



دانشکده فنی و مهندسی
گروه مهندسی کامپیوتر و فناوری اطلاعات

گزارش سمینار کارشناسی ارشد رشته مهندسی کامپیوتر نرم افزار (M.Sc)

عنوان سمینار:

رویکرد مبتنی بر مدل برای پردازش داده ها در

بستر اینترنت اشیا IoT

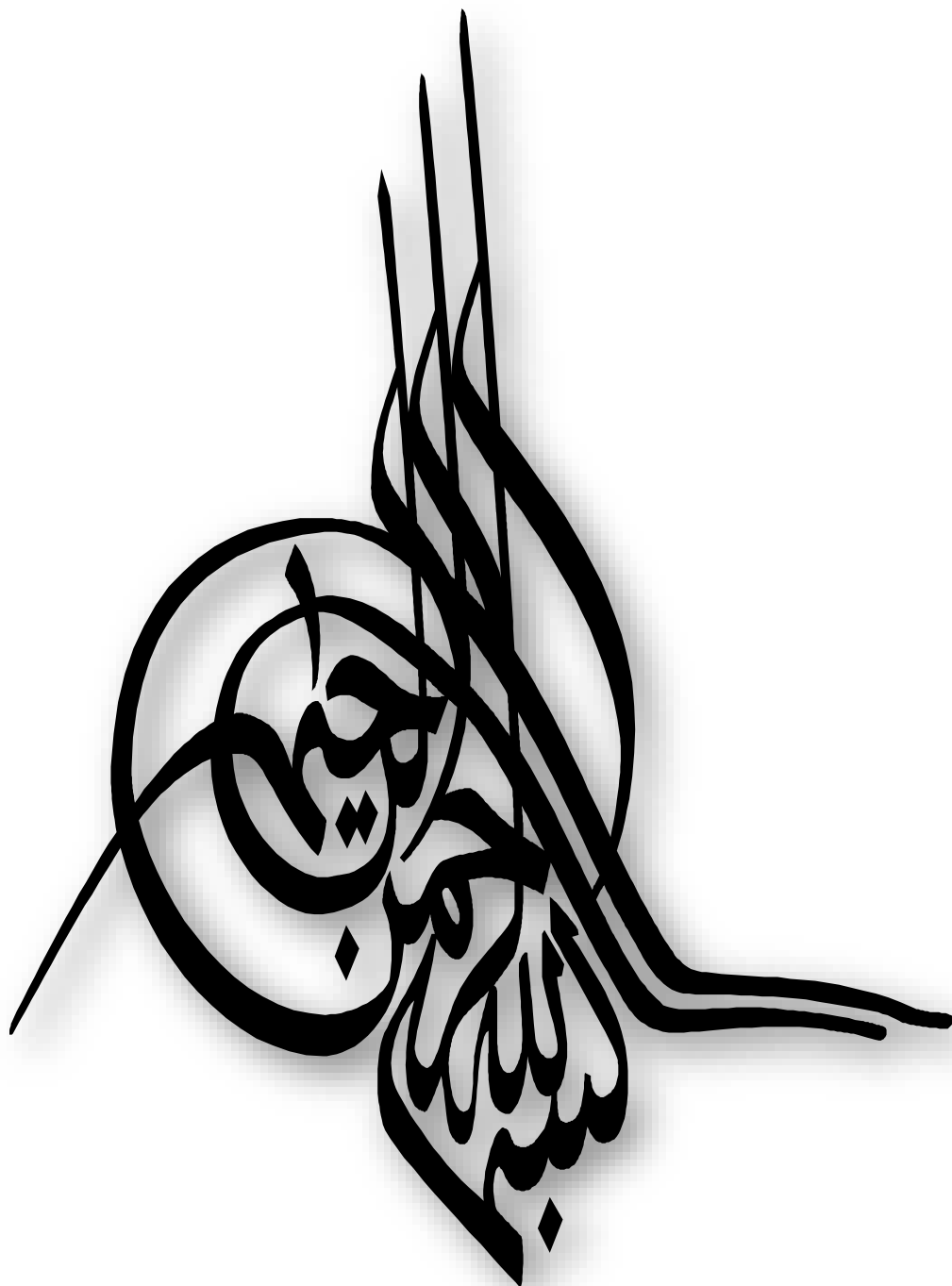
استاد راهنما:

دکتر سیدعلی رضوی

نگارنده:

مجید لطفی

پاییز ۱۴۰۰



فهرست مطالب

۱ . مقدمه	۵
۱-۱ تعریف مسئله و بیان سؤال‌های اصلی تحقیق	۶
۲-۱ ضرورت تحقیق	۶
۳-۱ هدف‌ها	۶
۴-۱ فرضیات تحقیق	۷
۵-۱ چه کاربردهایی از انجام این تحقیق متصور است؟	۷
۶-۱ روش و مراحل انجام تحقیق	۷
۷-۱ سازمان پایان‌نامه مورد بررسی	۸
۸-۱ ساختار گزارش تحقیق	۸
۲ . ادبیات تحقیق و پیش زمینه	۹
۱-۲ تعاریف پس زمینه	۹
۱-۱-۲ اینترنت اشیاء	۹
۲-۱-۲ پردازش جریان داده و داده پیچیده	۹
۳-۱-۲ مشکل قراردادن اپراتورها	۱۰
۴-۱-۲ تکنولوژی TOSCA	۱۱
۵-۱-۲ تعیین IoTEM	۱۲
۶-۱-۲ قابلیت‌های اشیاء و ارتباطات	۱۲
۷-۱-۲ تعریف DSPM	۱۵
۸-۱-۲ اپراتورهای پردازش	۱۷
۹-۱-۲ زبان توضیح برای اینترنت اشیاء (Topic Description Language)	۱۹
۱۰-۱-۲ MBP	۲۲
۳ . پیشینه تحقیق	۲۴
۱-۳ مقدمه	۲۴
۲-۳ پیشینه مربوط به سال‌های اخیر	۲۵
۱-۲-۳ مدل‌های اینترنت اشیاء	۲۵
۲-۲-۳ اشیاء در محیط اینترنت اشیاء	۲۶
۳-۲-۳ اپراتورها در محیط اینترنت اشیاء	۲۷
۴-۲-۳ نظارت بر عملیات‌ها	۲۸
۴ . تجزیه و تحلیل موضوع انتخابی	۳۱
۱-۴ مقدمه	۳۱
۲-۴ مدل محیط اینترنت اشیاء	۳۲

۳-۴ معماری کلی	۳۴
۳-۴-۱ مولفه های معماری و پیاده سازی بخش مدل ساز و مدیر IoTEm	۳۶
۴-۴ مدل سازی پردازش جریان داده	۳۷
۴-۴-۱ ساختار معماری و پیاده سازی - مدل ساز و مدیر DSPM	۳۷
۴-۵ ارتباط بین DSPM ها در مدل های اینترنت اشياء (IoTEm)	۳۹
۴-۵-۱ رویکرد خودکار	۳۹
۴-۵-۲ رویکرد دستی	۴۳
۴-۵-۳ جزئیات معماری و پیاده سازی IoTEm و DSPM mapper	۴۷
۴-۶ استقرار اپراتورها در محیط های IoT	۴۸
۴-۶-۱ رویکرد خودکار	۴۸
۴-۶-۲ استقرار اپراتور مبتنی بر TOSCA	۵۰
۴-۶-۳ رویکرد استقرار نیمه اتوماتیک	۵۱
۴-۷ جزئیات معماری و پیاده سازی مدیر استقرار	۵۲
۴-۸ نظارت بر DSPM های مستقر شده	۵۳
۴-۸-۱ مدل سازی تشخیص اختلال	۵۳
۴-۸-۲ اجرای تشخیص اختلال	۵۷
۴-۹ جزئیات معماری و پیاده سازی تشخیص اختلال	۶۴
۴-۱۰ ارزیابی	۶۶
۴-۱۱ ملاحظات بیشتر	۷۰
۴-۱۲ پاسخ به سوالات تحقیق	۷۳
۵. جمع بندی و پیشنهادها	۷۵
۵-۱ مقدمه	۷۵
۵-۲ نتایج حاصل از تحقیق	۷۵
۵-۳ بررسی معایب پایان نامه مورد بررسی و بیان پیشنهاد	۷۶
۵-۴ ارائه ایده برای پایان نامه های جدید تکمیلی	۷۷
۵-۵ جمع بندی و نتیجه گیری	۷۷
مراجع	۷۹
ABSTRACT	۸۰

