
بسم الله الرحمن الرحيم

دانشگاه پیام نور تهران

واحد شهر ری

سمینار

عنوان:

رویکرد مبتنی بر مدل برای پردازش داده ها در بستر اینترنت اشیاء IoT

استاد راهنما:

جناب آقای‌دکتر سید علی رضوی ابراهیمی

نگارش:

مجید لطفی

پاییز ۱۴۰۰

چکیده

پیشـرفت‌های اخیر در زمینه‌های مختلف، از جمله فناوری‌های حسـگر، شـبکه و پردازش داده‌ها، باعث شـده اند چشـم‌انداز اینترنت اشیا (IoT) هر روز بیشـتر و بیشـتر به واقعیت تبدیل شـود. در نتیجه این پیشـرفت‌ها، اینترنت اشیا امروزی امکان توسعه برنامه‌های کاربردی پیچیده برای محیط‌های اینترنت اشیا، مانند شهرهای هوشمند، خانه‌های هوشمند یا کارخانه‌های هوشمند را فراهم می‌کند و باعث شده با توجه به تبادل مکرر داده‌ها آنها به شکل جریان‌های داده دربیاید.

با این حجم فزاینده داده ای که به طور مداوم پردازش می شـود، چالش های متعددی چون جلوگیری از تداخل و آسـیب در فرآیند های گذرا به وجود می آید که نیازمند بررسی پردازش مبتنی بر جریان داده در محیط های IoT می باشد و از سوی دیگر با شبکه توزیع شده ناهمگون مواجه هسـتیم متشـکل از انواع سـخت افزار ها و سـنسور ها که باید بتوان اطلاعات آنها را پردازش کرد که بهترین محیط برای پردازش، محیط ابری می باشد.

اما با وجود حجم بالای اطلاعات و پردازش های طولانی، نمی توان تمامی اطلاعات در فضای ابری متمرکز پردازش نمود و پیشنهاد مناسب انجام این پردازش ها در محل نزدیک تولید اطلاعات می باشد (گره پردازشی) و ارائه مدل اجرای این ساختار بصورت جریان داده و سیستم ناهمگن و توزیع شده، هدف اصلی این پایان نامه دکتری می باشد.

کلمات کلیدی: اینترنت اشیاء ، داده ، جریان داده ، پردازش ابری ، سرویس‌دهنده ابری

فهرست مطالب

فهرست اشکال

فصل اول

آشنایی و معرفی

مقدمه

پیشـرفت‌های اخیر در زمینه‌های مختلف، از جمله فناوری‌های حسـگر، شـبکه و پردازش داده‌ها، چشـم‌انداز اینترنت اشـیا (IoT) را قادر سـاخته اسـت که هر روز بیشـتر و بیشـتر به واقعیت تبدیل شود و در نتیجه این پیشرفت‌ها، این فناوری امکان توسعه برنامه‌های کاربردی پیچیده مانند شهرها ، خانه‌ها یا کارخانه‌های هوشمند را فراهم می‌کند.

در محیط IoT با توجه به اندازه‌گیری‌های مداوم حسگر و تبادل مکرر داده‌ها بین اشیاء ، داده‌های تولید شده به شکل جریان‌های داده درآمده اند و با این حجم فزاینده داده ای که به طور مداوم پردازش می شـــود، چالش های متعددی برای پردازش کارآمد داده های اینترنت اشـــیا وجود دارد. به عنوان مثال، چگونه می توان پردازش داده های اینترنت اشـــیا را بدون تأثیر بر واکنش پذیری برنامه های آن تحقق بخشـــید. علاوه بر این، چگونه می‌توان از طریق پردازش داده‌های IoT، نیازمندی‌های مختلف عملکردی، غیرعملکردی و تعریف‌شده توسط کاربر برنامه‌ها را برآورده کرد.

در این پـایان نامه دکتری، یک رویکرد کلی جدید برای پردازش برنامه های کاربردی مبتنی بر جریان داده در محیط های IoT ارائه شـده اسـت که تمرکز آن بر قرار دادن کارآمد اپراتورهای برنامه های کاربردی در محیط های ناهمگن، توزیع شده و پویا است.

این پایان نامه دکتری توســط مدل های اطلاعاتی مختلف و تکنیک های قرار دادن اپراتور پشـــتیبانی می شـــود، به طوری کـه کـل چرخـه حیـات محیط هـای اینترنت اشـــیا و برنامه های مبتنی بر جریان داده را می توان به راحتی مدیریت کرد.

در این رویکرد، یکی از اهداف اصلی پردازش داده‌های اینترنت اشیا تا حد امکان نزدیک به منابع داده است، به طوری که زیرسـاخت‌های ابری تنها در مواردی اسـتفاده می‌شـوند که محیط‌های اینترنت اشـیا منابع پردازش کافی را برای کاربرد اینترنت اشیا ارائه نمی‌دهند.

از طریق رویکرد این پایان نامه دکتری، پردازش داده های برنامه های کاربردی اینترنت اشـــیا را می توان برای موارد اســتفاده خاص، پشــتیبانی از نیازهای خاص دامنه ها، و علاوه بر این، کاربران برنامه های اینترنت اشــیا، تنظیم کرد. پس از تعیین مکان‌های امکان‌پذیر، اپراتورهای پردازش با استفاده از استانداردهایی مانند TOSCA بر روی اشیاء IoT مربوطه مســتقر می‌شوند و برنامه اینترنت اشیا آماده و اجرا می‌شود. در نهایت، محیط اینترنت اشیا به‌منظور شناسایی و واکنش به اختلالات مؤثر بر پردازش داده‌های برنامه‌های کاربردی اینترنت اشــیا مســتقر شــده، به‌طور مداوم نظارت می‌شود.

رویکرد این پایان‌نامه دکترا توســط پلتفرم چند منظوره Binding and Provisioning (MBP) که یک پلت‌فرم منبع باز اینترنت اشیا، می باشد پشتیبانی می‌شود.

۱-۱ تعریف مسئله و سؤال

چنانچه در مقدمه بیان شــد موضــوع تبادل اطلاعات در محیط IoT نیازمند جریان داده و محیط ابری می باشــد و پردازش جریان داده نیز چالش خاص را داشــته و نیازمند سیستم های توزیع شده می باشد که ما در اینجا با توجه به تنوع دســتگاه ها، با یک سیستم ناهمگن روبرو هستیم. در این نوع سیستم ها با توجه به حجم بالای اطلاعات بهترین راهکــار برای پردازش اطلاعات، قراردادن پردازش در اپراتورهای پردازش (گره های پردازشــی) نزدیک به محل ایجاد اطلاعات و ارســال پردازش نهایی به ســیســتم ابری اســت. لذا ســؤالاتی که تا پایان این تحقیق پاسـخ داده می‌شود را این‌گونه بیان می‌کنیم.

۱- چگونه باید پردازش مناسب برای انواع جریان داده را تشخیص داد؟
۲- چگونه می توان اپراتورهای پردازش جریان داده را در محیط های ناهمگن و پویا اجرا کرد؟
۳- چگونه می توان مدل مناسب را طراحی، اجرا و کارائی این مدل را تضمین کرد؟

۱-۲ ضرورت تحقیق

ما در اینترنت اشــیاء با تنوع اشــیاء و داده های آنها مواجه هســتیم که در یک جریان داده در حال اجرا بوده و هر دســتگاه و شــیء در این شــبکه نیازمند پردازش هایی می باشد که باید پردازش داده ها به موقع و کارآمد باشد . این موضــوع نیازمند بررســی تمام جوانب و ایجاد مدل مناسب می باشد که بتواند الزامات IoT را پشتیبانی کرده و مدلی مناسب و آسان برای کاربران باشد.

۱-۳ اهداف

با توجه به موضوعات مطرح شده در مقدمه و ضروریات تحقیق می توان این اهداف را برای این پایان نامه متصور بود:

الف) پردازش به موقع و کارآمد داده ها در محیط های اینترنت اشیاء.

ب) مدل سازی مناسب و آسان برای کاربران در محیط های اینترنت اشیاء.

ج) مدل‌سازی پردازش جریان داده و تشخیص اختلالات که الزامات اینترنت اشیا را پشتیبانی کند.

د) قرار دادن کارآمد اپراتورهای پردازش بر اساس نیازها در محیط های پویا و ناهمگن اینترنت اشیاء.

ایجاد چنین مدل هایی با دو روش خودکار و دستی امکان پذیر بوده که در این پایان نامه به روش دستی که در آن خود تحلیل گر با توجه به الزامات و ساختار IoT تصمیم می گیرد پردازش ها و نوع ارتباط داده ها به چه صورت باشد.

۱-۴ جمع‌بندی

اینترنت اشیاء به زبان ساده، ارتباط حسگرها و دستگاه‌ها با شبکه‌ای است که از طریق آن می‌توانند با یکدیگر و با کاربرانشان تعامل کنند. هر کدام از این دستگاها یا اشیاء دارای داده هایی می باشند که مداوم و آنلاین در حال ساخت می باشد که خود باعث ایجاد یک جریان داده می شود. در این محیط از یک سو جریان داده و از سوی دیگر پردازش این داده ها را داریم لذا باید بدانیم این ساختار را چگونه و با چه مدلی اجرا نماییم.
این مدل باید منابع داده، سینک های داده، اپراتورهای پردازش و جریان داده را در میان آنها توصیف کند و علاوه بر این، الزامات اپراتورهای پردازش برای اشیاء IoT را شرح دهد.

فصل دوم

پیشینه و سابقه

۲-۱ اینترنت اشیاء

اصطلاح اینترنت اشیا (IoT) برای اولین بار در اواخر دهه ۹۰ ظاهر شد، با ایده اشتون که به رایانه ها اجازه می دهد همه چیز را در مورد چیزها بدانند، این ایده ابتداعا تقویت رایانه‌هایی با فناوری‌های شناسایی فرکانس رادیویی (RFID) و حسگر برای جمع‌آوری اطلاعات، مشاهده و شناسایی یک محیط بدون نیاز به کمک انسانی بود. به این ترتیب، امکان ردیابی و نظارت بر موارد به منظور کاهش هزینه ها و علاوه بر آن، اطلاع از زمان نیاز به تعمیر یا تعویض وجود دارد. ورمسان و همکاران اینترنت اشیا را به عنوان الگویی تعریف می کند که در آن چیزهای مختلفی وجود دارد.

در این محیط ، این اشیاء به صورت بی سیم یا سیمی متصل می شوند، به طور منحصر به فرد قابل شناسایی هستند و می توانند برای رسیدن به اهداف مشترک با یکدیگر همکاری کنند. برای امکان پذیر ساختن این پارادایم، اینترنت اشیا از چندین فناوری توانمند منشأ گرفته از زمینه های تحقیقاتی مختلف، مانند ارتباطات ماشین به ماشین، RFID، شبکه های حسگر بی سیم (WSN)، داده های معنایی، رایانش ابری، و خدمات بهره می برد.

امروزه، بسیاری از برنامه‌های کاربردی برای اینترنت اشیا در حوزه‌های مختلفی مانند مراقبت‌های بهداشتی نظارت بر محیط زیست، یا کارخانه‌های هوشمند و حتی شهر های هوشمند توسعه یافته‌اند.

۲-۲ پردازش جریان داده و داده پیچیده

الزامات برنامه های کاربردی اینترنت اشیا نمی توانند با به کارگیری سیستم های مدیریت پایگاه داده سنتی (DBMS) که با استفاده از فرآیندهای استخراج-تبدیل بار (ETL)، که نیاز به ذخیره و فهرست بندی داده ها برای پردازش دارند، برآورده شوند. بنابر این یکی از نیازهای مهم برنامه های اینترنت اشیا، توانایی پردازش جریان‌های داده‌ای چندگانه، پیوسته، سریع و متغیر می باشد که این موضوع برنامه های اینترنت اشیا را قادر می سازد مقیاس پذیر، پویا و واکنش پذیر باشند.

پردازش جریان داده، که تکاملی از پردازش داده در DBMS اســت، پرس‌وجوهای پیوســته را بر روی جریان‌های داده ورودی اجرا می‌کند و ســیســتم های مدیریت جریان داده (DSMS) بر روی داده های گذرا کار میکنند، یعنی داده هایی که به طور مداوم به روز می شوند.

علاوه بر این، پرس و جوها در DBMS ها یک بار اجرا می شــوند و پاســخ های کامل را برمی گردانند، در حالی که DSMS ها پرس و جوها را به طور مداوم اجرا می کنند و پس از رسـیدن داده های جدید، پاسخ های به روز ارائه می دهند.

به طور معمول، DSMS جریان‌های داده را از طریق دنباله‌ای از تبدیل‌های مبتنی بر عملگرهای SQL، مانند انتخاب، تجمیع، یا پیوســتن، که توســط جبر رابطه‌ای تعریف می‌شــود، پردازش می‌کند. از ســوی دیگر، پردازش رویدادهای پیچیده (CEP) شــامل مجموعه‌ای از اصــول و تکنیک‌ها برای تجزیه و تحلیل مجموعه‌هایی از رویدادها اســت که تا حدی بر اســاس زمان با رســیدن این رویدادها مرتب شــده‌اند. یعنی CEP ابزاری را برای پردازش مجموعه ای از رویدادهای مرتبط به هم به صورت پیوسته و به موقع فراهم می کند.

رویـدادهـا معمولـاً برای پردازش در الگوها ارائه می‌شــوند، با این حال، می‌توانند با رویدادهای نامرتبط دیگر مخلوط شــونـد. ویژگی مهم CEP توانـایی تشــخیص الگوهـا (به عنوان مثال، روابط) بین رویدادها اســت به عنوان مثال، موقعیت‌هایی مانند خرابی ماشین. تصمیم گیری که کدام رویکرد پردازش باید استفاده شود. از سوی دیگر، اگر برنامه نیاز به تجزیه و تحلیل مجموعه ای از رویدادهای نامرتب داشته باشد، CEP باید اعمال شود.

۲-۳ مشکل قراردادن اپراتورها

پردازش جریان های داده می تواند از طریق یک سـیستم متمرکز انجام شود یا می توان آن را بین گره های پردازشی مختلف برای اجرا توزیع کرد. با این حال، پردازش توزیع شده علاوه بر پردازش به موقع، چالش دیگری را نیز به همراه دارد که آن مشکل قرار دادن اپراتور می باشد.

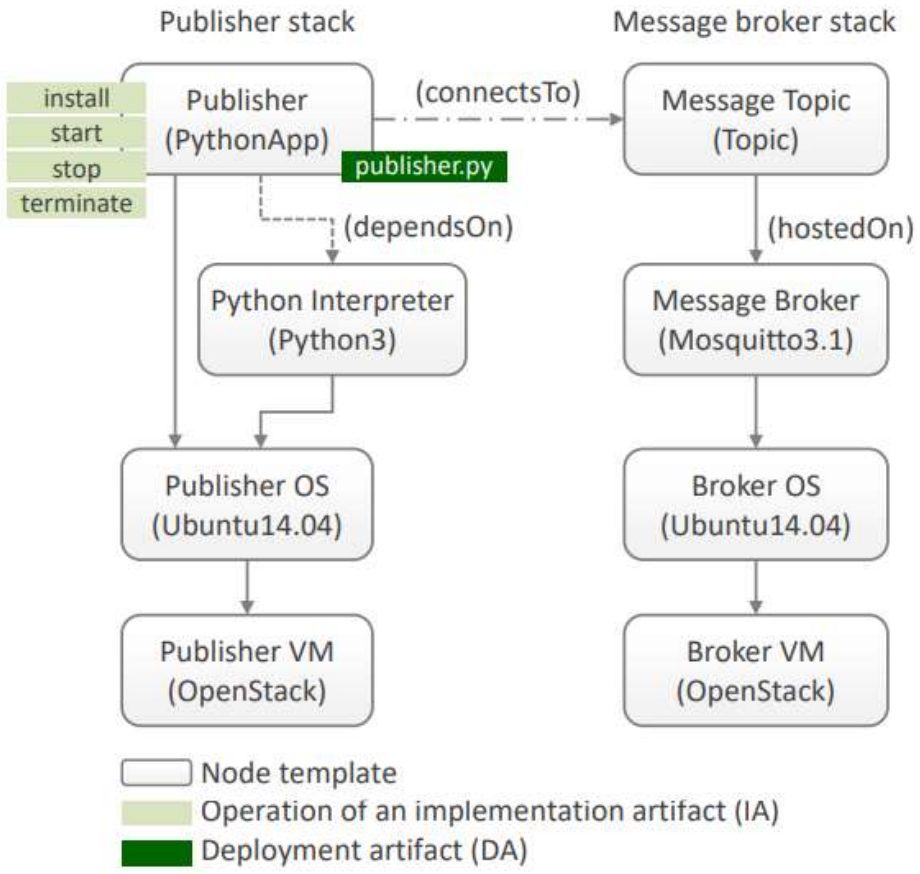
هدف این مشــکل یافتن یک مکان بهینه از کل پرس و جوهای پیوســته یا اپراتورهای منفرد در مجموعه ای از گره های پردازشــی مختلف اسـت که در سراسر یک شبکه توزیع شده اند. مکان بهینه معمولاً بر اساس توابع هزینه تعریف شـده توسـط سـیسـتم یا تعریف شـده توسـط کاربر محاسبه می شود، که هدف آن ارائه، برای مثال، عملکرد بالاتر یا توزیع بار بهتر است.

۲-۴ تکنولوژی TOSCA

فنـاوری رایـانش ابری اخیراً برای میزبـانی و ارائـه خـدمات از طریق اینترنت پدید آمده و به طور فزاینده ای همراه با پارادایم اینترنت اشـیا به کار گرفته شـده است تا بتواند باعث مقیاس پذیری و قابلیت همکاری آن شود. رایانش ابری می‌تواند در عین هزینه‌های پایین برای اسـتقرار کل محیط‌های اینترنت اشـیا ، قابلیت راه‌اندازی و ادغام سـریع اشیاء جدید و برنامه‌های کاربردی اینترنت اشیا را ایجاد کند.

فنـاوری TOSCA (Topology and Orchestration Specification for Cloud Applications) یـک استاندارد OASIS تایید شده برای مدل سازی، استقرار و مدیریت برنامه های کاربردی ابری است که دارای دو بخش اصـلی اسـت: (۱) مدل توپولوژی یک برنامه کاربردی که شامل اجزای نرم افزار ، پلت فرم و زیرساخت آن می باشد، و (۲) مراحل استقرار این برنامه.

TOSCA بسـیار عمومی اسـت به طوری که تعریف انواع دلخواه را برای توصـیف مؤلفه های برنامه، به نام انواع گره، و وابسـتگی های آنها، به نام انواع رابطه، قادر می سـازد. تصـویر زیر یک نمونه از مدل TOSCA برای یک نرم افزار کاربردی انتشار و اشتراک اطلاعات می باشد.



شکل ۱

TOSCA اسـتقرار و مدیریت برنامه های ابری را در سـراسـر چرخه زندگیشان آسان تر می کند بدون اینکه خللی در الزامات امنیتی ، حاکمیتی و انطباقی آنها پیش آید . همچنین توانایی آن برای تسـهیل یک اکوسـیسـتم اسـت که اجرای پورتابل برنامه ها برای ابر و بین ابرها را فراهم می سـازد و این امکان را فراهم می سـازد که برنامه های ابری در هر ابری به صورت پورتابل توصیف ، مدل سازی ، پکیج بندی ، تنظیم و مانیتور شوند .

فصل سوم

بررسی اجمالی پایان نامه

۳-۱ دستاوردها

دستاورد ها و پژوهش های ارائه شده در این پایان نامه بر اساس استانداردهای تعیین شده (مانند MQTT، TOSCA، XML) است تا از کاربرد طولانی مدت آنها اطمینان حاصل شود. این دستاوردها به شرح ذیل می باشند:

- مدل پردازش جریان داده در محیط اینترنت اشیاء :

این مدل، حاوی اطلاعاتی در مورد نحوه دسترسی به اشیاء اینترنت اشیا (به عنوان مثال، آدرس IP یک دستگاه) و در مورد قابلیت های آنها (به عنوان مثال، قدرت پردازش، ذخیره سازی موجود یک دستگاه) است. علاوه بر این، این تحقیق مدل پردازش جریان داده (DSPM) را برای توصیف منطق پردازش داده یک برنامه IoT ارائه می‌کند. این مدل منابع داده (به عنوان مثال، حسگرها)، مخازن داده (به عنوان مثال، محرک)، اپراتورهای پردازش (به عنوان مثال، فیلتر داده، تجمع)، و جریان داده در میان آنها را توصیف می کند. علاوه بر این، الزامات اپراتورهای پردازش برای اشیاء IoT، مانند حداقل حافظه موجود مورد نیاز را شرح می دهد.

- استقرار مدل پردازش جریان داده (DSPM) در محیط اینترنت اشیاء (IoTEM) :

در این تحقیق، ابزارهایی برای استقرار مدل‌های پردازش جریان داده (DSPM) بر روی مدل‌های محیط اینترنت اشیا (IoTEMs - IoT environment model)، با در نظر گرفتن الزامات اپراتورهای پردازش و قابلیت‌های اشیاء IoT ارائه شده‌اند. این تحقیق دو رویکرد اصلی را برای استقرار مدل ارائه می‌کند:

۱- یک رویکرد خودکار، که در آن یک طرح نقشه‌برداری حاوی حداقل یک مجموعه از اشیاء IoT که الزامات DSPM را برآورده می‌کنند، به‌طور خودکار تولید می‌شود که برای محیط های بزرگ‌تر مناسب است، به عنوان مثال، در حوزه کارخانه هوشمند، که در آن تعداد زیادی از اشیاء در دسترس هستند.

۲- یک رویکرد دستی، که در آن تحلیلگران خودشان تصمیم می گیرند که کدام اشیاء IoT باید کدام اپراتورها را اجرا کنند. برای این کار، تحلیلگران به صورت دستی یک مدل استقرار ایجاد می کنند که برای موارد کوچک توصیه می شود، به عنوان مثال، برای خانه های هوشمند.

- استقرار اپراتورها در محیط های IoT :

اساس این تحقیق از استاندارد TOSCA که مورد تایید سازمان پیشرفت استانداردهای اطلاعات ساختاریافته (OASIS) استفاده می‌کند که در دو رویکرد خودکار و دستی الزامی می باشد.

- نظارت بر DSPM های مستقر شده :

پس از اســـتقرار اپراتورهای پردازش، پردازش کلی جریان داده یک برنامه IoT آغاز می شـــود و برای اطمینان از اینکه این پردازش تا زمانی که برنامه IoT به آن نیاز دارد درست می ماند باید ابزاری را برای تشخیص اختلالات مؤثر بر مدل پردازش جریان داده مسـتقر شده (DSPM) فراهم کرد. چنین اختلالاتی می تواند از طریق تغییرات در محیط اینترنت اشـــیا (به عنوان مثال، یک دســـتگاه معیوب) یا تغییر در DSPM (به عنوان مثال، با افزودن یک نیاز جدید) رخ دهد. برای تشـخیص چنین اختلالاتی، MBP به طور مداوم اشـیاء IoT و DSPM ها را نظارت می کند. این نظارت عمدتاً با اسـتفاده از تکنیک‌های پردازش رویداد پیچیده (CEP) انجام می‌شـود. چنین تکنیک هایی به خوبی تثبیت شـده اند و برای پردازش مســـتمر مقادیر زیادی از داده ها، به عنوان مثال، برای تشـــخیص به موقع موقعیت های بحرانی (یعنی تغییراتی که نیاز به اقدامات اصلاحی دارند) استفاده می شوند.

۳-۲ رویکرد تخصصی

متخصصان دامنه (Domain experts) دانش فنی در مورد اشیاء سخت افزاری (یعنی دستگاه ها، حسگرها، محرک ها)، اشـیاء مجازی (یعنی ماشـین های مجازی) و اتصالات شبکه آنها در یک محیط IoT دارند. این دانش فنی شامل اطلاعاتی در مورد قابلیت های محاسباتی (حافظه اصلی موجود، اتصال، قدرت پردازش) است.

همچنین، این کارشـناسـان در مورد نحوه دسـترسـی به این اشـیاء برای مثال، اسـتخراج داده‌های حسـگر یا ارسـال دسـتورات کنترلی به یک محرک، دانش دارند. وظیفه اصـلی این متخصـصـان ، ایجاد IoTEM ها و ثبت خودکار در MBP و ایجاد DSPM ها می باشـد زیرا دانش کافی در مورد پردازش داده های تولید شده در محیط اینترنت اشیاء را دارند، به عنوان مثال، دانش لازم را برای مدل سـازی برنامه های کاربردی اینترنت اشیا مختلف برای موارد استفاده خاص.

یک DSPM مبتنی بر نمودار اســـت و شـــامل منابع داده، ســـینک های داده، و اپراتورهای پردازش به عنوان گره، و اتصالات جریان داده بین این گره ها به عنوان لبه است. DSPM ها بر اساس الگوی طراحی لوله ها و فیلترها هستند.

بر اسـاس قابلیت‌های کلی IoTEM و الزامات مشخص‌شده توسط DSPM باید تصمیم‌گیری کرد که اپراتورها بر روی کدام اشـیاء IoT مسـتقر شـوند و این کار توسـط الگوریتم هایی انجام می شـود که هم قابلیت‌های اشیاء IoT، و هم قابلیت‌های اتصالات شبکه را در نظر می‌گیرند.

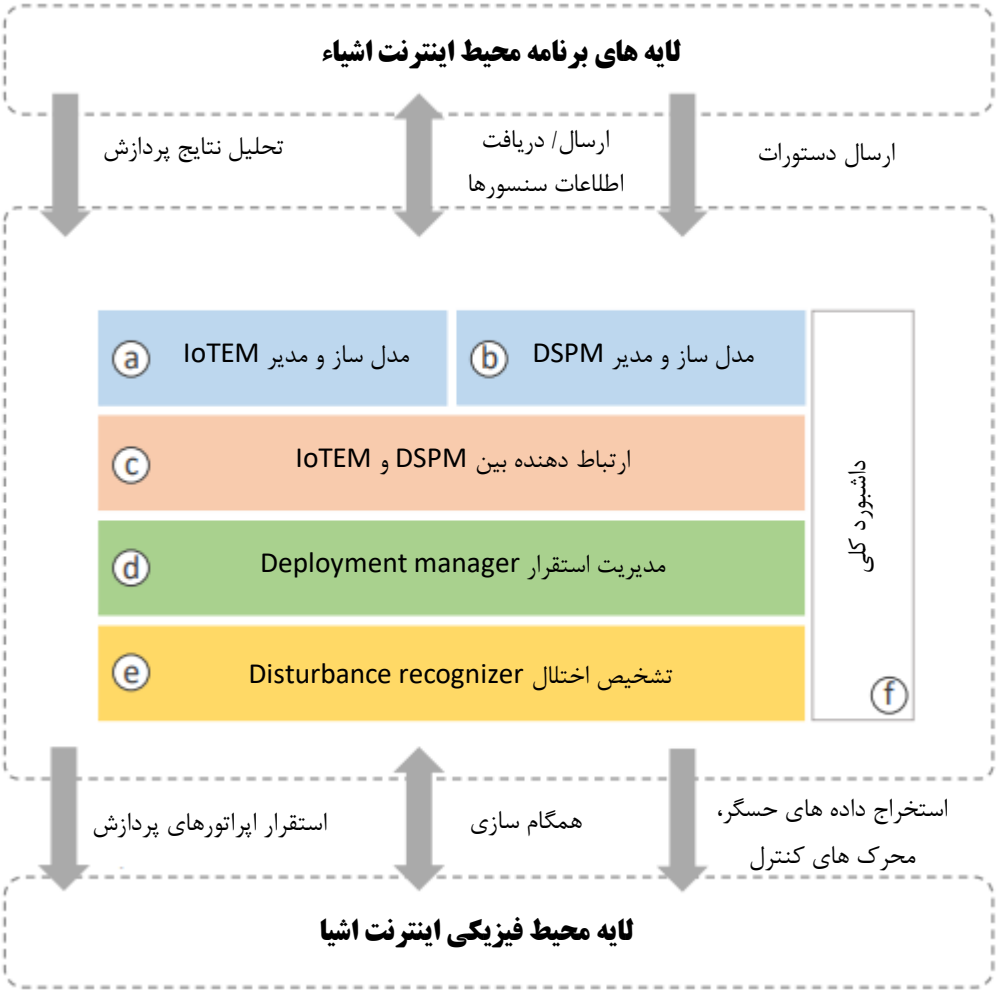
قرار دادن اپراتورها باید تا حد امکان به منابع داده نزدیک باشـد تا احتمالاً ترافیک شـبکه و حجم داده مبادله شده در یک محیط IoT کاهش یابد. تحلیلگران دامنه به صـورت دسـتی یک نقشـه ایجاد می کنند و با توجه به طرح نقشـه، اپراتورهای پردازش می توانند بر روی اشـیاء IoT مربوطه خود مسـتقر شـوند و اجرای DSPM می تواند آغاز شـود. اشیاء IoT ابتدا برای اجرای اپراتورهای پردازشی آماده می شوند، به عنوان مثال، نرم افزار طراحی شده مورد نیاز یک اپراتور پردازشـگر نصـب و در صـورت لزوم پیکربندی می شـوند. نرم افزارهای طراحی شده بیشتری برای جمع آوری مقادیر تجربی و نظارت بر اپراتورهای پردازش و اشـیاء اینترنت اشـیا نیز نصـب و پیکربندی می شـوند. پس از آن، اپراتورهای پردازش بر روی اشـیاء IoT متناظر خود مسـتقر می شـوند و سپس راه اندازی می شوند، که منجر به یک نمونه در حال اجرا از DSPM می شـود که dDSPM نامیده می شـود. آماده سازی اشیاء IoT و استقرار متعاقب آن اپراتورهای پردازش با استفاده از استاندارد TOSCA محقق می شود.

برای اطمینـان از اینکه پردازش کلی در محیط اینترنت اشـــیا درســـت می‌ماند، اشـــیاء IoT و dDSPM به‌منظور تشـخیص اغتشـاشـات به‌طور مداوم نظارت می‌شـوند. اگر اختلالی بر پردازش اصلی تأثیر منفی بگذارد، الگوریتم‌های استقرار مجدداً راه‌اندازی می‌شوند و یک طرح نقشه‌برداری جدید ایجاد می‌شود.

اپراتورهای پردازش و نرم افزارهای طراحی شـــده که دیگر مورد نیاز نیســـتند متوقف و حذف می شـــوند و پردازش جریان داده برنامه IoT می تواند کنار گذاشته شود و اشیاء IoT پاک شوند.

۳-۳ معماری کلی

معماری کلی طبق تحقیقات این پایان نامه از ســه لایه اصـلی تشکیل شده است: لایه محیط فیزیکی IoT، لایه برنامه کاربردی IoT و لایه پلتفرم اتصـــال و تامین چند منظوره (MBP) که شـــکاف بین محیط های فیزیکی IoT و برنامه های کاربردی IoT را پر می کند. تصویر زیر این نمای کلی را نشان میدهد :



شکل ۱

سهم این پایان نامه در لایه میانی قرار دارد که MBP برای آن طراحی شده است.

بخش مدلساز و مدیر IoTEM نقطه ورود رویکرد روشمند است و ابزارهایی را برای کارشناسان دامنه برای ایجاد، ذخیره و مدیریت IoTEM ها و DSPM هایی را که منطق پردازش برنامه‌های اینترنت اشیا را توصیف می‌کنند فراهم می کند و از طریق این بخش، اشیاء IoT یک IoTEM در MBP ثبت می‌شوند و همتاهای دیجیتالی آن‌ها نمونه‌سازی می‌شوند. همتایان دیجیتالی API هایی را ارائه می‌کنند که می‌توانند توسط برنامه‌های IoT، به عنوان مثال، برای دسترسی به دستگاه‌ها، حسگرها و محرک‌های ثبت‌شده در MBP به آن‌ها دسترسی داشته باشند.

از طریق بخش مدلساز و مدیر IoTEM ، تحلیلگران داده همچنین می‌توانند نمونه‌های در حال اجرا DSPM (dDSPMs) را زمانی که دیگر نیازی به پردازش نباشد متوقف کنند.

بخش ارتباط دهنده بین IoTEM و DSPM ابزاری را برای پشتیبانی از تصمیم در مورد جایی که اپراتورهای پردازش باید مستقر شوند فراهم می کند. علاوه بر این، الگوریتم هایی را برای تصمیم گیری خودکار در مورد قرار دادن اپراتور، بر اساس الزامات اپراتورهای پردازش و قابلیت ها ارائه می دهد.

بخش مدیریت استقرار (Deployment manager) مسئول استقرار اپراتورهای پردازش بر روی اشیاء IoT است، به طوری که اجرای DSPM ها می تواند آغاز شود. علاوه بر این، هر نرم افزار طراحی شده مورد نیاز دیگر برای نظارت بر اشیاء اینترنت اشیا، می تواند از طریق این مؤلفه نیز مستقر شود.

در بخش شناسایی اختلال (Disturbance recognizer) ، همتایان دیجیتالی اشیاء IoT و dDSPM به طور مداوم نظارت می شوند تا اختلالات در طول پردازش داده را تشخیص دهند. این نظارت را می توان از طریق اسکریپت ها، مانند اسکریپت های پایتون (Python) یا شل (Shell)، یا از طریق پیاده سازی های پیچیده تر، مانند انجام پرس و جوهای پردازش رویداد پیچیده (CEP) باشد.

بخش شناسایی اختلال محیط های زمان اجرا مختلفی را برای تحقق نظارت فراهم می کند، از جمله یک موتور CEP برای ارزیابی مستمر درخواست های CEP. در نهایت، ابرداده و داده های پویا اشیاء اینترنت اشیا را می توان توسط بخش داشبورد مشاهده کرد. برای مثال، اطلاعاتی در مورد در دسترس بودن و فضای دیسک فعلی اشیاء IoT، داده های تاریخی و آخرین اندازه گیری مقادیر حسگر ارائه می دهد.

فصل چهارم

مدل محیط اینترنت اشیاء و پردازش جریان داده

۴-۱ مدل محیط اینترنت اشیاء

پیشرفت مستمر در فناوری‌های حسگر و شبکه، وجود دستگاه‌های اینترنت اشیا را امکان‌پذیر کرده است که به هم پیوسته هستند و به طور مداوم اطلاعات پیرامون خود و خود را مبادله می‌کنند و محیطی که شامل یک یا چند دستگاه از این قبیل باشد، محیط اینترنت اشیا نامیده می شود.

چنین محیط‌هایی در حوزه‌های مختلفی مانند خانه‌های هوشمند ، کارخانه‌های هوشمند یا شهرهای هوشمند وجود دارند. علاوه بر این، در این محیط می توان برای تقویت قدرت محاسباتی و مدیریت حجم زیاد داده از منابع مجازی ارائه‌شده توسط فناوری‌های رایانش ابری نیز استفاده کرد.

به طور معمول، اشیاء IoT توسط پلتفرم های IoT مدیریت می شوند که دسترسی به برنامه های IoT را از طریق API های سطح بالا فراهم می کنند و در سال‌های اخیر، بسیاری از پلتفرم‌های IoT توسعه یافته‌اند. در بسیاری از رویکردها، مانند FIWARE, IBM Watson IoT, OpenMTC, Microsoft Azure IoT اشیاء IoT به صورت دستی و جداگانه ثبت و با این پلتفرم‌های IoT پیکربندی می‌شوند. با این حال، این یک کار پیچیده و زمان بر است. علاوه بر این، محیط های IoT بسیار پویا هستند، به عنوان مثال، دستگاه ها ممکن است معیوب شوند. بنابراین، پلتفرم‌های اینترنت اشیا باید با محیط‌های اینترنت اشیا همگام باشند تا برنامه‌های اینترنت اشیا از تغییرات آگاه شوند.

برای این منظور، مدل محیط IoT و پردازش جریان داده، IoTEM را برای توصیف کل محیط‌های IoT فراهم می‌کند. IoTEM ها در پلتفرم چند منظوره Binding and Provisioning (MBP) ادغام شده و توسط آن برای ثبت و پیکربندی خودکار محیط های IoT به عنوان یک کل به جای پیکربندی هر شی اینترنت اشیا به صورت جداگانه استفاده می شود. محیط‌های ثبت‌شده اینترنت اشیا دائماً برای تشخیص تغییرات حیاتی در محیط نظارت می‌شوند ،به عنوان مثال، زمانی که یک دستگاه معیوب می‌شود.

