هو الحق

دانشکده‌ی مهندسی کامپیوتر

CE-40443 پاییز ۱۳۹۷

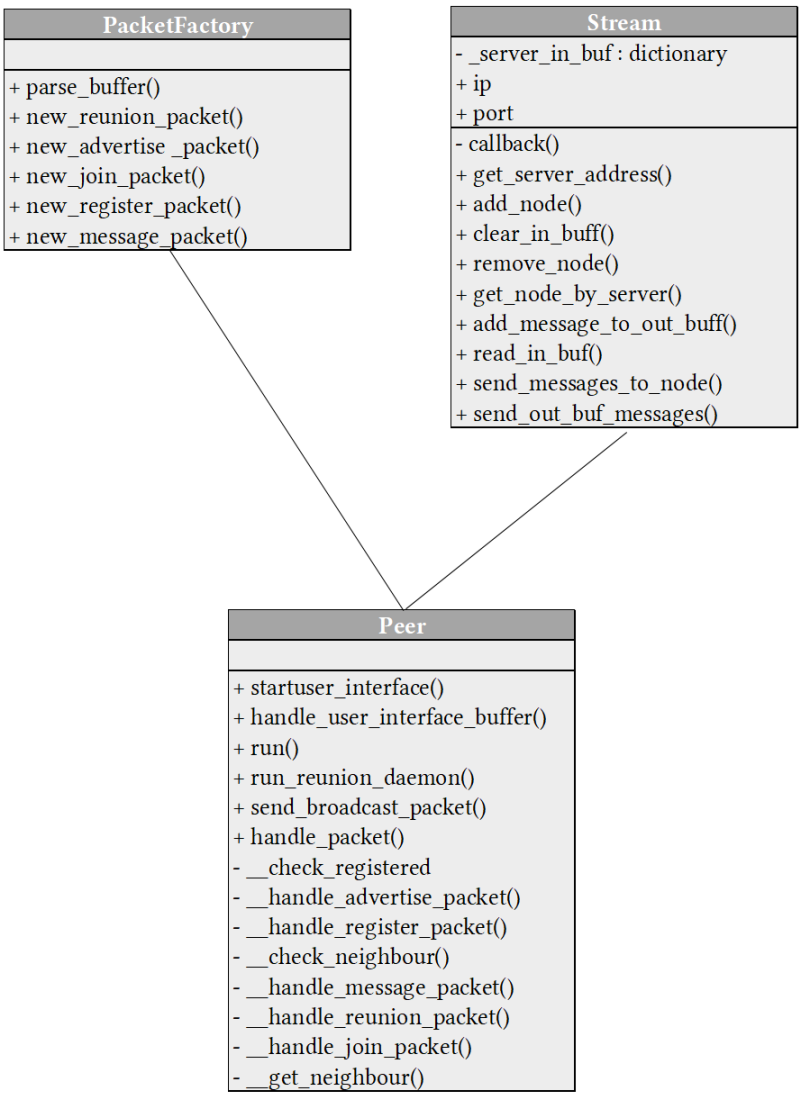
پیاده‌سازی شبکه‌ی P2P

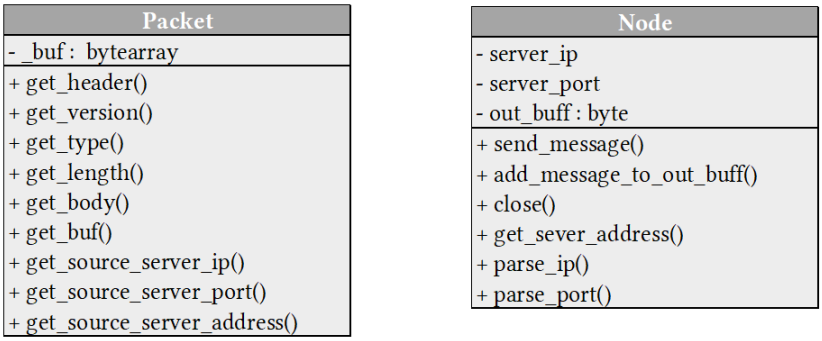
استاد: مهدی جعفری *موعد تحویل: ۱ بهمن، ساعت ۵ صبح*

