شبکه نرم‌افزار محور (نرم‌افزاری تعریف شده)

2019 19th international conference on Sciences and Techniques of Automatic control & computer engineering (STA), Sousse, Tunisia, March 24-26, 2019

STA2019_Paper_81_TCE

Software Defined Networking

Hend Abdelgader Eissa
Department of Computing and Informatics
Faculty of Electronic Technology
Tripoli, Libya
namarek2010@gmail.com

Kenz A. Bozed
Benghazi University , Faculty of Information Technology, Department of Computer System Design
Benghazi, Libya
kenz.bozed@uob.edu.ly

Hadil younis
Department of Computing and Informatics
Faculty of Electronic Technology
Tripoli, Libya
hadilyounes19@gmail.com

ترجمه

حامد عبدالراق

https://github.com/hamed-abd

شماره دانشجویی: ۹۴۹۷۲۸۳۲۸

چکیده:

این مقاله مفهوم شبکه نرم‌افزار محور (SDN) را بررسی می‌کند، که واسط جنوبی آن می‌تواند از طریق پروتکل Open Flow اعمال شود. هدف از این مطالعه، کشف معماری SDN و استاندارد Open Flow با برخی جزئیات است. علاوه بر این، به دنبال اجرای ابزارهای خود در شبکه شرکت نفت هستند. شبیه‌سازی با استفاده از ترکیب شبیه‌ساز شبکه Mininet و کنترل‌کننده Ryu با استفاده از اسکریپت‌های پایتون انجام می‌شود.

۱- مقدمه

طراحی معماری شبکه‌های سنتی، به کنترل توزیع و پروتکل‌های شبکه حمل و نقل بستگی دارد که در روترها و سوئیچ‌هایی که بسته‌ها را هدایت می‌کنند، اجرا می‌شود و به آن‌ها اجازه می‌دهد تا حرکت کنند. این ادغام بین صفحه کنترل و صفحه داده، مدیریت شبکه‌ها را دشوار می‌کند [۱]. SDN یک چارچوب مدل پویا، کارآمد و قابل انعطاف است و به‌عنوان راه‌حل خوبی برای تولید پهنای باند بالا و ماهیت اجرای امروزی می‌باشد. این معماری توابع کنترل و انتقال شبکه را از هم جدا می‌کند، و کنترل شبکه را قادر می‌سازد تا به برنامه‌نویسی تبدیل شوند و زیرساخت‌های اساسی برای خدمات شبکه به‌صورت انتزاعی گردند. پروتکل Open Flow از SDN پیروی می‌کند و کنترل قابل برنامه‌ریزی جریان‌ها را به مدیران شبکه می‌دهد که به کنترل‌کننده اجازه می‌دهد مسیری را که جریان از مبدا به مقصد از نظر توپولوژی شبکه طی می‌کند، تعیین نماید و از جریان مبتنی بر پردازش برای بسته‌های حمل و نقل استفاده می‌کند. Open Flow یک رابط استاندارد می‌باشد که برای SDN طراحی شده‌است و در بین توسعه‌دهندگان و صنعت‌گران شبکه بسیار مورد توجه قرار گرفته‌است. جداسازی صفحه‌های کنترل و داده، دستگاه‌های هدایت را به سوئیچ dump تبدیل می‌کند، که در آن منطق کنترل در یک کنترل‌کننده متمرکز یا سیستم عامل شبکه جداگانه قرار دارد. در SDN، یک یا چند ماشین کنترل‌کننده با جمع‌آوری مجموعه‌ای از قوانین حمل و نقل بسته، یک برنامه با هدف عمومی را برای پاسخگویی به رویدادهایی مانند تغییر در توپولوژی شبکه، ایجاد ارتباطات توسط کاربران نهایی و تغییر در بار ترافیک انجام می‌دهند. کنترل‌کننده‌ها از طریق پروتکلی

مانند Open Flow قوانین را به سوئیچ‌ها سوق می‌دهند. سپس سوئیچ‌ها عملکرد خود را با استفاده از سخت‌افزار پردازش بسته به‌طور کارآمد پیاده‌سازی می‌کنند [۱ ، ۲].

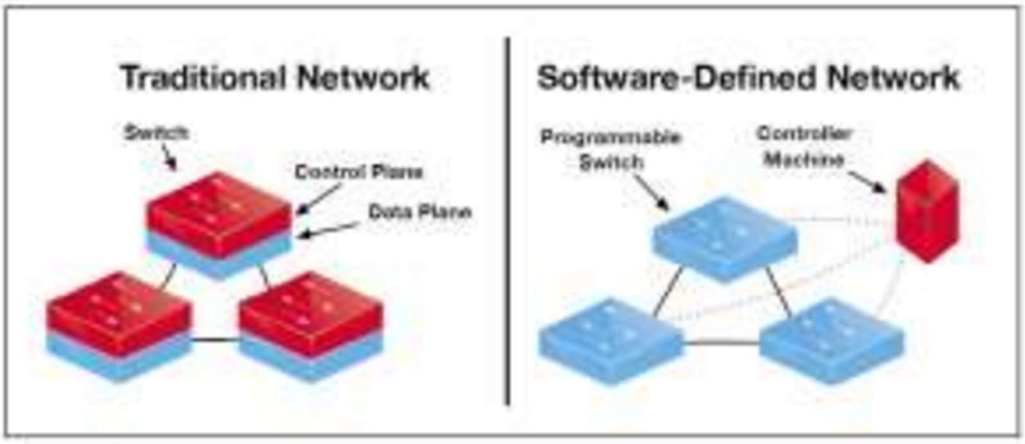
کمک اصلی این مقاله عبارتند از:

- مطالعه معماری شبکه نرم‌افزار محور و اصول نهفته در طراحی آن.
- مطالعه مشخصات OpenFlow با جزئیات همراه با کنترل‌کننده Ryu و پیام‌های خاص رد و بدل شده بین آن‌ها.
- استفاده از محیط نرم‌افزار کنترل‌کننده Ryu برای ساخت یک برنامه کاربردی شبکه در حال اجرا با یک شبکه Mininet. شبیه‌سازی مزایای استفاده از معماری SDN را نشان می‌دهد.

