کنترل کننده SFC: کنترل رفتار فورواردسازی صحیح زنجیره عملکرد خدمات

چکیده:

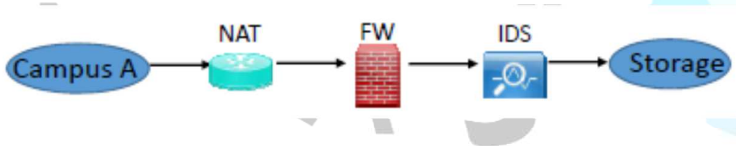
جعبه های میانی شبکه مدیریت و عیب یابی مشکلی به دلیل طراحی یکپارچه خصوصی شان دارند. با حرکت به سوی مجازی سازی عملکردهای شبکه یا NFV، کاربردهای جعبه میانی مجازی سازی شده می تواند به طور قابل انعطاف تری ذکر شده و به نحو پویایی زنجیره سازی شوند و در نتیجه عیب یابی را مشکل تر از قبل می کنند. برای تضمین موجودیت درجه حامل و حداقل سازی قطعی ها، گردانندگان به راههایی برای تایید خودکار این امر نیاز دارند که شبکه استقرار یافته و پیکربندی های جعبه میانی از سیاستگزاریهای شبکه سطح بالاتری تبعیت کنند. در این مقاله، ما ابتدا به تعریف و شناسایی چالشهای کلیدی برای کنترل رفتار فورواردسازی صحیح زنجیره های عملکرد خدمات یا SFC می پردازیم. ما سپس به طراحی و ابداع یک چارچوب تشخیص شبکه می پردازیم که به مجریان شبکه کمک می کند تا صحت اجرای سیاستگزاری SFC را تایید کنند. پیش نمونه ما (یعنی کنترل کننده SFC) می تواند زنجیره های خدمات حالتمند را به طور کارامدی تایید کند، که با تحلیل قوانین فورواردسازی کلیدها و رفتار فورواردسازی حالتمند جعبه های میانی انجام می شود. ما براساس مدلهای عملکرد شبکه یک الگوریتم تشخیصی را مطرح و ابداع کرده ایم که قادر به کنترل رفتار فورواردسازی حالتمند یک زنجیره از عملکردهای خدمات شبکه می باشد.

1-مقدمه

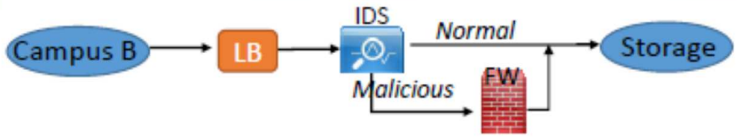
مجازی سازی عملکردهای شبکه یا NFV یک تغییر و تحول معنی دار فراساختار Telco برای کاهش هم CAPEX و هم OPEX می باشد ضمن اینکه سطوح خدمات درجه حامل بالایی را حفظ می کند. حرکت به سمت عملکردهای شبکه مجازی سازی شده یا NFها روی سرورهای استاندارد باعث ایجاد احتمال عملکرد کاهش یافته و افزایش تعداد خطاها و وقفه ها می شود. با اینحساب عیب یابی و تشخیص مسائل خیلی قبل از استقرار یک مسئله حیاتی و مهم می باشد. یک الزام خوب NFV همان زنجیره سازی عملکرد خدمات یا SFC می باشد که طی آن ترافیک از طریق یک تعیین توالی NFها به نحو پویا هدایت می شود. حتی با NFهای فیزیکی امروزی، ساخت یک زنجیره خدماتی دربردارنده مولفه های متعددی است: تعریف سیاستگزاری، برنامه ریزی کنترل کننده SDN، نصب جداول

جریان کلیدی، و پیکربندی NFها. اشتباهات در هر یک از این مولفه ها می تواند باعث شود که بسته ها به سمت NF اشتباه پیش بروند یا اینکه به ترتیب اشتباه رفته یا حذف شوند. با ظهور NFV، مقیاس و پویایی زنجیره سازی NFهای مجازی یا VNFها محتملا به نحو معنی داری افزایش می یابد (این خطاها تنها شایع تر خواهند شد). با اینحساب تایید و عیب زدایی SFC به طور روزافزونی برای موفقیت اتخاذ NFV امری حیاتی است. هدف بلندمدت ما ساخت یک چارچوب تشخیص و عیب یابی جامع NFV می باشد که داخل آن ابزار شبکه، NF و ابزار عیب یابی SFC را هم برای تشخیص اشتباه ایستا و هم پویا و نیز هم پیش فعال و هم واکنشی بتوان پلاگ کرد و به کارایی عملیاتی کمک می کند حین اینکه SLAهای لازم را هم حفظ می نماید.

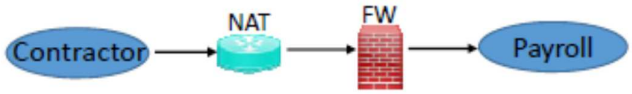
در ین مقاله با توجه به هدف متعالی فوق، ما یک ابزار عیب یابی و تشخیص SFC را ابداع کرده ایم. به طور اخص تر، ما به بررسی این امر پرداخته ایم که آیا جریانات به طرز صحیحی طبق به سیاستگزاریهای زنجیره سازی خدمات سطح بالا فوروارد می شوند یا خیر.



الف) پیمایش صحیح زنجیره خدمات ایستا



ب) پیمایش صحیح زنجیره خدمات پویا



ج) پیکربندی NF: HTTP از پیمانکار به فهرست حقوق بگیران که توسط FW مسدودسازی شده است.

شکل 1-سوالات مثال

ما این مورد را کنترل یا تشخیص رفتار فورواردسازی یک SFC می نامیم. این مورد شامل سه جنبه است، همانگونه که از طریق سه مثال تصویری شکل 1 نشان داده شده است. این مورد ابتدا باید توالی NFها را هر بار که جریانی باید عبور کند، کنترل کند. در شکل 1الف) سیاستگزاری ملزم می دارد که همه ترافیک HTTP از دانشگاه A به

سرورهای ذخیره سازی باید بوسیله یک NAT مورد رسیدگی قرار گیرد که با یک دیوار آتشین و سرانجام یک IDS دنبال می شود. برای کنترل اجرای صحیح این سیاستگزاری، ما نه تنها باید کنترل قوانین فورواردسازی را روی کلیدها انجام دهیم بلکه باید کنترل کنیم که چگونه NFها بسته ها را فورواردسازی می کنند. دوم اینکه زنجیره خدمات می تواند به نحو پویایی در زمان اجرا طبق حالات NF برگشته تغییر کند. در شکل 1ب، جریان در آغاز از طریق یک IDS و یک متعادلسازی کننده بار یا LB عبور می کند. اگر IDS یک امضای حمله را در جریان شناسایی نماید، زنجیره خدمات را جوری تغییر می دهد تا حاوی یک دیوار اتشین برای تحمیل سیاستگزاری بشود از جمله حذف ترافیک بدخیم. ما این کار را زنجیره های خدمات پویا می نامیم و هدفمان کنترل اجرای صحیح آن هم در کلیدها و هم در NFها می باشد. در کنار کنترل مسیر یک شبکه، همچنین ما به کنترل پیکربندی های NF می پردازیم. مثال سوم در شکل 1 ج) نشان دهنده مشکلات اجرای چنین تحلیل استاتیکی در حضور NFها می باشد. سیاستگزاری مشخص می کند که یک درخواست وب از سرور پیمانکار به سرور فهرست حقوق بگیران باید توسط FW مسدود بشود. بررسی این سیاستگزاری در مسیری بدون NF ها به طور قابل مقایسه ای سهل و آسان است: بررسی قوانین در زمینه FW و دیدن اینکه آیا دامنه صحیح نشانی های IP منبع مسدود شده است یا خیر. ولیکن کار زمانی دشوار می شود که یک NAT ادرسهای IP منبع اصلی را قبل از FW پنهان کرده باشد. توجه داشته باشید که ما تنها رفتارهای فورواردسازی یک SFC را مورد رسیدگی قرار می دهیم. سایر رفتارهای مرتبط غیرفورواردسازی NF، مانند شمارش، بهینه سازی ترافیک از دامنه کار این مقاله خارج است. ما قصد داریم تا این کارهای اضافی را در کارهای اتی مطرح کنیم.