۱ چکیده

هدف این پروژه پیاده‌سازی یک شبکه‌ی Peer to Peer و واسط کاربری برای ارتباط با شبکه است. این شبکه شامل دو جز اصلی است:

- یک Root که در نقش DNS عمل می‌کند.
- تعدادی Node که به صورت گراف همبند بدون دور (درخت) با ریشه‌ی Root به یک‌دیگر وصل شده و تشکیل شبکه می‌دهند. برنامه‌ی نوشته شده باید قابلیت اجرا شدن هم در نقش Root و هم در نقش Peer را داشته باشد و به گونه‌ای طراحی شود که تبادل پیام در شبکه قابل مشاهده باشد.


PacketFactory
+ parse_buffer()
+ new_reunion_packet()
+ new_advertise _packet()
+ new_join_packet()
+ new_register_packet()
+ new_message_packet()
Stream
- _server_in_buf : dictionary
+ ip
+ port
- callback()
+ get_server_address()
+ add_node()
+ clear_in_buff()
+ remove_node()
+ get_node_by_server()
+ add_message_to_out_buff()
+ read_in_buf()
+ send_messages_to_node()
+ send_out_buf_messages()
Peer
+ startuser_interface()
+ handle_user_interface_buffer()
+ run()
+ run_reunion_daemon()
+ send_broadcast_packet()
+ handle_packet()
- __check_registered
- __handle_advertise_packet()
- __handle_register_packet()
- __check_neighbour()
- __handle_message_packet()
- __handle_reunion_packet()
- __handle_join_packet()
- __get_neighbour()


UserInterface
+ buffer : byte
+ run()
Packet
- _buf : bytearray
+ get_header()
+ get_version()
+ get_type()
+ get_length()
+ get_body()
+ get_buf()
+ get_source_server_ip()
+ get_source_server_port()
+ get_source_server_address()
Node
- server_ip
- server_port
- out_buff : byte
+ send_message()
+ add_message_to_out_buff()
+ close()
+ get_sever_address()
+ parse_ip()
+ parse_port()

۲ مقدمه

P2P یا شبکه‌ی همتا به همتا توزیعی از معماری شبکه‌های کامپیوتری است که در آن هر Peer هم نقش Client و هم نقش Server را بازی می‌کند. و از این طریق می‌تواند به تبادل پیام در شبکه‌ای از Peerها بپردازد. در این نوع شبکه‌ها بر خلاف شبکه‌های Client-Server، سرور مرکزی وجود ندارد. Torrent، Tor و Bitcoin مثال‌هایی شناخته شده از شبکه‌های P2P هستند. نکته‌ای که حائز اهمیت است آن است که در عمل شبکه‌ای که به طور کامل P2P باشد وجود ندارد و حتی در شبکه‌های P2P هم Nodeهایی وجود دارد که سبب پابرجایی شبکه هستند و حذف آن‌ها از شبکه سبب اخلال در شبکه می‌شود.

۳ پروتکل شبکه

۱.۳ Client

اولین گام پس از ورود هر Client به شبکه رجیستر شدن آن در شبکه است. برای این کار Client با فرستادن پیام Register Request به Root شبکه خود را در شبکه ثبت می‌کند.

در ادامه هر Client باید آدرس Client پدر (اولین همسایه) را از DNS بگیرد. برای این کار Client با فرستادن پیام Advertise به Root درخواست یک آدرس کرده و در جواب آن آدرسی از طرف DNS (که در این شبکه همان Root است) به Client ارسال می‌شود.