فهرست اشکال

۱۱.....	شکل شماره ۱.....
۱۷.....	شکل شماره ۲.....
۲۱.....	شکل شماره ۳.....
۳۵.....	شکل شماره ۴.....
۳۶.....	شکل شماره ۵.....
۳۸.....	شکل شماره ۶.....
۴۱.....	شکل شماره ۷.....
۴۴.....	شکل شماره ۸.....
۴۷.....	شکل شماره ۹.....
۴۹.....	شکل شماره ۱۰.....
۵۰.....	شکل شماره ۱۱.....
۵۵.....	شکل شماره ۱۲.....
۵۸.....	شکل شماره ۱۳.....
۵۹.....	شکل شماره ۱۴.....
۶۱.....	شکل شماره ۱۵.....
۶۵.....	شکل شماره ۱۶.....
۶۶.....	شکل شماره ۱۷.....
۶۸.....	شکل شماره ۱۸.....

چکیده

پیشرفت‌های اخیر در زمینه‌های مختلف، از جمله فناوری‌های حسگر، شبکه و پردازش داده‌ها، باعث شده اند چشم‌انداز اینترنت اشیا (IoT) هر روز بیشتر و بیشتر به واقعیت تبدیل شود. در نتیجه این پیشرفت‌ها، 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 شود، چالش های متعددی چون جلوگیری از تداخل و آسیب 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 هستیم متشکل از انواع سخت افزار ها و سنسور 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 اشیا^۱، داده^۲، جریان داده^۳، پردازش ابری^۴، سرویس دهنده ابری^۵

^۱ IoT (Internet Of Thing)

^۲ Data

^۳ Data stream

^۴ Cloud Processing

^۵ Cloud server

فصل اول

۱. مقدمه

پیشرفت‌های اخیر در زمینه‌های مختلف، از جمله فناوری‌های حسگر، شبکه و پردازش داده‌ها، چشم‌انداز اینترنت اشیا (IoT) را قادر ساخته است که هر روز بیشتر و بیشتر 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 شود، چالش‌های متعددی برای پردازش کارآمد داده‌های اینترنت اشیا وجود دارد. به عنوان مثال، چگونه می‌توان پردازش داده‌های اینترنت اشیا را بدون تأثیر بر واکنش پذیری برنامه‌های آن تحقق بخشید. علاوه بر این، چگونه می‌توان از طریق پردازش داده‌های IoT، نیازمندی‌های مختلف عملکردی، غیرعملکردی و تعریف‌شده توسط کاربر برنامه‌ها را برآورده کرد. در این پایان نامه دکتری، یک رویکرد کلی جدید برای پردازش برنامه‌های کاربردی مبتنی بر جریان داده در محیط های IoT ارائه شده است که تمرکز آن بر قرار دادن کارآمد اپراتورهای برنامه‌های کاربردی در محیط‌های ناهمگن، توزیع شده و پویا است.

این پایان نامه دکتری توسط مدل‌های اطلاعاتی مختلف و تکنیک‌های قرار دادن اپراتور پشتیبانی می‌شود، به طوری که کل چرخه حیات محیط‌های اینترنت اشیا و برنامه‌های مبتنی بر جریان داده را می‌توان 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 زیرساخت‌های ابری تنها در مواردی استفاده می‌شوند که محیط‌های اینترنت اشیا منابع پردازش کافی را برای کاربرد اینترنت اشیا ارائه نمی‌دهند.

از طریق رویکرد این پایان نامه دکتری، پردازش داده‌های برنامه‌های کاربردی اینترنت اشیا را می‌توان برای موارد استفاده خاص، پشتیبانی از نیازهای خاص دامنه‌ها، و علاوه بر این، کاربران برنامه‌های اینترنت اشیا، تنظیم کرد. پس از تعیین مکان‌های امکان‌پذیر، اپراتورهای پردازش با استفاده از استانداردهایی مانند TOSCA بر روی اشیاء IoT مربوطه مستقر می‌شوند و برنامه اینترنت اشیا آماده و اجرا می‌شود. در نهایت، محیط اینترنت اشیا به منظور شناسایی و واکنش به اختلالات مؤثر بر پردازش داده‌های برنامه‌های کاربردی اینترنت اشیا مستقر شده، به طور مداوم نظارت می‌شود.

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

۱-۱ تعریف مسئله و بیان سؤال‌های اصلی تحقیق

چنانچه در مقدمه بیان شد موضوع تبادل اطلاعات در محیط IoT نیازمند جریان داده و محیط ابری می باشد و پردازش جریان داده نیز چالش خاص را داشته و نیازمند سیستم های توزیع شده می باشد که ما در اینجا با توجه به تنوع دستگاه ها، با یک سیستم ناهمگن روبرو هستیم. در این نوع سیستم ها با توجه به حجم بالای اطلاعات بهترین راهکار برای پردازش اطلاعات، قراردادن پردازش در اپراتورهای پردازش (گره های پردازشی) نزدیک به محل ایجاد اطلاعات و ارسال پردازش نهایی به سیستم ابری است. لذا سؤالاتی که تا پایان این تحقیق پاسخ داده می‌شود را این‌گونه بیان می‌کنیم:

۱. چگونه باید پردازش مناسب برای انواع جریان داده در محیط IoT 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 در محیط IoT را طراحی، اجرا و کارائی این مدل را تضمین کرد؟

۲-۱ ضرورت تحقیق

ما در اینترنت اشیا با تنوع اشیا و داده های آنها مواجه هستیم که در یک جریان داده در حال اجرا بوده و هر دستگاه و شیء در این شبکه نیازمند پردازش هایی می باشد که باید پردازش داده ها به موقع و کارآمد باشد . این موضوع نیازمند بررسی تمام جوانب و ایجاد مدل مناسب می باشد که بتواند الزامات **IoT** را پشتیبانی کرده و مدلی مناسب و آسان برای کاربران باشد..