در این پایان نامه، یک بررسی جامع با مقایسه مدل های مختلف اینترنت اشیا انجام شده تا مدل مناسبی به عنوان IoTEM انتخاب شود. در جدول زیر نتایج این ارزیابی نشان داده شده است که در آن مقایسه ای مبتنی بر معیار از چندین مدل اینترنت اشیا ارائه شده است.

عنوان مدل	تکامل	سلسله مراتبی	دسترس بودن	پیاده سازی	موقعیت جغرافیایی	کاربرد
homeML	Non-standard	✗	✗	✗	✓	خانه های هوشمند
IEEE ۱٤٥۱	standard	✗	✓	✓	✗	متمرکز بر روی سنسور ها
IoT ARM	Non-standard	✓	✗	✗	✓	مدل مرجع عمومی
IoT-Lite	submitted	✓	✓	✓	✓	حسگرها و SSN ontology
IoT MC	standard	✓	✓	✓	✗	IoTivity
IoT-O	standard ext	✓	✗	✗	✗	حسگرها و SSN ontology
Nexus	Non-standard	✓	✓	✓	✓	جغرافیایی سازی
oneM۲M	standard	✓	✓	✓	✓	تمرکز بر خدمات دستگاه های IoT
OPC-UA	standard	✓	✓	✓	✗	کارخانه های هوشمند
SenML	standard	✗	✓	✓	✓	تمرکز بر حسگرها و مقادیر حسگر
SensorML	standard	✗	✓	✓	✓	پشتیبانی از پردازش ها
SSN	standard	✓	✓	✓	✓	مورد استفاده IoT-Lite / IoT-O
TDLIoT	Non-standard	✗	✓	✓	✓	نمونه اولیه تحقیق
Vorto	Non-standard	✗	✓	✓	✗	زبان برنامه نویسی

معیار تکامل نشان می دهد آیا مدل یک مدل استاندارد تایید شده یک سازمان، مانند OASIS، W۳C یا OGC است یا خیر و می توان فرض گرفت سازمان‌ها تحت یک فرآیند بررسی کامل قرار گرفتند و یک استاندارد مزایایی را ارائه می‌کند.

معیار سلسله مراتبی یک عامل مهم هنگام مدل‌سازی محیط‌ها در اینترنت اشیا است، زیرا آنها معمولاً شامل استقرارهای سلسله مراتبی در میان اشیاء مختلف اینترنت اشیا هستند. دو نوع اصلی سلسله مراتب وجود دارد، گروه بندی و انتزاع. از طریق گروه‌بندی، باید بتوان سیستم‌های پیچیده را مدل‌سازی کرد، مانند ماشین‌های تولیدی در یک کارخانه هوشمند، که حاوی مقدار بالایی از دستگاه‌ها، حسگرها و محرک‌ها هستند و از طریق انتزاع، می توان انواع عمومی را تعریف کرد. به عنوان مثال، ماژول های سنسور مختلف اندازه گیری دما را می توان توسط سنسور دما از نوع عمومی جمع کرد.

معیار در دسترس بودن به این موضوع اشاره دارد که آیا مدل اینترنت اشیا به صورت عمومی در دسترس است یا نه، و علاوه بر این، آیا جامعه وسیعی در توسعه آینده آن مشارکت دارد یا خیر. واضح است که برای ایجاد و توسعه بیشتر یک مدل اینترنت اشیا، جامعه بزرگی از کاربران و توسعه دهندگان، یا یک سازمان بزرگتر مورد نیاز است.

برای درک این موضوع، مدل باید یا منبع باز در دسترس باشد، یا اگر منبع بسته است، باید توسط یک سازمان بزرگتر توسعه و استفاده شود.

معیار پیاده سازی به این موضوع اشاره دارد که آیا پیاده سازی شده مدل وجود دارد یا خیر. به عنوان مثال، در مقالات علمی، مفاهیم جالبی ایجاد می شود که ممکن است پیاده سازی متناظری نداشته باشند. برای استفاده در سناریوهای دنیای واقعی، پیاده سازی در دسترس از اهمیت حیاتی برخوردار است. این شامل ابزارهای موجود برای ایجاد و مدیریت مدل نیز می شود.

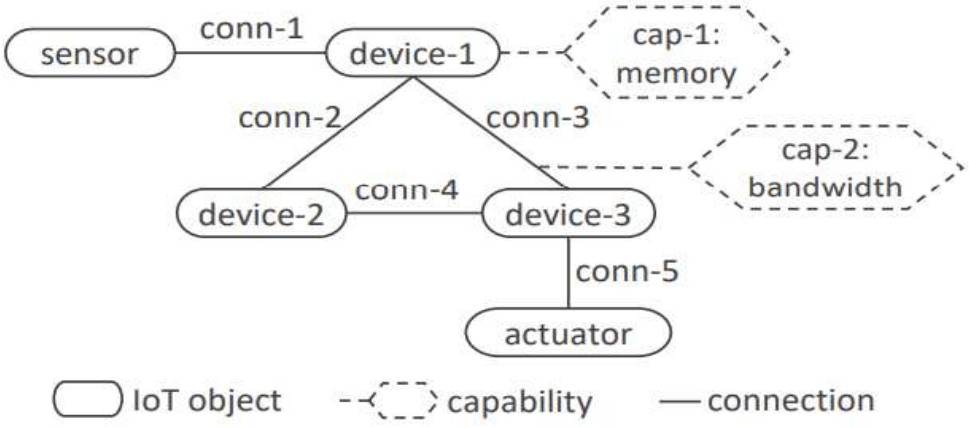
معیار موقعیت جغرافیایی به این اشاره دارد که آیا مدل اینترنت اشیا می‌تواند مکان‌های (جغرافیایی) اشیاء اینترنت اشیا را توصیف کند، که ویژگی‌های پیچیده‌ای مانند پرس و جو مبتنی بر مکان را فعال می‌کند.

در اینترنت اشیا، مکان آنها مهم هستند، چون رویدادهایی که ممکن است نیاز به واکنش داشته باشند در هر محیطی متفاوت می باشد، به عنوان مثال، رویدادهایی که در یک خانه هوشمند رخ می دهد.

۴-۱-۱ تعیین IoTEM

تعریف IoTEM یک گراف بدون جهت است که شامل اشیاء IoT به عنوان گره و اتصالات شبکه آنها به عنوان یالهای گراف است که بر اساس فاصله اتصالات شبکه وزن دهی شده است. ساختار آن به شرح ذیل می باشد :

مدل در محیط IoT متشکل از چند بخش است : اشیاء (Object)، اتصالات بین اشیاء (Connection)، فاصله (Distance) (تابع برای وزن دهی اتصالات شبکه بر اساس فاصله)، قابلیت‌های شی (ObjectCapabilities)، قابلیت های اتصال (ConnectionCapabilities)، تخصیص قابلیت شی (objectCapabilityAssignment)، تخصیص قابلیت اتصال (connectionCapabilityAssignment) که در تصویر زیر این مدل نمایش داده شده.



Objects = {device-1, sensor, device-2, device-3, actuator}
Connections = {conn-1, conn-2, conn-3, conn-4, conn-5}
ObjectCapabilities = {memory}
ConnectionCapabilities = {bandwidth}
objectCapabilityAssignment = {(device-1, cap-1)}
connectionCapabilityAssignment = {(conn-3, cap-2)}
distance = {(conn-1, 0), (conn-2, 10), ...}

شکل ۲

۴-۱-۲ قابلیت های اشیاء و ارتباطات

از نظر قابلیت های محاسباتی، McEwen و همکاران یک شی IoT را به عنوان یک کامپیوتر کوچک تعریف می کند که اتصال شبکه، قدرت پردازش و ذخیره سازی را فراهم می کند. نمونه هایی از اشیاء سخت افزاری در اینترنت اشیا Arduino Yún ، BeagleBone، ESP۸۲۲۶ و Raspberry Pi هستند.

قابلیت‌هایی که می‌توانند برای توصیف یک شی IoT استفاده شوند، از McEwen و همکاران ارائه شده‌اند. همچنین Cipriani و همکاران موضوعاتی مانند تعریف و محدودیت های محیط های ناهمگن و توزیع شده را ارائه داده اند.

- احراز هویت (Authentication): در حوزه IoT مجوز ارتباط ماشین با ماشین یا انسان به ماشین توسط مجوز ماشین است. این قابلیت مکانیسم های تأیید هویت پشتیبانی شده یک دستگاه IoT را توصیف می کند. نمونه‌هایی از مکانیسم‌های احراز هویت عبارتند از گذرواژه، اعتبار دیجیتال (digital credentials) و گواهی‌ها
- محرمانه بودن (Confidentiality): این قابلیت توضیح می دهد که چگونه اطلاعات ذخیره شده در دستگاه IoT محافظت می شود حتی اگر دسترسی غیرمجاز رخ دهد. برای مثال می توان با استفاده از مکانیسم های رمزگذاری به محرمانگی دست یافت
- دسته دستگاه (Device category): این پایان نامه چهار دسته دستگاه را تعریف می کند که بین پارادایم های مختلف استقرار بر روی دستگاه ها متمایز می شود. این دسته‌بندی‌ها نتیجه تحقیقات ما درباره نحوه قرار دادن اپراتور (به عنوان مثال، استقرار نرم‌افزار) در انواع مختلف دستگاه‌های IoT هستند:
  - یک دستگاه plug-and-play دارای حسگرها و/یا محرک‌های تعبیه‌شده است. با این حال، اجازه استقرار از راه دور را نمی دهد، بلکه API ها را از طریق یک سرور ابری برای دسترسی به حسگرها و محرک های آن فراهم می کند. نمونه هایی از چنین دستگاه هایی با پوشش بی سیم هستند که دائماً داده های خود را با سرورهای ابری همگام می کنند. بنابراین، قرار دادن اپراتور در دستگاه plug-and-play امکان پذیر نیست، با این حال، اپراتورها را می توان در سایر اشیاء IoT متصل به دستگاه plug-and-play از طریق API های ارائه شده مستقر کرد.
  - یک دستگاه قابل تنظیم، مانند رایانه Raspberry Pi، به حسگرها و محرک‌ها اجازه می‌دهد از طریق رابط فیزیکی (مانند GPIO) به آن متصل شوند. چنین دستگاهی دارای یک سیستم عامل (OS) است که پیکربندی از راه دور و استقرار اپراتورها را قادر می سازد، به عنوان مثال، برای استخراج و ارسال داده های حسگر به یک پلت فرم IoT.
  - یک دستگاه قابل تنظیم محدود، مانند Arduino Yún و Bosch XDK Node ، دارای حسگرها و محرک‌هایی است که به آن تعبیه شده یا به آن متصل شده‌اند و پیکربندی از راه دور و استقرار نرم‌افزار را امکان‌پذیر می‌سازد. با این حال، چنین دستگاه‌هایی محدود هستند، یعنی قابلیت‌های پردازش و ذخیره‌سازی محدودی دارند. در این مورد، باید از اپراتورهای پردازش پیچیده در این نوع دستگاه اجتناب شود و داده‌های مورد پردازش باید به جای اینکه توسط خود دستگاه مدیریت شوند، به دستگاه‌های قدرتمندتر ارسال شوند.

- یک دستگاه وابسته به دروازه پیکربندی خود (به عنوان مثال، فلش کردن سیستم عامل firmware flashing) و استقرار نرم افزار که فقط از طریق اتصال به یک دستگاه میزبان امکان پذیر می کند، در این پایان نامه دروازه (gateway) نامیده می شود.

به طور کلی، یک دروازه دو سیستم را با استفاده از قالب بندی، پروتکل های ارتباطی یا معماری های مختلف به هم متصل می کند. نمونه هایی برای چنین دستگاه هایی ماژول های ESP۸۲۶۶ یا بردهای Arduino Leonardo . در این مورد، نقش یک دروازه را می توان توسط دستگاه های قابل تنظیمی که به طور فیزیکی به دستگاه های وابسته به دروازه متصل هستند (به عنوان مثال، از طریق یک پورت USB) فرض کرد.

از طریق چنین اتصالات فیزیکی، آپلود اپراتورها در دستگاه های وابسته به دروازه امکان پذیر است.

- حافظه (Memory): این قابلیت با حافظه کاری موجود (یعنی RAM) دستگاه IoT مطابقت دارد.
- نوع شبکه (Networking type). این قابلیت نحوه اتصال یک دستگاه اینترنت اشیا به دستگاه های دیگر در محیط اینترنت اشیا را توضیح می دهد. نمونه هایی از انواع شبکه عبارتند از: اترنت (IEEE۸۰۲,۳)، Wi-Fi (IEEE۸۰۲,۱۱)، بلوتوث (IEEE۸۰۲,۱۵,۱)، بلوتوث کم انرژی (IEEE۸۰۲,۱۵,۴)، LTE۴G، یا G۵ .
- حالت مصرف برق (Power consumption mode). در حوزه اینترنت اشیا، دستگاه‌های وابسته به باتری باید بتوانند در حالت‌های مصرف کم مصرف کار کنند تا بتوانند با برنامه‌های طولانی مدت اینترنت اشیا همگام شوند. این قابلیت حالت‌های مصرف انرژی را که یک دستگاه اینترنت اشیا می‌تواند پشتیبانی کند، توصیف می‌کند. به عنوان مثال، Raspberry Pi چهار حالت قدرت را ارائه می دهد، از جمله یک حالت اجرا (پیش فرض) و یک حالت آماده به کار .
- قدرت پردازش (Processing power). این قابلیت با سرعت CPU دستگاه اینترنت اشیا مطابقت دارد. به عنوان مثال، Raspberry Pi۳ مدل B دارای چهار پردازنده با فرکانس ۱,۲ گیگاهرتز است.
- گنجایش دخیره سازی (Storage capacity). این قابلیت با مقدار ذخیره سازی موجود مطابقت دارد. این برای اپراتورهایی که داده‌های حسگر را جمع‌آوری می‌کنند یا نتایج میانی را ذخیره می‌کنند، مهم است.
- پشتیبانی از زمان اجرا (Supported runtime): این قابلیت مربوط به لیستی از محیط‌های زمان اجرا نرم‌افزار پشتیبانی شده و در دسترس (مانند موتورهای جاوا، پایتون، CEP) یک دستگاه اینترنت اشیا است.

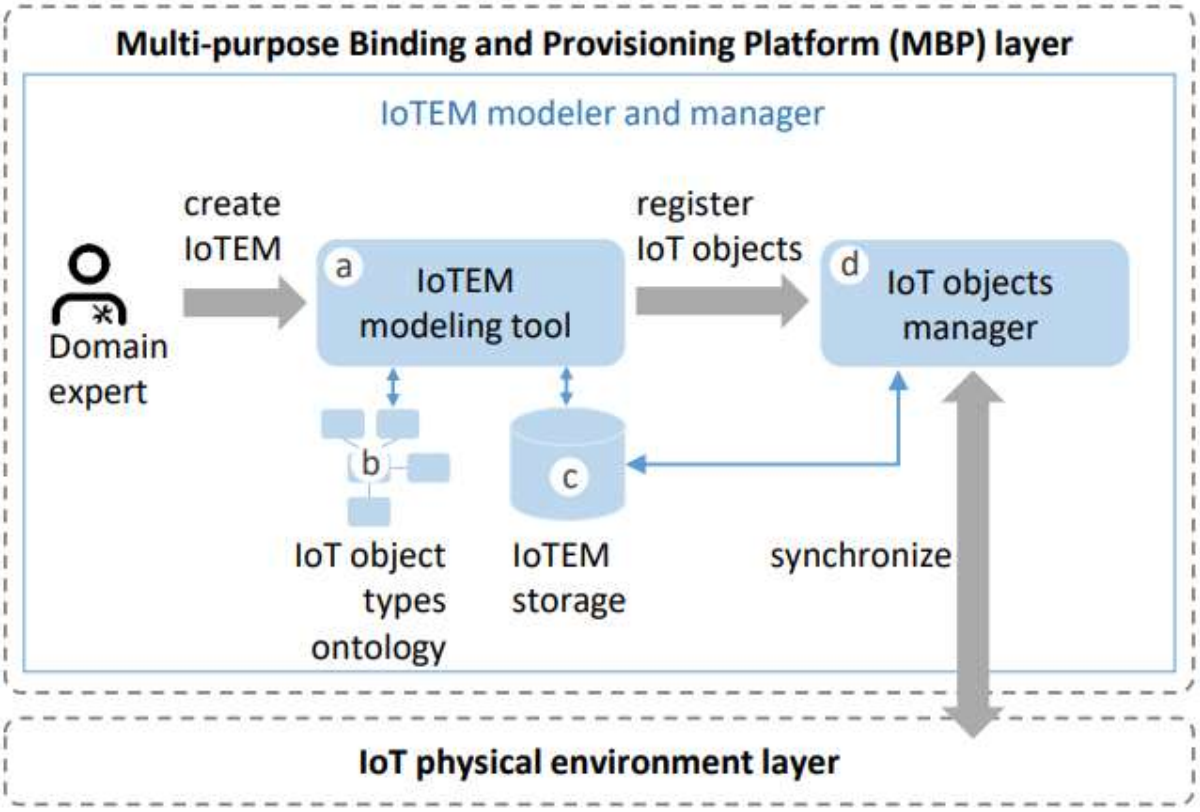
همانطور که در بخش ۴-۱ بیان شد، قابلیت های اتصالات شبکه در بین اشیاء IoT توسط IoTEM مدل شده است. لیست زیر، زیرمجموعه ای از قابلیت های اتصال ممکن را نشان می دهد و از این رو کامل نیست :

- پهنای باند (Bandwidth): این قابلیت تخمینی در مورد میزان داده ای که می تواند در مدت زمان ثابتی توسط اتصال منتقل شود را ارائه می دهد.
- رمزگذاری (Encryption): این قابلیت توضیح می دهد که آیا تبادل داده از طریق اتصال ارجاع شده رمزگذاری شده است.
- تاخیر (Latency): این قابلیت تخمینی از زمان مورد نیاز برای تبادل داده توسط یک اتصال را ارائه می دهد.

- نوع شـبکه (Networking type). این قابلیت نوع اتصال شبکه را توصیف می کند، به عنوان مثال، با سیم یا بی سیم.

۳-۱-۴ مولفه های معماری و پیاده سازی بخش مدل ساز و مدیر IoTEM

در معماری کلی (در بخش ۳-۳)، بخش مدل‌سـاز و مدیر IoTEM با پشـتیبانی متخصـصان دامنه امکان ارائه ابزاری برای ایجاد و ذخیره IoTEM و علاوه بر این، مدیریت IoTEM های ثبت‌شده در MBP را فراهم می کند.
بخش مدل ساز و مدیر IoTEM در شکل زیر نشان داده شده است.



شکل ۳

این مدل شامل ابزار مدل‌سازی گرافیکی IoTEM (بخش a) است که برای مدل‌سازی محیط اینترنت اشیا با کشیدن و رها کردن انواع شی IoT (بخش b) در یک محیط مدل‌سازی استفاده می‌شود.
انواع شـی IoT، مانند Raspberry Pis، حسـگرهای دما، و ماشـین های مجازی، توسـط یک هسـتی شـناسـی (ontology) ارائه می شـوند. IoTEM ها، که نمونه هایی از هسـتی شـناسـی هسـتند، در ذخیره سـازی IoTEM (بخش c) ذخیره می شوند.
علاوه بر این، ابزار مدل‌سـازی IoTEM به متخصـصـان حوزه امکان می‌دهد، علاوه بر ذخیره و بازیابی IoTEM، تمام اشیاء IoT یک IoTEM را به‌طور هم‌زمان در MBP ثبت کنند. این ثبت، مدیر اشیاء اینترنت اشیا (بخش d) را فعال می کند تا همتایان دیجیتالی (به عنوان مثال، دوقلوهای دیجیتال) اشیاء IoT مدل شده را نمونه برداری کند.

این همتاهای دیجیتالی در حافظه IoTEM ذخیره می شـوند. مدیر اشـیاء IoT دائماً با محیط فیزیکی اینترنت اشـیا هماهنگ می شـود تا اختلالات موجود در محیط فیزیکی را تشـخیص دهد، مانند زمانی که دسـتگاه ها معیوب می شـوند. وضـعیت فعلی اشـیاء ثبت شـده IoT را می توان در داشبورد MBP بررسی و تجسم کرد. چگونگی تشخیص اختلالات به تفصیل در فصل ۷ توضیح داده خواهد شد.

۴-۱-۴ کارهای مرتبط

چنانچه قبلا بیان شد رویکردهای زیادی برای مدل‌سازی محیط‌های IoT وجود دارد و مدل‌ها و ابزارهای IoT را برای ایجاد و مدیریت نمونه‌های این مدل‌های IoT ارائه شـده اند. پشـتیبانی ابزار از اهمیت حیاتی برخوردار اسـت زیرا پیچیـدگی مدل‌های اینترنت اشـیا (به عنوان مثال، فرمت‌های پیچیده داده) را انتزاعی می‌کند و کار مدل‌سـازی و مدیریت را آسان می‌کند. با این حال، رویکردهای پیشرفته ابزاری برای پشتیبانی از کل چرخه حیات محیط‌های IoT با مدل‌سازی می باشند و نه از طریق استقرار و یک مفهوم کل نگر مانند این پایان نامه برای مدل سازی، پیکربندی و نظارت بر محیط های IoT ارائه نمی کنند.

مک دونالد و همکاران یک ابزار مدل‌سـازی برای مدل homeML معرفی می‌کند، که تنها برای خانه‌های هوشـمند طراحی شـده اسـت. در مقابل، این پایان نامه از یک مدل اینترنت اشـیا استفاده می کند که عمومی است و می تواند در سایر حوزه های اینترنت اشیا مانند کارخانه هوشمند یا شهر هوشمند نیز اعمال شود.

مایر و همکاران رویکردی را نشـان می‌دهد که ابرداده‌های معنایی و استدلال را با مدل‌سازی گرافیکی ترکیب می‌کند، کـه در آن می‌توان اهداف یک محیط IoT را پیکربندی کرد، به عنوان مثال، دمای مورد نظر یک اتاق در طول روز. با این حال، این رویکرد فرض می‌کند که محیط IoT قبلاً مسـتقر شـده اسـت و اپراتورهای محاسـباتی در حال اجرا هسـتند. علاوه بر این، تعداد زیادی پلتفرم IoT وجود دارد FIWARE, GSN, IBM Watson IoT , OpenIoT OpenMTC و Microsoft Azure IoT .

با این حال، آنها ابزاری برای مدل‌سـازی و مدیریت محیط‌های کامل اینترنت اشـیا فراهم نمی‌کنند، به‌عنوان مثال، اشـیاء IoT به صـورت دسـتی و جداگانه در این پلت‌فرم‌های IoT ثبت و پیکربندی می‌شـوند. این پایان‌نامه از هسـتی‌شـناسی IoT-Lite برای مدل‌سازی محیط‌های اینترنت اشیا استفاده می‌کند و یک ابزار مدل‌سازی و مدیریت گرافیکی برای محیط‌های IoT، ابزار مدل‌سازی IoTEM ارائه می‌کند .

علاوه بر هسـتی‌شـناسـی IoT-Lite، فرمت‌ها و اسـتانداردهای دیگری نیز می‌توانند برای مدل‌سازی محیط‌های IoT، مانند هسـتی‌شـناسـی‌های SSN یا OneM۲M Base اسـتفاده شـوند. در این پایان نامه، بررسی ادبیات کاملی در مورد مدل های پیشـرفته اینترنت اشـیا انجام شده و در آن مقایسه ای مبتنی بر معیار از مدل های اینترنت اشیا ارائه شـده اسـت. در نتیجه این مقایسـه، هسـتی شناسی IoT-Lite بهترین مدل مناسب برای اهداف این پایان نامه نشان داده شده است.

علاوه بر این، می‌توان از مدل‌های عمومی‌تری که به‌طور خاص برای محیط‌های IoT طراحی نشده‌اند نیز استفاده کرد. برای مثال، اسـتاندارد توپولوژی و هماهنگ‌سـازی برای برنامه‌های کاربردی ابری TOSCA می‌تواند مورد استفاده قرار گیرد، که در اصـل برای برنامه‌های کاربردی ابری طراحی شـده اسـت، اما با این وجود ابزاری را برای مدل‌سـازی محیط‌های اینترنت اشیا و برنامه‌های کاربردی اینترنت اشیا ارائه می‌دهد.

۴-۲ مدل سازی پردازش جریان داده

این بخش به طور مفصل مدل پردازش جریان داده (DSPM) را توضیح می دهد. به دلیل خواندن مداوم حسگر و تبادل مکرر داده در بین اشیاء IoT مقادیر زیادی داده تولید می شود. این داده‌ها شکل جریان‌های داده‌ای را در خود جای می‌دهند که پایدار نیستند، بلکه برای پردازش در جریان‌های متعدد، پیوسته، سریع و متغیر با زمان می‌آیند. رویکردهای تثبیت شده خوبی برای پردازش جریان های داده، پردازش رویداد پیچیده و پردازش جریان است.

۴-۲-۱ تعریف DSPM

DSPM یک مدل مبتنی بر نمودار است که شامل منابع داده، مخزن داده، اپراتورهای پردازش و جریان داده بین اپراتورها است. این مدل بر اساس الگوی طراحی لوله‌ها و فیلترها (pipes and filters) است که ساختاری برای پردازش یک جریان داده فراهم می‌کند.

ورودی برای پردازش توسط منابع داده ارائه می شود، به عنوان مثال، حسگرها، که توالی مقادیر داده را در همان ساختار تولید می کنند. از طرف دیگر، خروجی پردازش به سینک های (sinks) داده می رسد. اپراتور پردازش مربوط به یک فیلتر (filters) است، و جریان داده بین دو اپراتور پردازش مجاور مربوط به یک لوله (pipes) است.

علاوه بر این، اتصالات بین منابع داده و اپراتورهای پردازش و بین اپراتورهای پردازش و سینک های داده نیز با لوله ها مطابقت دارد. این پایان نامه از یک نوع لوله و فیلتر استفاده می کند که در آن یک اپراتور پردازش می تواند چندین ورودی و خروجی داشته باشد. DSPM به صورت زیر قابل تعریف می باشد:

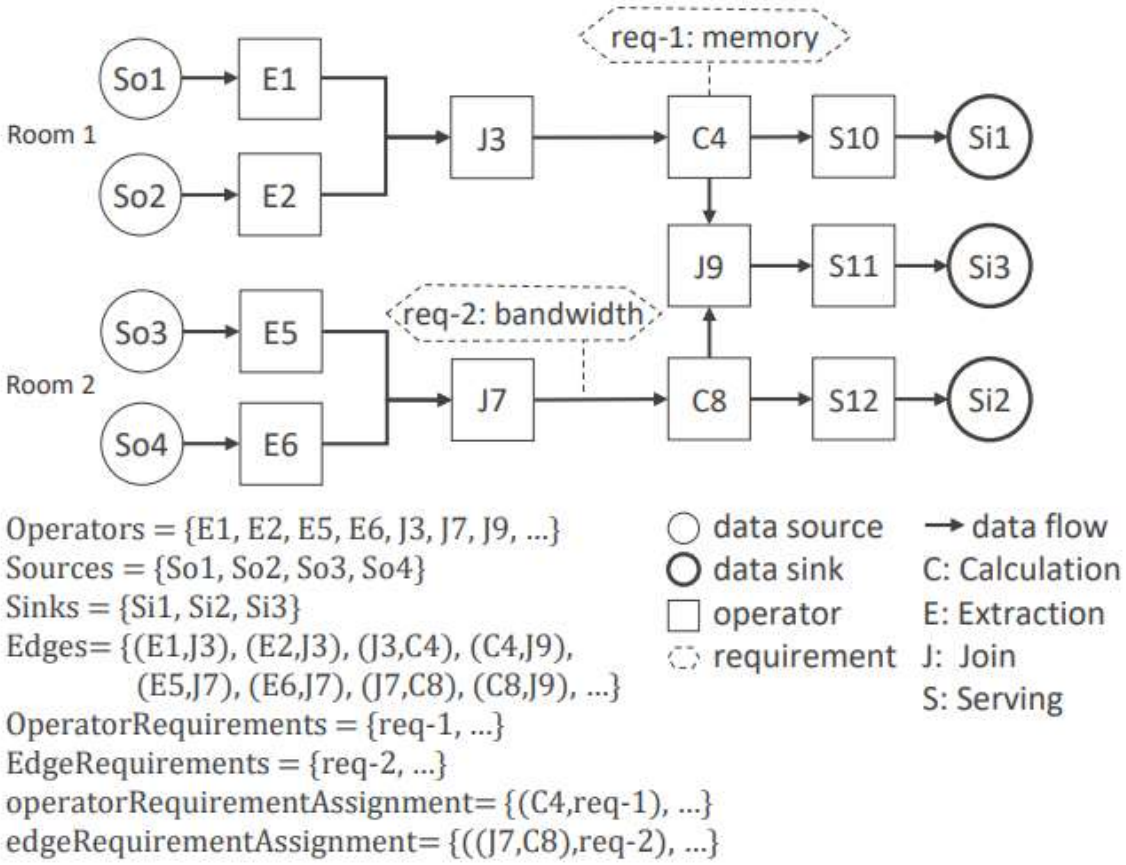
تعریف (DSPM) : یک مدل پردازش جریان داده چندگانه است: شامل (منابع داده (Sources)، سینک ها (Sinks)، اپراتورها (Operators) تمام پردازش ها شامل استخراج و سینک ها، لبه ها (Edges) ارتباط بین پردازش ها و پردازش و داده یا سینک، الزامات اپراتور (Operator Requirements)، الزامات لبه (Edge Requirements)، تخصیص نیاز اپراتور (operator Requirement Assignment)، تخصیص نیاز لبه (edge Requirement Assignment) ، که در آن (اپراتورها، لبه ها) یک گراف جهت دار، بدون وزن، بدون حلقه و غیر چرخه ای را تشکیل می دهد.

علاوه بر این، DSPM جریان داده را توصیف می کند و نه جریان کنترل. برای ایجاد صحیح DSPM، تحلیلگران دامنه باید بدانند که کدام منابع داده و سینک ها در محیط IoT در دسترس هستند. برای این کار، فقط باید توصیفات کلی از حسگرها و محرک ها از IoTEM استخراج شود تا بتوان آنها را به تحلیلگران حوزه به عنوان منابع داده و سینک انتزاع کرد. علاوه بر این، اپراتورها و اتصالات با الزامات محاسباتی مشروح می شوند.

بر اساس نیازهای محاسباتی اپراتورها، می توان به دنبال قرارگیری اپراتور مناسب در محیط IoT بود که تمام الزامات یک DSPM مدل شده را برآورده می کند. الزامات محاسباتی احتمالی آنهایی هستند که قابلیت‌های محاسباتی نمونه دستگاه‌های IoT را که در بخش ۴-۱-۲ توضیح داده شده است، برآورده می‌کنند. برای برخی نیازها، مانند حافظه اصلی مورد نیاز، تحلیلگران دامنه معمولاً نمی دانند از چه مقادیر مشخصی باید استفاده شود.

این دانش به شـدت به مقدار و پیچیدگی داده ها بسـتگی دارد و بنابراین، باید از طریق مقادیر تجربی استنباط شود. بنابراین، مقادیر تجربی عملگرها در حین اجرای آنها جمع آوری می شـود. سپس این مقادیر با استفاده از تکنیک‌های تحلیلی پردازش می‌شوند تا توصیه‌هایی در مورد الزامات در زمان مدل‌سازی به تحلیل‌گران حوزه ارائه شود.

شـکل زیر نمونه ای از یک DSPM را نشـان می دهد که شـامل منابع داده، سـینک ها و عملگرهای پردازشی حاوی نمایش های گرافیکی و رسمی است.



شکل ۴

هدف از برنامه نمونه، نظارت بر سـطوح قالب در اتاق‌های مختلف یک سـاختمان به منظور تشـخیص زمانی اسـت که سـطوح قالب به محدوده‌های ناسالم افزایش می‌یابد. منابع داده با سنسورهای دما (So۱، So۳) و سنسورهای رطوبت (So۲، So۴) نشـان داده می شـوند، در حالی که سـینک های داده توسـط داشبورد (Si۱، Si۲، Si۳) نشان داده می شـوند. علاوه بر این، این مثال از اپراتورها برای اسـتخراج داده‌های حسـگر، برای پیوستن به داده‌های حسگر بر اساس پنجره‌های زمانی، برای محاسـبه سـطوح قالب و ارائه داده‌ها به سینک‌های داده استفاده می‌کند. برای هر اتاق، سطح قالب به طور مداوم بر اسـاس مقادیر دما و رطوبت محاسـبه می شـود و در طول زمان به منظور تشـخیص افزایش ها تجزیه و تحلیل می شود.

۴-۲-۱ اپراتورهای پردازش

در این پـایان نامه، یک اپراتور پردازش یک جزء نرم افزاری اســـت که عملیاتی (به عنوان مثال، فیلتر کردن داده ها، تـابع میانگین) را بر روی یک یا چند جریان داده ورودی اجرا می کند. این اجزای نرم افزار می توانند به عنوان مثال، اسـکریپت های پایتون یا شل، فایل های JAR یا پرس و جوهای پردازش مداوم باشند. محیط اجرای نرم افزار مناسب (به عنوان مثال جاوا RE، مفســـر پایتون، موتور پردازش جریان) مورد نیاز یک اپراتور پردازشـــگر باید در DSPM به عنوان نیاز محاســـباتی این اپراتور مدل شـــود. اپراتورهای پردازش معمولاً به‌طور پیوســـته اجرا می‌شـــوند و براســـاس پنجره‌ها، یعنی فواصـــل زمانی داده‌ها بر اســـاس زمان یا تعداد عناصـــر، به دلیل ماهیت بی‌نهایت بودن جریان داده هستند.

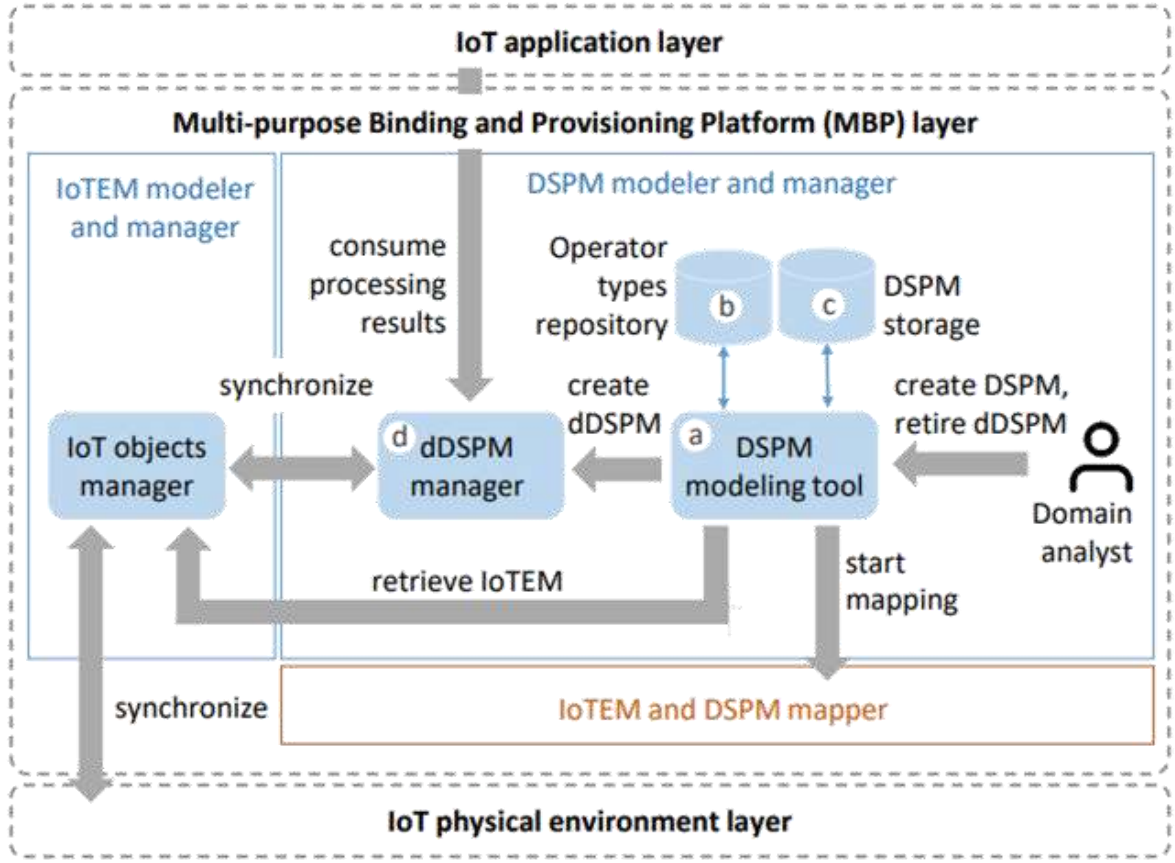
علاوه بر این، عملیات مبتنی بر پنجره ضـــروری اســـت، زیرا داده‌ها در محیط‌های IoT می‌توانند نادقیق باشـــند یا به ســـرعت کهنه شـــوند. خروجی یک اپراتور پردازشـــگر منجر به یک یا چند جریان داده می شـــود که می تواند به اپراتورهای پردازشگر بعدی یا به سینک های داده ارسال شود.

