOT(quit()));
8. QVBoxLayout \*lop = new QVBoxLayout();
9. lop->addWidget(trangThai);
10. lop->addWidget(thoat);
11. setLayout(lop);
12. setWindowTitle(tr("Máy chủ Aloo!"));
14. // Quan ly may chu
15. mayChu = new QTcpServer(this);
16. if (!mayChu->listen(QHostAddress::Any, 50885)) { // Khoi dong may chu o cong 50585 và tren tat ca cac dia chi IP cua may
17. // Neu may chu chua duoc khoi dong
18. trangThai->setText(tr("Máy chủ chưa khởi động được. Lý do :<br />") +mayChu->errorString());
19. } else {
20. // Neu may chu duoc khoi dong thanh cong
21. trangThai->setText(tr("Máy chủ được mở trên cổng <strong>") + QString::number(mayChu->serverPort()) + tr("</strong>.<br />Người dùng có thể nhanh chóng kết nối để tán gẫu."));
22. connect(mayChu, SIGNAL(newConnection()), this, SLOT(ketNoiMoi()));
23. }
25. kichThuoc = 0;
26. }

Trong đoạn mã trên, tôi đã cố gắng thêm nhiều chú thích để cho mọi người có thể hiểu dễ dàng hơn chuyện gì đang diễn ra.

Chúng ta có thể chia rõ ra 2 phần, bao gồm phần đầu là xây dựng của sổ giao diện và phần tiếp theo là khởi động máy chủ.

Phần thứ 2 là chủ đề thú vị của bài học lần này. Chúng ta đã tạo ra 1 đối tượng *QTcpServer* sau đó truyền cho nó tham số là con trỏ trỏ tới cửa sổ đang tạo. Điều này giúp cho khi chúng ta đóng cửa sổ thì chương trình máy chủ sẽ tự động kết thúc theo.

Sau đó chúng ta khởi động máy chủ với câu lệnh mayChu->listen(QHostAddress::Any, 50885). Câu lệnh này yêu cầu 2 tham số.

* Địa chỉ IP : đây là địa chỉ IP của máy chủ mà trên đấy nó sẽ thực hiện quan sát xem nếu có người dùng nào kết nối đến không. Như đã nói lúc trước, mỗi thiết bị có thể có nhiều IP khác nhau. Cụm mã QHostAddress::Any cho phép tất cả các kết nối dù là nội tại (từ trên cùng thiết bị với chương trình máy chủ), từ thiết bị trên cùng mạng nội bộ hay từ thiết bị trên Internet đều có thể được thiết lập.
* Cổng : là cổng mà chúng ta muốn chạy chương trình máy chủ. Ở đây, tôi đã chọn 1 giá trị bất kỳ nằm giữa 1024 và 65535. Tham số này không phải là bắt buộc. Nếu chúng ta không đưa ra giá trị của tham số này thì máy tính sẽ tự động chọn 1 cổng đang rảnh để chạy chương trình. Nếu trên máy các bạn mà cổng này đã được sử dụng bới chương trình khác, đừng ngại chọn 1 cổng khác để thử lại.

Phương thức listen() trả về kết quả kiểu boolean với giá trị đúng nếu máy chủ được khởi động thành công và sai nếu có vấn đề xảy ra trong quá trình đó. Trong trường hợp có vấn đề, chúng ta sẽ cho hiện ra 1 thông báo trong cửa sổ giao diện.

Nếu máy chủ được khởi động thành công thì chúng ta sẽ kết nối tín hiện newConnection() tới slot tự chếketNoiMoi() chuyên xử lý việc có người dùng mới kết nối tới hệ thống tán gẫu.

Nếu mọi chuyện ổn thỏa thì chúng ta sẽ nhận được kết quả sau :

Nếu có vấn đề, chúng ta sẽ nhận được các thông báo lỗi tương ứng. Hãy thử kiểm tra nếu chúng ta lại khởi động chương trình thêm lần nữa dù chưa kết thúc lần chạy trước.

Bởi vì cổng 50885 đã được chương trình máy chủ chạy trước sử dụng nên chương trình được khởi động sau sẽ không thể chạy trên cổng đó nữa. Vậy là chúng ta thấy có lỗi trên xảy ra.

**Slot ketNoiMoi()**

Slot này được kích hoạt khi có người dùng mới kết nối tới máy chủ.

1. void CuaSoMayChu::ketNoiMoi() {
2. guiTinNhanChoMoiNguoi (tr("<em>Một người mới  vừa tham gia với chúng ta !</em>"));
3. QTcpSocket \*nguoiDungMoi = mayChu->nextPendingConnection();
5. nguoiDung << nguoiDungMoi;
7. connect(nguoiDungMoi, SIGNAL(readyRead()), this, SLOT(nhanDuLieu()));
8. connect(nguoiDungMoi, SIGNAL(disconnected()), this, SLOT(ngatKetNoi()));
9. }

Chúng ta gửi cho tất cả những người dùng đã kết nối từ trước 1 tin nhắn thông báo có người mới vừa gia nhập. Mã xử lý của phương thức guiTinNhanChoMoiNguoi() sẽ được bàn đến sau.

Mỗi người dùng được đại diện bằng 1 đối tượng *QTcpSocket*. Để tạo ra socket (hãy tưởng tượng nó như 1 cái cửa) tương ứng với người dùng mới vừa kết nối, chúng ta sử dụng phương thức nextPendingConnection() của lớp *QTcpServer*. Phương thức này gửi trả về đối tượng *QTcpSocket* dành cho người dùng mới.

Như đã bàn bên trên, chúng ta sẽ lưu trữ danh sách người dùng đang đăng nhập vào trong 1 mảng tên là*nguoiDung*.

Mảng này được quản lý bởi lớp *QList*. Lớp này khá dễ sử dụng. Để thêm vào 1 thành viên cho mảng, chúng ta sử dụng câu lệnh dưới đây.

1. nguoiDung << nguoiDungMoi;

Lợi ích của việc ghi đè toán tử << bắt đầu hiện ra cool.

Tiếp đến, chúng ta kết nối các tín hiệu mà người dùng có thể gửi đến với các slot đã tạo ra. Có 2 tín hiệu cần chúng ta xử lý :

* readyRead() : tín hiệu thông báo người dùng đã gửi dữ liệu. Tín hiệu được phát ra mỗi khi chúng ta nhận được 1 gói tin TCP. Khi người dùng gửi 1 tin nhắn đến, tín hiệu này có thể được phát ra nhiều lần cho đến khi tất cả các gói tin thành phần đều đến nơi. Slot mà chúng ta tự viết nhanDuLieu() sẽ chịu trách nhiệm xử lý các gói tin này.
* disconnected() : tín hiệu thông báo người dùng đã thoát kết nối. Slot chúng ta cần viết sẽ thông báo cho tất cả những người dùng khác về việc 1 người dùng rời đi và xóa đối tượng *QTcpSocket* đại diện của người dùng này trong danh sách người đang kết nối.

**Slot nhanDuLieu()**

Đây chính là phần quan trọng nhất của bài học, slot được kích hoạt mỗi khi chúng ta nhận 1 gói tin TCP của người dùng.

Có ít nhất 2 vấn đề cần giải quyết ở đây :

* Do tin nhắn bị chia thành nhiều gói tin nhỏ, cần phải chờ sau khi nhận toàn bộ trước khi có thông báo là đã nhận được tin nhắn.
* Cúng 1 slot được kích hoạt bất kỳ là do tin nhắn đến từ người dùng nào. Vậy làm sao để xác nhận được ai là chủ nhân tin nhắn để có thể tìm nhận được dữ liệu ?