۳-۱ هدف‌ها

- با توجه به موضوعات مطرح شده در مقدمه و ضروریات تحقیق می توان این اهداف را متصور بود:
- الف) پردازش به موقع و کارآمد داده ها در محیط های اینترنت اشیا.
 - ب) مدل سازی مناسب و آسان برای کاربران در محیط های اینترنت اشیا.
 - ج) مدل سازی پردازش جریان داده و تشخیص اختلالات که الزامات اینترنت اشیا را پشتیبانی کند.
 - د) قرار دادن کارآمد اپراتورهای پردازش (خودکار و دستی) بر اساس نیازها در محیط های پویا و ناهمگن IoT

۱-۴ فرضیات تحقیق

- ۱) فرض میکنیم که کاربران با اهمیت و ساختار اینترنت اشیا آشنایی کافی داشته و پیش نیازهای مرتبط با ایجاد یک سامانه توزیع شده از دستگاه ها و سیستم های رایانه ای را به خوبی درک نمایند. برای مثال سامانه دسترسی از راه دور کنترل و پردازش دوربین ها و ماشین آلات یک کارخانه و یا سامانه مدیریت بخش های یک خانه مانند دوربین ها ، درب ها ، کلید های الکتریکی و ...
- ۲) فرض میکنیم که دستگاه ها یا اصطلاحا اشیاء موجود در یک شبکه اینترنت اشیا ، دارای قابلیت ارسال و دریافت اطلاعات جهت کنترل بوده و با تنوع انواع داده ها و اشیاء مواجه هستیم مثلا خانه هوشمند که دارای بخش های مختلف شامل کنترل درب ها ، پنجره ها ، روشنایی ، انشعابات اصلی آب ، برق و گاز بوده و تمامی این موارد بصورت دسترسی از دور قابل کنترل باشد .
- ۳) فرض میکنیم که اشیاء موجود در یک سامانه IoT علاوه بر امکان دسترسی ، نیازمند کنترل و بررسی داخلی بوده و اتفاقات احتمالی یا پیش بینی نشده مانند از کارافتادن و تعویض، باعث ایست کامل سامانه نشود ..

۱-۵ چه کاربردهایی از انجام این تحقیق متصور است؟

- طراحی محیط های گسترده و پیچیده اینترنت اشیا با قابلیت کارایی و سرعت بهینه با بررسی مدل ها و نرم افزارهای موجود به سمت مدلی بهینه تر و پویا تر
- طراحی و اجرای سامانه های کنترلی در اینترنت اشیا جهت ایجاد یک سیستم توزیع شده از منابع تولید و پردازش داده بصورت بهینه نسبت به ذخیره سازی داده ها و اجرای عملیات ها
- طراحی نرم افزاری سامانه های گسترده مانند خانه هوشمند و کارخانه هوشمند بر اساس یک مدل پویا و دارای خطاهای کمتر و بهینه سازی شده

۱-۶ روش و مراحل انجام تحقیق

روش انجام این تحقیق به صورت کتابخانه ای است. منابع مورد استفاده شامل پایان نامه، مقالات، تحقیقات علمی و پژوهشی، کتب و جستجوهای اینترنتی در زمینه ی متدولوژی های مدیریت اطلاعات برنامه های حساس از دیدگاه مهندسی نرم افزار است.

در این راستا یک پایان نامه انتخاب شد (Franco da Silva, 2020) و با بررسی ساختار پایان نامه و منابع مرجع، توانستم موضوع درک و تجزیه و تحلیل و بیان کنم.

۷-۱ سازمان پایان نامه مورد بررسی

فصل های پایان نامه مورد بررسی به صورت ذیل مرتب شده است:

فصل ۱ مقدمه و ضرورت تحقیق

فصل ۲ پیشینه مربوط به این تحقیق و مشکلات و نواقص سایر تحقیقات را مورد بحث قرار می دهد.

فصل ۳ دستاوردها ، رویکردها و موارد فنی را توضیح می دهد.

فصل ۴ و فصل ۵ رویکرد و مدل های اجرایی را توضیح می دهد.

فصل ۶ و فصل ۷ و فصل ۸ نحوه اجرا و مدیریت و ارزیابی مدل معرفی شده را بیان می کند.

فصل ۹ نتیجه گیری و کارهای آینده را بیان می کند.

۸-۱ ساختار گزارش تحقیق

فصل اول به تعریف و مقدمه و دلایل نیاز به طرح ارائه شده پرداخته می شود.

فصل دوم به پیش زمینه و کارهای وابسته پرداخته می شود.

فصل سوم مروری است بر کارهای انجام شده طرح پیشنهادی پایان نامه

فصل چهارم به کاربردها و مزایا و معایب روش های مطرح شده پرداخته می شود.

فصل پنجم نیز به جمع بندی و نتیجه گیری پرداخته می شود.

فصل دوم

۲. ادبیات تحقیق و پیش زمینه

در این فصل، ابتدا تعاریف و پیشینه فنی مورد نیاز برای فهم را معرفی می کنیم. سپس درباره ی فعالیتهای مربوطه بحث می کنیم.

۲-۱ تعاریف پس زمینه

در ادامه تعاریف مربوط و فناوریهای اصلی که به رویکردها قدرت میدهند را شرح میدهم.

۲-۱-۱ اینترنت اشیا

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 ابتداءاً تقویت راینه‌هایی 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 اشیا 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.

۲-۱-۲ 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 است، پرس وجوهای پیوسته را بر روی جریانهای داده ورودی اجرا می کند و سیستم 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 شوند و پاسخ های کامل را برمی گردانند، در حالی که DSMS ها پرس و جوها را به طور مداوم اجرا می کنند و پس از رسیدن داده های جدید، پاسخ های به روز ارائه می دهند.

به طور معمول، DSMS جریانهای داده را از طریق دنباله ای از تبدیل های مبتنی بر عملگرهای SQL، مانند انتخاب، تجمیع، یا پیوستن، که توسط جبر رابطه ای تعریف می شود، پردازش می کند. از سوی دیگر، پردازش رویدادهای پیچیده (CEP) شامل مجموعه ای از اصول و تکنیک ها برای تجزیه و تحلیل مجموعه هایی از رویدادها است که تا حدی بر اساس زمان با رسیدن این رویدادها مرتب شده اند. یعنی CEP ابزاری را برای پردازش مجموعه ای از رویدادهای مرتبط به هم به صورت پیوسته و به موقع فراهم می کند.

رویدادها معمولاً برای پردازش در الگوها ارائه می شوند، با این حال، می توانند با رویدادهای نامرتب دیگر مخلوط شوند. ویژگی مهم CEP توانایی تشخیص الگوها (به عنوان مثال، روابط) بین رویدادها است 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.

۲-۱-۳ مشکل قراردادن اپراتورها

پردازش جریان های داده می تولند از طریق یک سیستم 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.

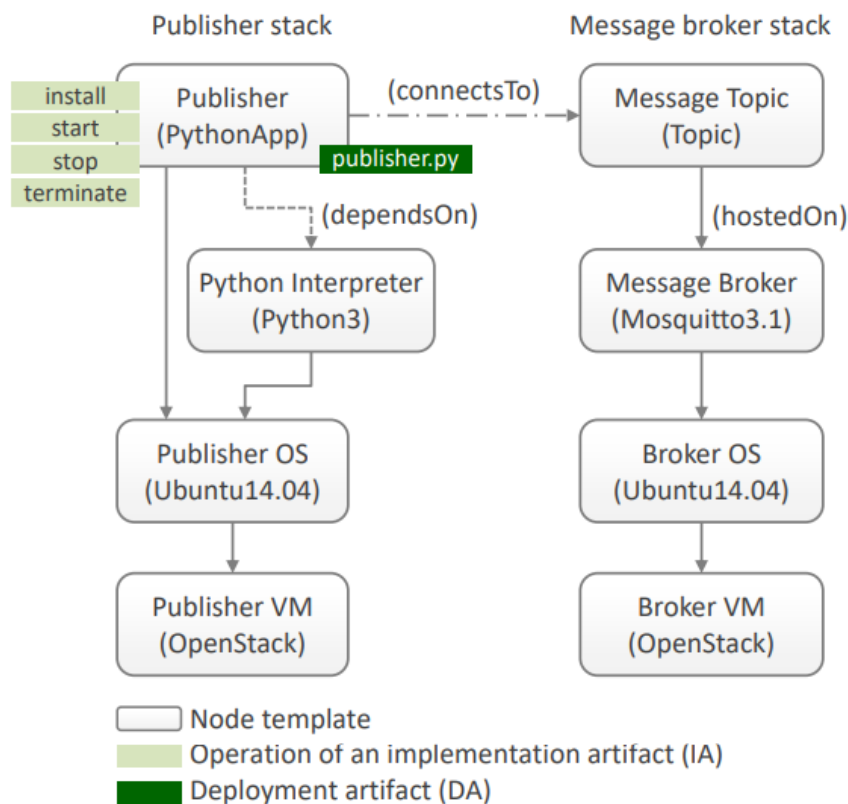
هدف این مشکل یافتن یک مکان بهینه از کل پرس و جوهای پیوسته یا اپراتورهای منفرد در مجموعه ای از گره های پردازشی مختلف است که در سراسر یک شبکه توزیع شده اند. مکان بهینه معمولاً بر اساس توابع هزینه تعریف شده توسط سیستم یا تعریف شده توسط کاربر محاسبه می شود، که هدف آن ارائه، برای مثال، عملکرد بالاتر یا توزیع بار بهتر است.