فرض بر این اسـت که اپراتورهای پردازش برای رویه‌های قرار دادن اپراتور آماده هستند، یعنی مکانیسم‌های لازم برای توزیع پردازش داده‌ها قبلاً به کار گرفته شـده اسـت. رویکردهایی که پردازش جریانی را در محیط های توزیع شـده و محیط های IoT انجام میدهند را می توان در لیست زیر نشان داد:

- تجمع (Aggregation): این عملگر یک تابع تجمیع را به مجموعه ای از مقادیر (به عنوان مثال، یک محیط اجرایی window) در یـک جریـان داده اعمـال می کنـد و یـک مقدار واحد را برمی گرداند. نمونه هایی از تجمیع محاسبات میانگین، حداقل مقدار و حداکثر مقدار هستند.
- محاســبه (Calculation). این پردازشگر یک محاسبه را بر اساس مقادیر یک یا چند جریان ورودی انجام می دهد. به عنوان مثال، عملگر محاســبه تابعی اسـت که سطوح قالب را بر اساس مقادیر دما و رطوبت محاسبه می کند.
- اســـتخراج (Extraction): این اپراتور داده‌هـا را از منـابع داده اســـتخراج می‌کند، برای مثال، می‌تواند کد نرم‌افزاری باشـد که به رابط فیزیکی یک حسـگر (مثلاً از طریق GPIO) برای دریافت اندازه‌گیری‌های واقعی حسگر دسترسی دارد.
- فیلتر (Filter). این عملگر داده ها را بر اساس پارامترها فیلتر می کند.
- پیوستن (Join). این اپراتور دو جریان داده را پیوند می دهد.
- تشـخیص الگو (Pattern detection). این اپراتور الگوهای خاصـی را در یک جریان داده تشخیص می دهد. نمونه ای از یک الگوی مشـــتق شـــده از مفاهیم CEP یک توالی رویداد اســـت، به عنوان مثال، رویدادهای مشخص شده در یک ترتیب خاص رخ می دهد.
- خدمت (Serving). این اپراتور داده‌ها را به سـینک‌های داده ارائه می‌کند، برای مثال، می‌تواند کد نرم‌افزاری باشـد که به رابط فیزیکی یک محرک برای کنترل آن دسـترسی دارد، یا کد نرم‌افزاری باشد که داده‌ها را به یک برنامه داشبورد ارسال می‌کند.
- دگرگونی (Transformation). این اپراتور تبدیل ها را متوجه می شـــود، به عنوان مثال، در بین قالب های مختلف داده یا طرحواره های داده.

۴-۲-۲ ساختار معماری و پیاده سازی – مدل ساز و مدیر DSPM

در معمـاری کلی (به بخش ۳,۳ رجوع کنید)، مؤلفه مدلســـاز و مدیر DSPM از تحلیلگران دامنه با ارائه ابزاری برای ایجاد و ذخیره DSPMها و علاوه بر این، مدیریت DSPMهای مســتقر در محیط های IoT پشــتیبانی می کند. این مشخصات در شکل زیر نشان داده شده است:



شکل ۵

این پایان نامه از ابزار FlexMash (Flexible Modeling and Execution of Data Mashups) که توسـط Hirmer و همکاران، به عنوان ابزار مدل سازی DSPM (بخش a) برای ایجاد DSPMs پیشنهاد شد.

این یک رابط گرافیکی ارائه می‌کند و مدل زیربنایی مبتنی بر JSON آن را می‌توان برای فعال کردن حاشـیه‌نویسـی نیـازمنـدی‌های محاســـباتی در گره‌ها و لبه‌ها گســـترش داد، که مقدمه‌ای برای مفاهیم این پایان‌نامه اســـت. ابزار مدل‌سـازی DSPM با کشـیدن و رها کردن منابع داده، سـینک‌های داده و اپراتورهای پردازش به منطقه مدل‌سازی، ایجاد DSPM را امکان‌پذیر می‌سازد. در حالی که اپراتورها در مخزن نوع اپراتور (بخش b) موجود هستند، منابع داده و سـینک ها از حسـگرها و محرک های موجود در مدل های IoTEM که از مدیر اشـیاء IoT بازیابی می شـوند، انتزاع می شـوند. علاوه بر این، DSPM مدل‌سـازی‌شده در ذخیره‌سازی DSPM (بخش c) ذخیره می‌شود. از طریق ابزار مدل سـازی DSPM، نگاشت یک DSPM بر روی یک IoTEM آغاز می شود و متعاقبا، استقرار بر اساس

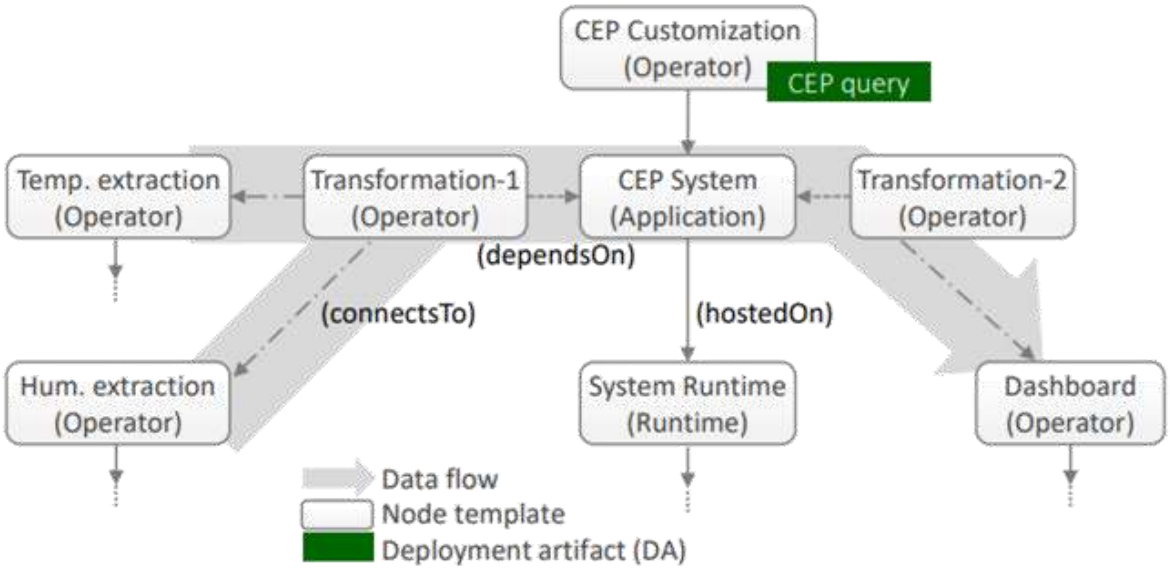
نقشه نگاشت ایجاد شده نیز آغاز می شود. هنگامی که یک DSPM با موفقیت مستقر شد و توسط مؤلفه Deployment Manager شروع شد (به فصل ۶ مراجعه کنید)، مدیر dDSPM از موفقیت آمیز بودن استقرار مطلع می شود و یک dDSPM مربوطه ایجاد می کند که حاوی اطلاعاتی است در مورد اینکه کدام شی IoT در حال پردازش کدام اپراتور است.
مدیر dDSPM دائماً اطلاعات وضعیت را از مدیر اشیاء IoT بازیابی می کند تا مثلاً بینشی از سلامت اپراتورهای مستقر و اشیاء موجود IoT بدست آورد. این اطلاعات را می توان در داشبورد MBP مشاهده کرد. در نهایت، ابزار مدل‌سازی تحلیلگران دامنه را قادر می‌سازد تا dDSPMها را توسط مدیر dDSPM متوقف نماید.

۴-۲-۴ کارهای مرتبط

رویکردهای زیادی برای مدل‌سازی جریان داده و پردازش داده در محیط اینترنت اشیا وجود دارد مانند COMPOSE Huginn, IFTTT, LabVIEW, Node-RED, WoTKit, WoT Flow یا Yahoo Pipes! .

با این حال، آنها یا فرض می‌کنند که پردازش به صورت متمرکز انجام می‌شود یا اینکه اپراتورها و خدمات از قبل در محیط‌های IoT در حال اجرا هستند. یک مرور ادبیات کامل بر روی مدل‌های جریان داده پیشرفته توسط Hirmer انجام شد، که رویکردی را برای مدل‌سازی و اجرای جریان‌های داده مبتنی بر نیاز پیشنهاد کرد. در این کار، ابزار FlexMash توسعه داده شده است، که به عنوان پایه ای برای این پایان نامه در رابطه با مدل سازی پردازش جریان داده عمل می کند. مدل‌های عمومی بیشتری که به‌طور خاص برای پردازش داده‌های مدل طراحی نشده‌اند نیز می‌توانند مورد استفاده قرار گیرند، به عنوان مثال، TOSCA .

نحوه مدل‌سازی پردازش رویداد پیچیده در TOSCA را بررسی می‌کنیم و پس از آن از مدل TOSCA حاصل برای استقرار و شروع پردازش استفاده می‌کنیم. در شکل زیریک مدل توپولوژی انتزاعی TOSCA برای پردازش رویداد پیچیده نشان داده شده است.



شکل ۶

در این شـکل چندین نوع عملگر مدل شـده اسـت. به عنوان مثال، شـامل اپراتورهایی اسـت که مقادیر سنسور دما و رطوبت را اسـتخراج می کنند و متعاقباً، یک اپراتور این مقادیر را به قالبی تبدیل می کند که توسـط سـیسـتم CEP قابل درک باشد. بسیاری از پلتفرم‌های IoT توسعه یافته‌اند که شامل یک لایه پردازش داده می‌باشند با این حال، آنها یا ابزاری برای مدل‌سـازی پردازش داده‌ها به روشی کاربرپسند ارائه نمی‌کنند (مثلاً، مدل‌سازی با ورودی متنی محقق می‌شـود) یا فقط مدل‌سازی قوانین ساده (مثلاً بیانیه‌های if) پشتیبانی می‌شود. . علاوه بر این، آنها فرض می‌کنند که پردازش به‌طور متمرکز انجام می‌شـود، به عنوان مثال، توسـط خود پلتفرم IoT و به این ترتیب از مفاهیم قرار دادن اپراتور پشتیبانی نمی‌کنند.

فصل پنجم

ارتباط بین DSPM ها در مدل های اینترنت اشیاء (IoTEM)

۵-۱ رویکرد خودکار

هدف این رویکرد این است که مدل به طور خودکار تصمیم بگیرد که کدام اشیاء IoT باید توسط کدام اپراتورها پردازش شود. این امر با استفاده از الگوریتم‌هایی محقق می‌شود که قادر به ارزیابی و مطابقت با قابلیت‌های کلی IoTEM و الزامات DSPM هستند. در این پایان نامه، به این مسئله به عنوان مسئله قرار دادن اپراتور اشاره می شود. مشکل توزیع عملگرها NP-complete است، الگوریتم‌های ابتکاری، مانند الگوریتم‌هایی که توسط V.M. Lo ارائه شده‌اند، معمولاً برای حل این نوع مسائل استفاده می‌شوند. بر اساس DSPM و IoTEM، مشکل قرار دادن اپراتور ها به این صورت قابل تعریف می باشد:

اجزای IoTEM : Objects, Connections, Distance, Object Capabilities, Connection Capabilities, Object Capability Assignment, Connection Capability Assignment که پیش تر بیان شد.

اجزای DSPM : Operators, Sources, Sinks, Edges, Operator Requirements, Edge Requirements, Operator Requirement Assignment, edge Requirement Assignment نیز پیش تر بیان شد.

راه حل : (operator Mapping, edge Mapping) که در آن :

Source Mapping: Sources → Objects

ارتباطی که منابع داده DSPM را به اشیاء اختصاصی IoT در IoTEM اختصاص می دهد.

Sink Mapping: Sinks → Objects

ارتباطی که سینک های داده DSPM را به اشیاء اختصاصی IoT در IoTEM اختصاص می دهد.

Operator Mapping: Operators → Objects

ارتباطی که اپراتورهای محاسباتی DSPM را به اشیاء IoT IoTEM اختصاص می دهد.

Edge Mapping: Edges → Paths (IoTEM):

ارتباطی که لبه هایی را در DSPM به مسیرهای اتصال IoTEM اختصاص می دهد.

در این پایان نامه، دو الگوریتم برای حل مسئله قرارگیری اپراتور DSPM ارائه شده است: (i) یک روش حریصانه و یک (ii) روش عقبگرد که نوع حریصانه راه حل را به موقع پیدا می کند، با این حال، یافتن بهترین راه حل ممکن را تضمین نمی کند.

Lo نشـان می دهد که یک الگوریتم حریصـانه سـاده بسـیار کارآمدتر اسـت از الگوریتم های ابتکاری پیچیده تر. در مقابل، الگوریتم های عقبگرد تمام راه حل های ممکن را محاســبه می کند، بنابراین، می تواند بهترین راه حل ممکن را انتخاب کند. با این حال، نوع عقبگرد به زمان اجرای بالاتری نیاز دارد و بنابراین باید برای سـناریوهای ساده ای که شامل اشیاء IoT کمتری هستند استفاده شود.

هدف اصــلی هر دو الگوریتم یافتن مجموعه‌ای از اشــیاء IoT اســت که قابلیت‌های آنها نیازهای اپراتورها را برآورده می‌کند. قراردادن اشــیاء IoT نزدیک به منابع داده ترجیح داده می شــوند، زیرا تشــخیص زودهنگام موقعیت های بحرانی و اقدامات به موقع مربوطه را ممکن می ســازد. علاوه بر این، حجم داده‌های مبادله شــده در محیط اینترنت اشـیا را می‌توان با جمع‌آوری و فیلتر کردن داده‌ها در نزدیکی منابع آنها و در اولین فرصت ممکن در زنجیره پردازش داده کاهش داد.

برای هر دو الگوریتم، فرض بر این اسـت که اشـیاء IoT که از طریق رابط‌های سـخت‌افزاری به حسـگرها یا محرک‌ها متصـل هستند، قابلیت‌های محاسباتی کافی را برای اجرای استخراج داده‌های حسگر یا اپراتورهای ارائه داده را دارند. به این معنا که یک اپراتور اسـتخراج داده همیشـه می تواند مسـتقیماً روی شی IoT قرار گیرد که به طور فیزیکی به حسگر مربوطه متصل است.

۵-۱-۱ الگوریتم تطبیق (حریصانه)

الگوریتم در شــبه کد نشــان داده شــده که IoTEM و DSPM را به عنوان ورودی نیاز دارد. خروجی نقشــه ای از عملگرهـا و لبه های DSPM را بر روی اشــیاء IoT و اتصــالات آنها به IoTEM برمی گرداند. در ادامه الگوریتم به صورت گام به گام شرح داده شده است.

```
١. function GreedyMatching((Sources, Sinks,Operators, Edges, ...), (Objects, Connections, ...))
٢.   DSPM ← (Sources, Sinks,Operators, Edges, ...)
٣.   IoTEM ← (Objects, Connections, ...)
٤.   operatorMapping : Operators → Objects
٥.   edgeMapping : Edges → Paths(IoTEM)
٦.   order ← GetTopologicalOrder(Operators, Edges)
٧.   sourceMapping ← FillSourceMapping(IoTEM, DSPM)
٨.   sinkMapping ← FillSinkMapping(IoTEM, DSPM)
٩.   i ← ٠
١٠.  while i < |Operators| do
١١.    opj ← order(i)
١٢.    if (opj .isConnectedToSource()) then
١٣.      opObject ← sourceMapping(opj)
١٤.      ConsumeCaps(operatorRequirementAssignment(opj), objectCapabilityAssignment(opObject))
١٥.      operatorM apping(opj ) ← opObject
١٦.    else if (opj .isConnectedToSink()) then
١٧.      opObject ← sinkMapping(opj )
١٨.      ConsumeCaps(operatorRequirementAssignment(opj), objectCapabilityAssignment(opObject))
١٩.      operatorMapping(opj ) ← opObject
٢٠.    else
٢١.      PreOperators ← {opi ∈ Operators : ∃(opi , opj ) ∈ Edges}
```

```
22.     PreObjects ← operatorMapping(PreOperators)
23.     closestOrder ← GetClosestObjects(IoTEM, PreObjects)
24.     foundObject ← false
25.     j ← ۰
26.     while ( j < |Objects| & foundObject) do
27.       opObject ← closestOrder(j)
28.       if TryMapOperator(opj , opObject, operatorMapping, edgeMapping, DSPM, IoTEM) then
29.         foundObject ← true
30.       end if
31.     end while
32.     if foundObject then
33.       return "No solution found"
34.     end if
35.   end if
36.  end while
37.  return (operatorM apping, edgeMapping)
38. end function
```

در خط ۶ تابع GetTopologicalOrder فراخوانی می شود که DSPM را به همراه Operators و Edge هایش بـه عنوان ورودی می گیرد. این تـابع یـک ترتیـب نقشـــه برداری را برمی گرداند: {۰، ۱، . . . ، که با مرتب ســـازی توپولوژیکی عملگرها مطابقت دارد.

در خط ۷، تـابع FillSourceMapping نـامیـده می شـــود که DSPM و IoTEM را به عنوان ورودی می گیرد و نقشــه ای از منابع انتزاعی شده از حسگرها و اشیاء اختصاصی اینترنت اشیا متصل به این حسگرها را برمی گرداند. به طور مشــابه، تابع FillSinkMapping در خط ۸ نقشــه‌ای از ســینک‌های انتزاعی از محرک‌ها و اشــیاء اختصــاصــی اینترنت اشیا متصل به این محرک‌ها را برمی‌گرداند.

در مرحله بعد، لیســت مرتب شده اپراتورها پیمایش می شود. اگر اپراتور فعلی به یک گره منبع داده متصل باشد، شی IoT متصــل به منبع داده (یعنی حســگر) بازیابی می شود (خط ۱۳). سپس قابلیت های این شی اینترنت اشیا با کم کردن منابع مورد نیاز اپراتور تنظیم می شود (خط ۱۴).

در نهایت، اپراتور بر روی شــی IoT (خط ۱۵) نگاشت می شود. سپس، حلقه با عملگر بعدی ادامه می یابد. اگر اپراتور فعلی به یک گره ســینک داده (خط ۱۶) متصــل باشد، مانند منبع داده همانطور که در بالا توضیح داده شد، مدیریت می شــود. در خط ۲۱، لیستی از اپراتورهای پیشین اپراتور فعلی در DSPM تعیین می شود و متعاقبا، اشیاء اینترنت اشیا میزبان این اپراتورهای قبلی نیز بازیابی می شوند (خط ۲۲).

تابع GetClosestObjects همه اشیاء IoT را در IoTEM (خط ۲۳) بر اساس فاصله شبکه آنها با اشیاء IoT قبلی مرتب می کند. نتیجه این است که نقشه به ترتیب بسته می‌شود: {۰، ۱، . . . ، } در حالی که closestoOrder(۰) به شی IoT نزدیک ترین به اشیاء IoT قبلی اشاره دارد. اشیاء قبلی IoT نیز بخشی از این مرتب سازی هستند و ابتدا ظاهر می شــوند زیرا فاصــله شــبکه آنها ۰ اســت. پس از آن، لیســت نزدیکترین اشــیاء IoT طی می شود (خط ۲۶) و بررســی می شــود که آیا می توان اپراتور را روی یکی از آنها نگاشت کرد یا خیر. سپس چک کردن نزدیک ترین اشیاء اینترنت اشیا بسته به قابلیت های موجود آنها.

اگر نگاشـــت امکان پذیر باشـــد، به عنوان مثال، تابع TryMapOperat مقدار true را برمی گرداند (خط ۲۸)، یک شـی IoT پیدا شـد و اپراتور با موفقیت بر روی شـی IoT یافت شده نگاشت شد. در غیر این صورت، نزدیکترین شی IoT بعدی بررسـی می شـود. اگر هیچ شـیء IoT قابلیت های لازم را نداشته باشد، هیچ راه حلی نمی توان یافت. در این حالت، الگوریتم از یک زیرسـاخت ابری به عنوان یک بازگشـت اسـتفاده می‌کند تا از یک راه‌حل اطمینان حاصل شود. در نهایت، الگوریتم مجموعه ای از نگاشت عملگرها و اشیاء را برمی گرداند (خط ۳۷).

نوع حریصـانه راه حلی را ارائه می دهد، با این حال، ممکن اسـت بهترین راه حل در مورد حداقل فاصـله شـبکه بین اپراتورها نباشد.

۵-۱-۲ الگوریتم تطبیق (روش عقبگرد)

هدف این الگوریتم نه تنها یافتن یک راه حل، بلکه بهترین راه حل اسـت. ورودی آن مانند الگوریتم قبلی است، یعنی IoTEM و DSPM. نمونه شبه کد آن به این صورت است:

```
۱.  function BACKTRACKING MATCHING(Sources, Sinks, Operators, ...), (Objects, Connections, ...))
۲.    DSPM ← (Sources, Sinks, Operators, ...)
۳.    IoTEM ← (Objects, Connections, ...)
۴.    operator Mapping : Operators →Objects
۵.    edgeMapping: Edges → Paths(IoTEM)
۶.    solution operator Mapping, edgeMapping)
۷.    source Mapping ← FILLSOURCEMAPPING(IOTEM, DSPM)
۸.    sink Mapping ← FILLSINKMAPPING(IOTEM, DSPM)
۹.    RestOperators ← Operators
۱۰.   for op € RestOperators do
۱۱.     if (op.isConnectedToSource() then
۱۲.       opObject ← sourceMapping(op)
۱۳.       CONSUMECAPS(operator RequirementAssignment(op) objectCapability Assignment(opObject))
۱۴.       operator Mapping(op) ← opObject
۱۵.       RestOperators ←  RestOperators \ op
۱۶.     else if (op.isConnected ToSink) then
۱۷.       opObject ←  sink Mapping(op)
۱۸.       CONSUMECAPS(operator RequirementAssignment(op) objectCapability Assignment(opObject))
۱۹.       operator Mapping(op) ← opObject
۲۰.       RestOperators ← Rest Operators \ op
۲۱.     end if
۲۲.   end for
۲۳.   Solutions ← Φ
۲۴.   found Solution ←  FINDSOLUTION(solution, RestOperators, Edges, Solutions, DSPM, IOTEM)
۲۵.   if (Solutions ←  found Solution) then
۲۶.     return "No solution found"
۲۷.   end if
۲۸.   return GETBESTSOLUTION(Solutions, Edges, distance)
۲۹.  end function
```

در خطوط ۱۱-۱۵، تمام اپراتورهای متصـل به منابع DSPM پیمایش می شـوند، اشـیاء مربوط به IoT بازیابی می شـوند و عملگرهای اسـتخراج داده بر روی این اشـیاء IoT نگاشت می شوند. پس از آن عملگرهای استخراج از لیست عملگرها حذف می شوند.

در خطوط ۱۶-۲۱، تمام اپراتورهای متصل به سینک ها نیز پیمایش می شوند، اشیاء IoT مربوطه بازیابی می شوند و اپراتورهای ارائه دهنده داده بر روی این اشـیاء IoT نگاشـت می شـوند. پس از آن اپراتورهای سـرویس دهنده نیز از لیسـت اپراتورها حذف می شـوند. سـپس تابع FindSolution فراخوانی می شود (خط ۲۴) که شامل منطق تطابق نیازمندی ها و قابلیت ها است. کد این تابع به صورت زیر می باشد.

```
۱. function FindSolution(solution, Opera tors, Edges, Solutions, DSPM, IoTEM)
۲.  if Operators≠ Φ then
۳.  foundSolution ← false
٤.  nextOperator ← GetOneElement(Operators)
٥.  for currObject ∈ Objects do
٦.  if SatisfiesReqs(operatorRequirementAssignment(nextOperator),
    objectCapabilityAssignment(currObject)) then
۷.  ConsumeCaps(operatorRequirementAssignment(nextOperator),
    objectCapabilityAssignment(currObject))
۸.  operatorMapping(nextOperator) ← currObject
۹.  Operators ← Operators \ {nextOperator}
۱۰.  if FindSolution(solution,Operators, Edges, Solutions, DSPM, IoTEM) then
۱۱.  foundSolution ← true
۱۲.  return true
۱۳.  else
۱۴.  Undo(operatorRequirementAssignment(nextOperator),
    objectCapabilityAssignment(currObjec t))
۱۵.  operatorMapping(nextOperator) ← null
۱۶.  Operators ← Operators ∪ {nextOperator}
۱۷.  end if
۱۸. end if
۱۹. end for
۲۰. return foundSolution
۲۱. else
۲۲. if Ed ges≠ Φ then
۲۳.  // similar to operators
۲۴. else
۲۵.  Solutions ← Solutions ∪ {solution}
۲۶.  return true
۲۷. end if
۲۸. end if
۲۹. end function
```

این تابع FindSolution، یک جستجوی عمقی را برای راه حل های ممکن با استفاده از بازگشت انجام می دهد. در خط ۵ بررسی می شود که الگوریتم تا چه حد پیشرفت کرده است. اگر مجموعه اپراتورها همچنان حاوی عناصر باشد، ابتدا این عناصر بر روی اشیاء IoT نگاشت می شوند.

در خط ۳، متغیر foundSolution تعریف شده است که به خاطر می‌سپارد که آیا راه‌حلی پیدا شده است یا خیر.

در خط ۴، تابع GetOneElement عملگر فعلی را برمی گرداند. پس از آن، اشیاء IoT که برای اجرای این اپراتور مناسب هستند با تکرار تمام اشیاء IoTEM جستجو می شوند. توجه به این نکته ضروری است که ترتیب پردازش اشیاء IoT برای هر تکرار یکسان است. در غیر این صورت، دیگر امکان عقب نشینی وجود ندارد، یعنی بعداً مراحل نقشه برداری را لغو کنید و به دنبال جایگزین باشید.

در خط ۶، بررسی می شود که آیا دستگاه فعلی نیازهای اپراتور nextOperat را برآورده می کند یا خیر. اگر اینطور نباشد، شیء IoT بعدی در نظر گرفته می شود. اگر یک شی IoT مطابقت داشته باشد، اپراتور از مجموعه اپراتورهایی که هنوز نیاز به نقشه برداری دارند حذف می شود (خط ۱۲). سپس تابع FindSolution به صورت بازگشتی فراخوانی می شود.

با انجام این کار، ضمن ارتباط هر اپراتور با هر دستگاه اینترنت اشیا، تمام راه حل های ممکن پیدا می شود. این منجر به زمان اجرا بالاتر در مقایسه با الگوریتم حریصانه می شود. در نتیجه، برای ایجاد نقشه برداری در زمان اجرا یا برای برنامه هایی که نیاز به به روز رسانی مداوم نقشه برداری از اپراتورها دارند، الگوریتم عقبگرد توصیه نمی شود. نگاشت لبه ها مشابه نگاشت اپراتور است که در بالا توضیح داده شد. با این حال، در این مورد، تمام لبه‌ها به عنوان راه‌حل‌های ممکن بر اساس نگاشت موجود از اپراتورها روی دستگاه‌ها بررسی می‌شوند. اگر مجموعه عملگرها و مجموعه یال ها هر دو خالی باشند، همه آنها نگاشت شده و یک راه حل پیدا شد. در این حالت راه حل پیدا شده به مجموعه Solutions (خط ۲۵) اضافه می شود.

پس از خاتمه تابع FindSolution، تابع GetBestSolution در خط ۲۸ در الگوریتم قبلی بهترین راه حل مجموعه کل راه حل ها را تعیین می کند. این تابع تمام راه حل های یافت شده را طی می کند و مجموع فواصل مسیرهای موجود را محاسبه می کند و راه حل با کمترین فاصله بهترین راه حل در نظر گرفته می شود.

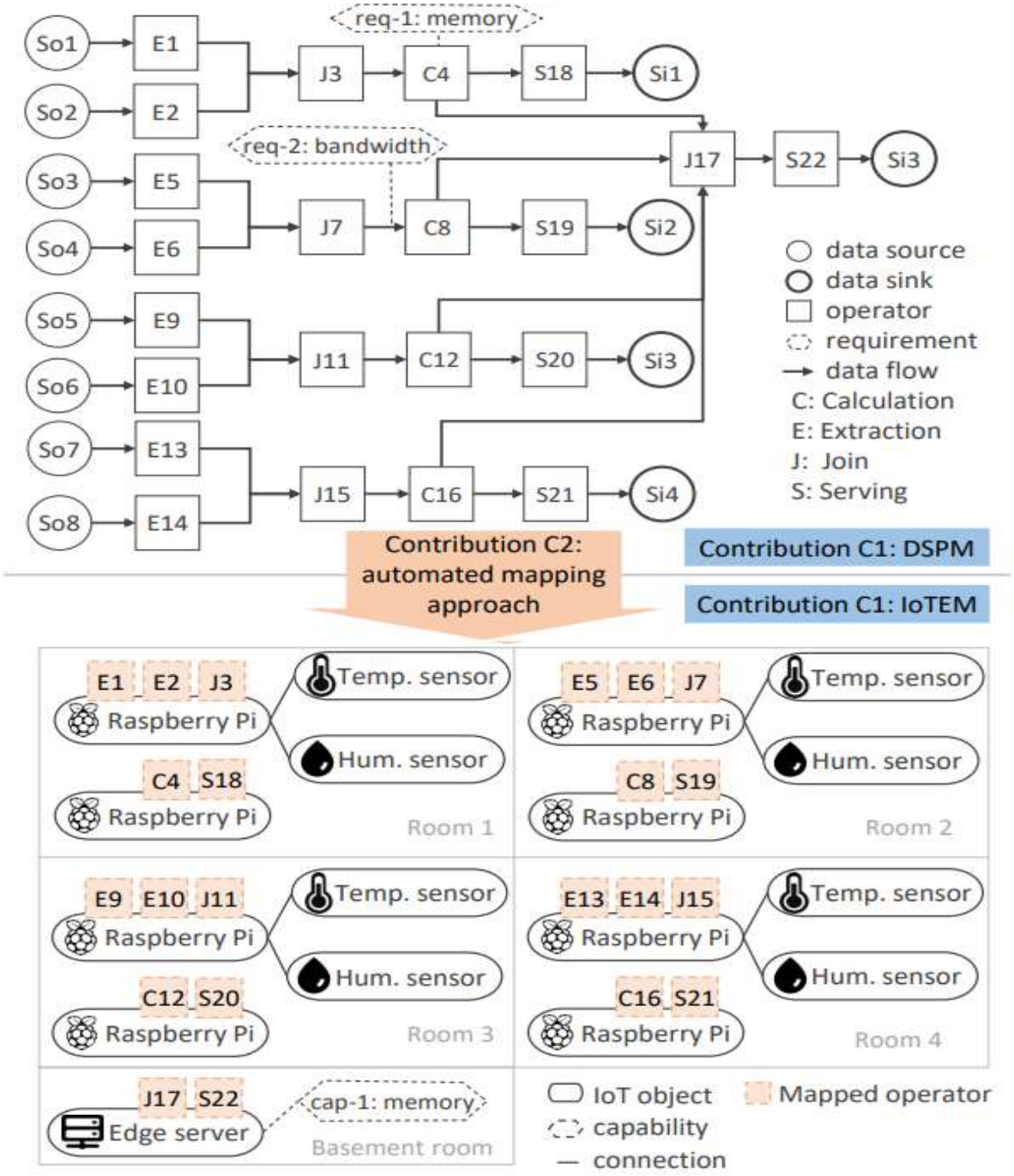
از آنجایی که الگوریتم عقبگرد تمام راه حل های ممکن را محاسبه می کند، زمان اجرا آن نمایی می شود. بنابراین، الگوریتم عقبگرد برای محیط های کوچک اینترنت اشیا و مدل های پردازش جریان داده توصیه می شود. اگر یک شی IoT از زیرساخت ابری در IoTEM مدل شود، الگوریتم از آن به عنوان راه حل بازگشتی استفاده می کند و میتواند یک راه حل مطمئن باشد. در غیر این صورت، الگوریتم ها ممکن است راه حلی را برنگردانند.

برای بهینه‌سازی الگوریتم‌های معرفی‌شده، می‌توان روش ابتکاری را در نظر گرفت که یک نقشه بزرگ از عملگرهای خاص را از پیش تعریف می‌کند. الگوریتم‌های معرفی‌شده یک روش ابتکاری را در نظر می‌گیرند، که اپراتورهای منبع را بر روی شی IoT که به حسگر مربوطه برای استخراج داده‌ها از رابط‌های سخت‌افزار متصل است، نگاشت می‌کند.

اگر مشخص باشد که برخی از حسگرها مقدار زیادی داده تولید می کنند، به عنوان مثال، جریان های ویدئویی، اپراتورهایی که این داده ها را پردازش می کنند، می توانند از قبل روی اشیاء قدرتمند IoT نگاشت شوند، در حالی که اشیاء IoT با منابع کم در نظر گرفته نمی شوند.

۵-۱-۳ سناریوی موردی: نظارت بر سطوح قالب در ساختمان های هوشمند

در ادامه، یک سـناریوی در حوزه ساختمان هوشمند و نحوه استفاده از مفاهیم نگاشت برای اجرای آن ارائه می کنیم. سـاختمان هوشمند در این پایان نامه به عنوان ساختمانی مجهز به محاسبات و فناوری اطلاعات تعریف شده است که نیازهای سـاکنان خود را برای تامین زندگی راحت آنها تامین میکند. هدف این سناریوی ، نظارت بر اتاق‌های مختلف یک سـاختمان به منظور تشـخیص زمانی است که به محدوده‌های ناسالم افزایش می رسند. این سناریو در شکل زیر نشان داده شده است.