Cần phải dùng đối tượng *QTcpSocket* để nhận gói tin được mạng lưới truyền tới. Nhưng làm sao giải quyết được vấn đề là tất cả tín hiệu từ tất cả người dùng đếu được kết nối đến cùng 1 slot ?

Làm sao để slot biết được đối tượng *QTcpSocket* nào là nơi đã nhận dữ liệu.

Các bạn không cần mất công đoán vì thậm chí bản thân tôi cũng không nghĩ ra được ý tưởng gì vào thời điểm tôi bắt đầu viết bài thực hành này.

Thế rồi, trong 1 phút lóe sang, tôi phát hiện ra phương thức sender() của lớp *QObject* có thể được gọi từ trong slot, trả về kết quả là con trỏ trên nguồn phát của tin nhắn. Hữu dụng phết !

Vấn đề mới sinh ra là phương thức này trả về kết quả là 1 đối tượng *QObject* bởi về cơ bản thì nó không biết từ trước kết quả trả về thuộc về lớp cụ thể nào. Để chuyển đổi nó từ *QObject* chung chung về thành *QTcpSocket*mà chúng ta muốn sử dụng, cần thực hiện ép buộc chuyển đổi kiểu đối tượng với phương thứcqobject\_cast().

Tóm lại, để nhận được đổi tượng *QTcpSocket* nguồn phát tin nhắn, chúng ta sẽ sử dụng câu lệnh sau.

1. QTcpSocket \*socket = qobject\_cast<QTcpSocket \*>(sender());

*! Phương thức qobject\_cast() tương tự với dynamic\_cast() của thư viện chuẩn của C++. Chức năng của nó là ép buộc chuyển đổi đối tượng từ kiểu này sang kiểu khác.*

Trong 1 số trường hợp, có thể qobject\_cast() không hoạt động (do nguồn phát về bản chất không phải là đối tượng thuộc kiểu *QTcpSocket* như chúng ta mong đợi. Trong trường hợp đó, phương thức này sẽ trả về 0. Vì thế, chúng ta cần kiểm tra là phương thức này đã thực hiện thành công việc chuyển đối kiểu trước khi tiếp tục thực hiện xử lý.

Chúng ta sẽ sử dụng lệnh return để thoát khỏi slot trong trường hợp phương thức chuyển đối không thực hiện thành công.

1. QTcpSocket \*socket = qobject\_cast<QTcpSocket \*>(sender());
2. if (socket == 0) { // Neu khong xac dinh duoc nguon phat, chung ta se dung xu ly
3. return;
4. }

Chúng ta có thể tiếp tục xử lý lấy dự liệu, bắt đầu bằng việc tạo ra luồng để đọc dữ liệu lưu trong socket.

1. QDataStream in(socket);

Đối tượng *in* sẽ cho phép chúng ta đọc lấy nội dung từ gói tin nhận được trong socket tương ứng của người dùng.

Đây là lúc chúng ta sẽ sử dụng đến dữ liệu *kichThuoc* mà chúng ta đã đưa vào thành thuộc tính của lớp. Nếu khi slot được kích hoạt mà giá trị thuộc tính này đang là 0 thì có nghĩa là chúng ta đang nhận mảnh đầu của tin nhắn. Chúng ta sẽ hỏi xem socket nhận được bao nhiêu dữ liệu nhờ phương thức byteAvailable(). Nếu kết quả nhận được nhỏ hơn kích thước của 1 biến kiểu *quint16* thì chúng ta sẽ dừng xử lý và đợi đến lần kích hoạt tiếp theo của slot để kiểm tra xem chúng ta có thể lấy ra được dữ liệu về kích thước của tin nhắn người dùng không.

1. if (kichThuoc == 0) { //Neu chua biet kich thuoc tin nhan thi chung ta se thu tim trong goi du lieu vua toi
2. if (socket->bytesAvailable() < (int)sizeof(quint16)) { //Kich thuoc goi tin nho hon kich thuc kieu so nguyen
3. return;
4. }
6. in >> kichThuoc; // Neu nhan duoc kich thuoc tin nhan thi lay ra gia tri do
7. }

Dòng lệnh thứ 5 chỉ được thực hiện nếu như chúng ta nhận đủ dữ liệu để xác định kích thước tin nhắn. Nếu không thi xử lý sẽ bị dừng từ trước nhờ lệnh return.

Bây giờ, kích thước tin nhắn đã được lấy ra và lưu lại.

Để hiểu rõ đoạn mã này, trước hết cần nhớ lại việc tin nhắn của chúng ta sẽ bị chia nhỏ thành nhiều gói tin.

Slot được kích hoạt mỗi khi một gói tin nhỏ tới nơi.

Mỗi lần như thế, chúng ta sẽ xác định đã đủ lượng dữ liệu để lấy ra thông tin kichThuoc tin nhắn chưa. Nếu chưa thì thoát khỏi slot và chờ đến lần kích hoạt sau.

Bây giờ, sau khi đã nhận được được kích thước của tin nhắn, chúng ta sẽ tiếp tục xử lý lấy ra nội dung tin nhắn.

1. // Biet kich thuoc, chung ta se kiem tra xem da nhan duoc toan bo tin nhan chua
2. if (socket->bytesAvailable() < kichThuoc) { // Neu chua nhan du tin nhan thi thoat xu ly
3. return;
4. }

Nguyên lý vẫn như trên, chúng ta không thực hiện xử lý khi nào mà chưa nhận đủ tin nhắn.

Nếu mọi thứ diễn ra tốt đẹp, chúng ta sẽ chuyển qua xử lý tiếp theo. Nếu những lệnh này được thực hiện thì nghĩa là chúng ta đã nhận được tin nhắn hoàn chỉnh và sẽ ghi nó vào đối tượng kiểu *QString*.

1. QString tinNhan;
2. in >> tinNhan;

Lúc này, *tinNhan* chứa nội dung tin nhắn của người dùng.

Phù, vậy là máy chủ đã nhận được tin nhắn rồi. Thế nhưng công việc của chúng ta vẫn chưa kết thúc ở đây. Máy chủ bây giờ sẽ gửi tin nhắn đó đến cho tất cả người dùng. Xử lý này được thực hiện trong phương thứcguiTinNhanChoMoiNguoi() mà chúng ta đã nhắc đến lúc trước.

1. // Gui tin nhan den tat ca nguoi dung
2. guiTinNhanChoMoiNguoi(tinNhan);

Gần xong rồi, chỉ còn việc đặt lại giá trị 0 cho thuộc tính *kichThuoc* để máy chủ có thể tiếp tục tiếp nhận tin nhắn khác.

1. // Dat lai kich thuoc la 0 de cho tin nhan tiep theo
2. kichThuoc = 0;

Nếu chúng ta không thực hiện bước này thì máy chủ sẽ nghĩ tin nhắn tiếp theo có cùng kích thước với tin nhắn vừa nhận và điều này thì không phải lúc nào cũng đúng.

Tóm lại, đây là mã hoàn chỉnh slot này của chúng ta.

