دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده مهندسی برق

پایان‌نامه کارشناسی‌ارشد

گرایش سیستم‌های الکترونیک دیجیتال

تخصیص منابع در شبکه‌های تعریف‌شده با نرم‌افزار (SDN) برای کاربردهای نسل پنجم موبایل

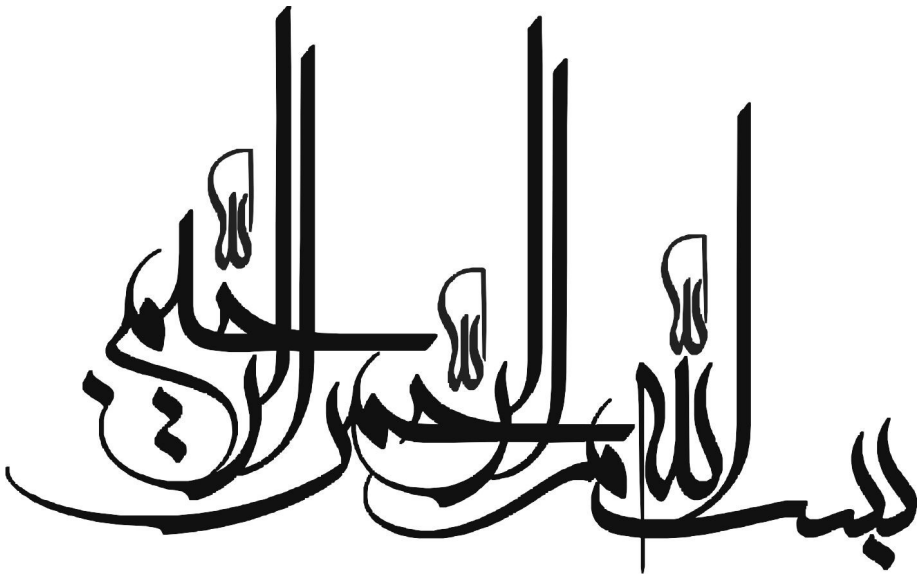
نگارش

علیرضا فرشین

استاد راهنما

دکتر سعید شریفیان

تیرماه ۱۳۹۶


بسم الله الرحمن الرحيم

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تایید و تصویب پایان نامه موسوم به فرم کمیته دفاع- موجود در پرونده آموزشی- را قرار دهید.

نکات مهم:

- نگارش پایان نامه/رساله باید به زبان فارسی و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت پشت و رو(دورو) بلامانع است و انجام آن توصیه می شود.

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

به نام خدا

تعهدنامه اصالت اثر

تاریخ: تیرماه ۱۳۹۶

اینجانب **علیرضا فرشین** متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

علیرضا فرشین

امضا

سپاس‌گزاری

با کمال ادب و احترام از استاد فاضل و اندیشمند خود جناب آقای دکتر سعید شریفیان به سبب رهنمودها و همکاری‌های صمیمانه‌شان قدردانی می‌نمایم که علی‌رغم مشغله فراوان، همواره بنده را مورد لطف و محبت خود قرارداده‌اند.

علیرضا فرشین
تیرماه ۱۳۹۶

چکیده

در سال‌های اخیر بدلیل پیشرفت‌های صورت گرفته در صنعت ارتباطات، نرخ ترافیک شبکه‌های سیار به‌شدت افزایش یافته است و این افزایش با توجه به پیدایش اینترنت اشیاء و تجارت‌های مبتنی بر رایانش ابری روزبه‌روز با شیب بیشتری ادامه می‌یابد. از آنجایی که سیستم‌های کنونی ارتباطات بی‌سیم قادر به برآورده کردن نیازهای این حجم از داده نیستند، نسل بعدی استاندارد ارتباطات موبایل قصد دارد با استفاده از تکنولوژی‌های جدیدی نظیر رایانش ابری، مجازی‌سازی توابع شبکه و شبکه‌های تعریف‌شده با نرم‌افزار این مشکل را برطرف نماید. رایانش ابری منابع لازم برای برآورده کردن نیازهای ترافیک داده‌ها را فراهم نموده، شبکه‌های تعریف‌شده با نرم‌افزار (SDN) با جداسازی صفحه‌های داده و کنترل در شبکه به مدیریت آسان‌تر این شبکه‌ها کمک کرده و مجازی‌سازی توابع شبکه (NFV) با پیاده‌سازی مجازی توابع شبکه منجر به سهولت پیاده‌سازی و افزایش سرعت تحقیق و توسعه می‌شود.
با توجه به اندازه‌ی بزرگ شبکه‌های موبایل و استفاده از رایانش ابری در طراحی استاندارد نسل پنجم موبایل، تخصیص منابع در این شبکه‌ها باید به صورت بهینه صورت پذیرد تا علاوه بر برآورده شدن نیازمندی‌های کاربران، در مصرف منابع موجود نیز صرفه‌جویی شود. لذا در این پژوهش، قصد داریم ابتدا با استفاده از تکنولوژی‌های مطرح‌شده در بالا، معماری جدیدی برای نسل پنجم موبایل پیشنهاد داده و سپس مسئله‌ی تخصیص منابع در این شبکه‌ها را از دو منظر مختلف بررسی نماییم.
در بخش اول، ابتدا به معرفی مسئله‌ی جانمایی و فراهم‌سازی کنترلرها در شبکه‌های تعریف‌شده با نرم‌افزار پرداخته و سپس چارچوب نرم‌افزاری جدیدی مبتنی بر الگوریتم‌های بهینه‌سازی فرا ابتکاری برای پیدا نمودن تعداد بهینه کنترلرها و ارتباط بهینه‌ی بین کنترلرها و سوئیچ‌ها در نسل جدید شبکه‌های موبایل ارائه می‌نماییم. الگوریتم ارائه‌شده در این بخش در مقایسه با الگوریتم‌های دیگر فرا ابتکاری مثل PSO دقت و همگرایی را به ترتیب ۱ و ۶۸ درصد بهبود می‌بخشد.
در بخش دوم، به بررسی مسیریابی جریان‌های شبکه در شبکه‌های تعریف‌شده با نرم‌افزار که یکی از اساسی‌ترین مسائل موجود در شبکه‌های مختلف است می‌پردازیم و سپس چارچوب نرم‌افزاری جدیدی مبتنی بر الگوریتم‌های بهینه‌سازی فرا ابتکاری برای مسیریابی جریان‌های شبکه به‌صورت بهینه با استفاده از اطلاعات استخراج‌شده از کنترلرهای شبکه‌های تعریف‌شده با نرم‌افزار ارائه می‌نماییم. چارچوب نرم‌افزاری پیشنهادی منجر به انعطاف‌پذیری بیش‌تر در روند مسیریابی شده و با توزیع توابع شبکه بین ابرک‌ها، توازن‌بار ایجاد می‌کند. الگوریتم معرفی‌شده در این بخش نیز در مقایسه با الگوریتم‌های سنتی فرا ابتکاری منجر به بهبود ۳۳ درصدی همگرایی می‌شود.

واژه‌های کلیدی:

شبکه‌های تعریف‌شده با نرم‌افزار، تخصیص منابع، جانمایی و فراهم‌سازی کنترلرها، مسیریابی جریان، الگوریتم‌های بهینه‌سازی فرا ابتکاری، کیفیت سرویس، نسل پنجم موبایل.

فهرست مطالب

عنوان صفحه

فهرست شکل‌ها

شکل صفحه

فهرست جدول‌ها

جدول صفحه

فهرست نمادها

نماد	مفهوم
N	تعداد کلاس‌های کیفیت سرویس در شبکه
Φ_i	کلاس i ام در شبکه
M_i	تعداد سوئیچ‌ها در کلاس i ام
C_j^i	کنترلر j ام در کلاس i ام
λ_k	نرخ ارسال درخواست توسط سوئیچ k ام
λ	مجموع λ_k ها در یک کلاس
μ	نرخ سرویس‌دهی کنترلر
c	تعداد هسته‌های کنترلر
P_k	احتمال حضور k بسته در کنترلر
t_s	زمان موردنیاز برای پردازش بسته توسط کنترلر
t_{RI}	زمان موردنیاز برای ارسال دستورات به سوئیچ‌ها و نصب قوانین
n^i	تعداد کنترلرهای کلاس i ام
T^i	مقدار سودمندی زمانی برای کلاس i ام
C^i	مقدار سودمندی هزینه‌ی کنترلرها در کلاس i ام
F^i	مقدار سودمندی شاخص برابری در کلاس i ام
L_i	بار ابرک i ام

$\tau_{i,j}$	پارامتر فرومون
$\eta_{i,j}$	پارامتر مکاشفه‌ای
$\theta_{i,j}$	پارامتر انبار
Θ	ماتریس پارامتر انبار

فهرست اختصارات

ACS	Ant Colony System
API	Application Programming Interface
ATM	Asynchronous Transfer Mode
BTS	Base Transceiver Station
CAPEX	Capital Expenditures
CR	Core Network
C-RAN	Cloud Radio Access Network
DSP	Digital Signal Processing
GWO	Grey Wolf Optimizer
H-CRAN	Heterogeneous Cloud Radio Access Network
ILP	Integer Linear Programming
IoT	Internet of Things
IP	Internet Protocol
JSON	Java Script Object Notation
LTE	Long Term Evolution
MKP	Multi Knapsack Problem
MPLS	Multi-Protocol Label Switching
MVNP	Mobile Virtual Network Provider
MVNO	Mobile Virtual Network Operator
NAT	Network Address Translation
NFV	Network Function Virtualization
OPEX	Operational Expenditures
OSPF	Open Shortest Path First
PM	Physical Machine
PPP	Public Private Partnership
PSO	Particle Swarm Optimization
QoS	Quality of Service

RAN	Radio Access Network
SDMN	Software-Defined Mobile Networks
SDN	Software-Defined Networking
SFOP	Shortest-Feasible OpenFlow Path
SLA	Service Level Agreement
VANET	Vehicular Ad-Hoc Network
VM	Virtual Machine
VNI	Visual Networking Index
VoD	Video on Demand
WAN	Wide Area Network

فصل اول

مقدمه

نرخ ترافیک شبکه‌های موبایل در سال‌های اخیر به‌شدت افزایش پیداکرده است و طبق شاخص تصویری شبکه[۱] مربوط به شرکت سیسکو، مقدار ماهانه‌ی ترافیک شبکه‌های موبایل تا سال ۲۰۱۸ از سقف ۱۵.۹ اگزاپایت نیز تجاوز خواهد کرد [1]. به‌علاوه، با پیدایش اینترنت اشیاء[۲]، تمامی دستگاه‌های موبایل به اینترنت متصل شده و داده تولید می‌کنند که این امر خود منجر به افزایش نمایی ترافیک داده در شبکه‌ها خواهد شد [2]. با توجه به محدودیت‌های موجود در نسل فعلی شبکه‌های موبایل، امکان برآورده نمودن نیازمندی‌های کاربران و تأمین کیفیت سرویس[۳] ارتباطات با این حجم بسیار زیاد از داده وجود ندارد و نیاز به معرفی نسل جدیدی از شبکه‌های موبایل احساس می‌شود. به همین منظور، PPP در اروپا [3, 4] برای نسل پنجم ارتباطات موبایل پنج هدف اصلی زیر را تعریف کرده است:

۱. ۱۰ تا ۱۰۰ برابر افزایش نرخ داده برای کاربران معمولی

۲. ۱۰ تا ۱۰۰ برابر افزایش در تعداد کاربران

۳. ۱۰ برابر کاهش در مصرف انرژی شبکه

۴. کمتر از ۱ میلی‌ثانیه تأخیر سرتاسری

۵. ۱۰۰۰ برابر افزایش نرخ داده برای هر منطقه‌ی جغرافیایی

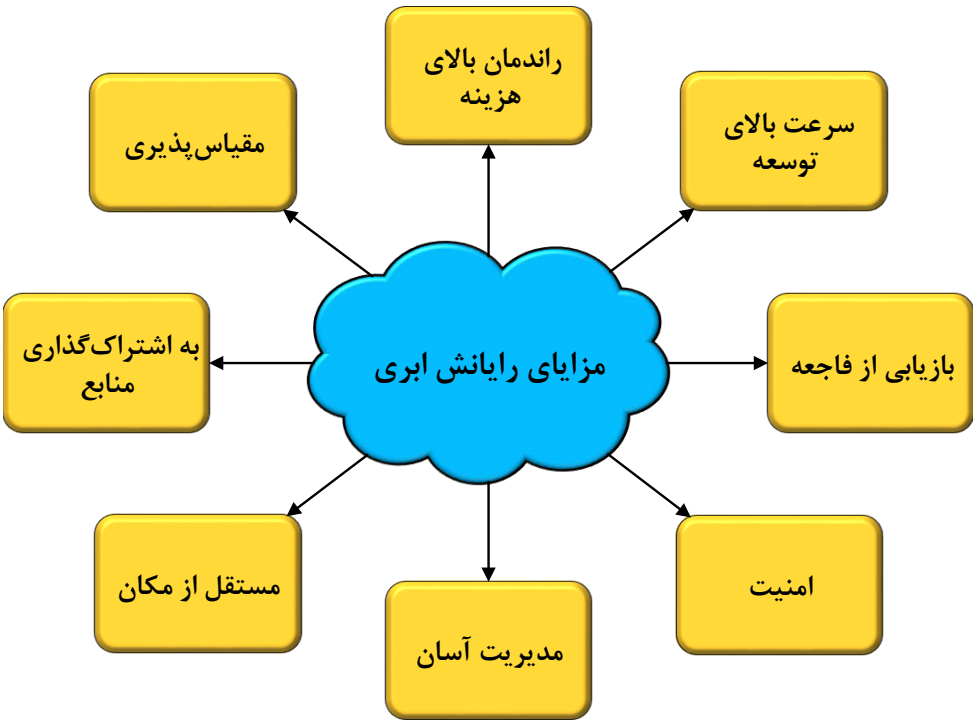
علاوه بر این اهداف، با پیدایش مجازی‌سازی شبکه‌های موبایل، تأمین‌کنندگان مجازی شبکه‌ی موبایل[۴] باید بتوانند طیف رادیویی و زیرساخت ارتباطی موجود را با اپراتورهای مجازی[۵] به اشتراک بگذارند [5]. لذا، امکان به اشتراک‌گذاری طیف رادیویی و زیرساخت ارتباطی نیز باید جز اهداف نسل پنجم استاندارد شبکه‌های موبایل مدنظر قرارگرفته شود.

برای تأمین نیازمندی‌های ذکرشده در بالا، نسل پنجم شبکه‌های موبایل می‌بایست از زیرساخت‌های ابری و فناوری‌های بروز نظیر رایانش ابری[۶]، مجازی‌سازی شبکه[۷]، شبکه‌های تعریف‌شده با نرم‌افزار[۸] و مجازی‌سازی توابع شبکه[۹] [6, 7, 8] استفاده نماید. بدین منظور، معماری جدیدی مبتنی بر شبکه‌های تعریف‌شده با نرم‌افزار برای شبکه‌های موبایل پیشنهادشده است که در بخش‌های بعدی از همین فصل به توضیح آن می‌پردازیم.

[۱] Visual Networking Index (VNI)
[۲] Internet of Things (IoT)
[۳] Quality of Service (QoS)
[۴] Mobile Virtual Network Provider (MVNP)
[۵] Mobile Virtual Network Operator (MVNO)
[۶] Cloud Computing (CC)
[۷] Network Virtualization
[۸] Software-Defined Networking (SDN)
[۹] Network Function Virtualization (NFV)

۱-۱ فناوری‌های مورداستفاده در نسل پنجم موبایل

برای اینکه نسل پنجم موبایل بتواند نیازمندی‌های مربوط به نرخ بسیار زیاد داده در شبکه‌ها را برآورده سازد می‌بایست از فناوری‌های جدید بهره جوید. یکی از این فناوری‌ها، رایانش ابری است که امکان داشتن یک مرکز برای انجام محاسبات و ذخیره‌سازی اطلاعات را ممکن می‌سازد. این مراکز، که معمولاً بانام مراکز داده شناخته می‌شوند، می‌توانند از راه دور به کاربران در نقاط مختلف سرویس بدهند. با داشتن یک یا چندین مرکز داده با ظرفیت محاسباتی بالا می‌توان نیازمندی‌های مربوط به حجم بسیار زیاد داده‌ها در شبکه‌های موبایل را برآورده نمود. علاوه بر ایجاد تمرکز، رایانش ابری مزایای دیگری را به دنبال دارد که برخی از آن‌ها در شکل (۱-۱) نشان داده‌شده‌اند.



شکل ۱-۱: مزایای رایانش ابری برای نسل پنجم موبایل

شبکه‌های موبایل شامل حجم بسیار زیادی از کارهای مربوط به پردازش سیگنال‌های دیجیتال (DSP) نظیر فیلترینگ، رمزنگاری و رمزگشایی هستند که در حال حاضر توسط BTS ها[1] انجام می‌شوند. به همین خاطر، پیاده‌سازی فناوری‌های جدید مخابراتی مستلزم نصب کردن واحدهای مربوطه در تمامی BTS است که پروسه‌ی نصب و راه‌اندازی این فناوری‌ها را بسیار کند و طولانی می‌کند. در صورت استفاده از رایانش ابری در نسل جدید شبکه‌ها موبایل، به‌جای استفاده از سخت‌افزار اضافی در BTS ها برای انجام عملیات پردازش سیگنال‌های دیجیتال، می‌توان این عملیات را به‌صورت برنامه‌ی نرم‌افزاری در مراکز داده

[1] Base Station Transceiver

پیاده‌سازی نمود و BTS ها تنها داده‌های خام را به این مراکز داده ارسال کنند [9]. این مراکز داده معمولاً به‌صورت توزیع‌شده و نزدیک شبکه‌ی دسترسی به رادیو[۲] ساخته می‌شوند تا تأخیر ناشی از انتقال اعمال پردازش سیگنال‌های دیجیتال به مراکز داده قابل‌توجه نشود [10, 11]. همچنین پیشرفت‌های صورت گرفته در صنعت رایانش ابری و مخابرات نوری نیز به کم‌تر شدن این تأخیر کمک می‌کنند. این نمونه‌ی استفاده از رایانش ابری برای نسل جدید شبکه‌های موبایل که بانام شبکه دسترسی رادیو ابری[۳] شناخته می‌شود برای اولین بار توسط China Mobile Research [12] معرفی شد و بعدها نسخه‌ی کامل‌تر آن نیز بانام H-RAN [2] معرفی شد. استفاده از این معماری‌های پیشنهادی برای زیرساخت نسل پنجم شبکه‌های موبایل منجر به افزایش انعطاف‌پذیری این شبکه‌ها می‌شود و به کمک آن‌ها می‌توان فناوری‌های جدید مخابراتی را با سرعت بیشتری راهی بازار کرد. همچنین استفاده از این معماری‌ها به حل مسائل تخصیص منابع نیز کمک بسیار زیادی خواهد نمود.

اگرچه استفاده از رایانش ابری خود به‌تنهایی می‌تواند خیلی از نیازمندی‌های نسل پنجم موبایل را حل نماید ولی با ترکیب آن با فناوری‌های جدید دیگری نظیر شبکه‌های تعریف‌شده با نرم‌افزار و مجازی‌سازی توابع شبکه می‌توان به حل این مشکلات سرعت بیشتری بخشید. ازآنجایی‌که هریک از جریان‌هایی[۱] که وارد شبکه می‌شوند به تعدادی سرویس و توابع شبکه[۲] نیازمندند، برای انجام هریک از آن‌ها می‌بایست یک سخت‌افزار اضافی میانی[۳] اختصاص داد و افزایش بیش‌ازحد حجم ترافیک داده در شبکه‌ها مشکلات زیادی در اجرای این سرویس‌ها ایجاد می‌کند. بعضی از این سرویس‌ها [13]، مربوط به افزایش عملکرد شبکه (مثل توازن بار[۴])، مربوط به افزایش امنیت (همانند: دیوار آتش[۵] و سیستم تشخیص ورود بی‌اجازه[۶]) و مربوط به سیاست‌گذاری در شبکه (مثل ترجمه‌ی آدرس شبکه[۷]) هستند. همان‌طور که در مرجع [14] بیان شده است، در شبکه‌های امروزی تعداد تجهیزات سخت‌افزاری میانی تقریباً با تعداد روترها برابری می‌کند. به همین خاطر نیاز به توسعه‌ی فناوری جدیدی برای بهتر نمودن انعطاف‌پذیری، مقیاس‌پذیری و مدیریت این تجهیزات سخت‌افزاری احساس می‌شود [15]. NFV توابع شبکه را از زیرساخت شبکه جدا می‌کند و پیشنهاد می‌دهد تا به‌جای استفاده از تجهیزات سخت‌افزاری میانی برای پیاده‌سازی توابع شبکه، آن‌ها را به‌صورت مجازی بر روی سرورهای موجود در مراکز داده پیاده‌سازی کنیم. علاوه بر این، با سامان‌دهی[۸] و ترکیب کردن توابع شبکه و سرویس‌هایی که بر روی ماشین‌های مجازی[۹] پیاده‌سازی شده‌اند، می‌توانیم فنّاوری‌های مختلف را به‌صورت زنجیره‌ای از سرویس‌های[۱۰] [16] مختلف پیاده‌سازی

[۲] Radio Access Network (RAN)
[۳] Cloud RAN(C-RAN)
[۱] Flow
[۲] Network Applications
[۳] Middlebox
[۴] Load Balancing
[۵] Firewall
[۶] Intrusion Detection System (IDS)
[۷] Network Address Translators (NAT)
[۸] Orchestrate
[۹] Virtual Machines (VM)
[۱۰] Chain of Services

نماییم. لذا، با استفاده از NFV می‌توان برای پیاده‌سازی توابع شبکه در زیرساخت مخابراتی و طراحی مجدد و بهینه‌ی هسته‌ی مرکزی[۱۱] شبکه‌ی موبایل نسل پنجم موبایل استفاده نمود.

با توجه به تحرک بالای موجود در شبکه‌های موبایل، توپولوژی و نرخ ترافیک داده در این شبکه‌ها به‌طور مدارم در حال تغییر است و مدیریت این شبکه‌ها را دشوار می‌کند؛ بنابراین، برای دستیابی به اهداف تعریف‌شده برای نسل جدید موبایل، نیاز به سیستمی برای مدیریت پویای شبکه‌ها کاملاً احساس می‌شود. خوشبختانه، شبکه‌های تعریف‌شده با نرم‌افزار (SDN) که در سال‌های اخیر معرفی‌شده است می‌تواند مدیریت این شبکه‌ها را تا حد بسیار زیادی آسان نماید.

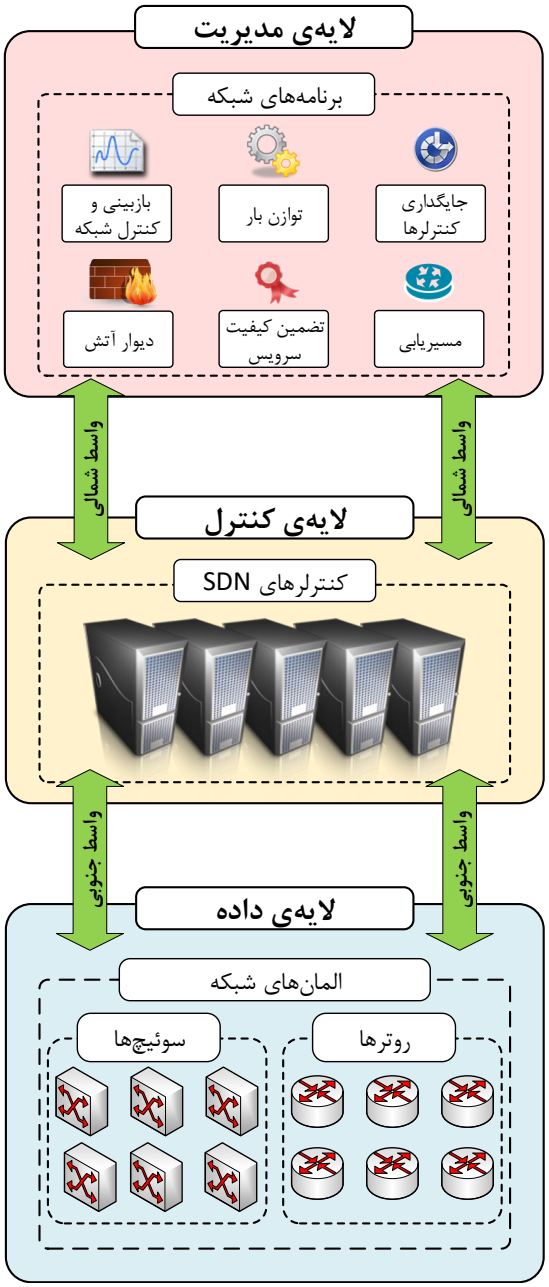
شبکه‌های تعریف‌شده با نرم‌افزار روش جدیدی برای مدیریت شبکه معرفی می‌کنند که در آن برخلاف شبکه‌های سنتی، بخش تصمیم‌گیری شبکه از المان‌های اصلی شبکه نظیر سوئیچ‌ها و روترها به کنترلر های نرم‌افزاری منتقل می‌شود. با جدا شدن صفحه‌های داده[۱] و کنترل[۲] در این شبکه‌ها، المان‌های شبکه تنها وظیفه‌ی انتقال بسته‌ها در شبکه را عهده‌دار شده و تصمیم‌گیری در خصوص نحوه‌ی مسیریابی و انتقال این بسته‌ها توسط کنترلرهای نرم‌افزاری صورت می‌گیرد. این امر منجر به پیاده‌سازی سریع سیاست‌ها و الگوریتم‌های جدید در شبکه می‌شود و در پی آن مدیریت شبکه را آسان و متمرکز[۳] می‌کند [17, 18].

همان‌طور که در شکل (۱-۲) نشان داده‌شده است، ارتباط بین صفحه‌های داده و کنترلر توسط واسط‌های برنامه‌نویسی استاندارد[۴] بانام واسط‌های نرم‌افزاری جنوبی[۵] صورت می‌گیرد که به‌عنوان‌مثال می‌توان از استاندارد OpenFlow [19] و استاندارد OnePK شرکت سیسکو [20] نام برد. استفاده از شبکه‌های تعریف‌شده با نرم‌افزار منجر به پیاده‌سازی سریع‌تر سیاست‌ها و الگوریتم‌ها جدید در شبکه می‌شود. به‌منظور پیاده‌سازی راحت‌تر این الگوریتم‌ها و عدم‌تغییر نرم‌افزار موجود در کنترلرها، صفحه‌ای جدید بانام صفحه‌ی مدیریت[۶] معرفی می‌شود که این صفحه‌ی با استفاده از واسط‌های نرم‌افزاری شمالی[۷] می‌توانند با صفحه‌ی کنترل در ارتباط باشند و تغییرات لازم را در شبکه ایجاد نمایند.

استاندارد RESTful API [21] یکی از شناخته‌شده‌ترین و پرکاربردترین این واسط‌های نرم‌افزاری است. برای پیاده‌سازی برنامه‌های شبکه مانند برنامه‌های مسیریابی و توازن بار با استفاده از RESTful API، برنامه‌های شبکه با هر زبان برنامه‌نویسی دلخواه پیاده‌سازی می‌شوند و سپس با دستوراتی شبیه دستورات HTTP (مانند: PUT و GET) و فرمت JSON اطلاعات بین صفحه‌های مدیریت و کنترل ردوبدل می‌شوند.

---

[۱۱] Core Network (CR)
[۱] Data Plane
[۲] Control Plane
[۳] Centralized
[۴] Application Programming Interface (API)
[۵] Southbound API (SB)
[۶] Management Plane
[۷] Northbound API (NB)



شکل ۱-۲: معماری شبکه‌های تعریف‌شده با نرم‌افزار

۱-۲ شبکه‌های موبایل تعریف‌شده با نرم‌افزار (SDMN)

با توجه به ویژگی‌های بسیار خوب فناوری‌های ذکرشده در بخش قبلی می‌توانیم از آن‌ها برای طراحی معماری جدیدی برای نسل پنجم موبایل استفاده نماییم؛ بنابراین، در ادامه به معرفی یک معماری پیشنهادی با عنوان شبکه‌های موبایل تعریف‌شده با نرم‌افزار[۱] می‌پردازیم. همان‌طور که در شکل (۱-۳)

[۱] Software-Defined Mobile Network (SDMN)

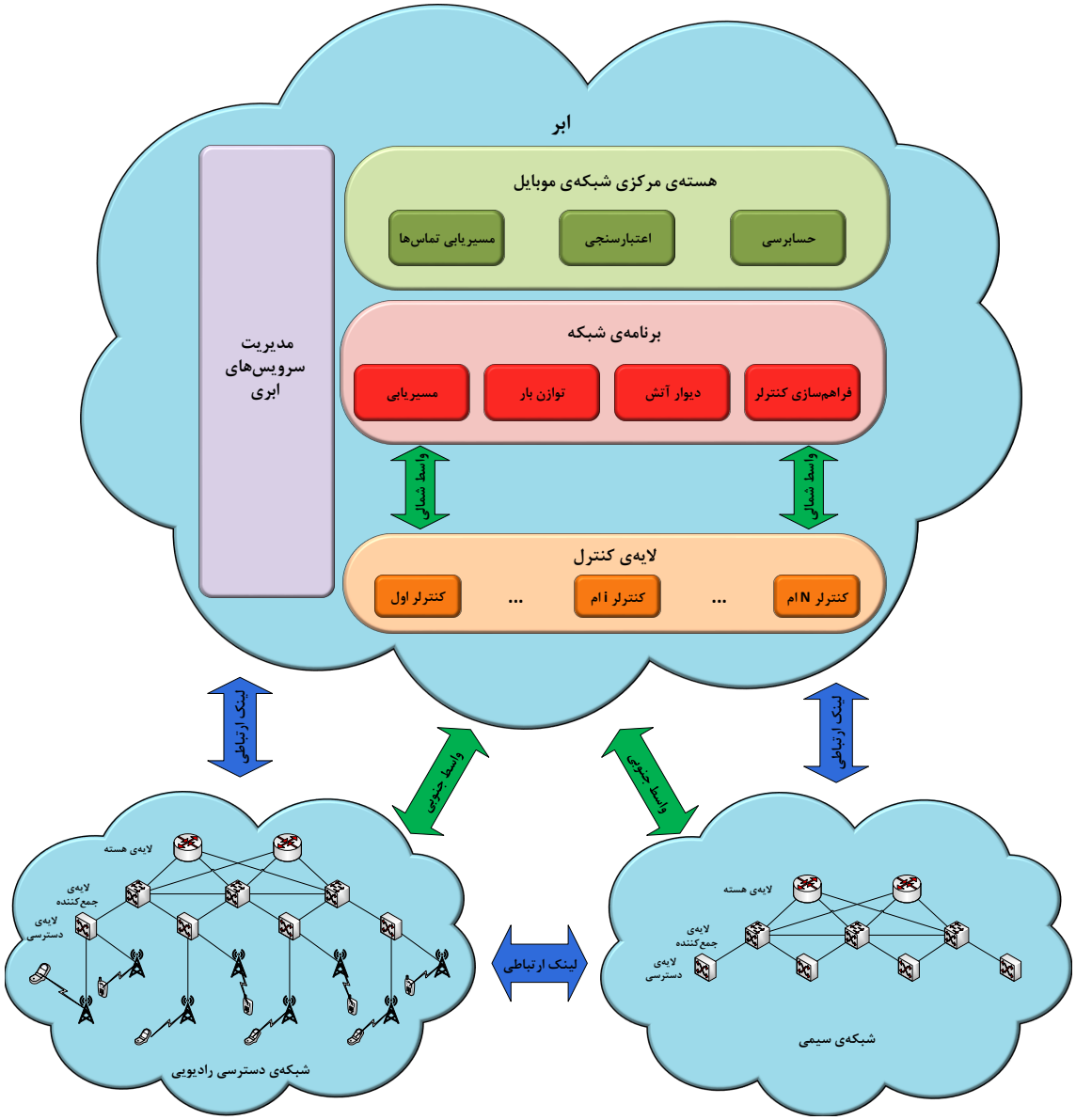
نشان داده‌شده است. معماری SDMN به سه بخش اصلی به شرح زیر تقسیم می‌شود:

۱. **شبکه دسترسی رادیویی**: یکی از اصلی‌ترین بخش‌های یک شبکه‌ی موبایل شبکه‌ی دسترسی رادیویی آن است که شامل تمامی آنتن‌های رادیویی و سوئیچ‌های شبکه بوده و دستگاه‌های موبایل را به هسته‌ی مرکزی شبکه وصل می‌کند. مشابه هر شبکه‌ی بزرگ دیگری، ساختار سوئیچینگ شبکه‌ی دسترسی رادیویی نیز از یک توپولوژی درختی دو یا سه لایه تشکیل‌شده است [22, 23]. لایه‌ی دسترسی[۱] مستقیماً به کاربران و مصرف‌کنندگان نهایی از طریق آنتن‌های رادیویی متصل است و در اغلب مواقع ارتباط پشتیبانی بین دستگاه‌ها و سوئیچ‌های موجود در این لایه وجود ندارد. با افزایش حجم شبکه و تعداد کاربران آن، تعداد سوئیچ‌های لایه‌ی دسترسی نیز افزایش پیدا می‌کند و نیاز به ایجاد لایه‌ی جدیدی بانام لایه‌ی جمع کننده[۲] احساس می‌شود تا ترافیک داده از لایه‌ی دسترسی را جمع‌آوری نموده و آن را با لینک‌هایی با سرعت‌بالا به لایه‌ی آخر یعنی لایه‌ی هسته[۳] برساند. ایجاد لایه‌ی میانی جمع کننده منجر به مدیریت آسان‌تر سوئیچ‌ها و صرفه‌جویی در مصرف کابل نیز می‌شود. لایه‌ی هسته بالاترین لایه در ساختار سوئیچینگ شبکه‌ها بوده و تمام ترافیک موجود در شبکه از سوئیچ‌های این لایه گذر می‌کند.

۲. **هسته‌ی مرکزی**: بخش مرکزی هر شبکه‌ی موبایل هسته‌ی مرکزی آن است که سرویس موجود در یک شبکه‌ی موبایل نظیر اعتبارسنجی و حسابرسی را به کاربران و مشتریان شبکه ارائه می‌کند. این بخش همچنین واسط بین شبکه‌های بی‌سیم و شبکه‌های سیمی نیز هست.

۳. **ابر**: یکی از اساسی‌ترین بخش‌های معماری‌های نسل پنجم موبایل، ابر یا مراکز داده هستند که منظور از آن‌ها، مراکز داده‌ی توزیع‌شده نزدیک شبکه‌ی دسترسی رادیویی بوده که مسئولیت انجام عملیات پردازش سیگنال‌های دیجیتال را بر عهده‌دارند. به کمک فناوری NFV می‌توان علاوه بر عملیات DSP، واحدهای اصلی شبکه‌ی موبایل را نیز به‌صورت مجازی در این مراکز داده پیاده‌سازی نمود. همچنین به کمک فناوری SDN مدیریت کلی شبکه‌ی موبایل و مراکز داده می‌تواند توسط صفحه‌ی کنترل در این شبکه‌های صورت پذیرد و توابع شبکه با سهولت بیشتری پیاده‌سازی شوند. همان‌طور که در شکل (۱-۳) نشان داده‌شده است، علاوه بر واحدهای ذکرشده، این شبکه‌ها مشابه هر شبکه‌ی ابر محور دیگری نیاز به واحدی برای مدیریت سرویس‌های ابری خواهند داشت تا مسئولیت کنترل و مدیریت ابر را به عهده بگیرد.