شکل ۷

مطابق شکل برای هر اتاق، سطح قالب به طور مداوم بر اساس مقادیر دما و رطوبت اندازه گیری شده زنده محاسبه می شود. سطوح قالب در طول زمان تجزیه و تحلیل می شوند تا هر گونه افزایش را تشخیص دهند.
ساختمان هوشمند این مطالعه از چهار اتاق و یک اتاق دیگر در زیرزمین تشکیل شده است. هر اتاق دارای دو دستگاه Raspberry Pis به عنوان IoT است که یکی از آنها به یک سنسور دما و یک سنسور رطوبت متصل است. Raspberry Pi دوم هر اتاق در صورت لزوم منابع محاسباتی اضافی را فراهم می کند. علاوه بر این، اتاق زیرزمین مجهز به یک سرور لبه به عنوان یک دستگاه اینترنت اشیا است که نسبت به دیگر Raspberry Pis در ساختمان هوشمند دارای قابلیت های محاسباتی بیشتری است. دستگاه های IoT درگیر به یک شبکه متصل هستند، یعنی میتوانند از طریق پروتکل های استاندارد اینترنت با یکدیگر ارتباط برقرار کنند. در این سناریو، اپراتورهای DSPM باید بین دستگاه‌های IoT موجود در ساختمان توزیع شوند. نوع حریصانه الگوریتم تطبیق به منظور تصمیم گیری برای استقرار عملگرها استفاده می شود.
در مرحله اول، اپراتورهای استخراج، E۱ تا E۱۴، بر روی دستگاه‌های IoT مستقر می‌شوند، که حسگرها به طور فیزیکی به آن متصل می‌شوند (همانطور که در شکل نشان داده شده است). این عملگرها شامل اسکریپت هایی برای استخراج داده های حسگر و ارسال آنها به اپراتور پردازشگر بعدی هستند. بنابراین، این اپراتورهای استخراج باید مستقیماً روی دستگاه اینترنت اشیا متصل به سنسورها مستقر شوند. بنابراین، فرض بر این است که منابع کافی در این دستگاه ها موجود است.
پس از نگاشت اپراتورهای استخراج، منابع مورد نیاز آنها از قابلیت های دستگاه های IoT منتخب کم می شود تا در مرحله بعدی تصمیم گیری شود که آیا عملیات اضافی را می توان بر روی این دستگاه های IoT نگاشت یا خیر. پس از آن، مشابه عملگرهای استخراج، اپراتورهای سرویس دهنده S۱۸، S۱۹، S۲۰ و S۲۱ نگاشت می شوند.
بعد از اینکه عملگرهای سرویس نگاشت شدند، عملگرهای Join در مرحله بعد نگاشت می شوند. برای J۳ که به داده های خروجی از اپراتورهای استخراج E۱ و E۲ می پیوندد، نزدیکترین دستگاه به اپراتورهای استخراج مستقر شده در مدل محیط اینترنت اشیا جستجو می شود. بدیهی است که نزدیکترین دستگاه همان دستگاهی است که اپراتورهای استخراج روی آن مستقر هستند. با فرض اینکه این دستگاه الزامات متصل به عملیات اتصال J۳ را برآورده می کند و قابلیت های کافی باقی مانده است، بنابراین، اپراتور Join J۳ می تواند بر روی همین دستگاه مستقر شود.
همین رویه برای بقیه عملگرهای Join J۷، J۱۱ و J۱۵ انجام می شود. پس از آن بار دیگر منابع مورد نیاز از قابلیت های دستگاه ها کم می شود. سپس عملگرهای محاسباتی C۴، C۸، C۱۲ و C۱۶ نگاشت می شوند. برای C۴، دوباره، نزدیکترین دستگاه در مدل محیط اینترنت اشیا به اپراتور قبلی خود (یعنی اپراتور نگاشت شده J۳) جستجو می‌شود. این دستگاهی است که قبلاً شامل عملگرهای نقشه‌برداری شده E۱، E۲ و J۳ است.
با این حال، با فرض اینکه محاسبه بر اساس یک الگوریتم پیچیده برای محاسبه سطح قالب مصرف کننده منابع بیشتر باشد، این دستگاه با الزامات اپراتور C۴ مطابقت ندارد. در نتیجه نزدیک‌ترین دستگاه بعدی جستجو می‌شود که در همان اتاق دستگاه حاوی E۱، E۲ و J۳ است. با فرض اینکه این دستگاه قابلیت محاسباتی کافی برای محاسبه سطح قالب را دارد، اپراتور C۴ بر روی این دستگاه نگاشت شده و منابع آن مصرف می شود. در اتاق‌های باقی‌مانده، عملگرها به طور مشابه نگاشت می‌شوند.

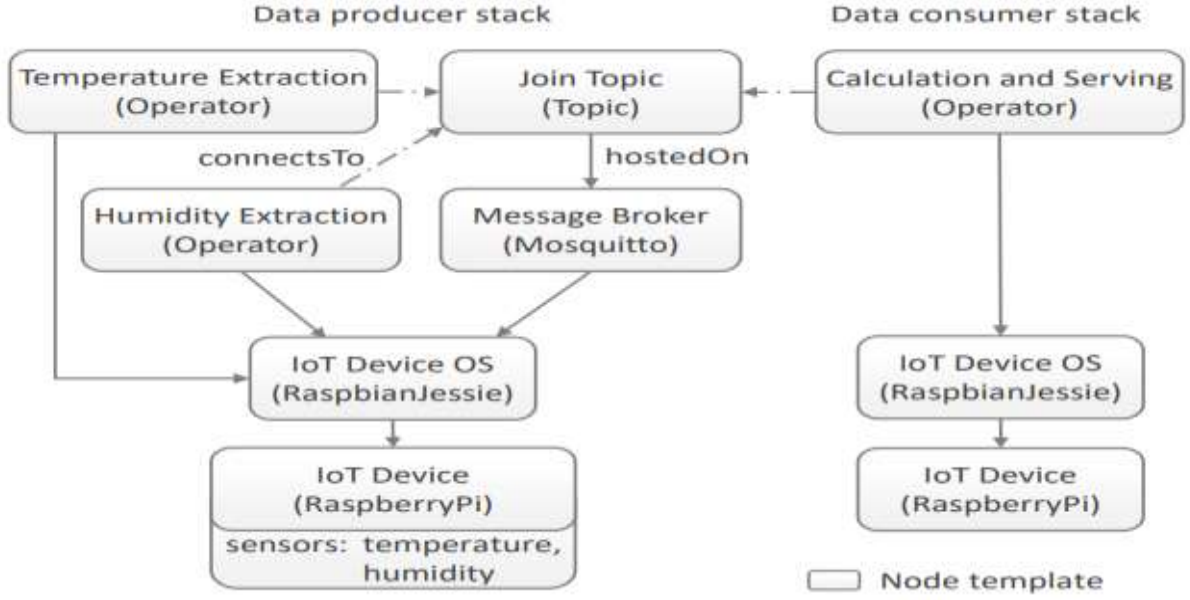
در نهایت، سطح قالب محاسبه‌شده هر چهار اتاق با استفاده از عملگر J۱۷ جمع‌آوری می‌شود، که داده‌های خروجی مناسب برای تجسم داشبورد برای کاربران تولید می‌کند. در این مثال، دستگاه‌های اینترنت اشیا در اتاق‌های مختلف فاصله شبکه یکسانی با یکدیگر دارند. در نتیجه، الگوریتم آنها را به طور تصادفی انتخاب می کند. در مرحله بعد، بررسی می شود که آیا یک دستگاه IoT می تواند الزامات اپراتور J۱۷ را برآورده کند یا خیر. اگر نه، دستگاه های دیگر را بررسی می کند. در این مثال، فرض بر این است که هیچ یک از دستگاه های IoT در چهار اتاق نمی توانند الزامات اپراتور J۱۷ را برآورده کنند. در نتیجه، J۱۷ باید روی یک دستگاه اینترنت اشیا دیگر نگاشت شود.

در این حالت سرور لبه در اتاق زیرزمین ساختمان برای اجرای این عملگر علامت گذاری شده است. از آنجایی که مفاهیم توضیح داده شده برای نگاشت بر اساس یک روش عمومی است به عنوان مثال، مدل های عمومی به کار گرفته شده اند (IoTEM، DSPM)، و علاوه بر این، مجموعه ای از الگوریتم های نگاشت قابل توسعه ارائه شده است، این رویکرد قابل انتقال به روش های دیگر است. دامنه ها نیز مانند دامنه کارخانه هوشمند. این رویکرد همچنین برای چنین سناریوهای بزرگ مقیاس می شود، یعنی الگوریتم ها در نهایت یک راه حل را بر می گردانند.

۵-۲ رویکرد دستی

در این رویکرد، یک طرح نگاشت به صورت دستی توسط تحلیلگران دامنه ایجاد می شود که مکان قرارگیری را به صراحت برای هر اپراتور درگیر در پردازش داده های برنامه های کاربردی اینترنت اشیا تعیین می کنند. به این معنی که تحلیلگران دامنه باید بدانند که آیا اشیاء IoT منابع کافی را فراهم می‌کنند یا خیر، و علاوه بر این، الزامات اپراتورهای برنامه اینترنت اشیا را برآورده می‌کنند.

این پایان نامه از استاندارد TOSCA برای ایجاد نقشه های نگاشت به صورت دستی استفاده می کند. تا اینجا نشان داده‌ایم که چگونه برنامه‌های نگاشت دستی مبتنی بر TOSCA برای سناریوهای اینترنت اشیا در حوزه خانه هوشمند می‌توانند محقق شوند، و علاوه بر این، چگونه می‌توان آنها را در مرحله بعدی با استفاده از یک موتور استقرار مبتنی بر TOSCA به کار برد. شکل زیر یک نگاشت دستی با توپولوژی TOSCA را نشان می دهد



شکل ۸

در این رویکرد اشـیاء IoT و منطق پردازش برنامه IoT برای نظارت بر سـطوح قالب در یک اتاق تک نشان داده شده اسـت. به عنوان مثال، اشیاء IoT یک IoTEM به عنوان الگوهای گره TOSCA انتزاع می شوند. طرح نگاشت مبتنی بر TOSCA برای سـناریوی موردی ارائه شـده در بخش ۵-۱-۳ ایجاد شده است که هدف آن نظارت بر سطوح قالب در ساختمان های هوشمند است.

همانطور که در شـکل ۵,۲ در پایین نشـان داده شـده اسـت. علاوه بر این، عملگرهای پردازش یک DSPM به عنوان الگوهای گره نیز انتزاع شـده و در شـکل ۵,۲ در بالا نشـان داده شده است. توپولوژی TOSCA مدل شده به دو پشته تقسیم می شود: پشته تولید کننده داده و پشته مصرف کننده داده. تبادل داده بین تولید کننده و مصرف کننده داده از طریق الگوی انتشار-اشتراک مبتنی بر موضوع تحقق می یابد.

در این الگوی ارتباطی، یک تولیدکننده داده پیام‌هایی را برای یک موضـوع میزبانی شـده در یک کارگزار پیام منتشر می‌کند، که پیام‌های منتشر شده را به مشترکین مربوطه هدایت می‌کند به عبارت دیگر مصرف کنندگان داده.

به طور معمول، یک پشته تولید کننده داده شامل اجزای زیر است که به عنوان الگوهای گره مدل شده اند:

۱. یک یا چند دستگاه فیزیکی اینترنت اشیا، به عنوان مثال. Raspberry Pis، ساعت‌های هوشمند یا تلفن‌های هوشمند، که می‌توانند با حسگرهای مختلفی که داده‌ها را تولید می‌کنند، تعبیه یا متصل شوند،
۲. سیستم عامل دستگاه، به عنوان مثال، یک سیستم عامل میزبانی شده بر روی دستگاه IoT،
۳. یک اپراتور که متصل می‌شود. حسگرها، قادر به استخراج داده‌های خود و تحویل آنها، و
۴. موضوعی که داده‌ها برای آن منتشر شده و توسط پشته مصرف‌کننده داده قابل دسترسی است.

برای دسـتگاه هایی که سیستم عامل ارائه نمی دهند، مانند ساعت هوشمند، یک الگوی گره برای سیستم عامل مورد نیاز نیسـت. در مقابل، یک الگوی گره زمان اجرا از راه دور (به عنوان مثال، یک سـرور لبه) که به عنوان یک دروازه عمل می کند باید ارائه شـود، که امکان قرار دادن اپراتور روی آن را فراهم می کند که به چنین دسـتگاه هایی متصل می شـود و داده های حسـگرهای تعبیه شده مربوطه را استخراج می کند. مثال شرح داده شده در زیر عمدتاً بر روی دستگاه‌های IoT متمرکز است که یک سیستم عامل را ارائه می‌دهند بطور مثال : Raspberry Pis.

پشـته تولید کننده داده در شـکل ۸ شـامل یک دسـتگاه اینترنت اشـیا اسـت که دارای سـنسورهای دما و رطوبت و زیرسـاختی اسـت که دسـترسـی به این مقادیر حسـگر را فراهم می کند. به عنوان مثال، مقادیر دما و رطوبت توسط حسـگرهای متصل به Raspberry Pi اندازه‌گیری می‌شوند، که دو عملگر استخراج را برای اندازه‌گیری این مقادیر و ارسال آن‌ها به یک کارگزار پیام مبتنی بر موضوع اجرا می‌کند.

در این حالت، مقادیر دما و رطوبت با ارسـال هر دو مقدار به یک موضوع در کارگزار پیام، به یکدیگر متصل می شوند. با اسـتفاده از مفهوم انواع گره TOSCA، یک نوع گره Raspberry Pi و خواص آن مانند حسگرهای متصل یا تعبیه شده، مدل‌سازی می‌شوند. علاوه بر این، الگوی گره سیستم عامل دستگاه IoT نوع سیستم عامل دستگاه اینترنت اشیا مورد اسـتفاده را مشخص می کند، در این مورد، سیستم عامل Raspbian Jessie. برای مدل سازی مناسب چنین سیستم عاملی، باید اطلاعاتی در مورد نوع آن (مثلاً مبتنی بر یونیکس)، نحوه دسترسی به آن (مثلاً استفاده از اتصالات SSH) و اعتبار دسـترسی آن، که می تواند از طریق احراز هویت کاربر یا بر اساس آن باشد، ارائه شود. یک کلید SSH بر اسـاس این سـیسـتم عامل، اسـکریپت های اپراتور را می توان به طور خودکار برای دسـترسی به رابط های سخت

افزاری حسگرها و استخراج داده های آنها مستقر کرد. الگوی گره استخراج دما و الگوی گره استخراج رطوبت به عنوان رابط بین دستگاه فیزیکی اینترنت اشیاء عمل می کنند. به عبارت دیگر Raspberry Pi، و بخشی که داده ها را در اختیار مصرف کنندگان قرار می دهد. علاوه بر این، قالب گره Join Topic باید مدل شود. این موضوع بر روی یک جزء نرم افزار کارگزار پیام میزبانی می شود که توسط یک الگوی گره کارگزار پیام مربوطه در توپولوژی TOSCA نشان داده شده است. این واسطه پیام می تواند، برای مثال، Mosquitto Broker که یک میان افزار پیام رسانی مبتنی بر پروتکل MQTT است.

در نهایت، این الگوهای گره استخراج شده توسط یک الگوی اتصال به رابطه به الگوی گره Join Topic متصل میشوند. پشته مصرف کننده داده در شکل ۸ یک دستگاه اینترنت اشیا و زیرساختی برای مصرف، پردازش و نظارت بر مقادیر حسگر از پشته تولید کننده داده را در بر می گیرد. این پشته یک اپراتور محاسبه و سرویس ارائه می دهد که بر روی دستگاه IoT اجرا می شود. این اپراتور برای دریافت مقادیر حسگر و همچنین برای محاسبه سطح قالب فعلی در اتاق بر اساس مقادیر سنسور دما و رطوبت، به واسطه پیام پشته تولید کننده داده مشترک می شود.

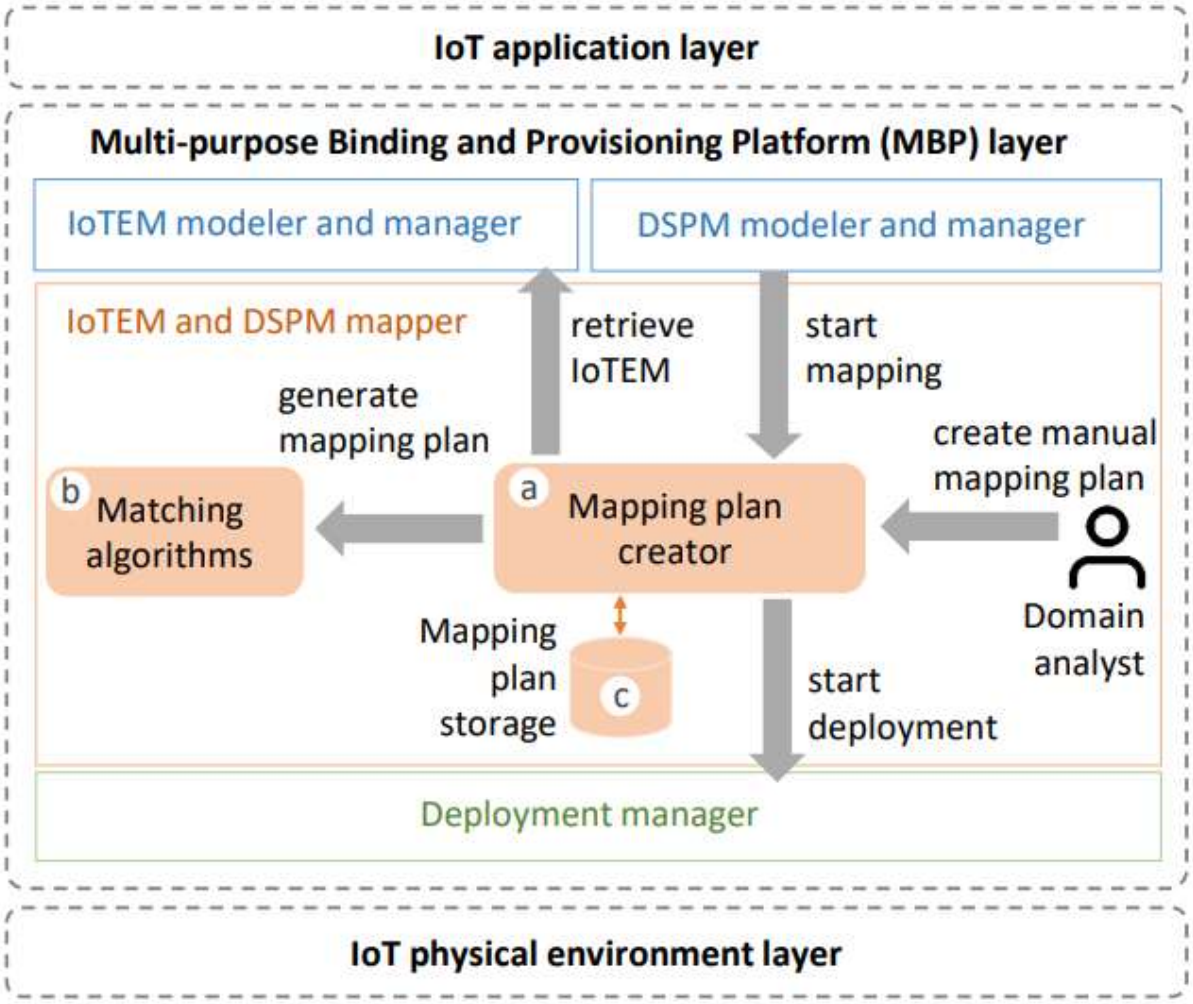
مصرف کنندگان داده به روشی مشابه با تولیدکنندگان داده کار می کنند به این معنا که بر اساس الگوی انتشار-اشتراک ایجاد می کنند. در نتیجه، برای مدل‌سازی آنها با استفاده از TOSCA، می‌توان از انواع گره‌های معرفی‌شده از قبل استفاده مجدد کرد، یعنی انواع Raspberry Pi، RaspbianJessie و Node Operator.

همانطور که در شکل نشان داده شده است، سه الگوی گره لازم است: دستگاه اینترنت اشیا، سیستم عامل دستگاه، و اپراتور محاسبه و سرویس دهی، که داده های دریافتی را از طریق الگوی Join Topic node مصرف و پردازش میکند. شبیه به الگوهای گره استخراج، الگوی گره Calculation and Serving نیز توسط یک الگوی اتصال به رابطه به قالب گره Join Topic متصل می شود. جریان داده های سناریو به طور صریح در توپولوژی TOSCA مدل سازی نشده است، بلکه به طور ضمنی در مصنوعات نرم افزاری اپراتورها پیاده سازی شده است. به عنوان مثال، جریان داده از سنسورهای دما و رطوبت به عنوان منابع داده سرچشمه می گیرد، از موضوع میزبانی شده در کارگزار پیام عبور می کند و به اپراتور محاسبه و سرویس میزبانی شده در Raspberry Pi می رسد که نشان دهنده یک سینک داده است.

به طور خلاصه، برای ایجاد یک طرح نگاشت در قالب یک مدل توپولوژی TOSCA برای چنین سناریویی، اجزای ذکر شده در بالا باید مدل‌سازی شده و مطابق شرح داده شوند. با این حال، رویکرد نگاشت دستی فقط برای موارد استفاده با مقدار کم اشیا یا اپراتورهای اینترنت اشیا توصیه می شود. یعنی ایجاد دستی یک نقشه نقشه پیچیدگی این کار را به میزان قابل توجهی افزایش می دهد و در سناریوهای بزرگتر مانند کارخانه های هوشمند یا شهرهای هوشمند مستعد خطا می شود.

۳-۵ جزئیات معماری و پیاده سازی IoTEM و DSPM mapper

در معماری کلی (به بخش ۳-۳ رجوع کنید)، بخش IoTEM و DSPM mapper ابزاری را برای پشتیبانی از تصمیم در مورد مکان اجرای اپراتورها بر اساس نیازهایشان فراهم می‌کند. در این مؤلفه، که در شکل ۹ نشان داده شده است، تحلیلگران دامنه این امکان را دارند که نگاشت اپراتورها و اشیاء اینترنت اشیا را برای DSPM و IoTEM فعلی با استفاده از دو رویکرد دستی و خودکار انجام دهند:



شکل ۹

در رویکرد خودکـار، خالق طرح تطبیقی بخش a الگوریتم‌های تطبیق b را شـــروع می‌کند که یک طرح تطبیقی را برمی‌گرداند. در رویکرد دســتی، تحلیل‌گران دامنه می‌توانند خودشــان از طریق ایجادکننده نقشــه تطبیقی طرح‌های تطبیقی ایجاد کنند.

در این مورد، یک مدل‌ســـاز برای مدل‌های توپولوژی TOSCA ، مانند Winery می‌تواند برای ایجاد و دخیره یک نقشــه تطبیقی مبتنی بر TOSCA استفاده شود. هنگامی که یک طرح تطبیقی ایجاد شد، استقرار طرح های تطبیقی مبتنی بر TOSCA را می توان با اســتفاده از یک موتور اســتقرار مبتنی بر TOSCA ، مانند OpenTOSCA ، با فعال کردن بخش Deployment Manager شـروع کرد. برای رویکرد نگاشت خودکار، خالق طرح تطبیقی و الگوریتم های تطبیق (نوع حریصـانه و عقبگرد) در سمت سرور MBP در جاوا توسط Schneider بعنوان پایان نامه کارشناسی انجام شده.

برای ذخیره سـازی نقشــه ، از پایگاه داده MongoDB اسـتفاده می شـود. بخش ۸-۱ معماری یکپارچه سازی تمام اجزای معماری را ارائه می دهد و نشان می دهد که چگونه آنها با هم تطبیق می یابند.

۵-۴ کارهای مرتبط

Rizou و همکـاران یـک الگوریتم ابتکـاری را ارائـه می دهـد کـه به توزیع اپراتورهای جریان داده اجازه می دهد تا کمترین تـأخیر ممکن را برای پردازش داده هـا فعال کند. این الگوریتم بهترین توزیع ممکن را جســـتجو می کند تا تاخیر شبکه را به حداقل برساند.

برخلاف این پایان نامه، الگوریتم مسـتقیماً بر روی اشیاء IoT موجود، نه به صورت مرکزی، اجرا می شود. با این حال، رویکرد Rizou و همکاران. مستلزم آن است که اشیاء IoT در شبکه شناخته شده، متصل و قادر به برقراری ارتباط با یکدیگر باشند، به عنوان مثال، اطلاعات مربوط به تأخیر را به اشتراک بگذارند.

رویکرد در این پایان نامه بیشـتر به دنبال یک جفت آزاد بین اشـیاء IoT اسـت. در بهترین حالت، اشیاء اینترنت اشیا نباید یکدیگر را بشـــناســـند و در نتیجه نیازی به برقراری ارتباط مســـتقیم ندارند. علاوه بر این، رویکرد Rizou و همکاران. تنها تأخیر را به عنوان یک الزام برای پردازش داده ها در نظر می گیرد، بنابراین، سایر الزامات، همانطور که در این پایان نامه توضیح داده شده است، پشتیبانی نمی شوند.

Cipriani و همکاران رویکردی را برای قرار دادن اپراتور پرس‌وجوهای پردازش جریان ارائه می‌کند، که در آن اهداف خـاص (یعنی الزامات) برنامه‌های مبتنی بر جریان برای تحقق تصـــمیم‌گیری‌های لازم برای قرار دادن در نظر گرفته می‌شـــوند. این رویکرد، قرار دادن گراف‌های پرس و جو برای جریان‌های داده (M-TOP) توســـط اپراتور چند هدفه نامیده می‌شـود. مشـابه این پایان نامه، M-TOP حاشـیه نویسـی الزامات را بر روی اپراتورهای پردازش امکان پذیر میکند. این الزامات باید توسط گره‌های محاسباتی موجود (به عنوان مثال، اشیاء اینترنت اشیا) برآورده شوند تا به یک مکـان مناســـب اپراتور دســـت یابند. با این حال، بر خلاف رویکرد این پایان نامه، M-TOP قرار دادن اپراتور پس از استقرار را در نظر نمی گیرد، یعنی نظارت بر پردازش جریان مستقر در دسترس نیست.

علـاوه بر این، رویکرد در این پـایـان نامه الزامات مبتنی بر برنامه و مبتنی بر کاربر را نیز در نظر می گیرد که با آینده حوزه اینترنت اشـیا پدیدار شـد. بنابراین، انواع جدیدی از اشیاء سخت افزاری اینترنت اشیا که واجد شرایط گره های محاسباتی هستند نیز در نظر گرفته می شوند.

Bumgardner و همکاران یک زبان مدل‌سازی مبتنی بر نمودار به نام Cresco Application Model (CAM) را ارائه می‌کند تا منطق برنامه‌های کاربردی مبتنی بر جریان توزیع شـده را توصـیف کند. مدل های CAM مشـابه مدل های DSPM ارائه شـــده در این پایان نامه هســـتند. علاوه بر این، Bumgardner و همکاران. از طریق یک الگوریتم حریصـانه، توزیع بهینه اپراتورها را بر روی اشیاء اینترنت اشیا جستجو کنید. الگوریتم های ارائه شده در این پایان نامه، علاوه بر این، توزیع را بر اساس فاصله شبکه به منظور فعال کردن مسیرهای ارتباطی کوتاه بهینه می کند.

فصل ششم

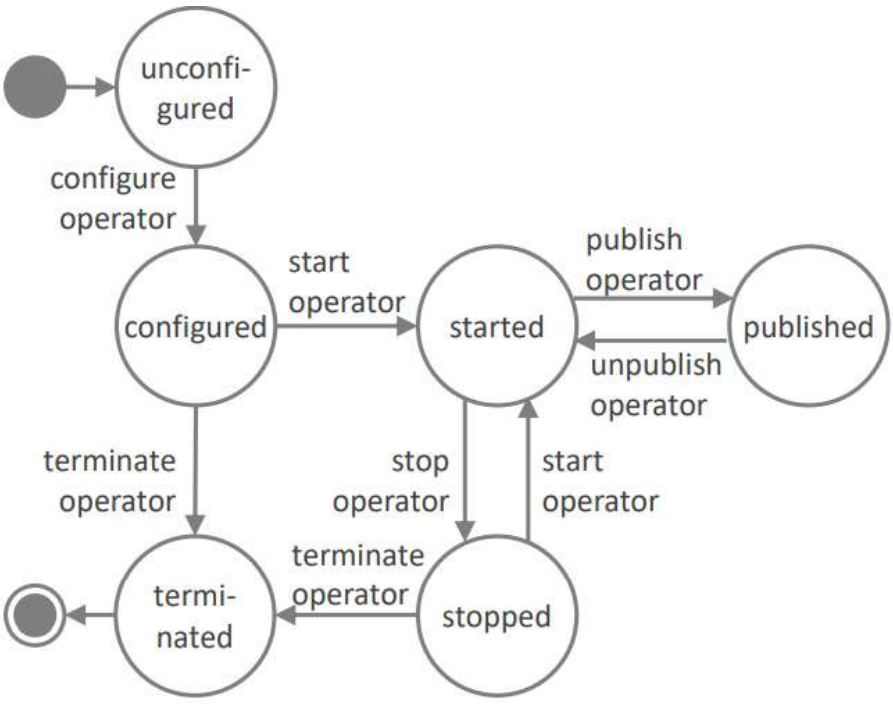
استقرار اپراتورها در محیط های IoT

۶-۱ رویکرد خودکار

این رویکرد شـامل نمایش اپراتور است که اپراتورهای یک DSPM را به اشیاء IoT مدل اختصاص می دهد. بر اساس نمایش اپراتور، امکان تکرار می باشـد و هر یک از آنها به طور خودکار با اسـتفاده از موتور استقرار مبتنی بر TOSCA بنام OpenTOSCA مسـتقر می شـوند. هر اپراتور می تواند به چندین حالت برسـد که نشـان دهنده وضـعیت فعلی استقرار یک اپراتور است. این حالات در ادامه توضیح داده شده است.

۶-۱-۱ حالت های استقرار یک اپراتور

این پایان نامه شـش حالت اسـتقرار یک اپراتور را تعریف می کند: (i) پیکربندی نشـده، (ii) پیکربندی، (iii) شـروع شده، (iv) منتشر شده، (v) متوقف شده، (vi) پایان یافته. این حالت های عملگر در شکل ۱۰ نشان داده شده است.



شکل ۱۰

وضـعیت اولیه یک اپراتور (unconfigured) پیکربندی نشـده اسـت. پس از شـروع استقرار، اولین کار پیکربندی شی IoT اختصـاص داده شـده برای اجرای اپراتور اسـت. برای این کار، اپراتور در قالب نرم افزار طراحی شده به شی IoT اختصـاص داده شده کپی می شود. علاوه بر این، وابستگی‌های نرم‌افزاری مورد نیاز (به عنوان مثال، کلاینت MQTT، مفسر پایتون) برای اپراتور بر روی شی IoT اختصاص‌یافته نصب و پیکربندی می‌شوند.

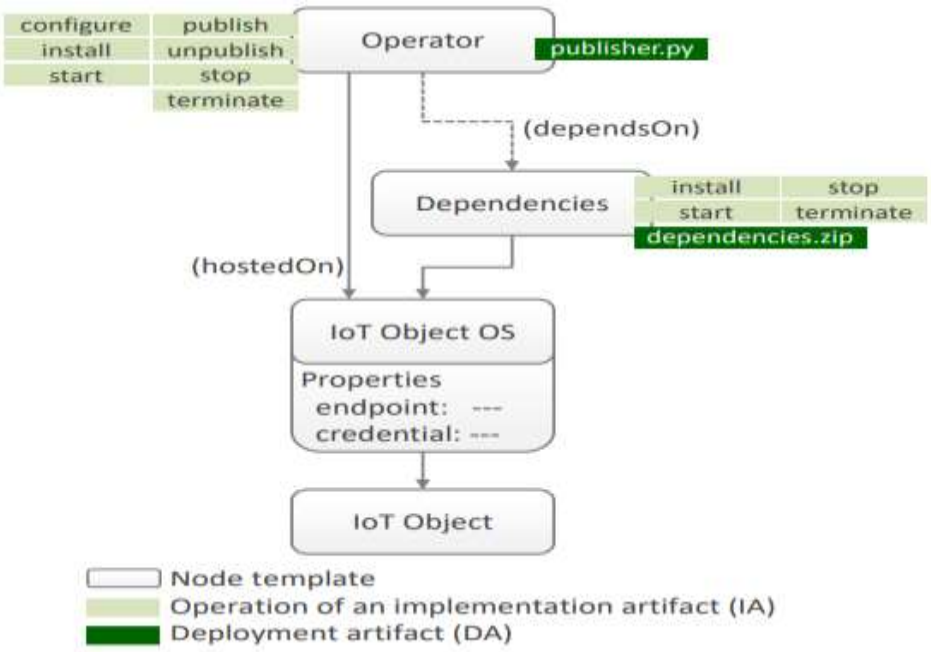
پس از پیکربندی، اپراتور به حالت پیکربندی شده (configured) منتقل می شود. یک اپراتور پیکربندی شده میتواند راه اندازی شـود، که هدف اصـلی اسـتقرار خودکار اسـت. هنگامی که همه اپراتورها در نقشـه برداری اپراتور به حالت شـروع رسـیدند، اسـتقرار خودکار به عنوان پایان یافته در نظر گرفته می شـود و در نتیجه اپراتورهای برنامه IoT راه اندازی و اجرا می شـوند. علاوه بر این، یک اپراتور پیکربندی شـده را می توان مسـتقیماً به عنوان مکانیزم بازگشـتی خاتمه داد در صورتی که اپراتور نمی تواند راه اندازی شود.

هنگامی که یک اپراتور راه اندازی می شـود، خروجی آن مانند داده های حسـگر استخراج شده، می تواند برای برنامه های خارجی در دسـترس قرار گیرد. در این کار تحلیلگران تصـمیم می گیرند آیا و کدام اپراتورها باید منتشـر شوند. برای این، توضیحاتی در مورد نحوه دسترسی و تجزیه خروجی عملگرها به طور خودکار ایجاد می شود.

چنین توصیفاتی و مفاهیم مربوط به مدیریت آنها به تفصیل در بخش ۶-۳ توضیح داده شده است. در نهایت، هنگامی که پردازش جریان داده کنار گذاشـته می شـود، هر یک از اپراتورهای قبلا مسـتقر شـده باید توسط تحلیلگران دامنه کنار گذاشـته شـوند. برای این کار، هر اپراتور شروع شده متوقف می شود و متعاقباً خاتمه می یابد. اگر اپراتور منتشر شده باشد، منتشر نشده، متوقف و خاتمه می شود. با پایان دادن به یک اپراتور، تمام نرم افزارهای طراحی شده که در شی IoT کپی شده اند حذف می شوند و وابستگی ها حذف می شوند.

۶-۱-۲ استقرار اپراتور مبتنی بر TOSCA

شکل ۱۱ یک مدل توپولوژی TOSCA را نشان می دهد



شکل ۱۱

این مدل در این پایان نامه به عنوان یک الگوی پایه برای استقرار هر اپراتور در یک نقشه استفاده می شود. این توپولوژی هنوز شامل تمام اطلاعات لازم برای استقرار نیست و باید با اطلاعات خاص هر اپراتور در نقشه نقشه در طول استقرار گسترش یابد.

الگوی گره Operator یک رابط چرخه حیات را تعریف می کند که شامل عملیات مربوط به انتقال حالت عملگر تعریف شده در این پایان نامه است. این عملیات عبارتند از: پیکربندی، نصب، شروع، انتشار، لغو انتشار، توقف و خاتمه. علاوه بر این، Dependencies قالب گره نیز یک رابط چرخه حیات را برای وابستگی های اپراتور مدل شده تعریف می کند.

برای استقرار عملگرها با زمان اجرا OpenTOSCA، عملگرها در نقشه تکرار می شوند و این توپولوژی با جزئیات خاص هر اپراتور کپی و گسترش می یابد. نرم افزارهای طراحی شده و وابستگی های یک اپراتور به صورت برنامه نویسی به توپولوژی به عنوان ابزار پیاده سازی و آرتیفکت های استقرار الگوهای گره Operator و Dependency اضافه می شوند. علاوه بر این، ویژگی های خاص شی IoT نقشه برداری شده نیز به توپولوژی اضافه می شود. در نهایت، توپولوژی های توسعه یافته را می توان به صورت متوالی یا موازی توسط چندین نمونه موتور OpenTOSCA مستقر کرد.

۶-۲ رویکرد استقرار نیمه اتوماتیک

همانطور که قبلا بیان شد OpenTOSCA را می توان برای استقرار اپراتورها در محیط های IoT به روشی کاملاً خودکار بر اساس توپولوژی ها استفاده کرد. این رویکردها فرض می‌کنند که اشیاء IoT در محیط آن از قبل برای استقرار اپراتورها پیکربندی شده‌اند. با این حال، معمولاً برای استقرار اپراتورهای برنامه های اینترنت اشیا، به عنوان مثال، وصل کردن حسگرها، اقدامات دستی، به نام وظایف انسانی، مورد نیاز است.