۲- شبکه نرم‌افزار محور

شبکه نرم‌افزار محور (SDN)، جداسازی فیزیکی صفحه کنترل شبکه از صفحه حمل و نقل است، که در آن یک صفحه کنترل واحد چندین دستگاه را کنترل می‌کند. روش SDN برای مدیریت جریان‌های ترافیکی است که باید از زیرساخت‌ها و سیستم‌های ضمنی ترافیک جدا شوند. SDN صفحه کنترل را از صفحه داده جدا می‌کند، سپس SDN صفحه کنترل را ادغام می‌کند، به طوری‌که همانطور که در شکل ۱ آمده است، یک برنامه کنترل، عناصر صفحه داده بیشتری را کنترل می‌نماید [۳]. جدایی صفحه کنترل و صفحه داده، به‌صورت رابط برنامه‌نویسی برنامه کاربردی[۱] (API) بین دستگاه شبکه و کنترل‌کننده SDN تعریف می‌شود. پروتکل OpenFlow [۴] مثالی برای یک API می‌باشد. یک سوئیچ با رابط قابل‌برنامه‌ریزی، کنترل‌کننده را قادر می‌سازد تا ارتباط برقرار کرده و قوانین را بر روی سوئیچ تنظیم کند. سوئیچ OpenFlow می‌تواند مانند یک روتر، سوئیچ، دیوارهای آتش و مترجم آدرس شبکه رفتار کند [۳].

[۱] application programming interface



شکل ۱- معماری‌های سنتی و نرم‌افزار محور

ب) انتزاعات SDN

انتزاع برای تعریف واسط‌های مربوطه برای تشکیل یک سیستم مقیاس‌پذیر مدولار (پیمانه‌ای) استفاده می‌شود. یک سیستم مدولار که امکان استفاده مجدد از کد را فراهم می‌کند. پیاده‌سازی می‌تواند اصلاح شود، اما اگر واسط کاربری ثابت بماند، بر سایر قسمت‌های سیستم نرم‌افزاری تأثیر نمی‌گذارد. انتزاع‌ها مزایای زیادی برای ساخت یک سیستم نرم‌افزاری مقیاس‌پذیر دارند، مدولار بودن مبتنی بر انتزاع، لازم می‌باشد [۵]. انتزاع SDN مشابه سیستم-های رایانه‌ای سنتی است که با سیستم عامل اختصاصی، سخت‌افزار و نرم‌افزار در مدل لایه‌ای با قابلیت انتخاب ویژگی مناسب در هر لایه ادغام شده است. SDN با انتزاع مناسب برای صفحه کنترل همراه می‌باشد. سه ستون اصلی برای جدا کردن صفحه کنترل معرفی می‌شوند:

- **انتزاع صفحه حمل و نقل:** انتزاع صفحه حمل و نقل، پنهان کردن پیچیدگی اجرای آن از تصمیمات کنترل می‌باشد. از یک واسط باز برای کنترل دستگاه‌های شبکه استفاده می‌شود. این بدان معنی است که نیازی به نگرانی در مورد فروشنده خاصی نیست.

- **انتزاع وضعیت شبکه:** دلیل پیچیدگی مدیریت و کنترل شبکه‌های فعلی، الگوریتم‌های توزیع‌شده پیچیده‌ای مانند OSPF است. این ایده، الگوریتم‌های پیچیده را انتزاعی می‌کند و با یک نمایش کلی شبکه برای کنترل‌کننده می‌تواند توابع برنامه را ساده نماید. به جای اینکه اجازه دهد دستگاه‌های شبکه با یکدیگر ارتباط برقرار کنند، کنترل‌کننده SDN از پروتکل خاصی (به‌عنوان مثالOpen Flow) برای برقراری ارتباط با دستگاه‌های شبکه با اطلاعات مربوط به شبکه استفاده می‌کند تا "نمای" یا نقشه توپولوژی را ایجاد نماید. تنظیمات ارسال شده به روترها و سوئیچ‌ها به حمل و نقل بستگی دارد.
- **انتزاع صفحه کنترل:** کنترل‌کننده SDN رابط‌های برنامه‌نویسی برنامه کاربردی را فراهم می‌کند که توسط برنامه‌های کاربردی قابل دسترسی هستند. برنامه‌های کاربردی خارجی می‌توانند با استفاده از جاوا یا REST شبکه را توسط APIها دستکاری کنند. توسعه‌دهندگان می‌توانند بدون نیاز به نوشتن نرم‌افزاری برای پشتیبانی از چندین سخت‌افزار و نرم‌افزار فروشنده، شبکه را پیکربندی و کنترل کنند.

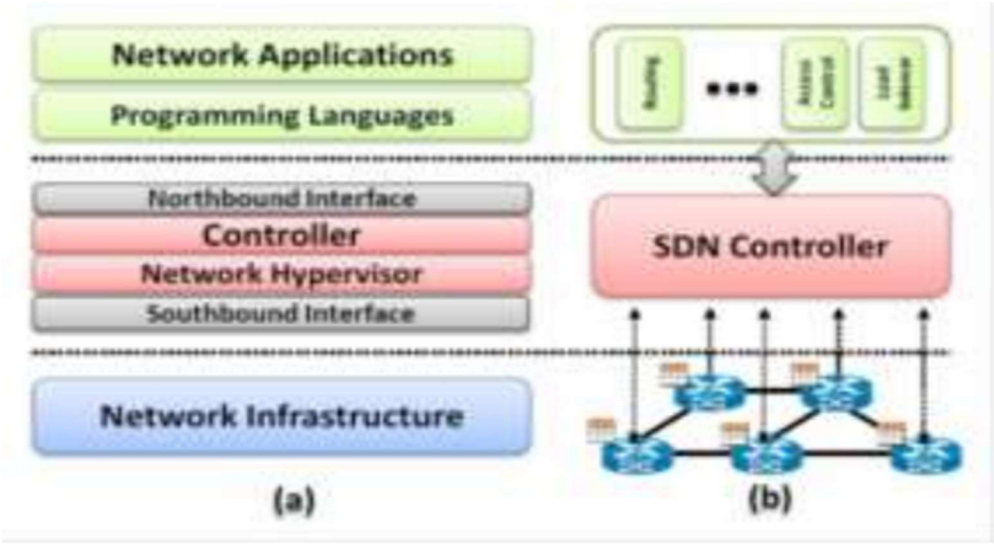
پس از اجرای انتزاع‌ها، کنترلر به‌عنوان یک سیستم عامل شبکه[1] (NOS) کار می‌کند و از طریق API معروف به API جنوبی با سوئیچ‌ها مذاکره می‌کند که در اینجا، برنامه‌های کاربردی کدهایی هستند که در کنترل‌کننده نوشته شده‌اند و از APIهای ارائه‌شده توسط سیستم عامل شبکه که API شمالی نام دارد، استفاده می‌کنند.