[۱] Edge/Access Layer
[۲] Aggregation Layer
[۳] Core Layer



شکل ۱-۳: معماری شبکه‌های موبایل تعریف‌شده با نرم‌افزار

۱-۳ هدف پژوهش

با توجه به‌اندازه‌ی بزرگ شبکه‌های موبایل و استفاده از رایانش ابری در نسل پنجم موبایل، تخصیص منابع در این شبکه‌ها باید به‌صورت بهینه صورت پذیرد تا علاوه بر رفع نیازمندی‌های کاربران، در مصرف منابع و هزینه‌های شبکه نیز صرفه‌جویی گردد. به همین منظور، در این پژوهش قصد داریم تا با پیشنهاد راه‌حل برای دو مسئله‌ی زیر در شبکه‌های موبایل تعریف‌شده با نرم‌افزار به تخصیص منابع در نسل پنجم شبکه‌های موبایل کمک نماییم.

۱-۳-۱ جانمایی و فراهم‌سازی کنترلرها

در شبکه‌های تعریف‌شده با نرم‌افزار برخلاف شبکه‌های سنتی، یک سوئیچ هیچ اطلاعاتی در خصوص سیاست‌ها و توپولوژی شبکه ندارد، بدین خاطر وقتی یک سوئیچ برای اولین بار یک بسته‌ی جدید را دریافت می‌کند، آن را برای کنترلر شبکه ارسال می‌کند. کنترلر پس از بررسی بسته‌ی ارسال‌شده و تصمیم‌گیری در خصوص آن بر اساس قوانین و سیاست‌های تعریف‌شده در شبکه، دستور مناسب را برای سوئیچ ارسال نموده و قوانین مناسب برای مسیریابی این بسته را در جدول مسیریابی سوئیچ یادداشت می‌کند. از آن به بعد و تا زمان معتبر بودن این دستورات، سوئیچ مربوطه می‌تواند بدون نیاز به کنترلر بسته‌های مشابه را در شبکه مسیریابی کند. مدت‌زمان موردنیاز برای تکمیل این فرآیند شامل دو بخش به شرح زیر است:

۱. **زمان موردنیاز برای پردازش بسته توسط کنترلر** (t_S): این زمان متناسب با توان پردازشی کنترلر و نرخ ارسال بسته توسط سوئیچ‌ها به کنترلرها است و می‌تواند در طول فعالیت کنترلر با تغییر پارامترهای مذکور تغییر کند.

۲. **زمان موردنیاز برای ارسال دستورات به سوئیچ‌ها و نصب قانون‌های جدید در آن‌ها** (t_{RI}): این زمان متناسب با تأخیر اتصالات، نوع واسط جنوبی، نوع سوئیچ‌ها و کنترلرها است و در یک شبکه معمولاً ثابت است.

با توجه به ظرفیت محاسباتی محدود کنترلرهای SDN، اتصال تعداد بیش‌ازحد سوئیچ به یک کنترلر منجر به افزایش زمان موردنیاز برای پردازش بسته (t_S) می‌شود. به همین دلیل، در شبکه‌های بزرگ نظیر شبکه‌های موبایل و شبکه‌های مراکز داده استفاده از یک کنترلر SDN در شبکه کافی نبوده و برای برطرف نمودن نیازمندی‌های مربوط به کیفیت سرویس می‌بایست صفحه‌ی کنترل را بین چندین کنترلر که در نقاط مختلف شبکه قرارگرفته‌اند، تقسیم نماییم. در پروسه‌ی توزیع نمودن صفحه‌ی کنترل، ازآنجایی‌که تعداد کنترلرها مستقیماً بر روی هزینه‌های اصلی[۱] و هزینه‌های نگهداری[۲] شبکه تأثیر می‌گذارند، مشخص نمودن تعداد دقیق کنترلرها از اهمیت خاصی برخوردار می‌شود و به همین منظور می‌بایست تعداد کنترلرها را در طول زمان متناسب با اندازه‌ی شبکه و حجم ترافیک داده به‌صورت پویا[۳] تعیین نماییم تا علاوه بر رفع نیازمندی‌های شبکه، منابع نیز به‌صورت بهینه در شبکه تخصیص داده شوند. به همین منظور، ما در قسمت اول این پژوهش، یک چارچوب نرم‌افزاری[۴] مبتنی بر الگوریتم‌های بهینه‌سازی فرا ابتکاری[۵] برای حل مشکل «فراهم‌سازی کنترلرها»[۶] در شبکه پیشنهاد می‌دهیم.

[۱] Capital Expenditure (CAPEX)
[۲] Operational Expenditure)OPEX)
[۳] Dynamic
[۴] Framework
[۵] Metaheuristic Algorithms
[۶] Controller Provisioning

همان‌طور که در مراجع [24, 25] بیان شده است، فراهم‌سازی کنترلرها در دسته‌ی مسائل NP-Hard قرار گرفته می‌شود و همچنین بزرگ شدن روزافزون اندازه و ترافیک شبکه‌ها نیز خود به سخت‌تر شدن این مسئله کمک می‌کند. به همین منظور ما در چارچوب نرم‌افزاری خود یک الگوریتم فرا ابتکاری چندجوابه بانام «گرگ‌های خاکستری[۷]» پیشنهاد نموده‌ایم که نسبت به الگوریتم‌های فرا ابتکاری دیگر عملکرد بهتری داشته و در حل مسائل NP-Hard رفتار بهتری از خود نشان می‌دهد. ما برای بهبود رفتار الگوریتم پیشنهادی ازسری‌های آشوبناک[۸] نیز بهره جسته‌ایم که در فصل‌های بعدی به‌تفصیل در خصوص آن‌ها توضیح خواهیم داد.

۱-۳-۲ مسیریابی جریان‌ها و جانمایی سرویس‌های شبکه

پیدایش فناوری شبکه‌های تعریف‌شده با نرم‌افزار منجر به تغییرات اساسی در شبکه‌ها می‌شود و به کمک آن می‌توان برای تمامی مسائل موجود در شبکه راه‌حل‌های جدید و مناسب‌تری ارائه نمود. یکی از این مسائل، مسیریابی جریان‌های شبکه بوده که از گذشته تلاش‌های بسیاری برای ارائه‌ی راه‌حلی مناسب برای آن انجام‌گرفته است. ما نیز در این پژوهش قصد داریم تا با پیشنهاد یک چارچوب نرم‌افزاری جدید برای انجام مسیریابی جریان‌ها در شبکه‌های موبایل تعریف‌شده با نرم‌افزار به حل این مسئله و تخصیص بهینه‌ی منابع کمک نماییم.

به دلیل بزرگ شدن اندازه‌ی شبکه‌ها و رشد روزافزون ترافیک داده در شبکه‌ها مسائل مختلف شبکه اغلب به مسائل بهینه‌سازی NP-Hard تبدیل‌شده‌اند. لذا، ما در چارچوب نرم‌افزاری خود یک الگوریتم فرا ابتکاری دانش‌محور[۱] پیشنهاد نموده‌ایم. چارچوب نرم‌افزاری ما با بررسی دقیق اطلاعات به‌دست‌آمده از شبکه توسط کنترلرهای SDN و نیازمندی‌های هر جریان جدید در شبکه، یک مسیر بهینه و چندین مسیر جایگزین برای آن جریان پیدا می‌کند. به‌علاوه ازآنجایی‌که هریک از جریان‌های ورودی در شبکه به تعداد زیادی توابع و سرویس‌های مختلف نیازمندند، ما برای تخصیص بهینه‌ی منابع در شبکه و استفاده از تمامی ظرفیت‌های پردازشی موجود، به‌جای انجام این سرویس‌ها به‌صورت متمرکز، این سرویس‌ها را بین نقاط مختلف شبکه که هریک می‌توانند به یک ابرک متصل باشند تقسیم می‌کنیم که در فصل‌های بعدی به‌تفصیل در خصوص آن‌ها توضیح خواهیم داد.

۱-۴ نوآوری پژوهش

در این پایان‌نامه به منظور تخصیص بهینه‌ی منابع در شبکه‌های موبایل تعریف‌شده با نرم‌افزار دو چارچوب نرم‌افزاری برای حل مسائل «فراهم‌سازی کنترلرها» و «مسیریابی جریان‌ها و جانمایی سرویس‌ها» پیشنهاد شده که نوآوری‌های ارائه‌شده در آن‌ها در جدول (۱-۱) نشان داده شده است.

[۷] Grey Wolf Optimizer
[۸] Chaotic Sequences
[۱] Knowledge-based

جدول ۱-۱: نوآوری‌های پژوهش

نوآوری‌ها	چارچوب نرم‌افزاری پیشنهادی
مدل‌سازی مسئله با استفاده از مدل‌های صف مختلف و توابع سودمندی مناسب	**فراهم‌سازی کنترلرها**
در نظر گرفتن معیار برابری در تخصیص کنترلرها	
استفاده از رویکرد فرا ابتکاری برای حل مسئله	
بهبود سرعت همگرایی الگوریتم استفاده شده با استفاده از سری‌های آشوبناک	
ارائه‌ی یک چارچوب نرم‌افزاری کامل برای حل مسئله	**مسیریابی جریان‌ها و جانمایی سرویس‌ها**
در نظر گرفتن جانمایی سرویس‌ها در فرآیند انتخاب مسیر	
استفاده از رویکرد فرا ابتکاری برای حل مسئله	
ایجاد انعطاف‌پذیری و افزایش ضریب اطمینان در فرآیند مسیریابی	
ترکیب دو الگوریتم فرا ابتکاری برای افزایش رفتار اکتشافی در حل مسئله	
بهبود سرعت همگرایی الگوریتم مسیریاب با افزودن پارامتر جدید به آن	

۱-۵ ساختار کلی پایان‌نامه

در ادامه‌ی این پایان‌نامه، در فصل دوم مروری بر کارهای انجام‌شده و الگوریتم‌های ارائه‌شده در دو زمینه‌ی جانمایی[۱] و فراهم‌سازی کنترلرها و مسیریابی جریان‌ها[۲] در شبکه‌های تعریف‌شده با نرم‌افزار پرداخته‌شده است. در فصل سوم به تشریح چارچوب نرم‌افزاری پیشنهادی برای فراهم نمودن تعداد بهینه‌ی کنترلرها در شبکه‌های موبایل تعریف‌شده با نرم‌افزار و تعریف الگوریتم‌های مورداستفاده در آن می‌پردازیم. در فصل چهارم یک چارچوب نرم‌افزاری برای مسیریابی جریان‌ها و جانمایی سرویس‌ها[۳] در شبکه‌های موبایل تعریف‌شده با نرم‌افزار پیشنهاد می‌شود که این چارچوب منجر به افزایش انعطاف‌پذیری در شبکه‌ها شده و منابع موجود را با دقت بیشتری استفاده می‌نماید. نتایج شبیه‌سازی و مقایسه با تحقیقات مشابه برای هر دو چارچوب نرم‌افزاری در فصل پنجم آورده شده است. درنهایت در فصل ششم نیز جمع‌بندی موضوع و چند پیشنهاد برای ادامه‌ی کار ارائه‌شده است.

[۱] Controller Placement
[۲] Flow Routing
[۳] Service Placement

فصل دوم

مروری بر کارهای گذشته

در این فصل قصد داریم تا با بررسی کارهای انجام‌شده در دو حوزه‌ی «جانمایی و فراهم‌سازی کنترلرها» و «مسیریابی جریان‌های شبکه و جانمایی سرویس‌ها و توابع شبکه»، نقاط قوت و ضعف آن‌ها را مشخص نموده و با ارائه‌ی چارچوب‌های نرم‌افزاری جدیدی برای این مسائل، نسبت به بهبود آن‌ها اقدام نماییم.

۲-۱ جانمایی و فراهم‌سازی کنترلرها

با رشد شبکه‌ها و بزرگ شدن اندازه‌ی آن‌ها، یک کنترلر SDN قادر به برآورده نمودن نیازمندی‌های کل ترافیک شبکه نخواهد بود و زمان موردنیاز برای پردازش بسته‌ها افزایش پیدا خواهد کرد؛ بنابراین برای رفع نیازمندی‌های کیفیت سرویس در شبکه و تفاهم‌نامه سطح کیفی خدمات[۱]، می‌بایست تعدادی کنترلر در نقاط مختلف شبکه اضافه شود ولی توزیع صفحه‌ی کنترل خود منجر مشکلات دیگری نظیر هماهنگ‌سازی ساختمان داده در کنترلرهای مختلف [26] می‌شود که باید در پروسه‌ی طراحی و پیاده‌سازی کنترلرها مدنظر قرار گیرد. به همین منظور، تحقیقات بسیاری در خصوص معماری کنترلرهای توزیع‌شده صورت گرفته است که ازجمله‌ی آن‌ها می‌توان به معماری‌های Kandoo [27]، Beehive [28]، HyperFlow [29]، ONOS [30] و ONIX [31] اشاره نمود.

پروسه‌ی زمانی کنترلرهای SDN از دو بخش تشکیل‌شده است: مدت‌زمان موردنیاز برای پردازش بسته‌ها (t_S) و مدت‌زمان موردنیاز برای ارسال دستورات به سوئیچ‌ها و نصب قانون‌ها بر روی آن‌ها (t_{RI}) که تغییر موقعیت کنترلرها و یا تغییر نرخ ترافیک شبکه می‌تواند منجر به تغییر این زمان‌ها شود. لذا، محاسبه‌ی موقعیت دقیق کنترلرها و تعداد دقیق آن‌ها از اهمیت ویژه‌ای برخوردار می‌شود. مشخص نمودن محل دقیق کنترلر بانام جانمایی کنترلرها و مشخص نمودن تعداد دقیق آن‌ها با عنوان فراهم‌سازی کنترلرها شناخته می‌شوند. از دیدگاه کیفیت سرویس، ایده‌آل‌ترین تعداد کنترلرها برای رفع نیازمندی‌های ترافیک در شبکه برابر با تعداد سوئیچ‌ها و یا روترها در شبکه هست ولی این دیدگاه ازلحاظ اقتصادی و استفاده‌ی مناسب از منابع اصلاً به‌صرفه نیست. لذا، به‌جای تخصیص تعداد زیادی کنترلر، می‌بایست تعداد آن‌ها را به‌صورت دوره‌ای[۲] و متناسب با اندازه شبکه و حجم ترافیک شبکه در طول زمان محاسبه نمود. تحقیقات بسیاری در این زمینه صورت گرفته است که در ادامه به توضیح آن‌ها می‌پردازیم:

در مرجع [32] برای اولین بار مسئله‌ی جانمایی کنترلرها مطرح شد که در آن، هلر[۳] و همکاران تأثیر تعداد و محل کنترلرها بر روی مقدار متوسط زمان تأخیر و بدترین زمان تأخیر در شبکه‌ها را به ترتیب با استفاده از روابط (۲-۱) و (۲-۲) بررسی نمودند که در این روابط v و s دو نود از گراف $G(V,E)$ هستند

[۱] Service Level Agreement (SLA)
[۲] Periodic
[۳] Heller

و S' مجموعه‌ای است که مکان کنترلرها در گراف شبکه را نشان می‌دهد.

$$L_{avg}(S') = \frac{1}{n} \sum_{v \in V} \min_{(s \in S')} d(v, s) \quad (۱-۲)$$

$$L_{wc}(S') = \max_{(v \in V)} \min_{(s \in S')} d(v, s) \quad (۲-۲)$$

در این پژوهش نشان داده شد که بین برآورده کردن نیازهای متوسط زمان تأخیر و نیازهای بدترین زمان تأخیر همیشه یک مصالحه[۱] وجود دارد که می‌بایست با توجه به نیازمندی‌ها و اولویت‌های شبکه در نظر گرفته شود. هلر و همکاران در محاسبات خود اثر تغییر بار سیستم در طول زمان و تخصیص کنترلرها به‌صورت پویا را در نظر نگرفتند که با توجه به تغییر شدید ترافیک در شبکه‌های موبایل می‌بایست مورد نظر قرار داده شود.

در مرجع [33] این مسئله با استفاده از دیدگاه تئوری بازی‌ها[۲] بررسی شد که در آن راث[۳] و همکاران یک بازی مجموع-ناصفر[۴] برای حل کردن یک مسئله‌ی بهینه‌سازی که فرآیند اضافه و کم کردن تعداد کنترلرها در شبکه را مدل می‌کند ارائه کردند. همان‌طور که در رابطه‌ی (۲-۳) نشان داده‌شده است، تابع هدف (f) این مسئله بهینه‌سازی بصورت یک تابع غیرخطی از تعداد کنترلرها (k) و هزینه‌های موجود در شبکه (c) تعریف شده و برای سودمندی کنترلرها دو حد بالا U_{th} و پایین U_{α} و برای تأخیر پردازش بسته‌ها یک حد بالا Δ_{th} در نظر گرفته شده است. در این پژوهش بر خلاف چارچوب نرم‌افزاری پیشنهادی ما، معیاری برای بر قرار شدن توازن بار بین کنترلرها در محاسبات مورد بررسی قرار نگرفته است.

$$\begin{aligned} \min_{k} \quad & f(k, c) \\ \text{s.t.} \quad & \Delta_i \leq \Delta_{th}, \forall i \in I \\ & U_{\alpha} \leq U_i \leq U_{th}, \forall i \in I \end{aligned} \quad (۳-۲)$$

در مرجع [24] برای پیدا کردن تعداد کنترلرها از روش برنامه‌نویسی خطی اعداد صحیح[۵] استفاده شده است که در آن تعداد کنترلرها بر اساس هزینه‌ی جمع‌آوری اطلاعات آماری، هزینه‌ی زمانی، هزینه‌ی

[۱] Tradeoff
[۲] Game Theory
[۳] Rath
[۴] Non-zero sum Game
[۵] Integer Linear Programming (ILP)

هماهنگ‌سازی کنترلرها و هزینه‌ی اتصال سوئیچ‌ها به کنترلرها مشخص می‌شود ولی به دلیل بزرگ‌شدن فضای جواب با رشد اندازه و ترافیک شبکه‌ها این روش مستلزم زمان زیادی برای بدست آوردن جواب خواهد بود. در ادامه، مرجع [34] الگوریتمی برای برقراری توازن بار بین یک دسته از کنترلرها با استفاده از مفهوم تعادل واردراپ[۶] پیشنهاد می‌کند. مرجع [35] برای حل کردن مسئله‌ی جانمایی کنترلرها در یک توپولوژی شبکه‌ی گسترده[۷] از روش دسته‌بندی طیفی[۸] برای تقسیم‌بندی یک شبکه‌ی بزرگ به محدوده‌ی کوچک‌تر استفاده می‌نماید. مرجع [36] از دیدگاه متفاوتی مسئله را بررسی می‌کند و برای حل آن هزینه‌های نصب کنترلرها، هزینه‌های ارتباط بین سوئیچ‌ها و کنترلرها و هزینه‌های ارتباط کنترلرها با یکدیگر را مدنظر قرار می‌دهد.

به خاطر افزایش غیرقابل توقف ترافیک داده در شبکه‌های موبایل، فراهم‌سازی کنترلرها به‌صورت پویا تبدیل به مسئله‌ی NP-Hard می‌شود و به همین خاطر بهتر است برای حل کردن این مسائل از الگوریتم‌های بهینه‌سازی فرا ابتکاری استفاده شود. در این خصوص، مراجع [37, 38] با استفاده از الگوریتم بهینه‌سازی ازدحام ذرات[۱] روش‌هایی برای حل این مسئله ارائه نموده‌اند ولی به دلیل مشکلات این الگوریتم (مثل سرعت همگرایی کم و گیرافتادن در نقاط بهینه‌ی محلی) و کامل نبودن راه‌حل‌های ارائه‌شده در کارهای انجام‌شده‌ی قبلی، ما قصد داریم تا با استفاده از یک الگوریتم فرا ابتکاری جدید بانام گرگ‌های خاکستری، یک چارچوب نرم‌افزاری کامل و دقیق برای پیدا کردن تعداد بهینه‌ی کنترلرها در یک شبکه‌ی موبایل تعریف‌شده با نرم‌افزار ارائه نماییم که در فصل سوم به تفضیل در خصوص آن توضیح داده می‌شود. جدول (۲-۱) خلاصه‌ی کارهای انجام‌شده در زمینه‌ی جانمایی و فراهم‌سازی کنترلرهای SDN را نشان می‌دهد.

جدول ۲-۱: کارهای انجام‌شده برای جانمایی و فراهم‌سازی کنترلرهای SDN

نوآوری	نویسنده
معرفی مسئله و بررسی تأثیر مقدار متوسط و بدترین زمان تأخیر بر روی جواب‌ها	Heller et al. [32]
استفاده از تئوری بازی‌ها برای حل مسئله	Rath et al. [33]
استفاده از برنامه‌نویسی خطی برای حل مسئله و در نظر گرفتن هزینه‌های ارتباطی کنترلرها	Bari et al. [24]
ایجاد توازن بار بین کنترلرها با استفاده از مفهوم تعادل واردراپ	Cimorelli et al. [34]
استفاده از روش طیف دسته‌ای برای جاگذاری کنترلرها	Xiao et al. [35]
در نظر گرفتن هزینه‌های مربوط به نصب و اتصال کنترلرها	Sallahi et al. [36]
استفاده از الگوریتم PSO برای حل مسئله	Gao et al. [37]
استفاده از الگوریتم PSO برای حل مسئله	Liu et al. [38]

[۶] Wardrop Equilibrium
[۷] Wide Area Network (WAN)
[۸] Spectral Clustering
[۱] Particle Swarm Optimization (PSO)

۲-۲ مسیریابی جریان‌ها و جانمایی سرویس‌ها و توابع شبکه

شبکه‌های سنتی اغلب برای ارائه‌ی کلاس‌های مختلف کیفیت سرویس به کاربران از شبکه‌های ATM و IP استفاده می‌کردند [39]. شبکه‌های ATM به‌صورت ذاتی شامل توابعی برای رفع نیازمندی‌های کیفیت سرویس هست ولی شبکه‌های IP مکانیسم خاصی برای برآورده کردن نیازهای جریان‌های مختلف شبکه نداشتند. به همین خاطر، مکانیسم‌های جدیدی نظیر IntServ [40]، DiffServ [41] و MPLS [39] برای غلبه به این مشکل در شبکه‌های IP ارائه گردید. روش‌هایی که در این شبکه‌ها برای مسیریابی جریان‌ها با در نظر گرفتن کیفیت سرویس صورت می‌گرفت، اغلب بر اساس الگوریتمOSPF [39] و متدهای نظیر کوتاه‌ترین مسیر[۲]، پهن‌ترین مسیر[۳] و کوتاه‌ترین-پهن‌ترین مسیر[۴] بود که نیازمندی‌های تأخیر و پهنای باند در شبکه را رفع می‌نمودند. به همین دلیل، برآورده کردن نیازمندی‌های کیفیت سرویس در شبکه‌های سنتی به دلیل ذات توزیع‌شده‌ی آن‌ها کار بسیار دشواری بود ولی با پیدایش SDN و معرفی روشی برای مدیریت متمرکز شبکه، تضمین کردن کیفیت سرویس و مهندسی شبکه می‌تواند خیلی آسان‌تر انجام شود [42].

به همین منظور، تحقیقاتی برای به‌کارگیری SDN برای پاسخ‌گویی به مشکلات شبکه‌های سنتی صورت گرفته است که به بررسی برخی از آن‌ها می‌پردازیم. . مرجع [43] یک مدل شبکه‌ی جبری[۱] پیشنهاد می‌کند که در آن از روش کلکولس[۲] برای پیدا کردن مسیرهای بهینه برای جریان‌های شبکه استفاده کرده است. مرجع [44] نسخه‌ی بهبودیافته‌ی الگوریتم کوتاه‌ترین مسیر را با عنوان SFOP برای شناسایی بهترین مسیر موجود بین دو گره مبدأ و مقصد ارائه کرده است. نگارندگان مرجع [45] از الگوریتم‌های بسیار پیچیده‌ی ریاضی برای تضمین نیازمندی‌های کیفیت سرویس داده‌های چندرسانه‌ای و پخش ویدئو استفاده کرده است و از روشی چند-محدودیتی کوتاه‌ترین مسیر برای بهبود پخش ویدئو در مرجع [46] بهره جسته‌اند. علاوه بر این روش‌های جبری ذکرشده برای مسیریابی توأم با تضمین کیفیت سرویس، به دلیل ذات NP-Hard بودن مسئله‌ی مسیریابی، پژوهش‌های زیادی از روش‌های بهینه‌سازی فرا ابتکاری استفاده کرده‌اند. یکی از این تحقیقات در مرجع [47] صورت گرفته است که در آن با استفاده از الگوریتم بهینه‌سازی کلونی مورچگان[۳]، ازدحام[۴] شبکه کاهش یافته و به دنبال آن سودمندی[۵] لینک‌های شبکه افزایش می‌یابد.

توسعه‌ی NFV به ما اجازه می‌دهد تا توابع شبکه را به‌صورت مجازی برای روی سرورها و ماشین‌های مجازی پیاده‌سازی نماییم ولی جانمایی سرویس‌های مختلف خود نیز حائز اهمیت است چراکه جانمایی

[۲] Shortest Path
[۳] Widest Path
[۴] Shortest-Widest Path
[۱] Deterministic Network Model
[۲] Calculus
[۳] Ant Colony Optimization (ACO)
[۴] Congestion
[۵] Utilization

درست توابع و سرویس‌ها در NFV می‌تواند منجر به کاهش هزینه‌ها و افزایش انعطاف‌پذیری آن نسبت به استفاده از تجهیزات سخت‌افزاری میانی شود. به همین خاطر، تحقیقاتی در این زمینه صورت گرفته است که برخی از آن‌ها به شرح زیر می‌باشند. مرجع [48] برای حل مسئله‌ی زنجیره‌ی سرویس‌ها از الگوریتم تبرید شبیه‌سازی‌شده[۶] استفاده کرده است. مرجع [49] مسئله‌ی جانمایی توابع شبکه را برای شبکه‌های نسل چهارم موبایل (LTE) حل نموده و مرجع [50] با استفاده از روش برنامه‌نویسی ترکیبی اعداد صحیح نسبت به حل مسئله‌ی جانمایی توابع شبکه اقدام نموده است. همچنین، یک پژوهش دیگر نیز با استفاده از روش الگوریتم ژنتیک کوانتومی[۷] برای حل مسئله‌ی جانمایی سرویس‌های مجازی در شبکه در مرجع [13] صورت گرفته است.

به منظور بهبود و تکمیل کارهای انجام‌شده، ما در این پژوهش قصد داریم تا یک چارچوب نرم‌افزاری برای پیدا کردن مسیر بهینه برای جریان‌های ورودی در شبکه معرفی کنیم که در آن پس از پیدا شدن مسیر بهینه با توابع و سرویس‌های موردنیاز هر جریان با در نظر گرفتن توازن بار ابرک‌ها بر روی نودهای مناسب قرار داده می‌شوند.

۲-۳ جمع‌بندی

در این فصل، ابتدا به معرفی دقیق‌تر مسئله‌های «جانمایی و فراهم‌سازی کنترلرها» و «مسیریابی جریان‌های شبکه و جانمایی سرویس‌ها و توابع شبکه» پرداختیم و در ادامه با معرفی کارهای انجام‌شده در رابطه با این مسائل، نقاط قوت و ضعف آن‌ها را بیان نمودیم ولی با توجه به کامل نبودن راه‌حل‌های ارائه‌شده در کارهای پیشین قصد داریم تا در فصول بعدی از این پایان‌نامه دو چارچوب نرم‌افزاری به‌منظور بهبود حل این مسائل باهدف در نظر گرفتن نیازمندی‌های نسل پنجم استاندارد موبایل ارائه نماییم.

[۶] Simulated Annealing (SA)
[۷] Quantum Genetic-based Algorithm

فصل سوم

چارچوب نرم‌افزاری برای فراهم‌سازی کنترلرهای SDN

در این فصل قصد داریم تا یک چارچوب نرم‌افزاری جدید برای فراهم‌سازی کنترلرها در شبکه‌های موبایل تعریف‌شده با نرم‌افزار ارائه نماییم. بدین منظور، ابتدا به تعریف فرضیات و علایم مورد نیاز برای مدل‌سازی و حل مسئله پرداخته و در ادامه در مورد جزئیات مدل‌سازی و حل مسئله بحث می‌نماییم.

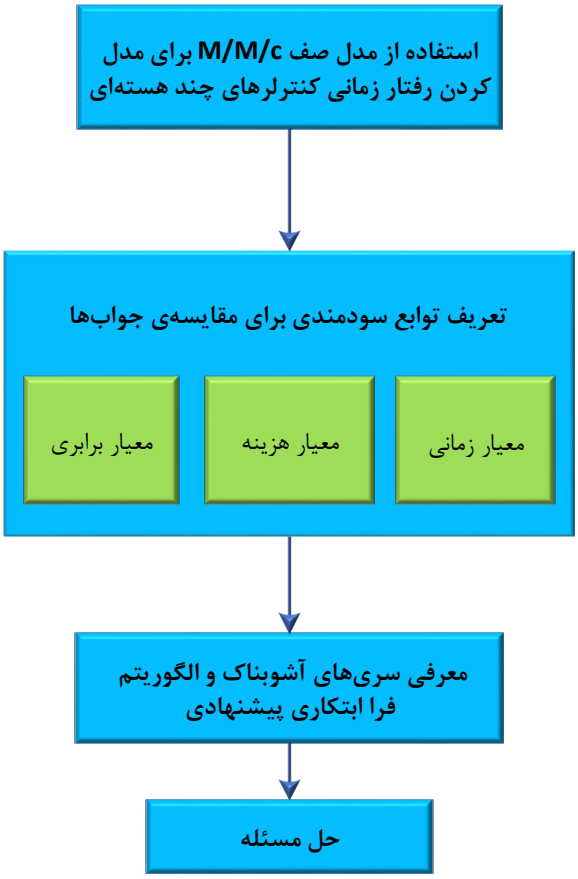
۳-۱ فرضیات و مراحل حل مسئله

معماری شبکه‌های تعریف‌شده با نرم‌افزار از ۳ بخش اصلی تشکیل‌شده است. ازآنجایی‌که دو بخش از این معماری یعنی RAN و ابر ممکن است نیازمندی‌های زمانی متفاوتی داشته باشند، در این فصل فرض کرده‌ایم که هریک از بخش‌ها می‌تواند به N کلاس مختلف تقسیم‌بندی شود و این کلاس‌ها را با $\Phi_i(i = 1, 2, ..., N)$ نمایش داده‌ایم. در ادامه با این فرض که هریک از این کلاس‌ها می‌توانند تعداد متفاوتی سوئیچ‌های پشتیبانی‌کننده از پروتکل OpenFlow داشته باشند، تعداد سوئیچ‌های موجود در هر کلاس را با M_i نشان می‌دهیم. شناسه‌ی کنترلر j ام از کلاس Φ_i را نیز با C_j^i نمایش می‌دهیم که در آن $1 \leq C_j^i \leq M_i$ خواهد بود. برای حل مسئله‌ی فراهم‌سازی پویای کنترلرها، می‌بایست ابتدا تعداد بهینه‌ی کنترلرهای هر کلاس (n_i) مشخص شود و سپس با توجه به عدد محاسبه‌شده برای هر کلاس بهترین ارتباط بین کنترلرها و سوئیچ‌ها باهدف رفع نیازمندی‌های کیفیت سرویس در شبکه محاسبه می‌شود. همان‌طور که در شکل (۳-۱) نشان داده‌شده است، برای حل مسئله به ترتیب زیر عمل می‌کنیم:

ابتدا، فرض می‌کنیم که کنترلرهای شبکه بر روی ماشین‌های مجازی چند هسته‌ای مشابه (با توان پردازشی و حافظه‌ی برابر) پیاده‌سازی شده‌اند و سپس با استفاده از مدل صف $M/M/c$ رفتار زمانی آن‌ها را مدل می‌کنیم. در مرحله‌ی بعدی، ازآنجایی‌که برای یافتن بهترین جواب نیاز داریم تا جواب‌های مختلف را با یکدیگر مقایسه کنیم، سه معیار مختلف برای مقایسه‌ی جواب‌ها تعریف می‌کنیم. در مرحله‌ی سوم از حل مسئله، به معرفی الگوریتم‌های بهینه‌سازی فرا ابتکاری و سری‌های آشوبناک پرداخته و با استفاده از این سری‌ها عملکرد الگوریتم گرگ‌های خاکستری را بهبود می‌بخشیم. در آخر نیز با استفاده از بخش‌های قبلی به حل مسئله می‌پردازیم.

۳-۲ استفاده از مدل صف $M/M/c$ برای مدل کردن رفتار زمانی کنترلرهای چند هسته‌ای

با توجه به پیشرفت‌های صورت گرفته در تکنولوژی، امروزه اغلب ماشین‌های مجازی می‌توانند به صورت چند هسته‌ای اجرا شوند. به همین منظور، ما فرض کرده‌ایم که کنترلرهای SDN بکار برده‌شده در شبکه‌های موبایل بر روی ماشین‌های مجازی با ۴ هسته اجرا می‌شوند و رفتار زمانی آن‌ها می‌تواند با



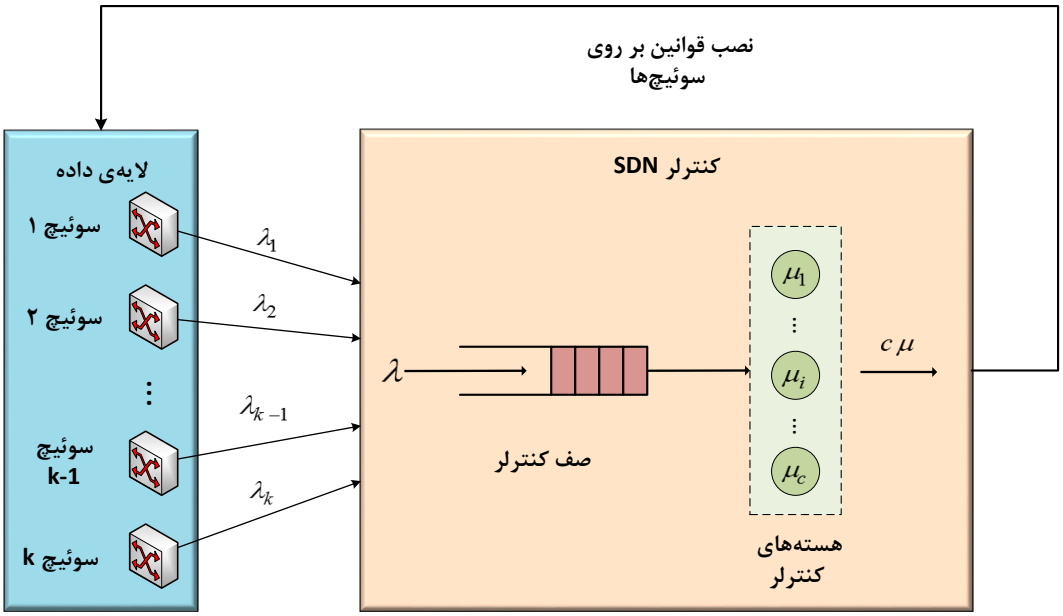
شکل ۳-۱: مراحل مدل‌سازی و حل مسئله

استفاده از مدل $M/M/c$ یا مدل ارلانگ سی [1] [51, 52] که یک مدل صف برای توصیف سیستم‌های چند سروری است، مدل شود ولی در صورت استفاده از ماشین‌های مجازی با یک هسته، رفتار آن‌ها را می‌توان با مدل $M/M/1$ و رابطه‌ی (۳-۱) مدل کرد [53].

$$t_S = \frac{1}{\mu - \lambda} \qquad (۳-۱)$$

همان‌طور که در شکل (۳-۲) نشان داده شده است، صفحه‌ی داده که شامل سوئیچ‌های شبکه است، درخواست‌های خود را به کنترلر ارسال کرده و کنترلر پس از پردازش بسته‌های دریافتی (درخواست سوئیچ‌ها) قوانین مناسب را بر روی سوئیچ‌ها نصب می‌نماید. برای استفاده از این مدل صف، فرض می‌کنیم که سوئیچ‌ها با توزیع پواسون درخواست‌های خود را برای کنترلر ارسال می‌کنند و کنترلر بسته‌های دریافتی را در یک صف قرار می‌دهد و سپس هریک از c هسته‌ی موجود در کنترلر (در اینجا ۴ هسته) بسته‌های دریافت شده را با توزیع نمایی [1] پردازش می‌کنند.