یک راه مستقیم برای کسب خطاها در فوروارد SFC همان نظارت بر جریان در زمان اجرا و بعد مقایسه مسیر مشاهده شده با سیاستگزاری می باشد. ولیکن تا زمانی که خطا مورد شناسایی قرار گیرد، ترافیک قبلا شناسایی شده است. در این کار، ما درباره چارچوب کاری تحلیل استاتیک برای کسب مسائل قبل از استقرار بحث می کنیم. این کار اغلب تحت عنوان تایید شبکه در زمینه SDN نامیده می شود. متفاوت از تایید کد رسمی، ابزار تایید شبکه اساسا به بررسی قوانین روی همه کلیدهای شبکه می پردازد.

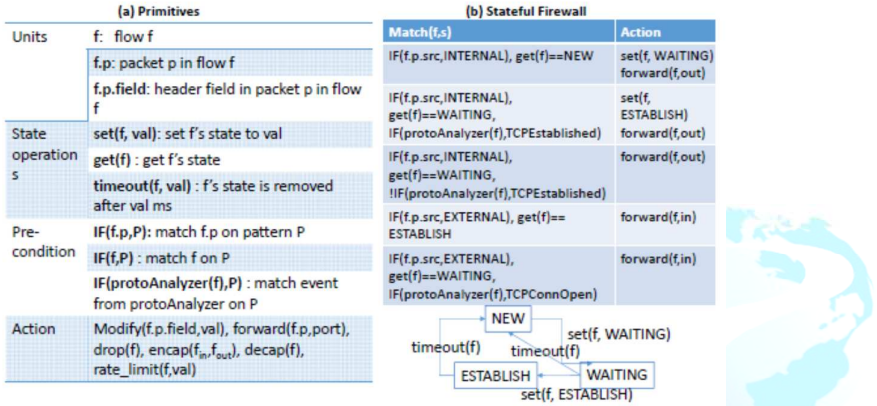
هدف ما دنبال کردن همان شیوه برای اجرای تایید روی رفتارهای فورواردسازی SFC می باشد. ولیکن دریافته ایم که روشهای موجود نمی تواند به طور مستقیم بکار بسته شود تا صحت SFC را به چند دلیل کوچک بررسی نماید.

اول اینکه یک سیاستگزاری کاری پیچیده و حالتمند است. برای مثال، یک سیاستگزاری می تواند مشخص کند که کاربران غیرمجاز از دسترسی به سرورهای حساس منع شوند. برای انجام چنین کاری، یک گرداننده می تواند یک دیوار اتشین حالتمند را استفاده کند تا تضمین شود که تنها ترافیکی که درون شبکه شروع شده اجازه دارد و در انجام این کار از اینرو کاربران را از ترافیک بدخیم مصون بدارد. NFها حالات هر جریان را نگه داشته و اقدامات مختلفی را براساس این حالات اجرا می کنند. دوم اینکه، برگرفته های فورواردسازی موجود (برای مثال OpenFlow) نمی تواند به طور مستقیم بکار بسته شود چرا که همه بسته های یک جریان مورد یکسانی را به کمک قانون تطابق-اقدام مورد رسیدگی قرار می دهند. ب ااینحساب ، ما به مشتق فورواردسازی جدیدی احتیاج داریم تا حالت عزیمت را برای جریانات انفرادی درنظر بگیریم. سرانجام اینکه برای کنترل یک SFC باید همه NFها و کلیدهایی را بررسی کنیم که جریانها از آن ها عبور می کنند: اساسا، تایید کل شبکه مورد نیازا ست. درحالیکه تایید وسایل شبکه بدون حالت از لحاظ محاسباتی خیلی چالش آور است، افزودن وسایل حالتمند باز این مسئله را پیچیده تر هم می سازد.

برای مطرح سازی این گونه چالشها، ما کارهای ذیل را انجام داده ایم. اول اینکه مدلهای برگرفته جعبه میانی موجود را مورد بهره برداری قرار داده و انها را به مدل فورواردسازی برای صفحه داده های NF تعمیم داده ایم. هر NF با استفاده از یک جدول جریان و یک ماشین حالت توضیح داده شده اند. ما OpenFlow موجود را براساس قوانین تطابق اقدام به دو شیوه بسط داده ایم: 1) شرایط تطابق شامل نه تنها سرایند بسته بلکه حالات داخلی NF ها می باشد و 2) اقدام شامل نه تنها اصلاح بسته ها بلکه تحریک انتقالهای حالت NF می باشد. باز، تطابق و اقدام می تواند علیه یا یک بسته منفرد یا یک جریان کامل تعریف بشود. رابطه موقت میان حالات با استفاده از یک ماشین حالت محدود یا FSM توضیح داده شده است. توجه داشته باشید که مدلسازی هر جزئیات یک NF پیچیده به وضوح بدون بکارگیری تکنیک های تحلیل کد پیچیده کاری پیچیده است. در عوض ما بر تنها رفتار فورواردسازی NF تمرکز کرده ایم. سوای بسطهای اخیر برای ایجاد حالتمندی در OpenFlow، مدل م مخصوص کنترل مسئله NF و SFC است. دوم اینکه ما یک الگوریتم کارامدی را برای تحلیل استاتیک شبکه های حالتمند ابداع کرده ایم. روشهای موجود گرافهای فورواردسازی را از روی قوانین می سازد و با استفاده از این گراف ها تایید می کند. ولیکن این رویکرد ناکافی است چرا که گرافهای فورواردسازی تنها به کسب رفتار فورواردسازی شبکه می پردازد، ولیکن

انتقالات حالت NF ها را کسب نمی کند. ما یک گراف فوروارد‌سازی حالتمند یا SFG را مطرح کرده ایم که هم انتقالات حالت و هم رفتار فوروارد‌سازی را کدگذاری می کند. ما یک الگوریتم را تدوین کرده ایم که به طور خودکار SFGها را از جداول NF و FSMها تولید می کنند. بعلاوه، ما الگوریتم های عبور گراف متعدد را روی SFG طراحی کرده ایم که به سوالات قابلیت دسترسی وابسته به حالت پاسخ می گویند.

در این کار، ما به طراحی و اجرای کنترل کننده SFC می پردازیم که یک چارچوبی است که کنترل صحت رفتار فوروارد‌سازی را برای زنجیره های عملکرد خدمات اجرا می کند.



شکل 2- اصول اولیه مدل و مثال

مشابه با کار تایید موجود، ما بر کنترل پایداری های قابل دسترسی حالتمند تمرکز کرده ایم: برای مثال، با درنظرگیری مسیر بسته ویژه، جریان از کدام توالی NFها عبور می کند، یا اینکه بعد از کدام توالی بسته ها، A اجازه ارتباط با B را دارد؟ ما یک پیش نمونه از کنترل کننده SFC ابداع کرده و انرا برای تحلیل سرعت تشخیص SFC و قابلیت ارتقای آن بکار برده ایم. نتایج ارزیابی اولیه نشان می دهد که ما می توانیم یک NF را با 27 حالت در عرض 1 ثانیه و یک شبکه 1800 NF را در هر مسیر در عرض 12 میلی ثانیه تایید نماییم. زمان ساختن گراف فوروارد‌سازی حالتمند حول 100 ms-300ms می باشد. برای NFها در ارزیابی، ما مدلهای فوروارد‌سازی را به شکل دستی ابداع کرده ایم. یک تلاش اتی بر تولید مدل خودکار متمرکز خواهد بود.