1. void CuaSoMayChu::nhanDuLieu() {
2. QTcpSocket \*socket = qobject\_cast<QTcpSocket \*>(sender());
3. if (socket == 0) { // Neu khong xac dinh duoc nguon phat, chung ta se dung xu ly
4. return;
5. }
7. QDataStream in(socket);
8. if (kichThuoc == 0) { //Neu chua biet kich thuoc tin nhan thi chung ta se thu tim trong goi du lieu vua toi
9. if (socket->bytesAvailable() < (int)sizeof(quint16)) { //Kich thuoc goi tin nho hon kich thuc kieu so nguyen
10. return;
11. }
12. in >> kichThuoc; // Neu nhan duoc kich thuoc tin nhan thi lay ra gia tri do
13. }
15. // Biet kich thuoc, chung ta se kiem tra xem da nhan duoc toan bo tin nhan chua
16. if (socket->bytesAvailable() < kichThuoc) { // Neu chua nhan du tin nhan thi thoat xu ly
17. return;
18. }
20. QString tinNhan;
21. in >> tinNhan;
23. guiTinNhanChoMoiNguoi(tinNhan);
25. // Dat lai kich thuoc la 0 de cho tin nhan tiep theo
26. kichThuoc = 0;
27. }

Hy vọng là tôi đã giải thích đủ rõ ràng vì slot này hơi phức tạp và khó theo dõi nếu chỉ nhìn đoạn mã. Điểm mấu chốt cần chú ý và hiểu là cách sử dụng các lệnh return trong đoạn xử lý. Vì slot được kích hoạt mỗi khi có 1 gói tin chuyển đến nên các câu lệnh có thể được thực hiện nhiều lần trong toàn bộ quá trình nhận 1 tin nhắn.

Nếu slot được thực thi từ đầu đến cuối thì nghĩa là tin nhắn đã được nhận đầy đủ.

**Slot ngatKetNoi()**

Slot này được kích hoạt khi một người dùng thoát khỏi hệ thống.

Trong trường hợp đó, chúng ta sẽ gửi tin nhắn cho tất cả những người còn lại để thông báo có 1 người vừa rời đi. Đồng thời chúng ta cũng sẽ xóa đối tượng đại diện *QTcpSocket* tương ứng với người dùng đó trong danh sách *QList* của chúng ta và máy chủ sẽ không để ý đến người dùng đó nữa.

1. void CuaSoMayChu::ngatKetNoi() {
2. guiTinNhanChoMoiNguoi(tr("<em>1 người dùng vừa mới rời đi</em>"));
4. // Xac dinh xem ai la nguoi ngat ket noi
5. QTcpSocket \*socket = qobject\_cast<QTcpSocket \*>(sender());
6. if (socket == 0) { // Neu khong tim thay nguoi gui tin hieu thi huy bo xu ly
7. return;
8. }
9. nguoiDung.removeOne(socket);
10. socket->deleteLater();
11. }

Vì đồng thời tín hiệu của tất cả mọi người đều kết nối đến cùng 1 slot này, chúng ta gặp lại cùng vấn đề như trên là không biết ai là chủ nhân của tín hiệu ngắt kết nối. Để xử lý thì chúng ta sử dụng kỹ thuật y như tình huống trước.

Phương thức removeOne() của *QList* cho phép xóa 1 con trỏ vốn là thành viên trong mảng. Danh sách người dùng của chúng ta sẽ được cập nhật.

Công việc của cùng chỉ còn là xóa bản thân đối tượng socket tương ứng với người dùng đó.

Để xóa đối tượng, thông thường chúng ta sẽ thực hiện delete nguoiDungVuaThoat;. Vấn đề là nếu chúng ta xóa nguồn phát tín hiệu của slot đang xử lý thì có thể làm treo xử lý của Qt. Thật may là mọi thứ đều đã được tính trước. Chúng ta chỉ cần gọi phương thức deleteLater() và Qt sẽ chịu trách nhiệm xóa đối tượng sau khi slot hoàn thành xử lý.

**Phương thức guiTinNhanChoMoiNguoi()**

Lần này thì không phải là 1 slot.

Đây đơn giản là đoạn mã mà chúng ta sử dụng lặp đi lặp lại nhiều lần nên tôi nghĩ là sẽ lưu nó thành 1 phương thức của lớp và gọi mỗi khi cần.

Trong slot nhanDuLieu(), chúng ta đã nhận 1 tin nhắn từ phía người dùng. Ở đây, công việc lại ngược lại, đó là gửi tin nhắn đi.

1. void CuaSoMayChu::guiTinNhanChoMoiNguoi(const QString &tinNhan) {
2. // Chuan bi tin nhan gui di
3. QByteArray goiTinNhan;
4. QDataStream out(&goiTinNhan, QIODevice::WriteOnly);
6. out << (quint16) 0; // Viet gia tri 0 o dau goi tin de dat truoc cho de viet kich thuoc tin nhan
7. out << tinNhan; // Viet noi dung tin nhan vao goi tin
9. out.device()->seek(0); // Quay ve dau goi tin
10. out << (quint16) (goiTinNhan.size() - sizeof(quint16)); // Thay 0 bang gia tri kich thuoc that cua tin nhan
12. // Gui tin cho tat ca nguoi dung ket noi
13. for (int i = 0; i < nguoiDung.size(); i++) {
14. nguoiDung[i]->write(goiTinNhan);
15. }
16. }

Thêm 1 chút giải thích để các bạn có thể hiểu rõ hơn.

Chúng ta tạo ra 1 đối tượng *QByteArray* tên là *goiTinNhan* sẽ chứa toàn bộ nội dung tin nhắn chúng ta gửi đi trên mạng. Lớp *QByteArray* dùng để quản lý 1 chuỗi dữ liệu nhị phân bất kỳ.

Chúng ta cùng dùng đến lớp *QDataStream* như trên để dễ dàng đọc và ghi vào đối tượng *QByteArray*. Nó cho phép chúng ta dùng toán tử <<.

Điều đặc biệt là ở đây, chúng ta viết nội dung tin nhắn trước rồi mới quay lại điền kích thước của nó và viết lên đầu gói tin.

Đây là các bước mà chúng ta đã thực hiện.

1. Ghi giá trị 0 kiểu *quint16* vào đầu gói tin nhắn để đặt chỗ trước cho dữ liệu kích thước.
2. Ghi nội dung tin nhắn *QString* vào gói tin nhắn. Nội dung tin được truyền như tham số của phương thứcguiTinNhanChoMoiNguoi().
3. Quay lại đầu gói tin và thay thế giá trị 0 bằng kích thước thật của tin nhắn. Kích thước này được tính thông qua phép trừ đơn giản, lấy kích thước của cả gói tin trừ đi kích thước của phần dữ liệu kích thước có kiểu *quint16*.

Thế là gói tin nhắn của chúng ta đã sắn sàng để chuyển đi. Để chuyển chúng đi trên mạng, cần sử dụng đến phương thức write() của socket.

Để chuyển tin nhắn này đến tất cả mọi người, chúng ta sẽ thực hiện vòng lặp trên các thành viên của mảng kiểu*QList* rồi gửi tin đến từng người.

Và thế là tin nhắn được chuyển đi !

**CuaSoMayChu.cpp hoàn chỉnh**

Và đây là mã nguồn hoàn chỉnh của tệp xử lý chúng ta vừa viết.