در این پایان نامه، مفهوم وظایف انسانی بر اساس مشخصات OASIS WS-HumanTask است. وظایف انسانی فعالیت هایی هستند که باید توسط افراد انجام شوند و بنابراین نمی توانند به طور خودکار انجام شوند. نمونه هایی از وظایف انسانی شامل تأیید فرآیندهای خاص، به عنوان مثال، برای خرید سخت افزار گران قیمت یا اعطای وام است. نمونه‌های بیشتر در زمینه محیط‌های اینترنت اشیا، وصل کردن حسگرها به دستگاه‌ها یا میکروکنترلرهای چشمک زن است. در نتیجه، وظیفه انسانی درخواستی از شخص برای انجام یک فعالیت خاص است.

برای تطبیق بیشتر برای چالش های ارائه شده در محیط های IoT، این پایان نامه یک رویکرد استقرار نیمه خودکار را ارائه می دهد که در آن وظایف انسانی نیز برای استقرار در نظر گرفته می شود. برای این کار، این پایان نامه کارشناسان حوزه را قادر می سازد تا وظایف انسانی را تعریف کرده و قبل از شروع استقرار آنها را به نقشه های نقشه اضافه کنند. علاوه بر این، با پشتیبانی از وظایف انسانی، قابلیت استفاده مجدد از طرح های نقشه برداری را برای محیط های مختلف اینترنت اشیا مشابه، به عنوان مثال، ساختمان های هوشمند متفاوتی که از اشیاء اینترنت اشیا یکسان تشکیل شده اند، افزایش می دهد.

در این حالت، مقادیر خاصی از اشیاء اینترنت اشیا، مانند آدرس‌های IP، نیازی به مدل‌سازی در IoTEM ندارند و می‌توانند توسط یک وظیفه انسانی در طول استقرار پر شوند. هنگامی که استقرار یک DSPM راه اندازی می شود، طرح نقشه برداری به عنوان ورودی برای استقرار دریافت می شود. این طرح نقشه برداری برای تعاریف وظایف انسانی و الگوهای ضمنی که دلالت بر نیاز به وظایف انسانی دارد جستجو می شود.

نمونه ای از چنین الگوی یک مقدار ویژگی خالی در توصیف یک شی اینترنت اشیا است. در این مورد، یک تعریف وظیفه انسانی به طور خودکار ایجاد می شود تا مقادیر خالی ویژگی را پر کند. در صورتی که نمایش کامل در نظر گرفته شود بعبارت دیگر طرح نقشه برداری شامل تعاریف یا الگوهای وظایف انسانی نیست،

استقرار همانطور که در بخش ۶-۱ توضیح داده شده انجام می شود. در نهایت، هنگامی که جستجو انجام شد، لیستی از تعاریف وظایف انسانی به یک جزء میان افزار به نام Human task manager ارسال می شود. این بخش به کاربران (به عنوان مثال، کارشناسان حوزه) در مورد وظایف انسانی که باید انجام شود اطلاع می دهد و همچنین موتورهای استقرار را در مورد وضعیت وظایف انسانی مطلع می کند.

۶-۳ زبان توضیح برای اینترنت اشیاء (Topic Description Language)

اینگونه تعریف می شود که ابزار ساده ای برای توصیف و یافتن موضوعات برای حسگرها و محرک های عمومی فراهم می کند. نماد TDLIoT (The Topic Description Language for the Internet) از بررسی متون و تجربیات گسترده در پلتفرم‌ها، پروتکل‌ها و برنامه‌های مختلف اینترنت اشیا مشتق شده است. رویکرد TDLIoT در این پایان نامه برای ایجاد توضیحاتی استفاده می شود که هدف آن سهولت دسترسی به داده های خروجی اپراتورهای منتشر شده (به عنوان مثال، داده های حسگر استخراج شده) توسط کاربردهای بیشتر IoT است.

یک مثال برای چنین برنامه IoT، تشخیص مستقیم فضاهای پارک خالی در یک شهر هوشمند است، که در آن رانندگان در نزدیکی با تلفن های هوشمند خود در مورد مکان های پارک خالی مطلع می شوند. حسگرهای فضای پارکینگ می توانند در دسترس بودن فضاهای پارک را تشخیص دهند و هنگامی که فضاهای پارک در دسترس هستند، رانندگان می توانند از طریق برنامه های تلفن هوشمند خود، محرک ها را فعال کنند.

با این حال، برای توسعه چنین برنامه‌ای IoT، توسعه‌دهندگان باید بدانند : (i) کدام حسگرها و محرک‌ها در زمینه برنامه وجود دارند، (ii) چه حسگرهای داده ارائه می‌دهند و در چه قالبی، (iii) کدام اقدامات را می توان برای محرک ها راه اندازی کرد، و (iv) چگونه می توان به حسگرها و محرک ها برای استفاده دسترسی داشت.

در اینترنت اشیا، دسترسی به حسگرها و محرک‌ها معمولاً از طریق مدل ارتباطی انتشار-اشتراک محقق می‌شود. این موضوعات برای ارائه مقادیر حسگر یا امکان دسترسی به محرک ها استفاده می شود. در محدوده این پایان نامه، موضوع (Topic)، بخشی است که امکان دریافت و ارسال داده ها را به صورت یکسان فراهم می کند. موضوعات را می توان از طریق مدل های ارتباطی مختلف، مانند انتشار-اشتراک یا درخواست-پاسخ، با استفاده از پروتکل های مختلف محقق کرد.

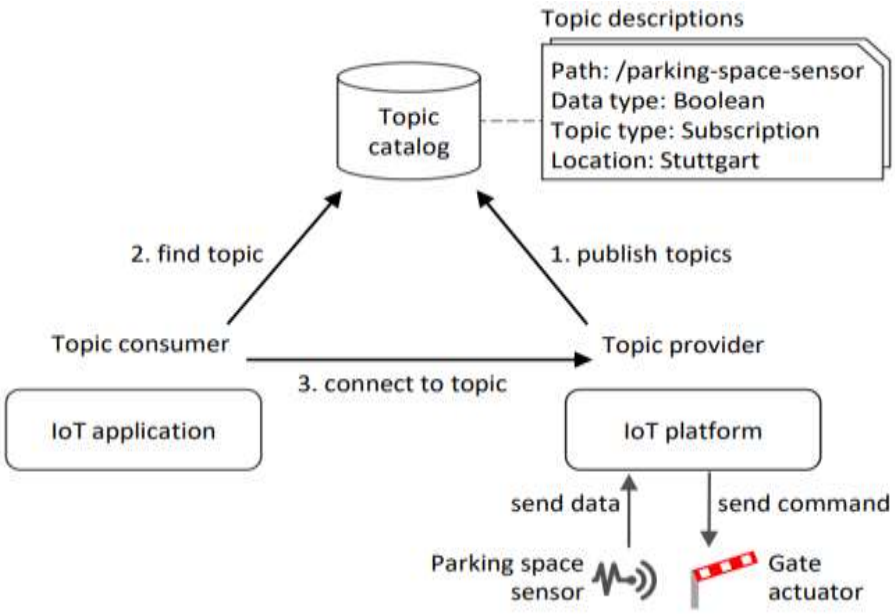
در مدل انتشار-اشتراک مبتنی بر موضوع محقق می شود، مشترکین در موضوعات مرتبط با زمینه خود ثبت نام می کنند تا زمانی که ناشران پیام هایی به این موضوعات ارسال می کنند به طور ناهمزمان مطلع شوند.

در مدل درخواست-پاسخ، برنامه‌ها به جای اطلاع‌رسانی، باید داده‌ها را درخواست کنند، به عنوان مثال، از طریق درخواست‌های HTTP. مدیریت اشتراک‌ها و تحویل پیام‌ها معمولاً توسط پلتفرم‌های IoT، مانند FIWARE - Mosquitto - OpenMTC یا RMP انجام می‌شود. به عنوان مثال، با توجه به اینکه مقادیر حسگر از طریق موضوعات میزبانی شده در چنین پلتفرم‌های اینترنت اشیا ارائه می‌شوند، با استفاده از مدل انتشار-اشتراک، یک برنامه پارکینگ شهر هوشمند مشترک موضوعات مربوط به همه حسگرهایی است که مکان‌های پارک را نظارت می‌کنند و زمانی که یک نقطه پارکینگ به‌عنوان برنامه خالی شناسایی شد، مطلع می‌شود.

توسعه دهندگان به دانستن همه چیز در مورد موضوعاتی که می توانند برای ساخت چنین برنامه های IoT استفاده کنند، وابسته هستند. این اطلاعات، که برای مثال شامل ساختار داده یا نحوه دسترسی به آن می‌شود، معمولاً تنها در صورتی شناخته می‌شود که توسعه‌دهندگان برنامه مالک اشیاء IoT درگیر باشند. سایر موضوعات موجود، که میتوانند اطلاعات بیشتری در مورد محیط ارائه دهند، اغلب در نظر گرفته نمی شوند، اما می توانند منجر به بهبود قابل توجه برنامه شوند، مانند از طریق پوشش بالاتر توسط سنسورهای اضافی.

به عنوان مثال، داده‌های اینترنت اشیا در دسترس عموم در پلتفرم‌هایی مانند dweet.io وجود دارد، اما جزئیاتی در مورد محتوای داده‌ها، به عنوان مثال، ساختار داده و نحوه تفسیر آنها ارائه نشده است. بنابراین، رویکرد TDLIoT ابزار ساده ای برای توصیف و یافتن موضوعات اپراتورهای منتشر شده، و علاوه بر این، منابع داده و سینک های انتزاعی از حسگرها و محرک ها فراهم می کند.

رویکرد TDLIoT : (i) یک توصیف جامع از موضوعات، (ii) یک کاتالوگ موضوع برای مرور توضیحات موضوع، و (iii) یک راه موثر برای یافتن موضوعات مناسب که دسترسی به حسگرها و محرک‌ها را به عنوان منابع داده و مخزن‌ها انتزاع می‌کند، ارائه می‌کند. به این ترتیب، توسعه اپلیکیشن IoT می تواند از طریق انتزاع از پلتفرم های خاص اینترنت اشیا آسان شود. موضوعات موجود در کاتالوگ را می توان به عنوان مثال با یک مکان خاص یا نوع حسگر جستجو کرد. رویکرد TDLIoT، که در شکل ۱۲ نشان داده شده است،



شکل ۱۲

از سه نقش اصلی تشکیل شده است: ارائه دهنده موضوع، مصرف کننده موضوع، و فهرست موضوع. ارائه دهنده موضوع، توضیحات موضوع را بر اساس TDLIoT ایجاد می کند و آنها را در کاتالوگ موضوع منتشر می کند. مصرف کننده موضوع توضیحات موضوع جالب را در کاتالوگ موضوع جستجو می کند و مستقیماً به ارائه دهندگان موضوع متصل می شود تا داده ها را منتشر کند یا داده های منتشر شده در موضوعات مشترک را دریافت کند. در محدوده این پایان نامه، یک کاتالوگ موضوعی TDLIoT ارائه شده است که در آن توضیحات مربوط به اپراتورها ذخیره میشود. بنابراین، این کاتالوگ می تواند توسط توسعه دهندگان خارجی به منظور ایجاد برنامه های کاربردی اینترنت اشیا بیشتر که به داده های قابل دسترس عموم نیاز دارند، به عنوان مثال، در حوزه شهر هوشمند جستجو شود.
یک TDLIoT باید حداقل دارای ویژگی‌های زیر باشد، با این حال، در صورت لزوم می‌توان آن را با ویژگی‌های بیشتر برای توصیف یک موضوع با جزئیات بیشتر گسترش داد:

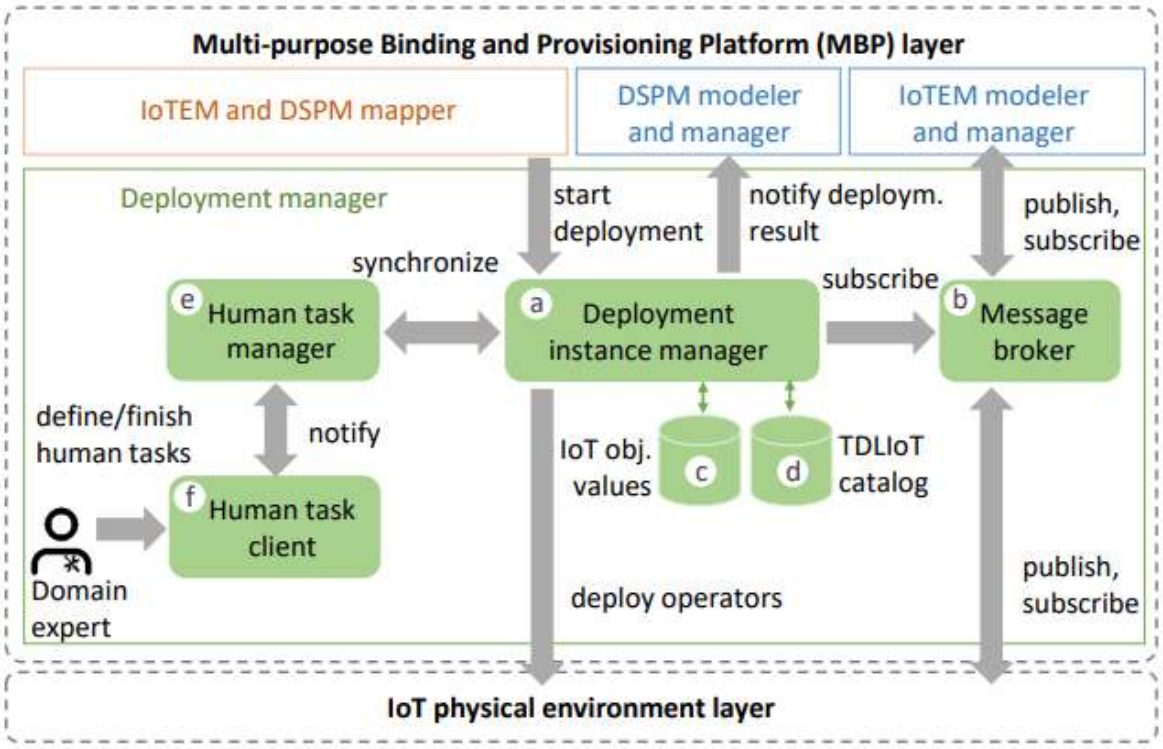
- نوع داده: نوع مقادیر ارائه‌شده توسط موضوع، به عنوان مثال، Boolean .
- نوع سخت افزار (اختیاری): نوع سخت افزاری که با موضوع نشان داده می شود، به عنوان مثال، یک سنسور
- مکان: محل حسگر یا محرک. این شامل نوع مکان، به عنوان مثال، GPS یا نام شهر خاص، و همچنین مقدار مکان، به عنوان مثال، مختصات GPS خاص است.
- قالب پیام: قالب پیام ارائه شده توسط موضوع، به عنوان مثال، JSON، YAML، یا XML.
- ساختار پیام: ساختار پیام که به عنوان فرامدل برای درک محتوای آن تعریف شده است. این شامل نوع متامدل است، به عنوان مثال، طرحواره JSON یا طرحواره XML و متامدل خاص.
- نقطه پایانی پلتفرم: نقطه پایانی پلتفرم اینترنت اشیا که میزبان موضوع است، به عنوان مثال، نقطه پایانی یک واسطه پیام در حال اجرا بر روی یک سرور.
- مالک: نام ارائه دهنده موضوع.
- مسیر: مسیر موضوع.
- پروتکل: پروتکل ارتباطی مورد استفاده، به عنوان مثال، MQTT یا HTTP.
- نوع موضوع: نوع موضوع، یعنی اشتراک یا فرمان.
- واحد (اختیاری): واحد داده ارائه شده از طریق موضوع، به عنوان مثال، سانتیگراد اگر موضوع مقادیر دما باشد

این موجودیت ها می توانند در تعداد دلخواه توصیفات رخ دهند. در مواردی که موضوعات داده‌های جمع‌آوری شده را ارائه می‌دهند، به عنوان مثال، از حسگرهای مختلف.

۴-۶ جزئیات معماری و پیاده سازی مدیر استقرار

در معماری کلی بخش ۳-۳، بخشDeployment manager ابزاری را برای استقرار اپراتورها بر روی اشیاء IoT فراهم می کند. این جزء در شکل ۱۳ نشان داده شده است. استقرار اپراتورهای یک DSPM را می توان از طریق Deployment instance manager بخش a شروع کرد، که به عنوان ورودی طرح نگاشت ایجاد شده از طریق تولید طرح نقشه برداری (ر.ک. بخش ۵-۱) یا ایجاد دستی (ر.ک. بخش ۵-۲) را دریافت می کند. علاوه بر این، پس از تکمیل استقرار DSPM ها، نمونه های در حال اجرا DSPM ها (dDSPM) توسط مدیر نمونه استقرار ایجاد و

مدیریت می شـوند. این نمونه‌های در حال اجرا به مؤلفه مدل‌سـاز و مدیر DSPM (به بخش ۴-۲-۳ مراجعه کنید)، که از موفقیت آمیز بودن اطلاعات مطلع می‌شود، ارسال می‌شوند.



شکل ۱۳

در رویکرد اسـتقرار خودکار، اپراتورها و نرم افزارهای طراحی شده مورد نیاز آنها بر روی اشیاء IoT نقشه برداری شده نصـب، پیکربندی و شـروع می شـوند. هنگامی که اجرای اپراتورها بر روی اشـیاء IoT آغاز می شـود، آنها داده ها را منتشر می کنند مانند مقادیر حسگر یا اطلاعات نظارت، به بخش Message broker در قسمت b .

بخش Deployment instance manager برای دریافت مداوم مقادیر شـی IoT و مقادیر نظارتی که در مخزن مقادیر شـی IoT بخش c ذخیره می شـوند، با Message broker مرتبط می شـود. توضـیحات نحوه دسـترسی و تجزیه خروجی اپراتورهای منتشـر شـده در کاتالوگ TDLIoT بخش d ذخیره می شـود. این کاتالوگ همچنین یک REST API برای جسـتجوی اپراتورهای در دسترس عموم، به عنوان مثال، مقادیر حسگرهای عمومی را استخراج می کند، ارائه می دهد.

در رویکرد اسـتقرار نیمه خودکار، Human task manager بخش e اقدامات دسـتی (به عنوان مثال، وظایف انسانی) را از نقشه تطبیقی استنباط می کند و برنامه کاربردی Human task client را از وجود آنها مطلع میکند. هنگامی که وظایف انسـانی توسـط ورودی متخصصان دامنه در سرویس گیرنده Human task تکمیل شد، به مدیر Deployment اطلاع داده می شود که استقرار اپراتورها را شروع یا ادامه دهد.

بخش Deployment manager تا حدی در MBP به عنوان نمونه اولیه پیاده سـازی شـده اسـت و همچنین از پیاده سازی های خارجی موجود استفاده می کند. OpenTOSCA به عنوان یک مدیر نمونه استقرار خارجی استفاده می شود، که قادر است به طور خودکار اپراتورهای نقشه برداری را بر اساس استاندارد TOSCA مستقر کند.

۶-۵ کارهای مرتبط

این بخش کارهای مرتبط با استقرار نرم افزار در محیط های IoT را توضیح می دهد. لی Li و همکاران پیشنهاد میکند که از TOSCA برای مشخص کردن اجزای اساسی برنامه های کاربردی IoT (مانند میکرو کنترلرها) و پیکربندی آنها استفاده شود تا به کارگیری آنها را در محیط های ناهمگن خودکار کند. بسط های این کار توسط Vögler و همکاران ارائه شده است.

نویسندگان فریمورک LEONORE Framework را برای استقرار و اجرای منطق برنامه های سفارشی به طور مستقیم در اتصال gateway اینترنت اشیا پیشنهاد می کنند. با این حال، برای اینکه Framework بتواند gateway موجود را بشناسد، آنها باید یک عامل تأمین محلی از پیش نصب شده داشته باشند. این عامل با ارائه شناسه منحصر به فرد و داده های مشخصات دروازه (مانند آدرس MAC، مجموعه دستورالعمل و مصرف حافظه) خود را در Framework ثبت می کند. در مقابل، رویکرد این پایان نامه نیازی به اجزای از پیش نصب شده بر روی اشیاء اینترنت اشیاء ندارد.

هور Hur و همکاران یک منطق توصیف خدمات معنایی (SSD) و یک معماری سیستم را برای استقرار خودکار دستگاه‌های IoT در میان‌افزارهای ناهمگن اینترنت اشیا، با هدف حل مشکلات قابلیت همکاری بین آنها پیشنهاد می‌کند. هدف این پایان نامه نیز مقابله با مشکلات قابلیت همکاری است، با این حال، یک رویکرد مبتنی بر استاندارد را با استفاده از TOSCA برای استقرار خودکار اپراتورهای محاسباتی در محیط های ناهمگن اینترنت اشیا پیشنهاد می کند.

هیرمر Hirmer و همکاران رویکردی را برای اتصال خودکار دستگاه‌های اینترنت اشیا با استفاده از میان‌افزاری به نام پلتفرم مدیریت منابع (RMP) معرفی می‌کند. RMP ثبت آسان دستگاه های IoT و اتصال آنها از طریق آداپتورها را امکان پذیر می کند. آداپتور قطعه‌ای از کد است که حاوی منطق خواندن مقادیر حسگر دستگاه‌های IoT، ارسال مقادیر حسگر به RMP و فراخوانی محرک‌ها است. برای اتصال، اسکریپت های آداپتور به طور خودکار بر روی دستگاه های اینترنت اشیا مستقر می شوند. توسعه RMP، که از TOSCA برای استقرار آداپتورها استفاده می کند، توسط Hirmer و همکارانش نیز توضیح داده شده است.

در مقابل این پایان نامه، هیرمر و همکاران. روی اتصال دستگاه‌های سخت‌افزاری تمرکز کنید، در حالی که رویکرد این پایان‌نامه به‌علاوه به استقرار اپراتورهای محاسباتی در کل محیط‌های IoT می‌پردازد. استقرار خودکار در محیط‌های اینترنت اشیا را می‌توان با چندین رویکرد، به عنوان مثال، با استفاده از اسکریپت‌های Shell، Chef یا Puppet تحقق بخشید. با این حال، بر خلاف این رویکردها، رویکردهای مبتنی بر استاندارد در این پایان‌نامه از TOSCA استفاده می‌کنند و یک رویکرد عمومی مبتنی بر مدل‌های توپولوژی و نماد گرافیکی مربوطه را امکان‌پذیر می‌سازند

این مدل‌های توپولوژی بسیار سازگار هستند به گونه‌ای که اجزای نرم‌افزاری منفرد یک توپولوژی را می‌توان به راحتی تعویض کرد، و علاوه بر این، گسترش داد. در روش‌های دیگر، این نیاز به تلاش زیادی برای انطباق دارد، به عنوان مثال، هنگام ویرایش اسکریپت‌های آشپز. علاوه بر این، از طریق مفاهیم گره و انواع رابطه، TOSCA سطح انتزاعی بالایی را ارائه می دهد که از قابلیت استفاده مجدد برای استقرار نرم افزار پشتیبانی می کند.

فصل هفتم

نظارت بر DSPM های مستقر شده

۷-۱ مدل سازی تشخیص اختلال

برنـامه های کاربردی متن آگاه (Context-aware) در ســـال های اخیر توجه زیادی را به خود جلب کرده اند، به ویژه در حوزه اینترنت اشـیا بخصـوص که اسـتخراج محتوای سطح بالا از داده‌های حسگر خام و سطح پایین، انطباق خودکار و تحقق محیط‌های اینترنت اشیاء خودسازمان‌یافته را ممکن می‌سازد. با این حال، چالش‌های زیادی در رابطه با کسب، مدل‌سازی و مدیریت اطلاعات زمینه وجود دارد.

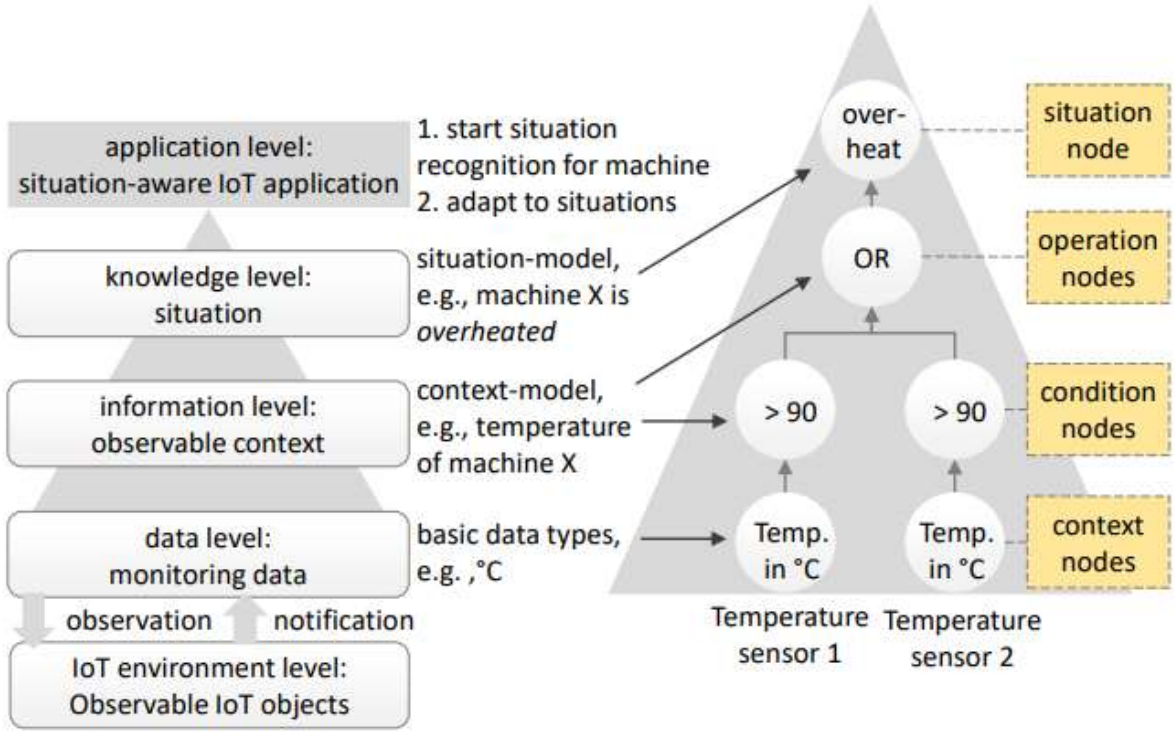
چالش دیگر این اســـت که چگونه می توان به پردازش کارآمد مقادیر زیادی از داده های حســـگر و در نتیجه کیفیت خوب دانش زمینه سـطح بالا دسـت یافت. برای این کار، تکنیک‌های تثبیت‌شده‌ای مانند پردازش رویدادهای پیچیده (CEP)، برای پردازش حجم زیادی از داده‌ها لحظه ای معرفی شده‌اند.

این پایان نامه از تعریف ارائه شـده توسـط Dey و Abowd اسـتفاده می کند و زمینه را به عنوان "هر اطلاعاتی که می تواند برای توصـیف وضعیت یک موجودیت استفاده شود، که در آن موجودیت می تواند یک شخص، مکان یا شی باشـد" توصیف می کند. در نتیجه، چنین اطلاعات زمینه ای را می توان برای استخراج اطلاعات محتوای سطح بالا که موقعیت نامیده می شود استفاده کرد.

در محدوده این پایان نامه، اختلال به عنوان یک وضـــعیت در نظر گرفته می شـــود که به عنوان یک اتفاق که ممکن اسـت نیاز به اقدامات اصـلاحی داشـته باشـد تعریف می شـود بنابراین در این پایان نامه از تشخیص اختلال با عنوان تشخیص موقعیت نیز یاد می شود.

هیرمر Hirmer و همکاران SitRS را پیشـنهاد می کند، یک سـرویس تشخیص موقعیت آماده برای ابر که تشخیص موقعیت را بر اسـاس داده های حسـگر خام امکان پذیر می کند. در رویکرد آنها، سـنسـورها به صـورت پویا محدود میشوند و داده های حسگر به روشی مبتنی بر کشش دریافت می شوند. تشخیص موقعیت در بازه های زمانی ثابت با کشـیدن داده های حسـگر و اسـتخراج موقعیت ها اجرا می شـود و بنابراین، تنها برای سناریوهای تشخیص موقعیت سـاده مناسـب اسـت. SitRS XT، یک سـرویس تشخیص موقعیت کارآمد و مقیاس پذیر می باشد و رویکرد هیرمر و همکاران را گسترش می دهد.

با فعال کردن پردازش مداوم داده‌های حسگر از طریق رویکرد مبتنی بر جریان با استفاده از فناوری‌های CEP. بدین ترتیب، مدل‌سازی و اجرای سناریوهای پیچیده‌تر تشخیص موقعیت فعال می‌شود. این پایان نامه از SitRS XT برای تشخیص اختلالات در DSPM های مستقر در محیط های IoT استفاده می کند. SitRS XT داده های نظارتی را در سطوح مختلف پردازش می کند که در شکل ۱۴ در سمت چپ نشان داده شده است.



شکل ۱۴

در سطح داده، فقط داده های نظارتی خام (به عنوان مثال، مقادیر سنسور دما) در دسترس است. این داده‌ها به سطح اطلاعات منتقل می‌شوند تا با اطلاعات مربوط به روابط داده‌های خام با چیزهای دنیای واقعی (مثلاً یک ماشین تولید) افزوده شوند و به این ترتیب به اطلاعاتی در مورد محیط اینترنت اشیا تبدیل شوند. بر اساس مشاهدات پیش زمینه، داده‌های نظارت به منظور استخراج موقعیت‌ها جمع‌آوری و تفسیر می‌شوند که منجر به دانش در مورد محیط اینترنت اشیا می‌شود. این دانش سطح بالا برای برنامه های کاربردی آگاه از موقعیت بسیار مهم است، زیرا می توان آن را در سطح بالاتری از انتزاع پردازش کرد.

در SitRS XT، موقعیت‌ها به‌عنوان الگوهای موقعیت مدل‌سازی می‌شوند، یک مدل خاص دامنه که از جزئیات پیچیده و فنی انتزاع می‌کند. این مدل شامل اشیاء IoT نظارت شده و شرایطی است که برای شناسایی یک موقعیت خاص باید مطابقت داشته باشند. الگوهای موقعیت بر اساس درختان تجمیع موقعیت (SAT) هستند که نمودارهای منسجم جهت دار هستند که توسط زوایگل و همکاران معرفی شده اند.

در SATها، حسگرها با گره‌های برگ به نام گره‌های زمینه مطابقت دارند و شاخه‌ها از پایین به بالا از طریق ترکیبی از به اصطلاح گره‌های شرطی و گره‌های عملیاتی تا رسیدن به گره ریشه (یعنی گره موقعیت) جمع می‌شوند. یک مثال ساده از یک الگوی موقعیت در شکل ۱۴ در سمت راست نشان داده شده است. شرایطی را برای تشخیص زمانی که دمای یک ماشین تولیدی از آستانه ۹۰ درجه سانتیگراد عبور می کند، تعریف می کند. گره‌های یک الگوی

موقعیت، سطوح پردازش فوق‌الذکر را منعکس می‌کنند: گره‌های زمینه نشان‌دهنده حسگرهایی هستند که یک شی IoT خاص را نظارت می‌کنند که با سطح داده مطابقت دارد. گره های زمینه به گره های شرطی متصل می شوند که روابط داده های حسگر را با اشیاء اینترنت اشیا (سطح اطلاعات) برقرار می کنند. علاوه بر این، گره های شرط، داده های نظارتی را بر اساس شرایط تعریف شده فیلتر می کنند. گره‌های شرطی را می‌توان توسط گره‌های عملیاتی با استفاده از عملیات منطقی جمع‌آوری کرد تا زمانی که به گره موقعیت رسید، که نشان‌دهنده وضعیتی است که باید شناسایی شود.

ترکیب گره‌های شرط، عملیات و موقعیت با سطح دانش مطابقت دارد، جایی که داده‌های حسگر تجمیع، تفسیر و به موقعیت‌ها مشتق می‌شوند. تعریف موقعیت ها با استفاده از الگوهای موقعیت، متخصصان حوزه را از ایجاد نمایش های پیچیده و اجرایی، مانند پرس و جوهای CEP، آزاد می کند. با این حال، چنین نمایش هایی هنوز برای استقرار در محیط های اجرا مورد نیاز است. ایجاد دستی چنین نمایش های پیچیده ای نیاز به دانش تخصصی دارد و بنابراین زمان بر و مستعد خطا است.

بنابراین، SitRS XT این پیچیدگی را با تبدیل خودکار قالب‌های موقعیت به نمایش‌های اجرایی مورد نیاز، یعنی پرس و جوهای CEP، کاهش می‌دهد. تبدیل الگوهای موقعیت به پرس و جوهای CEP، و در نتیجه، اجرای تشخیص اختلال بر اساس این نمایش های اجرایی در بخش ۷-۲ توضیح داده شده است. در رویکرد SitRS XT فوق الذکر برای تشخیص اختلال، فرض بر این است که پرس و جوهای CEP حاصل فقط در زیرساخت های فناوری اطلاعات یکپارچه اجرا می شوند. با این حال، برای پردازش کارآمدتر داده ها، رویکردهای بیشتری برای توزیع پرس و جوهای CEP در محیط IoT مورد نیاز است، به طوری که پردازش داده های توزیع شده با مسیرهای ارتباطی کوتاه و کاهش ترافیک شبکه فعال می شود.

بنابراین، ما رویکردی را برای ارسال پرس و جو CEP به محیط های IoT معرفی کردیم، به طوری که از اجرای تمام پرس و جوهای مورد نیاز CEP فقط در زیرساخت های فناوری اطلاعات یکپارچه می توان اجتناب کرد. در ادامه، یک سناریوی موردی بر اساس کار هوس و همکاران. ارائه شده است، که در آن می توان چندین پرس و جو CEP را به مکان های مختلف اجرا ارسال کرد (به عنوان مثال، مستقر کرد).

هدف از این سناریوی موردی، نظارت بر یک مرحله تولید در کف کارگاه یک شرکت تولیدی است تا در اسرع وقت اختلالات در مرحله تولید شناسایی شود. در این مرحله تولید، یک کارگر کف مغازه قطعه فلزی را وارد دستگاه می‌کند که آن را به شکل مورد نیاز برش می‌دهد. در این فرآیند، دو مشکل ممکن است رخ دهد: (۱) ابزار برش فلز با گذشت زمان پوسیده می شود، یا (ب) قسمت فلزی به اشتباه در دستگاه قرار می گیرد، به طوری که فلز را نمی توان به درستی برش داد. هر دوی این موارد منجر به یک محصول نهایی اشتباه می شود که گاهی اوقات در اواخر فرآیند تولید کشف می شود. این می تواند باعث هزینه های زیادی شود زیرا بسیاری از مراحل باید تکرار شوند یا حتی محصولات باید دور ریخته شوند. در نتیجه، این موارد خطا باید فوراً شناسایی شوند تا در هزینه ها صرفه جویی شود. در این سناریو، دو منبع داده وجود دارد، (i) یک حسگر موقعیت و (ii) یک حسگر وضعیت ابزار. سنسور موقعیت یک مقدار بولین را برمی گرداند که نشان می دهد آیا قطعه فلزی به درستی در دستگاه قرار گرفته است یا خیر. در صورت درست بودن، قطعه فلزی در موقعیت صحیح قرار دارد. در صورت کاذب، قسمت فلزی در موقعیت اشتباه قرار

دارد. سنسور وضعیت ابزار، وضعیت ابزار برش را به عنوان مقدار درصد برمی‌گرداند، که در آن ۰٪ به معنای پوسیدگی کامل ابزار است و ۱۰۰٪ ابزار استفاده نشده است. علاوه بر این، برای تحقق این سناریوی موردی، می توان سه پرس و جو CEP مختلف ایجاد کرد تا بتوان آنها را در مکان های مختلف اجرا کرد. با این حال، موتور CEP که این درخواست‌های CEP را اجرا می‌کند، باید توزیع شود، مانند ارائه شده توسط Cugola و همکاران.