2-برگرفته فوروارد‌سازی NF

NFها می توانند بسته هایی را در حالت پیچیده تغییر شکل بدهند. برای مثال یک NAT به اصلاح IP منبع و پورت منبع خواهد پرداخت. یک بهینه ساز WAN می تواند ارتباط TCP را از کلاینت خاتمه داده و یک نوع جدید

را با سرور شروع کند. بسیاری NFهای تجاری دارای ترکیبی از خصوصیات پیچیده می باشند. برای مثال یک دیوار اتشین Bluecoat می تواند همچنین به شکل یک پروکسی وب، یک IDS و یک مسیریاب عمل کند. ایجاد مدلی که هر جزئیات از یک NF را کسب کند کاری چالش برانگیز است. ولیکن با بهره گیری از کارهای قبلی، این امکان وجود دارد که مدلی را برای رفتار فورواردسازی یک NF ایجاد کنیم.

الف-روشهای مدلسازی موجود

به دلیل ماهیت خصوصی آنها، قابلیت عملکرد گوناگون و پیچیدگی بالا، مدلسازی رفتار یک جعبه میانی خیلی چالش برانگیز است. تقریبا کلیه کارهای موجود یک مدل برگرفته را باری توضیح رفتار سطح بالای NF ایجاد کرده اند. جدول 1 به خلاصه سازی تازه ترین کار می پردازد. انها یا براساس دانش دومین متخصص یا تحقیقات دستی کد منبع ساخته شده اند. ما انها را طبق نمایش فیلدهای تطابق آنها، نمایش حالتمندی، و اینکه اقدامات طبق یک بسته یا طبق حالت تعریف شده باشند، خلاصه سازی کرده ایم. ابزارهای تایید شبکه موجود و کنترل کننده شبکه مدلهای کارد میانی ساده سازی شده ای را بکار می گیرند که توالی حالت یا بسته را کسب نمی کنند. هرچند متفاوات از گرانولاریته و فرمت مدلسازی می باشد ، ما دریافته ایم که کلیه مدلها می توانند به شکلی تبدیل شوند که در ذیل توضیح انرا داده ایم، که مناسب اهداف تشخیص کارامد می باشد.

ب-مدل فورواردسازی NF

اقتباس فورواردسازی یعنی OpenFlow کلید اسانی است که باعث می شود تایید صفحه داده ها کاری عملی بشود

جدول 1-تاکسونومی مدلهای جعبه میانی

		MM [9]	VIP [7]	SymNet [10]	BUZZ [11]	HSA [2]	Pyretic [12]	SFC-Checker
How to obtain	Source code	x	x	x	✓	x	x	x
	Expert knowledge	✓	✓	✓	x	✓	✓	✓
Match fields	L2/L3 Header	✓	✓	✓	✓	✓	✓	✓
	L4-7 payload	✓	✓	✓	✓	x	x	✓
State representation	State	✓	x	✓	✓	x	x	✓
	Packet sequence	x	✓	x	x	x	x	✓
Action	On packets	✓	x	✓	✓	✓	✓	✓
	On states	✓	✓	✓	✓	x	x	✓

چرا که پیچیدگی صفحه کنترل را پنهان کرده و یک سطح مشترک متحدی را نمایان می سازد. ما با الهام از اقتباس تطابق –اقدام OpenFlow، ما یک اقتباس فورواردسازی NF جدید را مطرح کرده ایم. این اقتباس NF بر صرفا رفتار فورواردسازی NF متمرکز است، و در ضمن مشابه با مدلهای قبلی دارای سه تفاوت مهم می باشد:

1) مدل ما بویژه مناسب چارچوب تشخیص حالتمند ما می باشد. 2) ما رویدادهای انتقال حالت را در گرانولاریته اختیاری مدلسازی کرده ایم (از سطح بسته تا رویدادها مانند تایید ارتباط، که برای داشتن انفجار فضای حالت از آن بهره برداری کرده ایم.) و 3) مدل ما برای مدلسازی جعبه های میانی پیچیده تر با عملکردهای پردازش بسته داخلی اختیاری نویدبخش است. سرانجام اینکه مدلهای NF موجود می تواند به سهولت به مدل ما تبدیل بشود.

اقتباس NF ما از دو بخش تشکیل شده است: یک جدول تطابق اقدام و یک ماشین حالت. ماشین حالت یک نمایش طبیعی از فرایندهای حالتمند است. گره ها در ماشین حالت همان حالتهایی است که هر NF راجع به رفتار فوروارد‌سازی حفظ می کند و لبه ها به کسب شرایطی می پردازند که انتقالات حالت را تحریک می کند. جدول شامل قوانین تطابق اقدام می باشد: تطابق روی هم سرایند بسته و هم حالات داخلی، با اجرای اقدام روی بسته ها و تغییر حالات داخلی.

ما این مدل را به دو دلیل انتخاب کرده ایم. اول اینکه دریافته ایم که همه مدلهای موجود در جدول 1 می تواند به شکل ماشین حالت محدود یا FSM تبدیل بشوند. این امر بدان معناست که این نمایش به قدر کافی عمومی است که انواع جعبه های میانی را کسب کند. دوم اینکه ما دریافتیم که روشهای تایید کارامد موجود براساس الگوریتم های عبور گراف می باشد. مدل FSM+Table جعبه های میانی می تواند به سهولت در ساختارهای داده گراف موجود ترکیب بشود. چنین نمایشی برای ایجاد الگوریتم های قابل ارتقا و کارامد در راس آن اهمیت دارد.

اقتباس مطرح شده می تواند یا از روی تحلیل کد منبع ساخته بشود یا اینکه از روی مدلهای جعبه میانی سطح بالای موجود تبدیل بشود. در این مقاله ما بر ایجاد اتوماتیک مدل برگرفته تمرکز نکرده ایم. در عوض با درنظرگیری چنین مدل فوروارد‌سازی NF، ما بر الگوریتم های کنترل SFC/تشخیص متمرکز شده ایم.

اصول اولیه اقتباس: برای توضیح، ما یک مدل جعبه میانی موجود را به شکلی از اقتباس فوروارد‌سازی خودمان با سه تغییر اصلی تبدیل کرده ایم: 1) جداسازی ماشین حالت و مدل تطابق اقدام . 2) تعریف قوانین طبق بسته ها و نیز جریان . aka توالی بسته . 3) درنظرگیری یک مجموعه بزرگتر اقدامات.

شکل 2a نشان دهنده مجموعه ای از اصول اولیه است که در حال حاضر پشتیبانی می کنیم. ولیکن اصول اولیه دیگر می تواند به سهولت اضافه شود. واحدهای یک قانون می تواند یک فیلد، یک بسته یا یک جریان باشد چرا که ممکن است ما قوانین متفاوتی را برای بسته های همان جریان داشته باشیم. عملیات حالت شامل get(f) برای

بازیابی حالت داخلی NF برای جریان f، و set (f,val) برای شروع/اصلاح حالت داخلی NF برای جریان f به ارزش val می باشد. پیش شرط و اقدام می تواند براساس بسته ها و جریانها تعریف بشود. وانگهی، چون NFها ممکن است سرایند اپلیکیشن را از روی بارهای بسته تجزیه کنند، ما باز به تعریف اصول اولیه protoAnalyzer می پردازیم که فرض می کند NF از مشخصات پروتکل لایه 7-5 تبعیت کرده و تولید رویدادهای ویژه پروتکل را می کند برای مثال درخواست HTTP و درخواست FTP .