1. #include "CuaSoMayChu.h"
3. CuaSoMayChu::CuaSoMayChu () {
4. // Tao va sap xep widget
5. trangThai = new QLabel();
7. thoat = new QPushButton(tr("Thoát"));
9. connect(thoat, SIGNAL(clicked()), qApp, SLOT(quit()));
11. QVBoxLayout \*lop = new QVBoxLayout();
12. lop->addWidget(trangThai);
13. lop->addWidget(thoat);
14. setLayout(lop);
16. setWindowTitle(tr("Máy chủ Aloo!"));
18. // Quan ly may chu
19. mayChu = new QTcpServer(this);
20. if (!mayChu->listen(QHostAddress::Any, 50885)) { // Khoi dong may chu o cong 50585 và tren tat ca cac dia chi IP cua may
21. // Neu may chu chua duoc khoi dong
22. trangThai->setText(tr("Máy chủ chưa khởi động được. Lý do :<br />") +mayChu->errorString());
23. } else {
24. // Neu may chu duoc khoi dong thanh cong
25. trangThai->setText(tr("Máy chủ được mở trên cổng <strong>") + QString::number(mayChu->serverPort()) + tr("</strong>.<br />Người dùng có thể nhanh chóng kết nối để tán gẫu."));
26. connect(mayChu, SIGNAL(newConnection()), this, SLOT(ketNoiMoi()));
27. }
29. kichThuoc = 0;
30. }
32. void CuaSoMayChu::ketNoiMoi() {
33. guiTinNhanChoMoiNguoi (tr("<em>Một người mới  vừa tham gia với chúng ta !</em>"));
35. QTcpSocket \*nguoiDungMoi = mayChu->nextPendingConnection();
36. nguoiDung << nguoiDungMoi;
38. connect(nguoiDungMoi, SIGNAL(readyRead()), this, SLOT(nhanDuLieu()));
39. connect(nguoiDungMoi, SIGNAL(disconnected()), this, SLOT(ngatKetNoi()));
40. }
42. void CuaSoMayChu::nhanDuLieu() {
43. QTcpSocket \*socket = qobject\_cast<QTcpSocket \*>(sender());
44. if (socket == 0) { // Neu khong xac dinh duoc nguon phat, chung ta se dung xu ly
45. return;
46. }
48. QDataStream in(socket);
50. if (kichThuoc == 0) { //Neu chua biet kich thuoc tin nhan thi chung ta se thu tim trong goi du lieu vua toi
51. if (socket->bytesAvailable() < (int)sizeof(quint16)) { //Kich thuoc goi tin nho hon kich thuc kieu so nguyen
52. return;
53. }
54. in >> kichThuoc; // Neu nhan duoc kich thuoc tin nhan thi lay ra gia tri do
55. }
57. // Biet kich thuoc, chung ta se kiem tra xem da nhan duoc toan bo tin nhan chua
58. if (socket->bytesAvailable() < kichThuoc) { // Neu chua nhan du tin nhan thi thoat xu ly
59. return;
60. }
62. QString tinNhan;
63. in >> tinNhan;
65. guiTinNhanChoMoiNguoi(tinNhan);
67. // Dat lai kich thuoc la 0 de cho tin nhan tiep theo
68. kichThuoc = 0;
69. }
71. void CuaSoMayChu::ngatKetNoi() {
72. guiTinNhanChoMoiNguoi(tr("<em>1 người dùng vừa mới rời đi</em>"));
74. // Xac dinh xem ai la nguoi ngat ket noi
75. QTcpSocket \*socket = qobject\_cast<QTcpSocket \*>(sender());
76. if (socket == 0) { // Neu khong tim thay nguoi gui tin hieu thi huy bo xu ly
77. return;
78. }
80. nguoiDung.removeOne(socket);
81. socket->deleteLater();
82. }
84. void CuaSoMayChu::guiTinNhanChoMoiNguoi(const QString &tinNhan) {
85. // Chuan bi tin nhan gui di
86. QByteArray goiTinNhan;
88. QDataStream out(&goiTinNhan, QIODevice::WriteOnly);
89. out << (quint16) 0; // Viet gia tri 0 o dau goi tin de dat truoc cho de viet kich thuoc tin nhan
90. out << tinNhan; // Viet noi dung tin nhan vao goi tin
91. out.device()->seek(0); // Quay ve dau goi tin
92. out << (quint16) (goiTinNhan.size() - sizeof(quint16)); // Thay 0 bang gia tri kich thuoc that cua tin nhan
94. // Gui tin cho tat ca nguoi dung ket noi
95. for (int i = 0; i < nguoiDung.size(); i++) {
96. nguoiDung[i]->write(goiTinNhan);
97. }
98. }

**Khởi động máy chủ**

Tin tốt đây ! Chắc hẳn các bạn cũng tự đoán được rồi, đó là chương trình thực thi trên máy chủ của chúng ta đã hoàn thành. Phần khó nhất là viết mã xử lý đã xong, chỉ còn việc biên dịch đoạn mã này và chạy chương trình trên máy chủ là được.

Trong 1 số trường hợp, chúng ta có thể bị đập vào mặt (foot-in-mouth) 1 thông điệp cảnh báo từ phía tường lửa của Windows.

Lý do của cảnh báo này là vì chương trình của chúng ta sẽ thực hiện các xử lý giao tiếp với mạng lưới và cần nhận được sự cho phép của chúng ta để thực thi. Để đồng ý, hãy ấn vào nút “Allow access”.

*! Trong ví dụ này thì tôi giả định là tường lửa máy tính là tường lửa duy nhất được sử dụng giữa các thiết bị. Nếu các bạn sử Aloo! thông qua Internet cần tính trước khả năng phải yêu cầu****bộ định tuyến****(router) của mạng nội bộ của các bạn cho phép mở cổng 50885 với giao thức TCP. Nếu chương trình Aloo! không hoạt động thì nhiều khả năng là do nó đã bị chặn bởi 1 tường lửa ở đâu đó.*

Hoan hô ! Máy chủ của chúng ta đã khởi động thành công.

Chúng ta có thể để chương trình chạy ở trạng thái ẩn trên máy (thu nhỏ cửa sổ xuống thanh công cụ). Chương trình sẽ giúp chúng ta thực hiện phân phối tin nhắn giữa nhiều người dùng.

Kiệt sức chưa ? Tin buồn là đây mới chỉ là 1 nửa công việc chúng ta phải làm. Bây giờ là lúc chúng ta phải viết chương trình thực thi trên máy người dùng để họ có thể sử dụng được Aloo!. May mắn là chúng ta đã thực hiện phần khó nhất trong khi viết mã xử lý của nhanDuLieu(). Phần tiếp theo sẽ xong nhanh thôi.

**Chương trình trên máy người dùng**

Cửa sổ của chương trình thực thi trên máy chủ càng tối giản bao nhiêu thì cửa sổ trên máy người dùng càng tinh tế bấy nhiêu. Chúng ta cần thiết kế tương đối sáng sủa và thân thiện vì nó tương tác trực tiếp với người dùng. Người dùng sẽ thường xuyên dùng đến nó để soạn thảo tin nhắn.

Đây là cửa sổ mà chúng ta muốn thực hiện.

Dự án này cũng bao gồm 3 tệp mã nguồn : main.cpp, CuaSoNguoiDung.cpp và CuaSoNguoiDung.h.

**Thiết kế cửa sổ với QtDesigner**

Việc thiết kế cửa sổ sao cho đẹp mắt không phải là nội dung chính yếu của bài học lần này. Để tiết kiệm thời gian, tôi đề nghị là chúng ta sẽ sử dụng 1 bộ công cụ chuyên để thiết kế giao diện đồ họa của Qt, đó là QtDesigner. Chương trình này đã được tích hợp sẵn trong QtCreator của chúng ta. Nó ra đời chính là để giúp những lập trình viên như chúng ta, những người hiểu rõ cách thức xây dựng cửa sổ từ dòng lệnh, có thể giản lược công việc thiết kế với hàng tá lớp sắp xếp và widget khác nhau và tập trung vào mã xử lý logic của chương trình.

