نام و نام خانوادگی: پریسا یحیی‌پور فتیده
شماره دانشجویی: 810101551
گزارش پروژه 1 درس سیستم عامل

موجودیت‌های برنامه

به طور کلی این برنامه سه موجودیت دارد که عبارتند از: مسئول برگزاری، اتاق‌ها و بازیکنان. برای مشخص کردن نقش هر موجود باید توجه کنیم که درخواست اتصال توسط چه کسی ارسال شده و چه کسی در حال ارائه خدمت است.

مسئول برگزاری

با توجه به مسئولیت‌های این موجود در برنامه می‌توان اشاره کرد به طور کلی مسئول برگزاری سرور اصلی برنامه است. بازیکنان درخواست اتصال خود را به این موجود ارسال می‌کنند و مسئول برگزاری خدماتی ارائه می‌دهد از جمله:

1. فرستادن اطلاعات موجود که خود نیازمند جستجو و آماده‌سازی پیام برای ارسال به سمت کاربر است
2. دادن اطلاعات لازم برای اتصال بازیکن به اتاق از جمله شماره port که اتاق دارد (با توجه به اینکه IP در کل برنامه ثابت است و همان IP که به عنوان ورودی به تابع main به server.out و client.out دادیم)
3. بررسی خالی بودن یا نبودن اتاق انتخابی کاربر
4. خاتمه بازی و ارسال گزارش کلی به شکل عمومی

البته می‌توان در حالتی که اعلان‌های عمومی از سمت اتاق ارسال می‌شوند، مسئول برگزاری را یک client دانست. اما به طور کلی سرور در نظر می‌گیریم.

اتاق

با توجه به مسئولیت‌های این موجود می‌توان آن را یک زیرمجموعه‌ای از سرور[1] دانست. وظیفه اصلی اتاق، ارائه خدمت است. خدماتی که اتاق ارائه می‌دهد عبارتند از:

1. گرفتن انتخاب بازیکنان
2. داوری بین انتخاب بازیکنان
3. ارسال اعلان شروع بازی
4. ارسال اعلان خاتمه یک بازی به شکل عمومی

بازیکن

با توجه به ماهیت کارهایی که در طول برنامه انجام می‌دهد، آن را client در نظر می‌گیریم. چرا که همواره در حال ارائه اطلاعات یا ارسال درخواست یا اتصال به سرورها است. درخواست‌هایی که کاربر ارسال می‌کند شامل موارد زیر می‌شود:

1. ارسال درخواست اتصال به سرور اصلی
2. ارسال درخواست برای گرفتن اطلاعات اتاق‌های موجود
3. ارسال درخواست اتصال به اتاق
4. دریافت اعلان‌های عمومی سرورها
5. ارسال اطلاعات مورد نیاز جهت گرفتن سرویس درخواستی (شماره اتاق درخواستی، نام یا هویت، انتخاب در بازی و درخواست شرکت مجدد در بازی)

اتصالات

به طور کلی با توجه به ماهیت و حساسیت اطلاعات ارسالی می‌توان نوع اتصالات میان موجودیت‌های برنامه را تعیین کرد که در ادامه با جزئیات مشخص می‌شوند.

[1] Subserver

بازیکن - مسئول برگزاری

با توجه به اینکه اطلاعات ارسال شده به سرور اصلی باید از سایر کاربرها مخفی باشد، اتصال بین این دو باید از نوع Connection-Oriented باشد. همچنین چون ارتباط بین دو موجودیت رخ می‌دهد و اطلاعات نباید به بقیه برسد، Unicast است. برای این اتصال از پروتکل TCP استفاده می‌کنیم. یکی دیگر از اتصالاتی که آن را بین این دو در نظر می‌گیریم، Connectionless و Broadcast است تا بازیکن اعلان‌های عمومی مسئول برگزاری را دریافت کند.