حال Client با فرستادن پیام join به آدرسی که از Root دریافت کرده است، درخواست متصل شدن خود را به Client مورد نظر اعلام می‌دارد. Client پس از اتصال به اولین همسایه‌ی خود، جزئی از شبکه شده و می‌تواند پیام‌های خود را با دیگر کلاینت‌های داخل شبکه در قالب Broadcast به اشتراک بگذارد.

هر Client موظف است هر ۴ ثانیه یک بار با فرستادن پیام Reunion Hello از طریق پدرش به Root، و دریافت پاسخReunion Hello Back از طریق همان مسیر و از طرف Root، از اتصال به شبکه اطمینان حاصل نماید.

هر Client در صورت دریافت پیام Reunion Hello، آدرس خود را به انتهای پیام اضافه، فیلد Number of Entries را به روزرسانی کرده و پیام را به پدرش می‌فرستد. همچنین کلاینت‌ها در صورت دریافت پیامReunion Hello Back، آدرس خود را از انتهای پیام برداشته، فیلد Number of Entries را به روزرسانی کرده و پیام را به آدرس بعدی می‌فرستند.

در صورتی که پس از مدت زمان معینی (بدست آوردن این زمان به عهده‌ی خودتان است) Client پاسخ پیام Reunion Hello خود را دریافت نکرد، اتصال خود به شبکه را قطع شده فرض می‌کند و با فرستادن پیام Advertise Request، درخواستی مبتنی بر گرفتن آدرسی جدید به Root می‌فرستد.

نکته: در این شبکه عمق درخت حداکثر برابر ۸ است.

۲.۳ Root

اولین وظیفه‌ی Root در قبال هر Client، ثبت Client در شبکه با پاسخ به پیام Register Request فرستاده شده از Client است.

از دیگر وظایف Root فرستادن آدرس یک Client (هر Client بیشتر از دو فرزند ندارد) به عنوان Client پدر در پاسخ پیام Advertise Request است.

آخرین وظیفه‌ی Root انتظار به مدت لازم (بدست آوردن این زمان به عهده‌ی خودتان است) برای دریافت پیام Reunion Hello و فرستادن پیامReunion Hello Back در پاسخ پیام دریافتی است. در صورتی که پس از گذشت زمان معین از Client پیامی در قالب Reunion Hello دریافت نشود، Root آن Client را حذف شده در نظر گرفته و دیگر آدرس آن Client و زیر درخت مربوط به آن را Advertise نمی‌کند.

۴ پیاده‌سازی

در این بخش به معرفی وظایف اشیا ساخته شده از هر کلاس، تابع‌های موجود در آن‌ها و همچنین نحوه گرفتن ورودی در آن‌ها می‌پردازیم.

۱.۴ Peer

در این پروژه هر Node موجود در شبکه از جنس Peer است؛ چه Client باشد چه Root. به عبارتی ساده‌تر شبکه‌ی پیاده‌سازی شده عملاً اجتماعی از چندین Peer است که به نحوی به یکدیگر متصل‌اند (شبکه گرافی همبند است).

Peer همچنین وظیفه‌ی تولید Packet با استفاده از پیام‌های ورودی Stream یا به صورت دلخواه به کمک PacketFactory را دارد. تبدیل Packet به پیام و تحویل آن به Stream جهت ارسال آن در شبکه نیز از وظایف Peer می‌باشد. علاوه بر این انجام action هر Packet به عهده‌ی Peer است.

۲.۴ Stream

Stream در سطح شبکه پیاده‌سازی شده و مربوط به لایه‌ی کاربردی نیست. وظیفه‌ی اصلی Stream ارسال و دریافت پیام‌ها از طریق Nodeهای متفاوت ست. مدیریت در خصوص آن که هر Node چه پیام‌هایی باید بفرستد، چه پیام‌هایی را از چه Nodeهایی باید دریافت کند و در جواب آن‌ها چه پیام‌هایی را بفرستد، به عهده‌ی Stream است.

