شبیه‌سازی روش‌های کاهش وقفه زمانی برنامه‌ها به هنگام مهاجرت میان سرورهای Edge

دانشجو: علی عباسی علایی

اساتید راهنما: دکتر ناصر یزدانی، دکتر رضا شجاعی

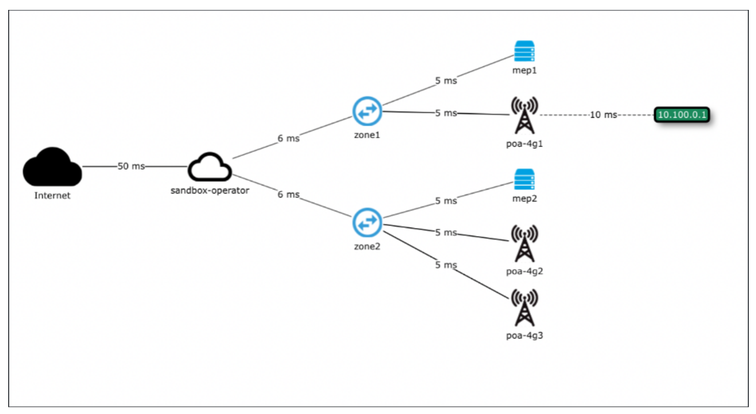
دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران

مقدمه

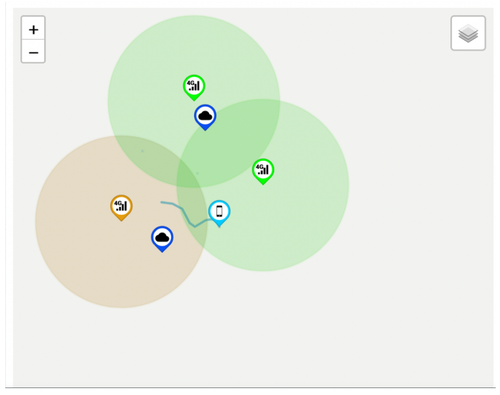
با انتقال ارائه دهندگان خدمات مخابراتی به نسل پنجم شبکه‌های تلفن همراه (5G)، تجربه حضور در جاده‌ها به طور قابل توجهی تغییر خواهد کرد. ویژگی تأخیر بسیار کم 5G، پشتیبانی از برنامه‌های جدید یا پیشرفته را برای مواردی مانند کمک راننده پیشرفته و خودروهای خودران، که در دسته برنامه‌های Vehicle-to-Everything قرار میگیرند امکان پذیر می‌کند. الگوی ابری برای برآوردن نیازهای تاخیر کم و قابلیت اطمینان بالا برخی از سرویس های V2X کافی نیست. در واقع، داشتن برنامه‌های کاربردی در فضای ابری به معنای تبادل داده با سرورهای دور است. منابع محاسباتی و ذخیره‌سازی داده‌های مورد نیاز باید نزدیک تر به لبه شبکه و در نتیجه نزدیک تر به کاربر نهایی در دسترس باشند. بنابراین، به غیر از 5G، این خدمات به فناوری توانمند دیگری مانند محاسبات لبه چند دسترسی (MEC) نیاز دارند. یکی از چالش‌های MEC، عملیات Handover، تغییر مکان کاربر و رویکرد آن برای مهاجرت برنامه اجرایی است. این تحقیق پیاده‌سازی تجربی از رویه‌های مهاجرت را ارائه می‌کند. مطالعه تجربی با استفاده از یک بستر آزمایشی انجام می‌شود که در آن تحرک کاربر توسط پلتفرم AdvantEDGE شبیه‌سازی می‌شود و این رویه‌ها با در نظر گرفتن زمان از کار افتادگی و مقدار حالت حفظ شده برنامه پس از مهاجرت بررسی و مقایسه می‌شوند.