۲-۱-۴ تکنولوژی TOSCA

فناوری رایانش ابری اخیراً برای میزبانی و ارائه خدمات از طریق اینترنت پدید آمده و به طور فزاینده ای همراه با پارادایم اینترنت اشیا به کار گرفته شده است تا بتواند باعث مقیاس پذیری و قابلیت همکاری آن شود. رایانش ابری می تواند در عین هزینه های پایین برای استقرار کل محیط های اینترنت اشیا ، قابلیت راه اندازی و ادغام سریع اشیاء جدید و برنامه های کاربردی اینترنت اشیا را ایجاد کند.

فناوری TOSCA (Topology and Orchestration Specification for Cloud Applications) یک استاندارد OASIS تایید شده برای مدل سازی، استقرار و مدیریت برنامه های کاربردی ابری است که دارای دو بخش اصلی است: (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 بسیار عمومی است به طوری که تعریف انواع دلخواه را برای توصیف مؤلفه های برنامه، به نام انواع گره، و وابستگی های آنها، به نام انواع رابطه، قادر می سازد. تصویر زیر یک نمونه از مدل TOSCA برای یک نرم افزار کاربردی انتشار و اشتراک اطلاعات می باشد.



شکل شماره ۱

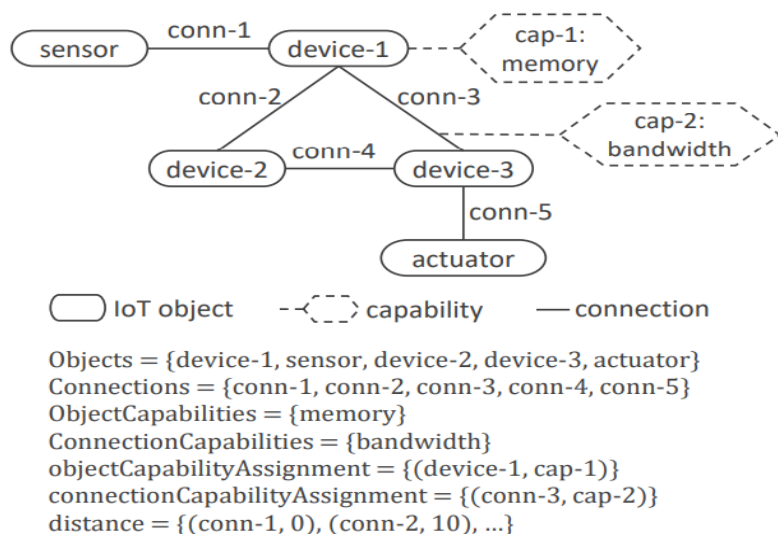
TOSCA استقرار و مدیریت برنامه های ابری را در سراسر چرخه زندگیشان آسان تر می کند بدون اینکه خللی در الزامات امنیتی، حاکمیتی و انطباقی آنها پیش آید. همچنین توانایی آن برای تسهیل یک اکوسیستم است که اجرای پورتابل برنامه ها برای ابر و بین ابرها را فراهم می سازد و این امکان را فراهم می سازد که برنامه های ابری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

۵-۱-۲ 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)، قابلیت های شی (ObjectCapabilities)، قابلیت های اتصال (ConnectionCapabilities)، تخصیص قابلیت شی (objectCapabilityAssignment)، تخصیص قابلیت اتصال (connectionCapabilityAssignment) که در نمودار زیر این مدل نمایش داده شده.



۶-۱-۲ 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، ESP8266 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 محدودیت‌های محیط‌های ناهمگن و توزیع شده را ارائه داده‌اند.

- احراز هویت (Authentication): در حوزه IoT مجوز ارتباط ماشین با ماشین یا انسان به ماشین توسط مجوز ماشین است. این قابلیت مکانیسم‌های تأیید هویت پشتیبانی شده یک دستگاه IoT را توصیف می‌کند. نمونه‌هایی از مکانیسم‌های احراز هویت عبارتند از گذرواژه، اعتبار دیجیتال (digital credentials) و گواهی‌ها

- محرمانه بودن (Confidentiality): این قابلیت توضیح می‌دهد که چگونه اطلاعات ذخیره شده در دستگاه IoT محافظت می‌شود حتی اگر دسترسی غیرمجاز رخ دهد. برای مثال می‌توان با استفاده از مکانیسم‌های رمزگذاری به محرمانگی دست یافت

- دسته دستگاه (Device category): این پایان‌نامه چهار دسته دستگاه را تعریف می‌کند که بین پارادایم‌های مختلف استقرار بر روی دستگاه‌ها متمایز می‌شود. 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 نمی‌دهد، بلکه API‌ها را از طریق یک سرور ابری برای دسترسی به حسگرها و محرک‌های آن فراهم می‌کند. نمونه‌هایی از چنین دستگاه‌هایی با پوشش بی‌سیم هستند که دائماً داده‌های خود را با سرورهای ابری همگام می‌کنند. بنابراین، قرار دادن اپراتور در دستگاه plug-and-play امکان‌پذیر نیست، با این حال، اپراتورها را می‌توان در سایر اشیاء IoT متصل به دستگاه plug-and-play از طریق API‌های ارائه شده مستقر کرد.

- یک دستگاه قابل تنظیم، مانند 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 می‌سازد، به عنوان مثال، برای استخراج و ارسال داده‌های 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 ESP8266 یا برد های 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 (IEEE802.3)، Wi-Fi (IEEE802.11)، بلوتوث (IEEE802.15.1)، بلوتوث کم انرژی (IEEE802.15.4)، LTE4G، یا G5
- حالت مصرف برق (Power consumption mode). در حوزه اینترنت اشیا، دستگاه های وابسته به باتری باید بتوانند در حالت های مصرف کم مصرف کار کنند تا بتوانند با برنامه های طولانی مدت اینترنت اشیا همگام شوند. این قابلیت حالت های مصرف انرژی را که یک دستگاه اینترنت اشیا می تواند پشتیبانی کند، توصیف می کند. 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 Pi ۳ مدل B دارای چهار پردازنده با فرکانس ۱.۲ گیگاهرتز است.
- گنجایش ذخیره سازی (Storage capacity). این قابلیت با مقدار ذخیره سازی موجود مطابقت دارد. این برای اپراتورهایی که داده های حسگر را جمع آوری می کنند یا نتایج میانی را ذخیره می کنند، 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 قبلا 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 شبکه (Networking type). این قابلیت نوع اتصال شبکه را توصیف می کند، به عنوان مثال، با سیم یا بی سیم.