[1] Erlang-C
[1] Exponential



شکل ۳-۲: کنترلر مدل‌شده با مدل $M/M/c$

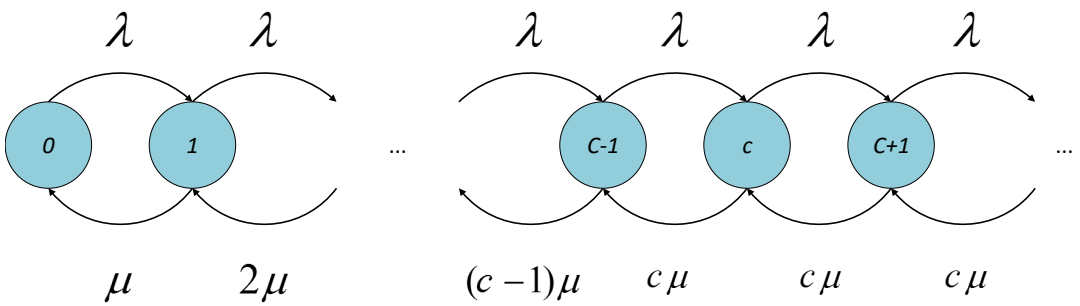
برای توصیف بهتر این مدل صف، ابتدا به توضیح علایم و پارامترهای بکار برده شده در آن می‌پردازیم:

- λ_k: متوسط نرخ ارسال درخواست توسط سوئیچ k ام که از توزیع پواسون پیروی می‌کند.
- λ: مجموع λ_j ها در یک کلاس Φ_i ($\lambda = \sum_{j=1}^{k} \lambda_j$)
- μ: نرخ سرویس‌دهی کنترلر که از توزیع نمایی پیروی می‌کند.
- c: تعداد هسته‌های هر کنترلر

در این مدل، حالت‌های مختلف صف در کنترلر را می‌توان با پروسه‌ی تولد-مرگ[۱] توصیف نمود و همان‌طور که شکل (۳-۳) نشان داده شده است، گذر حالت‌های آن مشابه یک زنجیره‌ی مارکوف[۲] خواهد بود. زمانی که تعداد بسته‌های موجود در کنترلر کمتر از c باشد، توان پردازشی کنترلر برابر با $k\mu$ و زمانی که این مقدار از c تجاوز کند ($k \geq c$)، توان پردازشی کنترلر به بیش‌ترین حد خود یعنی $c\mu$ می‌رسد.

اگر ρ را به صورت $\frac{\lambda}{c\mu}$ تعریف کنیم، تا زمانی که $\rho < 1$، سیستم در حالت پایدار خواهد بود و همان‌طور تئوری صف [51, 52] پیشنهاد می‌دهد، در حالت پایدار احتمال حالت k ام (P_k)، حالتی که در کنترلر k

[۱] Birth-Death
[۲] Markovian Chain



شکل ۳-۳: گذر حالت‌ها در مدل $M/M/c$

بسته وجود داشته باشد به صورت زیر قابل محاسبه است:

$$P_k = \begin{cases} \frac{P_0 c^k \rho^k}{k!} & \text{اگر } 0 \le k \le c \\ \frac{P_0 c^c \rho^k}{c!} & \text{اگر } k > c \end{cases} \qquad (۲-۳)$$

آنجایی که مجموع این احتمالات می‌بایست برابر با یک شود، می‌توانیم P_0 را به صورت زیر محاسبه کنیم:

$$\sum_{k=0}^{\infty} P_k = 1 \qquad (۳-۳)$$

$$P_0 = [\sum_{i=0}^{c-1} \frac{(c\rho)^i}{i!} + \frac{(c\rho)^c}{c!(1-\rho)}]^{-1} \qquad (۴-۳)$$

و در ادامه، با توجه به فرمول ارلانگ‌سی (رابطه‌ی ۳-۶) می‌توانیم متوسط زمان پاسخ‌دهی هر کنترلر (t_S) را که برابر با مجموع مدت زمان انتظار در صف و مدت زمان سرویس‌دهی است را به صورت زیر محاسبه نماییم:

$$\frac{\rho}{1-\rho} C(c, \frac{\lambda}{\mu}) + c\rho \qquad (۵-۳)$$

$$C(c, \frac{\lambda}{\mu}) = \frac{1}{1 + (1-\rho)(\frac{c!}{(c\rho)^c}) \sum_{i=0}^{c-1} \frac{(c\rho)^i}{i!}} \quad (۶-۳)$$

۳-۳ تعریف توابع سودمندی برای مقایسه‌ی جواب‌ها

بعد از محاسبه‌ی زمان متوسط پاسخگویی کنترلرها (t_S)، برای بررسی اینکه نیازمندهای زمانی توسط کنترلر برآورده می‌شود یا خیر می‌بایست به تعریف یک تابع بپردازیم. این تابع می‌بایست در مواقعی که زمان متوسط پاسخ‌دهی از t_{QoS} بیش‌تر می‌شود، به‌شدت کاهش پیدا کند. مناسب‌ترین تابع برای این رفتار مربوط به توابع نمایی است ولی ازآنجایی‌که توابع نمایی، سربار محاسباتی زیادی برای سیستم ایجاد می‌کنند، برای رسیدن به این رفتار از توابع چندجمله‌ای به شرح زیر برای تعریف تابع سودمندی زمانی استفاده کرده‌ایم:

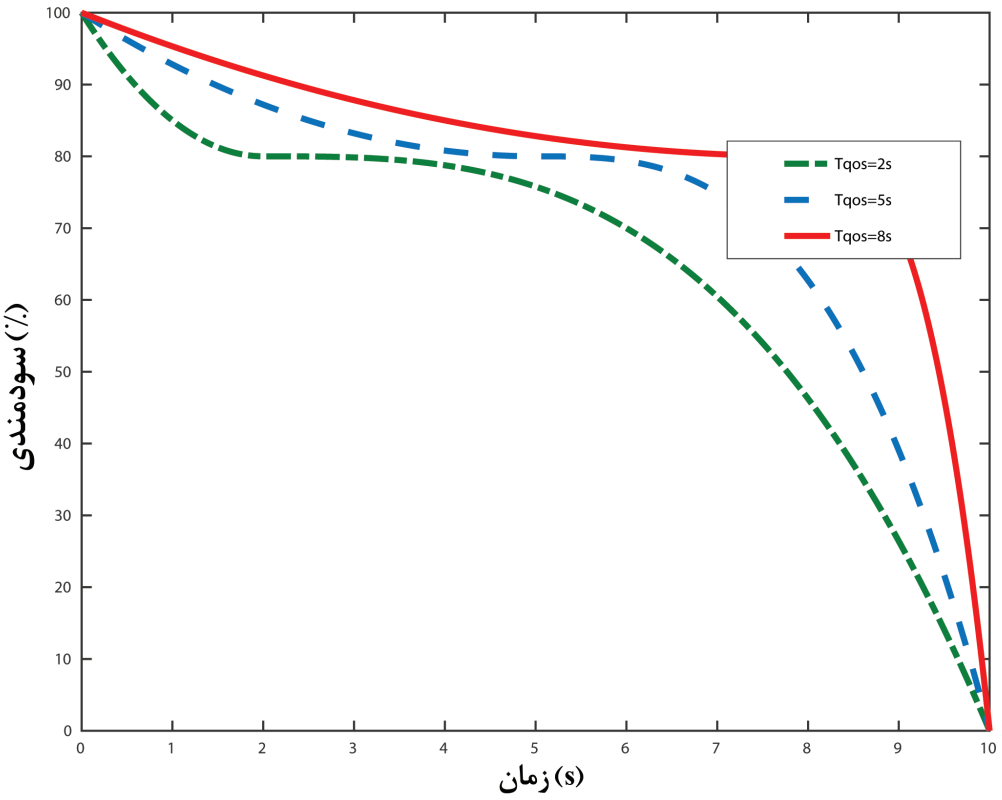
$$T_j = T(t_j) = \begin{cases} K + \alpha(t_j - t_{QoS})^2 & \text{اگر } t_j \leq t_{QoS} \\ K - \beta(t_j - t_{QoS})^3 & \text{اگر } t_j > t_{QoS} \end{cases} \quad (۷-۳)$$

در این رابطه، K یک عدد ثابت بوده که بیش‌ترین دامنه‌ی تابع را در زمان $t_j = t_{QoS}$ مشخص می‌کند. پس از آزمودن مقادیر مختلف برای K، درنهایت مقدار ۸۰ را برای آن برگزیدیم. α و β نیز مقادیر ثابتی هستند که نرخ افزایش/کاهش تابع را با توجه به محدوده‌ی زمان پاسخ‌دهی (t_j) و حدنصاب موردنیاز سیستم برای برآورده کردن نیازمندی‌های کیفیت سرویس (t_{QoS}) مشخص می‌کنند. برای اینکه مقدار سودمندی سیستم بین مقدار ۰ و ۱۰۰ تغییر کند از روابط زیر برای محاسبه‌ی α و β استفاده کرده‌ایم.

$$\alpha = \frac{(100 - K)}{t_{QoS}^2} \quad (۸-۳)$$

$$\beta = \frac{K}{(max(t_j) - t_{QoS})^3} \quad (۹-۳)$$

شکل (۳-۴) این تابع را برای t_{QoS} های مختلف و بازه‌ی $0 \leq t_j \leq 100$ نشان می‌دهد.



شکل ۳-۴: تابع سودمندی برای برآورده کردن نیازهای زمانی کیفیت سرویس

برای جلوگیری از اختصاص دادن بیش‌ازحد کنترلر در شبکه، یک تابع سودمندی برای هزینه‌ی کنترلرها تعریف می‌کنیم. اضافه کردن/روشن کردن یک کنترلر جدید می‌بایست منجر به افزایش هزینه‌ها و کاهش سودمندی شود و کم کردن/خاموش کردن یک کنترلر نیز باید این مقدار را افزایش دهد. برای کاهش محاسبات، برای این تابع از ساده‌ترین تابع ممکن یعنی یک تابع کسری یا هموگرافیک که از این رفتار تبعیت می‌کند استفاده کرده‌ایم که در آن n_i نشان‌دهنده‌ی تعداد کنترلرها در کلاس Φ_i بوده و K_{max} بیش‌ترین دامنه‌ی این مقدار سودمندی را مشخص می‌کند. مقدار بیشینه این تابع زمانی اتفاق می‌افتد که تنها یک کنترلر داشته باشیم و با افزایش مقدار n_i به‌تدریج مقدار آن کاهش خواهد یافت. ازآنجایی‌که می‌خواهیم مقادیر سودمندی در سیستم در بازه‌ی $0 \leq C^i \leq 100$ قرار گیرند، در اینجا مقدار ۱۰۰ را برای K_{max} انتخاب می‌کنیم.

$$C^i = \frac{K_{max}}{n_i} \quad (۱۰-۳)$$

یکی دیگر از معیارهای مهم در مشخص کردن تعداد کنترلرها، توزیع عادلانه‌ی سوئیچ‌ها بین کنترلرهاست به‌طوری‌که بین کنترلرهای مختلف توازن بار برقرار شود. برای برآورده کردن این شرط، ما از شاخص برابری جین [۱] [54] برای اندازه‌گیری برابری [۲] در فراهم‌سازی کنترلرها استفاده کرده‌ایم. با فرض اینکه بار متوسط هر کنترلر را می‌توان با استفاده از $\rho = \frac{\lambda}{c\mu}$ محاسبه نمود، شاخص برابری جین از رابطه‌ی زیر قابل‌محاسبه است:

$$F^i = \frac{(\sum_{j=1}^{n_i} \rho_j)^2}{n_i \sum_{j=1}^{n_i} \rho_j^2} \quad (۱۱-۳)$$

حال که تمامی معیارهای مختلف برای مقایسه‌ی جواب‌های مسئله را تعریف کرده‌ایم، باید بتوانیم به وضعیت کلی مرکز داده (ابر) یک مقدار اختصاص دهیم. بدین منظور، ابتدا مقدار سودمندی هر کلاس را به‌صورت میانگین وزن‌دار معیار زمانی (رابطه‌ی ۳-۷)، معیار هزینه (رابطه‌ی ۳-۱۰) و معیار برابری (رابطه‌ی ۳-۱۱) مطابق زیر محاسبه می‌کنیم که در آن γ_T، γ_C و γ_F وزن‌های هریک از توابع سودمندی بوده و مقدار سودمندی زمانی برای هر کلاس (T^i) برابر با متوسط مقدار سودمندی زمانی تمامی کنترلرهای موجود در آن کلاس (T_j) است.

$$U_i = \gamma_T.T^i + \gamma_C.C^i + \gamma_F.F^i \quad (۱۲-۳)$$

$$T_i = \frac{1}{n_i} \sum_{j=1}^{n_i} T_j \quad (۱۳-۳)$$

[۱] Jain's Fairness Index
[۲] Fairness

۳-۴ معرفی سری‌های آشوبناک و الگوریتم فرا ابتکاری پیشنهادی

در این بخش ابتدا به معرفی الگوریتم‌های بهینه‌سازی فرا ابتکاری پرداخته و در ادامه پس از معرفی نسخه‌ی بهبود یافته‌ی الگوریتم گرگ‌های خاکستری با استفاده از سری‌های آشوبناک به حل مسئله‌ی بهینه‌سازی مدل‌شده در بخش‌های پیشین می‌پردازیم.

۳-۴-۱ مقدمه‌ای در خصوص الگوریتم‌های بهینه‌سازی فرا ابتکاری و سری‌های آشوبناک

الگوریتم‌های بهینه‌سازی فرا ابتکاری یکی از روش‌های پرکاربرد برای حل مسائل مختلف بهینه‌سازی هستند و ویژگی‌های خاص آن‌ها نظیر سادگی[۱]، مقیاس‌پذیری[۲]، حجم محاسبات کم و بدون مشتق بودن، این الگوریتم‌ها را گزینه‌ی مناسبی برای حل مسائل بهینه‌سازی NP-Hard می‌کند. در پروسه‌ی حل مسائل بهینه‌سازی، الگوریتم‌های فرا ابتکاری از دو روش برای پیدا کردن بهترین جواب استفاده می‌کنند:

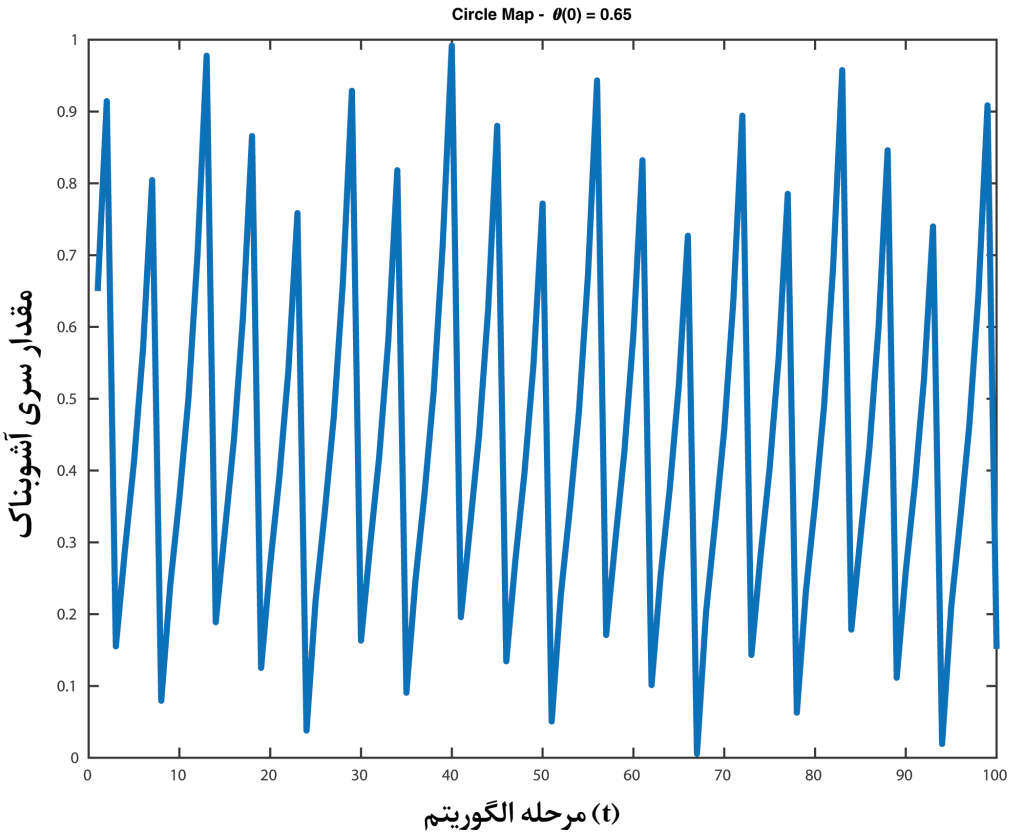
- **اکتشاف**[۳]: جستجوی کل فضای جواب
- **استثمار**[۴]: جستجو در اطراف بهترین جواب به‌دست‌آمده در مراحل قبل

این الگوریتم‌ها به دودسته‌ی کلی تک جوابی[۵] و چند جوابی[۶] تقسیم می‌شوند. الگوریتم‌های تک جوابی مثل الگوریتم تبرید شبیه‌سازی‌شده [55] بر روی یک جواب تمرکز می‌کنند و در طول زمان این جواب را بهتر می‌کنند ولی در الگوریتم‌های چند جوابی مثل الگوریتم ژنتیک[۷] [56] و الگوریتم بهینه‌سازی ازدحام ذرات[۸] [57] ابتدا چندین جواب اولیه تولید می‌شود و در طول زمان الگوریتم با ترکیب و تغییر این جواب‌ها به بهترین جواب موجود می‌رسد. علی‌رغم ویژگی‌های بسیار خوب الگوریتم‌های فرا ابتکاری، این الگوریتم‌ها در بعضی‌اوقات در نقاط بهینه‌ی محلی گیرکرده و در پروسه‌ی همگرایی[۹] بسیار آهسته عمل می‌کنند. به همین منظور، تحقیقات زیادی برای حل کردن این دو مشکل انجام‌شده و نسخه‌های جدیدی از الگوریتم‌ها در طول زمان ارائه‌شده است. ما نیز برای بهبود این دو مشکل در چارچوب نرم‌افزاری خود، از الگوریتم گرگ‌های خاکستری[۱۰] [58] استفاده می‌کنیم که پروسه‌ی یافتن جواب را نسبت به الگوریتم‌هایی مثل PSO بسیار بهبود می‌بخشد. همچنین، برای افزایش عملکرد الگوریتم گرگ‌های خاکستری ازسری‌های آشوبناک[۱۱] [59] که یکی از شناخته‌شده‌ترین روش‌ها برای جلوگیری از افتادن در نقاط بهینه‌ی محلی و افزایش سرعت همگرایی هستند استفاده کرده‌ایم. این سری‌ها توابع خاصی هستند که بجای تولید اعداد

[۱] Simplicity
[۲] Scalability
[۳] Exploration
[۴] Exploitation
[۵] Single Solution
[۶] Population-based
[۷] Genetic Algorithm (GA)
[۸] Particle Swarm Optimization (PSO)
[۹] Convergence
[۱۰] Grey Wolf Optimizer (GWO)
[۱۱] Chaotic Maps

کاملاً تصادفی، سری‌های تصادفی تولید می‌کنند که از یک رفتار جبری[۱۲] پیروی می‌کند و این امر باعث می‌شود تا عملکرد الگوریتم‌ها در پروسه‌ی جستجو و یافتن جواب بهینه، بهتر شود. سری‌های آشوبناک زیادی وجود دارند که برخی از آن‌ها در جدول (۳-۱) نشان داده‌شده است و ما پس از آزمودن آن‌ها در چارچوب نرم‌افزاری‌مان تابع Circle Map [60] را که بهترین نتیجه را داشت برای بهبود عملکرد الگوریتم گرگ‌های خاکستری برگزیدیم. این تابع (رابطه‌ی ۳-۱۴) با تنظیم پارامترهای آن به‌صورت $a = 0.5$ و $b = 0.2$ اعداد تصادفی در بازه $(0, 1)$ تولید می‌کند و ما از آن برای تولید تمامی اعداد تصادفی موردنیاز در چارچوب نرم‌افزاری استفاده کرده‌ایم. شکل (۳-۵) نمونه‌ای از اعداد تولیدشده توسط این تابع را با مقدار اولیه‌ی $\theta(1) = 0.65$ نشان می‌دهد.

$$\theta(t+1) = mod(1, [\theta(t) + b - \frac{a}{2\pi} sin(2\pi\theta(t)]) \qquad (۱۴-۳)$$



شکل ۳-۵: تابع Circle Map

[۱۲] Deterministic Random Sequences

جدول ۳-۱: توابع آشوبناک

نام تابع	رابطه‌ی تابع	بازه‌ی اعداد خروجی
Logistic Map [59]	$\theta(t+1) = a\theta(t)(1-\theta(t)), a = 4$	$(0,1)$
Tent Map [59]	$\theta(t+1) = \begin{cases} \frac{\theta}{0.7} & \text{اگر } \theta(t) < 0.7 \\ \frac{10}{3\theta(t)(1-\theta(t))} & \text{اگر } \theta(t) \geq 0.7 \end{cases}$	$(0,1)$
Sinusoidal Map [59]	$\theta(t+1) = a\theta^2(t)\sin(\pi\theta(t)), a = 2.3$	$(0,1)$
Gauss Map [59]	$\theta(t+1) = \begin{cases} 0 & \text{اگر } \theta(t) = 0 \\ \frac{1}{mod(1,\theta(t))} = \frac{1}{\theta(t)} - [\frac{1}{\theta(t)}] & \text{اگر } \theta(t) \neq 0 \end{cases}$	$(0,1)$
Circle Map [60]	$\theta(t+1) = mod(1, [\theta(t) + b - \frac{a}{2\pi} sin(2\pi\theta(t)])$	$(0,1)$
Chebyshev [59]	$\theta(t+1) = \cos(t\cos^{-1}(\theta(t))$	$(-1,1)$

۳-۴-۲ الگوریتم گرگ‌های خاکستری بهبود داده‌شده

الگوریتم گرگ‌های خاکستری یکی از جدیدترین الگوریتم‌ها بهینه‌سازی فرا ابتکاری بوده که از سلسله‌مراتب اجتماعی[۱] و شکار کردن گرگ‌های خاکستری الهام گرفته‌شده است. گرگ‌های خاکستری واقعی به‌صورت قبیله‌ای زندگی می‌کنند و قبیله‌ی آن‌ها شامل ۴ نوع دسته‌ی مختلف گرگ می‌شود. بالاترین رده در قبیله مربوط به گرگ‌های آلفاست که وظیفه‌ی رهبری را بر عهده‌دارند و بقیه‌ی گروه باید از دستورات آن‌ها پیروی کنند. گرگ‌های بتا و دلتا به ترتیب در رده‌های دوم و سوم گروه قرار دارند و به گرگ‌های آلفا درروند رهبری گروه کمک می‌کنند. بقیه‌ی گروه نیز بانام گرگ‌های امگا شناخته می‌شوند که پایین‌ترین رده را در گروه دارند و باید از سه گروه دیگر در تمامی کارها پیروی کنند. همانند بقیه‌ی الگوریتم‌های چند جوابی، الگوریتم گرگ‌های خاکستری از تعدادی جواب بانام کارگزار[۲] تشکیل‌شده است که این کارگزاران هریک به دنبال جواب بهینه در فضای جواب می‌گردند. برای اعمال سلسله‌مراتب اجتماعی گرگ‌های خاکستری بر روی این کارگزاران، به بهترین جواب پیداشده در هر مرحله از الگوریتم جواب آلفا و به دومین و سومین جواب‌های بعدازآن به ترتیب جواب بتا و جواب دلتا گفته می‌شود و بقیه‌ی کارگزاران بانام جواب‌های امگا شناخته می‌شوند.

علاوه بر سلسله‌مراتب اجتماعی گرگ‌های خاکستری، عملکرد شکار کردن آن‌ها نیز در ارائه‌ی یک الگوریتم فرا ابتکاری موردنظر قرارگرفته است. در طبیعت، زمانی که این گرگ‌ها یک شکار را پیدا می‌کنند، ابتدا دور آن شکار حلقه می‌زنند و در ادامه آن‌قدر شعاع دایره‌ی حلقه‌شان را کم می‌کنند تا دیگر شکار قادر به حرکت کردن نباشد. این رفتار نیز توسط مرجع [58] با استفاده از روابط زیر به‌صورت ریاضی مدل شده است.

[۱] Social Hierarchy
[۲] Agent

$$\vec{D_\alpha} = \left|\vec{C_\alpha}.\vec{\alpha} - \vec{\alpha}\right|, \vec{D_\beta} = \left|\vec{C_\beta}.\vec{\beta} - \vec{\beta}\right|, \vec{D_\delta} = \left|\vec{C_\delta}.\vec{\delta} - \vec{\delta}\right| \quad (۱۵-۳)$$

$$\vec{\alpha} = (\vec{\alpha} - \vec{A_\alpha}.\vec{D_\alpha}), \vec{\beta} = (\vec{\beta} - \vec{A_\beta}.\vec{D_\beta}), \vec{\delta} = (\vec{\delta} - \vec{A_\delta}.\vec{D_\delta}) \quad (۱۶-۳)$$

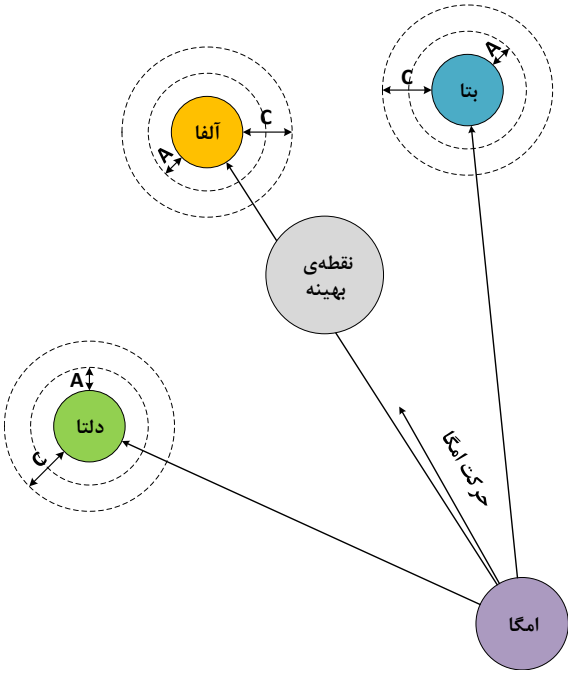
$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a} \quad (۱۷-۳)$$

$$\vec{C} = 2\vec{r_2} \quad (۱۸-۳)$$

در این روابط، A_k و C_k ($k = \alpha, \beta, \delta$) ضرایب برداری هستند و برای به‌روزرسانی آن‌ها در طول اجرای الگوریتم از روابط (۳-۱۷) و (۳-۱۸) استفاده می‌شود. $\vec{a}$ به‌عنوان پارامتری برای شبیه‌سازی رفتار حلقه‌زدن گرگ‌ها عمل می‌کند و در طول زمان به‌صورت خطی از ۲ تا ۰ کم می‌شود تا شعاع حلقه‌ی کارگزاران نیز در الگوریتم در طول زمان کم شود. r_1 و r_2 بردارهای تصادفی هستند که وجود آن‌ها منجر به رفتار اکتشافی در الگوریتم می‌شود. در هر مرحله از الگوریتم برای اعمال نمودن سلسله‌مراتب اجتماعی گرگ‌ها بر گروه کارگزاران، ابتدا جواب‌های آلفا، بتا و دلتا انتخاب می‌شوند و سپس با فرض این‌که این سه جواب اطلاعات بهتری نسبت به موقعیت نقطه‌ی بهینه دارند، موقعیت جواب‌های امگا با توجه به موقعیت این سه جواب و با استفاده از رابطه‌ی (۳-۱۹) و مطابق شکل (۳-۶) در فضای جواب تغییر می‌کنند.

$$\vec{\omega} = \frac{\vec{\alpha} + \vec{\beta} + \vec{\delta}}{3} \quad (۱۹-۳)$$

شبه کد مربوط به این الگوریتم در الگوریتم (۱) در بخش ضمیمه نشان داده‌شده است که در آن هریک از کارگزاران الگوریتم گرگ‌های خاکستری که نشان‌دهنده‌ی تعداد کنترلرها و یا شناسه‌ی کنترلر متصل شده به سوئیچ k ام در کلاس Φ_i هستند، با استفاده از تابع Circle Map مقداردهی اولیه می‌شوند. برای



شکل ۳-۶: حلقه زدن کارگزاران دور نقطه‌ی بهینه

محاسبه‌ی تابع هزینه‌ی این کارگزاران نیز از رابطه‌ی پیشنهادشده برای سودمندی مرکز داده (۳-۱۲) استفاده می‌شود.

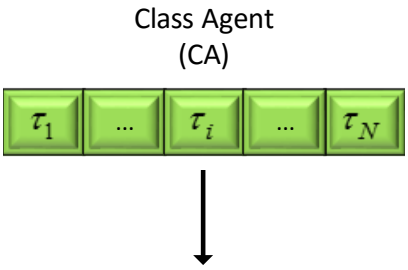
۳-۴-۳ حل مسئله

مسئله فراهم‌سازی کنترلرهای شبکه‌های موبایل تعریف‌شده با نرم‌افزار را می‌توان به‌عنوان یک مسئله معروف بهینه‌سازی با عنوان مسئله کوله‌پشتی چندتایی[۱] در نظر گرفت و آن را به‌راحتی با استفاده از الگوریتم معرفی‌شده در زیر بخش قبلی حل نمود ولی ابتدا می‌بایست فرم جواب‌های الگوریتم یا به عبارتی فرم کارگزاران الگوریتم گرگ‌های خاکستری را مشخص نماییم و سپس با استفاده از الگوریتم (۱) به حل مسئله بپردازیم. ازآنجایی‌که مرکز داده‌ی مورداستفاده در شبکه‌های موبایل دارای چندین کلاس کیفیت سرویس می‌باشد، ما باید از دو الگوریتم تودرتوی Chaotic GWO استفاده نماییم. در پروسه‌ی حل مسئله، ابتدا لایه‌ی اول این الگوریتم تعداد بهینه‌ی کنترلرها برای هر کلاس را مشخص می‌کند و در ادامه لایه‌ی دوم با توجه به عدد محاسبه‌شده توسط لایه‌ی اول بهترین ارتباط بین کنترلرها و سوئیچ را به دست می‌آورد؛ بنابراین برای حل این مسئله نیاز داریم تا برای هریک از این لایه‌ها یک فرم جواب (کارگزار) به شرح زیر تعریف کنیم:

۱. **کارگزار کلاس**[۲]: این کارگزار مربوط به لایه‌ی اول از الگوریتم‌ها بوده و همان‌طور که در شکل (۳-۷) نشان داده است، به‌صورت یک بردار $N \times 1$ است که هر المان آن (τ_i) تعداد کنترلرهای

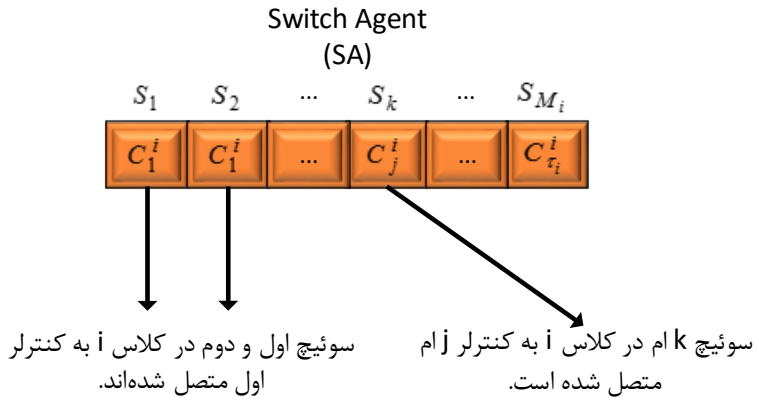
[۱] Multi Knapsack Problem (MKP)
[۲] Class Agent (CA)

موردنیاز برای کلاس Φ_i را مشخص می‌نماید. N نیز تعداد کلاس‌ها در شبکه خواهد بود.



شکل ۳-۷: کارگزار کلاس

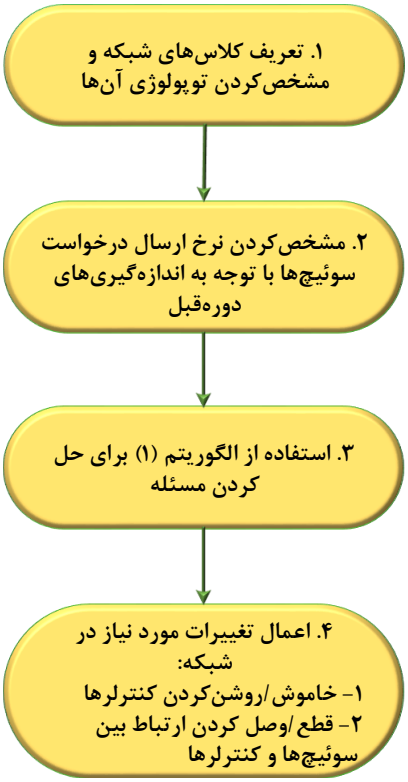
۲. **کارگزار سوئیچ**[1]: این کارگزار مربوط به لایه‌ی دوم الگوریتم بوده و می‌بایست اتصالات مناسب بین سوئیچ‌ها و کنترلرها را محاسبه کند. لذا، همان‌طور که در شکل (۳-۸) نشان‌داده شده است، آن را به‌صورت یک بردار $M_i \times 1$ تعریف کرده‌ایم که در آن هر المان بردار نشان‌دهنده‌ی شناسه‌ی کنترلری است که به سوئیچ k ام از کلاس Φ_i باید متصل شود که مقدار این شناسه می‌تواند بین ۱ تا τ_i تغییر کند.