*! Tôi cũng hoàn toàn có thể sử dụng QtDesigner trong khi thiết kế cửa sổ của BanDoX mà chúng ta đã cùng viết lúc trước. Tuy nhiên cửa số đó được khá phức tạp với nhiều thẻ khác nhau nên cuối cùng tôi đã tự xây dựng nó từ dòng lệnh. Thực ra, QtDesigner chỉ thể hiện ra ưu điểm của nó rõ nhất khi thiết kế những cửa sổ có độ phức tạp vừa phải như trong ví dụ này của chúng ta.*

*! QtDesigner là công cụ thiết kế sử dụng giao diện kéo thả và các bản mẫu. Nó không quá khó để sử dụng nên bài học riêng về công cụ này đã bị tôi bỏ qua. Các bạn có thể tự mình nghiên cứu cách sử dụng nó thông qua ví dụ mà tôi sẽ sử dụng trong bài học này.*

QtDesigner tạo ra 1 tệp có phần mở rộng là .ui. Để không mất thời gian của chúng ta, các bạn có thể trực tiếp tải tệp sinh ra từ thiết kế của tôi cho phần tiếp theo của bài học.

**NguoiDung.pro**

Trong tệp tùy chỉnh .pro của dự án này, chúng ta cần thêm dòng khai báo để Qt biết là chúng ta muốn sử dụng tệp .ui trong dự án cũng như là sẽ cần sử dụng tới module mạng.

Tệp .pro của dự án này sẽ có dạng như sau.

1. QT += widgets network
3. HEADERS += CuaSoNguoiDung.h
4. FORMS += CuaSoNguoiDung.ui
5. SOURCES += main.cpp CuaSoNguoiDung.cpp

**main.cpp**

Tệp này vẫn chỉ đơn giản là mở ra cửa sổ của chương trình.

1. #include <QApplication>
2. #include "CuaSoNguoiDung.h"
4. int main(int argc, char\* argv[]) {
5. QApplication app(argc, argv);
6. CuaSoNguoiDung cuaSo;
7. cuaSo.show();
9. return app.exec();
10. }

**CuaSoNguoiDung.h**

Trong quá trình biên dịch, tệp CuaSoNguoiDung.ui sẽ sinh ra tệp tiêu đề ui\_CuaSoNguoiDung.h. Tệp tiêu đề này sẽ định nghĩa lớp *Ui::CuaSoNguoiDung*. Khi xây dựng cửa sổ của chúng ta cần thêm tệp này vào đầu đoạn mã để sử dụng được lớp.

Lớp cửa sổ của chúng ta lần này sẽ thực hiện thừa kế phức nghĩa là đồng thới kế thừa phương thức và thuộc tính nhiều lớp.

1. #ifndef CUASONGUOIDUNG\_H
2. #define CUASONGUOIDUNG\_H
4. #include <QtWidgets>
5. #include <QtNetwork>
6. #include "ui\_CuaSoNguoiDung.h"
8. class CuaSoNguoiDung : public QWidget, private Ui::CuaSoNguoiDung {
9. Q\_OBJECT
11. public:
12. CuaSoNguoiDung();
14. private slots:
15. void anNutKetNoi();
16. void anNutGuiTin();
17. void anEnterGuiTin();
18. void nhanDuLieu();
19. void ketNoi();
20. void ngatKetNoi();
21. void loiSocket(QAbstractSocket::SocketError loi);
23. private:
24. QTcpSocket \*socket; // May chu
25. quint16 kichThuoc;
26. };
28. #endif

Chú ý đừng quên thêm tệp tiêu đề ui\_CuaSoNguoiDung.h vào đầu đoạn mã nguồn.

Cửa sổ có khá nhiều slot cần chúng ta viết mã xử lý. May là chúng không quá phức tạp.

* anNutKetNoi() : được kích hoạt khi nút “Kết nối” được ấn để kết nối thiết bị đến máy chủ.
* anNutGuiTin() : được kích hoạt khí nút “Gửi tin” được ấn để gửi tin nhắn trong khung tán gẫu tới máy chủ.
* anEnterGuiTin() : được kích hoạt khi chúng ta ấn *Enter* để thực hiện gửi tin nhắn đang soạn.
* nhanDuLieu() : kích hoạt khi nhận 1 gói dữ liệu từ máy chủ. Hoạt động của nó tương tự với slot cùng tên trong dự án máy chủ.
* ketNoi() : kích hoạt khi kết nối thành công tới máy chủ.
* ngatKetNoi() : kích hoạt khi thoát kết nối với máy chủ.
* loiSocket(QAbstractSocket::SocketError loi) : kích hoạt khi có vấn đề xảy ra trong mạng gây là lỗi kết nối.

Lớp của chúng ta cũng có 2 thuộc tính.

* QTcpSocket \*socket : là 1 socket đại diện cho cổng kết nối tới máy chủ. Chúng ta dùng đến nó khi muốn gửi tin từ máy người dùng tới máy chủ.
* quint16 kichThuoc : cho phép ghi nhớ kích thước của tin nhắn đang nhận. Thuộc tính này tương tự như thuộc tính cùng tên trong dự án máy chủ.

**CuaSoNguoiDung.cpp**

Bây giờ chúng ta phải lo mã xử lý cho tất cả những phương thức nêu ở trên. Cố gắng thêm chút nữa là có thể hưởng thụ thành phẩm ứng dụng tán gẫu của chính chúng ta rồi.

**Phương thức khởi tạo**

Phương thức khởi tạo của chúng ta cần sử dụng đến phương thức setupUi() của *Ui::CuaSoNguoiDung* để thiết lập việc sử dụng giao diện tạo ra nhờ Qt Designer. Đây chính là câu lệnh giúp chúng ta tiết kiệm hàng đống thời gian vốn cần dành để thiết kế giao diện.

1. CuaSoNguoiDung::CuaSoNguoiDung() {
2. setupUi(this);
4. socket = new QTcpSocket(this);
5. connect(socket, SIGNAL(readyRead()), this, SLOT(nhanDuLieu()));
6. connect(socket, SIGNAL(connected()), this, SLOT(ketNoi()));
7. connect(socket, SIGNAL(disconnected()), this, SLOT(ngatKetNoi()));
8. connect(socket, SIGNAL(error(QAbstractSocket::SocketError)), this, SLOT(loiSocket(QAbstractSocket::SocketError)));
10. connect(nutKetNoi, SIGNAL(clicked()), this, SLOT(anNutKetNoi()));
11. connect(nutGuiTin, SIGNAL(clicked()), this, SLOT(anNutGuiTin()));
12. connect(khungSoanThao, SIGNAL(returnPressed()), this, SLOT(anEnterGuiTin()));
14. kichThuoc = 0;
15. }

Ngoài thiết lập sử dụng giao diện với setupUi(), chúng ta cũng cần làm thêm 1 số xử lý :

* Tạo đối tượng *QTcpSocket* đại diện cho kết nối tới máy chủ.
* Liên kết các slot với các tín hiệu kích hoạt tương ứng.
* Đặt giá trị 0 cho thuộc tính *kichThuocTinNhan* để chuẩn bị cho việc nhận tin nhắn mới.

Chú ý là chúng ta không thực hiện kết nối với máy chủ bên trong phương thức khởi tạo. Chúng ta chỉ đơn giản là tạo ra đối tượng đại diện cho kết nối sẽ được thiết lập này. Việc kết nối thực sự chỉ diễn ra khi người dùng ấn nút “Kết nổi”.

**Slot anNutKetNoi()**

Slot này được kích hoạt khi người dùng ấn nút “Kết nối” ở phía trên cửa sổ.