۷-۱-۲ 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 می‌شود، به عنوان مثال، حسگرها، که توالی مقادیر داده را در همان ساختار تولید می‌کنند. از طرف دیگر، خروجی پردازش به سینک‌های (sinks) داده می‌رسد. اپراتور پردازش مربوط به یک فیلتر (filters) و جریان داده بین دو اپراتور پردازش مجاور مربوط به یک لوله (pipes) است.

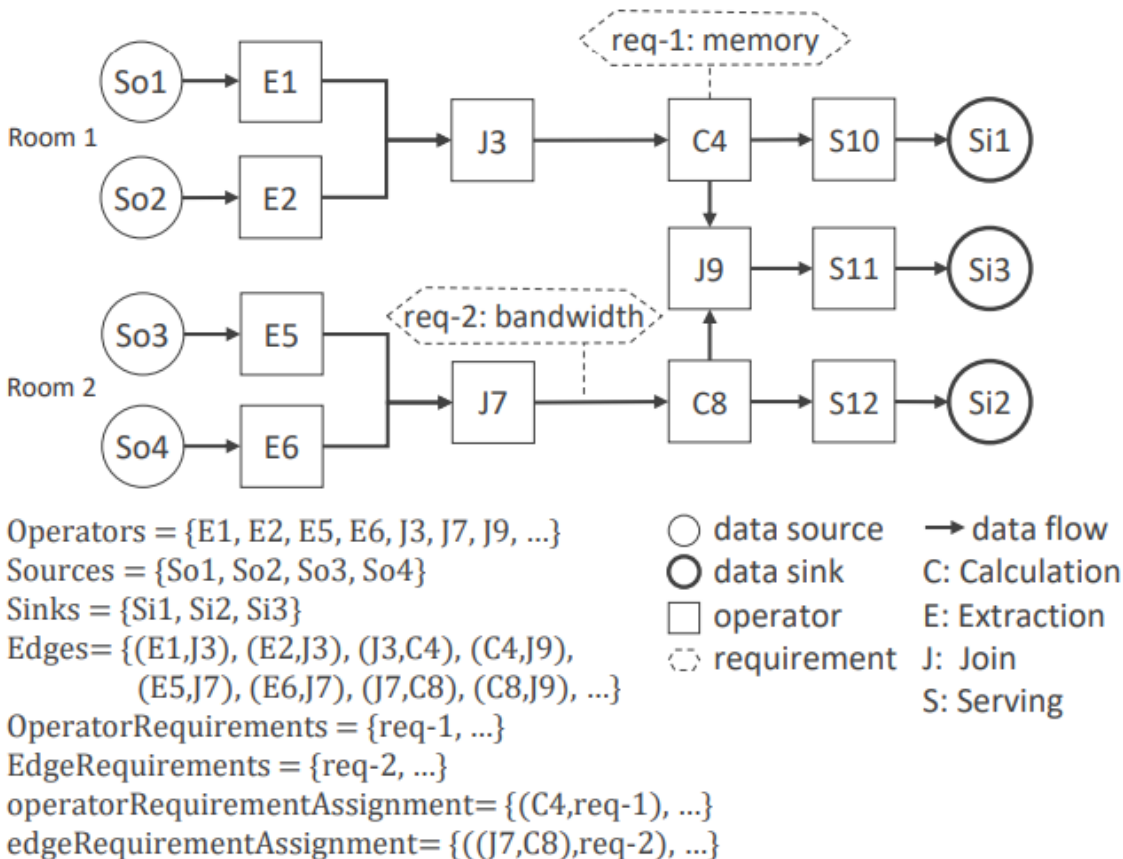
علاوه بر این، اتصالات بین منابع داده و اپراتورهای پردازش و بین اپراتورهای پردازش و سینک‌های داده نیز با لوله‌ها مطابقت دارد. این پایان‌نامه از یک نوع لوله و فیلتر استفاده می‌کند که در آن یک اپراتور پردازش می‌تواند چندین ورودی و خروجی داشته باشد. DSPM به صورت زیر قابل تعریف می‌باشد:

تعریف (DSPM) : یک مدل پردازش جریان داده چندگانه است: شامل (منابع داده (Sources)، سینک‌ها (Sinks)، اپراتورها (Operators) تمام پردازش‌ها شامل استخراج و سینک‌ها، لبه‌ها (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 قابلیت های محاسباتی نمونه دستگاه های IoT را که در بخش ۴-۱-۲ توضیح داده شده است، برآورده می کنند. برای برخی نیازها، مانند حافظه اصلی مورد نیاز، تحلیلگران دامنه معمولاً نمی دانند از چه مقادیر مشخصی باید استفاده شود. این دانش به شدت به مقدار و پیچیدگی داده ها بستگی دارد و بنابراین، باید از طریق مقادیر تجربی استنباط شود. بنابراین، مقادیر تجربی عملگرها در حین اجرای آنها جمع آوری می شود. سپس این مقادیر با استفاده از تکنیک های تحلیلی پردازش می شوند تا توصیه هایی در مورد الزامات در زمان مدل سازی به تحلیلگران حوزه ارائه شود.

شکل زیر نمونه ای از یک DSPM را نشان می دهد که شامل منابع داده، سینک ها و عملگرهای پردازشی حاوی نمایش های گرافیکی و رسمی است.



شکل شماره ۲

هدف از برنامه نمونه، نظارت بر سطوح قالب در اتاق‌های مختلف یک ساختمان به منظور تشخیص زمانی است که سطوح قالب به محدوده‌های ناسالم افزایش می‌یابد. منابع داده با سنسورهای دما (So1, So3) و سنسورهای رطوبت (So2, So4) نشان داده می‌شوند، در حالی که سینک‌های داده توسط داشبورد (Si1, Si2, Si3) نشان داده می‌شوند. علاوه بر این، این مثال از اپراتورها برای استخراج داده‌های حسگر، برای پیوستن به داده‌های حسگر بر اساس پنجره‌های زمانی، برای محاسبه سطوح قالب و ارائه داده‌ها به سینک‌های داده استفاده می‌کند. برای هر اتاق، سطح قالب به طور مداوم بر اساس مقادیر دما و رطوبت محاسبه می‌شود و در طول زمان به منظور تشخیص افزایش‌ها تجزیه و تحلیل می‌شود.

۲-۱-۸ اپراتورهای پردازش

در این پایان‌نامه، یک اپراتور پردازش یک جزء نرم‌افزاری است که عملیاتی (به عنوان مثال، فیلتر کردن داده‌ها، تابع میانگین) را بر روی یک یا چند جریان داده ورودی اجرا می‌کند. این اجزای نرم‌افزار می‌توانند به عنوان مثال، اسکریپت‌های پایتون یا شل، فایل‌های JAR یا پرس و جوهای پردازش مداوم باشند. محیط اجرای نرم

افزار مناسب (به عنوان مثال جاوا RE، مفسر پایتون، موتور پردازش جریان) مورد نیاز یک اپراتور پردازشگر باید در DSPM به عنوان نیاز محاسباتی این اپراتور مدل شود. اپراتورهای پردازش معمولاً به طور پیوسته اجرا می شوند و براساس پنجره ها، یعنی فواصل زمانی داده ها بر اساس زمان یا تعداد عناصر، به دلیل ماهیت بی نهایت بودن جریان داده هستند.

علاوه بر این، عملیات مبتنی بر پنجره ضروری است، زیرا داده ها در محیط های IoT می توانند نادقیق باشند یا به سرعت کهنه شوند. خروجی یک اپراتور پردازشگر منجر به یک یا چند جریان داده می شود که می تواند به اپراتورهای پردازشگر بعدی یا به سینک های داده ارسال شود.