شکل ۳-۸: کارگزار سوئیچ

پس از پیدا نمودن تعداد کنترلرهای بهینه و بهترین ارتباط بین سوئیچ‌ها و کنترلرها توسط الگوریتم (۱)، می‌بایست تغییراتی در شبکه ایجاد نماییم. ابتدا باید کنترلرهای اضافی را خاموش کرده و در صورت اضافه شدن کنترلر جدید به کلاس‌ها باید آن‌ها را بر روی ماشین‌های مجازی مناسب اجرا کنیم و به شبکه اضافه نماییم. شکل (۳-۹) مراحل مطرح‌شده برای حل مسئله را نشان می‌دهد.

[1] Switch Agent (SA)



شکل ۳-۹: مراحل حل مسئله

۳-۵ جمع‌بندی

در این بخش یک چارچوب نرم‌افزاری بر اساس الگوریتم‌های فرا ابتکاری برای فراهم‌سازی پویای کنترلرها در شبکه‌های موبایل تعریف‌شده در نرم‌افزار ارائه کردیم که در آن سه معیار زمان، هزینه و برابری برای فراهم‌سازی کنترلرها در نظر گرفته‌شده بود. این چارچوب نرم‌افزاری از دولایه‌ی مختلف تشکیل‌شده که هرکدام با استفاده از نسخه‌ی بهبودیافته و آشوبناک الگوریتم گرگ‌های خاکستری پیاده‌سازی شده‌اند. لایه‌ی اول وظیفه‌ی پیدا کردن تعداد کنترلرهای موردنیاز برای هر کلاس کیفیت سرویس را مشخص می‌کند و لایه‌ی دوم با توجه به تعداد کنترلرهای تخصیص داده‌شده به هر کلاس، ارتباط مناسب بین کنترلرها و سوئیچ‌ها را محاسبه می‌کند. در آخر، پس از مشخص‌شدن تعداد مناسب کنترلرها و ارتباط‌های مناسب بین کنترلرها و سوئیچ‌ها می‌بایست تغییرات موردنیاز برای روشن/خاموش شدن کنترلرها و قطع/وصل شدن ارتباطات صورت گیرد.

فصل چهارم
چارچوب نرم‌افزاری برای مسیریابی جریان‌ها و جانمایی سرویس‌ها

در این فصل قصد داریم تا یک چارچوب نرم‌افزاری برای مسیریابی جریان‌ها و جانمایی سرویس‌ها با در نظر گرفتن نیازمندی‌های نسل پنجم موبایل معرفی کنیم. چارچوب نرم‌افزاری پیشنهادی ما هر بار که یک جریان جدید وارد شبکه می‌شود با بهره‌گیری از ویژگی‌های شبکه‌های تعریف‌شده با نرم‌افزار، مجازی‌سازی توابع شبکه و الگوریتم‌های فرا ابتکاری، مسیر مناسب بین مبدأ و مقصد را برای جریان ورودی پیداکرده و سپس برای سرویس‌های موردنیاز جریان ورودی بهترین جانمایی سرویس‌ها را محاسبه می‌کند. به‌منظور مشخص‌شدن دلیل ارائه‌ی چنین چارچوب نرم‌افزاری برای شبکه‌های مدرن بخصوص شبکه‌های موبایل، در ادامه‌ی این فصل، ابتدا در مورد معیارهای موردنیاز برای انجام مسیریابی جریان‌ها و معماری موردنیاز برای جانمایی سرویس‌ها می‌پردازیم و سپس در خصوص معماری کلی و جزئیات چارچوب نرم‌افزاری پیشنهادی بحث می‌کنیم.

۴-۱ مسیریابی جریان‌ها

اغلب الگوریتم‌های مورداستفاده برای مسیریابی در شبکه و اختصاص دادن مسیر به جریان‌هایی که نیازمند کیفیت سرویس خاصی هستند از شبکه‌های سنتی به ارث گرفته‌شده است. این الگوریتم‌ها (مثل OSPF) بر روی پیدا کردن بهترین مسیر مطلق در شبکه بین دونقطه تمرکز می‌کنند و این امر در صورت داشتن چندین جریان بین مبدأ و مقصدهای برابر با کیفیت‌های سرویس متفاوت مشکل‌ساز خواهد شد. برای بهبود شرایط و برطرف کردن مشکل ذکرشده، بهتر است تا الگوریتم‌های مسیریابی، منابع شبکه را تنها به‌اندازه‌ی موردنیاز مورداستفاده قرار دهند و با انتخاب بهترین مسیر مطلق در شبکه برای هر جریان، بیش‌ازحد نیاز نیازمندی‌های جریان را برآورده نکنند که این امر با متمرکز شدن مدیریت شبکه و پیدایش فناوری شبکه‌های تعریف‌شده با نرم‌افزار ممکن شده است. در ادامه به شرح برخی از معیارهایی که می‌بایست در مسیریابی جریان در نظر گرفته شوند می‌پردازیم:

۱. **پهنای باند**[۱]: بسیاری از کاربردهای شبکه که اغلب برنامه‌های مبتنی بر ویدئو (مثل VoD[۲]) هستند، به‌صورت ذاتی نیازمند لینک‌هایی با سرعت‌بالا و حجم بیش‌تری از پهنای باند هستند. یک مسیر بین دونقطه از شبکه معمولاً شامل چندین لینک هست که هرکدام از آن‌ها ممکن است پهنای باند متفاوتی داشته باشد ولی برای محاسبه‌ی پهنای باند کلی یک مسیر می‌توان کم‌ترین پهنای باند مربوط به لینک‌های تشکیل‌دهنده‌ی مسیر را به شرح زیر به عنوان پهنای باند اصلی مسیر در نظر گرفت.

$$BW_{path} = min(BW_{link_1}, BW_{link_2}, ..., BW_{link_n}) \qquad (۱-۴)$$

[۱] Bandwidth
[۲] Video-on-Demand

که در آن $link_i$ ها، لینک‌های تشکیل‌دهنده‌ی مسیر بین مبدأ و مقصد بوده و BW_{link_i} به پهنای باند هریک از آن‌ها اطلاق می‌شود. BW_{path} نیز نشان‌دهنده‌ی پهنای باند کلی مسیر است.

۲. **تأخیر (Delay and Jitter)**: تأخیر برای کاربردهای Real-Time مثل VoIP[۱]، ویدئو کنفرانس[۲] و برنامه‌ی کنترلی اینترنت اشیاء، نقش بسیار مهمی ایفا می‌کند. این دو معیار توزیع‌های آماری بوده که می‌توان آن‌ها را با توزیع نرمال تخمین زد و برای محاسبه کردن تأخیر کلی بین دونقطه از شبکه کافی است تا مقدار میانگین تأخیر تمامی لینک‌های تشکیل‌دهنده‌ی یک مسیر را با یکدیگر جمع کنیم. در صورت نشان دادن تعداد لینک‌ها با N_L، تأخیر کلی مسیر از رابطه‌ی زیر قابل‌محاسبه است:

$$D_P = \sum_{i=1}^{N_L} D_L(i) \qquad (۲-۴)$$

که در آن $D_L(1)$ تا $D_L(N_L)$ تأخیر لینک‌های تشکیل‌دهنده‌ی مسیر بوده و D_P تأخیر کلی مسیر را نشان می‌دهد.

۳. **ضریب اطمینان[۳] و میزان از دست رفتن بسته‌ها[۴]**: مشابه معیارهای پهنای باند و تأخیر که برای آن‌ها کاربردهای متنوعی وجود دارد، خیلی از کاربردها مثل برنامه‌ی کنترلی در صنعت نیازمند ارتباط‌های با ضریب اطمینان بالا هستند. ازآنجایی‌که ممکن است هرکدام از لینک‌های تشکیل‌دهنده‌ی مسیر ضریب اطمینان متفاوتی داشته باشد، ضریب کلی مسیر را می‌توان از ضرب کردن ضرایب تک‌تک لینک‌های تشکیل‌دهنده‌ی مسیر محاسبه نمود. روابط (۴-۳) و (۴-۴) عبارت‌های موردنیاز برای انجام این محاسبات را نشان می‌دهند.

$$P_{RP} = \prod_{i=1}^{N_L} P_{RL}(i) \qquad (۳-۴)$$

$$P_{PL} = 1 - P_{RP}, 0 \leq P_{PL}, P_{RP} \leq 1 \qquad (۴-۴)$$

[۱] Voice over IP
[۲] Video Conferencing
[۳] Reliability
[۴] Packet Loss

که در آن‌ها P_{RP} و $P_{RL}(i)$ ضریب اطمینان کلی مسیر و ضریب اطمینان لینک i ام در مسیر هستند. N_L نیز نشان‌دهنده‌ی تعداد لینک‌های موجود در مسیر بوده و P_{PL} نیز احتمال از دست رفتن بسته‌ها در مسیر می‌باشد.

علاوه بر معیارهای بیان‌شده در بالا، می‌توان میزان توان مصرف‌شده توسط سوئیچ‌های موجود در شبکه را نیز درروند مسیریابی در نظر گرفت [61]. برای این منظور می‌بایست ابتدا میزان مصرف را به‌صورت تابعی از سودمندی سوئیچ‌ها (u_i) مدل کنیم و سپس با استفاده از روابط (۴-۵) و (۴-۶) به ترتیب میزان توان مصرفی لینک‌ها (p_i) و میزان توان مصرفی کل مسیر (P_{path}) محاسبه می‌گردند. در این روابط N_S نیز نشان‌دهنده‌ی تعداد سوئیچ‌های موجود در مسیر است.

$$P_{path} = \sum_{i=1}^{N_S} p_i \qquad (۴-۵)$$

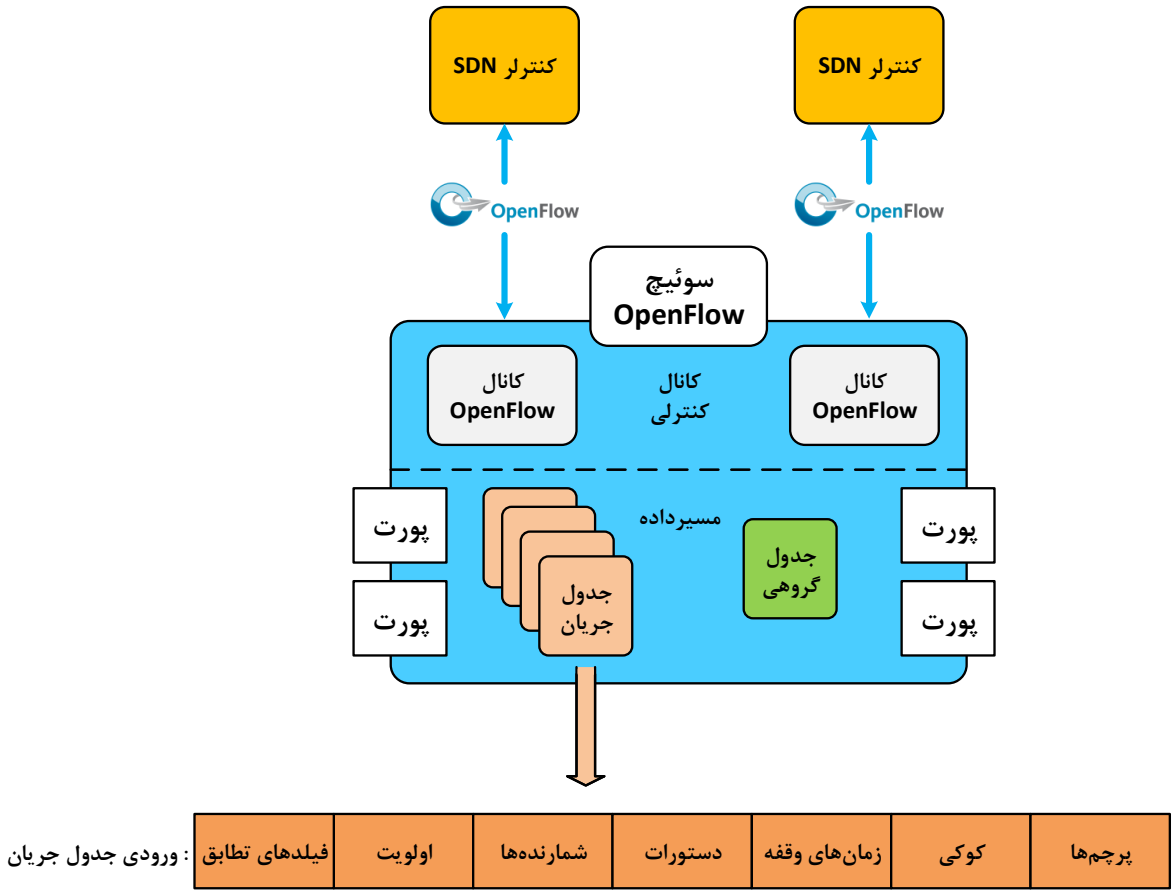
$$p_i = f(u_i) \qquad (۴-۶)$$

برای افزایش ضریب اطمینان مسیریابی جریان‌ها، چارچوب نرم‌افزاری پیشنهادی ما علاوه بر در نظر گرفتن معیار ذکرشده برای هریک از مسیرها در صورت امکان چندین مسیر جایگزین [۱] نیز پیدا می‌کند که درروند انتخاب آن‌ها مسیرهایی را برمی‌گزینند که با مسیر اصلی لینک‌های مشترک کمتری داشته باشند. این مسیرها منجر به افزایش ضریب برگشت‌پذیری [۲] شده و ازاین‌رو به آن‌ها، مسیرهای Disjointed Paths گفته می‌شود [62, 63].

معماری سوئیچ‌های پشتیبانی‌کننده از پروتکل OpenFlow امکانات جدیدی ایجاد می‌کنند که با استفاده از آن‌ها می‌توان چندین مسیر را در پروسه‌ی مسیریابی استفاده نمود. همان‌طور که در شکل (۴-۱) نشان داده‌شده است، یک سوئیچ پشتیبانی‌کننده از پروتکل OpenFlow [19] از دو بخش اصلی تشکیل می‌شود: کانال کنترلی [۳] که با استفاده از پروتکل OpenFlow با کنترلرهای SDN ارتباط برقرار می‌کند و بخش مسیر داده [۴] که خود از پورت‌های سوئیچ، جدول گروهی [۵] و چندین جدول جریان [۶] تشکیل‌شده که وظیفه‌ی انتقال بسته‌ها را بر عهده دارد. هرکدام از جدول‌های جریان شامل اطلاعات مربوط به جریان‌های

[۱] Redundant Paths
[۲] Resilience
[۳] Control Channel
[۴] Datapath
[۵] Group Table
[۶] Flow Table

مختلف هستند و هرکدام از سطرهای موجود در جدول خود از هفت بخش: فیلدهای تطابق[۷]، اولویت[۸]، شمارنده‌ها، دستورات، زمان‌های وقفه[۹]، کوکی[۱۰] و پرچم‌ها تشکیل‌شده است. برای مسیریابی جریان‌ها، مشخصات جریان در بخش فیلدهای تطابق و عمل موردنیاز برای جریان در بخش دستورات قرار می‌گیرند؛ بنابراین پس از پیدا شدن مسیر اصلی و مسیرهای جایگزین برای هر جریان توسط چارچوب نرم‌افزاری پیشنهادی ما، می‌توان این مسیرها را با مقادیر اولویت متفاوت بر روی جدول جریان سوئیچ‌ها نصب نمود. با این کار، درصورتی‌که یکی از لینک‌ها از دست برود و دیگر معتبر نباشد، کافی است تا قانون مربوط به آن را از سوئیچ‌ها حذف نماییم و بعد از آن سوئیچ‌ها با استفاده از مسیرهای جایگزینی که اولویت کمتری نسبت به مسیر اصلی دارند، بسته‌ها را در شبکه مسیریابی می‌کنند.



شکل ۴-۱: معماری سوئیچ OpenFlow

[۷] Match Fields
[۸] Priority
[۹] Timeouts
[۱۰] Cookie

۴-۲ جانمایی سرویس‌ها

در شبکه‌های مدرن، هر بار که جریانی وارد شبکه می‌شود می‌بایست بر روی آن سرویس‌ها و توابع شبکه‌ی مختلفی (مثل دیوارآتش، IDS، تبدیل آدرس IP نسخه‌ی ۴ به آدرس IP نسخه‌ی ۶[۱] و برعکس) اجرا شود و پیدایش NFV این اجازه را به ما می‌دهد تا بجای استفاده از تجهیزات سخت‌افزاری میانی، این سرویس‌ها و توابع را با استفاده از نرم‌افزارهای پیاده‌سازی شده بر روی سرورها اجرا نماییم. برای اینکه بتوانیم از ویژگی‌های بسیار مفید NFV در شبکه استفاده کنیم، ابتدا می‌بایست معماری شبکه را مطابق با نیازمندی‌های NFV از قبیل فراهم‌سازی[۲]، مدیریت و تنظیم[۳] توابع مجازی شبکه[۴] تغییر دهیم. یکی از معماری‌های پیشنهادشده برای استفاده از NFV که قادر به پاسخگویی به نیازهای عنوان‌شده در بالا می‌باشد، معماری MANO[۵] بوده که توسط اتحادیه‌ی مخابرات اروپا[۶] معرفی‌شده است. این معماری مطابق شکل (۴-۲) از سه بلوک اصلی NFVI[۷]، VNF و MANO تشکیل‌شده است. بلوک‌های NFVI و VNF، بلوک‌های اصلی تأمین‌کننده‌ی نیازهای NFV هستند که توابع مختلف شبکه با استفاده از آن‌ها اجرا می‌شوند و بلوک MANO مسئول فراهم‌سازی، تنظیم و مدیریت این دو بلوک خواهد بود [64]. بلوک MANO خود نیز از سه زیر بلوک به شرح زیر تشکیل‌شده است:

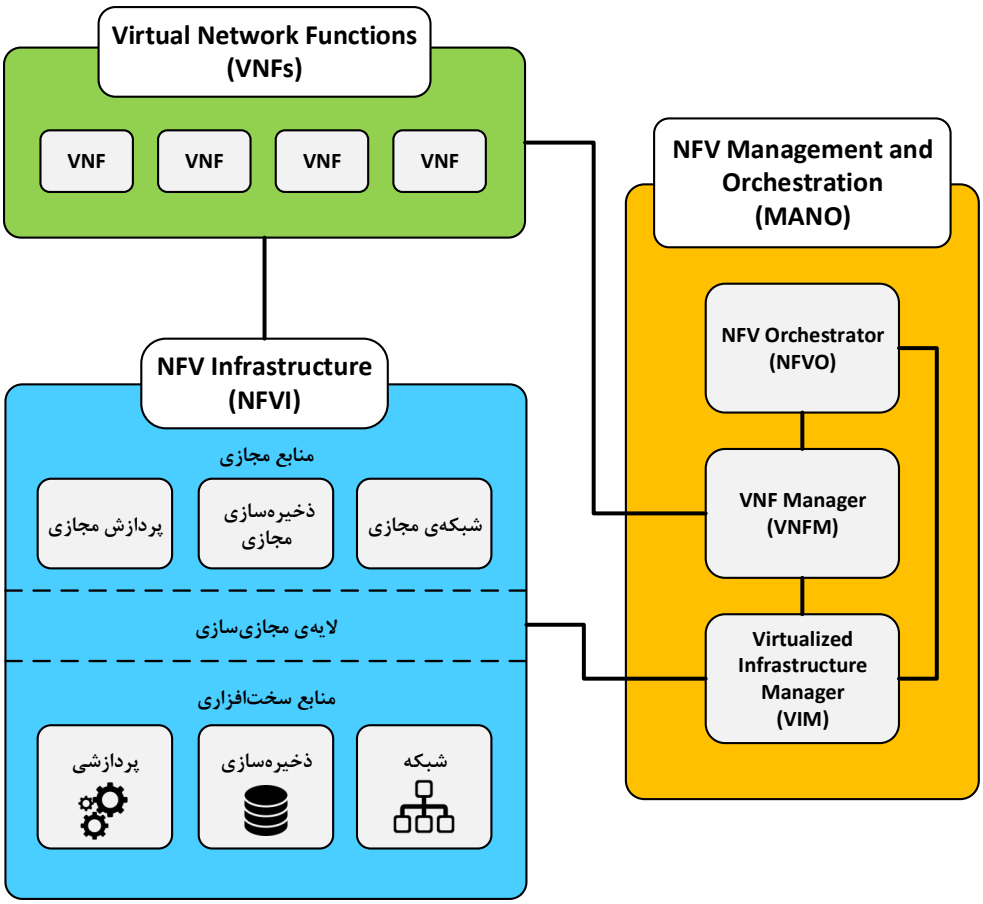
۱. **زیر بلوک VIM**[۱]: این زیر بلوک منابع فیزیکی و مجازی موجود در بلوک NFVI را مدیریت و کنترل می‌کند.

۲. **زیر بلوک VNFM**[۲]: این زیر بلوک طول عمر[۳] توابع مجازی شده را مدیریت می‌کند.

۳. **زیر بلوک NFVO**[۴]: این زیر بلوک این امکان را در شبکه ایجاد می‌کند تا با استفاده از آن بتوان با ترکیب سرویس‌ها و توابع مختلف زنجیره‌های پیچیده‌تر از توابع و سرویس‌ها را در شبکه اجرا نمود.

ما در چارچوب نرم‌افزاری پیشنهادی‌مان فرض کرده‌ایم که شبکه‌های موبایل می‌توانند با پیروی از ملزومات ارائه‌شده توسط معماری ETSI MANO پیاده‌سازی شوند و هریک از روترهای موجود در شبکه به یک ابرک[۵] (مرکز داده کوچک) متصل بوده که می‌تواند تمامی توابع شبکه را اجرا نماید؛ بنابراین، چارچوب

[۱] IPv4 to IPv6 NAT
[۲] Provision
[۳] Configuration
[۴] Virtual Network Functions (VNF)
[۵] Management and Orchestration
[۶] European Telecommunications Standards Institute (ETSI)
[۷] NFV Infrastructure
[۱] Virtual Infrastructure Manager
[۲] VNF Manager
[۳] Life Cycle
[۴] NFV Orchestrator
[۵] Cloudlet



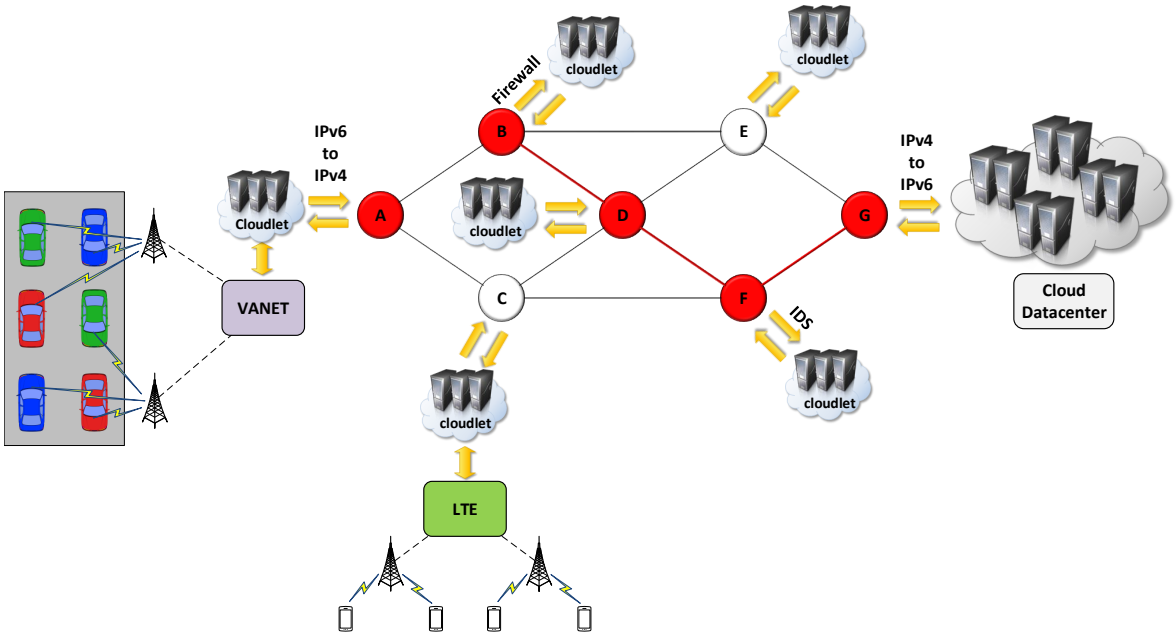
شکل ۴-۲: معماری ETSI MANO

نرم‌افزاری ما پس از پیدا کردن مسیر مناسب برای جریان‌های واردشده به شبکه، وظیفه‌ی اجرای توابع و سرویس‌های موردنیاز آن جریان را به‌صورت عادلانه بین ابرک‌های متصل به نودهای اختصاص داده‌شده به مسیر جریان تقسیم می‌کند. با این کار می‌توان از منابع موجود در تمامی ابرک‌ها به‌درستی استفاده کرد و بار کلی شبکه را متوازن نمود. پس از تقسیم شدن توابع شبکه بین ابرک‌ها، بلوک VIM شروع به تخصیص منابع فیزیکی و مجازی موردنیاز برای اجرای توابع اختصاص داده‌شده به ابرک‌ها می‌کنند سپس بلوک VNFM با ارسال Image های مناسب سرویس به ابرک‌ها، ماشین‌های مجازی اختصاص داده‌شده در ابرک‌ها را برای اجرای توابع آماده‌سازی می‌کنند [15]. در ضمن، علاوه بر آماده‌سازی‌های موردنیاز برای NFV، کنترلرهای SDN موجود در شبکه نیز باید قوانین مناسب را برای رسیدن بسته‌های جریان‌ها به ابرک‌های موردنظر، بر روی روترهای شبکه نصب نمایند تا وقتی که بسته‌ای از جریان به یک نود برسد، در صورت نیاز روتر مربوطه آن را برای اجرای سرویس موردنظر به ابرک مربوطه ارسال کند.

شکل (۴-۳) یکی موقعیت فرضی از یک شبکه‌ی بزرگ را نشان می‌دهد که در آن جریانی می‌خواهد از یک شبکه‌ی VANET[1] [65] متصل به نود A به سمت ابری در نود G مسیریابی شود. این جریان نیازمند اجرای چهار تابع: Firewall، IDS، IPv6 to IPv4 NAT و IPv4 to IPv6 NAT است که

[1] Vehicular Ad-Hoc Network

همان‌طور که در شکل نشان داده‌شده به‌جای استفاده از تجهیزات اضافی و یا اجرای تمامی آن‌ها در یک نود، اجرای آن‌ها بین ابرک‌های مختلف موجود در مسیر تقسیم‌شده است. در ضمن، مطابق شکل، هریک از ابرک‌ها می‌توانند به شبکه‌های مختلفی نظیر شبکه‌ی LTE و 4G متصل باشند.



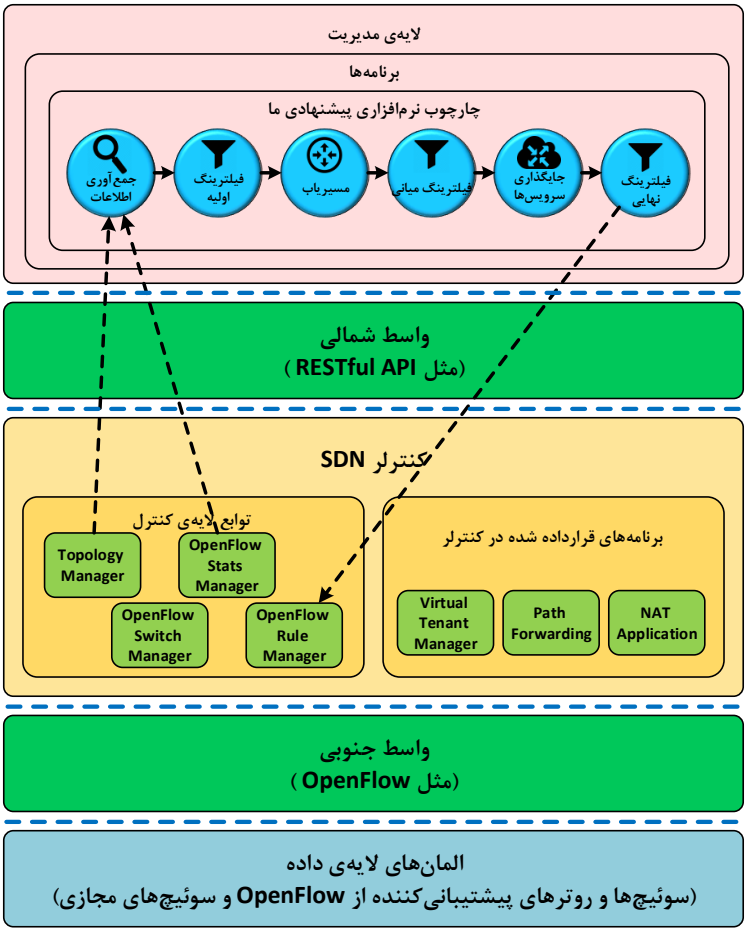
شکل ۴-۳: نمونه‌ای از جانمایی سرویس‌ها در شبکه

۴-۳ معماری کلی چارچوب نرم‌افزاری پیشنهادی

به دلیل بزرگ شدن اندازه شبکه‌ها و رشد روزافزون نرخ ترافیک در آن‌ها بخصوص در شبکه‌های موبایل، پیدا کردن مسیر مناسب و جانمایی سرویس‌ها برای جریان‌های شبکه تبدیل به مسائل بهینه‌سازی NP-Hard شده‌اند که به‌راحتی نمی‌توان آن‌ها را توسط روش‌ها جبری مثل مدل‌های فرم بسته‌ی ریاضیاتی[۱] و یا الگوریتم‌های حریصانه[۲] حل نمود. به همین منظور، ما در چارچوب نرم‌افزاری پیشنهادی خود برای حل مسئله از الگوریتم‌های فرا ابتکاری استفاده کرده‌ایم که در فصل گذشته در خصوص آن‌ها به‌صورت مفصل توضیح داده شد. برای حل مسئله‌ی مسیریابی جریان‌ها و جانمایی سرویس‌ها، ما از ترکیب نسخه‌ی دانش‌محور و بهبودیافته‌ی الگوریتم سیستم کلونی مورچگان[۳] (ACS) و الگوریتم گرگ‌های خاکستری آشوبناک استفاده می‌کنیم. در فصل گذشته، جزئیات الگوریتم گرگ‌های خاکستری آشوبناک به‌طور کامل

[۱] Closed-form mathematical models
[۲] Greedy Algorithms
[۳] Ant Colony System (ACS)

توضیح داده شد و در بخش‌های بعدی این فصل جزئیات الگوریتم ACS و نحوه‌ی استفاده از آن‌ها برای حل مسئله توضیح داده می‌شود.



شکل ۴-۴: چارچوب نرم‌افزاری ارائه‌شده برای مسیریابی جریان‌ها

چارچوب نرم‌افزاری پیشنهادی ما، مسیر بهینه و جانمایی سرویس‌ها برای جریان ورودی را در شش مرحله انجام می‌دهد. شکل (۴-۴) یک معماری نمونه از شبکه‌های تعریف‌شده با نرم‌افزار را نشان می‌دهد که در آن چارچوب نرم‌افزاری ما برای حل مسئله‌ی مسیریابی در صفحه‌ی مدیریت قرارگرفته است و می‌تواند با تعامل با واحدهای موجود در کنترلرهای شبکه، اطلاعات موردنیاز از عملکرد لینک‌ها و سوئیچ‌های شبکه را از آن‌ها بگیرد و پس از پیدا کردن جواب مناسب، می‌تواند با استفاده از واحدهای کنترلر قوانین مناسب را بر روی سوئیچ‌ها و روترها نصب نماید. اغلب کنترلرهای SDN همانند کنترلرهای OpenDayLight [66]، Floodlight [67] و ONOS [68] از دو بخش اصلی تشکیل‌شده‌اند که یکی از آن‌ها مسئول انجام توابع اصلی صفحه‌ی کنترلر و برقراری ارتباط با صفحه‌های داده و مدیریت بوده و بخش دوم شامل برنامه‌ی قرار داده‌شده[1] در کنترلر می‌شود که این برنامه‌ها می‌توانند در صورت غیاب

[1] Embedded

سرورهای خارجی برای انجام توابع شبکه، این توابع را اجرا نمایند. مراحل مختلف چارچوب نرم‌افزاری پیشنهادی ما که در شکل (۴-۴) به‌صورت بلوک‌های مختلف نشان داده‌شده‌اند، به شرح زیر می‌باشند:

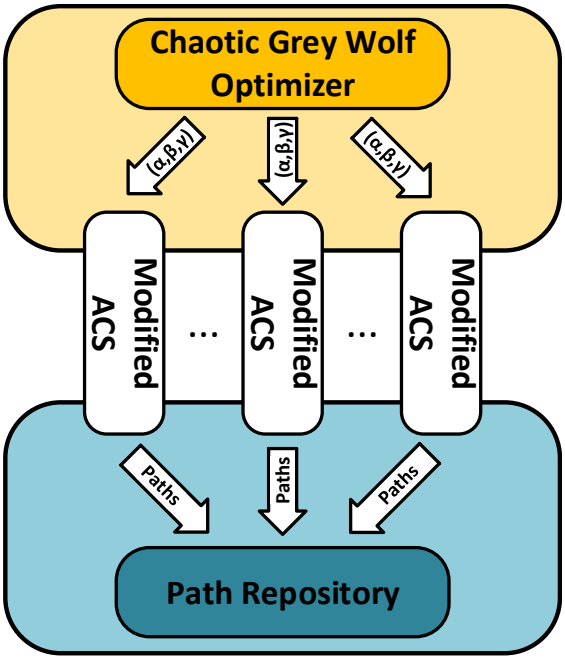
۱. **بلوک جمع‌آوری اطلاعات**: جدا شدن صفحه‌های کنترل و داده از یکدیگر توسط SDN این امکان را ایجاد می‌کند که یک دانش کلی و متمرکز از شرایط و توپولوژی شبکه داشته باشیم. در این مرحله از چارچوب نرم‌افزاری، برنامه‌ی شبکه‌ی ما آخرین وضعیت شبکه را از کنترلرهای SDN دریافت می‌کند. این اطلاعات توسط پروتکل‌های مشخصی [47] مثل OpenFlow، LLDP[۱] و CDP[۲] توسط کنترلرها جمع‌آوری می‌شوند و به دو بخش کلی: توپولوژی شبکه و عملکرد شبکه تقسیم می‌شوند. توپولوژی شبکه را می‌توان از واحدی بانام «مدیریت توپولوژی[۳]» در کنترلر دریافت کرده و اطلاعات مربوط به عملکرد شبکه که می‌تواند شامل پهنای باند، تأخیر، ضریب اطمینان و احتمال از دست رفتن بسته‌ها در شبکه و یا حتی سودمندی و مصرف توان سوئیچ‌ها در شبکه باشد را می‌توان از واحدهای «بررسی عملکرد[۴]» و «مدیریت آمار OpenFlow[۵]» در کنترلرها دریافت نمود.