1. // Thiet lap ket noi den may chu
2. void CuaSoNguoiDung::anNutKetNoi() {
3. // Thong bao la ket noi dang duoc thuc hien
4. cuocHoiThoai->append(tr("<em>Đang kết nối...</em>"));
5. nutKetNoi->setEnabled(false);
7. socket->abort();
8. socket->connectToHost(ipMayChu->text(), congMayChu->value()); // Ket noi toi may chu
9. }
10. Đầu tiên, chúng ta sẽ hiển thị 1 thông báo là chúng ta đang cố gắng kết nối trong khung chứa tổng hợp nội dung cuộc hội thoại.
11. Khóa không cho người dùng tiếp tục ấn vào nút “Kết nối” khi đang cố thiết lập 1 kết nối theo yêu cầu từ trước đó.
12. Nếu socket đã kết nối sẵn từ trước tới máy chủ, ngắt kết nối đó nhờ phương thức abort(). Nếu không có kết nối nào tồn tại từ trước thì câu lệnh này sẽ không có hiệu lực. Chúng ta làm vậy để chắc chắn không thể có 2 kết nối đồng thời tới máy chủ.
13. Thực hiện kết nói tới máy chủ sử dụng phương thức connectTohost(). Chúng ta sẽ dùng giá trị địa chỉ IP và cổng nhập vào bởi người dùng.

**Slot anNutGuiTin()**

Slot này được kích hoạt khi chúng ta gửi tin bằng cách ấn vào nút “Gửi tin” ở phía dưới của giao diện.

1. // Gui tin den may chu
2. void CuaSoNguoiDung::anNutGuiTin() {
3. QByteArray goiTin;
4. QDataStream out(&goiTin, QIODevice::WriteOnly);
6. // Chuan bi goi tin de gui di
7. QString tinGuiDi = tr("<strong>") + nickname->text() +tr("</strong> : ") + khungSoanThao ->text();
9. out << (quint16) 0;
10. out << tinGuiDi;
11. out.device()->seek(0);
12. out << (quint16) (goiTin.size() - sizeof(quint16));
14. socket->write(goiTin); // Gui goi tin
16. khungSoanThao ->clear(); // Xoa tin vua gui khoi khung soan thao
17. khungSoanThao ->setFocus();
18. }

Đoạn mã này tương tự như đoạn mã của phương thức guiTinNhanChoMoiNguoi() mà chúng ta đã viết trong dự án máy chủ. Đương nhiên, mục đích của nó cũng là gửi đi tin nhắn trên mạng.

1. Chuẩn bị 1 đối tượng *QByteArray* để chứa tin nhắn mà chúng ta muốn gửi đi.
2. Tạo ra đối tượng *QString* chứa nội dung của tin nhắn. Ở đây, chúng ta trực tiếp đính tên người gửi vào đầu nội dung tin nhắn. Thực ra xử lý này không được hay cho lắm. Lý tưởng nhất là chúng ta cần tách chúng ra thành 2 thành phần để xử lý riêng biệt. Tuy nhiên điều này sẽ khiến đoạn mã ví dụ trong bài này trở nên rắc rối hơn rất nhiều. Vì bản thân nội dung bài học đã khá là phong phú, tôi đề nghị chúng ta sẽ không phức hóa yêu cầu chương trình và đặt ra vấn đề này như 1 hướng đi có thể để cải tiến chương trình dành cho các bạn nào muốn đào sâu thêm.
3. Chúng ta ghi kích thước tin nhắn vào đầu gói tin.
4. Gửi gói tin đến máy chủ thông qua phương thức write() của socket.
5. Tự động xóa tin nhắn vừa gửi khỏi khung soạn thảo và tự động đưa con trỏ trở lại khung này để người dùng có thể ngay lập tức tiếp tục soạn thảo tin nhắn mới.

**Slot anEnterGuiTin()**

Slot này được kích hoạt khi chúng ta ấn phím *Enter* khi đang trong khung soạn thảo tin nhắn. Hành động này có kết quả tương tự như khi chúng ta ấn nút « Gửi tin », thế nên chỉ cần gọi slot mà chúng ta vừa viết bên trên là được.

1. void CuaSoNguoiDung::anEnterGuiTin() {
2. anNutGuiTin();
3. }

Một cách xử lý khác là chúng ta có thể nối 2 tín hiệu khác nhau lên cùng 1 slot trong phương thức khởi tạo của cửa sổ.

**Slot nhanDuLieu()**

Slot ma quỷ làm tốn cả đống thời gian của chúng ta ở bên trên. Nó không có quá nhiều khác biệt so với slot cùng tên trong dự án máy chủ nên tôi sẽ không giải thích lại nữa.

1. void CuaSoNguoiDung::nhanDuLieu() {
3. QDataStream in(socket);
4. if (kichThuoc == 0) {
5. if (socket->bytesAvailable() < (int)sizeof(quint16)) { //Kich thuoc goi tin nho hon kich thuc kieu so nguyen
6. return;
7. }
8. in >> kichThuoc; // Neu nhan duoc kich thuoc tin nhan thi lay ra gia tri do
9. }
11. // Biet kich thuoc, chung ta se kiem tra xem da nhan duoc toan bo tin nhan chua
12. if (socket->bytesAvailable() < kichThuoc) { // Neu chua nhan du tin nhan thi thoat xu ly
13. return;
14. }
16. QString tinNhan;
17. in >> tinNhan;
19. cuocHoiThoai->append(tinNhan);
21. // Dat lai kich thuoc la 0 de cho tin nhan tiep theo
22. kichThuoc = 0;
23. }

Điểm khác biệt duy nhất là chúng ta thêm tin nhắn vào cuối đoạn hội thoại với câu lệnh cuocHoiThoai->append(tinNhan);

**Slot ketNoi()**

Khi chúng ta kết nối thành công tới máy chủ, 1 tín hiệu sẽ phát để kích hoạt slot này.

1. // Slot kich hoat khi ket noi thanh cong
2. void CuaSoNguoiDung::ketNoi() {
3. cuocHoiThoai->append(tr("<em>Kết nối thành công !</em>"));
4. nutKetNoi->setEnabled(true);
5. }

Chúng ta đơn giản là hiển thị 1 thông báo kết nối thành công để thông báo cho người dùng là đã gia nhập mạng.

Chúng ta kích hoạt lại nút “Kết nối” để cho phép người dùng có thể sử dụng khi muốn tạo 1 kết nối tới máy chủ khác.

**Slot ngatKetNoi()**

Slot này hiển nhiên là được kích hoạt khi người dùng thoát khỏi mạng.

1. // Slot kich hoat khi thoat ket noi
2. void CuaSoNguoiDung::ngatKetNoi() {
3. cuocHoiThoai->append(tr("<em>Tạm biệt, hẹn gặp lại sau !</em>"));
4. }

Vẫn đơn giản là 1 thông báo cho người dùng biết là đã thoát kết nối.

**Slot loiSocket()**

Slot này hoạt động khi mà chúng ta gặp lỗi trong quá trình socket liên kết tới hệ thống.

1. // Slot kich hoat khi co loi socket
2. void CuaSoNguoiDung::loiSocket(QAbstractSocket::SocketError loi) {
3. switch(loi) { // Hien thi thong bao khac nhau tuy theo loi gap phai
4. case QAbstractSocket::HostNotFoundError:
5. cuocHoiThoai->append(tr("<em>LỖI : Không thể kết nối tới máy chủ ! Vui lòng kiểm tra lại địa chỉ IP và cổng truy cập.</em>"));
6. break;
7. case QAbstractSocket::ConnectionRefusedError:
8. cuocHoiThoai->append(tr("<em>LỖI : Máy chủ từ chối truy cập ! Vui lòng kiểm tra chắc chắn là máy chủ đã được khởi động. Lưu ý đồng thời lỗi địa chỉ IP và cổng truy cập.</em>"));
9. break;
10. case QAbstractSocket::RemoteHostClosedError:
11. cuocHoiThoai->append(tr("<em>LỖI : Máy chủ đã ngắt kết nối !</em>"));
12. break;
13. default:
14. cuocHoiThoai->append(tr("<em>LỖI : ") + socket->errorString() + tr("</em>"));
15. }
17. nutKetNoi->setEnabled(true);
18. }

Lỗi gặp phải sẽ được truyền như tham số của slot. Nó thuộc kiểu *QAbstractSocket::SocketError* vốn là 1 danh sách liệt kê.