فرض بر این است که اپراتورهای پردازش برای رویه های قرار دادن اپراتور آماده هستند، یعنی مکانیسم های لازم برای توزیع پردازش داده ها قبلاً به کار گرفته شده است. رویکردهایی که پردازش جریانی را در محیط های توزیع شده و محیط های IoT انجام می دهند را می توان در لیست زیر نشان داد:

- **تجمع (Aggregation):** این عملگر یک تابع تجمع را به مجموعه ای از مقادیر (به عنوان مثال، یک محیط اجرایی window) در یک جریان داده اعمال می کند و یک مقدار واحد را برمی گرداند. نمونه هایی از تجمع محاسبات میانگین، حداقل مقدار و حداکثر مقدار هستند.
- **محاسبه (Calculation):** این پردازشگر یک محاسبه را بر اساس مقادیر یک یا چند جریان ورودی انجام می دهد. به عنوان مثال، عملگر محاسبه تابعی است که سطوح قالب را بر اساس مقادیر دما و رطوبت محاسبه می کند.
- **استخراج (Extraction):** این اپراتور داده ها را از منابع داده استخراج می کند، برای مثال، می تواند کد نرم افزاری باشد که به رابط فیزیکی یک حسگر (مثلاً از طریق GPIO) برای دریافت اندازه گیری های 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.
- **تشخیص الگو (Pattern detection):** این اپراتور الگوهای خاصی را در یک جریان داده تشخیص می دهد. نمونه ای از یک الگوی مشتق شده از مفاهیم CEP یک توالی رویداد است، به عنوان مثال، رویدادهای مشخص شده در یک ترتیب خاص رخ می دهد.

- خدمت (Serving). این اپراتور داده‌ها را به سینک‌های داده ارائه می‌کند، برای مثال، می‌تواند کد نرم‌افزاری باشد که به رابط فیزیکی یک محرک برای کنترل آن دسترسی دارد، یا کد نرم‌افزاری باشد که داده‌ها را به یک برنامه داشبورد ارسال می‌کند.
- دگرگونی (Transformation). این اپراتور تبدیل‌ها را متوجه می‌شود، به عنوان مثال، در بین قالب‌های مختلف داده یا طرحواره‌های داده.

۲-۱-۹ زبان توضیح برای اینترنت اشیاء (Topic Description Language)

اینگونه تعریف می‌شود که ابزار ساده‌ای برای توصیف و یافتن موضوعات برای حسگرها و محرک‌های عمومی فراهم می‌کند. نماد TDLIoT (The Topic Description Language for the Internet) از بررسی متون و تجربیات گسترده در پلتفرم‌ها، پروتکل‌ها و برنامه‌های مختلف اینترنت اشیاء مشتق شده است. رویکرد TDLIoT در این پایان‌نامه برای ایجاد توضیحاتی استفاده می‌شود که هدف آن سهولت دسترسی به داده‌های خروجی اپراتورهای منتشر شده (به عنوان مثال، داده‌های 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 تلفن‌های هوشمند خود در مورد مکان‌های پارک خالی مطلع می‌شوند. حسگرهای فضای پارکینگ می‌توانند در دسترس بودن فضاهای پارک را تشخیص دهند و هنگامی که فضاهای پارک در دسترس هستند، رانندگان می‌توانند از طریق برنامه‌های تلفن هوشمند خود، محرک‌ها 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 می‌توان برای محرک‌ها راه‌اندازی کرد، و (iv) چگونه می‌توان به حسگرها و محرک‌ها برای استفاده دسترسی داشت.

در اینترنت اشیاء، دسترسی به حسگرها و محرک‌ها معمولاً از طریق مدل ارتباطی انتشار-اشتراک محقق می‌شود. این موضوعات برای ارائه مقادیر حسگر یا امکان دسترسی به محرک‌ها استفاده می‌شود. در محدوده این پایان‌نامه، موضوع (Topic)، بخشی است که امکان دریافت و ارسال داده‌ها را به صورت یکسان فراهم می‌کند. موضوعات را می‌توان از طریق مدل‌های ارتباطی مختلف، مانند انتشار-اشتراک یا درخواست-پاسخ، با استفاده از پروتکل‌های مختلف محقق کرد.

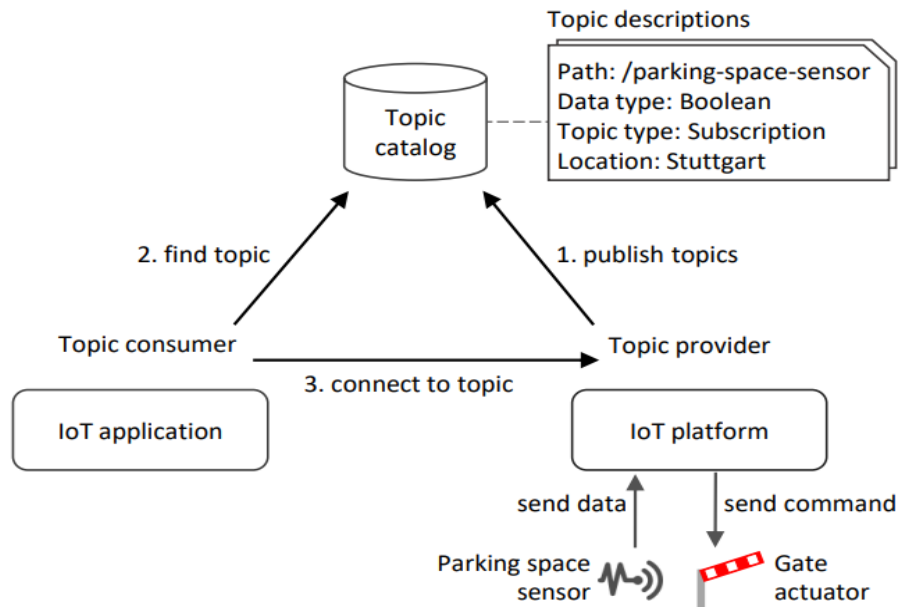
در مدل انتشار-اشتراک مبتنی بر موضوع محقق می‌شود، مشترکین در موضوعات مرتبط با زمینه خود ثبت نام می‌کنند تا زمانی که ناشران پیام‌هایی به این موضوعات ارسال می‌کنند به طور ناهمزمان مطلع شوند.

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 - 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 به عنوان برنامه خالی شناسایی شد، مطلع می‌شود.

توسعه دهندگان به دانستن همه چیز در مورد موضوعاتی که می‌توانند برای ساخت چنین برنامه‌های 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 نمی‌شوند، اما می‌توانند 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 ساده‌ای برای توصیف و یافتن موضوعات اپراتورهای منتشر شده، و علاوه بر این، منابع داده و سینک‌های انتزاعی از حسگرها و محرک‌ها فراهم می‌کند.

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** می‌تواند از طریق انتزاع از پلتفرم‌های خاص اینترنت اشیا آسان شود. موضوعات موجود در کاتالوگ را می‌توان به عنوان مثال با یک مکان خاص یا نوع حسگر جستجو کرد. رویکرد **TDLIoT**، که در شکل ۱۲ نشان داده شده است،



شکل شماره ۳

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 می شود. 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 ویژگی های زیر باشد، با این حال، در صورت لزوم می توان آن را با ویژگی های بیشتر برای توصیف یک موضوع با جزئیات بیشتر گسترش داد:

- نوع داده: نوع مقادیر ارائه شده 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
- این موجودیت ها می توانند در تعداد دلخواه توصیفات رخ دهند. در مواردی که موضوعات داده های جمع آوری شده را ارائه می دهند، به عنوان مثال، از حسگرهای مختلف.