ج) لایه‌های SDN

معماری SDN را می‌توان به‌عنوان یک ترکیب هفت لایه تعریف کرد، همانطور که در شکل ۲ نشان داده شده‌است. هر لایه اهداف خاص خود را دارد. برخی از آن‌ها به‌طور مداوم در معماری SDN ارائه می‌شوند، مانند API جنوبی،

[1] Network Operating System

سیستم عامل‌های شبکه، API شبکه شمالی و برنامه‌های کاربردی شبکه. سایر موارد را فقط می‌توان با ترتیبات خاص، مانند هایپروایزر (ناظران)[1] یا زبان‌های برنامه‌نویسی معرفی کرد [۶ ، ۷]. شکل زیر این لایه‌ها را نشان می‌دهد.



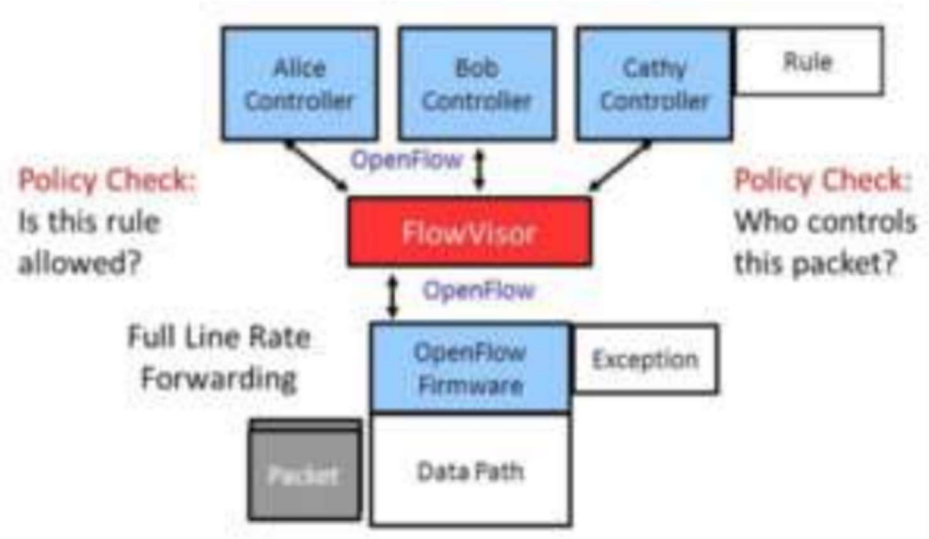
شکل ۲- شبکه‌های نرم‌افزار محور در (الف) لایه‌ها و (ب) معماری طراحی سیستم

- **زیر ساخت:** ابزارهای فیزیکی سنتی بدون کنترل یا برای تصمیم‌گیری، از اصول اساسی ساده پیشروی هستند. شبکه‌های جدید در بالای مرزهای استاندارد و باز ساخته شده‌اند تا سازگاری و قابلیت همکاری بین فروشندگان مختلف را تضمین کنند. علاوه بر این، واسط‌های باز موجودیت‌های کنترل‌کننده را قادر می‌سازند تا دستگاه‌های حمل و نقل ناهمگن را برنامه‌ریزی کنند، که در شبکه‌های سنتی دشوار می-باشد[۳].
- **واسط‌های جنوبی:** مرزهای جنوبی (SI) اتصالات ارتباطی بین ابزار کنترل و شبکه هستند، SI روش ارتباط بین دستگاه‌های پیشرو و صفحه کنترل را توضیح می‌دهد. این پروتکل روش تعامل عناصر صفحه

[1] hypervisor

کنترل و اطلاعات را ایجاد می‌کند. از طرف دیگر، این APIها هنوز در عناصر پیشرفته زیرساخت فیزیکی یا مجازی زیرساخت‌ها ایمن هستند [۶].

- **هایپروایزران (ناظران) شبکه:** مجازی‌سازی شبکه، انتزاع شبکه‌ای را نشان می‌دهد که از تجهیزات فیزیکی زیرین جدا شده است. این اجازه می‌دهد تا چندین شبکه مجازی در بالای زیرساخت مشترک به کار گرفته شود، که در آن هر شبکه مجازی می‌تواند به جای شبکه فیزیکی ضمنی، توپولوژی خود را داشته باشد. Flow Visor تلاش اولیه برای مجازی‌سازی SDN بود. همانطور که در شکل ۳ نشان داده شده‌است، Flow Visor مرحله‌ای است که به‌عنوان جایگزینی بین کنترل‌کننده و ابزارهای شبکه عمل می‌کند و یک لایه مفهومی را ارائه می‌دهد که صفحه داده Open Flow را به اشتراک می‌گذارد و به کنترل‌کننده‌های متعدد اجازه می‌دهد تا قسمت خاص خود را کنترل کنند. وظایف اصلی Flow Visor این است که تصمیم بگیرد چه کسی بسته‌های پیشرفته توسط سوئیچ را کنترل می‌کند و قوانینی را که توسط کنترل‌کننده‌ها تنظیم می‌شود، بررسی و تنظیم کند [۶،۸].



شکل ۳- مدیریت پیام Flow Visor

- **سیستم عامل‌های شبکه:** NOS عنصر اصلی در معماری SDN است. مشابه سیستم عامل فعلی، کنترل‌کننده، جزئیات پروتکل کنترل‌کننده به دستگاه SDN را که برنامه‌های فوق قادر به برقراری ارتباط با آن دستگاه‌های SDN هستند، بدون دانستن تفاوت انتزاعی می‌کند. این کنترل متمرکز توسط NOS باید مدیریت شبکه را تسهیل کند و سربار حل مشکلات شبکه را ساده نماید. ویژگی‌های اصلی کنترل-کننده عبارتند از:

۱) **کشف دستگاه کاربر نهایی:** کشف دستگاه‌های کاربر نهایی مانند لپ‌تاپ، دسکتاپ، دستگاه‌های تلفن همراه و غیره.