Chúng ta sử dụng lệnh switch để xử lý các lỗi khác nhau có thể gặp phải. Ở đây, tôi không xử lý tất cả các lỗi có thể xảy đến với socket. Để biết thêm về các lỗi có thể xảy ra, hãy tham khảo trong tài liệu Qt.

Phần lớn các lỗi có liên quan đến kết nối tới máy chủ. Tôi sử dụng các thông báo tự tạo để người dùng dễ hiểu hơn về nguyên nhân lỗi.

Trường hợp default của lệnh switch dùng để xử lý tất cả các lỗi khác chưa được liệt kê ra ở trên. Trong trường hợp đó, chúng ta hiển thị trực tiếp cho người dùng thông báo lỗi tiếng Anh gửi đến bới socket, dù sao còn hơn là không có gì.

**CuaSoNguoiDung.cpp hoàn chỉnh**

Xong rồi ! Phải công nhận là sau khi chiến đấu với dự án máy chủ thì dự án trên máy người dùng có vẻ đơn giản đi nhiểu nhỉ ?

1. #include "CuaSoNguoiDung.h"
3. CuaSoNguoiDung::CuaSoNguoiDung() {
4. setupUi(this);
6. socket = new QTcpSocket(this);
7. connect(socket, SIGNAL(readyRead()), this, SLOT(nhanDuLieu()));
8. connect(socket, SIGNAL(connected()), this, SLOT(ketNoi()));
9. connect(socket, SIGNAL(disconnected()), this, SLOT(ngatKetNoi()));
10. connect(socket, SIGNAL(error(QAbstractSocket::SocketError)), this, SLOT(loiSocket(QAbstractSocket::SocketError)));
12. connect(nutKetNoi, SIGNAL(clicked()), this, SLOT(anNutKetNoi()));
13. connect(nutGuiTin, SIGNAL(clicked()), this, SLOT(anNutGuiTin()));
14. connect(khungSoanThao, SIGNAL(returnPressed()), this, SLOT(anEnterGuiTin()));
16. kichThuoc = 0;
17. }
19. // Thiet lap ket noi den may chu
20. void CuaSoNguoiDung::anNutKetNoi() {
21. // Thong bao la ket noi dang duoc thuc hien
22. cuocHoiThoai->append(tr("<em>Đang kết nối...</em>"));
23. nutKetNoi->setEnabled(false);
25. socket->abort();
26. socket->connectToHost(ipMayChu->text(), congMayChu->value()); // Ket noi toi may chu
27. }
29. // Gui tin den may chu
30. void CuaSoNguoiDung::anNutGuiTin() {
31. QByteArray goiTin;
32. QDataStream out(&goiTin, QIODevice::WriteOnly);
34. // Chuan bi goi tin de gui di
35. QString tinGuiDi = tr("<strong>") + nickname->text() +tr("</strong> : ") + khungSoanThao ->text();
37. out << (quint16) 0;
38. out << tinGuiDi;
39. out.device()->seek(0);
40. out << (quint16) (goiTin.size() - sizeof(quint16));
42. socket->write(goiTin); // Gui goi tin
44. khungSoanThao ->clear(); // Xoa tin vua gui khoi khung soan thao
45. khungSoanThao ->setFocus();
46. }
48. void CuaSoNguoiDung::anEnterGuiTin() {
49. anNutGuiTin();
50. }
52. void CuaSoNguoiDung::nhanDuLieu() {
53. QDataStream in(socket);
54. if (kichThuoc == 0) {
55. if (socket->bytesAvailable() < (int)sizeof(quint16)) { //Kich thuoc goi tin nho hon kich thuc kieu so nguyen
56. return;
57. }
58. in >> kichThuoc; // Neu nhan duoc kich thuoc tin nhan thi lay ra gia tri do
59. }
60. // Biet kich thuoc, chung ta se kiem tra xem da nhan duoc toan bo tin nhan chua
61. if (socket->bytesAvailable() < kichThuoc) { // Neu chua nhan du tin nhan thi thoat xu ly
62. return;
63. }
64. QString tinNhan;
65. in >> tinNhan;
67. cuocHoiThoai->append(tinNhan);
69. // Dat lai kich thuoc la 0 de cho tin nhan tiep theo
70. kichThuoc = 0;
71. }
73. // Slot kich hoat khi ket noi thanh cong
74. void CuaSoNguoiDung::ketNoi() {
75. cuocHoiThoai->append(tr("<em>Kết nối thành công !</em>"));
76. nutKetNoi->setEnabled(true);
77. }
79. // Slot kich hoat khi thoat ket noi
80. void CuaSoNguoiDung::ngatKetNoi() {
81. cuocHoiThoai->append(tr("<em>Tạm biệt, hẹn gặp lại sau !</em>"));
82. }
84. // Slot kich hoat khi co loi socket
85. void CuaSoNguoiDung::loiSocket(QAbstractSocket::SocketError loi) {
86. switch(loi) { // Hien thi thong bao khac nhau tuy theo loi gap phai
87. case QAbstractSocket::HostNotFoundError:
88. cuocHoiThoai->append(tr("<em>LỖI : Không thể kết nối tới máy chủ ! Vui lòng kiểm tra lại địa chỉ IP và cổng truy cập.</em>"));
89. break;
90. case QAbstractSocket::ConnectionRefusedError:
91. cuocHoiThoai->append(tr("<em>LỖI : Máy chủ từ chối truy cập ! Vui lòng kiểm tra chắc chắn là máy chủ đã được khởi động. Lưu ý đồng thời lỗi địa chỉ IP và cổng truy cập.</em>"));
92. break;
93. case QAbstractSocket::RemoteHostClosedError:
94. cuocHoiThoai->append(tr("<em>LỖI : Máy chủ đã ngắt kết nối !</em>"));
95. break;
96. default:
97. cuocHoiThoai->append(tr("<em>LỖI : ") + socket->errorString() + tr("</em>"));
98. }
99. nutKetNoi->setEnabled(true);
100. }

**Thử nghiệm Aloo! và phương hướng cải tiển**

Thế là cả chương trình trên máy chủ lẫn máy người dùng đều đã hoàn thành.

Đầu tiên thì chúng ta hãy tận hưởng thành quả ứng dụng tán gẫu Aloo! mà chúng ta đã mất bao công sức để xây dựng. Tiếp theo đó chúng ta sẽ tìm ra 1 vài hướng cải tiến khiến chương trình trở nên hoàn chỉnh hơn.

**Thử nghiệm Aloo!**

Trước hết, các bạn có thể tải mã nguồn của các dự án mà chúng ta đã viết.

Các bạn có thể kích hoạt trực tiếp các tệp thực thi nếu đang sử dụng Windows (chú ý đừng quên thêm các thư viện động DLL cần thiết). Nếu các bạn sử dụng hệ điều hành khác thì để chạy chương trình, chúng ta sẽ cần biên dịch lại với qmake và make.

Thử nghiệm bước đầu sẽ là tán gẫu giữa nhiều cửa sổ cùng trên máy tính cá nhân của bạn. Chúng ta sẽ thực thi 1 cửa sổ máy chủ và 2 cửa sổ người dùng.

Chỉ cần thế là chúng ta có thể giả lập 1 cuộc hội thoại chỉ bằng cách sử dụng 1 máy tính. Mạng được sử dụng trong trường hợp này chính là mạng nội tại của máy tính của chúng ta.