۲. **بلوک فیلترینگ اولیه**: به خاطر وجود مرحله‌ی قبل، چارچوب نرم‌افزاری پیشنهادی ما در این مرحله تمامی اطلاعات موردنیاز برای انجام مسیریابی جریان‌ها را دارد و با داشتن آن‌ها می‌تواند ماتریس‌های عملکرد نظیر ماتریس تأخیر (D)، ماتریس پهنای باند (BW)، ماتریس احتمال از دست‌رفتن بسته‌ها (PL) و ماتریس ضریب اطمینان (R) را تشکیل دهد. این ماتریس‌ها، ماتریس‌های $N \times N$ بوده و شامل اطلاعات شبکه‌ای بین نودها در یک شبکه‌ای با تعداد N نود هستند. به‌عنوان‌مثال، $d_{i,j}$ یکی از المان‌های ماتریس تأخیر بوده است که تأخیر بین دو نود i و j را نشان می‌دهد. برای کوچک‌تر کردن فضای جستجو و افزایش سرعت پیدا کردن مسیر بهینه، در این مرحله، چارچوب نرم‌افزاری ما ماتریس تأخیر را بر اساس نیازمندی‌های جریان تغییر می‌دهد. مثلاً درصورتی‌که جریان ورودی به ۱۰۰ مگابیت در ثانیه پهنای باند نیاز داشته باشد، چارچوب نرم‌افزاری ما تمامی المان‌هایی از ماتریس تأخیر که مقدار آن‌ها در ماتریس پهنای باند کمتر از ۱۰۰ است را به بی‌نهایت تبدیل می‌کند. با این کار، فضای جواب برای جستجو در مراحل بعدی کوچک می‌شود و می‌تواند سرعت جستجو را تا حد بسیار زیادی متناسب با اندازه‌ی شبکه افزایش دهد. خروجی این مرحله ماتریس تغییریافته‌ی تأخیر است که در مرحله بعدی از آن بجای ماتریس تأخیر اولیه برای مسیریابی استفاده می‌شود.

۳. **بلوک مسیریاب**: در این مرحله، چارچوب نرم‌افزاری ما با استفاده از الگوریتم ACS به دنبال

[۱] Link Layer Discovery Protocol
[۲] Cisco Discovery Protocol
[۳] Topology Manager
[۴] Performance Monitoring
[۵] OpenFlow Stats Manager

کوتاه‌ترین مسیر موجود بین مبدأ و مقصد جریان می‌شود و در این حین، تمامی مسیرهای ممکن و قابل‌دسترس[۶] موجود بین این دونقطه را در فضایی با عنوان «انبار مسیر[۷]» ذخیره می‌کند. همان‌طور که در شکل (۴-۵) نیز نشان داده‌شده، چارچوب نرم‌افزاری ما علاوه بر ACS از الگوریتم گرگ‌های آشوبناک نیز در پروسه‌ی مسیریابی استفاده می‌کند و از آن برای تغییر پارامترهای الگوریتم ACS استفاده می‌شود که منجر به افزایش رفتار اکتشافی در پروسه‌ی مسیریابی و پرتر شدن انبار مسیرها می‌شود. وجود انبار مسیر به ما اجازه می‌دهد تا از بین مسیرهای پیداشده مناسب‌ترین آن‌ها و نه لزوماً کوتاه‌ترین آن‌ها را برای جریان ورودی در شبکه برگزینیم. علاوه بر این، از وجود چنین انباری می‌توان در بهبود عملکرد جستجوی الگوریتم ACS نیز استفاده نمود که در بخش‌های بعد به تفضیل در این خصوص توضیح داده خواهد شد.



شکل ۴-۵: بلوک مسیریاب

۴. **بلوک فیلترینگ میانی**: در این مرحله، مقادیر پهنای باند کلی، تأخیر کلی، ضریب اطمینان کلی و احتمال از دست رفتن بسته را برای مسیرهای موجود در انبار مسیرها محاسبه می‌کنیم و در ادامه آن مسیریابی که نیازمندی‌های موردنیاز جریان ورودی را برآورده نمی‌کنند از انبار مسیرها حذف می‌کنیم. در این پروسه، همچنین می‌توان معیارهای نظیر مصرف انرژی و سودمندی سوئیچ‌ها را نیز در نظر گرفت.

[۶] Feasible
[۷] Path Repository

۵. **بلوک جانمایی سرویس‌ها**: خروجی مرحله‌ی قبل به ما کلکسیونی از تمامی مسیرهای قابل‌قبول برای جریان ورودی را می‌دهد که از آن‌ها در این مرحله برای جانمایی سرویس‌ها استفاده می‌کنیم. در این مرحله، به دلیل پیچیدگی این مسئله و NP-Hard بودن آن، مجدداً از الگوریتم گرگ‌های خاکستری آشوبناک برای توزیع توابع شبکه بین ابرک‌های متصل به نودها استفاده می‌کنیم و در انجام آن به برقراری برابری و توازن بار بین ابرک‌ها و تخصیص بهینه‌ی منابع آن‌ها (مثل CPU و حافظه) توجه می‌کنیم. بدین منظور، مجدداً مشابه فصل گذشته از شاخص برابری جین استفاده می‌کنیم که می‌توان آن را با استفاده از رابطه‌ی زیر محاسبه نمود. در این رابطه، L_i نشان‌دهنده‌ی بار ابرک i ام بوده که می‌توان آن را به‌صورت میانگین وزنی از میزان مصرف منابع مختلف محاسبه نمود.

$$F = \frac{(\sum_{i=1}^{N} L_i)^2}{N \times \sum_{i=1}^{N} L_i^2} \qquad (۷-۴)$$

$$L_i = w_1 \times L_{CPU}^i + w_2 \times L_{Memory}^i \qquad (۸-۴)$$

L_{CPU}^i و L_{Memory}^i بارهای پردازنده و حافظه در ابرک i ام بوده و w_1 و w_2 وزن‌های آن‌ها در محاسبه‌ی بار کلی ابرک هستند که می‌توانند متناسب با نیازمندهای تأمین‌کنندگان ابر[۱] انتخاب شوند.

۶. **بلوک فیلترینگ نهایی و انتخاب مسیر**: در آخرین مرحله از چارچوب نرم‌افزاری، از بین مسیرهای موجود در انبار مسیر، ابتدا آن مسیرهایی را شاخص برابری قابل قبولی برای جانمایی سرویس‌ها ندارند را حذف کرده و در ادامه از بین مسیرهای باقی‌مانده مسیری که معیار موردنظر ما را بیشتر برآورده می‌کند به‌عنوان مسیر اصلی انتخاب می‌کنیم. برای انتخاب مسیرهای جایگزین نیز، ابتدا می‌بایست ضریب همبستگی[۲] بین مسیر اصلی و بقیه‌ی مسیرهای موجود در انبار مسیر را محاسبه کنیم. به‌عنوان‌مثال، درصورتی‌که دو مسیر به‌صورت:

$$P_1 :< A, B, E, G >$$

$$P_2 :< A, C, D, E, G >$$

[۱] Cloud Providers
[۲] Correlation Factor

از نود A به G داشته باشیم، ضریب همبستگی این دو مسیر ۳ بوده که برابر با تعداد اعضای مجموعه‌ی زیر است.

$$M = \{x | x \in P_1 \; and \; x \in P_2\} :< A, E, G > \qquad (۹-۴)$$

پس از محاسبه‌ی ضریب همبستگی برای تمامی مسیرها، ازآنجایی‌که قصد داریم تا مسیرهای Disjoint را به‌عنوان مسیرهای جایگزین انتخاب کنیم، مسیرهای موجود در انبار را به‌صورت نزولی و بر اساس ضرایب همبستگی محاسبه‌شده مرتب می‌کنیم و سپس دو مسیری که کمترین ضریب همبستگی رادارند به‌عنوان مسیرهای جایگزین انتخاب می‌کنیم.

۴-۴ الگوریتم دانش‌محور سیستم کلونی مورچگان

مورچه‌های واقعی در طبیعت برای ارتباط با یکدیگر و پیدا کردن کوتاه‌ترین مسیر بین لانه و منابع غذایی، یک ماده‌ی شیمیایی بانام فرومون[۱] در مسیر حرکتشان ترشح می‌کنند. به دنبال مشاهدات انجام‌شده بر روی رفتار مورچه‌ها، مارکو دوریگو[۲] مدل الگوریتمی برای این رفتار آذوقه جویی مورچه‌ها ارائه داد و نام آن را الگوریتم بهینه‌سازی کلونی مورچگان[۳] گذاشت [69]. این الگوریتم می‌تواند برای حل مسائل بهینه‌سازی ترکیبی و پیدا کردن زیرمجموعه‌هایی از گراف‌ها استفاده شود. از زمان ارائه‌ی این الگوریتم، تحقیقات بسیار زیادی در خصوص افزایش دقت و همگرای الگوریتم ACO انجام‌گرفته است که یکی از معروف‌ترین آن‌ها، الگوریتم سیستم کلونی مورچگان[۴] [70, 71] است که برای برقراری تعادل بین اکتشاف و استثمار در الگوریتم‌های مبتنی بر مورچه ارائه‌شده است. برای حل مسائل مسیریابی با استفاده از ACS، ابتدا می‌بایست تعدادی (n) مورچه در نود مبدأ از شبکه (گراف) قرارداد و سپس با استفاده از روابط (۴-۱۰) و (۴-۱۱)، هر مورچه می‌تواند نود بعدی خود (نود j) را بر اساس سه پارامتر انتخاب کند و این پروسه تا زمانی که مورچه به نود مقصد برسد ادامه خواهد داشت.

$$j = \begin{cases} argmax_{u \in N_i}(\tau_{i,u}^{\alpha}(t)\eta_{i,u}^{\beta}(t)\theta_{i,u}^{\gamma}(t)) & \text{اگر } r_0 \leq r \\ J & \text{اگر } r_0 > r \end{cases} \qquad (۱۰-۴)$$

[۱] Pheromone
[۲] Marco Dorigo
[۳] Ant Colony Optimization (ACO)
[۴] Ant Colony System (ACS)

$$P_{i,J}(t) = \frac{\tau_{i,J}^{\alpha}(t)\eta_{i,J}^{\beta}(t)\theta_{i,J}^{\gamma}(t)}{\sum_{u \in N_i} \tau_{i,u}^{\alpha}(t)\eta_{i,u}^{\beta}(t)\theta_{i,u}^{\gamma}(t)} \quad (۱۱-۴)$$

پارامترهای استفاده‌شده برای انتخاب نودها به شرح زیر هستند:

۱. پارامتر فرومون ($\tau_{i,j}$): نشان‌دهنده‌ی مقدار فرومون‌های ریخته شده بر روی لینک متصل‌کننده‌ی نود i و نود j

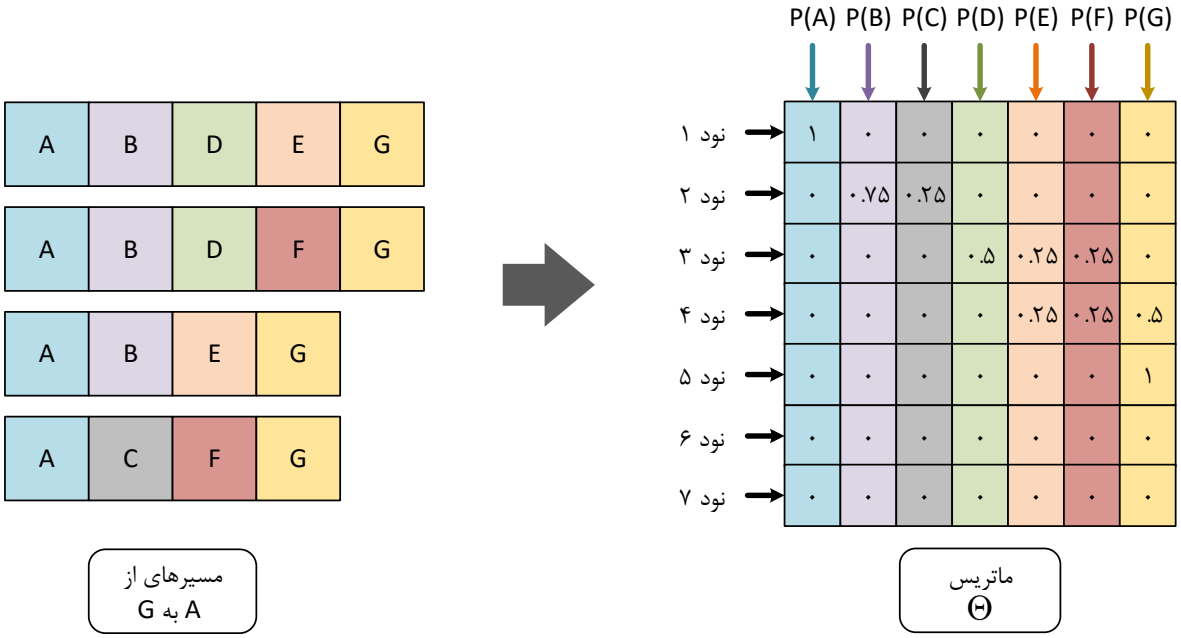
۲. پارامتر مکاشفه‌ای[۱] ($\eta_{i,j}$): برابر با عکس تأخیر یا فاصله‌ی بین دو نود i و j

۳. پارامتر انبار[۲] ($\theta_{i,j}$): ازآنجایی‌که برای بهتر شدن عملکرد مسیریابی، انباری برای ذخیره‌ی مسیرهای مختلف در سیستم در نظر گرفته‌شده است، می‌توان از آن برای هدایت الگوریتم ACS نیز استفاده نمود و بدین منظور، پارامتر ($\theta_{i,j}$) که نشان‌دهنده‌ی احتمال انتخاب نود j به‌عنوان i امین نود در مسیر است را به الگوریتم ACS اضافه کرده‌ایم. شکل (۴-۶) یک انبار مسیر نمونه را که شامل ۴ مسیر مختلف از نود A به نود G در توپولوژی شکل (۴-۳) را نشان می‌دهد. در این انبار فرضی، سه مسیر نود B و یک مسیر نود C را به‌عنوان دومین نود موجود در مسیر انتخاب کرده‌اند که به همین دلیل پارامتر ($\theta_{2,3}$) برابر با ۰.۲۵ خواهد شد به این معنی که احتمال انتخاب نود C به‌عنوان دومین نود در مسیر بین A و G برابر با ۲۵ درصد است. در شکل (۴-۶)، علاوه بر این احتمال، ماتریس Θ که حاوی تمامی پارامترهای انبار است نیز نشان داده‌شده است.

در رابطه‌ی (۴-۱۰)، r یک عدد تصادفی بین ۰ و ۱ بوده و r_0 یک حدنصاب[۳] تعریف‌شده توسط کاربر برای برقراری تعادل بین اکتشاف و استثمار در الگوریتم است. N_i مجموعه نودهای متصل به نود i بوده و J یکی از اعضای این مجموعه است که متناسب با احتمال انتخاب نودهای مختلف با استفاده از رابطه‌ی (۴-۱۱) انتخاب می‌شود. زمانی که $r_0 > r$ باشد، رابطه‌ی (۴-۱۰) نودی با بیشترین احتمال را برمی‌گزیند که منجر به رفتار استثماری شده و در زمان‌های دیگر، یکی از نودها را به‌صورت تصادفی انتخاب می‌کند که رفتار اکتشافی را به دنبال خواهد داشت.

ما k امین مسیر در t امین مرحله از الگوریتم را با $x_k(t)$ نشان داده و بهترین مسیر پیداشده تا اینجا را با $\hat{x}(t)$ نشان می‌دهیم. f نیز تابع موردنیاز برای محاسبه‌ی هزینه‌ی مورچه‌ها بوده که در اینجا برابر با مجموع تأخیر لینک‌های سازنده‌ی مسیر است. بعد از انتخاب مسیر برای هریک از مورچه‌ها و

[۱] Heuristic
[۲] Repository Factor
[۳] Threshold



شکل ۴-۶: پارامتر انبار

بعد از پایان هر مرحله از الگوریتم، می‌بایست فرومون‌های لینک‌های مختلف را بر اساس مسیر مورچه‌ها به‌روزرسانی نماییم. در حالت اول که به آن به‌روزرسانی محلی[۴] نیز گفته می‌شود، تمامی مورچه‌ها بعد از پیدا کردن مسیر با استفاده از رابطه‌ی (۴-۱۴) مقادیر فرومون‌ها را تغییر می‌دهند و در حالت دوم که بانام به‌روزرسانی کلی[۵] نیز شناخته می‌شود، در پایان هر مرحله از الگوریتم که بهترین مورچه با کوتاه‌ترین مسیر پیدا می‌شود، فرومون لینک‌هایی که بهترین مورچه از آن عبور می‌کند با استفاده روابط (۴-۱۲) و (۴-۱۳) به‌روزرسانی می‌شود. بخش اول روابط (۴-۱۲) و (۴-۱۴) بانام تبخیر فرومون[۶] شناخته می‌شود که منجر به کاهش تأثیر فرومون مراحل قبلی شده و به پروسه‌ی گیر نیفتادن در نقطه‌ی بهینه‌ی محلی کمک می‌کند. ρ_1 و ρ_2 ثابت‌هایی در بازه‌ی $(0,1)$ هستند و τ_0 یک ثابت مثبت و کوچک بوده که می‌تواند برابر با $\frac{1}{N \times f(x_k(t))}$ قرار داده شود [71] که در آن N تعداد نودها در مسئله و $f(x_k(t))$ هزینه‌ی مسیر پیموده شده توسط مورچه k ام در t امین مرحله از الگوریتم را نشان می‌دهند.

$$\tau_{i,j}(t+1) = (1-\rho_1)\tau_{i,j}(t) + \rho_1 \Delta\tau_{i,j}(t) \qquad (۴-۱۲)$$

[۴] Local Pheromone Update
[۵] Global Pheromone Update
[۶] Pheromone Vaporization

$$\Delta\tau_{i,j}(t) = \begin{cases} \frac{1}{f(\hat{x}(t))} & \text{اگر } (i,j) \in \hat{x}(t) \\ 0 & \text{اگر } (i,j) \notin \hat{x}(t) \end{cases} \quad (۱۳-۴)$$

$$\tau_{i,j}(t+1) = (1-\rho_2)\tau_{i,j}(t) + \rho_2\tau_0 \quad (۱۴-۴)$$

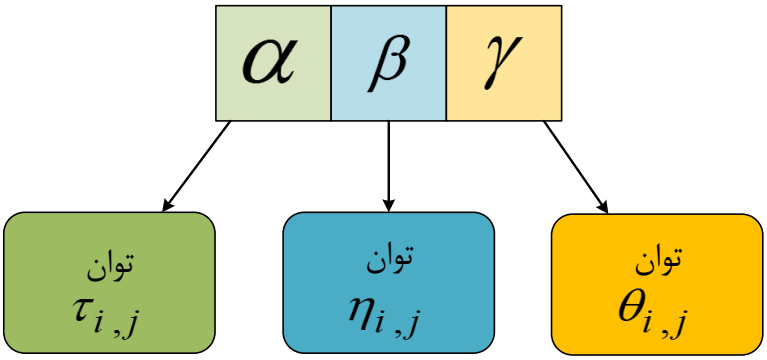
علاوه بر انجام به‌روزرسانی فرومون‌ها، چارچوب نرم‌افزاری ما در پایان هر مرحله از الگوریتم مسیرهای جدید پیداشده را به انبار مسیر می‌افزاید و سپس ماتریس دربردارنده‌ی پارامترهای انبار (Θ) را مجدداً محاسبه می‌کند. در پروسه‌ی محاسبه این ماتریس، برای جلوگیری از گیرکردن در نقطه‌ی بهینه‌ی محلی، المان‌های صفر در ماتریس را با مقدار ۰.۰۱ جایگزین می‌کنیم. شبه کد الگوریتم بهبودیافته و دانش‌محور ACS در الگوریتم (۲) در فصل پیوست آورده شده است.

۴-۵ الگوریتم گرگ‌های خاکستری آشوبناک

الگوریتم گرگ‌های خاکستری یکی از الگوریتم‌های مبتنی بر PSO بوده که از سلسله‌مراتب اجتماعی و نحوه‌ی شکار کردن گرگ‌های خاکستری در طبیعت الهام گرفته‌شده است. ما در چارچوب نرم‌افزاری خود برای بهبود سرعت همگرایی و جلوگیری از گیرکردن الگوریتم در نقاط بهینه‌ی محلی ازسری‌های آشوبناک برای بهبود رفتار این الگوریتم استفاده کرده‌ایم که در فصل گذشته در این خصوص نیز به‌تفصیل توضیحات لازم داده شد. ما در دو مرحله از چارچوب نرم‌افزاری مربوط به مسیریابی جریان‌ها و جانمایی سرویس‌ها نیز از این الگوریتم بهره برده‌ایم که به شرح زیر است:

۱. **مسیریابی**: برای افزایش رفتار اکتشافی در بلوک مسیریاب چارچوب نرم‌افزاری ارائه‌شده، ما از یک‌لایه‌ی GWO بر روی الگوریتم ACS استفاده می‌کنیم. برای داشتن این لایه، می‌بایست کارگزاران آن را مشابه شکل (۴-۷) تعریف نموده که هریک از آن‌ها یک سه‌تایی شامل (α, β, γ) که توان‌های پارامتر فرومون، پارامتر مکاشفه‌ای و پارامتر انبار برای الگوریتم ACS هستند را تولید می‌کند. ما برای این کار فرض کرده‌ایم که این مقادیر می‌توانند از بازه‌ی $(1, 5)$ انتخاب شوند.

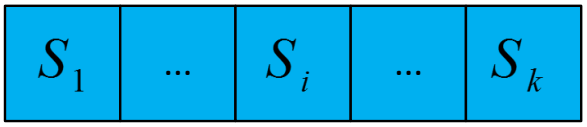
تابع هزینه برای این لایه از GWO برابر با هزینه‌ی بهترین مورچه‌ی پیداشده توسط لایه‌ی ACS خواهد بود. به‌عبارت‌دیگر، برای محاسبه‌ی هزینه‌ی هر یک از کارگزاران، یک الگوریتم ACS برحسب



شکل ۴-۷: کاگزاران مسیریاب

پارامترهای محاسبه‌شده توسط الگوریتم GWO اجرا می‌شود و هزینه‌ی بهترین مورچه‌ی پیداشده با این پارامترها به‌عنوان هزینه‌ی کارگزار در نظر گرفته می‌شود و در ادامه الگوریتم GWO سعی می‌کند تا با تغییر موقعیت کارگزاران خود بتواند مسیرهایی با هزینه‌ی بهترین پیدا کند و دراین‌بین موجب پر شدن انبار مسیرها نیز می‌شود. شبه کد مربوط به این الگوریتم نیز الگوریتم (۳) در فصل پیوست آورده شده است.

۲. **جانمایی سرویس**: پس از جمع‌آوری مسیرهای مناسب از انبار مسیرها و حذف مسیرهای غیرقابل‌قبول در سومین مرحله از چارچوب نرم‌افزاری، می‌بایست سرویس‌های موردنیاز جریان ورودی را بر روی نودهای تشکیل‌دهنده‌ی مسیرها پخش نماییم. برای این کار مجدداً از الگوریتم GWO استفاده کرده‌ایم و برای این منظور کارگزاران آن را مشابه شکل (۴-۸) تعریف کرده‌ایم که در آن هر المان از کارگزار الگوریتم شماره‌ی نودی که i امین سرویس باید برای روی آن اجرا شود را نشان می‌دهد. در اینجا فرض کرده‌ایم که بار تقریبی که هریک از سرویس‌های موردنیاز جریان در ابرک‌ها اشغال می‌کنند، پیش‌تر محاسبه‌شده و الگوریتم از آن مقادیر مطلع است و با این فرض با استفاده از شاخص برابری جین به محاسبه‌ی هزینه‌ی کارگزارها در الگوریتم می‌پردازد. علاوه بر این، درروند جانمایی سرویس‌ها فرض کرده‌ایم که اگر بار هریک از ابرک‌ها از ۸۰ درصد تجاوز کند مقدار هزینه‌ی آن برابر با منفی بی‌نهایت شود که دلیل رزرو کردن این ۲۰ درصد به خاطر عدم قطعیت ترافیک شبکه و عدم دقت بار اشغالی پیش‌بینی‌شده توسط سرویس‌ها است. با مدل کردن مسئله بدین‌صورت، چارچوب نرم‌افزاری ما علاوه بر قرار دادن سرویس‌ها بر روی نودهای کم‌بار تر، برابری را نیز در توزیع سرویس‌ها رعایت می‌کند. شبه کد این الگوریتم نیز در الگوریتم (۴) در فصل پیوست نشان داده‌شده است.



شکل ۴-۸: کارگزاران جانمایی سرویس

۴-۶ جمع‌بندی

در این فصل، یک چارچوب نرم‌افزاری برای مسیریابی جریان‌ها و جانمایی سرویس‌ها در شبکه‌های موبایل تعریف‌شده با نرم‌افزار ارائه کردیم که در آن از ترکیب نسخه‌ی های اصلاح‌شده‌ی دو الگوریتم فرا ابتکاری ACS و GWO استفاده‌شده است. برخلاف اغلب الگوریتم‌های مسیریابی که همیشه بهترین مسیر را به‌عنوان جواب انتخاب می‌کنند، چارچوب نرم‌افزاری پیشنهادشده در این فصل در خصوص انتخاب مسیر انعطاف‌پذیری بیشتری در شبکه ایجاد می‌کند و این امکان را ایجاد می‌کند که علاوه بر انتخاب مسیر بهینه، دو مسیر دیگر به‌عنوان مسیر جایگزین برای جریان انتخاب شود.

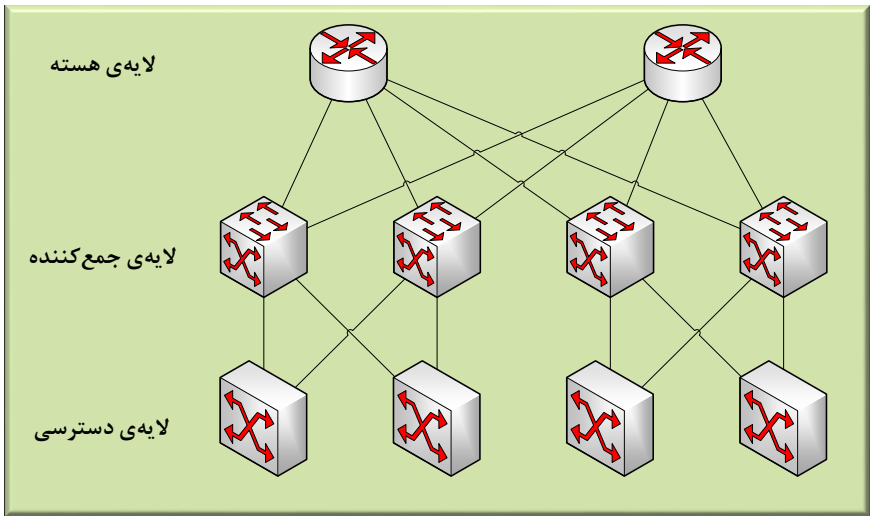
فصل پنجم

ارزیابی چارچوب‌های نرم‌افزاری پیشنهادی

در این فصل، نتایج حاصل از شبیه‌سازی‌های انجام‌شده بر روی چارچوب‌های نرم‌افزاری پیشنهادی در دو فصل قبلی بررسی‌شده و عملکرد آن‌ها با سایر الگوریتم‌های پیشنهادشده در مقالات مشابه مقایسه می‌شود. بدین منظور، برای هرکدام از چارچوب‌های نرم‌افزاری، ابتدا در خصوص محیط شبیه‌سازی[1] و نحوه‌ی انجام آن‌ها بحث می‌کنیم و در ادامه نتایج شبیه‌سازی‌ها و مقایسه‌ی آن‌ها با الگوریتم‌های دیگر آورده می‌شود.

۵-۱ نتایج شبیه‌سازی چارچوب نرم‌افزاری ارائه‌شده برای فراهم‌سازی کنترلرها

برای سنجش عملکرد و اثبات درستی این چارچوب نرم‌افزاری، ابتدا تمامی الگوریتم‌های بحث شده در فصل سوم را در نرم‌افزار MATLAB پیاده‌سازی کرده‌ایم. ما برای انجام شبیه‌سازی فرض کرده‌ایم که تنها یک کلاس کیفیت سرویس در شبکه وجود دارد که توپولوژی سوئیچینگ آن به‌صورت شکل (۵-۱) شامل ۳ لایه است.



شکل ۵-۱: توپولوژی استفاده‌شده در شبیه‌سازی مسئله‌ی فراهم‌سازی کنترلرها

برای اجرای شبیه‌سازی‌ها، ابتدا می‌بایست مقادیر مناسب برای نرخ ارسال درخواست‌ها توسط سوئیچ‌ها (λ_k)، نرخ سرویس‌دهی کنترلرها (μ) و پارامترهای استفاده‌شده در GWO را انتخاب نماییم که این کار به شرح زیر انجام‌شده است:

۱. **تعیین نرخ ارسال درخواست توسط سوئیچ‌ها** (λ_k): با توجه به اندازه‌گیری‌های انجام‌شده بر روی یک سوئیچ HP ProCurve 5406z در [72, 73]، فرض کرده‌ایم که نرخ اولیه‌ی ارسال

[1] Simulation Environment

درخواست توسط سوئیچ‌ها می‌تواند در بازه‌ی $[1500, 6000]$ انتخاب شود و متناسب با آن، نرخ ارسال بقیه‌ی لایه‌ها می‌تواند به‌صورت ضریبی از نرخ پایه مطابق رابطه‌ی زیر انتخاب شود.

$$w_{Edge} = 1, w_{Aggregation} = 2, w_{Core} = 4 \qquad (۱-۵)$$

۲. **تعیین نرخ سرویس‌دهی کنترلرها** (μ_j): بر اساس تحقیق صورت گرفته بر روی کنترلر NOX [74]، ما در شبیه‌سازی‌های انجام‌شده فرض کرده‌ایم که نرخ سرویس‌دهی هریک از کنترلرها می‌تواند ۳۰ هزار باشد و ازآنجایی‌که برای هرکدام از آن‌ها ۴ هسته در نظر گرفته‌ایم، نرخ سرویس‌دهی هر هسته برابر با ۷۵۰۰ خواهد بود.

۳. **تعیین پارامترهای الگوریتم GWO**: جدول (۵-۱) مقادیر انتخاب‌شده برای پارامترهای الگوریتم GWO را نشان می‌دهد. این مقادیر پس از انجام شبیه‌سازی‌های متنوع انتخاب‌شده‌اند و بهترین مقادیر متناسب با فضای شبیه‌سازی و فضای مسئله هستند. همان‌طور که قبلاً نیز توضیح داده‌شده است، برای تولید جمعیت اولیه‌ی الگوریتم از تابع CircleMap استفاده می‌کنیم و نقطه‌ی اولیه‌ی هر متغیر (سری CircleMap) را با استفاده از تابع تولید اعداد تصادفی در MATLAB مقداردهی اولیه می‌کنیم.

جدول ۵-۱: پارامترهای الگوریتم GWO برای مسئله‌ی فراهم‌سازی کنترلرها

پارامتر	مقدار
تعداد مراحل الگوریتم برای یافتن بهترین CA	۷۵
تعداد مراحل الگوریتم برای یافتن بهترین SA	۲۵
تعداد کارگزاران CA	۱۰
تعداد کارگزاران SA	۱۰
γ_T	۰.۷
γ_C	۰.۱۵
γ_F	۰.۱۵
λ_k	[۱۵۰۰,۶۰۰۰]
μ_j	۳۰۰۰۰

۵-۱-۱ نتایج شبیه‌سازی

برای بررسی چارچوب نرم‌افزاری ارائه‌شده به‌صورت دقیق و کامل، در جدول (۵-۲) چند سناریوی مختلف تعریف کرده‌ایم. سناریوی اول حالت اولیه‌ی شبکه را نشان می‌دهد و سناریوهای ۲ و ۳ حالتی از شبکه که

در آن‌ها نرخ ارسال درخواست توسط سوئیچ‌ها افزایش‌یافته یا به عبارتی حالتی که در آن ترافیک شبکه افزایش پیداکرده است را نشان می‌دهد.

جدول ۵-۲: سناریوهای تعریف‌شده برای مسئله‌ی فراهم‌سازی کنترلرها

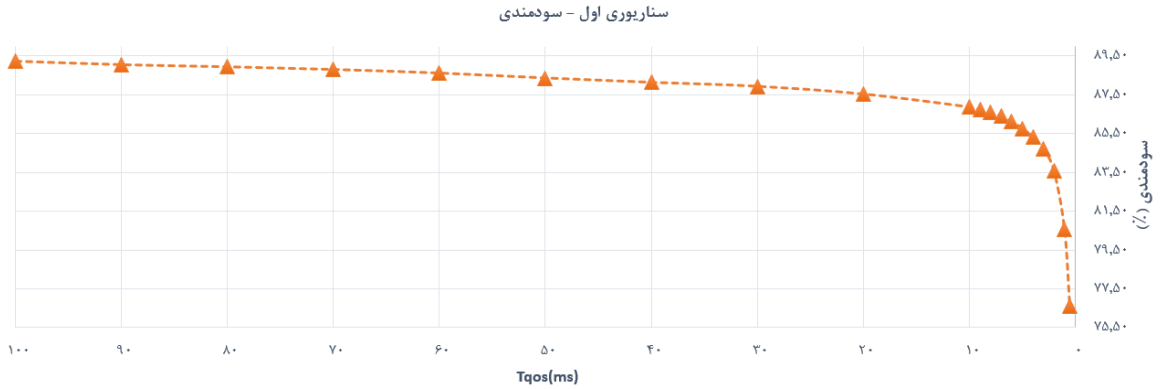
سناریو	$Min(\lambda_k$	$Max(\lambda_k)$	μ_j
۱	۱۵۰۰	۶۰۰۰	۴ ×۷۵۰۰
۲	۳۰۰۰	۱۲۰۰۰	۴ ×۷۵۰۰
۳	۶۰۰۰	۲۴۰۰۰	۴ ×۷۵۰۰

برای بررسی دقیق‌تر، علاوه بر تغییر نرخ ارسال درخواست توسط سوئیچ‌ها، ما برای هریک از سناریوهای تعریف‌شده مقدار (t_{QoS}) را هم تغییر داده‌ایم تا تأثیر نیازمندی زمانی کیفیت سرویس نیز بر تعداد کنترلرهای موردنیاز شبکه مشخص شود. جدول (۵-۳) جزئیات نتایج به‌دست‌آمده از سناریوی اول که شامل سودمندی شبکه، تعداد کنترلرها، نحوه‌ی اتصال سوئیچ‌ها و کنترلرها و زمان اجرای الگوریتم می‌شود را نشان می‌دهد. همچنین، برای بهتر مشخص‌شدن تأثیر t_{QoS} بر سودمندی شبکه و تعداد کنترلرها، نمودار تغییرات این مقادیر در شکل (۵-۲) نشان داده است که در آن با کاهش حدنصاب موردنیاز برای کیفیت سرویس (t_{QoS}) در شبکه، مقدار سودمندی شبکه کاهش می‌یابد و برای جبران این کاهش سودمندی، چارچوب نرم‌افزاری ما کنترلرهای بیشتری را برای شبکه فراهم می‌کند. همچنین، درصورتی‌که t_{QoS} را بیش از توان شبکه کاهش دهیم، چارچوب نرم‌افزاری قادر به پیدا کردن جواب قابل قبولی نخواهد بود و خروجی آن کم‌ترین مقدار ممکن برای کنترلرها (۱ عدد) و کمترین مقدار سودمندی یعنی ۳۰ درصد خواهد شد. همان‌طور که در سطر آخر جدول (۵-۳) نیز نشان داده، مقدار $t_{QoS} = 0.02ms$ یک نیازمندی غیرقابل‌قبول برای شبکه بوده و به همین خاطر مقدار سودمندی زمانی برابر با صفر شده و سودمندی کلی شبکه برابر با ۳۰ درصد (۱۵ درصد هزینه‌ی کنترلرها + ۱۵ درصد شاخص برابری) می‌شود.