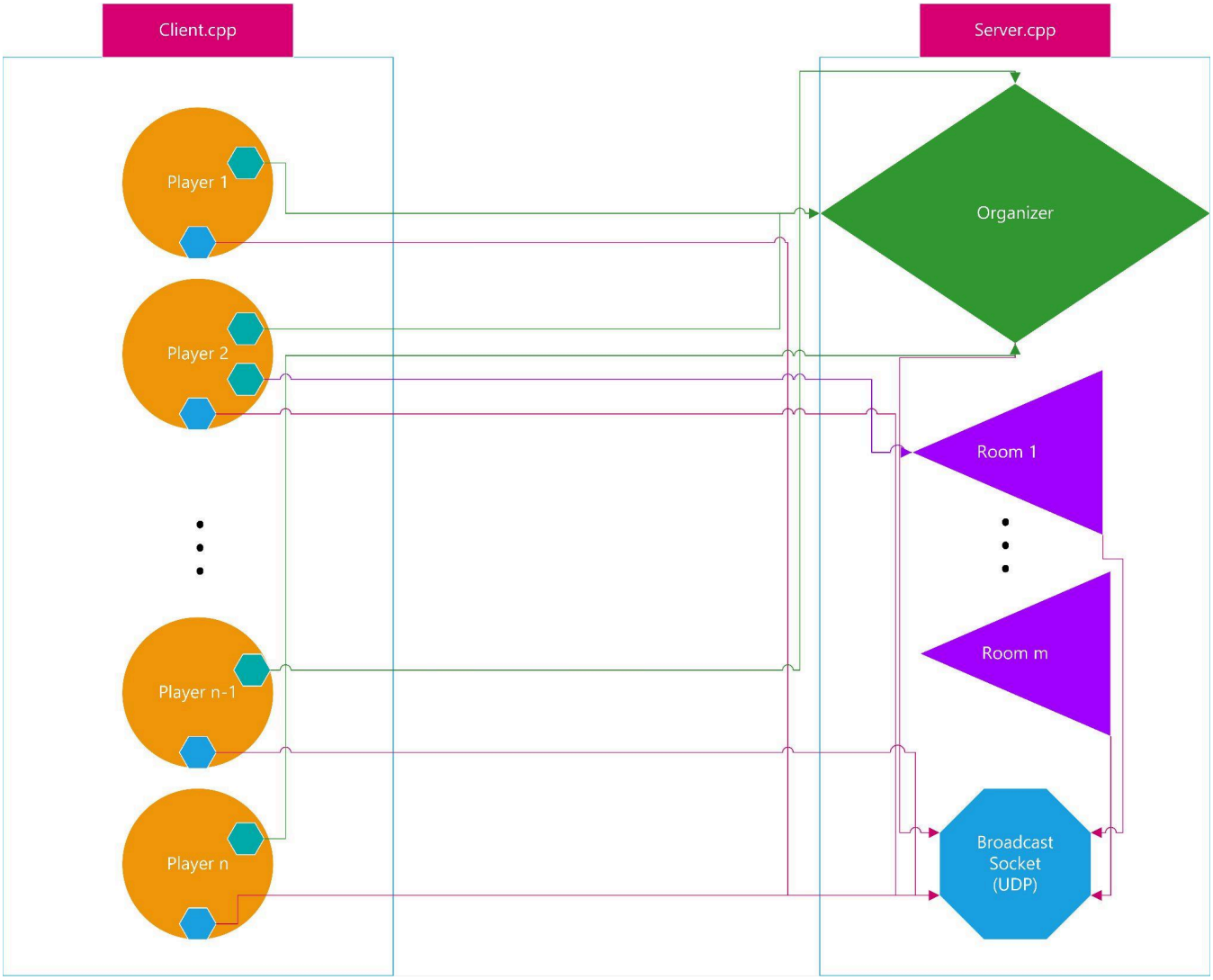
بازیکن - اتاق

دو نوع اتصال می‌توان بین بازیکن و اتاق قائل شد. برای ارسال انتخاب بازیکن با توجه به اینکه باید مخفی باشد، از اتصال Connection-Oriented و Unicast باید استفاده شود مانند TCP. اما برای دریافت اعلان‌های عمومی باید بین بازیکن و اتاق یک اتصال Connectionless و Broadcast ایجاد شود که می‌توان از پروتکل UDP استفاده کرد. چرا که این اطلاعات باید به دست همه برسد.

اتاق - مسئول برگزاری

با توجه به اینکه این دو موجودیت برنامه در یک پردازه هستند و از هم جدا نیستند، برای تبادل اطلاعات نیازی به استفاده از سوکت و اتصال نیست. در واقع برای تبادل اطلاعات به جای استفاده از روش message passing از روش shared memory استفاده می‌کنیم. اگر چه برای دریافت اعلان‌هایی که از مسئول برگزاری یا اتاق به شکل عمومی ارسال شده‌اند باید بین آنها یک اتصال Connectionless و Broadcast ایجاد شود که پروتکل آن UDP است.

نقشه طراحی اتصالات



به طور کلی اتصالات این‌گونه انجام می‌شود:

به ازای ورود هر کاربر یا در واقع اجرای برنامه client.out، ابتدا یک socket برای دریافت broadcast و یک socket برای اتصال به سرور اصلی ایجاد می‌شوند. همان طور که اشاره شد برای broadcast ارتباط Connectionless-Broadcast داریم که در نقشه بالا همان 8 ضلعی آبی موجود در هر بازیکن است. تمام این socketها به IP مربوط به broadcast که 127.255.255.255 است متصل می‌شوند تا در صورت دریافت اعلان عمومی اقدام لازم را انجام دهند. برای سرور اصلی اتصال Connection Oriented-Unicast داریم که 8 ضلعی