روش و مدل پیشنهادی

این تحقیق به پیاده‌سازی و شبیه‌سازی دو استراتژی Filesystem and Container Configuration Preservation و Simple Filesystem Preservation با استفاده از پلتفرم AdvantEDGE می‌پردازد. معماری این شبیه سازی بر پایه تعامل سه سرور با یکدیگر است. یک سرور مسئولیت اجرای پلتفرم بر روی Kubernetes و دو سرور دیگر مسئولیت اجرای برنامه خدمت دهنده به UE را دارند. هر سرور Edge به یک MNO متصل شده و در زمان تغییر مکان UE، که با یک رخداد به شبیه ساز منتقل می شود، کاربر اختصاص داده شده به آن تغییر می‌کند و عملیات مهاجرت برنامه از سرور مبدا به سرور مقصد آغاز می‌شود. ابتدا در زیر توپولوژی در نظر گرفته برای این شبیه سازی را مشاهده می‌کنیم:



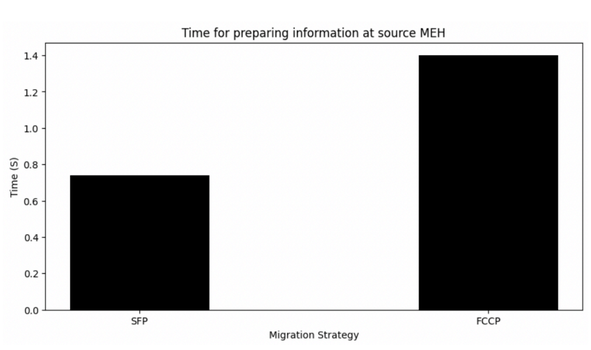
این توپولوژی شامل سه برج مخابراتی است که در دو ناحیه مختلف توزیع شده‌اند. همچنین، دو سرور ذکر شده در دو ناحیه قرار گرفته و در زمان ورود ترمینال کاربر به ناحیه هر برج، به آن متصل می‌شود. نحوه حرکت کاربر در این شبیه سازی به شکل زیر است:

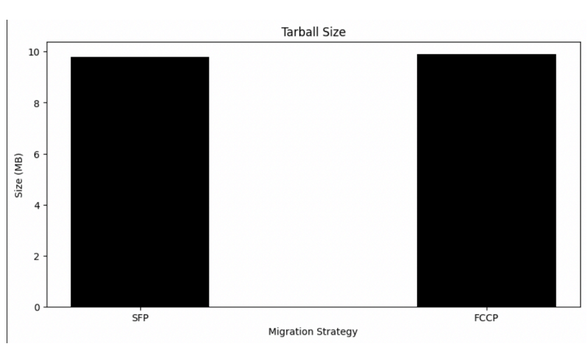


برنامه‌ای که در این مهاجرت مورد بررسی قرار می‌گیرد، یک برنامه شمارنده بر پایه کانتینر از تکنولوژی Docker است. در شروع مهاجرت، سرور منبع دستور commit (و یا export در رویه SFP) را انجام می دهد که تغییرات یا تنظیمات فایل کانتینر را در یک تصویر جدید اعمال می کند. پس از آن، دستور save را برای ذخیره یک تصویر جدید در یک فایل tar اجرا می کند. پس از آن، فایل tarball تولید شده را با استفاده از دستور scp لینوکس در سرور مورد نظر کپی می کند. در نهایت، متوقف می شود و کانتینر در حال اجرا را حذف می کند. در سرور مورد نظر، تصویر دریافتی بارگذاری و شروع می شود.