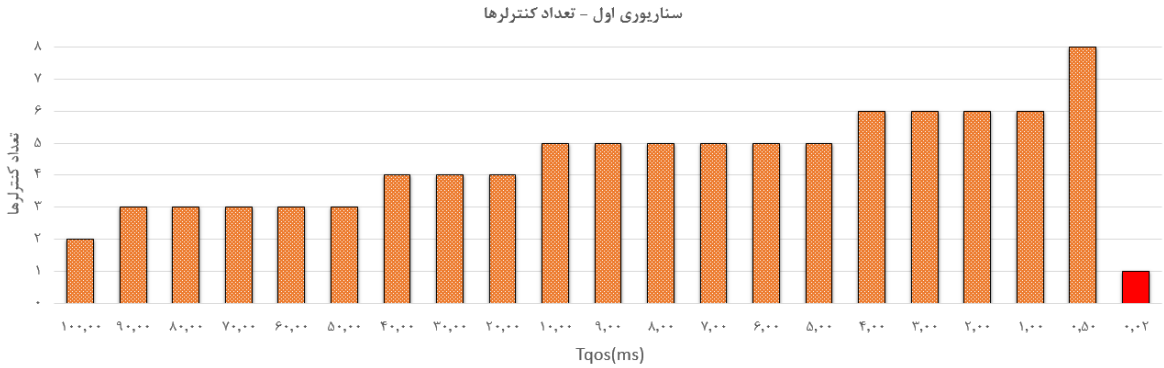
جدول ۵-۳: نتایج سناریوی اول

سناریوی اول				
زمان (ثانیه)	اتصالات کنترلرها و سوئیچ‌ها	تعداد کنترلرها	سودمندی(درصد)	t_{QoS}(ms)
۷۷.۱۸۷۵	[۲ ۱ ۲ ۱ ۲ ۱ ۲ ۲ ۱ ۱]	۲	۸۹.۲۲۴۶	۱۰۰
۸۶.۷۸۱۳	[۱ ۲ ۳ ۲ ۱ ۳ ۳ ۳ ۱ ۳]	۳	۸۹.۰۴۲۵	۹۰
۸۴.۲۰۳۱	[۳ ۱ ۱ ۲ ۲ ۳ ۲ ۳ ۲ ۲]	۳	۸۸.۹۳۴۳	۸۰
۱۰۲.۲۱۸۸	[۳ ۱ ۳ ۲ ۲ ۲ ۱ ۲ ۱ ۳]	۳	۸۸.۷۹۵۹	۷۰
۹۴.۲۶۵۶	[۱ ۳ ۲ ۲ ۱ ۲ ۳ ۲ ۳ ۱]	۳	۸۸.۶۱۲۶	۶۰
۸۸.۵۷۸۱	[۱ ۳ ۲ ۲ ۲ ۱ ۳ ۱ ۳ ۲]	۳	۸۸.۳۵۸۴	۵۰
۹۸.۵۶۲۵	[۴ ۲ ۱ ۳ ۱ ۳ ۱ ۲ ۳ ۴]	۴	۸۸.۱۳۳۱	۴۰
۱۱۰.۱۴۰۶	[۴ ۱ ۲ ۳ ۲ ۳ ۲ ۳ ۱ ۴]	۴	۸۷.۹۳۰۵	۳۰
۱۱۵.۹۶۸۸	[۱ ۴ ۳ ۳ ۲ ۲ ۴ ۲ ۳ ۱]	۴	۸۷.۵۳۰۱	۲۰
۱۱۹.۷۸۱۳	[۳ ۱ ۲ ۲ ۴ ۵ ۴ ۵ ۵ ۴]	۵	۸۶.۸۶۱۳	۱۰
۱۱۰.۳۴۳۸	[۲ ۱ ۵ ۳ ۴ ۵ ۳ ۴ ۳ ۴]	۵	۸۶.۷۳۷۷	۹
۱۱۲.۵۷۸۱	[۵ ۲ ۴ ۳ ۱ ۳ ۱ ۴ ۱ ۴]	۵	۸۶.۵۸۴۱	۸
۱۱۵	[۵ ۱ ۴ ۲ ۲ ۳ ۳ ۳ ۴ ۴]	۵	۸۶.۳۸۸	۷
۱۱۲.۸۹۰۶	[۴ ۵ ۲ ۳ ۱ ۱ ۲ ۲ ۳ ۳]	۵	۸۶.۱۲۸۹	۶
۱۱۷	[۵ ۴ ۳ ۲ ۲ ۱ ۳ ۱ ۱ ۳]	۵	۸۵.۷۷۰۸	۵
۱۱۱.۵۴۶۹	[۶ ۱ ۳ ۲ ۴ ۵ ۲ ۴ ۳ ۵]	۶	۸۵.۳۱۷۲	۴
۱۲۳.۵۴۶۹	[۶ ۱ ۲ ۴ ۳ ۵ ۲ ۴ ۳ ۵]	۶	۸۴.۷۲۱۸	۳
۱۲۴.۵۹۳۸	[۲ ۶ ۱ ۵ ۴ ۳ ۵ ۱ ۴ ۳]	۶	۸۳.۵۸۲۳	۲
۱۱۹.۳۵۹۴	[۵ ۱ ۳ ۴ ۲ ۶ ۲ ۴ ۳ ۶]	۶	۸۰.۵۷۴	۱
۱۴۲.۲۸۱۳	[۷ ۵ ۸ ۱ ۴ ۲ ۳ ۳ ۶ ۶]	۸	۷۶.۵۸۵۹	۰.۵
۷۳.۲۸۱۳	[۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱]	۱	۳۰	۰.۰۲

برای درک بهتر از خروجی چارچوب نرم‌افزاری برای اتصالات بین کنترلرها و سوئیچ‌ها، یک نمونه از آن را برای حالتی از سناریوی اول که $t_{QoS} = 100ms$ و $SA = [2, 1, 2, 1, 2, 1, 2, 2, 1, 1]$ است، در شکل (۵-۳) نشان داده‌شده است.

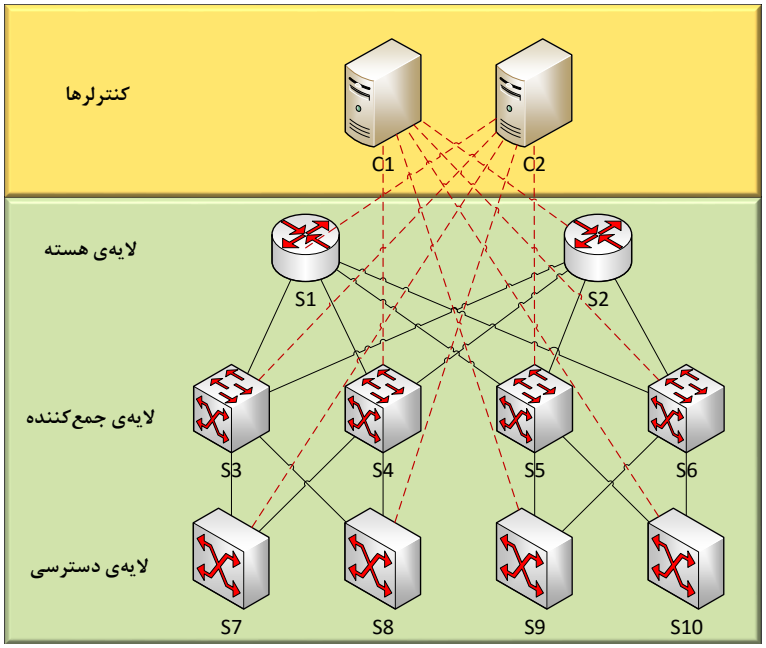


(آ) تأثیر t_{QoS} بر روی سودمندی شبکه



(ب) تأثیر t_{QoS} بر روی تعداد کنترلرها

شکل ۵-۲: نتایج سناریوی اول



شکل ۵-۳: نمونه‌ی اتصالات کنترلرها و سوئیچ‌ها

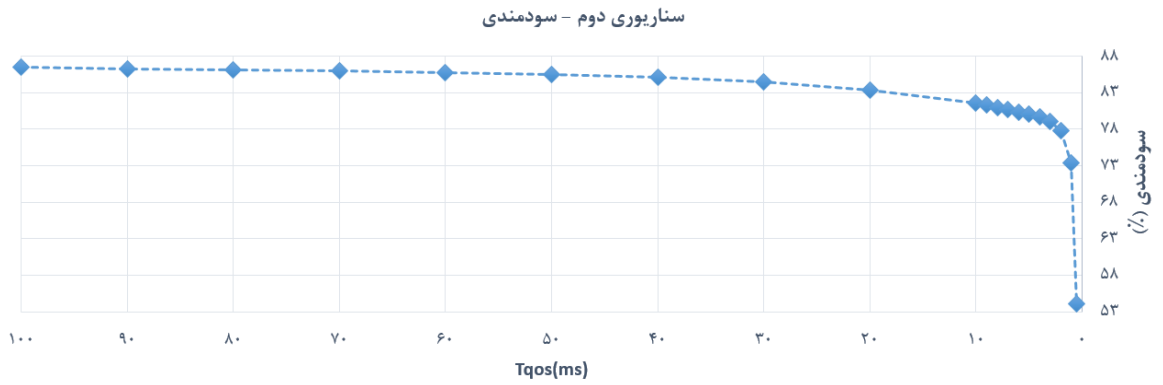
جزئیات نتایج سناریوی دوم و سوم نیز در جداول (۵-۴) و (۵-۵) نشان داده‌شده و همان‌طور که شکل‌های (۵-۴) و (۵-۵) نشان می‌دهند، در این سناریوها نیز مشابه سناریوی اول با کاهش حدنصاب موردنیاز برای کیفیت سرویس (t_{QoS}) در شبکه، سودمندی شبکه کاهش پیدا می‌کند ولی به دلیل ترافیک بیش‌تر در شبکه، کاهش سودمندی با شدت بیشتری صورت می‌گیرد و در ضمن تعداد کنترلرهای فراهم‌شده در این سناریوها در مقایسه با سناریوی اول خیلی بیش‌تر است.

جدول ۵-۴: نتایج سناریوی دوم

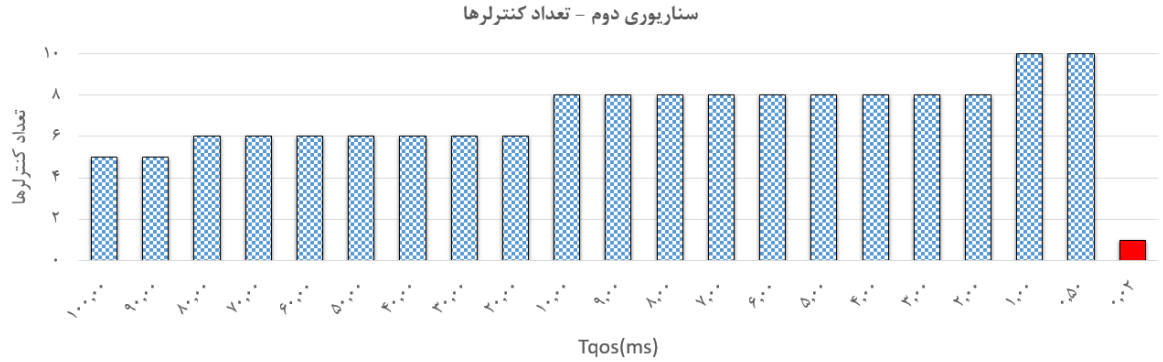
سناریوی دوم				
زمان (ثانیه)	اتصالات کنترلرها و سوئیچ‌ها	تعداد کنترلرها	سودمندی(درصد)	t_{QoS}(ms)
۱۱۰.۶۴۰۶	[۱ ۵ ۲ ۳ ۴ ۳ ۴ ۲ ۲ ۴]	۵	۸۶.۴۳۲۸	۱۰۰
۱۱۸.۷۱۸۸	[۴ ۱ ۲ ۵ ۳ ۵ ۲ ۳ ۲ ۳]	۵	۸۶.۲۶۴۴	۹۰
۱۲۹.۰۹۳۸	[۶ ۱ ۲ ۵ ۴ ۳ ۴ ۵ ۲ ۳]	۶	۸۶.۱۳۶	۸۰
۱۲۲.۳۷۵	[۳ ۲ ۴ ۵ ۱ ۶ ۵ ۶ ۱ ۴]	۶	۸۵.۹۸۷۹	۷۰
۱۲۱.۵۴۶۹	[۱ ۶ ۳ ۲ ۴ ۵ ۴ ۳ ۵ ۲]	۶	۸۵.۷۹۲۳	۶۰
۱۲۱.۸۱۲۵	[۱ ۵ ۲ ۴ ۳ ۶ ۴ ۲ ۶ ۳]	۶	۸۵.۵۲۲	۵۰
۱۲۷.۵۳۱۳	[۵ ۶ ۱ ۴ ۲ ۳ ۱ ۴ ۲ ۳]	۶	۸۵.۱۲۴۳	۴۰
۱۴۷.۵۳۱۳	[۶ ۱ ۲ ۵ ۳ ۴ ۳ ۲ ۴ ۵]	۶	۸۴.۴۸۲۱	۳۰
۱۲۹.۰۳۱۳	[۵ ۳ ۲ ۴ ۱ ۶ ۲ ۴ ۱ ۶]	۶	۸۳.۲۷۵۴	۲۰
۱۳۴.۹۶۸۸	[۳ ۷ ۲ ۸ ۱ ۶ ۴ ۵ ۵ ۴]	۸	۸۱.۵۴۲۱	۱۰
۱۵۵.۲۰۳۱	[۶ ۳ ۵ ۱ ۴ ۸ ۲ ۷ ۲ ۷]	۸	۸۱.۲۷۳۹	۹
۱۴۳.۷۸۱۳	[۸ ۱ ۶ ۷ ۴ ۲ ۵ ۳ ۳ ۵]	۸	۸۰.۹۷۹۶	۸
۱۵۳.۳۴۳۸	[۳ ۷ ۱ ۸ ۴ ۲ ۶ ۶ ۵ ۵]	۸	۸۰.۶۶۸۱	۷
۱۶۹.۶۴۰۶	[۶ ۵ ۸ ۲ ۷ ۱ ۴ ۳ ۳ ۴]	۸	۸۰.۳۷	۶
۱۵۸.۳۲۵۱	[۸ ۳ ۴ ۶ ۲ ۵ ۱ ۷ ۱ ۷]	۸	۸۰.۰۹۵۹	۵
۱۶۵.۰۷۸۱	[۷ ۸ ۴ ۶ ۲ ۵ ۱ ۳ ۳ ۱]	۸	۷۹.۶۹۹	۴
۱۴۷.۲۹۶۹	[۱ ۸ ۵ ۷ ۶ ۴ ۳ ۲ ۳ ۲]	۸	۷۹.۰۴۶۹	۳
۱۵۴.۱۵۶۳	[۱ ۴ ۲ ۸ ۳ ۶ ۵ ۷ ۷ ۵]	۸	۷۷.۷۳۱۹	۲
۱۵۷.۳۹۰۶	[۶ ۲ ۷ ۳ ۱۰ ۸ ۴ ۵ ۱ ۹]	۱۰	۷۳.۳۶۲۱	۱
۱۶۰.۸۵۹۴	[۶ ۹ ۲ ۵ ۴ ۱۰ ۱ ۸ ۳ ۷]	۱۰	۵۴.۰۶۲۷	۰.۵
۷۹.۵۱۵۶	[۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱]	۱	۳۰	۰.۳

جدول ۵-۵: نتایج سناریوی سوم

سناریوی سوم				
زمان (ثانیه)	اتصالات کنترلرها و سوئیچ‌ها	تعداد کنترلرها	سودمندی(درصد)	t_{QoS}(ms)
۱۳۰.۵۳۱۳	[۳ ۱ ۷ ۸ ۴ ۶ ۵ ۵ ۲ ۲]	۸	۸۱.۴۰۶۶	۱۰۰
۱۴۷.۱۵۶۳	[۵ ۲ ۷ ۴ ۸ ۱ ۶ ۳ ۶ ۳]	۸	۸۱.۰۹۷۷	۹۰
۱۴۰.۹۵۳۱	[۷ ۱ ۸ ۳ ۴ ۶ ۲ ۵ ۲ ۵]	۸	۸۰.۷۴۵۴	۸۰
۱۳۲	[۶ ۱ ۲ ۷ ۴ ۸ ۳ ۳ ۵ ۵]	۸	۸۰.۳۴۷۵	۷۰
۱۳۸.۸۲۸۱	[۴ ۵ ۲ ۶ ۱ ۸ ۳ ۷ ۳ ۷]	۸	۷۹.۹۱۳۵	۶۰
۱۵۱.۲۵	[۶ ۷ ۵ ۴ ۸ ۱ ۲ ۲ ۳ ۳]	۸	۷۹.۴۸۶۸	۵۰
۱۵۳.۱۵۶۳	[۱ ۷ ۳ ۴ ۸ ۵ ۶ ۲ ۶ ۲]	۹	۷۸.۹۵۹۷	۴۰
۱۵۹.۶۲۵	[۹ ۴ ۸ ۲ ۱۰ ۳ ۷ ۵ ۱ ۶]	۱۰	۷۸.۱۳۳۷	۳۰
۱۴۶.۶۴۰۶	[۷ ۳ ۸ ۴ ۱ ۵ ۶ ۴ ۲ ۹]	۹	۷۶.۳۴۲۶	۲۰
۱۴۹.۶۷۱۹	[۸ ۷ ۱ ۹ ۳ ۲ ۴ ۶ ۱۰ ۵]	۱۰	۷۴.۰۶۸۶	۱۰
۱۴۸.۵۴۶۹	[۳ ۹ ۵ ۴ ۱ ۱۰ ۷ ۸ ۲ ۶]	۱۰	۷۳.۲۰۹۱	۹
۱۴۹.۳۹۲۵	[۸ ۶ ۲ ۱۰ ۵ ۱ ۷ ۳ ۴ ۹]	۱۰	۷۱.۱۲۱۹	۸
۱۵۰.۰۱۵۶	[۱۰ ۱ ۷ ۴ ۲ ۹ ۳ ۶ ۵ ۸]	۱۰	۷۰.۱۴۹۲	۷
۱۴۷.۲۹۶۹	[۷ ۱ ۸ ۳ ۱۰ ۴ ۶ ۲ ۹ ۵]	۱۰	۶۷.۰۷۹	۶
۱۴۹.۸۱۲۵	[۳ ۲ ۱ ۱۰ ۷ ۶ ۵ ۸ ۹ ۴]	۱۰	۶۱.۱۷۲۷	۵
۱۴۵.۹۸	[۱ ۸ ۲ ۷ ۹ ۱۰ ۴ ۳ ۵ ۶]	۱۰	۴۶.۱۲۸۵	۴
۷۹.۵۱۵۶	[۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱ ۱]	۱	۳۰	۳

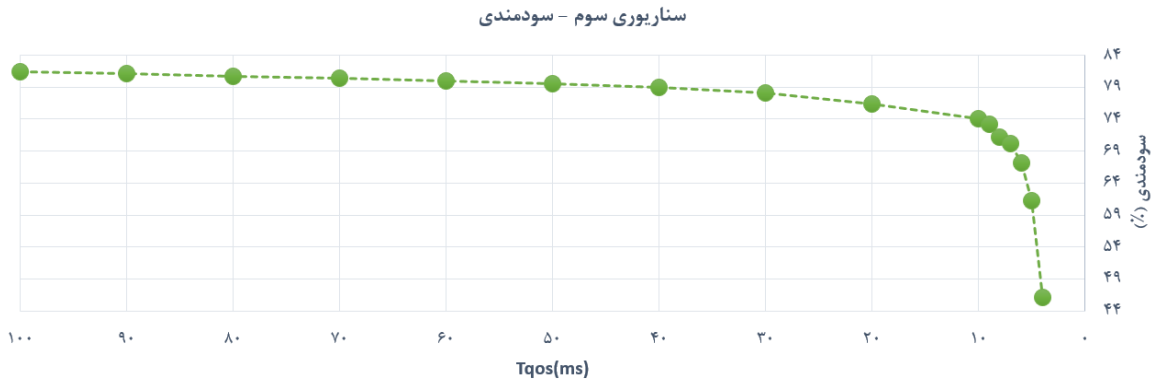


(آ) تأثیر t_{QoS} بر روی سودمندی شبکه



(ب) تأثیر t_{QoS} بر روی تعداد کنترلرها

شکل ۵-۴: نتایج سناریوی دوم



(آ) تأثیر t_{QoS} بر روی سودمندی شبکه

(ب) تأثیر t_{QoS} بر روی تعداد کنترلرها

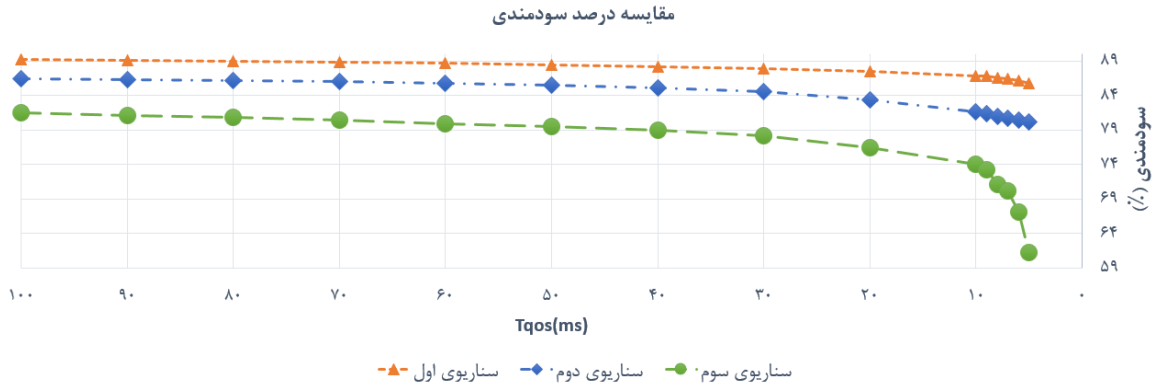
شکل ۵-۵: نتایج سناریوی سوم

برای مقایسه‌ی دقیق‌تر این سناریوها و مشخص‌شدن تأثیر افزایش نرخ ارسال درخواست سوئیچ‌ها (λ_k)، نتایج شبیه‌سازی‌های انجام‌شده برای هر سه سناریو در شکل (۵-۶) آورده شده است و همان‌طور که مشاهده می‌شود، افزایش ترافیک شبکه باعث می‌شود تا با کاهش t_{QoS} سودمندی شبکه با سرعت بیشتری اتفاق بیفتد و همچنین مقدار بیشینه و مقدار میانگین سودمندی نیز کاهش پیدا کند. به‌علاوه، تعداد کنترلرهای موردنیاز برای شبکه برای حالت‌هایی با t_{QoS} یکسان، با افزایش ترافیک با شیب بیشتری زیاد می‌شود.

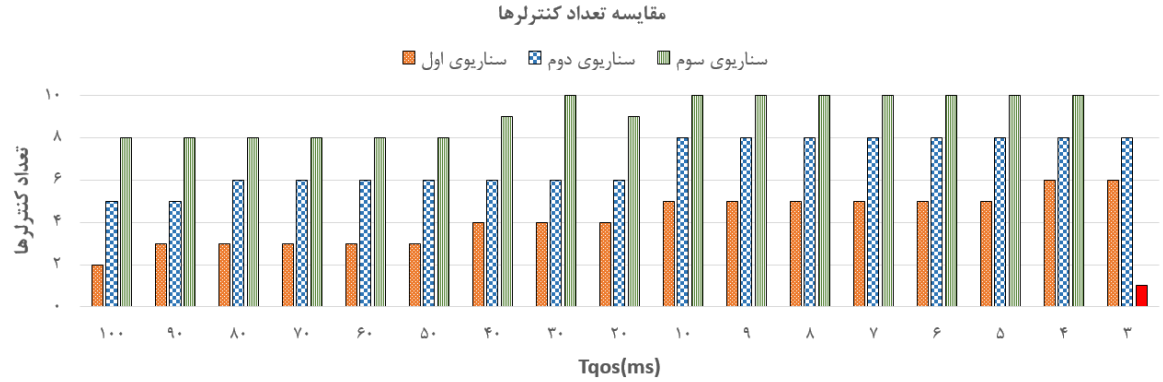
۵-۱-۲ مقایسه

برای نشان دادن ویژگی‌های خوب الگوریتم گرگ‌های خاکستری و سری‌های آشوبناک، در این بخش قصد داریم تا الگوریتم استفاده‌شده در چارچوب نرم‌افزاری‌مان را با الگوریتم PSO مقایسه کنیم. الگوریتم PSO [37, 38, 57] یک الگوریتم بهینه‌سازی فرا ابتکاری چند جوابی است که از رفتار اجتماعی پرندگان در طبیعت الهام گرفته‌شده است. به هر عضو از جمعیت الگوریتم PSO، ذره[1] گفته می‌شود که با استفاده

[1] Particle



(آ) تأثیر λ_k بر روی سودمندی شبکه



(ب) تأثیر λ_k بر روی تعداد کنترلرها

شکل ۵-۶: مقایسه‌ی نتایج سناریوها

از روابط (۵-۲) تا (۵-۴) و با توجه به موقعیت بهترین ذره‌ی محلی [۲] و بهترین ذره‌ی کلی [۳] موقعیت‌هایشان در هر مرحله از الگوریتم به‌روز می‌شود.

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (۵-۲)$$

$$V_i(t+1) = \omega(t+1)V_i(t) + C_1.r_1(Y_L - X_i(t)) + C_2.r_2(Y_G - X_i(t)) \quad (۵-۳)$$

$$\omega(t+1) = \omega_{damp}.\omega(t) \quad (۵-۴)$$

$X_i(t)$ و $V_i(t)$ نشان‌دهنده‌ی موقعیت و سرعت ذره‌ی i ام در t امین مرحله از الگوریتم هستند. Y_L موقعیت بهترین ذره‌ی محلی برای هر ذره و Y_G موقعیت بهترین ذره‌ی کلی به‌دست‌آمده تا به اینجا می‌شود. C_1 و C_2 ضرایب به‌روزرسانی موقعیت بر اساس نقاط بهینه‌ی محلی و کلی هستند که برای بهبود رفتار اکتشافی در الگوریتم این مقادیر در دو مقدار تصادفی (r_1, r_2) نیز ضرب می‌شوند. در الگوریتم PSO استفاده‌شده برای انجام مقایسه، ما پارامتر ω را نیز در نظر گرفته‌ایم که تأثیر سرعت به‌جامانده از مراحل قبلی الگوریتم را کاهش می‌دهد و منجر به بهبود عملکرد آن می‌شود. برای انجام شبیه‌سازی‌ها از پارامترهای نشان داده‌شده در جداول (۵-۱) و (۵-۶) استفاده می‌کنیم.

جدول ۵-۶: پارامترهای الگوریتم PSO برای مسئله‌ی فراهم‌سازی کنترلرها

پارامتر	مقدار
C_1	۲
C_2	۲
$\omega(1)$	۱
ω_{damp}	۰.۹

برای مقایسه‌ی این الگوریتم‌ها، الگوریتم PSO را برای پنج مقدار مختلف t_{QoS} اجرا کرده‌ایم و نتایج آن را با نتایج به‌دست‌آمده در بخش قبلی مقایسه کرده‌ایم که این مقادیر در جدول (۵-۷) نشان داده‌شده‌اند.

[۲] Local Best
[۳] Global Best

جدول ۵-۷: نتایج مقایسه - Chaotic GWO و PSO

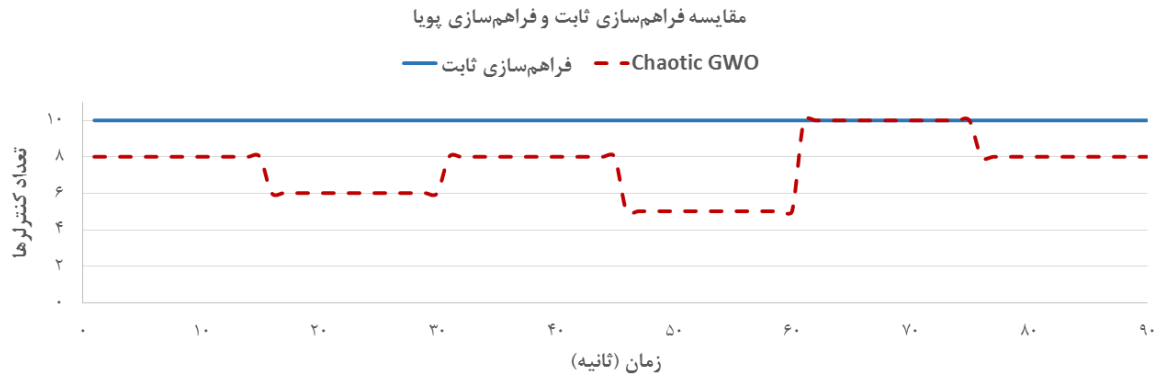
سناریو	QoS (ms)	سودمندی (درصد)		تعداد کنترلرها		زمان (ثانیه)		تعداد مراحل الگوریتم	
		Chaotic GWO	PSO	Chaotic GWO	PSO	Chaotic GWO	PSO	Chaotic GWO	PSO
۱	۱۰۰	۸۹.۲۲۴۶	۸۹.۲۲۴۶	۲	۲	۷۷.۱۸۷۵	۵۸.۹	۱	۱
	۲۰	۸۷.۵۳۰۱	۸۷.۵۳۰۱	۴	۴	۱۱۵.۹۶۸۸	۸۰.۸۱۲۵	۱۰	۳۲
	۵	۸۵.۷۷۰۸	۸۵.۶۸۲۶	۵	۶	۱۱۷	۹۹.۳۷۵	۱۶	۱
	۱	۸۰.۵۷۴	۸۰.۵۷۴	۶	۶	۱۱۹.۳۵۹۴	۱۰۶.۱۰۹۴	۱۱	۴۹
	۰.۵	۷۶.۵۸۵۹	۷۶.۴۰۳۲	۸	۶	۱۴۲.۲۸۱۳	۱۰۵.۲۹۶۹	۴	۶۸
۲	۱۰۰	۸۶.۴۳۲۸	۸۶.۴۳۲۸	۵	۵	۱۱۰.۶۴۰۶	۹۷.۶۰۹۴	۱۳	۷۳
	۲۰	۸۳.۲۷۵۴	۸۳.۰۵۰۱	۶	۷	۱۱۸.۷۱۸۸	۱۰۵.۳۷۵	۷	۱
	۵	۸۰.۰۹۵۹	۸۰.۰۹۵۹	۸	۸	۱۵۸.۳۲۵۱	۱۱۳.۱۲۵	۱۹	۴۷
	۱	۷۳.۳۶۲۱	۷۲.۷۸۲	۱۰	۸	۱۵۷.۳۹۰۶	۹۵	۱۸	۵۶
	۰.۵	۵۴.۰۶۲۷	۵۱.۰۱۹۳	۱۰	۸	۱۶۰.۸۵۹۴	۸۰.۹۸۴۴	۳	۲
۳	۱۰۰	۸۱.۴۰۶۶	۸۱.۴۰۶۶	۸	۸	۱۱۰.۶۴۰۶	۱۰۴.۴۵۳۱	۱۳	۱۱
	۲۰	۷۶.۳۴۲۶	۷۵.۵۱۱۴	۹	۸	۱۴۶.۶۴۰۶	۱۰۵.۱۰۹۴	۱	۳۴
	۱۰	۷۴.۰۶۸۶	۷۳.۴۴۰۶	۱۰	۹	۱۴۹.۶۷۱۹	۱۰۸.۹۶۸۸	۳	۲۲
	۵	۶۱.۱۷۲۷	۵۷.۵۱۶۷	۱۰	۷	۱۴۹.۸۱۲۵	۹۳.۴۳۷۵	۳	۲
	۴	۴۶.۱۲۸۵	۴۶.۱۲۸۵	۱۰	۱۰	۱۴۵.۹۸	۱۰۳.۷۸۱۳	۴	۲
متوسط		۷۵.۷۳۵۵	۷۵.۱۱۹۸۹	۷.۴	۶.۸	۱۳۰.۲۲۰۴	۹۷.۲۲۲۵۱	۸.۴	۲۶.۷۳۳

نتایج نشان می‌دهد که در بعضی از حالت‌ها، الگوریتم PSO نتوانسته است بهترین جواب ممکن را پیدا کند و در نقاط بهینه‌ی محلی گیرکرده است (خانه‌های قرمز رنگ در جدول). به همین دلیل، الگوریتم پیشنهادی ما یعنی گرگ‌های خاکستری آشوبناک در مقایسه با الگوریتم PSO عملکرد بهتری دارد و در ۶۸ درصد تعداد مراحل کمتر به جواب‌هایی با ۱ درصد دقت بیشتر می‌رسد.