Stream با توجه به مقصد پیام، بافر را به Node مورد نظر می‌رساند. Node پیام را از طریق Stream دریافت کرده و با طی روند خاصی پیام را تبدیل به بافر کرده و به Node مورد نظر ارسال میکند. از طرفی بافرهای ورودی هر Node را تبدیل به پیام می‌کند و به Peer تحویل می‌دهد.

Node ۳.۴

Stream برای کار با Peerهای دیگر از Node استفاده می‌کند. به بیانی واضح‌تر هر Stream شامل تعداد زیادی Node می‌شود که هر کدام از این Nodeها دارای یک سوکت است که این سوکت به سوکت سرور Peer دیگر متصل است. هر Node دو بافر ورودی و خروجی دارد. در بافر ورودی پیامّهای فرستاده شده و در بافر خروجی پیام‌هایی که باید فرستاده شوند وجود دارند.

Packet ۴.۴

تمامی پیام‌هایی که در شبکه انتقال می‌یابند از جنس Packet هستند. Packet شی‌ای است برای Peer قابل فهم است. اما هر Packet قبل از انتقال پارس شده و به بافر تبدیل می‌شود و این بافر است که در شبکه انتقال می‌یابد. پس بافری که Peer از Stream دریافت می‌کند، لازم است به Packet تبدیل شود تا قابل فهم شود. ساختار هر Packet در ادامه مشخص شده است:

Register Request •

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
REQ (3 Chars)		
IP (15 Chars)		
Port (5 Chars)		

Register Response •

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
RES (3 Chars)		
ACK (3 Chars)		

Advertise Request •

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
REQ (3 Chars)		

Advertise Response •

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
RES(3 Chars)		
Server IP (15 Chars)		
Server Port (5 Chars)		

- Join

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
JOIN (4 Chars)		

- Message

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
Message (#Length Chars)		

- Reunion Hello

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
REQ (3 Chars)		
Number of Entries (2 Chars)		
IP0 (15 Chars)		
Port0 (5 Chars)		
IP1 (15 Chars)		
Port1 (5 Chars)		
...		
IPN (15 Chars)		
PortN (5 Chars)		

Header		
Version (1 Char/2 Bytes)	Type (2 Chars/2 Bytes)	Length (8 Chars/4 Bytes)
Source Server IP (12 Chars/8 Bytes)		
Source Server Port (5 Chars/4 Bytes)		
Body		
REQ (3 Chars)		
Number of Entries (2 Chars)		
IPN (15 Chars)		
PortN (5 Chars)		
...		
IP1 (15 Chars)		
Port1 (5 Chars)		
IP0 (15 Chars)		
Port0 (5 Chars)		

۵.۴ PacketFactory

هر Peer برای تولید Packet به PacketFactory نیاز دارد. هایPacket تولید شده، توسط این کلاس به بافر تبدیل شده و برای انتقال از طریق Peer به Stream داده می‌شود.

۶.۴ UserInterface

همان‌طور که از نام این کلاس پیداست، وظیفه‌ی آن ایجاد یک واسط بین برنامه و کاربر است. از طریق این کلاس کاربر می‌تواند عملکرد Peer را کنترل کرده و دستور ایجاد Packetهای متفاوت را بدهد.

۵ نکات دیگر

- برای پیاده‌سازی پروژه تنها از زبان پایتون می‌توانید استفاده کنید.
- گروه‌ها به صورت دو نفره است.
- در فایل پیوست تابع‌هایی که برای هر کلاس نیاز است پیاده‌سازی شود، توضیح داده شده است.
- فایل‌های تحویل دادنی را به فرمت zip. درآورده و در سامانه‌ی کوئرا بارگذاری نمایید.
- نام فایل ارسالی باید به فرمت CN_Project_STDID1#_STDID2# باشد.

موفق باشید