نتایج

حال به بررسی و مقایسه دو روش SFP و FCCP می‌پردازیم. در مورد حفظ حالت برنامه، به هنگام استفاده از SFP هیچ اطلاعات پیکربندی در سرور مقصد در دسترس نیست. در نتیجه، اجرای دستور Docker نمی داند چگونه برنامه را راه اندازی کند. در مورد FCCP، اطلاعات پیکربندی کافی برای شروع برنامه از حالت قبل وجود دارد و با ایجاد checkpoint در tarball ارسالی، سرور مقصد میداند چگونه باید برنامه را آغاز سازد. در این شبیه سازی، این اتفاق به صورت از دست رفتن شمارش برنامه در روش SFP و ادامه شمارش از مقدار پیشین در FCCP است. در مورد زمان انتقال باید سه فاز مورد بررسی قرار بگیرند: فاز آماده سازی اطلاعات در سرور مبدا، فاز انتقال اطلاعات و فاز بارگذاری و اجرا توسط سرور مقصد. در فاز اول، در مورد SFP، این پارامتر تنها به دلیل زمان صرف شده توسط دستور export است. در عوض، برای FCCP، می توان آن را با مجموع زمان های صرف شده توسط دستورات commit و save تخمین زد. در فاز دوم، این پارامتر برابر است با زمانی که دستور scp برای انتقال tarball از منبع به مقصد صرف می کند. مقادیر مشاهده شده در شبیه سازی تقریباً برابر است با اندازه tarball تقسیم بر نرخ bandwidth شبکه. در نهایت در فاز آخر، این زمان مربوط به دستور load در مقصد می‌شود. که در این مرحله نیز با توجه به نزدیک بودن حجم tarball ها به یک زمان رسیده‌ایم. در شکل‌های زیر زمان و حجم به دست آمده در این شبیه سازی برای هر کدام از دو روش در دو فاز اول آمده است و درنهایت در معیار زمان وقفه، روش SFP بهتر عمل می‌کند.





جمع بندی

توانستیم شبیه سازی تجربی دو استراتژی مختلف مهاجرت را ارائه دهیم. مهاجرت به ویژگی های برنامه بستگی دارد. به طور خاص، الزامات کاربردی از نظر حفظ داده‌های برنامه و داده‌های پیکربندی، و همچنین زمان مهاجرت ویژگی‌های مهم آن هستند. مشاهده کردیم SFP سریع‌ترین استراتژی است و می توان آن را برای مهاجرت برنامه های کاربردی بدون نیاز به حفظ وضعیت در نظر گرفت. با این حال، در طرف مقابل، FCCP به اندازه کافی حالت برنامه را حفظ می کند و به اندازه کافی عمومی است که به طور بالقوه از تعداد زیادی برنامه پشتیبانی کند و از سرعت نسبتا خوبی نیز برخوردار است.

کاربرد های صنعتی:

با پیشرفت MEC و کاهش وقفه‌های زمانی در اثر تغییر سرور لبه شبکه، توسعه دهندگان بیشتری می‌توانند از تکنولوژی Offload کردن بار کاری از UE ها به سرورهای Edge بهره ببرند. با توسعه MEC، برنامه‌های جدید یا پیشرفته‌ای مانند پردازش تصویر لحظه‌ای توسط خودروها، کمک راننده پیشرفته و نهایتا رانندگی خودکار به طور چشمگیری بهبود می‌یابند.

مراجع اصلی

1. **InterDigitalInc, AdvantEDGE, https://github.com/InterDigitalInc/AdvantEDGE, 2023.**
2. **ETSI ISG, “Multi-access edge computing(mec); study on mec support for v2x use cases,” ETSI GR MEC 022 V2.1.1, 2018.**
3. **Mohammed A. Hathibelagal, Rosario G. Garroppo, Gianfranco Nencioni, “Experimental comparison of migration strategies for MEC-assisted 5G-V2X applications,” Computer Communications, Volume 197, 1 January 2023.**
4. **D. Merkel, “Docker: lightweight linux containers for consistent devel- opment and deployment,” Linux journal, vol. 2014, no. 239, p. 2, 2014.**
5. **Y. Liao, L. Shou, Q. Yu, Q. Ai, and Q. Liu, “Joint offloading decision and resource allocation for mobile edge computing enabled networks,” Computer Communications, vol. 154, pp. 361–369, 2020.**