به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

شبکه‌های کامپیوتری

پروژه‌ی سوم(P2P)

نام و نام خانوادگی	عارف افضلی امین محقق
شماره دانشجویی	810896042 810195579
تاریخ ارسال گزارش	

فهرست گزارش سوالات

مقدمه

مفهوم شبکه همتا به همتا

به بیان ساده سیستم همتا به همتا شبکه ای متشکل از کامپیوترها است که از ساختاری توزیع شده تبعیت می‌کنند و از طریق اینترنت به یکدیگر متصل هستند. فایل ها بدون نیاز به سرور مرکزی بین این کامپیوترها به اشتراک گذاشته می‌شود. در واقع هر کامپیوتر در شبکه همتا به همتا بطور همزمان هم یک سرور و هم کاربر است. یک کامپیوتر برای پیوستن به یک شبکه همتا به همتا نیاز به اینترنت و نرم افزاری برای اتصال به این شبکه دارد. پس از اتصال، شبکه به شما امکان جست و جو در فایل های کامپیوتر اشخاص دیگر را میدهد و به همین ترتیب سایر افراد می‌توانند فایلهای موجود روی کامپیوتر شما را جست و جو کنند. در واقع کاربران به تمام فایل های کامپیوتر شما دسترسی ندارند؛ بلکه فقط به فایل هایی که شما در پوشه ای مخصوص در کامپیوتر خود قرار داده اید و آن را به اشتراک گذاشته اید، دسترسی خواهند داشت.

شبکهP2P چگونه کار می‌کند؟

در واقع، یک سیستم همتا به همتا توسط شبکه ای توزیع شده از کاربران نگه داری می شود. معمولا این کاربران دارای مدیریت یا سرور مرکزی نیستند، چرا که هر نود یک نسخه از فایل ها را نگه داری کرده، و در نقش مشتری و همچنین در نقش سرور برای نودهای دیگر ظاهر می شود. بنابراین هر نود می توان فایل ها را از نودهای دیگر دانلود کرده و یا برای نودهای دیگر آپلود کند. همین مورد شبکه همتا به همتا را از سیستم های سرور-مشتری سنتی متفاوت می سازد، که در آن ها دستگاه های مشتریان باید فایل ها را از یک سرور متمرکز دانلود نماید.

مروری بر پروژه

در این پروژه قصدمان پیاده‌سازی یک شبکه همتا به همتا است. در این شبکه پیچیدگی اشتراک‌گذاری فایل‌ها را نداریم. همسایه بودن دو نود را به این شرط می‌دانیم که وصل بودن دو نود به صورت دوطرفه باشد. در شبکه ما، هر نود باید ببیند چه نودهایی با آن همسایه هستند و این کار توسط ارسال یک بسته HELLO انجام می‌گیرد. این بسته‌ها توسط هر نود به بقیه نودها هر ۲ ثانیه ارسال می‌شود و اگر نودی از یکی از همسایگان خود برای ۸ ثانیه بسته‌ای دریافت نکند، باید ارتباط خود را با آن نود قطع کند و به دنبال یک نود دیگر باشد برای جایگزینی این نود که این امر باید به صورت تصادفی صورت گیرد.

برای این شبکه از پروتکل udp استفاده شده است و همچنین نرخ اتلاف بسته ۵ درصد درنظر گرفته شده است. هر ۱۰ ثانیه یکبار یکی از نودها به صورت تصادفی به مدت ۲۰ ثانیه خاموش می‌شود.

در انتها به ازای هر نود یک فایل JSON خروجی داده می‌شود که شامل ۱. لیست تمام همسایگان از اول تا پایان زمان به صورت IP و پورت به همراه تعداد بسته‌های دریافتی و ارسالی از آن‌ها ۲. لیست همسایگان کنونی ۳. درصد زمانی که دو نود با هم همسایه بودند نسبت به کل زمان اجرا ۴. ساختار توپولوژی شبکه از دید آن نود.