مثالها: با استفاده از اصول اولیه، ما مدلهایی را برای شش تا NF ایجاد کرده ایم: یعنی NAT، متعادلسازی کننده بار، دیوار اتشین حالتمند، IDS، دروازه VPN، و دروازه PDN. این NFها به شدت مورد مطالعه بوده اند و رفتار فورواردسازی آنها می تواند با استفاده از این رویکرد مدلسازی بشود. انها 5 تا از 8 نوع جعبه میانی متداول را طبق یک تحقیق اخیر روی کاربرد جعبه میانی تحت پوشش قرار داده اند. برای اختصار، ما با استفاده از مثالی در شکل 2b درباره یک دیوار اتشین حالتمند توضیحاتی می دهیم که حاوی سه حالت ارتباطی است. ما از اصول اولیه TCP protoAnalyzer برای اقتباس رفتار حالتمند استفاده کرده ایم. نتیجه خروجی TCP protoAnalyzer همان رویدادهای مرتبط مانند ارتباط باز یا تایید جلسه می باشد. قوانین مشخص می کنند که یک بسته خارجی تنها در صورتی فورواردسازی می شود که یک SYN/ACK ی ارتباطی می باشد که توسط میزبانهای داخلی یا از یک ارتباط برقرار شده، شروع شده است.

3-الگوریتم تحلیل قابلیت دسترسی حالتمند

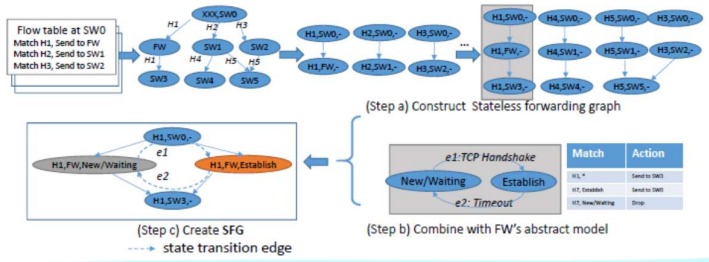
تحلیل استاتیک ما از رفتار فورواردسازی در سه مرحله رخ داده است: ما ابتدا یک تصویر کلی از حالت شبکه گرفته ایم یعنی توپولوژی، جداول جریان و مدل NF. دوم اینکه با استفاده ازا ین حالت ما یک گراف فورواردسازی حالتمند SFG را ایجاد کرده ایم که نمایانگر این است که چگونه بسته ها فورواردسازی شده و چگونه حالات NF مطابق با آن تغییر می کند و سرانجام ما الگوریتم های عبوری گراف را روی SFG تدوین کرده ایم که به سوالات مختلف برای کمک به گردانندگان در کارهای تشخیصی SFC خودشان پاسخ می گوید.

الف-گراف فورواردسازی حالتمند.

اگر کلیه وسایل بی حالت باشند، می توانیم یک گراف فورواردسازی بی حالت را ایجاد کنیم که نمایانگر توپولوژی و مسیرهای مسیریابی مشابه با رفرانس 1 می باشند. با وجود NFهای حالتمند، ما یک SFG را با ترکیب گراف

فورواردسازی با ماشین های حالت NF ایجاد کرده ایم. فرایند ترکیب یک مورد خاص از ترکیب گراف لکسیکوگراف است. رئوس ماشین حالت یک NF تنها با رئوس متناظر با NF در گراف فورواردسازی بی حالت ترکیب می شود. یک SFG کسب می کند که چگونه بسته ها بین وسایل فورورد شوند، چگونه حالات درون وسایل تغییر کنند، و چگونه تغییرات حالت بر مسیر فورواردسازی اثر داشته باشد.

در یک SFG، هر گره به شکل $< H, D, S >$ نمایش داده می شود که نمایانگر بسته ای در فضای سرایند بسته H می باشد که در یک وسیله شبکه ی D (کلید یا NF) می رسد، وقتی که وسیله شبکه در یک حالت ویژه S قرار داشته باشد. یک لبه با اشاره از یک گره $< H_1, D_1, S_1 >$ به گره دیگر $< H_2, D_2, S_2 >$ به معنای این است که زمانی که یک بسته H_1 در D_1 با حالت S_1 ارسال می شود، این مورد به H_2 اصلاح شده و به یک وسیله D_2 در حالت S_2 فورواردسازی می شود.



شکل 3-گراف فورواردسازی حالتمند و فرایند ساخت و ساز

در شکل 3c، بسته های H_1 از طریق SW_0 و FW و SW_3 عبور می کنند. ضرورتا، دو نوع لبه های SFG وجود دارد:

لبه فورواردسازی: یک لبه فورواردسازی $< H_1, D_1, S_1 >$ به $< H_2, D_2, S_2 >$ به معنای این است که D_1 بسته H_1 را به H_2 تغییر داده و بسته تغییر داده شده را به D_2 فورواردسازی می کند. ما این لبه ها را به شکل لبه های جامد در مثالهای خودمان تصویرسازی کرده ایم (شکل 3c). همه لبه ها در یک گراف فورواردسازی بی حالت لبه ها را فورواردسازی می کنند.

لبه انتقال حالت: در یک NF حالتمند، یک انتقال حالت می تواند در اثر بسته های قبلی یک جریان تحریک بشود. با اینحساب با بسته های همان جریان بسته به حالات داخلی آن می تواند به طرز متفاوتی رفتار بشود. ما این امر را با استفاده از یک لبه انتقال حالت نمایش داده ایم بدان معنا که حالت یک NF به دلیل عبور بسته کنونی از

میان حالت تغییر می کند. هر لبه انتقال حالت با یک شرایط انتقال همراه است برای مثال بسته بعدی یا جواب HTTP که به شناسایی رویداد می پردازد زمانی که انتقال رخ خواهد داد. این امر به شکل لبه نقطه گذاری e1 در شکل 3c نشان داده شده است. یک لبه انتقال حالت همیشه بین همان وسیله وجود دارد. مشابها، لبه انتقال حالت دیگر e2 افزوده می شود که بازتاب تغییر FW به حالت جدید/در حال انتظار می باشد.

در بسیاری از NFها، جریانات در جهات رو به جلو و برعکس در حالت داخلی یکسانی سهیم هستند. با اینحساب، یک انتقال حالت در یک جهت باید به نحو مناسبی در جهت دیگری بازتاب یابد. برای انجام چنین کاری، ما لبه های انتقال حالت را بین حالت NF در هردو جهت ایجاد کرده ایم. برای مثال در شکل 4، جریان خروجی (H1) از SW0 به SW3 از طریق دیوار آتشین حالتمند FW عبور می کند ضمن اینکه جریان ورودی (H7) مسیر برعکسی را در پیش می گیرد. FW تنها بسته های H2 را فوروواردسازی می کند اگر بسته های خروجی را قبلا دیده باشد. ما یک لبه را از $< H_1, FW, N/W >$ به $< H_7, FW, E >$ ایجاد می کنیم. این مورد با برچسب e1 نمایش داده می شود چرا که انتقال حالت با همان رویداد تحریک می شود: دست تکان دهی TCP با H1 شروع شده است.