در پرسش CEP Q۱ در زیر نحوه زبان پردازش رویداد (EPL) نشان داده شده است. این پرس و جو از مقادیر یک سنسور موقعیت به عنوان ورودی خود استفاده می کند و اگر سنسور مقدار false را تولید کند، یک رویداد خروجی ایجاد می کند. برای این کار، طرحی برای جریان رویداد ورودی با نام PositionSensorStream در خط ۱ تعریف میشود. به همین ترتیب، طرحی برای جریان رویداد خروجی با نام ProductionErrorStream در خط ۲ تعریف می شود. متعاقباً، پرس و جو CEP Q۱ در خطوط ۴ و ۵ تعریف می شود.

```
۱. create schema PositionSensorStream (value boolean);
۲. create schema ProductionErrorStream (value boolean);
۳.
۴. @Name('Q۱') insert into ProductionErrorStream
۵.              Select * from PositionSensorStream (value=false);
```

در پرس و جوی بعدی CEP Q۲ میتوان زمانی شرایط ابزار کمتر از ۸۰ درصد شود، یک رویداد خروجی تولید شود.

```
۱. create schema ConditionSensorStream(value integer);
۲. create schema ProductionErrorStream(value boolean);
۳.
۴. @Name('Q۲') insert into ProductionErrorStream
۵.              Select * from ConditionSensorStream(value<۸۰);
```

در نهایت پرس و جو CEP Q۳ ، به عنوان یک پرس و جو تجمعی عمل می کند که از رویدادهای خروجی تولید شده توسط پرس و جوهای Q۱ و Q۲ به عنوان ورودی استفاده می کند.

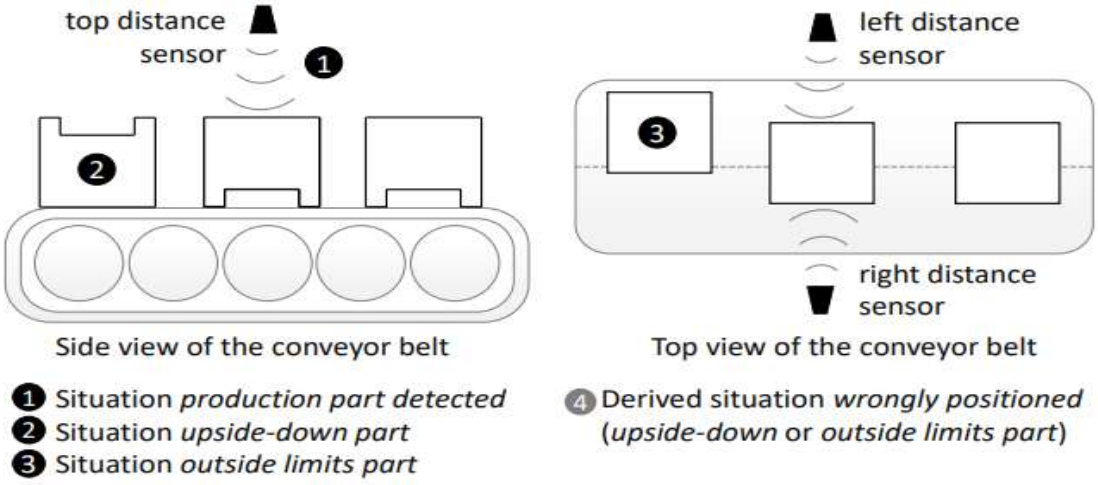
```
۱. create schema ProductionErrorStream(value boolean);
۲. @Name('Q۳') select count(*)
۳.              From ProductionErrorStream.win:time(۱۰sec)
۴.              Having count(*)>۱
```

در Q۳ بررسی می کند که آیا Q۱ یا Q۲ بیش از یک رویداد را برای یک پنجره زمانی ۱۰ ثانیه ای ایجاد می کند. پنجره زمانی برای جلوگیری از تشخیص رویدادهای مثبت کاذب به دلیل نقاط پرت استفاده می شود. فقط اگر چندین رویداد در طول این پنجره زمانی تولید شود، می توان مطمئن شد که یک خطای تولید واقعا رخ داده است. علاوه بر این، یک سرویس اعلان وجود دارد که به عنوان سینک داده عمل می کند، که رویدادهای Q۳ را دریافت می کند و به فرد مسئول برای مقابله با خطای رخ داده اطلاع می دهد، به عنوان مثال، یک مهندس تعمیر و نگهداری که می تواند قطعه تولید آسیب دیده را جایگزین کند، یا یک کارگر طبقه کارگاه که می تواند قسمت اشتباه را از فرآیند تولید حذف کند. در نهایت، نرم افزار برای اجرای پرس و جوهای CEP همانطور که در فصل ۶ توضیح داده شده است، مستقر می شود که به طور خودکار تشخیص اختلال را شروع می کند.

۲-۷ اجرای تشخیص اختلال

این بخش نحوه اجرای تشـخیص اختلال را بر اسـاس مفاهیم توضـیح داده شده در بخش ۷-۱ توضیح می دهد. برای این کار، موقعیت‌هایی که قبلاً به‌عنوان الگوهای موقعیت‌ها مدل‌سـازی شده‌اند، ابتدا به نمایش‌های اجرایی و مبتنی بر رویداد، یعنی پرس‌وجوهای CEP تبدیل می‌شـوند. سـپس این نمایش‌ها توسـط موتورهای CEP اجرا می‌شـوند، که داده‌های نظارتی ورودی را پردازش می‌کنند و زمانی که اختلالات بر اسـاس نمایش‌های اجرایی شـناسـایی می‌شـوند، اعلان‌هایی ایجاد می‌کنند.

در شـکل ۱۵، نمونه ای انتزاعی از سناریوهای تولید در دنیای واقعی نشان داده شده است که اختلالات در فرآیندهای تولید را به موقع تشـخیص می دهد. در این مثال، قطعات تولیدی روی یک تسـمه نقاله نظارت می‌شـوند تا تشخیص داده شود که چه زمانی اشتباه قرار گرفته‌اند.

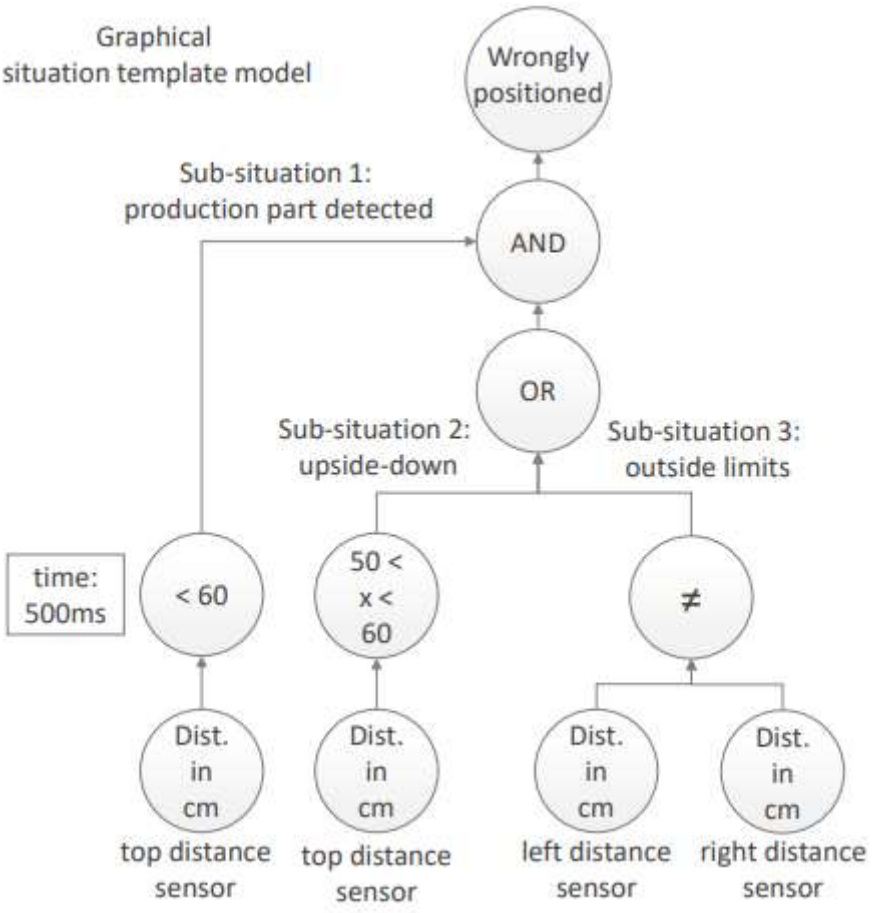


شکل ۱۵

تشـخیص اختلال بر اسـاس داده های تولید شده توسط چندین سنسور فاصله است که به تسمه نقاله متصل شده اند. در این سناریو چهار موقعیت تعریف شده است.

۱. وضـعیت تولید : نشـان دهنده وجود یک قطعه تولیدی بر روی تسـمه نقاله است. این وضعیت به تنهایی لزوماً نشان دهنده اختلال نیست، اما وقوع آن برای موقعیت های تعریف شده زیر ضروری است.
۲. وضعیت وارونه نشان دهنده یک قطعه تولید وارونه است و در نتیجه اختلالی در روند تولید رخ داده است.
۳. وضـعیت خارج از محدوده نشـان می دهد که قسـمت تولید خارج از حدود مجاز به چپ یا راسـت قرار گرفته است و بنابراین با اختلال در فرآیند تولید مطابقت دارد.
۴. موقعیت مشتق شده به اشتباه ترکیبی از موقعیت های فوق الذکر است و نشان می دهد که یک قطعه تولیدی یا وارونه است یا خارج از محدوده مجاز قرار گرفته است و بنابراین به اشتباه روی تسمه نقاله قرار گرفته است.

در شـکل ۱۵، یک الگوی موقعیت ترکیبی از موقعیت های فوق الذکر به تصویر کشیده شده است. این الگوی وضعیت به طور خودکار به یک نمایش اجرایی تبدیل می شود که در پایین نشان داده شده است.



شکل ۱۶

در این مثال، نمایش اجرایی با اســتفاده از زبان پرس و جو CEP EPL که می تواند توســط موتور Esper CEP اجرا شــود، تعریف شــده اســت. با این حال، طیف گســترده‌ای از قالب‌های اجرایی مبتنی بر CEP وجود دارد که می‌تواند توسط این رویکرد استفاده شود.

```
Select * From pattern
[ (every A۲_stream=DistanceSensorStream (sensorID='distTop', distance < ٦٠)>(timer:interval (٥٠٠msec) and
not DistanceSensorStream (sensorID='distTop', distance > ٦٠))) and
(( every A١_stream = DistanceSensorStream (sensorID='distTop',distance>٥٠,distance<٦٠)) or
 ( every A٠_stream=DistanceSensorStream (sensorID='distLeft') -> (A١_stream=DistanceSensorStream
(sensorID='distRight', A٠_stream.distance != distance) and not distanceSensorStream(sensorID='distLeft'))))]
```

برای تبــدیــل خودکــار یــک الگوی موقعیت به یک پرس و جو CEP، گره های الگوی موقعیت به منظور فرموله کردن یــک الگوی رویــداد پیچیــده، کــه از عبارات الگوی ترکیب شـــده از طریق عملگرهای منطقی (به عنوان مثال، یا، و) تشکیل شده است، پیمایش می شوند. در این رویکرد، الگوی رویداد پیچیده بر اساس مجموعه ای از گره های شرطی ســاخته می شــود که توســط گره های عملیاتی تجمیع می شــوند. هر گره شــرط مربوط به یک عبارت الگو است، در حالی که یک گره عملیاتی مربوط به یک عملگر منطقی است.

برای ساختن عبارت الگو برای گره شرط، شناسه حسگر نظارت شده و نوع حسگر از IoTEM بازیابی می‌شوند. برای ترسیم اینکه کدام حسگر فیزیکی برای یک شرایط خاص در الگوی موقعیت استفاده می شود، نقش حسگر، به عنوان مثال، "حسگر فاصله بالا" در گره زمینه مشخص می شود و تبدیل حسگر ثبت شده را با این نقش بازیابی می کند. دستور زیر ساختار یک عبارت الگو را بر اساس یک گره شرط نشان می دهد.

```
conditionNode_patternExpression=<sensor_type_stream>
        (sensor_id='<monitored_sensor_id>',
        sensor_role='<contextNode_sensor_role>',
        <conditionNode_condition>)
```

در شکل ۱۶ در پایین، پرس و جو CEP برای تشخیص موقعیت اشتباه نشان داده شده است. این پرس و جو CEP پیچیده است و بنابراین ایجاد دستی دشوار است. علاوه بر این، پرس و جوهای CEP ممکن است بسته به پیچیدگی موقعیتی که باید تشخیص داده شود، شلوغ شوند. با ارائه تبدیل خودکار از الگوهای موقعیت به پرس و جوهای CEP، متخصصان دامنه از بار ایجاد چنین پرس و جوهای پیچیده CEP خود رها می شوند. در نهایت، تشخیص اغتشاش با استقرار جستجوهای CEP حاصل در موتور اجرای CEP آغاز می شود.

این پایان نامه کارشناسی ارشد یک زبان پرس و جو انتزاعی را ارائه می دهد که ویژگی های مشترک CEP را در بین موتورهای مختلف CEP بررسی شده بیان می کند، در نتیجه، این پایان نامه علاوه بر استفاده از الگوهای موقعیت برای تشخیص موقعیت، مدل سازی با استفاده از این زبان پرس و جو انتزاعی را نیز امکان پذیر می کند. شبیه به الگوهای موقعیت، پرس و جوهایی که با استفاده از زبان پرس و جوی انتزاعی طراحی شده مدل‌سازی می‌شوند نیز به جستارهای CEP تبدیل می‌شوند که می‌توانند توسط موتورهای CEP خاص اجرا شوند. این امکان پشتیبانی از محیط‌های اجرایی مختلف را فراهم می‌کند و از وابستگی به یک موتور اجرای CEP خاص (قفل فروشنده) اجتناب می‌کند.

۷-۲-۱ سفارشی سازی و تهیه موتورهای CEP

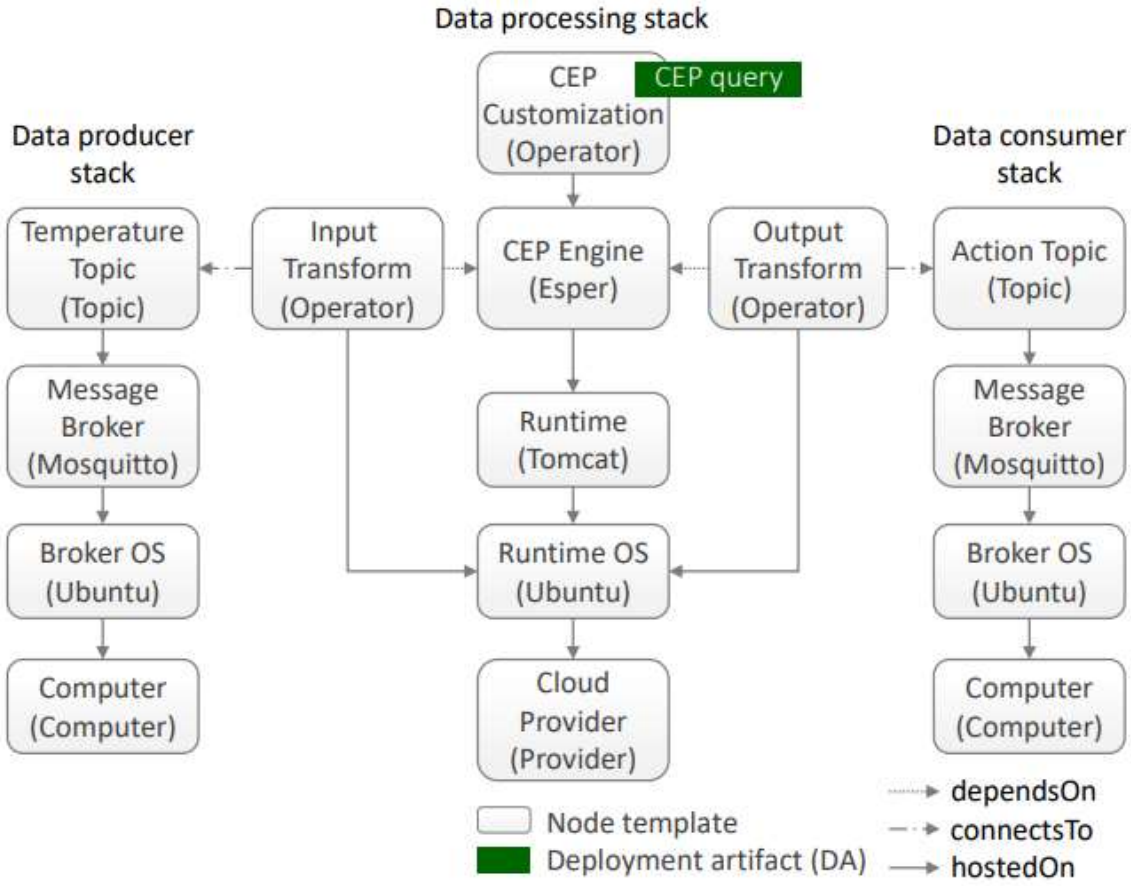
نرم‌افزارهایی که قابلیت‌های CEP را ارائه می‌کنند، عبارتند از Esper, flowthings.io, FIWARE CEP GE یا Odysseus . علاوه بر این، رویکردهای بسیاری برای ارائه موتورهای CEP به روشی عمومی وجود دارد، مانند Docker، Amazon AWS، Ansible یا Vagrant.

با این حال، رویکردهای عمومی که امکان ارائه موتورهای استاندارد CEP را فراهم می‌کنند، فاقد سفارشی‌سازی مورد نیاز حوزه اینترنت اشیا هستند. به‌طور دقیق‌تر، محیط‌های اینترنت اشیا از نظر اشیاء، برنامه‌های کاربردی و درخواست‌های پیوسته برای پردازش داده‌ها، بسیار ناهمگن هستند. در نتیجه، استفاده از خدمات ابری استاندارد برای چنین موتورهای CEP نیاز به تلاش سفارشی سازی بالایی دارد.

این مراحل سفارشی سازی عبارتند از: (i) پیکربندی موتور CEP، (ii) اتصال منابع داده و مخازن با نوشتن و استقرار کد اپراتور پیچیده، و (iii) استقرار جستجوهای CEP.

وقتی این مراحل به صورت دستی انجام شوند، خسته کننده و مستعد خطا هستند. برای مقابله با این مسائل، رویکردی را برای سفارشی‌سازی و ارائه موتورهای CEP، از جمله محیط‌های IoT، اپراتورهای مورد نیاز و جستارهای

CEP ارائه می‌کنیم. این رویکرد منجر به کاهش زیادی در تلاش سفارشی‌سازی هنگام استفاده از CEP در محیط‌های IoT می‌شود. علاوه بر این، چنین رویکرد سفارشی‌سازی شده، امنیت داده‌ها را از طریق ایجاد نمونه‌های ثابت و غیرقابل تغییر موتورهای CEP، مشابه نماها در پایگاه‌های داده، افزایش می‌دهد. این رویکرد مبتنی بر یک مدل توپولوژی TOSCA مستقل است،مدل توپولوژی باید شامل تمام اجزای نرم افزاری لازم برای راه اندازی موتور CEP باشد. چنین مدل توپولوژی در شکل ۱۷ نشان داده شده است.



شکل ۱۷

در این مثال، یک سناریوی ساده‌شده سیستم HVAC مدل‌سازی می‌شود، که در آن دمای یک اتاق به طور مداوم نظارت می‌شود تا زمانی که دما از منطقه آسایش حرارتی فراتر می‌رود یا کمتر می‌شود، واکنش نشان دهد. سفارشی‌سازی و تهیه موتورهای CEP حداقل به اجزای زیر نیاز دارد:

(۱) یک موتور CEP برای پردازش و جمع‌آوری داده‌های ورودی، (۲) جستارهای CEP در یک زبان پرس و جو سازگار، که نحوه پردازش داده‌های ورودی را برای محاسبه بالاتر توسط موتور CEP تعریف می‌کند. -موقعیت‌های سطح بر اساس داده‌های سطح پایین، (۳) تولیدکنندگان داده، که داده‌های سطح پایین را به موتور CEP ارائه می‌کنند، و (۴) مصرف‌کنندگان داده، که موقعیت‌های سطح بالاتر به دست آمده توسط موتور CEP را دریافت و پردازش می‌کنند.

بنابراین، توپولوژی به سه پشته تقسیم می شود: پشته تولید کننده داده، پشته پردازش داده حاوی موتور CEP و پرس و جوهای CEP، و پشته مصرف کننده داده. پشته تولید کننده داده شامل یک دستگاه اینترنت اشیا، یک حسگر دما و زیرساختی است که از طریق یک واسطه پیام به مقادیر حسگر دسترسی پیدا می کند. به عنوان مثال، مقادیر دما توسط یک سنسور متصل به Raspberry Pi اندازه‌گیری می‌شود، که اپراتور را برای استخراج این مقادیر و ارسال آن‌ها به موضوع خاصی که در یک کارگزار پیام میزبانی شده است، اجرا می‌کند. در شکل ۱۷، الگوهای گره برای شی IoT و عملگر استخراج به دلایل ساده حذف شده اند. نمونه ای از پشته تولید کننده داده شامل این الگوهای گره در شکل ۸ نشان داده شده است. پشته پردازش داده نشان داده شده در وسط شکل ۱۷ شامل زیرساخت و موتور CEP است که مقادیر حسگر را نظارت و پردازش می کند.

این پشته می‌تواند به دو صورت ایجاد شود: (۱) مدل‌سازی موتور CEP که بر روی زیرساخت و اجزای پلت فرم خود میزبانی می‌شود یا (۲) به عنوان یک سرویس، به طور مثال ارائه شده توسط یک ارائه دهنده ابر خارجی، مانند Amazon AWS یا Microsoft Azure.

در حالت اول، پشته توپولوژی معمولاً از هفت الگوی گره تشکیل شده است، همانطور که در شکل ۱۷ نشان داده شده است. با شروع از پایین، یک الگوی گره ارائه دهنده مربوط به ارائه دهنده منابع سخت افزاری است که ماشین های مجازی را ایجاد و اجرا می کند. علاوه بر این، قالب گره سیستم عامل Runtime سیستم عامل ماشین مجازی را مدل می کند، در حالی که الگوی گره Runtime زمان اجرا لازم را مدل می کند. G

یک وب سرور برای موتور CEP که در آن مستقر می شود. در این پایان نامه، رویکرد خود میزبانی فرض شده است تا رویکردی را ارائه دهد که عمومی است و به یک ارائه دهنده ابر خاص وابسته نیست. در ادامه، قالب های گره موتور سفارشی سازی شده CEP به تفصیل شرح داده شده است. الگوی گره موتور CEP نشان دهنده خود سیستم CEP است، به عنوان مثال Esper. این الگوی گره، که در قالب گره‌های Runtime میزبانی می‌شود، باید شامل موارد زیر باشد: (۱) یک مصنوع پیاده‌سازی (IA) برای نصب، پیکربندی و راه‌اندازی موتور CEP، و (۲) یک ویژگی ارائه‌دهنده نقطه پایانی API موتور CEP به برای سفارشی سازی، به عنوان مثال، برای فشار دادن رویدادها، استقرار پرس و جوها، و تعریف انواع رویداد استفاده شود. معمولاً، (۳) یک آرتیفکت استقرار (DA) باید ارائه شود که حاوی باینری‌های نصب‌کننده‌های موتور باشد. اگر این DA ارائه نشده باشد، این باینری ها باید به صورت پویا توسط IA بازیابی شوند.

هنگامی که موتور CEP راه اندازی شد، سفارشی سازی موتور CEP توسط الگوی گره سفارشی سازی CEP، که انواع رویداد را تعریف می کند، تحقق می یابد. مانند ، رویدادهای داده ورودی و خروجی.

به‌طور دقیق‌تر، داده‌ها بر اساس الگوی گره موضوعی مدل‌سازی‌شده ارائه یا مصرف می‌شوند. مثلا. الگوهای گره Temperature Topic و Action Topic در مدل توپولوژی نشان داده شده در شکل ۱۷.

علاوه بر این، الگوی گره سفارشی‌سازی CEP حاوی پرس‌وجوهای CEP سفارشی است که نحوه پردازش داده‌های ورودی رویدادهای ورودی تعریف‌شده توسط موتور CEP را تعریف می‌کند. هر دو تعاریف نوع رویداد و پرس و جوهای CEP به عنوان مصنوعات استقرار، به عنوان مثال، اشیاء یکپارچه، مانند فایل های متنی، که در موتور CEP مستقر

می شـــوند، نشـــان داده می شـــوند. علاوه بر این، الگوی گره ســـفارشـــی‌ســـازی CEP حاوی ویژگی‌هایی اســـت که سفارشی‌سازی‌های اضافی خاص سیستم، مانند (غیرفعال‌سازی) بهینه‌سازی‌های پرس و جو را تعریف می‌کنند.
برای اتصـال پشـته تولیدکننده داده به موتور CEP از طریق الگوی انتشـار اشـتراک، اپراتور مورد نیاز است که (i) در موضوعات تولیدکننده داده مشترک شود، (ii) داده های دریافتی را به قالبی تبدیل کند که موتور CEP درک میکند، و (iii) این را فشـار دهد. داده ها را از طریق رابط ارائه شـده و نقطه پایانی API به موتور CEP تبدیل کرد. برای آن، قالب گره تبدیل ورودی ارائه شـده اسـت که شامل چنین عملگر به عنوان یک مصنوع توسعه است. این الگوی گره به الگوی گره موضـوع دما وصـل می‌شـود و به الگوی گره موتور CEP بسـتگی دارد، زیرا نیاز به ارائه نقطه پایانی API موتور CEP برای انتقال رویدادها به آن دارد. علاوه بر این، الگوی گره حاوی یک آرتیفکت پیاده سـازی اسـت که این مصـنوعات اسـتقرار را در زمان اجرا مربوطه مسـتقر می کند. در مثال نشـان داده شـده، این زمان اجرای موتور CEP است.
با این حال، اگر قالب گره تبدیل ورودی به نوع متفاوتی از زمان اجرا نیاز داشـته باشد، می‌توان یک الگوی گره اضافی مدل‌سـازی کرد. مشابه با اتصال پشته تولید کننده داده، یک اپراتور دیگر لازم است تا اتصال پشته مصرف کننده داده به موتور CEP را درک کند که (i) وضـعیت حاصـل را که توسط موتور CEP بر اساس جستارهای CEP محاسبه شده اسـت، دریافت می کند، (ii) آن را به قالب داده ای تبدیل می کند که کارگزار پیام می فهمد، و (iii) آن را در موضوع مصرف کننده داده منتشر می کند.
بنابراین، قالب گره تبدیل خروجی ارائه شـده اسـت که شـامل چنین عملگر به عنوان یک مصـنوع توسـعه اسـت. این الگوی گره همچنین ممکن اسـت به چندین الگوی گره موضـوعی (به عنوان مثال، موضـوع اقدام) متصـل شـود و به الگوی گره موتور CEP بستگی دارد، زیرا به نقطه پایانی API موتور CEP نیز نیاز دارد که اپراتور موقعیت های مشتق شده را از آن دریافت می کند.
شـبیه به قالب گره تبدیل ورودی، تبدیل خروجی همچنین حاوی یک مصـنوع پیاده سـازی مربوطه است که استقرار این اپراتور را در زمان اجرا مناسـب انجام می دهد. مدل سـازی موتور CEP به عنوان یک سـرویس ارائه شـده توسط یک ارائه دهنده خارجی کمی با مدل در شکل ۱۷ متفاوت است.
قالب‌های گره ســـیســـتم‌عامل Runtime و Runtime نیازی به مدل‌ســـازی ندارند، با این حال، زیرســـاخت‌های قالب‌های گره تبدیل ورودی و تبدیل خروجی هنوز باید فراهم شـــود. مانند قبل، آنها می توانند با اســـتفاده از نقطه پایانی API قالب گره CEP Engine با موتور CEP ارتباط برقرار کنند.
به طور خلاصـه، برای ایجاد پشـته پردازش داده در مدل توپولوژی TOSCA، حاوی موتور سفارشی CEP، اجزای ذکر شـده در بالا باید مدل‌سـازی شـده و همانطور که توضـیح داده شـد، متصل شوند. پشته مصرف کننده داده به روشی مشابه پشته تولیدکننده داده کار می کند، زیرا بر اساس الگوی انتشار-اشتراک نیز ساخته می شود.
مصـرف‌کنندگان داده می‌توانند، برای مثال، برنامه‌هایی باشـند که در موقعیت‌های محاسـبه‌شـده توسـط موتور CEP واکنش نشـان می‌دهند. این پشـته شـامل یک دسـتگاه IoT، یک محرک و علاوه بر این، زیرسـاختی متشـکل از یک واســـطـه پیـام اســـت کـه از طریق آن می توان محرک را کنترل کرد. بـه عنوان مثـال، یـک محرک هواکش بـه Raspberry Pi وصل شده است.

در این دستگاه اینترنت اشیا، یک اپراتور کنترل میزبانی می‌شود، که مشترک موضوع اقدام خاص یک واسطه پیام برای استخراج موتورهای CEP می‌شود و یکی دیگر از ویژگی‌های مهم را فعال می‌کند: امنیت داده‌ها. مشابه مفهوم نماهای فقط خواندنی در سیستم های مدیریت پایگاه داده رابطه ای، این رویکرد قادر است درجه قرار گرفتن در معرض داده های اساسی را از طریق این رویکرد سفارشی سازی محدود کند.

این را می توان با ممنوع کردن پرس و جو و داده های اضافی در موتور CEP و همچنین خروجی اضافی از طریق کپسوله کردن موتور CEP (به عنوان مثال، از طریق مسدود کردن پورت) پس از تهیه آن و فقط اجازه دادن به ارتباط از طریق موضوعات مستقر شده محقق کرد.

علاوه بر این، با انجام این کار، مصرف‌کنندگان داده‌های پردازش‌شده توسط موتور CEP نمی‌توانند نحوه دستیابی به نتایج را بازیابی کنند، به‌عنوان مثال، ساختار ملموس پرس و جو CEP پنهان می‌شود.

۷-۲-۲ کلاس های اختلال (Disturbance classes)

با توجه به ماهیت پویای محیط های اینترنت اشیا، ممکن است در صورت بروز اختلال در محیط اینترنت اشیا، پیکربندی مجدد اپراتورهای DSPM مستقر شده نیاز باشد. بخش ۷-۱ ابزارهای مدل سازی تشخیص اختلال را ارائه می کند. این بخش یک نمای کلی از اینکه کدام اختلالات مربوط به مدل‌سازی و شناسایی در محدوده این پایان‌نامه است را ارائه می‌کند.

این پایان نامه دو کلاس اصلی را برای اختلالات در نظر می گیرد: (۱) تغییرات در زیرساخت شبکه محیط IoT، و (۲) تغییرات در DSPM اجرا شده در این محیط IoT.

فهرست زیر لیستی از اختلالات را نشان می‌دهد که باید شناسایی شوند، ولی کامل نیست. این اختلالات می تواند از طریق تغییرات در محیط اینترنت اشیا یا در DSPM یک برنامه اینترنت اشیا رخ دهد.

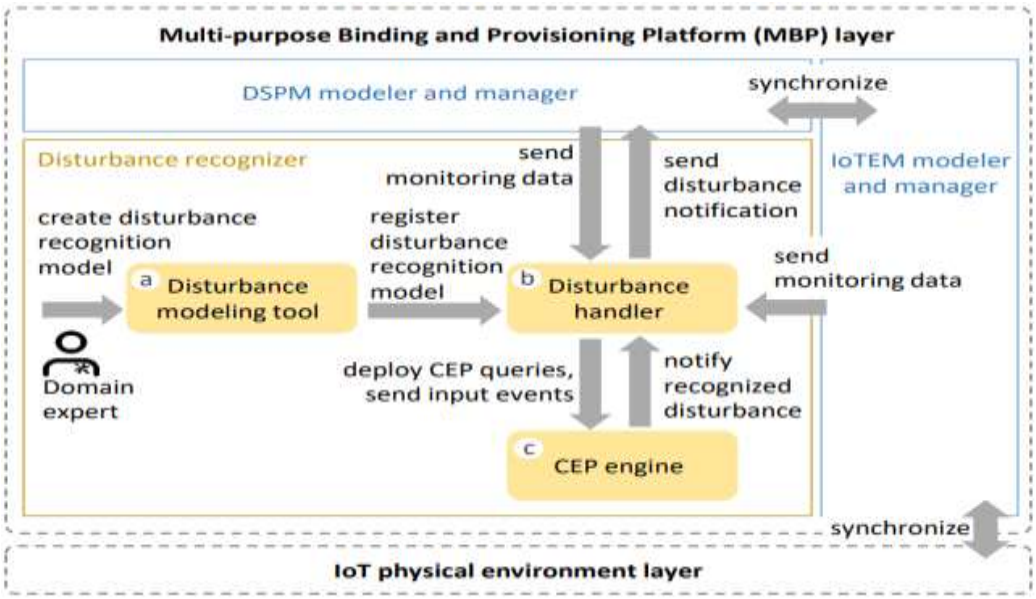
- تجزیه شی IoT (IoT object breakdown): در صورتی که یک شی IoT کار خود را متوقف کند، اگر شی IoT مربوطه به DSPM مستقر شده (dDSPM) تعلق داشته باشد، مثلا، اگر شی IoT در حال اجرای یک اپراتور باشد یا اگر یک منبع داده باشد، می تواند باعث اختلال شود. برای این، نظارت بر وضعیت اشیاء IoT انجام می شود. چنین مکانیزم‌های نظارتی اعلان‌هایی درباره رسیدن مقادیر بحرانی به آستانه یا خرابی‌های شرایط در اشیاء اینترنت اشیا ایجاد می‌کنند.
- به روز رسانی شی IoT (IoT object update). اگر قابلیت های به روز شده دیگر هیچ نیاز DSPM را برآورده نکنند، تغییرات در قابلیت ها و ویژگی های یک شی IoT ممکن است منجر به اختلال شود. علاوه بر این، اگر یک شی اینترنت اشیا، به عنوان مثال، یک ویژگی مهم، مانند آدرس IP آن، تغییر کند، ممکن است شی اینترنت اشیا غیر قابل دسترس شود یا جریان داده به خطر بیفتد.
- کم شدن منابع در اجرا (IoT object running low on resources). اگر یک شی IoT در طول زمان بیش از حد بارگذاری شود یا اگر وضعیت باتری آن کم باشد، باعث شه این گره پردازشی از کار بیفتد، این وضعیت ممکن است منجر به اختلال شود. در این حالت، در صورت امکان، می‌توان به شی اینترنت اشیا منابع بیشتری داد، یا اینکه اپراتور باید به یک شی اینترنت اشیا دیگر نگاشت شود.

- شـی IoT به IoTEM اضافه شده است. هنگامی که یک شی IoT جدید به IoTEM اضافه می شود که در آن یک DSPM قبلاً مسـتقر شـده است، در نتیجه، محیط IoT منابع اضافی را فراهم می کند که می تواند دفعه بعد که نقشه برداری از اپراتورها بر روی اشیاء IoT انجام شود در نظر گرفته شود.
- پردازش حذف اپراتور. در صـورتی که یک اپراتور پردازش، منبع داده یا سـینک داده از یک DSPM حذف شود، باید منابع تخصیص یافته آزاد شود و جریان داده باید بررسی تا مطمئن شود که به خطر نیفتاده است.
- پردازش بـه روز رسـانی اپراتور. تغییرات در الزامات یک اپراتور پردازش می‌تواند منجر به اختلال شـود اگر نیازهای جدید یا به‌روزرسانی شده دیگر توسط شی IoT تطبیق شده برآورده نشود.
- اپراتور پردازش به DSPM اضافه شد. هنگامی که یک اپراتور پردازش جدید، منبع داده یا سینک داده به یک DSPM مستقر شده اضافه می شود، عنصر جدید باید بر روی یک شی IoT مناسب نگاشت شود.