همان‌طور که در فصل‌های گذشته نیز توضیح داده شد، به دلیل تغییر مداوم ترافیک شبکه‌های موبایل بهتر است تا تعداد کنترلرهای موردنیاز در شبکه به‌صورت پویا و متناسب با ترافیک شبکه محاسبه شود. به همین خاطر، در این بخش قصد داریم تا چارچوب نرم‌افزاری پیشنهادمان را باحالتی که در آن تعداد کنترلرها با صورت ثابت[۱] مشخص می‌شود مقایسه کنیم. برای مشخص کردن تعداد ثابت کنترلرها، می‌بایست تعداد کنترلرها را برای بدترین حالت محاسبه کرد و باید برای هر سوئیچ یک کنترلر مجزا در نظر گرفته شود ولی در حالت پیشنهادی ما، می‌توان ترافیک شبکه را به‌صورت دوره‌ای به‌صورت یک توزیع پواسون در نظر گرفت و از λ آن به‌عنوان نرخ ارسال درخواست سوئیچ‌ها استفاده کرد. دادگان[۲] مورداستفاده قرارگرفته در این بخش از بررسی یک شبکه‌ی محلی[۳] استخراج‌شده است و در آن فرض کرده‌ایم که در شبکه تنها یک کلاس کیفیت سرویس با $t_{QoS} = 5ms$ داریم که نرخ ارسال درخواست سوئیچ‌ها برحسب شکل (۵-۷) در طول زمان تغییر می‌کنند. شکل (۵-۸) نیز تعداد کنترلرها با استفاده از دو روش را نشان می‌دهد.

[۱] Static Allocation
[۲] Dataset
[۳] Local Network Trace

شکل ۵-۷: تغییرات نرخ ارسال درخواست سوئیچ‌ها در طول زمان



شکل ۵-۸: مقایسه‌ی فراهم‌سازی کنترلرها به صورت ثابت و به صورت پویا

همان‌طور که مشاهده می‌شود، روش پیشنهادی ما در دوره‌های مختلف زمانی تعداد کنترلرها را متناسب با نرخ ارسال درخواست‌های سوئیچ‌ها محاسبه نموده که این روش منجر به کاهش هزینه‌های شبکه در مقایسه با فراهم‌سازی ثابت کنترلرها می‌شود.

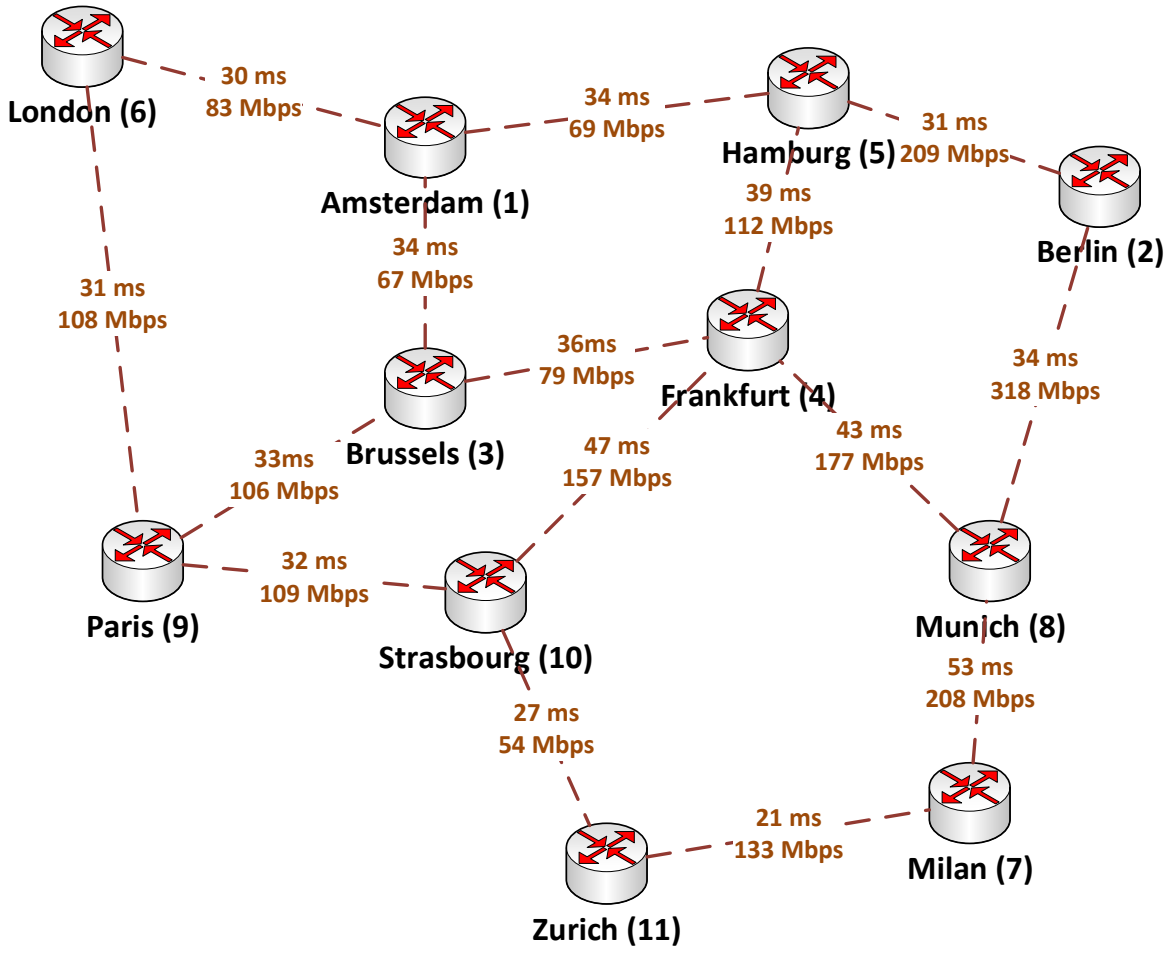
۵-۲ نتایج شبیه‌سازی چارچوب نرم‌افزاری ارائه‌شده برای مسیریابی جریان‌ها و جانمایی سرویس‌ها

برای سنجش عملکرد و اثبات درستی این چارچوب نرم‌افزاری، ابتدا تمامی الگوریتم‌های بحث شده در فصل چهارم را در MATLAB پیاده‌سازی کرده‌ایم و سپس با اجرای چارچوب نرم‌افزاری پیشنهادی برای جریان‌های مختلف و دو توپولوژی متفاوت آن را بررسی نموده‌ایم. به همین منظور ابتدا به معرفی

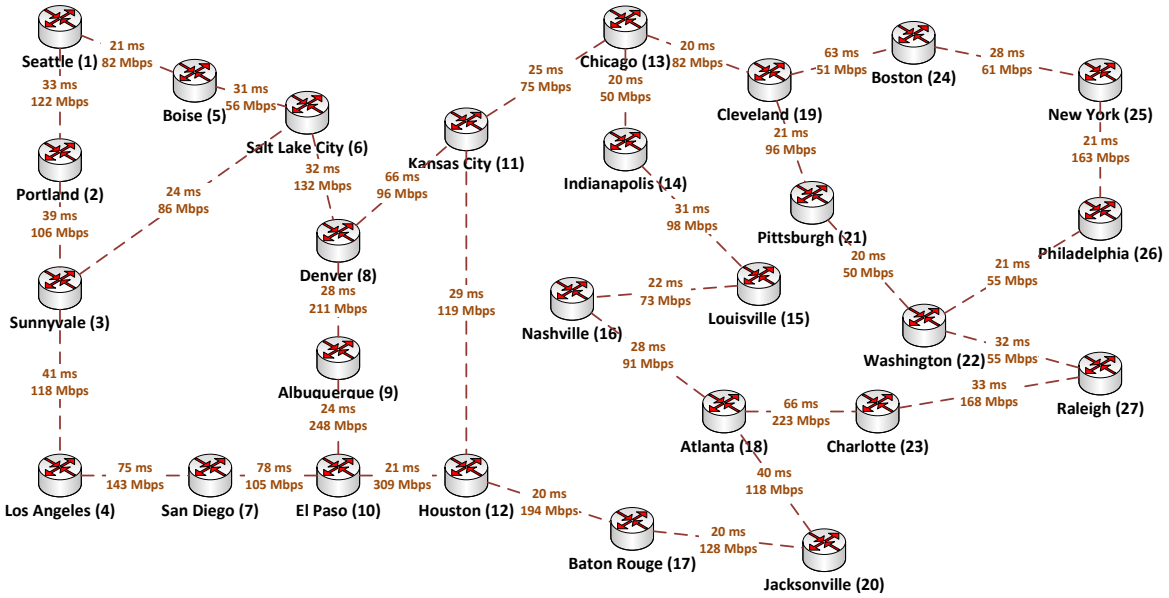
محیط شبیه‌سازی که شامل توپولوژی‌ها و پارامترهای مربوط به الگوریتم‌هاست می‌پردازیم و سپس نتایج شبیه‌سازی به‌دست‌آمده را بررسی و آن‌ها را با نتایج الگوریتم‌های مشابه مقایسه می‌نماییم.

۵-۲-۱ محیط شبیه‌سازی

ما از دو توپولوژی فرضی که ارتباط بین شهرهای مختلف در اروپا [75] و آمریکا [76] را نشان می‌دهد استفاده کرده‌ایم و برای سادگی شبیه‌سازی‌ها، تنها تأخیر و پهنای باند را به‌عنوان معیارهای موردنیاز برای مسیریابی جریان موردنظر قرار داده و این مقادیر را برای لینک‌های مختلف در توپولوژی‌ها استفاده‌شده به‌صورت تصادفی مقداردهی اولیه کرده‌ایم. در پروسه‌ی مقداردهی اولیه، بر اساس اطلاعات یک دادگان محلی فرض کرده‌ایم که تأخیر لینک‌ها در بازه‌ی $(20, 100)$ میلی‌ثانیه و پهنای باند لینک‌ها در بازه‌ی $(50, 500)$ مگابیت بر ثانیه قرار دارند. توپولوژی‌های مورداستفاده قرارگرفته در شکل‌های (۵-۹) و (۵-۱۰) نشان داده‌شده‌اند.



شکل ۵-۹: توپولوژی اروپا



شکل ۵-۱۰: توپولوژی آمریکا

جدول (۵-۸) پارامترهای مربوط به الگوریتم‌های ACS و GWO برای انجام شبیه‌سازی‌ها را نشان می‌دهند که این مقادیر پس از انجام شبیه‌سازی‌های مختلف انتخاب‌شده‌اند. همچنین، برای مقداردهی مقادیر فرومن‌ها در الگوریتم ACS اعداد تصادفی در بازه‌ی $(0, 1)$ تولید کرده‌ایم.

جدول ۵-۸: پارامترهای الگوریتم ACS

پارامتر	مقدار
تعداد مراحل الگوریتم ACS	۲۰
تعداد مورچه‌ها	۱۰
r_0	۰.۶
ρ_1	۰.۳
ρ_2	۰.۵
تعداد مراحل الگوریتم GWO	۲۵
تعداد کارگزاران الگوریتم GWO	۱۰

همان‌طور که قبلاً نیز توضیح داده‌شده، مشابه شبیه‌سازی‌های صورت گرفته برای چارچوب نرم‌افزاری پیشنهادشده برای فراهم‌سازی کنترلرها، در الگوریتم GWO برای تولید اعداد تصادفی از تابع CircleMap استفاده نموده و نقطه‌ی اولیه‌ی این تابع (θ_0) را با استفاده از تابع تولید اعداد تصادفی در MATLAB مقداردهی کرده‌ایم.

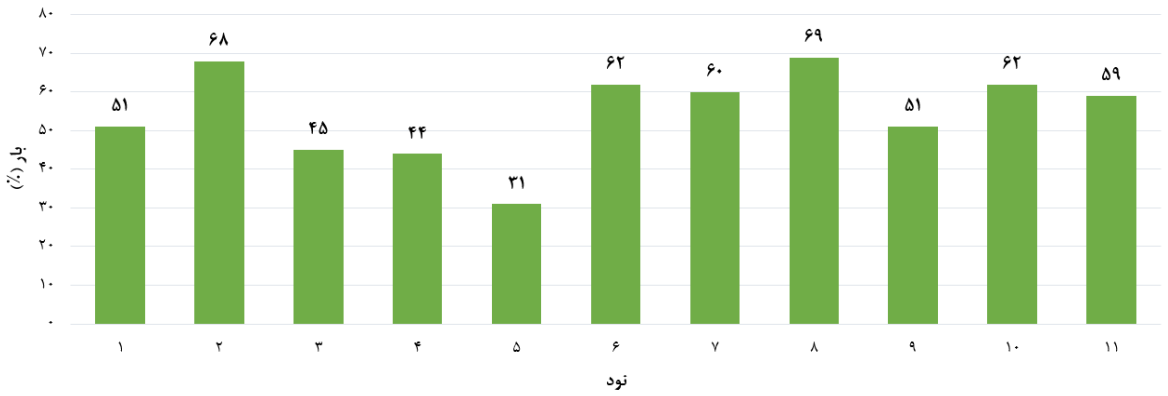
۵-۲-۲ نتایج شبیه‌سازی

برای انجام شبیه‌سازی‌ها، جریان‌های مختلفی را مطابق جدول (۵-۹) تعریف کرده‌ایم که هرکدام از آن‌ها نیازمندی‌های متفاوتی برای تأخیر، پهنای باند داشته و به سرویس‌های متفاوتی (با بار محاسباتی متفاوت) نیازمندند.

جدول ۵-۹: جریان‌های تعریف‌شده

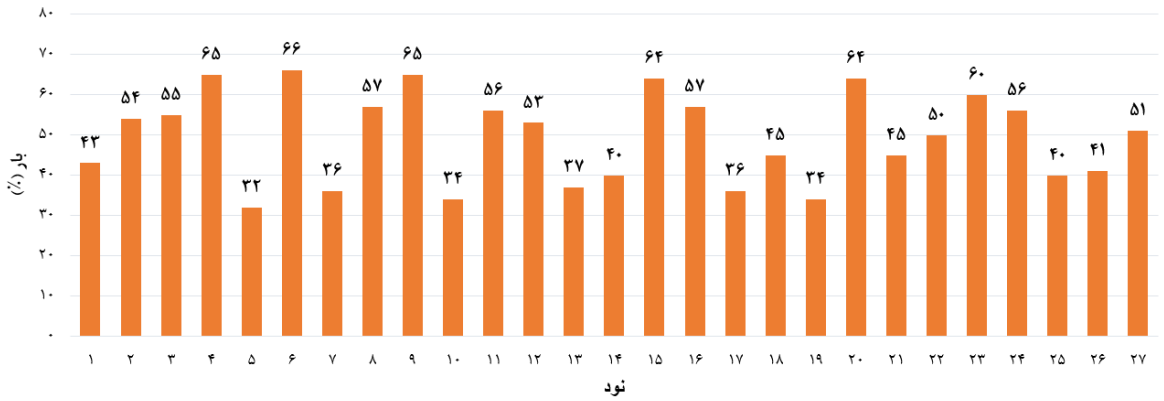
سرویس‌ها	کم‌ترین پهنای باند	بیش‌ترین تأخیر	مقصد	مبدأ	توپولوژی	جریان
S_1, S_2, S_3	$۵۰ Mbps$	$۵۰۰ ms$	میلان(۷)	لندن(۶)	اروپا	۱
S_1, S_2, S_4	$۱۰۰ Mbps$	$۵۰۰ ms$	میلان(۷)	لندن(۶)	اروپا	۱
S_1, S_4, S_5	$۵۰ Mbps$	$۳۰۰ ms$	میلان(۷)	لندن(۶)	اروپا	۱
S_2, S_4, S_5	$۱۰۰ Mbps$	$۳۰۰ ms$	میلان(۷)	لندن(۶)	اروپا	۱
S_1, S_2, S_3	$۵۰ Mbps$	$۸۰۰ ms$	رالی(۲۷)	سیاتل(۱)	آمریکا	۱
S_2, S_4, S_5	$۱۰۰ Mbps$	$۸۰۰ ms$	رالی(۲۷)	سیاتل(۱)	آمریکا	۱
S_1, S_4, S_5	$۵۰ Mbps$	$۵۰۰ ms$	رالی(۲۷)	سیاتل(۱)	آمریکا	۱
S_1, S_2, S_4	$۱۰۰ Mbps$	$۵۰۰ ms$	رالی(۲۷)	سیاتل(۱)	آمریکا	۱

ما همچنین فرض کرده‌ایم که تمامی ابرک‌های متصل به شبکه عیناً مثل هم[1] هستند و توان پردازشی یکسانی دارند. برای انجام شبیه‌سازی‌های مربوط به جانمایی سرویس‌ها، ابتدا بار هریک از ابرک‌های متصل به نودهای شبکه را به‌صورت تصادفی در بازه‌ی $(30, 70)$ مقداردهی کرده و سپس پنج نوع سرویس مختلف که هرکدام مقدار متفاوتی از توان پردازشی نیازمندند تعریف نموده‌ایم. بار اولیه‌ی ابرک‌ها و مشخصات مربوط به سرویس‌های تعریف‌شده به ترتیب در شکل‌های (۵-۱۱)، (۵-۱۲) و جدول (۵-۱۰) نشان داده‌شده‌اند.



شکل ۵-۱۱: بار اولیه‌ی ابرک‌های اروپا

[1] Identical



شکل ۵-۱۲: بار اولیه‌ی ابرک‌های آمریکا

جدول ۵-۱۰: سرویس‌های تعریف‌شده برای شبیه‌سازی

سرویس	درصد اشغالی توسط سرویس
S_1	۱۰
S_2	۱۵
S_3	۲۰
S_4	۳۰
S_5	۳۰

نتایج شبیه‌سازی‌ها که شامل مسیر اصلی، مسیرهای جایگزین، تعداد مسیرهای پیداشده‌ی موجه هستند در جداول (۵-۱۱) و (۵-۱۲) آورده شده است. ما در پروسه‌ی مسیریابی تأخیر را به‌عنوان معیار اصلی در نظر گرفته‌ایم و به همین منظور کوتاه‌ترین مسیر به‌عنوان مسیر اصلی انتخاب‌شده است ولی معیارهای دیگر نیز می‌توانند در پروسه‌ی انتخاب موردنظر قرارگرفته شوند.

جدول ۵-۱۱: نتایج مسیریابی - مسیر اصلی

جریان	مسیر اصلی	تأخیر (میلی‌ثانیه)	پهنای باند (مگابیت بر ثانیه)	تعداد مسیرهای پیداشده
۱	۶،۹،۱۰،۱۱،۷	۱۱	۵۴	۲۳
۲	۶،۹،۱۰،۴،۸،۷	۲۰۶	۱۰۸	۲
۳	۶،۱،۳،۴،۱۰،۱۱،۷	۱۹۵	۵۴	۱۲
۴	۶،۹،۱۰،۴،۵،۲،۸،۷	۲۶۷	۱۰۸	۱
۵	۱،۵،۶،۸،۹،۱۰،۱۲،۱۷،۲۰،۱۸،۲۳،۲۷	۳۳۶	۵۶	۱۲
۶	۱،۲،۳،۴،۷،۱۰،۱۲،۱۷،۲۰،۱۸،۲۳،۲۷	۴۶۶	۱۰۵	۱
۷	۱،۵،۶،۸،۹،۱۰،۱۲،۱۷،۲۰،۱۸،۲۲،۲۷	۳۳۶	۵۶	۹
۸	۱،۲،۳،۴،۷،۱۰،۱۲،۱۷،۲۰،۱۸،۲۳،۲۷	۴۶۶	۱۰۵	۱

جدول ۵-۱۲: نتایج مسیریابی - مسیرهای جایگزین

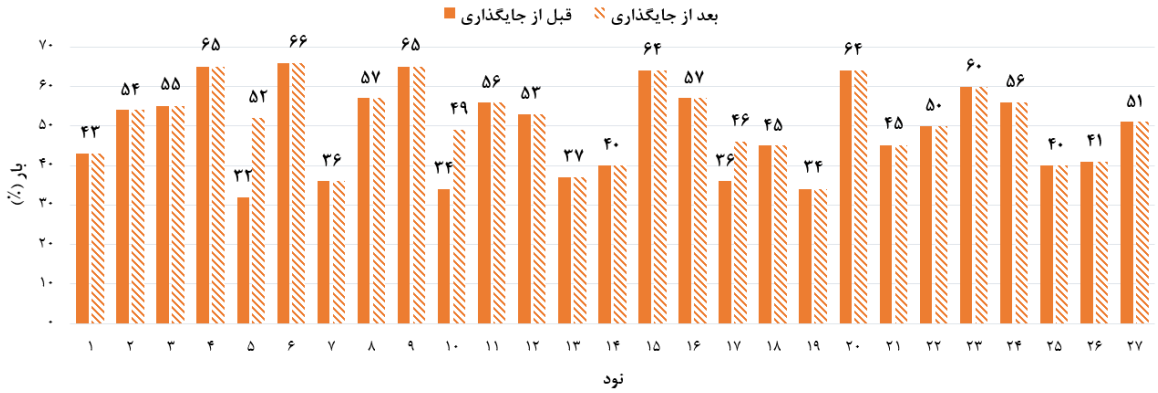
جریان	مسیرهای جایگزین	تأخیر(میلی‌ثانیه)	پهنای‌باند(مگابیت‌برثانیه)
۱	۶،۱،۵،۲،۸،۷	۱۸۲	۵۴
	۶،۱،۳،۴،۸،۷	۱۹۶	۵۴
۲	۶،۹،۱۰،۴،۵،۲،۸،۷	۲۶۷	۱۰۸
۳	۶،۹،۳،۴،۸،۷	۱۹۶	۶۷
	۶،۱،۵،۴،۸،۷	۱۹۹	۵۴
۴	مسیری یافت نشد	—	—
۵	۱،۲،۳،۶،۸،۱۱،۱۳،۱۹،۲۴،۲۵،۲۶،۲۲،۲۷	۳۹۹	۵۱
	۱،۲،۳،۴،۷،۱۰،۱۲،۱۱،۱۳،۱۹،۲۴،۲۵،۲۶،۲۲،۲۷	۵۲۱	۵۶
۶	مسیری یافت نشد	—	—
۷	۱،۲،۳،۶،۸،۱۱،۱۳،۱۹،۲۴،۲۵،۲۶،۲۲،۲۷	۳۹۹	۵۱
	۱،۵،۶،۸،۱۱،۱۳،۱۹،۲۴،۲۵،۲۶،۲۲،۲۷	۳۵۵	۵۶
۸	مسیری یافت نشد	—	—

نتایج مربوط به جانمایی سرویس برای هریک از جریان‌ها نیز در جدول (۵-۱۳) آورده شده است و شکل‌های (۵-۱۳) و (۵-۱۴) بار ابرک‌ها پس از جانمایی سرویس را برای جریان‌های ۴ و ۵ که بیش‌ترین شاخص برابری را به‌دست‌آمده آورده‌اند، نشان می‌دهد.

جدول ۵-۱۳: نتایج جانمایی سرویس

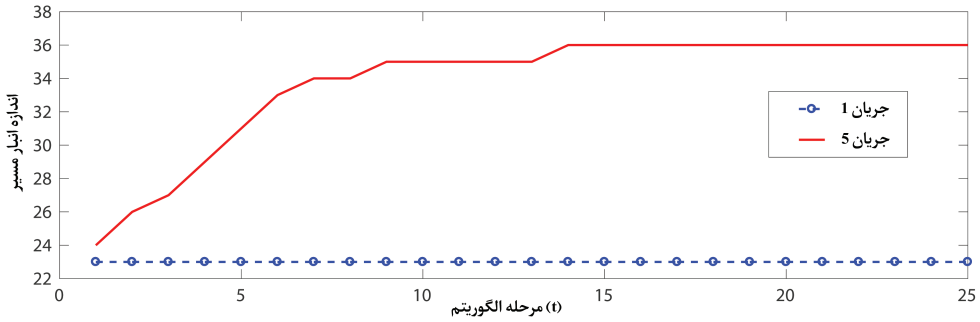
جریان	سرویس‌ها	بهترین جانمایی	شاخص‌برابری (درصد)
۱	S_1, S_2, S_3	۷،۱۱،۹	۹۵.۱۰۷۵
۲	S_1, S_2, S_4	۶،۹،۴	۹۵.۹۱۲۸
۳	S_1, S_4, S_5	۱،۴،۳	۹۶.۵۰۹۲
۴	S_2, S_4, S_5	۱۰،۵،۴	۹۷.۶۶۳۷
۵	S_1, S_2, S_3	۱۷،۱۰،۵	۹۶.۷۰۶۸
۶	S_2, S_4, S_5	۱۷،۷،۱۰	۹۶.۲۶۲۲
۷	S_1, S_4, S_5	۱۷،۱۰،۵	۹۶.۴۹۴۵
۸	S_1, S_2, S_4	۱۷،۷،۱۰	۹۹۶.۳۷۰۸

شکل ۵-۱۳: جانمایی سرویس برای جریان چهارم



شکل ۵-۱۴: جانمایی سرویس برای جریان پنجم

شکل (۵-۱۵) تغییر اندازه‌ی انبار مسیر برای جریان‌های ۱ و ۵ را در طول مراحل مختلف اجرای الگوریتم GWO نشان می‌دهد. همان‌طور که مشاهده می‌شود، اندازه انبار مسیرها تنها برای جریان ۵ در طول زمان تغییر کرده است که دلیل آن اندازه‌ی بزرگ‌تر توپولوژی در این جریان نسبت به جریان ۱ است؛ بنابراین، می‌توان به این نتیجه رسید که استفاده از الگوریتم دومی برای بهینه کردن توان پارامترهای موجود در تابع انتخاب در الگوریتم ACS تنها در صورتی کارآمد بوده و منجر به افزایش عملکرد می‌شود که اندازه‌ی مسئله به‌اندازه‌ی کافی بزرگ باشد.

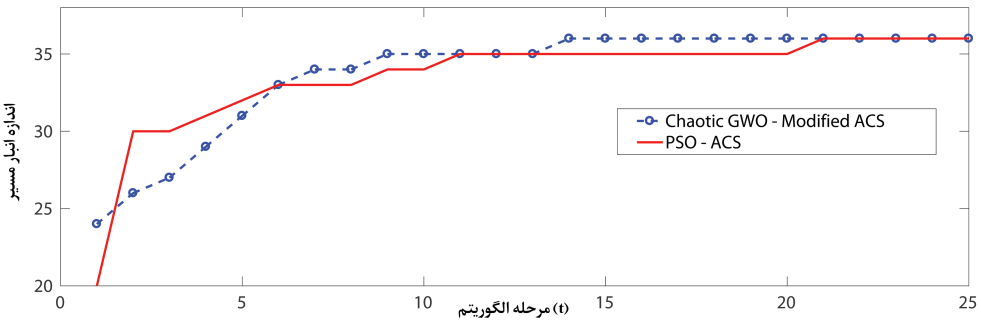


شکل ۵-۱۵: تغییر اندازه‌ی انبار مسیرها

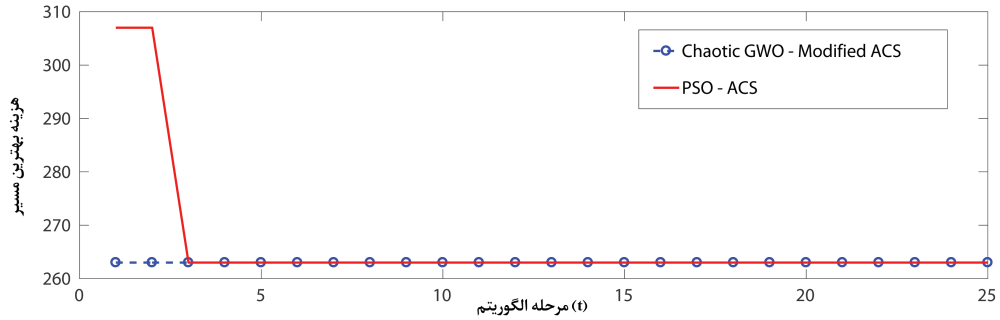
۵-۲-۳ مقایسه

برای مشخص‌شده تأثیر استفاده از الگوریتم GWO و اضافه کردن پارامتر انبار به الگوریتم ACS، در این چارچوب نرم‌افزاری پیشنهادی‌مان را با یک چارچوب نرم‌افزاری فرضی و مشابه که از الگوریتم‌های PSO و ACS سنتی استفاده می‌کند، مقایسه می‌کنیم. توضیحات مربوط به الگوریتم PSO در بخش قبلی از همین فصل آورده شده و پارامترهای آن با استفاده از جدول (۵-۶) مقداردهی شده است.

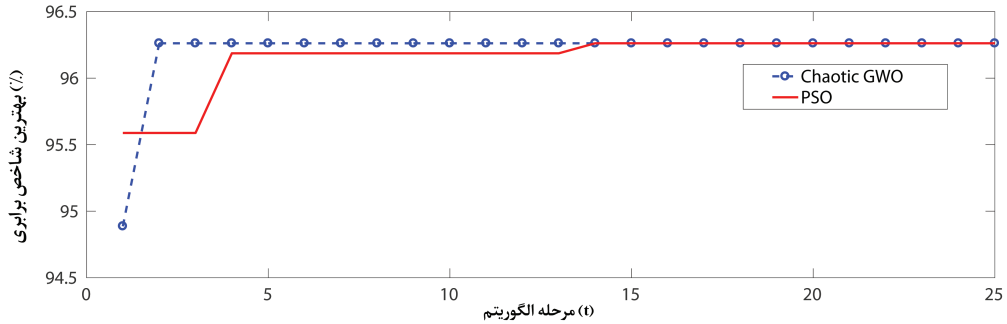
شکل‌های (۵-۱۶)، (۵-۱۷) و (۵-۱۸) به ترتیب تغییرات مربوط به‌اندازه‌ی انبار مسیر، هزینه‌ی بهترین مسیر و شاخص برابری را در طول زمان برای هر دو چارچوب نرم‌افزاری نشان می‌دهد و همان‌طور که مشاهده می‌شود الگوریتم پیشنهادی ما به سرعتی بیشتری همگرا می‌شود؛ بنابراین، استفاده از الگوریتم گرگ‌های خاکستری آشوبناک و نسخه‌ی بهبودیافته‌ی الگوریتم ACS برای توسعه‌ی سیستم‌ها می‌تواند منجر به کاهش زمان محاسبات و افزایش عملکرد شود.



شکل ۵-۱۶: همگرایی اندازه‌ی انبار مسیرها



شکل ۵-۱۷: همگرایی هزینه‌ی بهترین مسیر



شکل ۵-۱۸: همگرایی شاخص برابری

۵-۳ جمع‌بندی

در این فصل به‌منظور اثبات درستی و کارآمد بودن چارچوب‌های نرم‌افزاری ارائه‌شده، آن‌ها را برای حالت‌های مختلف شبیه‌سازی کردیم و در ادامه نتایج به‌دست‌آمده را با نتایج چارچوب‌های نرم‌افزاری مشابه مقایسه نمودیم. نتایج به‌دست‌آمده نشان می‌دهد که الگوریتم‌های استفاده‌شده در چارچوب‌های پیشنهادی ما رفتار بسیار مناسب‌تری نسبت به الگوریتم‌های دیگر نشان می‌دهد.

فصل ششم

جمع‌بندی، نتیجه‌گیری و پیشنهادات

پیدایش فنّاوری‌های مبتنی بر ابر و پیشرفت‌های صورت گرفته در عرصه‌ی اینترنت اشیاء منجر به افزایش بیش‌ازحد ترافیک شبکه‌ها بخصوص شبکه‌های موبایل شده است. به همین خاطر، نسل فعلی شبکه‌های موبایل قادر به پاسخگویی نیازهای موجود نیست و می‌بایست نسل جدیدی برای شبکه‌های موبایل در سال‌های آتی ارائه گردد. در این پژوهش ابتدا در مورد معماری نسل پنجم موبایل و فناوری‌های جدیدی مثل رایانش ابری، مجازی‌سازی توابع شبکه و شبکه‌های تعریف‌شده با نرم‌افزار که به برآورده کردن نیازمندی‌ها و اهداف نسل پنجم موبایل کمک می‌کند بحث کردیم و در ادامه با بررسی معماری شبکه‌های موبایل تعریف‌شده با نرم‌افزار، برای دو مسئله‌ی اساسی برای تخصیص منابع در این شبکه‌ها دو چارچوب نرم‌افزاری ارائه نمودیم.

چارچوب نرم‌افزاری اول با استفاده از دولایه از الگوریتم فرا ابتکاری گرگ‌های خاکستری تعداد بهینه کنترلرها در این شبکه‌ها را به‌صورت پویا محاسبه می‌نمود و اتصالات مناسب بین کنترلرهای فراهم‌شده و سوئیچ‌های موجود در شبکه را پیدا می‌کرد. در این بخش، به‌منظور بهبود عملکرد این الگوریتم، ازسری‌های آشوبناک نیز استفاده کردیم که نتایج به‌دست‌آمده در فصل پنجم نشان داد که این امر موجب افزایش دقت و همگرایی الگوریتم پیشنهادی ما در مقایسه با بقیه‌ی الگوریتم‌ها می‌شود.

در بخش دوم از این پایان‌نامه، چارچوب نرم‌افزاری برای مسیریابی جریان‌ها و جانمایی سرویس‌های شبکه ارائه نمودیم. این چارچوب نرم‌افزاری علی‌رغم اغلب الگوریتم‌های موجود برای مسیریابی که همیشه کوتاه‌ترین مسیر را به‌عنوان جواب برمی‌گزینند، انعطاف‌پذیری ما را در انتخاب مسیر افزایش می‌دهد و می‌تواند معیارهای مختلف را در انتخاب مسیر مدنظر قرار دهد. علاوه بر این، چارچوب نرم‌افزاری پیشنهادی ما به‌جز مسیر اصلی، دو مسیر جایگزین نیز ارائه می‌کند که ضریب اطمینان و برگشت‌پذیری شبکه را افزایش می‌دهد. در این بخش از ترکیب دو الگوریتم ACS و GWO استفاده نمودیم و برای افزایش عملکرد آن‌ها ازسری‌های آشوبناک و دانش‌محور نمودن الگوریتم ACS استفاده نمودیم که نتایج شبیه‌سازی کارآمد بودن این روش‌ها در مقایسه با الگوریتم‌های دیگر را اثبات نمود.

در ضمن لازم به ذکر است که چارچوب نرم‌افزاری پیشنهادشده در بخش اول این پایان‌نامه منجر به چاپ مقالات [53] و [77] شده است.

۶-۱ پیشنهادات

برای ادامه تحقیقات انجام‌شده در این پژوهش و بهبود عملکرد آن‌ها چند پیشنهاد به شرح زیر ارائه می‌شود:

- استفاده از مدل‌های صف دقیق‌تر برای مدل‌سازی رفتار زمانی کنترلرهای SDN
- استفاده از الگوریتم‌های بهینه‌سازی چند هدفه[1] برای حل مسائل مطرح‌شده

[1] Multi-Objective

- پیاده‌سازی الگوریتم‌های پیشنهادشده در این پژوهش با استفاده از کنترلرهای واقعی مثل Floodlight و ONOS و استفاده از شبیه‌سازهای شبکه نظیر Mininet برای بررسی دقیق‌تر و مقایسه‌ی آن‌ها با الگوریتم‌های دیگر

پیوست – شبه کد الگوریتم‌ها

در این بخش شبه کد الگوریتم‌های تعریف‌شده در فصل‌های گذشته آورده شده است.