۲-۱-۱۰ MBP

از زمان رواج اینترنت اشیا، بسیاری از پلتفرم های تجاری و غیرتجاری اینترنت اشیا برای کمک به کاربران غیرمتخصص در مدیریت اشیاء در محیط های اینترنت اشیا توسعه یافتند. با این حال، اتصال و ارائه اشیاء IoT به این پلتفرم ها همچنان برای کاربران غیرمتخصص وظایف بسیار چالش برانگیزی است.

ما اساساً MBP را برای سهولت مدیریت محیط های اینترنت اشیا طراحی کردیم، با این حال، ما همچنین یک گام عمیق تر به سمت حمایت از کاربران در طول اتصال و ارائه محیط های IoT برداشتیم. در بسیاری از پلتفرم های اینترنت اشیا، مانند: Microsoft Azure IoT - OpenMTC - IBM Watson IoT - FIWARE اشیاء به صورت دستی ثبت، متصل شده و در اختیار پلتفرم های اینترنت اشیا قرار می گیرند.

چنین وظایفی پیچیده هستند و به دانش فنی در مورد اشیاء IoT نیاز دارند. یعنی اپراتورها (کد نرم افزار) برای استخراج و ارائه داده های حسگر به برنامه های IoT و همچنین دریافت دستورات کنترل محرک از برنامه های IoT مورد نیاز هستند. چنین اپراتورهایی باید برای هر سنسور و محرک به صورت دستی ایجاد و مستقر شوند.

علاوه بر این، عملکرد نظارت باید به صورت دستی نیز پیاده سازی و مستقر شود.

استقرار اپراتورها به صورت دستی مستعد خطا و زمان بر است، زیرا یک متخصص سخت افزار باید اشیاء اینترنت اشیا را پیکربندی کند، اپراتورهای لازم را برای حسگرها و محرک های خاص نصب کند، آنها را متصل کند و

رابطه‌های قابل دسترسی برای برنامه‌های اینترنت اشیا ارائه دهد. در سناریوهای دنیای واقعی، به عنوان مثال، برای تشخیص موقعیت، نیازهای کارایی و دقت بسیار مهم هستند. با این حال، این الزامات را نمی‌توان از طریق صحافی و تهیه دستی برآورده کرد.

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

فصل سوم

۳. پیشینه تحقیق

۳-۱ مقدمه

عبارت اینترنت اشیا برای نخستین بار در سال ۱۹۹۹ توسط کوین اشتون مورد استفاده قرار گرفت و جهانی را توصیف کرد که در آن هر چیزی، از جمله اشیای بی جان، برای خود هویت دیجیتال داشته باشند و به کامپیوترها اجازه دهند آن‌ها را سازماندهی و مدیریت کنند. اینترنت در حال حاضر همه مردم را به هم متصل می‌کند ولی با اینترنت چیزها تمام اشیاء به هم متصل می‌شوند. البته پیش از آن کوین کلی در کتاب قوانین نوین اقتصادی در عصر شبکه‌ها (۱۹۹۸) موضوع نودهای کوچک هوشمند (مانند سنسور باز و بسته بودن درب) که به شبکه جهانی اینترنت وصل می‌باشند را مطرح نمود.

در سال‌های ۲۰۰۲-۲۰۰۳، امارت و وزارت دفاع ایالات متحده اولین سازمان‌های بزرگی بودند که از مدل اشتون برای ردیابی موجودی با استفاده از برچسب‌گذاری، *RFID* و اینترنت اشیا استقبال کردند. تکنولوژی زنگ در حلقه در سال ۲۰۱۱ توسط جیمی سیمینوف ساخته شد، زیرا او می‌خواست ببیند در حالی که در گاراژ مشغول کار است چه کسی پشت در خانه او است. او نمی‌توانست زنگ در را از گاراژ بشنود و مدام تحویل را از دست می‌داد.

یک مؤلفه اضافی و مهم در توسعه یک *IoT* کاربردی در ژوئن ۲۰۱۲ اتفاق افتاد، زمانی که ارائه دهندگان خدمات اینترنتی و شرکت‌های وب توافق کردند که فضای آدرس در اینترنت جهانی را با فعال کردن *IPv6* برای خدمات و محصولات خود افزایش دهند. در همان سال، اداره فدرال انرژی سوئیس یک برنامه آزمایشی به نام "شهر هوشمند سوئیس" را آغاز کرد. آنها نمایندگانی از دانشگاه‌ها، بازرگانی و مدیریت دولتی را گرد هم آوردند تا درباره ایده‌های جدید برای محیط شهری بحث کنند. شهر هوشمند سوئیس بیش از شصت پروژه در دست اجرا دارد و از مشارکت‌های علمی و نوآوری‌های جدید حمایت می‌کند.

در سال ۲۰۱۳، اینترنت اشیا به سیستمی با استفاده از فناوری‌های متعدد، از اینترنت گرفته تا ارتباطات بی‌سیم و از سیستم‌های میکرو الکترومکانیکی (*MEMS*) تا سیستم‌های تعبیه‌شده تبدیل شد. گوشی‌های هوشمند بخشی از اینترنت اشیا هستند و به ابزار ارتباطی مهمی برای بسیاری از افراد تبدیل شده‌اند. در سال ۲۰۱۵، آنها با اشتیاق

بالایی از سوی بازاریابان به اینترنت اشیا پیوستند. حسگرهای درون این دستگاه‌ها توسط بخش‌های بازاریابی نظارت می‌شوند که بر اساس مشتری و موقعیت محصول، تبلیغات خاصی را ارسال می‌کنند.

۳-۲ پیشینه مربوط به سال‌های اخیر

اقدامات و کارهای انجام شده در خصوص پردازش داده در محیط‌های اینترنت اشیا را می‌توان در چند بخش مورد بررسی قرار داد :

۳-۲-۱ مدل‌های اینترنت اشیا

رویکردهای زیادی برای مدل‌سازی محیط‌های *IoT* وجود دارد و مدل‌ها و ابزارهای *IoT* را برای ایجاد و مدیریت نمونه‌های این مدل‌های *IoT* ارائه شده‌اند. پشتیبانی ابزار از اهمیت حیاتی برخوردار است زیرا پیچیدگی مدل‌های اینترنت اشیا (به عنوان مثال، فرمت‌های پیچیده داده) را انتزاعی می‌کند و کار مدل‌سازی و مدیریت را آسان می‌کند. با این حال، رویکردهای پیشرفته ابزاری برای پشتیبانی از کل چرخه حیات محیط‌های *IoT* با مدل‌سازی می‌باشند و نه از طریق استقرار و یک مفهوم کل نگر مانند این پایان نامه برای مدل‌سازی، پیکربندی و نظارت بر محیط‌های *IoT* ارائه نمی‌کنند.

مک دونالد و همکاران یک ابزار مدل‌سازی برای مدل *homeML* معرفی می‌کند، که تنها برای خانه‌های هوشمند طراحی شده است. در مقابل، این پایان نامه از یک مدل اینترنت اشیا استفاده می‌کند که عمومی است و می‌تواند در سایر حوزه‌های اینترنت اشیا مانند کارخانه هوشمند یا شهر هوشمند نیز اعمال شود.

مایر و همکاران رویکردی را نشان می‌دهد که ابر داده‌های معنایی و استدلال را با مدل‌سازی گرافیکی ترکیب می‌کند، که در آن می‌توان اهداف یک محیط *IoT* را پیکربندی کرد، به عنوان مثال، دمای مورد نظر یک لثاق در طول روز. با این حال، این رویکرد فرض می‌کند که محیط *IoT* قبلاً مستقر شده است و اپراتورهای محاسباتی در حال اجرا هستند. علاوه بر این، تعداد زیادی پلتفرم *IoT* وجود دارد ، *FIWARE*, *GSN*, *IBM Watson IoT* , *OpenIoT* و *Microsoft Azure IoT* .