۲) **کشف دستگاه شبکه:** کشف دستگاه‌های شبکه‌ای که شامل زیرساخت‌های شبکه هستند مانند سوئیچ‌ها، روترها و نقاط دسترسی بی‌سیم.

۳) **مدیریت توپولوژی:** حفظ اطلاعات مربوط به جزئیات اتصال دستگاه‌های شبکه به یکدیگر و به کاربر نهایی.

۴) **مدیریت جریان:** حفظ یک بانک اطلاعاتی از جریان‌های کنترل‌شده توسط کنترل‌کننده و انجام کلیه تنظیمات لازم با دستگاه‌ها جهت اطمینان از همگام‌سازی ورودی‌های جریان دستگاه با آن پایگاه داده.

- **واسط‌های جنوبی:** واسط‌های جنوبی انتزاعی هستند و به برنامه‌های شبکه اجازه می‌دهند تا برای ساده-سازی برنامه‌نویسی شبکه به پیاده‌سازی های خاص وابسته نباشند. برعکس واسط‌های جنوبی، واسط‌های شمالی، بیشتر یک سیستم نرم‌افزاری هستند که برنامه‌هایی مانند مسیریابی توسط زبان‌های برنامه‌نویسی مانند Python یا Java به صورت برنامه‌نویسی ساخته می‌شوند، این امر امکان توسعه سریع‌تر، هزینه‌های سرمایه‌گذاری پایین‌تر و عیب‌یابی آسان‌تر در مقایسه با API Southbound را فراهم می‌کند [۹]. کنترل-کننده، اعمال روال‌هایی که در شبکه اتفاق می‌افتد را مطلع می‌کند. وقایع ممکن است مربوط به یک بسته جداگانه باشد که توسط کنترل‌کننده یا تغییر وضعیت در توپولوژی ایجاد شده‌است، مانند قطع

شدن اتصال. برنامه‌های کاربردی در پاسخ به این رویداد از رویکردهای مختلف درخواست می‌کنند. این ممکن است شامل کاهش یا ارسال بسته در صورت یک رویداد بسته دریافت‌شده باشد.

- **زبان‌های برنامه‌نویسی:** زبان‌های برنامه‌نویسی، APIهای سطح بالایی هستند که مفهومی از خود شبکه را گسترش می‌دهند، بنابراین نیازی نیست که برنامه‌نویس به فکر دستگاه‌های جداگانه باشد بلکه کافی است کمی به‌طور کلی به شبکه بپردازد [۹]. Pyretic، Python و Frenetic در میان چندین زبان برنامه-نویسی دیگر برای SDN برنامه‌ریزی شده‌اند.
- **برنامه‌های کاربردی شبکه:** برنامه‌های کاربردی شبکه منطق کنترل را اجرا می‌کنند، که در دستگاه‌های شبکه ترجمه و تعریف شده‌اند و رفتار آن‌ها را تعیین می‌نمایند. برنامه کاربردی شبکه به‌عنوان "مغز شبکه" در نظر گرفته می‌شود و به‌عنوان شنونده برخی از رویدادها که قبلا مشخص شده‌است، ثبت می‌گردد، سپس کنترل کننده هر زمان که چنین رویدادی رخ می‌دهد، روش پاسخ تماس برنامه را فراخوانی می‌کند و همچنین آن‌ها را روی ورودی‌های خارجی مانند انجام روش‌های امنیتی اعمال می‌نماید [۹]. یک کاربرد ساده یعنی مسیریابی، منطق این برنامه می‌باشد و توصیف مسیری است که در آن بسته‌ها از نقطه A تا نقطه B اجرا خواهند شد. برای انجام این هدف، یک برنامه مسیریابی باید براساس سهم توپولوژی، در مسیر استفاده تصمیم‌گیری کند و به کنترل‌کننده آموزش دهد تا قوانین پیشرفت مربوطه در همه وسایل نقلیه در مسیر انتخاب‌شده از A تا B را متصل کند [۱۰]. علیرغم تنوع گسترده موارد استفاده، اکثر برنامه‌های SDN را می‌توان در یکی از پنج کلاس مهندسی ترافیک، تحرک و بی‌سیم، اندازه‌گیری و نظارت، امنیت و قابلیت اطمینان و شبکه مرکز داده جمع‌آوری کرد [۶].

۳- OPEN FLOW

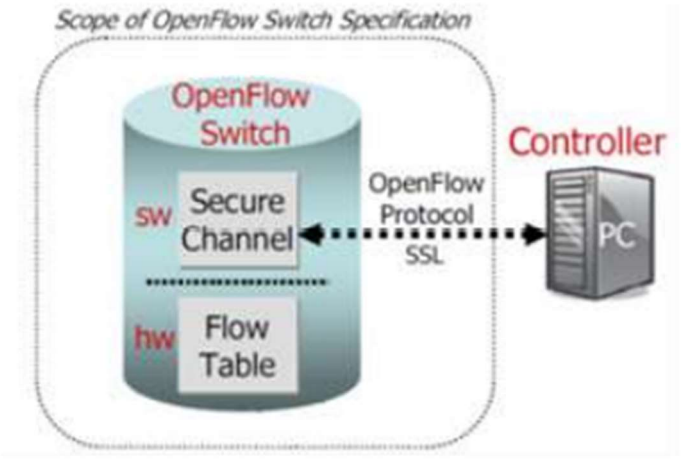
Open Flow گسترده‌ترین استاندارد باز API جنوبی برای SDN است. این استاندارد ویژگی مشترکی برای پیاده‌سازی ابزارهای مجهز به Open Flow و برای کانال ارتباطی بین داده‌ها و دستگاه‌های صفحه کنترل فراهم

می‌کند (به‌عنوان مثال، سوئیچ‌ها و کنترل‌کننده‌ها). پروتکل OpenFlow سه منبع اطلاعاتی برای سیستم عامل‌های شبکه فراهم می‌کند. ابتدا پیام‌های مبتنی بر رویداد هنگام تولید پیوند یا تغییر پورت، توسط دستگاه‌های انتقال به کنترل‌کننده، ارسال می‌شوند. دوم، آمار جریان توسط دستگاه‌های حمل و نقل تولید می‌شود و توسط کنترل-کننده جمع‌آوری می‌شود. سوم، پیام‌های بسته‌ای ورودی هنگامی‌که نمی‌دانند با جریان ورودی جدید چه کاری باید انجام بدهند، یا اینکه عملکرد صحیح "ارسال به کنترل‌کننده" در ورودی همسان جدول جریان وجود دارد، توسط دستگاه‌های حمل و نقل به کنترل‌کننده ارسال می‌شوند. این کانال‌های اطلاعاتی وسیله اساسی برای ارائه اطلاعات سطح جریان به سیستم عامل شبکه هستند [۹].