ب-ساخت SFG

یک راه تازه برای ساخت یک SFG همان برشمردن مسیر و حالاتی است که هربسته ای از آن عبور می کند. به وضوح، این روش قابل ارتقا نیست چرا که فضای سرایند بسته می تواند عظیم باشد و با اینحساب اندازه SFG قابل رهگیری نیست.برای غلبه بر این چالش، ما از بینشی بهره گرفته ایم که بسیاری جریانات همان مجموعه NFها و کلیدها را طی می کنند که می تواند با هم به شکل یک رده معادلی گروه بندی بشود. یک جریان یک مجموعه بسته هایی است که برخی فیلدهای سرایند مشترک را به اشتراک دارد مانند 5 چندتایی. دوم اینکه بسیاری جریانات با اقدام یکسانی هنگام بودن در حالت یکسانی مواجه می شوند (برای مثال حالت برقراری ارتباط) در نتیجه می توانند با هم نیز گروه بندی بشوند. براساس این دو فرضیه، ما جریانات را به رده های معادل گروه بندی کرده ایم. مشابه رفرانس 1 و 2؛ یک رده معادل بنا به تعریف یک مجموعه جریاناتی است که در اقدام یکسانی روی همه وسایل شبکه در هر حالتی اشتراک دارد. جریانات در همان رده معادل از همان SFC عبور می کند و دارای

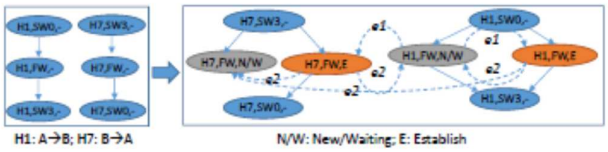
همان اقدامات می باشد. جریاناتی که همان اقدام را در یک وسیله خاصی به اشتراک دارند با صرفا یک گره در یک SFG نمایان می شود.

ما یک الگوریتم ساخت SFG را در ذیل مطرح کرده ایم. این الگوریتم با یک فضای جریان تجمع یافته بزرگی شروع می شود برای مثال $< XXX, SW_0 >$ در شکل 3a. برای ساده سازی، ما از وسایل بی حالت در اینجا استفاده کرده ایم بنابراین حالات همگی – می باشند. انگاه ما اقدامات فوروواردسازی را برای فضای جریان تجمع یافته از قوانین می توانیم کسب کنیم. جدول جریان در شکل 3 حاکی از سه جهش بعدی می باشد. ما گره های جهش بعدی را طبق شرایط تطابق ایجاد کرده ایم. وقتی گره های جهش بعدی ایجاد می شود، ما گره والد $<XXX, SW_0 >$ را به سه گره $< H_x, SW_0 >, x = 1,2,3$ تقسیم بندی کرده ایم. سپس ما گره بعدی را گرفته و همان فرایند را تکرار می کنیم. وقتی یک گره فرزند تقسیم می شود، اثر باید انتشار برگشتی به همه والدهای مستقیم آن داشته باشد. این فرایند همچنان ادامه می یابد تا زمانی که همه قوانین تجزیه بشود. در پایانا، این فرایند یک گراف فوروواردسازی بی حالتی را ایجاد می کند.

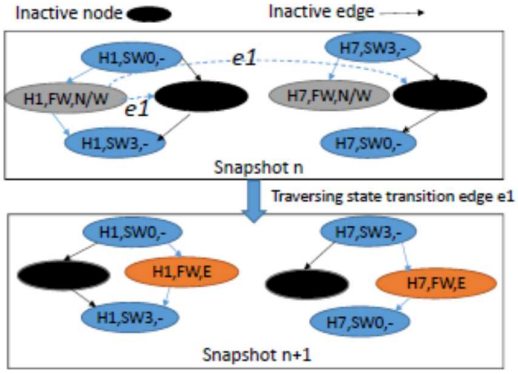
بعد ما قوانین فوروواردسازی NF و ماشین های حالت را گرفته و هر گره NF حالتمند را در گراف اصلی به گره های بسیاری بسط می دهیم. هر گره به طور منحصر به فردی نمایانگر NF در حالت خاصی می باشد. با استفاده از گره های به رنگ خاکستری درشکل 3a و 3b، ما گره ها را با FW به چهار گره بسط می دهیم که نمایانگر FW هم در جهت فوروواردسازی و هم در جهت برعکس می باشد. گره ها نمی تواند روی جهات فوروارد و معکوس اشتراک داشته باشند چرا که FW دارای اقدامات مختلفی است. لبه های انتقال حالت نیز طبق ماشین حالت FW و قوانین فوروواردسازی اضافه می شوند. در مثال دیوار اتشین حالتمند، چون وقفه (e2) می تواند هم در جهات فوروواردسازی و هم در جهات برعکس آن رخ بدهد، چهار لبه e2 وجود دارد که در شکل 4 نشان داده شده است.

ج-تایید روی SFG

تحلیل استاتیک از این مشاهده بهره می گیرد که برای هر جریان NF در یک و تنها یک حالت در هر زمان معین می باشد. اساسا، تنها یک گره در حالت یک NF فعال خواهد بود.



شکل 4-SFG با جهات فوروارد و برعکس



شکل5-تحلیل رفتار فورواردسازی موقت

با فعالسازی گره های مختلف (متناظر با حالات مختلف یک NF) طی فرایند تحلیل استاتیک، ما قادریم تا سناریوهای فورواردسازی مختلف را بین NFها و حالات کنترل کنیم. برای انجام چنین کاری، ما SFG را افزایش می دهیم به نحوی که هر گره NF حالتمندی (برای مثال $< H_1, FW, N/W >$) دارای یک بیت باشد که نشان می دهد آیا در قطعه کنونی روایی سازی فعال است یا خیر. کنترل SFC در مراحل متعدد رخ می دهد.

شروع: برای شروع SFG، ما یک قطعه از SFG را ایجاد می کنیم که در آن همه NFها در حالت اولیه خود قرار دارند. شکل 5 نشان دهنده یک مثالی می باشد: در قطعه n، شعب سمت چپ در هر دو جهت فعال هستند درحالیکه شعب سمت راست غیرفعال می باشند.

عبور: بعد، با درنظرگیری قطعه شبکه، ما یک الگوریتم عبور گراف را اجرا کرده ایم که به شناسایی مسیرهایی می پردازد که سوالات را اثبات می کند برای مثال با پاسخ به سوال 1 در §III-E، ما بدنبال SFG برای همه مسیرها از A به B می پردازیم.

تحلیل قابلیت حصول: وجود چنین مسیری تعیین می کند که A می تواند با B صحبت کند ولیکن شرایط این امر رخ نمی دهد. برای تعیین شرایط، ما باید مسیر را بررسی کرده و مجموعه انتقالات حالت را شناسایی کنیم که

برای حرکت شبکه از قطعه 0 به یک قطعه دیگر N لازم است به نحوی که همه گره ها در مسیر تحت پوشش بین A و B فعالسازی بشود.

برای انجام چنین کاری، ما بدنبال همه گره های حالتمند در مسیر می باشیم. برای هر گره ای، ما کنترل می کنیم تا ببینیم آیا یک لبه انتقال حالت نقطه گذاری ورودی باعث اتصال این گره به گره حالتمند دیگری شده یا خیر. اگر چنین باشد، آنگاه سعی می کنیم که شرایطی را بیابین که باید برای فعالسازی این گره والد با آن روبرو شد (به محض فعالسازی، هم گره والد و هم گره اصلی فعال می شوند). ما یادآور می شویم که فرایند فعالسازی والد حالت بازگشتی دارد، چرا که خود والد ممکن است نیاز به فعالسازی داشته باشد. برای مثال، در شکل 5 کنترل H7 در DAG سمت چپی نیاز به فعالسازی گره خلفی دارد. برای انجام این کار الگوریتم تحلیل استاتیک از لبه انتقال حالت به گره والد تبعیت می کند $< H_1, FW, N/W >$ و تعیین می کند که e1 برای فعالسازی لازم است (به محض بکارگیری e1، دو گره سیاه فعالسازی می شوند).

```
Verification(source, destination){
  EventList = []
  foreach SFG in SFG List
    If SFG:root == source
      verify(SFG, destination)
}

Verify(SFG, dst){
  EventList = []
  Paths = BreadthFirstSearch(SFG.root, dst)
  foreach path in Paths
    foreach node in path
      If node:isNotActive
        EventList.add(SFG.Activate(SFG, node))
  return EventList }

Activate(SFG, node){
  EventList = []
  foreach ancestor in node:getParents
    If ancestor:ContainsStateEventTransitionTo(node)
      EventList.add(ancestor.getStateTransitionTo(node))
      EventList.add(Verify(SFG, ancestor))
  return EventList
}
```

شکل 6-کد کاذب برای پیمایش SFC : سوال 1.