هنگامی که چنین تغییری تشـخیص داده شـد، در صـورتی که اختلال باشد، اقدامات اصلاحی ممکن می تواند توسط متخصـصـان دامنه یا تحلیلگران دامنه انجام شـود. مانند اسـتقرار مجدد اپراتورهای پردازشی ضروری است (به عنوان مثال، مهاجرت اپراتور). البته حل و اجرای اقدامات اصلاحی برای تشخیص اختلالات بخشی از کار آینده این پایان نامه اسـت. برای این کار، نقشـه برداری مجدد کامل از اپراتورهای پردازش و اشـیاء IoT باید انجام شـود. سـپس می توان طرح نقشه برداری قبلی یک dDSPM را با طرح نقشه برداری جدید مقایسه کرد و تفاوت را استخراج کرد. بر اساس این تفـاوت، اپراتورهای پردازش را می توان دوباره مسـتقر کرد و dDSPM را می توان بر این اسـاس به روز کرد. استقرار مجدد اپراتورهای بدون وضعیت کار ساده تری است، زیرا چنین اپراتورهایی فقط باید دوباره مستقر شوند.

۳-۷ جزئیات معماری و پیاده سازی تشخیص اختلال

در معماری کلی بخش تشـخیص‌دهنده اختلال ابزاری را برای نظارت بر اشـیاء IoT و DSPMهای مسـتقر شـده به منظور شناسایی اختلالات در طول پردازش داده‌ها فراهم می‌کند که این بخش در شکل ۱۸ نشان داده شده است.



شکل ۱۸

مدل‌سازی اختلال‌هایی که باید شناسایی شوند توسط متخصصان حوزه در ابزار مدل‌سازی اختلال بخش (a) انجام می‌شود، که اختلالات مدل‌سازی‌شده را در کنترل‌کننده اختلال بخش (b) ثبت می‌کند. کنترل‌کننده اختلال، مدل‌های تشخیص اختلال را به جستارهای CEP تبدیل می‌کند (شکل ۱۶)، که روی موتور بخش (c) CEP مستقر شده‌اند. علاوه بر این، کنترل کننده اختلال داده های نظارتی را در مورد محیط اینترنت اشیا و اپراتورهای پردازش دریافت می کند. این داده های نظارتی به رویدادهای ورودی CEP تبدیل شده و به موتور CEP ارسال می شوند.

هنگامی که موتور CEP یک اختلال را تشخیص داد، به کنترل کننده اختلال اطلاع می‌دهد، که اعلان‌های مربوط به اختلالات را به مؤلفه مدل‌ساز و مدیر DSPM (به بخش ۴-۲-۳ مراجعه کنید) و به داشبورد MBP ارسال می‌کند. جزء تشخیص اختلال به طور نمونه اولیه به عنوان بخشی از MBP پیاده سازی شده است.

بخش ۸-۲ مروری بر عملکردهای MBP می دهد. ابزار مدل‌سازی Disturbance۱ در جاوا اسکریپت پیاده‌سازی شده است و از کتابخانه jQuery استفاده می‌کند. مدل‌های تشخیص اختلال ایجاد شده در پایگاه داده MongoDB ذخیره می‌شوند. کنترل‌کننده اختلال، مدل‌های تشخیص اختلال را به جستارهای CEP تبدیل می‌کند که می‌توانند توسط Esper CEP engine۱ پردازش شوند. کنترل کننده Disturbance در جاوا در سمت سرور MBP پیاده سازی شده است. بخش ۸-۱ معماری یکپارچه سازی تمام اجزای معماری را ارائه می دهد و نشان می دهد که چگونه آنها با هم تطبیق می یابند.

۴-۷ کارهای مرتبط

این بخش کار مرتبط با شناسایی موقعیت در محیط های IoT را شرح می دهد. بسیاری از رویکردها وجود دارند که از هستی شناسی ها برای تشخیص موقعیت استفاده می کنند با این حال، این رویکردها یا بر روی سناریوهای مورد استفاده خاص متمرکز هستند یا نمی توانند کارایی مناسب برای سناریوهای بحرانی بلادرنگ را ارائه دهند به عنوان مثال، در کارخانه های هوشمند.

این محدودیت ها در مورد کارایی در رویکردهای یادگیری ماشینی نیز رخ می دهد. در مقابل، رویکرد در این پایان نامه با تشخیص موقعیت ها در میلی ثانیه به جای ثانیه یا حتی دقیقه. این امکان کاربرد را در سناریوهای دنیای واقعی حساس زمانی، مانند کارخانه‌های هوشمند، که در آن‌ها زمان‌های تشخیص سریع از اهمیت حیاتی برخوردار است را ممکن می‌سازد.

چندین سیستم تشخیص موقعیت با استفاده از پردازش رویداد پیچیده در استفاده از هستی‌شناسی‌ها را برای مشخص کردن و تشخیص رویدادهای پیچیده پیشنهاد می‌کنند که وقوع آن‌ها را می‌توان در پیام‌های دیجیتالی که از شبکه‌های حسگر متعدد پخش می‌شوند، شناسایی کرد. هستی شناسی توسعه یافته از طریق یک رابط کاربری قابل دسترسی است، جایی که کاربر رویدادهای مورد علاقه را مشخص می کند. سپس مشخصات یک رویداد مورد علاقه به منظور تولید دستورات پیکربندی برای یک سیستم CEP پردازش می شود.

سیستم CEP بر جریان‌های داده مشخص شده نظارت می‌کند و اعلان‌هایی را تولید می‌کند که می‌توانند هنگام وقوع رویداد به مشتریان تحویل داده شوند. در این مقاله، رویدادهای پیچیده به طور مستقیم مشخص نمی شوند، بلکه به

عنوان موقعیت ها انتزاع می شـــوند. رویکرد در این پایان نامه همچنین تبدیل مشـــخصـــات رویداد (به عنوان مثال، موقعیت های مدل شـده) به پرس و جوهای CEP برای یک سـیسـتم CEP را درک می کند. تفاوت این است که این پایان نامه از هسـتی شـناسی برای مدل سازی موقعیت های مورد علاقه استفاده نمی کند. حسن و همکاران پیشنهاد می کند از CEP همراه با غنی سازی پویا از داده های حسگر به منظور تحقق آگاهی از موقعیت استفاده شود. در این رویکرد، موقعیت های مورد علاقه مستقیماً در موتور CEP تعریف می شوند، یعنی کاربر موقعیت های مورد علاقه را با استفاده از زبان های پرس و جو CEP فرموله می کند.

یک جزء غنی‌سـازی پویا، داده‌های حسگر را قبل از ارزیابی موتور CEP پردازش و غنی می‌کند. این رویکرد و رویکرد موجود در این پایان نامه به شـرح زیر اسـت: هیچ غنی سـازی دینامیکی پیچیده ای از داده های حسگر در این پایان نامه انجام نشـده اسـت. اطلاعات لازم در مورد سـنسور برای تشخیص موقعیت (به عنوان مثال، شناسایی سنسور) در حداقل نگه داشـته می شود. این اطلاعات همراه با خواندن سنسور مستقیماً در اختیار موتور CEP قرار می گیرد و هر مرحله پردازش بیشـتر داده های حسگر را انجام می دهد. علاوه بر این، به جای تعریف موقعیت ها به طور مستقیم به عنوان پرس و جوهای CEP، که بسته به موقعیت می تواند طولانی و پیچیده باشد،

این پـایـان نـامـه موقعیـت های مورد علاقه را به عنوان الگوهای موقعیت تعریف می کند. انتزاع ارائه شـــده از طریق الگوهای موقعیت، اســـتفاده از موتورهای CEP و همچنین اســـتفاده از فناوری‌های دیگر را برای تشـــخیص موقعیت ممکن می‌ســـازد. علاوه بر این، اســـتفاده از الگوهای موقعیت نیز مرحله مدل‌ســـازی موقعیت‌ها را برای کاربر تســـهیل می‌کند، به طوری که کاربر مجبور نیست با پیچیدگی فرمول‌بندی پرس و جوهای CEP سروکار داشته باشد.

فرمول پرس و جوهای CEP توسـط تبدیل ها مراقبت می شـود، که به طور خودکار پرس و جوهای CEP لازم را برای یک الگوی موقعیت معین ایجاد می کند. گلومبیفســـکی و همکاران رویکرد مشـــابهی را ارائه می دهد که زمینه را از طیف گسـترده ای از منابع برای شناسایی موقعیت با استفاده از فناوری های پردازش رویداد یکپارچه می کند. با این حال، آنها هیچ گونه انتزاعی ارائه نمی دهند، به عنوان مثال، کاربران تشـــخیص موقعیت باید خودشـــان جســـتارهای CEP را ایجاد کنند. این امر به خصـوص برای کارشـناسان حوزه، به عنوان مثال، در کارخانه ها که دانش گسترده ای در زمینه علوم کامپیوتر ندارند، دشوار است.

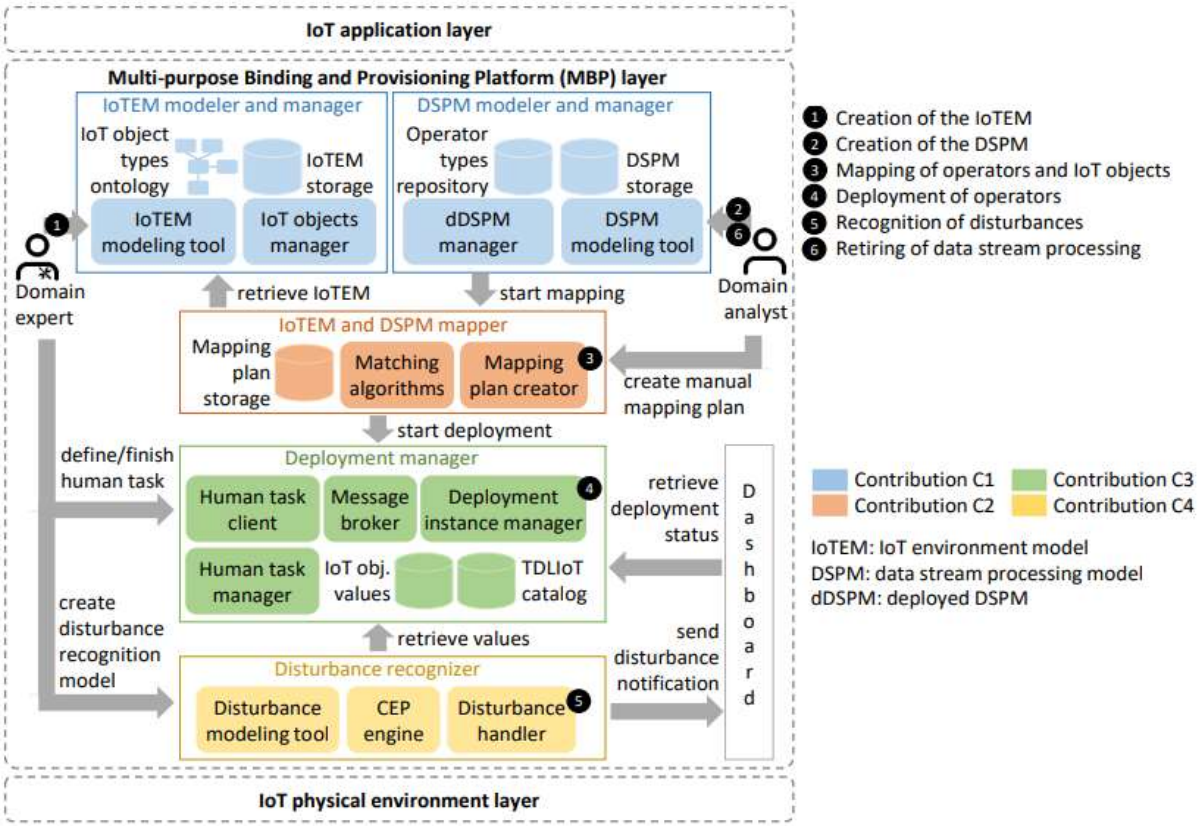
در این پایان نامه، یک انتزاع بر اساس الگوهای موقعیت و یک رابط گرافیکی ارائه شده است که امکان استفاده توسط متخصصان دامنه را بدون دانش لازم از جزئیات فنی، مانند پرس و جوهای پردازش رویداد، فراهم می کند.

فصل هشتم

ارزیابی

۸-۱ معماری یکپارچه سازی و نمونه اولیه

شـکل ۱۹ معماری دقیق حاصـل از این پایان نامه را به تصـویر می کشـد، که اجزای هر دسـتاورد را ادغام می کند و نشان می دهد که چگونه آنها با هم تطبیق می یابند. دستاورد ها با رنگ برجسته می شوند و مراحل رویکرد روشمند نیز نشان داده شده است.



شکل ۱۹

اجزای معماری و پیاده سـازی های مربوط به هر دسـتاورد به ترتیب در فصل های ۴ تا ۷ به تفصیل توضیح داده شده است. در معماری کلی (مراجعه کنید به بخش ۳-۳)، دستاورد C۱ شامل دو جزء معماری، مدل ساز و مدیر IoTEM (به بخش ۴-۱-۳ مراجعه کنید)، و مدل سـاز و مدیر DSPM (به بخش ۴-۲-۳ مراجعه کنید) را شامل می شود. این

اجزا بیشتر به اجزای کوچکتر تقسیم می شوند و در شکل ۱۹ با رنگ آبی نشان داده شده اند. بخش مدلساز و مدیر IoTEM نقطه ورود رویکرد روشمند (مرحله ۱) است و از متخصصان دامنه در طول ایجاد و مدیریت IoTEM پشتیبانی می کند.

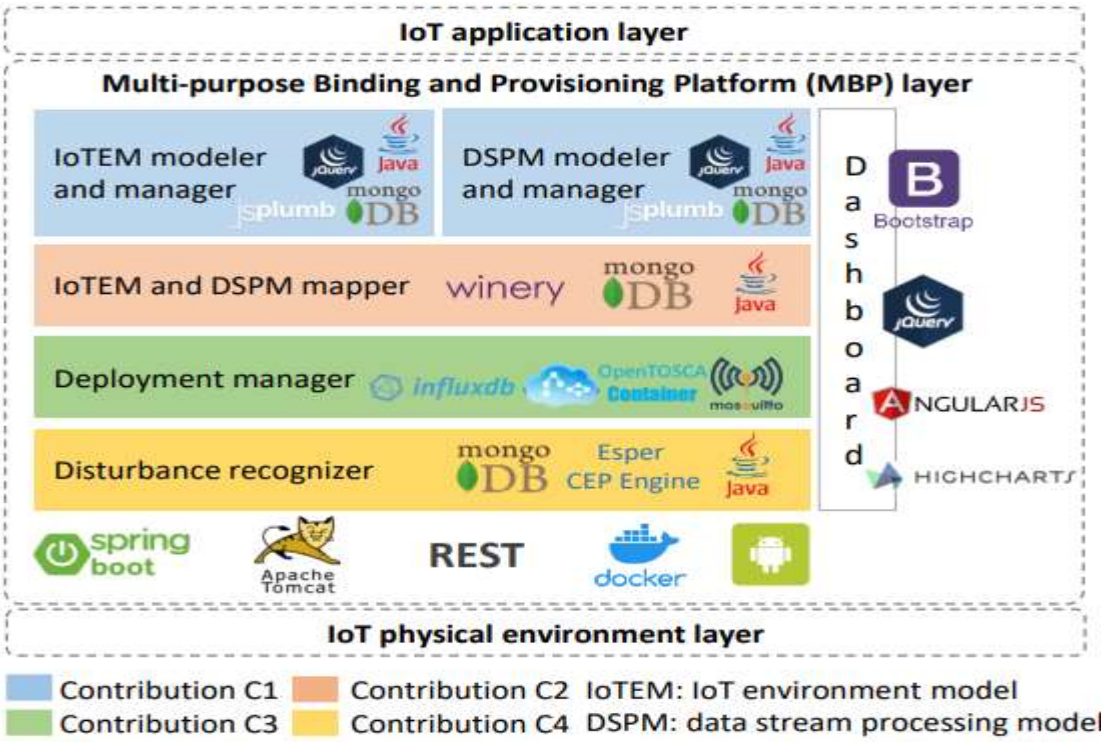
بخش مدل‌ساز و مدیر DSPM با ارائه ابزاری برای ایجاد و مدیریت DSPM (مرحله ۲)، و بعداً برای توقف DSPM مستقر در محیط اینترنت اشیا (مرحله ۶) از تحلیلگران دامنه پشتیبانی می‌کند. دستاورد C۲ شامل اجزای معماری IoTEM و DSPM mapper است که از تحلیلگران دامنه در تصمیم گیری درباره محل استقرار اپراتورهای پردازش پشتیبانی می کند.

این جزء معماری بیشتر به اجزای کوچکتر تقسیم می شود و در شکل ۱۹به رنگ نارنجی نشان داده شده است. این جزء با اجزای معماری مشارکت C۱ برای بازیابی IoTEM و DSPM ارتباط برقرار می کند. ابزاری برای تحقق نقشه IoTEM و DSPM (مرحله ۳) به طور خودکار از طریق الگوریتم ها یا به صورت دستی توسط تحلیلگران دامنه فراهم می کند. C۳ Contribution شامل بخش معماری Deployment Manager است که ابزاری را برای استقرار اپراتورها بر روی اشیاء IoT فراهم می کند (مرحله ۴).

این جزء با رنگ سبز در شکل ۱۹ نشان داده شده است. این کامپوننت طرح نقشه برداری را از مولفه معماری مشارکت C۲ دریافت می کند و بر اساس این نقشه نگاشت استقرار را آغاز می کند. وضعیت فعلی استقرار یک اپراتور را می توان در داشبورد مشاهده کرد. در صورت نیاز به اقدامات دستی توسط استقرار (به عنوان مثال، وصل کردن سنسورها)، این مؤلفه ابزاری را برای متخصصان حوزه فراهم می کند تا وظایف انسانی را تعریف کنند. (به بخش ۶-۲ رجوع کنید)، در مورد وظایف انسانی که باید اجرا شوند مطلع شوند، و وظایف انسانی را به پایان برسانند. بعد از اینکه به صورت دستی اجرا شدند.

دستاورد C۴ شامل شناسایی کننده اختلال جزء معماری است که با رنگ زرد در شکل ۱۹ نشان داده شده است. این مؤلفه ابزاری برای نظارت بر اشیاء اینترنت اشیا و DSPMهای مستقر شده به منظور شناسایی اختلالات در طول پردازش داده ها (مرحله ۵) فراهم می کند. این ابزار یک ابزار مدل‌سازی اختلال را فراهم می‌کند که کارشناسان حوزه می‌توانند آن را مدل‌سازی کرده و شناسایی اختلالات را آغاز کنند. این مؤلفه به طور مداوم مقادیر خروجی اپراتورهای مستقر و مقادیر نظارت بر اشیاء IoT را از مؤلفه معماری مشارکت C۳ بازیابی می کند. این مقادیر به موتور CEP ارسال می‌شوند، جایی که بر اساس مدل‌های تشخیص اختلال پردازش می‌شوند. در صورت شناسایی اختلالات، اعلان‌های مربوط به اختلالات به داشبورد ارسال می‌شود، جایی که می‌توان آن‌ها را برای کارشناسان دامنه یا تحلیلگران دامنه نشان داد.

در شکل ۲۰، فناوری های نرم افزاری به کار رفته در اجرای نمونه اولیه معماری MBP نشان داده شده است. پیاده سازی MBP عمدتاً از جاوا اسکریپت و انگولار برای رابط کاربری MBP و جاوا برای سرور MBP استفاده می کند. REST API بر روی چارچوب بوت Spring ساخته شده است و کل مؤلفه سرور به عنوان یک برنامه وب در حال اجرا بر روی سرور برنامه Java Apache Tomcat ارائه می شود. MBP را می توان بر روی سیستم عامل های ویندوز، مک یا لینوکس با نصب اجزای نرم افزار مورد نیاز آن به صورت جداگانه یا با یک اسکریپت نصب در لینوکس نصب کرد.



شکل ۲۰

علـاوه بر این، MBP را می توان بـه عنوان یک ظرف Docker اجرا کرد. پس از نصـــب، رابط کاربری از طریق یک مرورگر اینترنت (به عنوان مثال، کروم یا فایرفاکس) قابل دسـترسی است. رابط کاربری MBP بر اساس الگوهای بوت اسـترپ اسـت و از چارچوب جاوا اسـکریپت Angular JS استفاده می کند. نمادها از کتابخانه متریال طراحی گوگل اسـتخراج می شـوند. علاوه بر این، Highcharts کتابخانه در داشـبورد برای تجسـم مقادیر خروجی زنده و تاریخی اپراتورهای مسـتقر و نمایش داده های نظارتی اشـیاء اینترنت اشـیا استفاده شد. یک REST API که منعکس کننده همان عملکردهای ارائه شده توسط رابط کاربری MBP است در سرور MBP نیز پیاده سازی شده است.

در مورد مؤلفه مدل‌سـاز و مدیر IoTEM، ابزار مدل‌سـازی IoTEM در جاوا اسـکریپت پیاده‌سـازی شـده اسـت و از کتابخانه JavaScript jQuery استفاده می‌کند. jsPlumb Toolkit برای پیاده‌سازی یک ویرایشگر کشیدن و رها کردن برای مدل‌سـازی محیط‌های IoT اسـتفاده شـده اسـت. مدیر شی IoT در سرور MBP با استفاده از جاوا پیاده سازی می شود. پایگاه داده MongoDB برای ذخیره انواع شی IoT و IoTEM های مدل شده استفاده می شود.

در مورد مؤلفه مدل سـاز و مدیر DSPM، ابزار FlexMash به عنوان ابزار مدل سازی DSPM استفاده می شود که توسـط Hirmer و همکاران توسـعه داده شـده اسـت. در محدوده این پایان نامه، رابط گرافیکی FlexMash و مدل زیربنایی مبتنی بر JSON در پایان نامه کارشـناسی ارشد انجام شده توسط Chaudhry گسترش یافت. برنامه های افزودنی شـــامل امکان حاشـــیه نویســـی الزامات در اپراتورهای پردازش و لبه های بین اپراتورها اســـت. علاوه بر این، FlexMash برای اتصـال به MBP و امکان وارد کردن منابع و سـینک ها از IoTEM گسترش یافت. رابط گرافیکی FlexMash در جاوا اسکریپت پیاده سازی شده و از jQuery و jsPlumb استفاده می کند.

DSPM های مدل شـده در MBP در پایگاه داده MongoDB ذخیره می شـوند. مدیر dDSPM با استفاده از جاوا در سـرور MBP پیاده سـازی می شود. مخزن انواع اپراتور مربوط به یک سیستم فایل است که در محل نصب MBP پیکربندی شـده است. چندین عملگر در Python و Shell پیاده سازی شده اند که می توان آنها را در پروژه MBP GitHub[۱] یافت. در مورد مؤلفه نقشـه‌بردار IoTEM و DSPM، Winery به‌عنوان ایجادکننده طرح نقشـه‌برداری خارجی برای رویکرد نقشه‌برداری دستی استفاده می‌شود.

در این مورد، نقشـه های تطبیق بر اسـاس اسـتاندارد TOSCA اسـت و در یک سـیسـتم فایل پیکربندی شـده برای Winery ذخیره می شـود. برای رویکرد نگاشـت خودکار، خالق طرح نقشـه برداری و دو الگوریتم تطبیق، یک نوع حریص و یک نوع عقبگرد، در سـرور MBP در جاوا بصـورت پایان نامه کارشناسی توسط اشنایدر پیاده سازی شدند. در این مورد، نقشـه های تطبیقی به طور خودکار تولید شـده در پایگاه داده MongoDB ذخیره می شوند. در مورد مولفه Deployment Manager، کانتینر OpenTOSCA به عنوان یک مدیر نمونه اسـتقرار خارجی استفاده می شود که قادر است به طور خودکار اپراتورهای نقشه های نقشه بر اساس استاندارد TOSCA را مستقر کند.

در محدوده این پایان نامه، یک پایان نامه کارشـناسـی توسـط Kutger انجام شـد که مدیر وظیفه انسانی و مشتری وظیفه انسانی را پیاده سازی کرد تا ظرف OpenTOSCA را برای پشتیبانی از وظایف انسانی فعال کند. مدیر وظایف انسـانی مبتنی بر پیاده سـازی Wagner اسـت که مربوط به یک برنامه وب جاوا اسـت که روی آپاچی تامکت اجرا میشود. مشتری وظایف انسانی به عنوان یک برنامه تلفن همراه مبتنی بر اندروید پیاده سازی شد.

علاوه بر این، MBP چندین اپراتور استخراج و کنترل آماده برای استفاده را به عنوان اسکریپت‌های پایتون و شل ارائه می‌کند که سـپس می‌توانند در MBP ثبت شـوند و توسـط MBP بر روی اشـیاء IoT مسـتقر شـوند. هنگامی که اپراتورها مسـتقر شدند، با اتصال و انتشار به واسطه پیام، که با استفاده از کارگزار MQTT Mosquitto پیاده سازی می شود، مقادیر را به MBP ارسال می کنند. سپس MBP می تواند مقادیر حسگر را به صورت نمودارهای خطی در داشـبورد تجسم کند. مدیریت و پردازش داده در MBP یک معماری مبتنی بر لامبدا را پیاده‌سازی می‌کند که در آن داده‌هـای زنـده و تـاریخی را می‌توان مـدیریـت و پردازش کرد. برای ذخیره داده هـای اندازه گیری، یعنی داده های حسگر و داده های مبتنی بر مهر زمانی، از پایگاه داده سری زمانی InfluxDB استفاده می شود.

فراداده‌هـای بیشـــتر، مدل‌های محیط اینترنت اشـــیا و مدل‌های پردازش جریان داده در پایگاه‌های داده NoSQL MongoDB ذخیره می‌شـوند تا داده‌های اندازه‌گیری و ابرداده مربوطه را به وضـوح جدا کنند. کاتالوگ TDLIoT به طور نمونه در خارج از MBP به عنوان یک پروژه منبع باز GitHub پیاده سازی شده است.

این در جاوا پیاده سـازی شده است و از پایگاه داده MongoDB برای ذخیره توضیحات TDLIoT استفاده می کند. این یک رابط وب پیاده سـازی شـده در جاوا اسکریپت و یک API REST برای تعامل با کاتالوگ TDLIoT فراهم می کند. در رابطه با مولفه تشخیص اختلال، ابزار مدل‌سازی اختلال، در جاوا اسکریپت پیاده‌سازی شده است و از کتابخانه jQuery استفاده می‌کند. مدل‌های تشخیص اختلال ایجاد شده در پایگاه داده MongoDB ذخیره می‌شوند. کنترل کننده اختلال، مدل های تشـخیص اختلال را به جستارهای CEP تبدیل می کند، که می تواند توسط موتور Esper

CEP پردازش شـود. کنترل کننده Disturbance در جاوا پیاده سـازی شـده اسـت. در محدوده این پایان نامه، MBP۲Go، یک برنامه مشـتری موبایل مبتنی بر اندروید که در جاوا برای اتصـال به MBP پیاده سـازی شده است، توسـعه یافته است. MBP۲Go یک نمای کلی از اشیاء ثبت شده اینترنت اشیا در MBP ارائه میدهد. می تواند اشیاء IoT را ثبت و حذف کند و مقادیر حسگر فعلی و تاریخی را تجسم کند.

مدل‌سـازی اشـیاء اینترنت اشـیا با MBP۲Go می‌تواند با اسـکن الگوهای کد QR ارائه شده در مخزن MBP۲Go GitHub[۱] خودکار شـود. با اسـکن چنین الگوهای کد QR، MBP۲Go به طور خودکار ویژگی هایی را پر می کند که امکان استنباط برای نوع شی IoT در حال مدل سازی وجود دارد.

۸-۲ نمای کلی MBP

از زمـان رواج اینترنـت اشـیـا ، بسـیـاری از پلتفرم‌های تجاری و غیرتجاری اینترنت اشـیا برای کمک به کاربران غیرمتخصـص در مدیریت اشیاء در محیط‌های اینترنت اشیا توسعه یافتند. با این حال، اتصال و ارائه اشیاء IoT به این پلتفرم‌ها همچنان برای کاربران غیرمتخصص وظایف بسیار چالش برانگیزی است.

ما اسـاسـاً MBP را برای سـهولت مدیریت محیط‌های اینترنت اشیا طراحی کردیم، با این حال، ما همچنین یک گام عمیق‌تر به سـمت حمایت از کاربران در طول اتصـال و ارائه محیط‌های IoT برداشـتیم. در بسـیاری از پلتفرم‌های اینترنت اشـیا، مانند : FIWARE - IBM Watson IoT - OpenMTC - Microsoft Azure IoT اشـیاء به صورت دستی ثبت، متصل شده و در اختیار پلت‌فرم‌های اینترنت اشیا قرار می‌گیرند.

چنین وظایفی پیچیده هسـتند و به دانش فنی در مورد اشـیاء IoT نیاز دارند. یعنی اپراتورها (کد نرم افزار) برای اسـتخراج و ارائه داده های حسگر به برنامه های IoT و همچنین دریافت دستورات کنترل محرک از برنامه های IoT مورد نیاز هسـتند. چنین اپراتورهایی باید برای هر سـنسـور و محرک به صورت دستی ایجاد و مستقر شوند. علاوه بر این، عملکرد نظارت باید به صورت دستی نیز پیاده سازی و مستقر شود.

اسـتقرار اپراتورها به صـورت دستی مستعد خطا و زمان‌بر است، زیرا یک متخصص سخت‌افزار باید اشیاء اینترنت اشیا را پیکربندی کند، اپراتورهای لازم را برای حسگرها و محرک‌های خاص نصب کند، آنها را متصل کند و رابط‌های قابل دسـترسـی برای برنامه‌های اینترنت اشـیا ارائه دهد. در سـناریوهای دنیای واقعی، به عنوان مثال، برای تشـخیص موقعیت، نیازهای کارایی و دقت بسـیار مهم هسـتند. با این حال، این الزامات را نمی توان از طریق صـحافی و تهیه دستی برآورده کرد.

برای مقابله با مسـائل فوق، MBP برای پشتیبانی از کاربران در کل چرخه حیات محیط‌های IoT توسعه داده شد، به طوری که میزان کارهای دسـتی به حداقل ممکن برسـد. این شامل استقرار خودکار، مدیریت و نظارت بر محیط های IoT می شود. عملکردهای اصلی MBP در ادامه توضیح داده شده است.

۸-۲-۱ مدل سازی محیط های اینترنت اشیا

اشـیاء محیط IoT را می توان به صورت جداگانه یا به عنوان بخشی از یک محیط خاص اینترنت اشیا در MBP ثبت کرد. گزینه دوم علاوه بر این کاربر را قادر می سـازد تا اتصـالات بین اشـیاء IoT را در محیط IoT مدل کند. در ابزار مدلسازی MBP IoTEM، اشیاء شامل خواص و اتصالات آنها مشخص شده است.

این ویژگی ها اطلاعات خاصی را در مورد اشیاء IoT، مانند نوع شی IoT، شناسه یا آدرس MAC توصیف می کنند. برای خودکارسازی مدل‌سازی اشیاء اینترنت اشیا، MBP قالب‌های کد QR و یک برنامه تلفن هوشمند مبتنی بر اندروید را برای اسکن این کدهای QR و پر کردن خودکار ویژگی‌هایی که ممکن است برای شی مدل‌سازی شده استنباط شود، ارائه می‌کند.

علاوه بر این، با پیروی از همان روش، اشیاء IoT را می توان به طور خودکار با اتصال به نقطه اتصال Wi-Fi ثبت MBP کشف کرد. یعنی MBP به اتصالات جدید به این هات اسپات پیکربندی شده گوش می دهد و به طور خودکار مدل هایی با ویژگی های استنباط شده برای اشیاء متصل IoT ایجاد می کند.

هنگامی که یک محیط IoT مدل سازی و ذخیره می شود، می تواند به طور خودکار از طریق رابط کاربری MBP ثبت شود. در این مرحله، اطلاعات نظارتی اولیه در مورد اشیاء اینترنت اشیا را می توان از قبل در داشبوردها مشاهده کرد، مانند دسترسی به شبکه، استفاده از CPU، و دمای CPU یک دستگاه.

۸-۲-۲ استقرار اپراتورها در محیط های اینترنت اشیا

فناوری MBP چندین اپراتور استخراج و کنترل آماده برای استفاده را در قالب اسکریپت ارائه می کند که در پروژه MBP GitHub یافت می شود. یکی از اپراتورهای استخراج نمونه، مقادیر اندازه گیری یک سنسور دمای آنالوگ متصل به Raspberry Pi را می خواند.

این اپراتور مقادیر استخراج شده را از طریق MQTT که یک پروتکل ارتباطی Publicsubscribe است به MBP ارسال می کند. چنین اپراتورهایی را می توان به سادگی به دستگاه های ثبت شده مرتبط کرد و به ترتیب روی این دستگاه ها مستقر شد. از طریق این رویکرد، حسگرها به‌طور خودکار به MBP متصل می‌شوند و مقادیر اندازه‌گیری شده حسگر را می‌توان به صورت زنده مشاهده کرد و همچنین به عنوان داده‌های تاریخی در دسترس است. MBP همچنین به کاربران امکان می دهد تا اپراتورهای خود را ارائه دهند که می تواند در هر زبان برنامه نویسی پیاده سازی شود. MBP فقط به وجود اسکریپت های مدیریت چرخه حیات خاص (به عنوان مثال، نصب، شروع، توقف یا خاتمه) برای اپراتور نیاز دارد تا بتواند به طور خودکار استقرار اپراتورهای تعریف شده توسط کاربر را انجام دهد. منطق کاربردی این اسکریپت‌های مدیریتی همچنان توسط کاربر قابل تعریف است، به عنوان مثال، مشخص کردن نرم‌افزار ضروری که باید نصب شود. به این ترتیب، کاربران می توانند اپراتورهایی ایجاد کنند که نیازهای خاص خود را برآورده کنند، مانند استفاده از انواع سخت افزار خاص یا کتابخانه های برنامه نویسی.

۸-۲-۳ نظارت بر محیط های اینترنت اشیا

هنگامی که اشیاء IoT به MBP متصل می شوند، می توانند توسط کاربران از طریق داشبوردهای ارائه شده نظارت شوند. این داشبوردها وضعیت و اطلاعات آماری مربوط به دستگاه‌ها، حسگرها و محرک‌ها را نشان می‌دهند. علاوه بر این، داده‌های حسگر زنده و داده‌های حسگر تاریخی را می‌توان تجسم کرد.

برای تحقق یک نظارت خودکار بر اساس داده‌های حسگر و فعال کردن محرک‌ها به طور خودکار بر اساس اختلالات شناسایی شده، MBP ابزاری را برای تشخیص اختلالات بر اساس قوانین تعریف‌شده توسط کاربر فراهم می‌کند. چنین قوانینی، که بر اساس الگوی رویداد-شرط-عمل هستند، می‌توانند در ابزار مدل‌سازی اختلال MBP تعریف

شوند. این قوانین سطح بالا، مانند ، مدل‌های تشخیص اختلال، سپس به صورت داخلی به پرسش‌های پردازش رویداد پیچیده (CEP) تبدیل می‌شوند و با استفاده از سیستم‌های CEP مربوطه ارزیابی می‌شوند.
با به کارگیری CEP، MBP از مزیت دستیابی به پردازش داده های حسگر کارآمد و به موقع برخوردار است.