یک نمونه فایل خروجی برای نود صفر در زیر آمده است:

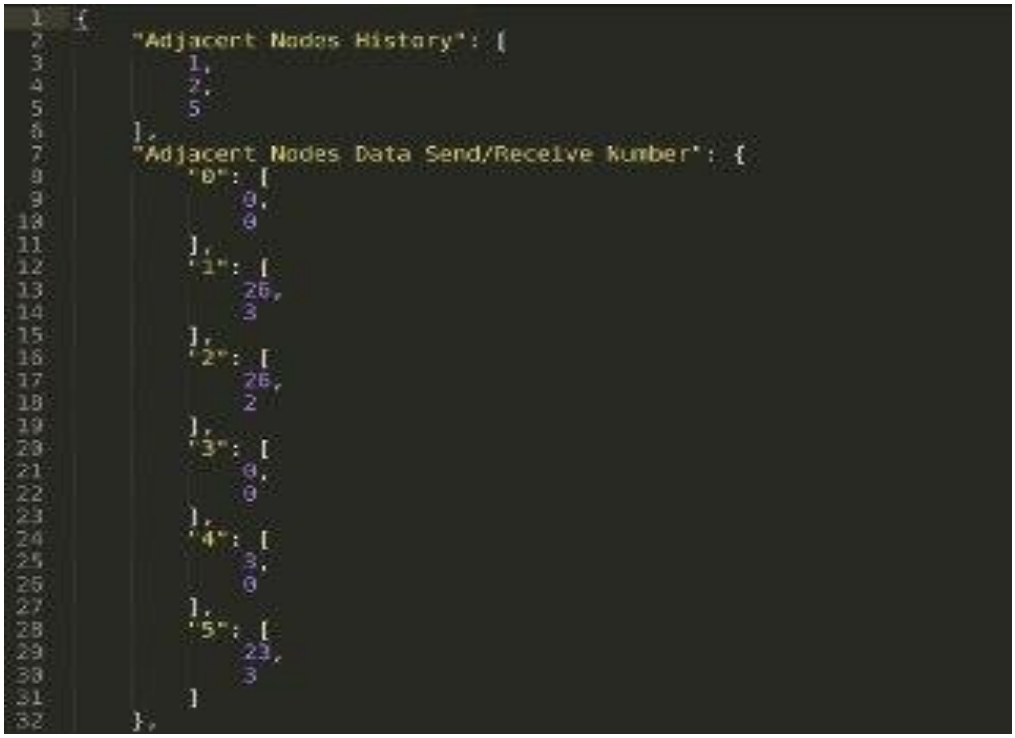

Figure 1 JSON File Output Part 1

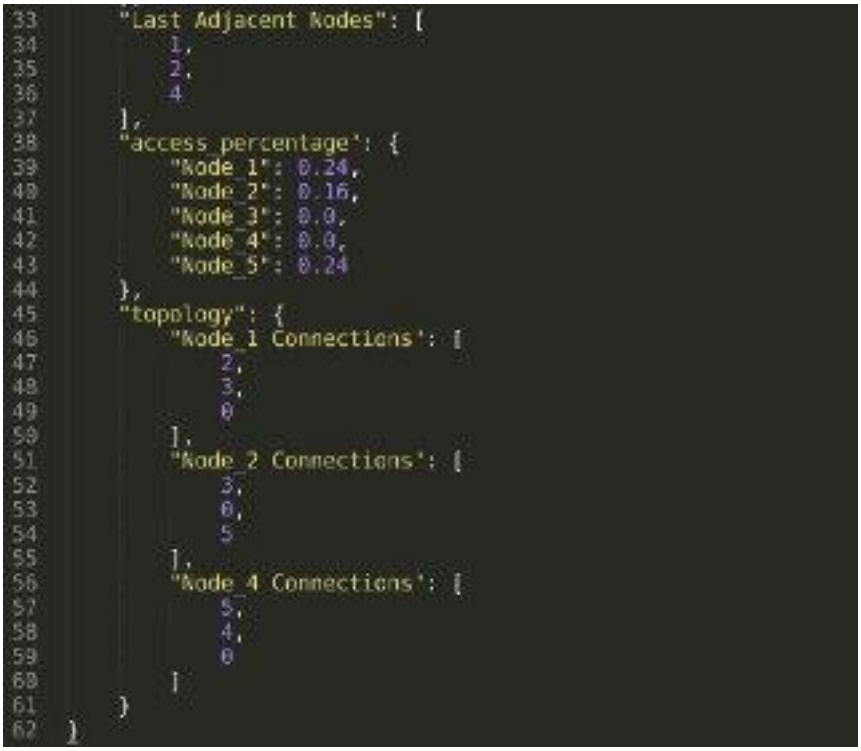

Figure 2 JSON File Output Part 2

بسته‌های HELLO ذکر شده باید شامل موارد روبرو باشند: ۱. شناسه گره فرستنده ۲. آدرس IP فرستنده ۳. آدرس پورت فرستنده ۴. نوع پیغام ۵. لیست همسایگان فرستنده ۶. زمان آخرین بسته ارسال شده از فرستنده به گیرنده ۷. زمان آخرین بسته دریافت شده از گیرنده به فرستنده.