برای فعالسازی گره ها، کنترل کننده رویداد مناسبی را به SFG بکار می گیرد و در انجام چنین کاری یک قطعه جدید از شبکه را ایجاد می کند. هنگام فعالسازی یک گره حالت برای FW، کنترل کننده باید تضمین کند که گره های حالت دیگر برای FW غیرفعال شده اند. این فرایند بکارگیری یک رویداد به قطعه کنونی و فعالسازی و غیرفعالسازی گره ها ایجاد یک قطعه جدید می کند. برای مثال، در شکل 5 ما رویداد e1 را به قطعه N بکار بسته ایم که باعث خاموش شدن گره های خاکستری شده و گره های نارنجی را فعالسازی می کند.

مثال: ما از شکل 4 به عنوان مثالی برای پیمودن کل الگوریتم استفاده می کنیم. با فرض اینکه بسته ها از A به B در فضای سرایند H7 می باشند، ما ابتدا از یک جستجوی اول عرضی برای یافتن دو مسیری که H7 پیموده

استفاده می کنیم. بعد ما مسیری را می یابیم که A را به B وصل می کند که حاوی گره های $< H_1, SW_3, - >, < H_1, FW, E >, < H_1, SW_0, - >$ می باشد. ما e1 را به EventList می افزاییم به این معنا که برای اینکه مسیر فعالسازی بشود، e1 باید رخ بدهد. بعد از اینکه ما همه مسیر و همه گره ها را با لبه نقطه گذاری وارده پیمودیم، به فهرست لینک EventList بر می گردیم.

قابلیت ارتقای الگوریتم: اجازه دهید یک شبکه ای را با n مسیر و طول مسیر متوسط l فرض نماییم. تعداد m ی NF روی هر مسیر وجود دارد که هر یک دارای k حالت می باشد. آنگاه اندازه SFG برابر با $n(l + km)$ می باشد.

د-بهینه سازی کارایی الگوریتم

مرتب سازی رویداد: همانگونه که قبلا بحث گردید، طی تحلیل قابلیت حصول، فرایند فعالسازی یک گره مستلزم بررسی و احتمالا فعالسازی برگشتی سایر گره ها می باشد. ترتیب پیمایش تعیین کننده تعداد گره هایی است که برای فعالسازی برگشتی به آن نیاز داریم (این امر به نوبه خود بر زمان اجرای الگوریتم ما اثر بد دارد). ما از دانش ویژه دومین برای تعیین اینکه کدام گره والد دارای یک مجموعه کوچکتری از وابستگی های احتمالی است، استفاده می کنیم. برای مثال، یک گره والد که نیاز به رویداد بسته SYN دارد دارای یک وابستگی کوچکتری نسبت به گره والدی دارد که به درخواست HTTP نیاز دارد (چرا که دومی نیاز به سه بسته برای مثال SYN و SYN-ACK و ACK+Request دارد).

نمایش سمبولیک:

یک NF ممکن است سرایند بسته را به مقدار دامنه ای اصلاح کند که در آن مقدار واقعی تنها در زمان اجرا انتخاب خواهد شد. برای مثال، یک NAT یک پورت را به یک جریانی در زمان اجرا اختصاص خواهد داد، ازا ینرو در تحلیل استاتیک، ما تنها دامنه پورت را خواهیم دانست. در این خصوص، ما از یک متغیر سمبلیک استفاده می کنیم که نمایانگر پورت اصلاح شده می باشد. متغیر با یک دامنه متغیر همراه است. فرایند تحلیل قابلیت حصول طبق هر ارزش احتمالی کنترل انجام می دهد. این رویکرد به کاهش تعداد گره ها در SFG کمکم می کند.

هـ- سوالات متعدد

ما الگوریتم هایی را تدوین کرده ایم تا از چهار تا سوالات مختلف پشتیبانی کنیم. سوالات مشابه در زمینه شبکه بی حالت بررسی شده اند. ما سوالات مشابهی را با بسط حالتمند مطرح کرده ایم. به دلیل فضای محدود، تنها کدکاذب اولین سوال را نشان می دهیم.

-سوال 1: تحت چه سناریویی، همه بسته های A می توانند به B برسند؟ الگوریتم برای این سوال در شکل 6 نشان داده شده است. این الگوریتم از یک جستجوی اول عرضی در هر قطعه SFG استفاده می کند و گراف را با استفاده از عملکرد فعالسازی برای هر رویداد اصلاح می کند.

-سوال 2: با درنظرگیری یک توالی بسته/رویداد، آیا A می تواند با B صحبت کند؟ این امر بسته ها و رویدادهایی را (برای مثال وقفه ها) را یک به یک تزریق می کند، و از یک جستجوی اول عمیق برای یافتن مسیر استفاده کرده و در صورتی که بسته کنونی انتقال را تحریک کند، حالاتی را فعالسازی می کند

-سوال 3: کدام زنجیره های خدماتی است که یک توالی بسته خواهد پیمود؟ این امر توالی بسته را تزریق می کند، اولین جستجوی عمیق را در هر قطعه اجرا می کند و همه مسیرها را ثبت می کند. این مورد زنجیره های خدمات چندگانه را گزارش می دهد اگر زنجیره در میانه جریان تغییر نماید.

-سوال 4: آیا هر گونه جریانات و توالی های بسته وجود خواهد داشت که باعث شود حالت m ی NF X و حالت nی NF Y با هم بوجود آیند؟ تحلیل قابلیت حصول یک جستجوی اول عرضی را اجرا می کند تا همه مسیرهایی که هر یک از این دو حالت را می پیمایند بیابد و شناسایی کند که آیا همپوشانی وجود دارد یا خیر.

قسمت چهارم-پیش نمونه و ارزیابی

ما یک پیش نمونه کنترل کننده SFC را در تقریبا 2279 خط از برنامه جاوا ساخته ایم. پیش نمونه ما قوانین مسیریابی را به شکل ورودی از Mininet می گیرد تا توپولژی شبکه را ایجاد نماید. ما به ارزیابی پیچیدگی زمانی کنترل کننده SFC روی هم توپولوژی خطی و هم یک توپولوژی درخت ریشه دار منفرد سطح k با تعداد کل 2^k کلید پرداخته ایم. برای ارزیابی قابلیت ارتقای آن، ما یک NF عمومی را اجرا می کنیم که می توانیم تعداد حالات را تغییر بدهیم. ما همچنین این امر را روی یک بستر آزمون با چهار نوع NF واقعی ارزیابی کرده ایم.

اصول سنجش: برای درک قابلیت ارتقای کنترل کننده SFC، ما این را روی دو اصول سنجش ارزیابی کرده ایم (زمان ساخت SFG و زمان کنترل SFC یعنی زمان پیمایش SFG) ما به تحلیل دو بعد پرداخته ایم: 1) تعداد حالات به ازای NF و 2) تعداد NFهای حالتمند به ازای هر زنجیره خدماتی.