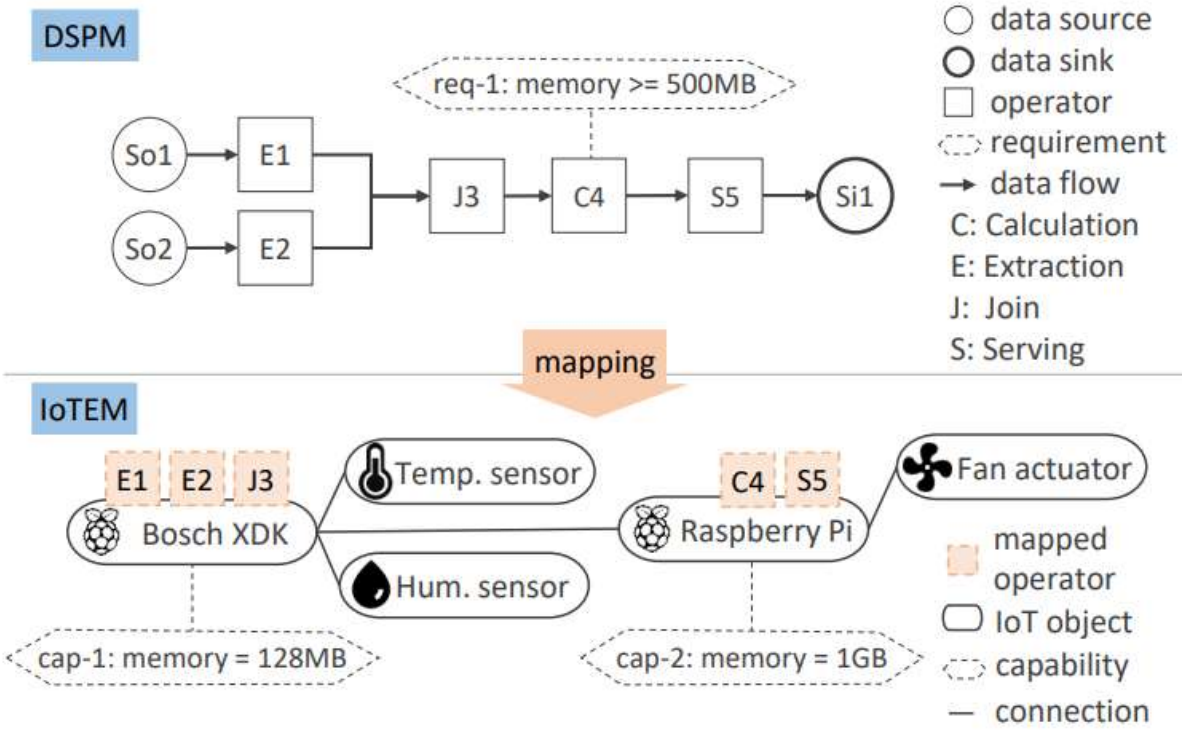
۸-۲-۴ نمونه نمایش : دفتر هوشمند

این بخش توضیح می دهد که چگونه یک سناریوی ساده اینترنت اشیا، یک دفتر هوشمند، با MBP قابل تحقق است. دفتر به عنوان اتاقی تعریف می شود که افراد وظایف کاری خود را در آن انجام می دهند. به منظور هوشمند ساختن یک دفتر، می توان آن را با قدرت محاسباتی و قابلیت های سنجش و عمل از طریق دستگاه های IoT، حسگرها و محرک ها افزایش داد.
یک روش معمول در سناریوهای دنیای واقعی، ادغام سیستم گرمایش، تهویه، تهویه مطبوع (HVAC) در اتاق ها است. هدف چنین سیستمی این است که محیط داخلی را برای سرنشینان راحت نگه دارد و در عین حال وظایفی را بر عهده می گیرد که می توانند به طور خودکار اجرا شوند. به عنوان مثال، دمای اتاق را می توان به طور خودکار برای ساکنان آن بر اساس اهداف تعریف شده توسط کاربر و اندازه گیری های مداوم حسگر تنظیم کرد.
در محیط Lego IoT که در شکل ۲۱ نشان داده شده است، چندین دفتر مینیاتوری Lego با اشیاء مختلف اینترنت اشیا تقویت شده اند که در آن بسیاری از برنامه های IoT، از جمله یک سیستم HVAC، قابل تحقق هستند.

شکل ۲۱

برای اجرای سیستم تهویه مطبوع، می توان از دو دستگاه اینترنت اشیا استفاده کرد: (i) یک دستگاه Bosch XDK مجهز به هشت حسگر، از جمله سنسورهای رطوبت و دما، و (ii) یک Raspberry Pi، که به یک محرک رله متصل است که یک فن خنک کننده را کنترل می کند.

برای تحقق سناریوی اینترنت اشیا، دستگاه‌های اینترنت اشیا، حسگرهای دما و رطوبت و محرک رله در ابزار مدل‌سازی IoTEM مدل‌سازی شده‌اند (مرحله ❶ شکل ۲۰). IoTEM مدل شده در شکل ۲۲ نشان داده شده است.



شکل ۲۲

پس از این، ابزار مدل‌سازی DSPM (ابزار FlexMash) IoTEM مدل‌سازی شده را وارد کرده و حسگر دما، سنسور رطوبت و محرک رله را به‌عنوان دو منبع داده و یک سینک داده خلاصه می‌کند. سپس یک DSPM حاوی منطق پردازش سیستم HVAC برای این IoTEM مدل‌سازی می‌شود (مرحله ❷ در شکل ۲۰).

DSPM از دو عملگر استخراج، یک عملگر اتصال، یک عملگر محاسبه و یک اپراتور سرویس تشکیل شده است. DSPM مدل شده در شکل ۲۲ نشان داده شده است.

پیاده سازی عینی این عملگرها اسکریپت های Python و Shell هستند که در MBP ذخیره می شوند. عملگرهای استخراج برای سنسورهای دما و رطوبت خاص XDK و برای محرک رله در مخزن MBP GitHub ارائه شده است. عملگر اتصال یک مقدار دما و یک مقدار رطوبت را در یک اندازه گیری بر اساس یک پنجره زمانی ترکیب می کند. اپراتور محاسبات بررسی می کند که آیا مقدار دما زیر یا بالاتر از منطقه آسایش حرارتی در داخل دفتر است، به عنوان مثال، از ۲۰ تا ۲۲ درجه سانتیگراد. این اپراتور همچنین بر اساس مقادیر دما و رطوبت محاسبه می کند که آیا سطح قالب فعلی به یک منطقه ناسالم رسیده است. اگر دما بالاتر از منطقه آسایش حرارتی باشد یا سطح قالب ناسالم باشد، این اپراتور فرمانی را ایجاد می کند که نشان می دهد فن کولر باید روشن شود. اپراتور سرویس فرمان های

روشن و خاموش را از اپراتور محاسبه دریافت می کند و محرک رله را که فن کولر را کنترل می کند، سوئیچ می کند. برای سناریوهای ساده، قوانین رویداد-شرطی که توسط تشخیص اختلال ارائه شده است، می تواند استفاده شود. برای سناریوهای پیچیده‌تر، مانند سناریوی Smart Office، مدل‌سازی یک DSPM مورد نیاز است که ابزار پیچیده‌تری برای مدل‌سازی پردازش داده‌ها در میان چندین دستگاه، حسگرها و محرک‌ها فراهم می‌کند.

از طریق ابزار FlexMash، الگوریتم نگاشت خودکار شروع شده و یک طرح نقشه برداری ایجاد می شود (مرحله ❸ در شکل ۲۰). نوع حریصانه الگوریتم تطبیق برای تصمیم گیری در محل استقرار عملگرها استفاده می شود. یک نقشه برداری ممکن در شکل ۲۲ نشان داده شده است.

اپراتورهای استخراج E۱ و E۲ به دستگاه IoT نگاشت شده اند که حسگرها به صورت فیزیکی به آن متصل می شوند. به طور مشابه، اپراتور سرویس دهنده S۵ به دستگاه اینترنت اشیا، که محرک رله به طور فیزیکی به آن متصل است، نگاشت می شود. پس از آن، Join Operator J۳ به نزدیکترین دستگاه به اپراتورهای استخراج مستقر شده نگاشت می شود. اپراتور محاسبه C۴ به دستگاهی که با نیاز اپراتور مطابقت دارد، یعنی حداقل ۵۰۰ مگابایت حافظه، نگاشت می شود. بر اساس طرح نقشه برداری، استقرار اپراتورها آغاز می شود (مرحله ❹ در شکل ۸,۱).

هنگامی که استقرار کامل شد، اطلاعات نظارتی اولیه در مورد دستگاه ها را می توان در داشبوردها مشاهده کرد، مانند دسترسی به شبکه و دمای CPU دستگاه ها. مقادیر حسگر را می توان در نمای دقیق سنسور MBP نیز مشاهده کرد. برای اطمینان از اینکه پردازش مدل پردازش جریان داده مستقر شده (dDSPM) تا زمانی که سیستم تهویه مطبوع به آن نیاز دارد صحیح می ماند، چندین قانون سطح بالا، مانند ، مدل‌های تشخیص اختلال، برای محیط راه‌اندازی اینترنت اشیا، به منظور شناسایی اختلالات ایجاد شده‌اند (مرحله ❺ در شکل ۲۰).

برای تشخیص اینکه آیا دستگاه‌ها بیش از حد گرم می‌شوند، یک مدل تشخیص اختلال در ابزار مدل‌سازی اختلال MBP ایجاد می‌شود که به طور مداوم بررسی می‌کند که آیا دمای CPU دستگاه‌ها نزدیک به حد دمای عملیاتی (۸۵ درجه سانتی‌گراد برای Raspberry Pi) است یا خیر. در صورت تشخیص اختلال، اعلان‌های مربوط به اختلال در داشبورد MBP نشان داده می‌شود.

۸-۳ ملاحضات بیشتر

در سال ۲۰۱۸، معماری مرجع OpenFog به عنوان یک استاندارد رسمی برای محاسبات ابری پذیرفته شد. این استاندارد که با نام IEEE ۱۹۳۴-۲۰۱۸ شناخته می‌شود، هشت اصل اصلی را ارائه می‌کند که پلتفرم‌های اینترنت اشیا به منظور برآورده کردن الزامات داده فشرده موجود در محیط‌های اینترنت اشیا باید پیاده‌سازی کنند.

این اصول عبارتند از: امنیت، مقیاس پذیری، باز بودن، استقلال، RAS (قابلیت اطمینان، در دسترس بودن، و قابلیت سرویس دهی)، چابکی، سلسله مراتب و قابلیت برنامه ریزی. در این بخش معماری MBP بر اساس این اصول مورد ارزیابی قرار می گیرد. هدف اصل امنیت دستیابی به تراکنش های ایمن و قابل اعتماد در محیط های IoT است.

برای دستیابی به این موضوع، یک رویکرد برای کشف، تأیید و تأیید اشیاء IoT قبل از ایجاد اعتماد مورد نیاز است. در MBP، اشیاء IoT از طریق مکانیسم‌های احراز هویت، مانند رمز عبور، اعتبار دیجیتال یا گواهی‌ها تأیید می‌شوند. علاوه بر این، تبادل داده بین اشیاء IoT و MBP توسط مکانیسم های امنیتی به خوبی تثبیت شده مانند HTTPS یا

OAuth۲ پشـــتیبانی می شـــود. علاوه بر این، MBP مفاهیم مدیریت و مالکیت کاربر را ارائه می دهد، به طوری که اشیاء IoT فقط برای صاحبان آنها یا سایر کاربران با مجوزهای دسترسی اعطا شده قابل دسترسی هستند.
هدف اصـل مقیاس پذیری بهره مندی از فرصـت های مقیاس پذیری ارائه شده در محیط های IoT است. این فرصت هـا در ابعـاد مختلفی مـاننـد عملکرد مقیاس پذیر، ظرفیت مقیاس پذیر و نرم افزار مقیاس پذیر وجود دارند. عملکرد مقیـاس‌پذیر به عملکرد مورد نیاز برنامه‌های اینترنت اشـــیا اشـــاره دارد، که باید با تأخیر کمی بین اندازه‌گیری‌های حسـگر و فرمان‌های محرک در نتیجه ارائه شـود، حتی زمانی که تعداد اشـیاء متصـل به پلتفرم IoT افزایش می‌یابد. برای این کـار، MBP از فنـاوری‌هـای CEP اســـتفـاده می‌کند، که قادر به پردازش مداوم مقادیر زیادی از داده‌ها و پردازش کارآمد و به موقع داده‌های حسگر ، به طوری که دستورات محرک را می‌توان در اسرع وقت صادر کرد.
ظرفیت مقیاس پذیر به توانایی اشـیاء IoT برای افزودن (یا حذف) به محیط های اینترنت اشـیا اشـاره دارد. MBP مکانیسـم‌هایی را برای به‌روزرسـانی محیط‌های IoT با افزودن، به‌روزرسـانی یا حذف اشیاء ارائه می‌کند، که هم شامل اشیاء سخت‌افزاری (مانند دستگاه‌ها، حسگرها، محرک‌ها) و هم اشیاء مجازی (مانند ماشین‌های مجازی) می‌شود.
در نهایت، نرم‌افزار مقیاس‌پذیر، در میان جنبه‌های دیگر، به مقیاس‌پذیری زیرسـاخت مدیریت یک پلت‌فرم IoT اشاره دارد، یعنی توانایی آن برای مدیریت بسـیاری از اشـیاء مطابق با سناریوهای IoT. با توجه به معماری ماژولار MBP و اســـتفاده از اجزای نرم افزاری مقیاس پذیر (مانند پایگاه های داده مقیاس) همانطور که در بخش ۸-۱ توضـــیح داده شده است، MBP همچنین می تواند این بعد مقیاس پذیری را ارائه دهد.
MBP به عنوان یک پروژه منبع باز GitHub در دسـترس اسـت و بنابراین، نه اختصـاصـی است و نه تک فروشنده. علاوه بر این، MBP بر اســـاس اســـتانداردهای ایجاد شـــده، مانند MQTT، TOSCA، XML، به منظور اطمینان از کاربرد طولانی مدت آن است. علاوه بر این، MBP هیچ محدودیتی برای اتصال اشیاء IoT به MBP ندارد.
از طریق مفاهیم اتصـــال پویا ، اپراتورهای هر نوع شـــی IoT را می توان ایجاد و در MBP ثبت کرد. MBP چندین عملگر آمـاده اســـتخراج و کنترل را در قـالـب اســـکریپـت ارائه می دهد، اما همچنین به کاربران امکان می دهد تا اپراتورهای خود را به هر زبان برنامه نویسی دلخواه ارائه دهند.
اصـل خودمختاری به اسـتقلال تصمیم گیری اشیاء اینترنت اشیا پس از وقوع خرابی اشاره دارد. MBP ابزاری را برای اســـتقرار اپراتورها بر روی اشـــیاء IoT فراهم می کند، به طوری که درجه ای از اســـتقلال اولیه اشـــیاء IoT از طریق اپراتورها فعال می شود. با این حال، پس از شکست های شناسایی شده در محیط های IoT متصل به MBP، تصمیم گیری مورد نیاز برای واکنش به خرابی ها عمدتا توســـط خود MBP انجام می شـــود. با این وجود، هنوز رویکردهای پیچیده تری برای دسـتیابی به اهداف این اصـل مورد نیاز اسـت و بنابراین، بخشی از کارهای آینده بر اساس مفاهیم این پایان نامه خواهد بود.
هدف اصـــل برنامه‌پذیری دســـتیابی به یک اســـتقرار بســـیار تطبیقی اســـت به طوری که دادن وظایف جدید (یعنی اپراتورها) به یک شـی IoT به صـورت خودکار انجام می‌شـود. این اصل در MBP از طریق مکان‌یابی اپراتور و مفاهیم استقرار پویا به دست می‌آید، که در آن یک اپراتور می‌تواند به‌طور خودکار به یک شی IoT دیگر مستقر شود.

هدف اصـل چابکی تبدیل حجم عظیمی از داده های سطح پایین به اطلاعات فشرده و سطح بالا است تا به کاربران در تجزیه و تحلیل این داده ها و تصـمیم گیری های تجاری کمک کند. علاوه بر این، به توانایی مقابله با محیط های پویا اینترنت اشیا از طریق واکنش سریع به تغییرات اشاره دارد.

MBP چندین رویکرد را برای کمک به کاربران برای شناسایی زودهنگام اختلالات در محیط های IoT و علاوه بر این، انجام اقدامات اصـلاحی در اسـرع وقت ارائه می دهد. مثال‌هایی برای چنین رویکردهایی ارائه اطلاعات در سطح بالا از طریق داشبورد، ابزار مدل‌سازی اختلال برای قوانین تعریف‌شده توسط کاربر، که واکنش خودکار مبتنی بر قانون را در تشخیص اختلال، و پردازش به موقع داده‌ها با استفاده از فناوری‌های CEP را امکان‌پذیر می‌سازد، است.

برای تحقق یک نظارت خودکار بر اسـاس داده‌های حسگر و فعال کردن محرک‌ها بر اساس اختلالات شناسایی شده، MBP ابزاری را برای تشخیص اختلالات بر اساس قوانین تعریف‌شده توسط کاربر فراهم می‌کند. چنین قوانینی، که بر اساس الگوی رویداد-شرط-عمل هستند، می‌توانند در ابزار مدل‌سازی اختلال MBP تعریف شوند.

هدف اصـل سـلسـله مراتب معماری های سـلسـله مراتبی متشـکل از چندین لایه است که در آن هر لایه به مشکلات خاصـی در سـناریوهای اینترنت اشیا می پردازد. همانطور که در شکل ۲۰ نشان داده شده است، MBP یک معماری لایه ای ارائه می کند، که در آن از لایه پایین تا لایه بالا، هر لایه با سـطح متفاوتی از انتزاع مطابقت دارد. برای مثال، تشـخیص‌دهنده اختلال داده‌های حسـگر سـطح پایین را پردازش می‌کند، در حالی که مدل‌سـاز و مدیر DSPM با منطق برنامه‌های سطح بالا سر و کار دارد.

در نهایت، اصـل RAS (قابلیت اطمینان، در دسـترس بودن، قابلیت سـرویس دهی) به سـه جنبه تقسـیم می شـود. قابلیت اطمینان به عنوان توانایی پلت فرم اینترنت اشیا برای ارائه عملکردهای طراحی شده حتی در شرایط نامطلوب، به عنوان مثال، زمانی که یک شـی اینترنت اشـیا معیوب می شـود، تعریف می شـود. در دسـترس بودن به مدیریت مستمر و هماهنگ سازی محیط اینترنت اشیا اشاره دارد.

قابلیت سـرویس دهی به عملکرد صـحیح برنامه های IoT در محیط IoT، به عنوان مثال، از طریق اسـتقرار و تعمیر خودکار اشاره دارد. در سراسر مشارکت های این پایان نامه (ر.ک. فصل های ۴ تا ۷)، نشان داده شده است که چگونه MBP به نگرانی های مربوط به قابلیت اطمینان، در دسترس بودن و سرویس دهی می پردازد.

برای افزایش قابلیت اطمینان، رویکردی برای شناسایی هرچه زودتر اختلالات در فصل ۷ ارائه شده است. علاوه بر این، برای افزایش در دسـترس بودن، MBP محیط‌های اینترنت اشیا و اپراتورهای مستقر را به طور مداوم مدیریت می‌کند (ر.ک. فصـل ۴) و نظارت می‌کند. در نهایت، فصـل‌های ۵ و ۶ قابلیت سـرویس‌دهی را از طریق چندین مفهوم برای استقرار و استقرار مجدد اپراتورها در محیط‌های اینترنت اشیا پشتیبانی می‌کنند.

فصل نهم

نتیجه گیری و کار آینده

مقدمه

در دهه های گذشته، چشم انداز اینترنت اشیا (IoT) بیش از پیش به واقعیت تبدیل شده است و پیشرفت‌ها در فناوری‌های سخت‌افزار و شبکه، وجود محیط‌های IoT حاوی دستگاه‌ها، حسگرها و محرک‌ها را امکان‌پذیر کرده است که می‌توانند برای تحقق برنامه‌های پیچیده متنوع، مانند شهرهای هوشمند، خانه‌های هوشمند، یا کارخانه‌های هوشمند مورد استفاده قرار گیرند.

در محیط های اینترنت اشیا، مقادیر زیادی از جریان های داده به طور مداوم تولید می شود که منجر به چالش های بزرگی در زمینه پردازش و مدیریت آنها می شود. در بسیاری از رویکردها، داده های اینترنت اشیا به زیرساخت های ابری منتقل می شوند، جایی که پردازش به صورت متمرکز اجرا می شود.

با این حال، بسته به برنامه در دست، ممکن است تاخیر و ترافیک شبکه افزایش یابد و علاوه بر این، تاخیر قبل و بعد از پردازش را ایجاد کند اما روش‌های دیگر، پردازش داده‌ها را در محیط IoT اجرا می‌کنند تا از تاخیر و ترافیک شبکه جلوگیری کنند و علاوه بر این، به پردازش به موقع جریان‌های داده دست یابند. برای این کار، قرار دادن اپراتورهایی برای پردازش قبل از ارسال ضروری است.

در دهه‌های گذشته، رویکردهای بسیاری برای مشکل مکان‌یابی اپراتور ایجاد شده اند. بسیاری از آنها مکان قرارگیری را بر اساس برآورده شدن الزامات QoS، مانند تأخیر و پهنای باند، تعیین می کنند. با این حال، در حوزه اینترنت اشیا، جنبه‌های بیشتری ظاهر شده است که علاوه بر این باید موضوع قرارگیری اپراتور در محیط‌های اینترنت اشیا، مانند ناهمگونی گره‌های پردازش، در نظر گرفته شوند.

رویکردهای فعلی در قرار دادن اپراتور برای محیط های IoT ، امکان پشتیبانی از الزامات خاص معرفی شده توسط دامنه IoT را ندارند. همچنین معمولاً آنها الزامات غیر کاربردی و تعریف شده توسط کاربر برای برنامه های IoT مانند حریم خصوصی داده ها و ناشناس سازی، امنیت و قابلیت همکاری را در نظر نمی گیرند. بنابراین، در این پایان نامه دکتری، رویکردی برای قرار دادن اپراتورهای پردازش جریان داده در محیط های اینترنت اشیا ارائه شد که ویژگی های حوزه اینترنت اشیا و همچنین الزامات غیر کاربردی و تعریف شده توسط کاربر را در طول تصمیم گیری برای قرار دادن اپراتور در نظر می گیرد.

برای اینکار، یک محیط IoT و قابلیت های پردازش آن توسط یک مدل محیط اینترنت اشیا (IoTEM) توصیف شده است ومنطق تجاری یک برنامه IoT و الزامات آن توسط یک مدل پردازش جریان داده (DSPM) تعریف می شود. سپس این مدل‌های اطلاعاتی برای فعال کردن تصمیم‌گیری برای قرار دادن اپراتور برای محیط‌های IoT به کار می‌روند، یعنی تصمیم می‌گیرند که اپراتورهای پردازشی باید بر اساس تطابق الزامات و قابلیت‌ها در محیط‌های IoT قرار گیرند. از طریق رویکرد این پایان نامه دکتری، پردازش داده های برنامه های کاربردی اینترنت اشیا را می توان برای موارد استفاده خاص، پشتیبانی از الزامات خاص دامنه اینترنت اشیا و علاوه بر آن، کاربران برنامه های اینترنت اشیا تنظیم کرد.

۹-۱ نتیجه گیری

این پایان نامه دکترا چهار فناوری (C) ارائه می کند که به سؤالات تحقیق می پردازد، و علاوه بر این، مفاهیمی را برای دستیابی به اهداف شرح داده شده در بخش ۲-۱ ارائه می دهد:

C۱: مدل سازی محیط اینترنت اشیا و پردازش جریان داده

C۲: نگاشت DSPM ها بر روی IoTEM

C۳: استقرار اپراتورها در محیط های IoT

C۴: نظارت بر DSPM مستقر شده.

در بخش C۱ (ر.ک. فصل ۴)، دستیابی به اهداف G۲ و G۳ با ارائه IoTEM برای مدل‌سازی محیط‌های IoT و DSPM برای مدل‌سازی منطق تجاری برنامه‌های IoT حاصل شد.

این فناوری، یک مدل‌سازی کاربرپسند و آسان از محیط‌های اینترنت اشیا را از طریق یک ابزار مدل‌سازی گرافیکی امکان‌پذیر می‌سازد که در آن می‌توان اشیاء ناهمگن اینترنت اشیا، اتصالات آنها و علاوه بر آن، قابلیت‌های اشیاء و اتصالات اینترنت اشیا را مدل‌سازی کرد.

از سوی دیگر، مدل‌سازی پردازش جریان داده‌ای که نیازهای اینترنت اشیا را پشتیبانی می‌کند، از طریق ابزار مدل‌سازی گرافیکی دیگری به دست آمد، که در آن منطق پردازش برنامه‌های اینترنت اشیا برای موارد استفاده خاص دامنه، از جمله اینترنت اشیا و نیازهای کاربر، مدل‌سازی می‌شود.

علاوه بر این، بخش C۲ (ر.ک. فصل ۵) با امکان قرار دادن مبتنی بر الزامات اپراتورهای پردازش در محیط های IoT به هدف G۴ دست می یابد. برای این کار، این فناوری چندین الگوریتم را برای انجام نگاشت مدل‌های پردازش جریان داده (DSPM) بر روی مدل‌های محیط اینترنت اشیا (IoTEMs) ارائه می‌کند، با در نظر گرفتن الزامات اپراتورهای پردازشی که باید توسط قابلیت‌های اشیاء IoT برآورده شوند.

در بخش C۳ (ر.ک. فصل ۶)، هدف G۵ با ارائه چندین رویکرد استقرار بر اساس استاندارد TOSCA به دست آمد. این رویکردها قادر به مقابله با ماهیت ناهمگون و پویا محیط های اینترنت اشیا هستند و از این طریق به استقرار کارآمد اپراتورها بر روی اشیاء اینترنت اشیا دست می یابند.

علاوه بر این، بخش C۴ (ر.ک. فصل ۷) با نظارت مستمر محیط‌های اینترنت اشیا و اپراتورهای مستقر با استفاده از تکنیک‌های CEP به‌خوبی تثبیت‌شده، به هدف G۶ دست می‌یابد، به طوری که اختلالات در اسرع وقت شناسایی می‌شوند. به طور خلاصه، همه اهداف تحت پوشش مشارکت قرار می گیرند. فناوری این پایان نامه، پلت فرم IoT Binding and Provisioning Multipurpose (MBP) به عنوان یک نمونه اولیه توسعه داده شده است.

MBP به عنوان یک پروژه منبع باز در چندین کار دانشجویی تحت نظارت در محدوده این پایان نامه دکتری توسعه یافته است. علاوه بر این، مفاهیم MBP به عنوان یک پلت فرم IoT به عنوان یک مقاله نمایشی منتشر شده است و بنابراین برای امکان‌سنجی تایید شده است.

در فصل ۸، مروری بر عملکردهای اصلی MBP ارائه شد و معماری آن در برابر معماری مرجع OpenFog ارائه شده توسط استاندارد IEEE۱۹۳۴-۲۰۱۸ برای محاسبات ابری ارزیابی شد. در شکل ۲۰، معماری کامل به دست آمده از جمله اجزای معماری منفرد هر فناوری، به تصویر کشیده شده است. فناوری ها با رنگ برجسته می‌شوند و مراحل رویکرد روشمند به کارگیری این معماری نیز نشان داده شده‌اند.

مولفه‌های معماری هر فناوری به تفصیل در فصل‌های ۴ تا ۷ توضیح داده شده است. رویکرد روش‌مند از شش مرحله اصلی تشکیل شده است:

۱- ایجاد مدل محیط اینترنت اشیا (IoTEM)،
۲- ایجاد مدل پردازش جریان داده (DSPM).
۳- تطبیق اپراتورهای پردازش و اشیاء اینترنت اشیا،
٤- استقرار اپراتورهای پردازش بر روی اشیاء اینترنت اشیا،
٥- تشخیص اختلالات موثر بر پردازش داده ها،
٦- توقف و انفصال پردازش داده ها.

در رویکرد روشمند، دو نقش اصلی تعریف شده است، متخصص دامنه که مرحله 1 و بخشی از مرحله 5 را انجام می دهد و تحلیلگر دامنه که مرحله 2 و بخشی از مرحله 6 را انجام می دهد.

متخصصان دامنه دانش فنی در مورد اشیاء سخت افزاری (یعنی دستگاه ها، حسگرها، محرک ها)، اشیاء مجازی (یعنی ماشین های مجازی) و اتصالات شبکه آنها در یک محیط IoT دارند. علاوه بر این، متخصصان دامنه از نحوه دسترسی به این اشیاء برای استخراج داده‌های حسگر یا ارسال دستورات کنترلی به یک محرک تسلط دارند.

بنابراین، در مرحله 1، وظیفه اصلی متخصصان دامنه، ایجاد IoTEM ها است که گراف های هدایت شده حاوی اشیاء اینترنت اشیا به عنوان گره و اتصالات شبکه آنها به عنوان لبه هستند. تحلیلگران دامنه دانش دامنه در مورد پردازش داده های تولید شده در محیط اینترنت اشیا دارند، به عنوان مثال، آنها دانش لازم را برای مدل سازی برنامه های کاربردی اینترنت اشیا مختلف برای موارد استفاده خاص دامنه دارند.

در مرحله 2، وظیفه اصلی تحلیلگران دامنه ایجاد DSPM هایی است که منطق پردازش برنامه های کاربردی اینترنت اشیا را نشان می دهد. علاوه بر این، آنها می توانند پردازش جریان داده یک برنامه IoT را در مرحله 6 متوقف و منفصل کنند. فناوری ها در مجموع دستیابی به هدف G۱ را امکان‌پذیر می‌کنند، یعنی پردازش به موقع و کارآمد داده‌های اینترنت اشیا را در محیط‌های اینترنت اشیا امکان‌پذیر می‌کنند.

این فناوری ها بر اســاس اســتانداردهای ایجاد شــده، به عنوان مثال، MQTT، TOSCA، XML، به منظور اطمینان از کاربرد طولانی مدت آنها، و علاوه بر این، آنها در چندین مجلات، کنفرانس های ملی و بین المللی منتشر شده اند. مشارکت‌ها بـا ادغـام آن‌هـا در پروژه‌های تحقیقاتی مختلف، مورد ارزیابی قرار گرفتند، که از طریق آن می‌توان کاربرد آنها را تأیید کرد. این پروژه ها شامل پروژه های تحقیقاتی SitOPT - SmartOrchestra – IC۴.

هدف پروژه SitOPT توسـعه مفاهیم و روش‌هایی اسـت که به برنامه‌های مبتنی بر موقعیت اجازه می‌دهد تا بتوانند به طور مسـتقل با محیط پویایی که در آن اجرا می‌شـوند سـازگار شـوند. پروژه SmartOrchestra با هدف توسعه یک پلتفرم باز برای ترکیب ایمن و هماهنگ‌سـازی مبتنی بر TOSCA خدمات هوشـمند برای برنامه‌های فیزیکی سایبری و علاوه بر این، برای بازاریابی مؤثر چنین خدمات هوشــمندی انجام شــد. پروژه IC۴F با هدف توســعه راه حل های ارتباطی ایمن، قوی و بلادرنگ برای صـنعت تولید انجام شد. در نهایت، نرم افزار توسعه یافته به عنوان بخشی از این پایان نامه به عنوان پروژه های منبع باز در GitHub در دسترس قرار گرفته است.

۹-۲ کار آینده

در حـال حـاضــر، زمـانی که اختلالات در محیط‌های IoT شــناســایی می‌شــوند که نیاز به یک اقدام اصــلاحی دارند، تصـمیم‌گیری‌های جدید اپراتور باید مجدداً محاسـبه شوند، که توسط MBP به صورت مرکزی تحقق می‌یابد. در کار آینده، این رویکرد را می توان برای توزیع منطق تصـمیم گیری در محل اپراتور در نمونه های مختلف، از جمله اشـیاء اینترنت اشیا، به منظور جلوگیری از یک نقطه شکست (SPOF)، و علاوه بر این، افزایش استقلال اشیاء پس از وقوع خرابی، گسترش داد. بنـابراین، اشــیـاء IoT و اپراتورهای مســتقر شــده باید با یک مفهوم خودمختاری، همانطور که توســط معماری مرجع OpenFog برای محاسبات ابری (استاندارد IEEE۱۹۳۴-۲۰۱۸) توصیه می‌شود، توسعه داده شوند.

به این ترتیب، اشـیاء IoT و اپراتورهای مسـتقر شـده قادر خواهند بود به طور مستقل اختلالات را تشخیص دهند و تا حدی یک مکان اپراتور جدید را برای خود محاسبه کنند. علاوه بر این، مفاهیم این پایان نامه را می توان برای فعال کردن تجزیه و تحلیل پیش بینی برای داده های اینترنت اشـیا گسـترش داد، به عنوان مثال، پیش بینی کرد که دسـتگاه ها، حسـگرها یا محرک های اینترنت اشـیا ممکن اسـت معیوب شـوند، نیاز به تعمیر و نگهداری داشـته باشند، یا سخت افزار نیاز به کالیبره شـدن جدید داشـته باشـد. با فعال کردن چنین پیش‌بینی‌هایی، قابلیت اطمینان محیط‌های اینترنت اشیا را می‌توان افزایش داد و در نتیجـه، اجرای برنامه‌های اینترنت اشــیا در چنین محیط‌هایی می‌تواند اســتحکام و کیفیت نتایج پردازش آن‌ها را بسیار بهبود بخشد.

کار مقدماتی قبلاً در محدوده این پایان نامه توسـط نظرسنجی دانشجویی باچارو و همکاران انجام شده است. در نهایت، یکی از جنبه‌های مهم و به‌روز پلتفرم‌های اینترنت اشـیا و اشـیاء آن این اسـت که چگونه می‌توان اطمینان حاصل کرد که حداقل امنیت لازم و حفظ حریم خصـوصـی داده‌ها را فراهم می‌کنند. به دلیل ماهیت بسـیار بهم پیوسـته اینترنت اشـیا، اجزای آن مسـتعد حملاتی هسـتند که نه تنها مجزا هسـتند، بلکه می توانند در کل محیط های اینترنت اشـیا منتشـر شوند. بنابراین، مفاهیم این پایان نامه باید با اقدامات حفظ حریم خصـوصی و امنیتی بیشتر گسترش یابد. کار مقدماتی قبلاً در محدوده این پایان نامه توسط نظرسنجی دانشجویی از Glaub و همکاران انجام شده است.

منابع و مراجع

فهرست منابع غیرفارسی و اینترنتی

١) A. Botta, W. de Donato, V. Persico, A. Pescapé. “Integration of Cloud computing and Internet of Things: A survey.”

٢) A. Buchmann, B. Koldehofe. “Complex Event Processing.”

٣) A. C. Franco da Silva, P. Hirmer, B. Mitschang. “Model-based Operator Placement for Data Processing in IoT Environments.”

٤) Asghari, Parvaneh, Rahmani, Amir Masoud, Javadi, Hamid Haj Seyyed. “Internet of Things applications: A systematic review.”

٥) B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom. “Models and Issues in Data Stream Systems.”

٦) D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik. “Aurora: a new model and architecture for data stream management.”.

٧) H. Derhamy, J. Eliasson, J. Delsing, P. Priller. “A survey of commercial frameworks for the internet of things.”

٨) K. Hur, S. Chun, X. Jin, K.-H. Lee. “Towards a Semantic Model for Automated Deployment of IoT Services Across Platforms.”

٩) L. Atzori, A. Iera, G. Morabito. “The internet of things: A survey.”

١٠) M. Blackstock, R. Lea. “Toward a distributed data flow platform for the web of things (distributed node-red).”

١١) O. Vermesan, P. Friess, P. Guillemin, H. Sundmaeker, M. Eisenhauer, K. Moessner, F. Le Gall, P. Cousin. “Internet of Things Strategic Research and Innovation Agenda.”

١٢) T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, S. Wagner. “OpenTOSCA – A Runtime for TOSCA-based Cloud Applications.”

١٣) V. K. C. Bumgardner, C. Hickey, V. W. Marek. “An Edge-Focused Model for Distributed Streaming Data Applications.

ABSTRACT

Recent advances in various fields, including sensor technologies, networking, and data processing, have made the IoT vision more and more a reality. As a result of these advances, today's IoT enables the development of complex applications for IoT environments, such as smart cities, smart homes, or smart factories, and transforms them into data streams due to the frequent exchange of data.

With this increasing volume of data being processed continuously, there are several challenges such as preventing interference and damage in transient processes that require the study of data-based processing in IoT environments, and on the other hand with We are faced with a heterogeneous distributed network consisting of a variety of hardware and sensors that must be able to process their information, the best environment for processing is the cloud environment.

However, despite the high volume of information and lengthy processing, not all information can be processed in a centralized cloud space, and it is appropriate to perform these processes near the information production site (processing node) and provide a model for implementing this structure as a data stream and system. Heterogeneous and distributed is the main purpose of this doctoral dissertation.

Keywords

Internet of Things, data, data stream, cloud computing, cloud server.

Payam Noor University

Department of Computer Engineering and Information Technology

Seminar Report (M.Sc)

Title:

A model-based approach for data processing in IoT environments

Supervisor:

Dr. Ali Razavi

By:

Majid Lotfi

September ۲۰۲۰