ب) سوئیچ OpenFlow

سوئیچ Open Flow حداقل شامل سه قسمت است: ۱) یک جدول جریان، با عملکرد مربوط به هر ورودی جریان، برای تغییر نحوه تمرین جریان، ۲) یک کانال امن که سوئیچ را به فرآیند کنترل از راه دور متصل می‌کند (کنترل-کننده نامیده می‌شود)، اجازه می‌دهد دستورالعمل‌ها و بسته‌ها با استفاده از یک کنترل‌کننده و سوئیچ ارسال شوند، ۳) پروتکل Open Flow که روشی باز و عادی برای اتصال کنترل‌کننده با سوئیچ ارائه می‌دهد. شکل ۴ نمونه‌ای از سوئیچ Open Flow را نشان می‌دهد [۱۰]. مفهوم اساسی ساده است: اکثر سوئیچ ها و روترهای اترنت هم دوره، شامل جداول جریان هستند که به صورت خطی برای اجرای دیوارهای آتش دستگاه، NAT، QoS، جهت جمع-آوری ارقام استفاده می‌شوند. با اینکه جدول جریان هر فروشنده متفاوت است، مجموعه مشترکی از اهداف وجود دارند که در بسیاری از سوئیچ‌ها و مسیریاب‌ها اجرا می‌شوند. Open Flow از این اهداف مشترک استفاده می‌کند.

ایده اصلی ساده است: اکثر سوئیچ‌ها و روترهای مدرن اترنت از جداول جریان استفاده می‌کنند که برای اجرای دیوارهای آتش، NAT، QoS و جمع‌آوری ارقام با نرخ خط اجرا می‌شوند. در حالی‌که جدول جریان هر فروشنده متفاوت است، یک هدف مشترک وجود دارد که در سوئیچ‌ها و روترهای متعددی اجرا می‌شود.



شکل ۴- سوئیچ Open Flow

منابع Open Flow این مجموعه اهداف عمومی را استخراج می‌کنند. سوئیچ‌های Open Flow در دو متنوع ارائه می‌شوند: سوئیچ‌های خالص (OpenFlowonly) که فاقد هرگونه ویژگی موروثی و کنترل داخلی هستند و برای تصمیم‌گیری در مورد حمل و نقل کاملاً به کنترل‌کننده اعتماد می‌کنند. و سوئیچ‌های ترکیبی (با OpenFlow فعال) که علاوه بر روند و رویه‌های منسوخ، از Open Flow پشتیبانی می‌کنند. سودآورترین تغییرات موجود امروزه ترکیبی هستند.

ج) جداول Open Flow

یک سوئیچ Open Flow شامل یک جدول جریان است، که به جستجوی بسته و ارسال دست می‌یابد. هر جدول جریان در تغییر، مجموعه‌ای از ورودی‌های جریان را نگه می‌دارد.

- **ورودی جریان:** یک جدول جریان شامل ورودی‌های جریان است. هر دسترسی به جریان شامل موارد زیر می‌باشد:

۱) **فیلدهای تطبیق:** برای مطابقت با بسته‌ها می‌باشد. این موارد شامل درگاه ورودی و هدرهای بسته است.

۲) **اولویت:** اولویت مربوط به ورودی جریان.

۳) **شمارنده‌ها:** برای به‌روزرسانی بسته‌های همسان.

۴) **دستورالعمل‌ها:** برای اصلاح مجموعه عملکرد یا پردازش خط لوله.

۵) **مهلت زمانی:** حداکثر مجموع زمان یا زمان بیکاری قبل از جریان توسط سوئیچ کاهش می‌یابد.

۶) **کوکی:** علامت برای فیلتر کردن ورودی‌های دقیق هنگام درخواست ارقام استفاده می‌شود، و هنگام پردازش بسته‌ها استفاده نمی‌شود.

- **تطبیق:** با دریافت یک بسته، سوئیچ Open Flow با انجام جستجوی جدول در اولین جدول جریان شروع به اهداف می‌کند و بر اساس پردازش خط لوله، ممکن است به جستجوی جدول در سایر جدول‌های جریان دست یابد. فیلدهای تطابق بسته از بسته حذف می‌شوند. اگر استانداردهای موجود در فیلدهای مطابقت بسته که برای جستجو در نظر گرفته شده، مطابقت داشته باشند، یک بسته با ورودی جدول جریان مطابقت دارد. اگر یک قسمت ورود به جدول جریان دارای هر فیلدی باشد که از آن حذف شده

باشد، تمام استانداردهای بالقوه موجود در هدر را در نظر می‌گیرد. اگر سوییچ از bitmasks دلخواه در زمینه‌های دقیق مطابقت پشتیبانی کند، این پوشش‌ها می‌توانند مطابقت دقیق‌تری داشته باشند.

- **شمارنده‌ها:** شمارنده‌ها برای جمع‌آوری ارقام فرآیندها و فرآیندهای نگهداری شده توسط سوئیچ OpenFlow استفاده می‌شوند. آن‌ها برای هر جدول جریان، ورودی جریان و درگاه حفظ می‌شوند.
- **اقدامات:** برای پردازش بسته‌ها، ورودی‌های جریان با یک عمل یا لیستی از اقدامات انجام شده دنبال می‌شوند. سوئیچ برای پشتیبانی از همه نوع اقدامات ضروری نمی‌باشد، و فقط برای مواردی که در زیر به‌عنوان "اقدام مورد نیاز" مشخص شده‌است، به‌کار می‌رود.

۱) **خروجی / رها کردن (الزامی):** عملکرد خروجی یک بسته را به یک پورت Open Flow مشخص هدایت می‌کند. سوئیچ‌های Open Flow باید از پیشرفت به پورت‌های فیزیکی، پورت‌های منطقی تعریف شده و پورت‌های رزرو شده مورد نیاز پشتیبانی کنند. Drop برای انداختن (رها کردن) بسته استفاده می‌شود.