زمان ساخت SFG: شکل 7 نشان دهنده زمانی برای ساخت SFG با اندازه توپولوژی مختلف روی یک توپولوژی خطی (بالا) و با NFهای پیچیده تر (پایین) می باشد. درکل، زمان ساخت کوچک است: برای یک شبکه متشکل از 200 گره (هر گره دارای 1 حالت است) زمان برابر با 100 ms می باشد. SFG حاصله حاوی 11938 گره می باشد و تقریبا حافظه 45MB را مصرف می کند. وقتی تعداد حالات به 30 افزایش پیدا کرد، در یک توپولوژی 10 گره ای، زمان ساخت SFG برابر با 310ms می شود. هر دو ازمایشات دارای واریانس کوچکی می باشند.

زمان کنترل SFC: شکل 8 ترسیم متوسط زمان بررسی روی 100 اجرای مختلف است حین اینکه اندازه توپولوژی برای چهار سوال در پاراگراف سوم افزایش می یابد. زمان تحلیل قابلیت حصول بستگی به طول زنجیره خدمات واقعی دارد. سوال 4 نشان داده نشده است چرا که هر گره تنها یک حالت در این ازمایش دارد. سوال 1 گرانترین خواهد بود چون همه مسیرها و حالات ممکن را جستجو می کند. زمان از 2 ms تا 12 ms می باشد. شکل 9 نشان دهنده این است که زمان کنترل برای سوال 1 و 4 به وضوح با تعداد حالات در هر NF همبستگی دارد ولیکن نه برای دو سوال دیگر.

جدول 2-نتایج اجرای SFC مختلف

SFC Implementation	Avg. rules	SFG construction	SFC checking
ODL SFC L2 [17]	3102	7.3ms	14.9ms
ODL SFC NSH [18]	2716	5.5ms	14.8ms
ContexNet [19]	10241	7.9ms	15.1ms

در سوال 1-زمان کنترل از 8ms به 1s تغییر می کند ضمن اینکه حالات پیش NF از 2 به 27 تغییر می کند. وقتی که حالات پیش NF به 52 رشد یافت، حدود 20S وقت می برد تا سوال 1 پاسخ داده شود و حدود 0.2s هم برای سوال 4 زمان برده می شود. در بیشتر موارد، تعداد حالات به ازای هر NF به ازای هر جریان زیر 20 می باشد، که به معنای زمان کنترل کمتر از ثانیه است. در کنار زمان ساخت SFG ده میلی ثانیه ای، ما قابلیت ارتقای کنترل کننده SFC را با یک شرایط واقع گرایانه نشان داده ایم. ما همچنین با زنجیره خدمات 4-NF روی یک

توپولوژی درخت دارای SFG با 11k گره ازمایش انجام داده ایم و زمان کنترل کمتر از 200ms بوده است. این مزیت سرعت از این موارد ناشی می شود که از این قرارند: 1) ترکیب جریانات و حالات به رده های معادل، 2) الگوریتم ساخت SFG که مکررا گره ها را از بالا به پایین تقسیم بندی می کند به جای اینکه تعداد زیادی گره ها را از پایین به بالا ترکیب کند و 3) الگوریتم کنترل کننده که پردازش کننده رویدادهایی است که برحسب دانش حوزه مرتب سازی شده است.

ارزیابی بستر ازمون: ما به ارزیابی کنترل کننده SFC روی ContextNet پرداخته ایم که یک سیستم SFC تجاری است که در آزمون های شبکه های موبایل و در بسیاری گردهمایی های صنعتی بکار می رود. این مورد از توپولوژی Gi-LAN با 4 کلید و 15 NF شامل کنترل های والدی، بهینه سازی ویدئویی، غنی سازی سرایند HTTP ، اصول تحلیل داده های بزرگ، و دیوار اتشین تقلید می کند. این مورد از OpenFlow و کنترل کننده OpenDayLight استفاده می کند. ما مدلهایی را برای این 15 تا NF با تجزیه پیکربندی آنها ایجاد کرده ایم و قوانین OpenFlow را به شکل ورودی کنترل کننده SFC اقتباس کرده ایم. کنترل کننده SFC قادر به کنترل قابلیت حصول در عرض 15ms می باشد. SFG در عرض 8ms با اندازه 112 گره ها ساخته شده است.

پشتیبانی از اجرای مختلف SFC: ما کنترل کننده SFC را با انواع اجراهای SFC درجه صنعتی مختلف تست کرده ایم تا قابلیت کاربرد آن را در عمل نشان بدهیم. نتایج در جدول 2 امده است. اجرای مختلف منجر به تعداد مختلف قوانین در کلیدها می شود و با اینحساب دارای اثر اندکی روی زمان ساخت SFG می باشد ولیکن روی زمان کنترل SFC چنین نیست.

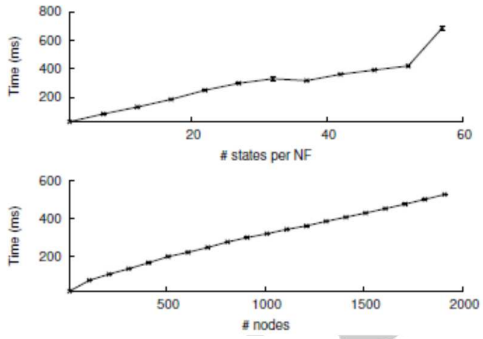
5-کارهای مرتبط

مدلسازی جعبه میانی-اقتباس NF ما ملهم از خط طولانی تحقیقاتی برای درک و مدلسازی NF ها می باشد که در رفرانسهای 7 و 9 و 16 و 21 الی 26 آمده است. بویژه مقاله 22 و 16 نیز به مدلسازی حالات داخلی NFها پرداخته اند ولیکن انها برای منظور مهاجرت طراحی شده اند. کار ما متمرکز بر اقتباس فورواردسازی NF برای منظور تحلیل فورواردسازی می باشد.

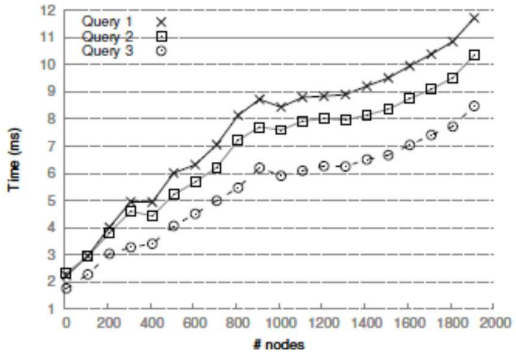
SDN حالتمند-اخیرا پیشنهادات تحقیقاتی وجود دارد که باعث شده صفحه کاربری SDN قابل برنامه ریزی و حالتمند بشود.رفرانس 4 افزودن حالت ساده را به OpenFlow مطرح کرده و رفرانس 5 هم یک صفحه داده های

قابل برنامه ریزی حالتمند را مطرح کرده است. SFA در رفرانس 27 نیز یک اجرای سخت افزاری اقتباس فورواردسازی حالتمند را مطرح کرده استا و کاملا در SDPA ارزیابی نکرده است. درحالیکه این کار متمرکز بر قابلیت برنامه ریزی و تحقق چنین صفحه مشترک حالتمندی می باشد، کنترل کننده SFC اشکارا به مدلسازی انتقال بین حالات می پردازد، و اقدامات را روی بسته ها بر حسب جریانات متمایز می کند و با استفاده از NFهای واقعی توضیح می دهد.