الگوریتم ۱ GWO برای فراهم‌سازی کنترلرها

Set the number of agents(n) and maximum number of iterations(MaxIt)

Initialize Grey Wolf agents using Circle Map function (Eq.14-3)

$Agent_i$ *represents Number of Controllers in* Φ_i *or ID of a controller which is connected to* S_k

Calculate Utilization of each class based on the Agent's position using Eq. (12-3)

Find Alpha Wolf (α) $=$ *The best agent*

Find Beta Wolf (β) $=$ *The second best agent*

Find Delta Wolf (δ) $=$ *The third best agent*

while $t <$ *Maximum number of iteration* **do**

Calculate encircling coefficient for updating agents: $a = 2 - 2 \times (\frac{t}{MaxIt})$

for $i = 1, 2, ..., n$ **do**

Generate random variables using Eq.(14-3): r_1 *and* r_2

Calculate algorithm coefficients for updating agents: A and C using Eq.(17-3, 18-3)

Update Alpha, Beta, and Delta using Eq. (15-3,16-3)

Update agents(ω_i) *using Eq.(19-3)*

end for

Calculate Utilization of each class based on the Agent's position using Eq. (12-3)

Find Alpha, Beta, Delta Wolves

$t \leftarrow t + 1$

end while

$Return$ *the best solution*

الگوریتم ۲ الگوریتم دانش‌محور ACS

Get α, β, and γ from GWO algorithm
Get the source and the destination nodes
Set the number of ants(n) and the number of iteration(MaxIt)
Initialize the iteration counter: $t = 0$
Initialize ACS parameters: $\rho_1, \rho_2, r_0, \tau_0, n_G$
Initialize the best ant: $\hat{x}(t) = \emptyset$
for *each link (i,j)* **do**
    Initialize Phermone matrix:
    $\tau_{i,j} = \tau_0 \times$ *random number between 0 and 1*
end for
while $t <$ *Max Number of Iteration* **do**
    for *each ant* $k = 1, 2, ..., n$ **do**
        Place the ant on the source node: $x_k(t) =$ *The source node*
        repeat
            Generate a random number(r)
            Select the next node(j) using Eq. (10-4) and (11-4)
            $x_k(t) = x_k(t) \cup j$
        until *the* k^{th} *ant get to the destination:* $j ==$ *The destination node*
        Calculate the path cost: $f(x_k(t))$
        Update phermones using Eq. (14-4)
        Update the repository
    end for
    Find the best ant and set $\hat{x}(t)$
    Update phermones of the best path using Eq. (12-4)
    $t \leftarrow t + 1$
end while
Return $\hat{x}(t)$

الگوریتم ۳ GWO برای مسیریابی

Set the number of agents(n) and maximum number of iterations(MaxIt)
Initialize Grey Wolf agents using Circle Map function (Eq.14-3)
$Agent_i$ *represents the exponents of ACS probability function (Eq.11-4)*
Initialize the path repository for saving the paths calculated by ACS instances
Call an ACS instance for each agent and wait for getting the cost
Find Alpha Wolf (α) = *The best agent*
Find Beta Wolf (β) = *The second best agent*
Find Delta Wolf (δ) = *The third best agent*
while $t <$ *Maximum number of iteration* **do**
    Calculate encircling coefficient for updating agents: $a = 2 - 2 \times (\frac{t}{MaxIt})$
    for $i = 1, 2, ..., n$ **do**
        Generate random variables using Eq.(14-3): r_1 *and* r_2
        Calculate algorithm coefficients for updating agents: A *and* C *using Eq.(17-3, 18-3)*
        Update Alpha, Beta, and Delta using Eq. (15-3,16-3)
        Update agents(ω_i) *using Eq.(19-3)*
    end for
    Call an ACS instance for each agent and wait for getting the cost
    Find Alpha, Beta, Delta Wolves
    $t \leftarrow t + 1$
end while
$Return$ *the path repository*

الگوریتم ۴ GWO برای جایگذاری سرویس‌ها

Set the number of agents(n) and maximum number of iterations(MaxIt)
Get the latest state of cloudlet loads
Initialize Grey Wolf agents using Circle Map function (Eq.14-3)
Map the initialized agents to the switches of the path
The k^{th} element of $Agent_i$ represents the switch on which the k^{th} service is going to run
if *The load of any cloudlet is more than 80%* **then**
    Set the cost of the agent to $-\infty$
else
    Calculate the cost of agents using Eq.(7-4)
end if
Find Alpha Wolf (α) = *The best agent*
Find Beta Wolf (β) = *The second best agent*
Find Delta Wolf (δ) = *The third best agent*
while $t <$ *Maximum number of iteration* **do**
    Calculate encircling coefficient for updating agents: $a = 2 - 2 \times (\frac{t}{MaxIt})$
    for $i = 1, 2, ..., n$ **do**
        *Generate random variables using Eq.(**??**):* r_1 *and* r_2
        Calculate algorithm coefficients for updating agents: A and C using Eq.(17-3, 18-3)
        Update Alpha, Beta, and Delta using Eq. (15-3,16-3)
        Update agents(ω_i) *using Eq.(19-3)*
    end for
    if *The load of any cloudlet is more than 80%* **then**
        Set the cost of the agent to $-\infty$
    else
        Calculate the cost of agents using Eq.(7-4)
    end if
    Find Alpha, Beta, Delta Wolves
    $t \leftarrow t + 1$
end while
Return α *which is the best service placement for the input path and flow required services*

منابع و مراجع

[1] Rep, Tech. Visual networking index report: global mobile data traffic forecast update, 20132018. tech. rep., Cisco, 2014.

[2] Peng, M., Li, Y., Zhao, Z., and Wang, C. System architecture and key technologies for 5g heterogeneous cloud radio access networks. *IEEE Network*, 29(2):6–14, March 2015.

[3] Andrews, J. G., Buzzi, S., Choi, W., Hanly, S. V., Lozano, A., Soong, A. C. K., and Zhang, J. C. What will 5g be? *IEEE Journal on Selected Areas in Communications*, 32(6):1065–1082, June 2014.

[4] Chen, T., Matinmikko, M., Chen, X., Zhou, X., and Ahokangas, P. Software defined mobile networks: concept, survey, and research directions. *IEEE Communications Magazine*, 53(11):126–133, November 2015.

[5] Liang, C. and Yu, F. R. Wireless network virtualization: A survey, some research issues and challenges. *IEEE Communications Surveys Tutorials*, 17(1):358–380, Firstquarter 2015.

[6] Gomes, Andre S, Sousa, Bruno, Palma, David, Fonseca, Vitor, Zhao, Zhongliang, Monteiro, Edmundo, Braun, Torsten, Simoes, Paulo, and Cordeiro, Luis. Edge caching with mobility prediction in virtualized lte mobile networks. *Future Generation Computer Systems*, 70:148–162, 2017.

[7] Jararweh, Yaser, Al-Ayyoub, Mahmoud, Benkhelifa, Elhadj, Vouk, Mladen, Rindos, Andy, et al. Software defined cloud: Survey, system and evaluation. *Future Generation Computer Systems*, 58:56–74, 2016.

[8] Akyildiz, Ian F, Lin, Shih-Chun, and Wang, Pu. Wireless software-defined networks (w-sdns) and network function virtualization (nfv) for 5g cellular systems: An overview and qualitative evaluation. *Computer Networks*, 93:66–79, 2015.

[9] Wubben, Dirk, Rost, Peter, Bartelt, Jens Steven, Lalam, Massinissa, Savin, Valentin, Gorgoglione, Matteo, Dekorsy, Armin, and Fettweis, Gerhard. Benefits and impact of cloud computing on 5g signal processing: Flexible centralization through cloud-ran. *IEEE signal processing magazine*, 31(6):35–44, 2014.

[10] Barbarossa, Sergio, Sardellitti, Stefania, and Di Lorenzo, Paolo. Communicating while computing: Distributed mobile cloud computing over 5g heterogeneous networks. *IEEE Signal Processing Magazine*, 31(6):45–55, 2014.

[11] Aissioui, Abdelkader, Ksentini, Adlen, Gueroui, Abdelhak Mourad, and Taleb, Tarik. Toward elastic distributed sdn/nfv controller for 5g mobile cloud management systems. *IEEE Access*, 3:2055–2064, 2015.

[12] Mobile, China. C-ran: the road towards green ran. *White Paper, ver*, 2, 2011.

[13] Xiong, Gang, Hu, Yu-xiang, Tian, Le, Lan, Ju-long, Li, Jun-fei, and Zhou, Qiao. A virtual service placement approach based on improved quantum genetic algorithm. *Frontiers of Information Technology & Electronic Engineering*, 17(7):661–671, 2016.

[14] Sherry, Justine, Hasan, Shaddi, Scott, Colin, Krishnamurthy, Arvind, Ratnasamy, Sylvia, and Sekar, Vyas. Making middleboxes someone else's problem: network processing as a cloud service. *ACM SIGCOMM Computer Communication Review*, 42(4):13–24, 2012.

[15] Ding, Wanfu, Qi, Wen, Wang, Jianping, and Chen, Biao. Openscaas: an open service chain as a service platform toward the integration of sdn and nfv. *IEEE Network*, 29(3):30–35, 2015.

[16] Bhamare, Deval, Jain, Raj, Samaka, Mohammed, and Erbad, Aiman. A survey on service function chaining. *Journal of Network and Computer Applications*, 75:138–155, 2016.

[17] Kreutz, Diego, Ramos, Fernando MV, Verissimo, Paulo Esteves, Rothenberg, Christian Esteve, Azodolmolky, Siamak, and Uhlig, Steve. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[18] Bernardos, Carlos J, De La Oliva, Antonio, Serrano, Pablo, Banchs, Albert, Contreras, Luis M, Jin, Hao, and Zúñiga, Juan Carlos. An architecture for software defined wireless networking. *IEEE wireless communications*, 21(3):52–61, 2014.

[19] *Open networking foundation, openflow switch specification, v1.3.4, March, 27, 2014.*, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.1.pdf.

[20] *Ciscos one platform kit (onepk),* http://www.cisco.com/en/US/prod/iosswrel/onepk.html.

[21] Richardson, Leonard and Ruby, Sam. *RESTful web services*. " O'Reilly Media, Inc.", 2008.

[22] Greenberg, Albert, Hamilton, James R, Jain, Navendu, Kandula, Srikanth, Kim, Changhoon, Lahiri, Parantap, Maltz, David A, Patel, Parveen, and Sengupta, Sudipta. Vl2: a scalable and flexible data center network. in *ACM SIGCOMM computer communication review*, vol. 39, pp. 51–62. ACM, 2009.

[23] Al-Fares, Mohammad, Loukissas, Alexander, and Vahdat, Amin. A scalable, commodity data center network architecture. in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74. ACM, 2008.

[24] Bari, Md Faizul, Roy, Arup Raton, Chowdhury, Shihabur Rahman, Zhang, Qi, Zhani, Mohamed Faten, Ahmed, Reaz, and Boutaba, Raouf. Dynamic controller provisioning in software defined networks. in *Network and Service Management (CNSM), 2013 9th International Conference on*, pp. 18–25. IEEE, 2013.

[25] Hollinghurst, Joseph, Ganesh, Ayalvadi, and Baugé, Timothy. Controller placement methods analysis. in *Information Communication and Management (ICICM), International Conference on*, pp. 239–244. IEEE, 2016.

[26] Guo, Zehua, Su, Mu, Xu, Yang, Duan, Zhemin, Wang, Luo, Hui, Shufeng, and Chao, H Jonathan. Improving the performance of load balancing in software-defined networks through load variance-based synchronization. *Computer Networks*, 68:95–109, 2014.

[27] Hassas Yeganeh, Soheil and Ganjali, Yashar. Kandoo: a framework for efficient and scalable offloading of control applications. in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 19–24. ACM, 2012.

[28] Yeganeh, Soheil Hassas and Ganjali, Yashar. Beehive: Towards a simple abstraction for scalable software-defined networking. in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, p. 13. ACM, 2014.

[29] Tootoonchian, Amin and Ganjali, Yashar. Hyperflow: A distributed control plane for openflow. in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pp. 3–3, 2010.

[30] Berde, Pankaj, Gerola, Matteo, Hart, Jonathan, Higuchi, Yuta, Kobayashi, Masayoshi, Koide, Toshio, Lantz, Bob, O'Connor, Brian, Radoslavov, Pavlin, Snow, William, et al. Onos: towards an open, distributed sdn os. in *Proceedings of the third workshop on Hot topics in software defined networking*, pp. 1–6. ACM, 2014.

[31] Koponen, Teemu, Casado, Martin, Gude, Natasha, Stribling, Jeremy, Poutievski, Leon, Zhu, Min, Ramanathan, Rajiv, Iwata, Yuichiro, Inoue, Hiroaki, Hama, Takayuki, et al. Onix: A distributed control platform for large-scale production networks. in *OSDI*, vol. 10, pp. 1–6, 2010.

[32] Heller, Brandon, Sherwood, Rob, and McKeown, Nick. The controller placement problem. in *Proceedings of the first workshop on Hot topics in software defined networks*, pp. 7–12. ACM, 2012.

[33] Rath, Hemant Kumar, Revoori, Vishvesh, Nadaf, SM, and Simha, Anantha. Optimal controller placement in software defined networks (sdn) using a non-zero-sum game. in *A World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2014 IEEE 15th International Symposium on*, pp. 1–6. IEEE, 2014.

[34] Cimorelli, Federico, Priscoli, Francesco Delli, Pietrabissa, Antonio, Celsi, Lorenzo Ricciardi, Suraci, Vincenzo, and Zuccaro, Letterio. A distributed load balancing algorithm for the control plane in software defined networking. in *Control and Automation (MED), 2016 24th Mediterranean Conference on*, pp. 1033–1040. IEEE, 2016.

[35] Xiao, Peng, Qu, Wenyu, Qi, Heng, Li, Zhiyang, and Xu, Yujie. The sdn controller placement problem for wan. in *Communications in China (ICCC), 2014 IEEE/CIC International Conference on*, pp. 220–224. IEEE, 2014.

[36] Sallahi, Afrim and St-Hilaire, Marc. Optimal model for the controller placement problem in software defined networks. *IEEE Communications Letters*, 19(1):30–33, 2015.

[37] Gao, Chuangen, Wang, Hua, Zhu, Fangjin, Zhai, Linbo, and Yi, Shanwen. A particle swarm optimization algorithm for controller placement problem in software defined network. in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 44–54. Springer, 2015.

[38] Liu, Shuai, Wang, Hua, Yi, Shanwen, and Zhu, Fangjin. Ncpso: a solution of the controller placement problem in software defined networks. in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 213–225. Springer, 2015.

[39] Tanenbaum, Andrew S and Wetherall, David J. *Computer networks*. Pearson, 2011.

[40] Braden, Robert, Clark, David, and Shenker, Scott. Integrated services in the internet architecture: an overview. tech. rep., 1994.

[41] Blake, Steven, Black, David, Carlson, Mark, Davies, Elwyn, Wang, Zheng, and Weiss, Walter. An architecture for differentiated services. tech. rep., 1998.

[42] Karakus, Murat and Durresi, Arjan. Quality of service (qos) in software defined networking (sdn): A survey. *Journal of Network and Computer Applications*, 2016.

[43] Guck, Jochen W and Kellerer, Wolfgang. Achieving end-to-end real-time quality of service with software defined networking. in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*, pp. 70–76. IEEE, 2014.

[44] Al-Sadi, Ameer Mosa, Al-Sherbaz, Ali, Xue, James, and Turner, Scott. Routing algorithm optimization for software defined network wan. in *Multidisciplinary*

in IT and Communication Science and Applications (AIC-MITCSA), Al-Sadeq International Conference on, pp. 1–6. IEEE, 2016.

[45] Egilmez, Hilmi E, Dane, S Tahsin, Bagci, K Tolga, and Tekalp, A Murat. Open-qos: An openflow controller design for multimedia delivery with end-to-end quality of service over software-defined networks. in *Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pp. 1–8. IEEE, 2012.

[46] Egilmez, Hilmi E, Civanlar, Seyhan, and Tekalp, A Murat. An optimization framework for qos-enabled adaptive video streaming over openflow networks. *IEEE Transactions on Multimedia*, 15(3):710–715, 2013.

[47] Wang, YanLing, Yuan, KunJie, Fang, Wang, Liu, YuHuai, and Jun, Ma. Research of a sdn traffic scheduling technology based on ant colony algorithm. *DEStech Transactions on Engineering and Technology Research*, (iect), 2016.

[48] Cheng, Guozhen, Chen, Hongchang, Hu, Hongchao, Wang, Zhiming, and Lan, Julong. Enabling network function combination via service chain instantiation. *Computer Networks*, 92:396–407, 2015.

[49] Basta, Arsany, Kellerer, Wolfgang, Hoffmann, Marco, Morper, Hans Jochen, and Hoffmann, Klaus. Applying nfv and sdn to lte mobile core gateways, the functions placement problem. in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*, pp. 33–38. ACM, 2014.

[50] Mohammadkhan, Ali, Ghapani, Sheida, Liu, Guyue, Zhang, Wei, Ramakrishnan, KK, and Wood, Timothy. Virtual function placement and traffic steering in flexible and dynamic software defined networks. in *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*, pp. 1–6. IEEE, 2015.

[51] Gross, Donald. *Fundamentals of queueing theory*. John Wiley & Sons, 2008.

[52] Barbeau, Michel and Kranakis, Evangelos. *Principles of ad-hoc networking*. John Wiley & Sons, 2007.

[53] Farshin, Alireza and Sharifian, Saeed. Map-sdn: a metaheuristic assignment and provisioning sdn framework for cloud datacenters. *The Journal of Supercomputing*, pp. 1–25, 2017.

[54] Jain, Raj, Chiu, Dah-Ming, and Hawe, William R. *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*, vol. 38. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984.

[55] Kirkpatrick, Scott, Gelatt, C Daniel, Vecchi, Mario P, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[56] Bonabeau, Eric, Dorigo, Marco, and Theraulaz, Guy. *Swarm intelligence: from natural to artificial systems*. no. 1. Oxford university press, 1999.

[57] Kennedy, James. Particle swarm optimization. in *Encyclopedia of machine learning*, pp. 760–766. Springer, 2011.

[58] Mirjalili, Seyedali, Mirjalili, Seyed Mohammad, and Lewis, Andrew. Grey wolf optimizer. *Advances in Engineering Software*, 69:46–61, 2014.

[59] Sheikholeslami, R and Kaveh, A. A survey of chaos embedded meta-heuristic algorithms. *Int J Optim Civil Eng*, 3(4):617–33, 2013.

[60] Zheng, Wei-Mou. Kneading plane of the circle map. *Chaos, Solitons & Fractals*, 4(7):1221–1233, 1994.

[61] Zhu, Hao, Liao, Xiangke, de Laat, Cees, and Grosso, Paola. Joint flow routing-scheduling for energy efficient software defined data center networks: A prototype of energy-aware network management platform. *Journal of Network and Computer Applications*, 63:110–124, 2016.

[62] Pan, Qingfeng and Zheng, Xianghan. Multi-path sdn route selection subject to multi-constraints. 2015.

[63] Abe, John Olorunfemi, Mantar, Haci Ali, and Yayimli, Aysegul Gencata. k-maximally disjoint path routing algorithms for sdn. in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2015 International Conference on*, pp. 499–508. IEEE, 2015.

[64] Mijumbi, Rashid, Serrat, Joan, Gorricho, Juan-luis, Latre, Steven, Charalambides, Marinos, and Lopez, Diego. Management and orchestration challenges in network functions virtualization. *IEEE Communications Magazine*, 54(1):98–105, 2016.

[65] Campolo, Claudia, Molinaro, Antonella, and Scopigno, Riccardo. Vehicular ad hoc networks. *Standards, Solutions, and Research*, 2015.

[66] *OpenDaylight Platform,* https://www.opendaylight.org/.

[67] *Floodlight Project,* http://www.projectfloodlight.org/floodlight/.

[68] *ONOS - A new carrier-grade SDN network operating system designed for high availability, performance, and scaleout,* http://onosproject.org/.

[69] Dorigo, Marco, Birattari, Mauro, and Stutzle, Thomas. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[70] Dorigo, Marco and Gambardella, Luca Maria. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, 1(1):53–66, 1997.

[71] Engelbrecht, Andries P. *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[72] Curtis, Andrew R, Mogul, Jeffrey C, Tourrilhes, Jean, Yalagandula, Praveen, Sharma, Puneet, and Banerjee, Sujata. Devoflow: Scaling flow management for high-performance networks. *ACM SIGCOMM Computer Communication Review*, 41(4):254–265, 2011.

[73] Cheng, Tracy Yingying, Wang, Mengqing, and Jia, Xiaohua. Qos-guaranteed controller placement in sdn. in *Global Communications Conference (GLOBECOM), 2015 IEEE*, pp. 1–6. IEEE, 2015.

[74] Tavakoli, Arsalan, Casado, Martin, Koponen, Teemu, and Shenker, Scott. Applying nox to the datacenter. in *HotNets*, 2009.

[75] Abedifar, Vahid, Eshghi, Mohammad, Mirjalili, Seyedali, and Mirjalili, S Mohammad. An optimized virtual network mapping using pso in cloud computing.

in *Electrical Engineering (ICEE), 2013 21st Iranian Conference on*, pp. 1–6. IEEE, 2013.

[76] Gomes, Rafael L, Bittencourt, Luiz F, Madeira, Edmundo RM, Cerqueira, Eduardo, and Gerla, Mario. Bandwidth-aware allocation of resilient virtual software defined networks. *Computer Networks*, 100:179–194, 2016.

[77] Farshin, Alireza and Sharifian, Saeed. A chaotic grey wolf controller allocator for software defined mobile network (sdmn) for 5th generation of cloud-based cellular systems (5g). *Computer Communications*, 108:94 – 109, 2017.

واژه‌نامه‌ی فارسی به انگلیسی

آ

ابر . . . . . Cloud

ابرک . . . . . Cloudlet

اپراتورهای مجازی . . . . . MVNO

ازدحام . . . . . Congestion

استثماری . . . . . Exploitation

اکتشافی . . . . . Exploration

الگوریتم بهینه‌سازی ازدحام ذرات . . . . . PSO

الگوریتم بهینه‌سازی کلونی مورچگان . . ACO

الگوریتم تبرید شبیه‌سازی‌شده . . . . . SA

الگوریتم ژنتیک . . . . . Genetic Algorithm

الگوریتم سیستم کلونی مورچان . . . . . ACS

الگوریتم گرگ‌های خاکستری . . . . . GWO

الگوریتم‌های فرا ابتکاری . . . Metaheuristic

الگوریتم‌های حریصانه Greedy Algorithms

انبار مسیرها . . . . . Path Repository

اینترنت اشیاء . . . . . Internet of Things

ب

بازی مجموع-ناصفر . Non-zero sum Game

برابری . . . . . Fairness

بررسی عملکرد . Performance Monitoring

برنامه‌نویسی خطی اعداد صحیح . . . . . ILP

برنامه‌های شبکه . . Network Applications

پ

پردازش سیگنال‌های دیجیتال . . . . . DSP

پواسون . . . . . Poisson

پویا . . . . . Dynamic

ت

تأمین‌کنندگان ابر . . . . . Cloud Providers

تأمین‌کنندگان مجازی شبکه‌ی موبایل MVNP

ترجمه‌ی آدرس شبکه . . . . . NAT

تعادل واردراپ . . . . Wardrop Equilibruim

تفاهم‌نامه سطح کیفی خدمات . . . . . SLA

تک جوابی . . . . . Single Solution

تنظیمات . . . . . Configuration

توابع مجازی شبکه . . . . . VNF

توازن بار . . . . . Load Balancing

تئوری بازی . . . . . Game Theory

ج

جایگذاری سرویس‌ها . . Service Placement

جایگذاری کنترلرها Controller Placement

جدول جریان . . . . . Flow Table

جدول گروهی . . . . . Group Table

جریان . . . . . Flow

چ

چارچوب نرم‌افزاری . . . . . Framework

چند جوابی . . . . . Population-based

چند هدفه . . . . . Multi-Objective

ح

حد نصاب . . . . . Threshold

د

دادگان . . . . . Dataset

دانش محور . . . . . Knowledge-based

دوره‌ای . . . . . Periodic

دیوار آتش . . . . . Firewall

ر

رایانش ابری . . . . . Cloud Computing

ز

زمان‌های وقفه . . . . . Timeouts

زنجیره‌ی سرویس‌ها . . . Chain of Services

زنجیره‌ی مارکوف . . . . . Markovian Chain

س

سادگی . . . . . Simplicity

سامان‌دهی . . . . . Orchestrate

سخت‌افزار میانی . . . . . Middlebox

سری‌های آشوبناک . . . Chaotic Sequences

سلسله مراتب اجتماعی . . Social Hierarchy

سودمندی . . . . . Utilization

سیستم تشخیص ورود بی‌اجازه . . . . . IDS

ش

شاخص برابری جین Jain’s Fairness Index

شاخص تصویری شبکه . . . . . VNI

شبکه‌های تعریف‌شده با نرم‌افزار . . . . . SDN

شبکه‌های موبایل تعریف‌شده با نرم‌افزار SDMN

شبکه‌ی دسترسی به رادیو . . . . . RAN

شبکه‌ی دسترسی رادیوی ابری . . . . C-RAN

شبکه‌ی گسترده . . . . . WAN

ض

ضریب اطمینان . . . . . Reliability

ضریب برگشت‌پذیری . . . . . Resilience

ضریب همبستگی . . . . Correlation Factor

ط

طول عمر . . . . . Life Cycle

طیف دسته‌ای . . . . . Spectral Clustering

طیف رادیویی . . . . . Spectrum

ف

فراهم‌سازی . . . . . Provision

فراهم‌سازی کنترلرها . . . . . Controller Provisioning

فرومون . . . . . Pheromone

فیلدهای تطابق . . . . . Match Fields

ق

قابل دستریس . . . . . Feasible

قرار داده‌شده . . . . . Embeded

ک

کارگزار سوئیچ . . . . . Switch Agent

کارگزار کلاس . . . . . Class Agent

کانال کنترلی . . . . . Control Channel

کیفیت سرویس . . . . . QoS

ل

لایه‌ی جمع‌کننده . . . . Aggregation Layer

لایه‌ی داده . . . . . Data Plane

لایه‌ی دسترسی . . . . . Access Layer

لایه‌ی کنترل . . . . . Control Plane

لایه‌ی مدیریت . . . . . Management Plane

لایه‌ی هسته . . . . . Core Layer

م

ماشین مجازی . . . . . VM

متمرکز . . . . . Centralized

مجازی‌سازی توابع شبکه . . . . . NFV

مجازی‌سازی شبکه . . . . . Network Virtualization

مدیریت . . . . . Management

مدیریت توپولوژی . . . Topology Manager

مسیر داده . . . . . Datapath

مسیرهای جایگزین . . . . Redundant Paths

مسیریابی جریان . . . . . Flow Routing

مسئله‌ی کوله‌پشتی چندتایی . . . . . MKP

مصالحه . . . . . Tradeoff

مقیاس‌پذیری . . . . . Scalability

میزان از دست رفتن بسته‌ها . . Packet Loss

ن

نقطه‌ی بهینه‌ی محلی . . . Local Optimum

نمایی . . . . . Exponential

و

واسط برنامه‌نویسی استاندارد . . . . . API

واسط نرم‌افزاری جنوبی . Southbound API

واسط نرم‌افزاری شمالی . . Northbound API

ه

هزینه‌های اصلی . . . . . CAPEX

هزینه‌های نگه‌داری . . . . . OPEX

همگرایی . . . . . Convergence

واژه‌نامه‌ی انگلیسی به فارسی

A

Access Layer . . . . . لایه‌ی دسترسی

Aggregation Layer . . . . لایه‌ی جمع‌کننده

ACO . . الگوریتم بهینه‌سازی کلونی مورچگان

ACS . . . . . الگوریتم سیستم کلونی مورچان

API . . . . . واسط برنامه‌نویسی استاندارد

C

CAPEX . . . . . هزینه‌های اصلی

Centralized . . . . . متمرکز

Chain of Services . . . زنجیره‌ی سرویس‌ها

Chaotic Sequences . . . سری‌های آشوبناک

Class Agent . . . . . کارگزار کلاس

Cloud . . . . . ابر

Cloud Computing . . . . . رایانش ابری

Cloud Providers . . . . . تأمین‌کنندگان ابر

Cloudlet . . . . . ابرک

Configuration . . . . . تنظیمات

Congestion . . . . . ازدحام

Control Channel . . . . . کانال کنترلی

Control Plane . . . . . لایه‌ی کنترل

Controller Placement جایگذاری کنترلرها

Controller . . . . . فراهم‌سازی کنترلرها
Provisioning

Convergence . . . . . همگرایی

Cookie . . . . . کوکی

Core Layer . . . . . لایه‌ی هسته

Correlation Factor . . . . ضریب همبستگی

C-RAN . . . . شبکه‌ی دسترسی رادیوی ابری

D

Data Plane . . . . . لایه‌ی داده

Datapath . . . . . مسیر داده

Dataset . . . . . دادگان

Deterministic . . . . . مدل شبکه‌ی جبری
Network Model

DSP . . . . . پردازش سیگنال‌های دیجیتال

Dynamic . . . . . پویا

E

Embeded . . . . . قرار داده‌شده

Exploitation . . . . . استثماری

Exploration . . . . . اکتشافی

Exponential . . . . . نمایی

F

Fairness . . . . . برابری

Feasible . . . . . قابل دستریس

Firewall . . . . . دیوار آتش

Flow . . . . . جریان

Flow Routing . . . . . مسیریابی جریان

Flow Table . . . . . جدول جریان

Framework . . . . . چارچوب نرم‌افزاری

G

Game Theory . . . . . تئوری بازی

Genetic Algorithm . . . . . الگوریتم ژنتیک

Greedy Algorithms الگوریتم‌های حریصانه

GWO . . . . . الگوریتم گرگ‌های خاکستری

Group Table . . . . . جدول گروهی

I

ILP . . . . . برنامه‌نویسی خطی اعداد صحیح

Internet of Things . . . . . اینترنت اشیاء

IDS . . . . . سیستم تشخیص ورود بی‌اجازه

J

Jain's Fairness Index شاخص برابری جین

K

Knowledge-based . . . . . دانش محور

L

Life Cycle . . . . . طول عمر

Load Balancing . . . . . توازن بار

Local Optimum . . . نقطه‌ی بهینه‌ی محلی

M

Management . . . . . مدیریت

Management Plane . . . . . لایه‌ی مدیریت

Markovian Chain . . . . . زنجیره‌ی مارکوف

Match Fields . . . . . فیلدهای تطابق

Metaheuristic . . . الگوریتم‌های فرا ابتکاری

Middlebox . . . . . سخت‌افزار میانی

MVNO . . . . . اپراتورهای مجازی

MVNP تأمین‌کنندگان مجازی شبکه‌ی موبایل

MKP . . . . . مسئله‌ی کوله‌پشتی چندتایی

Multi-Objective . . . . . چند هدفه

N

NAT . . . . . ترجمه‌ی آدرس شبکه

شبکه‌ی دسترسی به رادیو . . . . . RAN

مسیرهای جایگزین . . . . Redundant Paths

ضریب اطمینان . . . . . Reliability

ضریب برگشت‌پذیری . . . . . Resilience

S

مقیاس‌پذیری . . . . . Scalability

تفاهم‌نامه سطح کیفی خدمات . . . . . SLA

جایگذاری سرویس‌ها . . Service Placement

سادگی . . . . . Simplicity

الگوریتم تبرید شبیه‌سازی‌شده . . . . . SA

تک جوابی . . . . . Single Solution

سلسله مراتب اجتماعی . . Social Hierarchy

شبکه‌های موبایل تعریف‌شده با نرم‌افزار SDMN

شبکه‌های تعریف‌شده با نرم‌افزار . . . . . SDN

واسط نرم‌افزاری جنوبی . Southbound API

طیف دسته‌ای . . . . . Spectral Clustering

طیف رادیویی . . . . . Spectrum

کارگزار سوئیچ . . . . . Switch Agent

T

حد نصاب . . . . . Threshold

زمان‌های وقفه . . . . . Timeouts

برنامه‌های شبکه . . Network Applications

مجازی‌سازی توابع شبکه . . . . . NFV

مجازی‌سازی شبکه . . . . . Network Virtualization

بازی مجموع-ناصفر . Non-zero sum Game

واسط نرم‌افزاری شمالی . . Northbound API

O

هزینه‌های نگه‌داری . . . . . OPEX

سامان‌دهی . . . . . Orchestrate

P

میزان از دست رفتن بسته‌ها . . Packet Loss

الگوریتم بهینه‌سازی ازدحام ذرات . . . . . PSO

انبار مسیرها . . . . . Path Repository

بررسی عملکرد . Performance Monitoring

دوره‌ای . . . . . Periodic

فرومون . . . . . Pheromone

پواسون . . . . . Poisson

چند جوابی . . . . . Population-based

فراهم‌سازی . . . . . Provision

Q

کیفیت سرویس . . . . . QoS

R

توابع مجازی شبکه . . . . . VNF

شاخص تصویری شبکه . . . . . VNI

W

تعادل واردراپ . . . Wardrop Equilibruim

شبکه‌ی گسترده . . . . . WAN

پهن‌ترین مسیر . . . . . Widest Path

مدیریت توپولوژی . . . Topology Manager

مصالحه . . . . . Tradeoff

U

سودمندی . . . . . Utilization

V

ماشین مجازی . . . . . VM

Abstract

There has been a significant increase in mobile data traffic in recent years and by the advent of the IoT (Internet of Things), it will continue to rise. Since the current wireless systems cannot handle this amount of traffic, the next generation of the mobile standard is planning to overcome the obstacles by using new technologies such as Cloud Computing, Software Defined Networking (SDN), and Network Function Virtualization (NFV). Cloud Computing will provide sufficient resources, required for the mobile networks, to handle the ever-increasing amount of traffic. Additionally, using SDN as well as NFV in cloud datacenters can lead to better manageability in the networks and easier development of network applications in the future. Since cloud datacenters as well as mobile networks are inherently large-scale networks, the resources must be allocated optimally so that we can reduce the networks' expenditures while satisfying the QoS requirement. Therefore, we are going to investigate resource allocation problem in this thesis from two different aspects.

In the first part, we introduce controller placement and controller provisioning problems in software-defined networks and then we will propose a metaheuristic-based framework for dynamic controller allocation for the application of the 5th generation of mobile technology (5G). Our proposed algorithm increases the convergence speed (68%) and the accuracy (1%) of the results compared to other metaheuristic algorithms like PSO.

In the second part, we are going to investigate the flow-routing problem, which is one of the most significant problems in networking, and then we will propose a metaheuristic framework for allocating main paths and redundant paths to incoming flows by using the knowledge gained by SDN controllers, which will improve the convergence speed up to 33%. In addition to allocating paths in this part, we are also going to distribute different networking functions among cloudlets which are connected to each router in the network so that we can use the resources more efficiently.

Key Words:

Software-Defined Networks, Resource Allocation, Controller Placement and Provisioning, Flow Routing, Metaheuristic Algorithms, Quality of Service, 5G.

Amirkabir University of Technology
(Tehran Polytechnic)

Electrical Engineering Department

M.Sc. Thesis

Resource allocation in Software-Defined Networks (SDN) for 5G applications

By

Alireza Farshin

Supervisor

Dr. Saeed Sharifian

July 2017