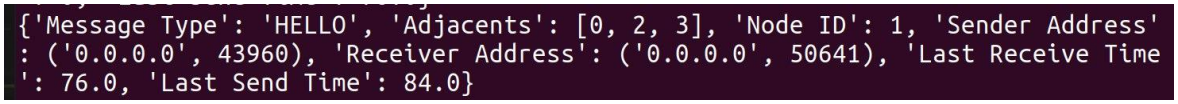

Figure 3 HELLO Package

برنامه

نحوه اجرای برنامه

کد این پروژه شامل یک فایل پایتون به اسم main.py است که می‌توانید با اجرای ساده این فایل پایتون، ورودی‌های مورد نیاز را با توجه به درخواست گفته شده، بدهید یا می‌توانید به صورت آرگومان پارامترها را پاس بدهید که به ترتیب باید آرگومان‌های تعداد نودها و تعداد اتصالات به هر نود را ورودی بدهید و در این حالت زمان اجرا به صورت پیش‌فرض ۱۰۰ ثانیه در نظر گرفته می‌شود.

همچنین برای افزایش سرعت پردازش از کتابخانه‌ی multiprocessing پایتون استفاده شد.

توضیح کد

در ابتدا تابع initial_net صدا زده می‌شود که با توجه به ارگومان‌هایی که از کاربر میگیرد(تعداد Nodeها و تعداد Connectionها برای هر Node و همچنین مدت زمان simulation) Nodeها را درست میکند و پس از initial کردن برخی مقادیر برای Node تابع Run تمام Nodeها را صدا میزند. سپس تابع initial net وارد یک حلقه می‌شود که هر 10 ثانیه یکبار شماره‌ی یک گره را به تصادف انتخاب کرده و به آن گره پیغام می‌فرستد تا به مدت ۲۰ ثانیه هیچ پیغام hello به گره‌های دیگر نفرستد.

تابع run مربوط به هر گره به این صورت عمل میکند که ابتدا دو thread ایجاد می‌کند که وظیفه‌ی thread اول انجام دادن تابع do_stuff است که اگر نیاز به تبادل داده‌ای بین گره‌های همسایه بود این تابع ارتباط TCP را برقرار میکند.(که البته در این پروژه نیازی به این نبود و این تابع تنها تا پایان مدت شبیه‌سازی صبر میکند.)

وظیفه‌ی Thread دوم اجرا کردن تابع hello_process است. این تابع هر دو ثانیه یک بار تابع send_hollo را صدا می‌زند که به نودهای در لیست adj_list پیغام hello میفرستد(با استفاده از UDP). و هر هشت ثانیه یک بار تابع process_recv_data را صدا می‌زند که این تابع بررسی میکند در هشت ثانیه‌ی گذشت چه پیغام‌های Hello از گره‌های دیگر دریافت شده است و در صورت دو طرفه بودن ارتباط آن گره را در adj_list نگه می‌دارد و اگر از یکی از گره‌های موجود در adj_list پیغام hello دریافت نکرده باشد ان گره را از لیست حذف کرده و به تصادف یک گره دیگر انتخاب کرده و به آن پیغام hello میفرستد.

نحوه‌ی محاسبه‌ی زمان هم به این صورت است که هر هشت ثانیه یک بار یک شمارنده به اسم total_time به روز رسانی می‌شود و همچنین اگر ارتباط به گره‌ای به صورت دو طرفه باقی مانده باشد شمارنده‌ی مربوط به ان گره هم یک واحد افزوده می‌شود. در نهایت از تقسیم شمارنده‌ی مربوط به هر گره بر total_time درصد ارتباط بین دو گره بدست می‌آید.

به منظور شمارش تعداد بسته‌های ارسالی و دریافتی هم متغیری به اسم num_data_send تعریف شد که به ازای هر هاست دو عدد دارد که عدد اول به معنای تعداد بسته‌های فرستاده شده به آن هست(از هاست صاحب آن متغیر) و متغیر دوم تعداد بسته‌های دریافی از آن هاست(به هاست صاحب متغیر) است.

پس از سپری شدن زمان sim_time به تمامی نودها پیغام فرستاده می‌شود تا simulation را متوقف کرده و هر گره در یک فایل json مقادیر خواسته شده را مینویسد.