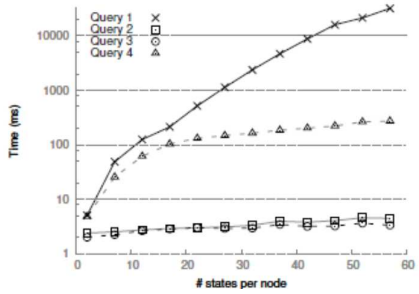
عیب یابی جعبه میانی-کار ما خیلی به BUZZ مرتبط است که FSM را از کد منبع NF می سازد و بعد بسته های ازمون گیری را براساس FSM تولید می کند.



شکل 7-زمان ساخت SFG



شکل 8-زمان کنترل SFC (اندازه های شبکه مختلف)



شکل 9-زمان کنترل SFC (حالات NF مختلف)

کار ما عمودی است چرا که ما بر تحلیل قابلیت حصول در سطح شبکه به جای هر NF منفرد تمرکز کرده ایم. ضمن اینکه FSM ساخته شده از روی BUZZ می تواند در کنترل کننده SFC هم استفاده بشود، می تواند همچنین مدلهای سطح بالای NF را به خود بگیرد و صحت پیکربندی ها را تایید کند. تازه ترین کار (رفرانس 7) یک زبان مدلسازی را مطرح کرده است و از یک حل کننده SAT برای کنترل خصوصیات جداسازی استفاده کرده است. قابلیت ارتقای این روش به دلیل حل کننده SAT محدود می باشد. حل کننده SAT به وضوح از سوالات حالتمندی که کنترل کننده SFC تایید می کند، پشتیبانی نمی کند. سایر تایید شبکه در مقالات 2 و 1 و 6 و 28 انحصارا متمرکز بر وسایل شبکه لایه 2 و لایه 3 می باشد (با نادیده گرفتن وسایل حالتمندی مانند NFها). بهبودهای اخیر SymNet از اجرای سمبلیک روی جعبه های میانی حالتمند استفاده می کند ولیکن مستلزم نوشتن NFها با استفاده از زبان SEFL آنها می باشد.

6-نتیجه گیری

در این مقاله، ما کنترل کننده SFC را مطرح کرده ایم که یک چارچوب تشخیص شبکه است که صحت رفتارهای فوروارد‌سازی یک زنجیره سازی عملکرد خدمات را در شبکه کنترل می کند. ما ابتدا یک مدل اقتباسی ساده را مطرح کرده ایم که حالات سطح بالایی که یک NF حفظ می کند، کسب کرده است. عملکرد یک NF بستگی به سرایند بسته و حالات داخلی آن دارد. با استفاده از مدل، ما یک الگوریتمی را مطرح کرده ایم که به کنترل رفتارهای فوروارد‌سازی شبکه تحت سناریوهای پویای مختلف می پردازد: یعنی توالی های بسته مختلف و حالات NF حاصله آن. کنترل کننده SFC یک الگوریتم را برای ترکیب کارامد فضای جریان معادل و فضای حالت معادل بکار می گیرد. ضمن اینکه نتایج اولیه نویدهایی را نشان می دهد، ما برنامه ریزی کرده ایم که قابلیت کارایی مدل اقتباسی خودمان را روی NFهای واقعی بیشتری روایی سازی کنیم و از انها برای ارزیابی بیشتر کنترل کننده SFC استفاده کنیم. ضمن اینکه پیکربندی استاتیک برای بررسی خطاهای پیکربندی مفید است، همچنین ما برنامه ریزی کرده ایم که این کار را به تایید زمان واقعی با کشاندن حالات از NFها به شکل واکنشی بسط دهیم.

References

[1] A. Khurshid, X. Zou, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Proc. NSDI*, 2013.

[2] P. Kazemian, G. Varghese, and N. McKeown, "Header space analysis: Static checking for networks," in *Proc. NSDI*, 2012.

[3] S. Zhu, J. Bi, C. Sun, and C. Wu, "Sdpa: Enhancing stateful forwarding for software-defined networking," in *Proc. International Conference on Network Protocols*, 2015.

[4] M. Moshref, A. Bhargava, A. Gupta, M. Yu, and R. Govindan, "Flow-level state transition as a new switch primitive for sdn," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, 2014.

[5] G. Bianchi, M. Bonola, A. Capone, and C. Cascone, "Openstate: Programming platform-independent stateful openflow applications inside the switch," *SIGCOMM Comput. Commun. Rev.*, vol. 44, April 2014.

[6] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, nsdi'13, 2013.

[7] A. Panda, O. Lahav, K. Argyraki, M. Sagiv, and S. Shenker, "Verifying isolation properties in the presence of middleboxes." http://arxiv.org/abs/1409.7687.

[8] "Bluecoat." https://bto.bluecoat.com.

[9] D. Joseph and I. Stoica, "Modeling middleboxes," *Netwrk. Mag. of Global Internetwkg.*, 2008.

[10] R. Stoenescu, M. Popovici, L. Negreanu, and C. Raiciu, "Symnet: Static checking for stateful networks," in *Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, HotMiddlebox '13, 2013.

[11] S. K. Fayaz, T. Yu, Y. Tobioka, S. Chaki, and V. Sekar, "Buzz: Testing context-dependent policies in stateful networks," in *Proc. NSDI*, 2016.

[12] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker, "Modular SDN Programming with Pyretic," *USENIX*, vol. 38, October 2013.

[13] A. Panda, O. Lahav, K. Argyraki, M. Sagiv, and S. Shenker, "Verifying isolation properties in the presence of middleboxes." Technical report: arXiv preprint:1409.7687,2014.

[14] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: network processing as a cloud service," in *Proc. ACM SIGCOMM*, pp. 13–24, 2012.

[15] "Mininet: An Instant Virtual Network on your Laptop." mininet.org/.

[16] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, 2014.

[17] "The Service Function Chaining (SFC) OpenFlow Renderer (SFCOFL2)." https://github.com/opendaylight/sfc/tree/master/sfc-py/sfc/nsh.

[18] " OpenDaylight sfc gerrit project." https://github.com/opendaylight/docs/blob/stable/beryllium/manuals/user-guide/src/main/asciidoc/sfc/odl-sfcofl2-user.adoc.

[19] H. ConteXtream, "Introduction to ContextNet." http://www8.hp.com/h20195/v2/GetPDF.aspx/c04725726.pdf.

[20] A. Noy and G. Mainzer, "Carrier Use Cases With OpenDaylight." OpenDayLight Summit 2015.

[21] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "Pga: Using graphs to express and automatically reconcile network policies," in *Proc. ACM SIGCOMM*, 2015.

[22] S. Rajagopalan, D. Williams, H. Jamjoom, and A. Warfield, "Split/merge: System support for elastic execution in virtual middleboxes," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation*, 2013.

[23] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, 2013.

[24] M. Dischinger, M. Marcon, S. Guha, K. P. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost: Enabling end users to detect traffic differentiation," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, 2010.

[25] M. T. Arashloo, Y. Koral, M. Greenberg, J. Rexford, and D. Walker, "Snap: Stateful network-wide abstractions for packet processing," in *SIGCOMM*, 2016.

[26] J. McClurg, H. Hojjat, N. Foster, and P. Černý, "Event-driven network programming," in *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 369–385, ACM, 2016.

[27] S. Zhu, J. Bi, and C. Sun, "Sfa: Stateful forwarding abstraction in sdn data plane," in *Presented as part of the Open Networking Summit 2014 (ONS 2014)*, 2014.

[28] W. Zhou, D. Jin, J. Croft, M. Caesar, and P. B. Godfrey, "Enforcing customizable consistency properties in software-defined networks," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, NSDI'15, 2015.

[29] R. Stoenescu, M. Popovici, L. Negreanu, and C. Raiciu, "Symnet: Scalable symbolic execution for modern networks," in *Proc. ACM SIGCOMM*, 2016.