۲) **تنظیم صف (اختیاری):** عمل تنظیم صف، شناسه خط را برای یک بسته تنظیم می‌کند. هنگامی‌که بسته با استفاده از عملکرد خروجی به درگاه منتقل می‌شود، شناسه صف کنترل می‌کند که کدام صف متصل به این درگاه برای زمانبندی و ارسال بسته استفاده شود. این عملکرد برای ارائه پشتیبانی پایه کیفیت سرویس (QoS) استفاده می‌شود.

۳) **گروه (الزامی):** بسته روی گروه مشخص شده انجام می‌شود.

۴) **Push-Tag/Pop-Tag (اختیاری):** سوئیچ‌ها ممکن است از قابلیت برچسب‌های push/pop پشتیبانی کنند. برای کمک به ترکیب با شبکه‌های فعلی، قابلیت برچسب‌های VLAN push/pop پیشنهاد می‌شود که پشتیبانی گردد.

۵) **تنظیم زمینه (اختیاری):** اقدامات متعدد تنظیم زمینه با توجه به نوع فیلد آن‌ها شناخته می‌شوند و مقادیر فیلدهای هدر خاص را در بسته تنظیم می‌کنند.

- **دستورالعمل‌ها:** هر ورودی جریان شامل مجموعه‌ای از دستورالعمل‌ها است که در هنگام مطابقت یک بسته با ورودی انجام می‌شود. این دستورالعمل‌ها در نتیجه تغییر در بسته، مجموعه اقدامات و / یا پردازش خط لوله است.

۴- طراحی SDN

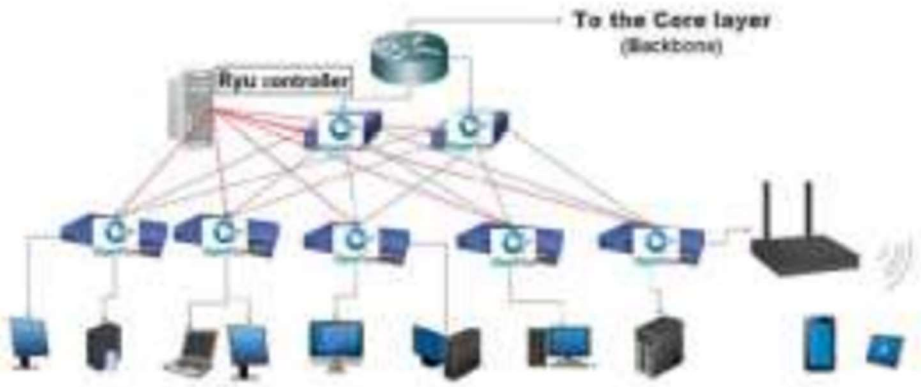
معماری SDN تا آنجا که دارای واسط‌های باز و قابل برنامه‌ریزی هستند، به یک کنترلر متصل به همه دستگاه‌های حمل و نقل نیاز دارد. Open Flow یک API جنوبی است که در اینجا استفاده شده‌است. بنابراین، تمام سوئیچ‌های موجود در لایه دسترسی و توزیع، سوئیچ‌های فعال شده Open Flow هستند. روتری که LAN را به ستون فقرات[۱] متصل می‌کند، می‌تواند یک روتر قدیمی باشد. پلت فرم Ryu به‌عنوان سیستم عامل و کنترل‌کننده شبکه حفره انتخاب شده‌است.

د) اشتراک‌گذاری بار SDN

برای استفاده از پیوندهای افزونگی و تقسیم بار بین سوئیچ‌ها، از پروتکل‌هایی مانند PVST استفاده می‌شود. اما پیکربندی به‌صورت ایستا انجام می‌شود. با استفاده از مفاهیم SDN، با اجرای یک الگوریتم بسته به میزان ترافیک آن‌ها، یک تقسیم بار پویا برای VLANها روی دو سوئیچ توزیع وجود دارد. الگوریتم برای ساخت یک برنامه کاربردی شبکه استفاده می‌شود که توسط کنترل‌کننده Ryu اجرا می‌شود [۱۱]. سوئیچ‌های Open Flow از مزایای استفاده از شمارنده‌های خود برای نظارت بر ترافیک و شمارش تعداد بسته‌های جاری و ورودی‌های جریان نصب‌شده در سوئیچ برخوردار هستند. پیام‌های رد و بدل شده بین Ryu و سوئیچ‌ها در اینجا برای درخواست اطلاعات شمارنده‌ها مورد استفاده قرار می‌گیرد. کنترل‌کننده از این اطلاعات در مورد ترافیک برای تغییر شکل

[۱] backbone

سوئیچ‌ها و ایجاد تعادل مجدد در ترافیک داخل شبکه بسته به بارهای فعلی VLAN استفاده خواهد کرد. شکل ۵، افزونگی اجرای پیشنهادی را در یک توپولوژی ساده شرکت نشان می‌دهد [۱۲].



شکل ۵- توپولوژی شبکه ساده‌شده

ب) شبیه‌سازی

در این شبیه‌سازی، یک شبکه کوچک به نمایندگی از شبکه شرکت با استفاده از نرم‌افزار Mininet ایجاد می‌شود. این شبکه از دو سوئیچ توزیع‌شده و شش سوئیچ دسترسی تشکیل خواهد شد. میزبان‌های Mininet ترافیکی را بین خود ایجاد می‌کنند، که در آن هر میزبان متعلق به یکی از هشت VLAN است که پیکربندی می‌شوند. برنامه کاربردی شبکه بر روی الگوریتمی خاص طراحی و اعمال می‌شود. این برنامه قرار است آزمایش شود و نتایج راه‌حل ارائه‌شده برای تقسیم بار را نشان می‌دهد.