سبز-آبی موجود در بازیکن است. اتصالات صورتی رنگ همان Connectionless-Broadcast یا UDP هستند و اتصالات سبز رنگ Connection Oriented- Unicast یا TCP هستند (برای سرور اصلی). پس از اتصال به سرور اصلی و دریافت لیست اتاق‌های موجود، کاربر از طریق خطوط سبز رنگ در نقشه بالا شماره اتاق درخواستی را به سرور اصلی ارسال می‌کند و سرور اصلی در صورتی که در این فاصله اتاق پر نشده باشد، شماره port اتاق درخواستی را از طریق همان خط سبز به بازیکن ارسال می‌کند و بازیکن حالا می‌تواند به اتاق مورد نظر متصل شود که از طریق یکی دیگر از اتصالات Connection Oriented-Unicast با انجام می‌شود که دومین 8 ضلعی سبز آبی موجود است. توجه کنیم در صورتی که بازیکن به سرور متصل شود اما درخواست اتصال به هیچ اتاقی را ارسال نکند، دومین سوکت TCP را نخواهد داشت مانند Player 1، Player n-1 و Player n در نقشه بالا که فقط یک 8 ضلعی سبز-آبی برای اتصال به سرور اصلی دارند.

در ابتدای برنامه که server.out را با IP و port و تعداد اتاق‌های درخواستی اجرا می‌کنیم، ابتدا یک socket برای Broadcast ایجاد می‌شود که همان 8 ضلعی آبی است (سمت سرور). تمام ارسال اعلان‌های عمومی چه از طریق اتاق و چه از طریق سرور توسط همین socket و ارتباط ایجاد می‌شود. همچنین یک socket برای سرور اصلی ایجاد می‌کنیم و ارتباط آن به شکل Connection Oriented-Unicast یا TCP است که روی آن در حال گوش دادن (listen) است تا بازیکنان بیایند و به آن متصل شوند. سپس به تعداد اتاق‌های درخواستی socket برای اتصال Connection Oriented-Unicast یا TCP ایجاد می‌شود و در حال گوش دادن است تا بازیکنان پس از دریافت port آنها از سرور اصلی به آنها متصل شوند. همچنین اتاق‌ها به دلیل اینکه با سرور اصلی در یک پردازه هستند، از همان socket-ای که در ابتدا برای Connectionless-Broadcast یا UDP ایجاد کردیم، برای ارسال نتایج بازی به تمام کاربران از جمله سرور اصلی از آن استفاده کنند.

ساختار داده‌ها

برای نگه‌داری اطلاعات اتاق‌ها از جمله شماره port، تعداد بازیکنان متصل، socket-ای که روی آن listen می‌کند و ...، یک Struct ایجاد می‌کنیم و vector-ای از آن را در همان کد مربوط به سرور نگه‌داری می‌کنیم. برای ذخیره اطلاعات بازیکنان، از آنجایی که بعد از ارسال اطلاعات

هویتی کاربر، سرور اصلی صاحب آنهاست، یک vector از Struct مربوط به بازیکنان در کد سرور می‌سازیم که اطلاعات بازیکن از جمله نام، file descriptor و ... را شامل می‌شود.
با توجه به این دو vector به راحتی می‌توان با توابع جستجو مختلف، اطلاعات مورد نظر را چه برای استفاده اتاق و چه برای استفاده سرور اصلی پیدا کرد.

نکات نهایی

فرض‌هایی که انجام شده شامل موارد زیر می‌شوند:

- برای تشخیص روند پاسخ به اطلاعات دریافتی، در هر نوع از پیام‌های ارسالی، کلیدواژه[2] متناسب با آن پیام وجود دارد تا دریافت کننده با دیدن این کلیدواژه عملیات مورد نیاز را انجام دهد. برای مثال در ابتدا که برنامه کاربر را اجرا می‌کنیم، برای ارسال نام از سمت بازیکن به سمت سرور از کلیدواژه name استفاده کردیم. سرور اصلی با دیدن این کلمه در پیام دریافتی متوجه می‌شود باید نام کاربر متناظر را ذخیره کند.
- در ورودی‌هایی که نیاز به وارد کردن کلیدواژه داریم، در پیام ارسال شده از سمت سرور یا اتاق، قالب[3] پیام هم نمایش داده می‌شود.
- برای اینکه broadcast-ها به درستی ارسال شوند و در واقع اتفاقات برنامه خیلی سریع رخ ندهد، بعد از توابع sendto برای چند میلی‌ثانیه از تابع sleep استفاده شده است.
- در صورتی که پیام‌های وارد شده در ورودی استاندارد[4] شامل هیچ کدام از کلیدواژه‌های تعریف شده نباشد، هیچ اتفاقی رخ نمی‌دهد.
- سایز buffer برای ارسال و دریافت اطلاعات 1024 در نظر گرفته شده است.
- فرض شده است در برنامه client هیچ وقت ctrl+c زده نمی‌شود.
- با وارد کردن دستور end_game در برنامه سرور تمام برنامه‌های در حال اجرا در terminal-های مختلف بعد از دریافت اعلان عمومی که همان نتایج بازی است، به طور موفقیت آمیز terminate می‌شوند. برنامه خود سرور پس از کمی مکث terminate می‌شود.

[2] Keyword
[3] Format
[4] STDIN