Và đây là kết quả.

Phù, may là nó hoạt động thật :-SS.

Bước tiếp theo, chúng ta có thể thoải mái thử nghiệm trong mạng nội bộ hoặc Internet với người thân và bạn bè của mình. Cần chú ý là cổng truy cập yêu cầu trên các thiết bị cần được mở vì nếu có vấn đề với chương trình thì 99,99% đấy là nguyên nhân chính.

Các bạn cũng có thể thử với mạng Internet. Thật ra, chúng ta không cần thử nghiệm với các mạng khác nhau vì nguyên lý hoạt động của chúng không có gì khác biệt, dù là mạng nội bộ hay mạng Internet. Chỉ đơn giản là chúng sử dụng các địa chỉ IP khác nhau.

**Các ý tưởng cải tiến**

Tôi tự cảm thấy mình cũng nỗ lực khá nhiều trong bài thực hành này rồi. Giờ mình nhường sấn khấu cho các bạn tỏa sáng chút đỉnh cho đỡ tự kỷ.

Dưới đây là 1 vài ý tưởng, từ dễ tới khó, theo mình có thể dùng để hoàn thiện chương trình tán gẫu siêu cấp Aloo!.

* Vô hiệu hóa khung soạn thảo cũng như nút gửi tin chừng nào mà người dùng còn chưa kết nối vào mạng.
* Trên cửa sổ của chương trình máy chủ, hiển thị số người dùng đang truy cập tới chương trình.
* Sắp xếp và tổ chức lại mã nguồn. Thay vì sử dụng danh sách đối tượng *QTcpSocket* thì sử dụng 1 danh sách đối tượng lớp *NguoiDung*. Đây là 1 lớp mới dùng để quản lý tất cả thông tin người dùng như*QTcpSocket* đại diện cho người dùng đó, nickname mà họ muốn dùng khi tán gẫu, ảnh đại diện, vv… Quyền quyết định nằm ở chính các bạn !
* Nhờ thay đổi ngay bên trên, chúng ta có thể hiển thị rõ ràng tên của người dùng vừa đăng nhập thay vì sử dụng 1 thông báo chung chung như hiện tại.
* Thêm vào 1 số khả năng cho phép người dùng chỉnh sửa kiểu chữ, màu chữ, vv… Thế rồi có thể thêm các biểu tượng nho nhỏ kiểu như Yahoo!Messenger vậy. Tuy nhiên đừng quá sa đà vào đây, dù sao thì 1 người cũng không thể xây dựng được 1 chương trình nhiều tính năng được như Yahoo.
* Quản lý danh sách người dùng đang đăng nhập. Tính năng này cũng khá tinh tế và phức tạp. Các bạn sẽ cần dùng đến kiến thức mà chúng ta đã thảo luận về mô hình MVC.
* Cải tiến chương trình máy chủ cho phép 2 người đồng thời gửi tin nhắn bởi chương trình của chúng ta hiện nay chỉ tiếp nhận được 1 tin nhắn 1 lúc (vì chỉ có 1 thuộc tính *kichThuoc*). Ý tưởng là sẽ có 1 thuộc tính cho mỗi người dùng đang đăng nhập để cho phép nhận nhiều tin nhắn cùng lúc.

Thế đã là cả đống công việc rồi nhỉ !

Luôn luôn ghi nhớ là đừng bao giờ để ý tưởng của các bạn bó hẹp trong những gì đã học. Dám nghĩ và dám thử !

Không có gì ngăn cản chúng ta viết 1 chương trình tán gẫu nhưng không phải theo dạng phòng nhiều người mà kiểu tin nhắn riêng như của Yahoo.

Mạng cũng không phải chỉ dùng để tán gẫu, chúng ta có thể tạo ra những trò chơi đơn giản. Lúc đấy thì gói tin chuyển đi trên mạng sẽ không chỉ đơn giản là tin nhắn kiểu văn bản nữa mà có thể là hình ảnh, mệnh lệnh hành động cho nhân vật, vv… Các bạn chỉ cần chỉnh sửa đoạn mã chúng ta đã cùng viết cho phù hợp hoàn cảnh là được.

Chắc tôi sẽ dùng bài học này ở đây. Tôi tin rằng đây là bài học dài nhất mà tôi từng viết rồi !

Hy vọng là các bạn hài lòng với chương giáo trình về Qt. Dù chúng ta không thể thảo luận về tật cả mọi thứ nhưng cũng đủ nhiều điều mới mẻ rồi.

Giờ đến lượt các bạn tự mình chiến đầu với mạng, widget, đồ họa, vv… Và đừng quên rằng tài liệu Qt luôn ở bên cạnh các bạn, cũng như « Lập trình Tân Binh » vậy !

[3.13. Kiến trúc MVC và các widget phức hợp](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/55/3.13.-Kien-truc-MVC-va-cac-widget-phuc-hop)

**Chia sẻ :**

Top of Form

Bottom of Form

[Lập trình với ngôn ngữ C++](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C)

* [Chương 1 [lý thuyết] Giới thiệu về Lập trình C++](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/3/1.1.-C-la-gi)
* [Chương 2 [lý thuyết] Lập trình hướng đối tượng](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/15/2.1.-Gioi-thieu-ven-man-bi-an-cua-kieu-du-lieu-string)
* [**Chương 3 [thực tiễn] Tạo ứng dụng cửa sổ với Qt**](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/25/3.1.-Gioi-thieu-ve-thu-vien-Qt)
  + [3.1. Giới thiệu về thư viện Qt](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/25/3.1.-Gioi-thieu-ve-thu-vien-Qt)
  + [3.2. Biên dịch chương trình cửa sổ đầu tiên](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/26/3.2.-Bien-dich-chuong-trinh-cua-so-dau-tien)
  + [3.3. Tùy chỉnh các widget](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/27/3.3.-Tuy-chinh-cac-widget)
  + [3.4. Các tín hiệu và các slot](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/28/3.4.-Cac-tin-hieu-va-cac-slot)
  + [3.5. Các hộp thoại thông dụng](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/29/3.5.-Cac-hop-thoai-thong-dung)
  + [3.6. Cách sử dụng tài liệu Qt](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/30/3.6.-Cach-su-dung-tai-lieu-Qt)
  + [3.7. Sắp xếp bố cục các widget](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/31/3.7.-Sap-xep-bo-cuc-cac-widget)
  + [3.8. Điểm danh những widget thông dụng](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/45/3.8.-Diem-danh-nhung-widget-thong-dung)
  + [3.9. [Thực hành] KienTrucSuX](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/46/3.9.-Thuc-hanh-KienTrucSuX)
  + [3.10. Cửa sổ chính](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/48/3.10.-Cua-so-chinh)
  + [3.11. Hỗ trợ đa ngôn ngữ nhờ Qt Linguist](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/51/3.11.-Ho-tro-da-ngon-ngu-nho-Qt-Linguist)
  + [3.12. [Thực hành] BanDoX, trình duyệt web của Tân binh](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/54/3.12.-Thuc-hanh-BanDoX-trinh-duyet-web-cua-Tan-binh)
  + [3.13. Kiến trúc MVC và các widget phức hợp](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/55/3.13.-Kien-truc-MVC-va-cac-widget-phuc-hop)
  + [**3.14. Giao tiếp qua mạng**](http://laptrinhtanbinh.com/course/1/Lap-trinh-voi-ngon-ngu-C/57/3.14.-Giao-tiep-qua-mang)

**Giữ liên lạc với bộ chỉ huy để nâng cao kỹ năng chiến đấu**

Top of Form

Bottom of Form

**Chúng ta nói về chúng tôi**