ج) برنامه کاربردی شبکه

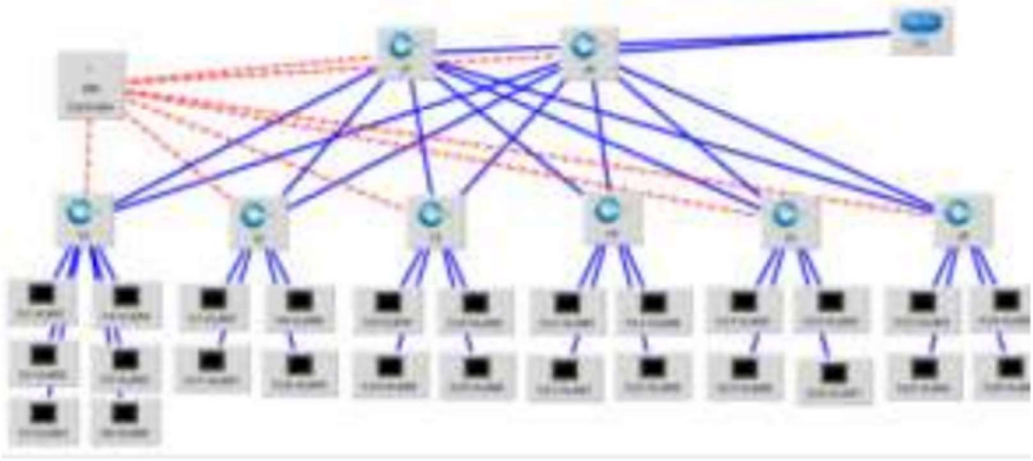
برنامه کاربردی شبکه برای استقرار الگوریتم زیر طراحی شده‌است:

- در ابتدا، VLANها بین دو سوئیچ توزیع‌شده تقسیم می‌شوند، یک سوئیچ به‌عنوان یک پل ریشه برای گروه VLANها عمل می‌کند، و سوئیچ دوم VLANهای باقیمانده را می‌گیرد. این مرحله با پروتکل PVST برابر است.
- پس از یک دوره زمانی خاص، کنترل‌کننده پیام Read-State را به پل‌های ریشه ارسال می‌کند و در مورد بارهای ترافیکی هر VLAN پرس و جو می‌کند. دو سوئیچ بلافاصله با پیغام‌های پاسخ متوالی، پاسخ می‌دهند.
- کنترل‌کننده از اطلاعات مربوط به ترافیکی که توسط سوئیچ‌های ریشه ارسال می‌شود، استفاده می‌کند تا VLANها را به همان اندازه و در آن‌ها تخصیص دهد.
- برای اعمال تخصیص‌های جدید سوئیچ‌های ریشه، کنترل‌کننده پیام‌های Modify-State را به سوئیچ‌ها می‌فرستد. این شامل حذف، افزودن یا اصلاح برخی از ورودی‌های جدول جریان است.
- روند به‌روزرسانی به‌صورت دوره‌ای اعمال می‌شود. برای جلوگیری از فرآیندهای طاقت‌فرسا در کنترل-کننده، مدت زمان نباید کوتاه باشد و همچنین برای پیگیری ترافیک غیرقابل عبور VLANها نباید خیلی طولانی باشند.

برنامه کاربردی شبکه به‌عنوان اسکریپت‌های Python در پرونده‌ای با نام " "Dynamic Utilize ذخیره شده در پوشه برنامه‌های کاربردی Ryu ایجاد می‌گردد.

د) شرح (توصیف) شبکه

توپولوژی شبکه کوچک به‌منظور تقریب استعدادهای دستگاه‌های شبکه مهندسی، کاربران و VLANها طراحی شده‌است. توپولوژی شبکه در شکل ۶ نشان داده شده‌است.



شکل ۶- توپولوژی شبکه شبیه‌سازی‌شده

۸ رابط VLAN برای ۲۶ میزبان متصل به سوئیچ‌های دسترسی پیکربندی شده‌است. VLANها به‌طور عادلانه بین میزبان‌ها توزیع می‌شوند تا گروه‌های کاربری را که در این آزمون هدف قرار داده‌اند را نشان دهند. جدول زیر شامل گروه‌های کاربری، شناسه VLAN آن‌ها، میزبان‌ها و آدرس IP زیر شبکه است. جدول ۱ گروه‌های کاربری، VLANها و آدرس‌های آن‌ها را نشان می‌دهد.

جدول ۱- گروه‌های کاربری و VLANS

User Group	VLAN ID	Hots	Network IP
Employees	1	h1-h7-h13-h25	10.0.1.0
Technical staff	2	h2-h8-h15	10.0.2.0
Web services	3	h3-h9	10.0.3.0
Events Management	4	h4-h10-h14-h26	10.0.4.0
IT and Networking	5	h5-h11-h16-h18	10.0.5.0
IT and Networking	6	h6-h13-h20	10.0.6.0
Treasury	7	H17-h21-h24	10.0.7.0
Visitors	8	H190h220h23	10.0.8.0

پ) پیکربندی‌های طراحی

یک محیط در Mininet ایجاد کنید. دستوری که باید وارد شود، همان دستورالعمل زیر است که اسکریپت پایتون از توپولوژی سفارشی را اجرا می‌کند. پس از اجرای دستور، یک شبکه کامل با سوئیچ‌ها، میزبان‌ها و پیوندهای مجازی OpenFlowenable ساخته شده‌است که در توپولوژی مشخص شده‌است. همانطور که در شکل ۷ نشان داده شده‌است. VLAN IDروی واسط هر میزبان تنظیم شده‌است. این شامل حذف آدرس IP است که به‌طور خودکار بر روی هر میزبان اختصاص داده می‌شود و یک آدرس IP جدید تنظیم می‌گردد.

```
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6633
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
h19 h20 h21 h22 h23 h24 h25 h26
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7 s8
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s2)
(h8, s2) (h9, s3) (h10, s3) (h11, s4) (h12, s4) (h13, s5) (h14,
s5) (h15, s6) (h16, s6) (h17, s2) (h18, s2) (h19, s3) (h20, s3)
(h21, s4) (h22, s4) (h23, s5) (h24, s5) (h25, s6) (h26, s6) (s1,
s7) (s1, s8) (s2, s7) (s2, s8) (s3, s7) (s3, s8) (s4, s7) (s4,
s8) (s5, s7) (s5, s8) (s6, s7) (s6, s8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18
h19 h20 h21 h22 h23 h24 h25 h26
*** Starting controller
c0
*** Starting 8 switches
s1 s2 s3 s4 s5 s6 s7 s8
*** Starting CLI:
mininet>
```

شکل ۷- فرمان ایجاد محیط شبکه

ف) سناریوی به اشتراک‌گذاری بار

چندین ترافیک مختلف به‌طور تصادفی بین میزبان‌ها ایجاد می‌شود. از ابزار Ping برای تولید چهار ترافیک استفاده می‌شود: A، B، C و D. هر ترافیک به‌طور متوالی دو بار ایجاد می‌شود، یکی قبل از به‌روزرسانی و دیگری بعد از آن. این امر نشان می‌دهد که چگونه می‌توان با تغییر مکان VLANها به سوئیچ‌های ریشه در بهبود استفاده از پیوندهای افزونگی کمک نمود. ترافیک در طی هشت دقیقه شبیه‌سازی ایجاد می‌شود.

- **VLANها:** این ستون تخصیص VLANها را برای هر سوئیچ ریشه در طی یک دوره زمانی که در آن ترافیک ایجاد می‌شود، مشخص می‌کند.

- **تعداد میزبان‌ها (H):** این ستون نشان‌دهنده تعداد میزبانانی است که به ایجاد ترافیک کمک می‌کنند. توجه داشته باشید که تعداد بیشتری از میزبان‌ها برای سوئیچ ریشه به میزان ترافیک بالاتری احتیاج ندارند.
- **زمان:** زمان دوره‌ای ۵ دقیقه انتخاب شده‌است؛ در این مدت ترافیک ایجاد می‌شود. در پایان زمان، کد برنامه کاربردی شبکه به کنترل‌کننده دستور می‌دهد تا از آمار ترافیک عملکرد سوئیچ‌های ریشه Ryu درخواست کند تا میزان ترافیک را به ناظر نشان دهد و استفاده از سوئیچ‌های ریشه را محاسبه نماید.
- **مقدار ترافیک (TRF):** این مقدار نشان می‌دهد که چه میزان از ترافیک از طریق سوئیچ عبور می‌کند و برابر با بسته‌های ارسال شده توسط سوئیچ است. ما از شمارنده‌های سوئیچ Open Flow استفاده کردیم که تعداد بسته‌های مطابق عبور داده‌شده از سوئیچ را می‌شمارد و آن را به‌عنوان پاسخ برای پیام درخواست aggregate_stats_request برای کنترل‌کننده ارسال می‌کند.
- **بهره‌وری**: هر سهم سوئیچ را از ترافیک کلی مشخص می‌کند.

$$UT(S1) = \frac{TRF(S1)}{TRF(S1)+TRF(S2)} \times 100 \qquad (1)$$

$$UT(S2) = \frac{TRF(S2)}{TRF(S1)+TRF(S2)} \times 100 \qquad (2)$$

مقدار مطلوب این ورودی ٪ ۵۰ است که وقتی هر سوئیچ ریشه دقیقاً نیمی از ترافیک را کنترل می‌کند، اتفاق می‌افتد. پس از به‌روزرسانی برای هر ترافیک، می‌توان پیشرفتی در این ستون مشاهده نمود؛ همانطور که در جدول ۲ نشان داده شده‌است.

جدول ۲- نتایج: ترافیک و بهره‌وری

Traffic pattern	Time (minutes)	Root-SW1 (S7)				Root-SW2 (S8)			
		VLANs	H	TRF	utilization	VLANs	H	TRF	utilization
Traffic A	5 m	1,3,5,7	11	83	%58	2,4,6,8	9	59	%42
	10 m	1,2,3,7	10	71	%50	4,5,6,8	10	71	%50
Traffic B	15 m	1,2,3,7	12	119	%71	4,5,6,8	9	48	%29
	20 m	1,5,3	10	83	%51	2,4,7,6,8	11	79	%49
Traffic C	25 m	1,5,3	8	47	%23	2,4,7,6,8	15	155	%77
	30 m	4,1,5,7	14	107	%53	2,8,6,3	9	95	%47
Traffic D	35 m	4,1,5,7	10	71	%60	2,8,6,3	8	47	%40
	40 m	4,5,7	8	59	%50	1,2,3,6,8	10	59	%50

با شروع شبیه‌سازی، کنترل‌کننده سوئیچ‌های دسترسی را با توجه به تخصیص اولیه با استفاده از تابع _init_ در کد پیکربندی می‌کند. وقتی VLANها، ۱،۳،۵،۷ باشند، ترافیک به Root-Switch-1 ارسال می‌شود، و Root-Switch-2، VLANهای باقیمانده ۲،۴،۶،۸ را می‌گیرد. ترافیک ایجاد شده بارهای متفاوتی را در سوئیچ‌های ریشه ایجاد می‌کند زیرا ترافیک VLANها برابر نیست. پس از ۵ دقیقه، کنترل‌کننده با استفاده از توابع: send، _aggregate ، _stats ، _request و aggregate ، reply و handler، آمار ترافیک موجود در هر سوئیچ ریشه را جمع‌آوری کرد. پس از به‌روزرسانی، استفاده از آن بهبود یافته‌است و اکنون هر سوئیچ ریشه دقیقاً نیمی از ترافیک (۵۰٪) را به خود اختصاص می‌دهد که تعادل مطلوب بار را نشان می‌دهد.

۵- نتیجه‌گیری

این کار بررسی ادبیات SDN را ارائه می‌دهد. شرح کاملی از ساختار SDN و اجزای اصلی آن، از جمله استاندارد Open Flow، شبکه نرم‌افزار محور، مدیریت شبکه را آسان می‌کند، نوآوری را تسریع می‌کند، هزینه‌ها را کاهش می‌دهد و برنامه‌نویسی را در شبکه‌ها فعال می‌نماید. شبیه‌سازی شبکه نرم‌افزار محور با استفاده از شبیه‌ساز Mininet اجرا شد. این شبکه، کنترل‌کننده Ryu را به‌عنوان سیستم عامل شبکه و سوئیچهای Open Flow را توسط Mininet تطبیق داده‌است، یک برنامه کاربردی شبکه برای استفاده از افزونگی ایجاد شده‌است که با توجه به بارهای مختلف ترافیک در طول زمان، تنظیمات بهتری برای تقسیم بار ایجاد می‌کند. برنامه کاربردی شبکه یک اسکریپت پایتون است که از قابلی های کنترل‌کننده Ryu برای جمع‌آوری آمار ترافیک از سوئیچ‌های Open Flow استفاده می‌کند. نتایج نشان داد که یک تقسیم بهینه یا تقریبی بار می‌تواند به سوئیچ‌های ریشه داده شود.