با این حال، آنها ابزاری برای مدل‌سازی و مدیریت محیط‌های کامل اینترنت اشیا فراهم نمی‌کنند، به عنوان مثال، اشیا *IoT* به صورت دستی و جداگانه در این پلت‌فرم‌های *IoT* ثبت و پیکربندی می‌شوند. این پایان‌نامه از هستی‌شناسی *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 هستی‌شناسی *IoT-Lite*، فرمت‌ها و استانداردهای دیگری نیز می‌توانند برای مدل‌سازی محیط‌های *IoT*، مانند هستی‌شناسی‌های *SSN* یا *OneM2M Base* استفاده شوند. در این پایان‌نامه، بررسی ادبیات کاملی در مورد مدل‌های پیشرفته اینترنت اشیا انجام شده و در آن مقایسه‌ای مبتنی بر معیار از مدل‌های اینترنت اشیا ارائه شده است. در نتیجه این مقایسه، هستی‌شناسی *IoT-Lite* بهترین مدل مناسب برای اهداف این پایان‌نامه نشان داده شده است.

علاوه بر این، می‌توان از مدل‌های عمومی‌تری که به‌طور خاص برای محیط‌های *IoT* طراحی نشده‌اند نیز استفاده کرد. برای مثال، استاندارد توپولوژی و هماهنگ‌سازی برای برنامه‌های کاربردی ابری *TOSCA* می‌تواند مورد استفاده قرار گیرد، که در اصل برای برنامه‌های کاربردی ابری طراحی شده است، اما با این وجود ابزاری را برای مدل‌سازی محیط‌های اینترنت اشیا و برنامه‌های کاربردی اینترنت اشیا ارائه می‌دهد.

۳-۲-۲ اشیا در محیط اینترنت اشیا

Rizou و همکاران یک الگوریتم ابتکاری را ارائه می‌دهد که به توزیع اپراتورهای جریان داده اجازه می‌دهد تا کمترین تأخیر ممکن را برای پردازش داده‌ها فعال کند. این الگوریتم بهترین توزیع ممکن را جستجو می‌کند تا تاخیر شبکه را به حداقل برساند.

برخلاف این پایان‌نامه، الگوریتم مستقیماً بر روی اشیا *IoT* موجود، نه به صورت مرکزی، اجرا می‌شود. با این حال، رویکرد *Rizou* و همکاران. مستلزم آن است که اشیا *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 پایان‌نامه بیشتر به دنبال یک جفت آزاد بین اشیا *IoT* است. در بهترین حالت، اشیا اینترنت اشیا نباید یکدیگر را بشناسند و در نتیجه نیازی به برقراری ارتباط مستقیم ندارند. علاوه بر این، رویکرد *Rizou* و همکاران. تنها تأخیر را به عنوان یک الزام برای پردازش داده‌ها در نظر می‌گیرد، بنابراین، سایر الزامات، همانطور که در این پایان‌نامه توضیح داده شده است، پشتیبانی نمی‌شوند.

Cipriani و همکاران رویکردی را برای قرار دادن اپراتور پرس‌وجوهای پردازش جریان ارائه می‌کند، که در آن اهداف خاص (یعنی الزامات) برنامه‌های مبتنی بر جریان برای تحقق تصمیم‌گیری‌های لازم برای قرار دادن در نظر گرفته می‌شوند. این رویکرد، قرار دادن گراف‌های پرس و جو برای جریان‌های داده (*M-TOP*) توسط اپراتور چند هدفه نامیده می‌شود. مشابه این پایان‌نامه، *M-TOP* حاشیه نویسی الزامات را بر روی اپراتورهای پردازش امکان پذیر میکند. این الزامات باید توسط گره‌های محاسباتی موجود (به عنوان مثال، اشیا اینترنت

اشیا) برآورده شوند تا به یک مکان مناسب اپراتور دست 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.

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

Bumgardner و همکاران یک زبان مدل سازی مبتنی بر نمودار به نام *Cresco Application Model (CAM)* را ارائه می کند تا منطق برنامه های کاربردی مبتنی بر جریان توزیع شده را توصیف کند. مدل های *CAM* مشابه مدل های *DSPM* ارائه شده در این پایان نامه هستند. علاوه بر این، *Bumgardner* و همکاران. از طریق یک الگوریتم حریصانه، توزیع بهینه اپراتورها را بر روی اشیا 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.

۳-۲-۳ اپراتورها در محیط اینترنت اشیا

لی *Li* و همکاران پیشنهاد می کند 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 اجرای از پیش نصب شده بر روی اشیا اینترنت اشیا ندارد.

هور *Hur* و همکاران یک منطق توصیف خدمات معنایی (*SSD*) و یک معماری سیستم را برای استقرار خودکار دستگاه های *IoT* در میان افزارهای ناهمگن اینترنت اشیا، با هدف حل مشکلات قابلیت همکاری بین آنها پیشنهاد می کند. 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 دستگاه های اینترنت اشیا با استفاده از میان افزاری به نام پلتفرم مدیریت منابع (*RMP*) معرفی می کند. *RMP* ثبت آسان دستگاه های *IoT* و اتصال آنها از طریق آداپتورها را امکان پذیر می کند. آداپتور قطعه ای از کد است که حاوی منطق خواندن مقادیر حسگر دستگاه های *IoT*، ارسال مقادیر حسگر به *RMP* و فراخوانی محرک ها 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 دستگاه های سخت افزاری تمرکز کنید، در حالی که رویکرد این پایان نامه به علاوه به استقرار اپراتورهای محاسباتی در کل محیط های *IoT* می پردازد. استقرار خودکار در محیط های اینترنت اشیا را می توان با چندین رویکرد، به عنوان مثال، با استفاده از اسکریپت های *Shell*، *Chef* یا *Puppet* تحقق بخشید. با این حال، برخلاف این رویکردها، رویکردهای مبتنی بر استاندارد در این پایان نامه از *TOSCA* استفاده می کنند و یک رویکرد عمومی مبتنی بر مدل های توپولوژی و نماد گرافیکی مربوطه را امکان پذیر می سازند.

این مدل های توپولوژی بسیار سازگار هستند به گونه ای که اجزای نرم افزاری منفرد یک توپولوژی را می توان به راحتی تعویض کرد، و علاوه بر این، گسترش داد. در روش های دیگر، این نیاز به تلاش زیادی برای انطباق دارد، به عنوان مثال، هنگام ویرایش اسکریپت های 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.

۳-۲-۴ نظارت بر عملیات ها

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. هستی‌شناسی توسعه یافته از طریق یک رابط کاربری قابل دسترسی است، جایی که کاربر رویدادهای مورد علاقه را مشخص می‌کند. سپس مشخصات یک رویداد مورد علاقه به منظور تولید دستورات پیکربندی برای یک سیستم CEP پردازش می‌شود.

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 نمی‌شوند، بلکه به عنوان موقعیت‌ها انتزاع می‌شوند. رویکرد در این پایان‌نامه همچنین تبدیل مشخصات رویداد (به عنوان مثال، موقعیت‌های مدل شده) به پرس و جوهای CEP برای یک سیستم CEP را درک می‌کند. تفاوت این است که این پایان‌نامه از هستی‌شناسی برای مدل‌سازی موقعیت‌های مورد علاقه استفاده نمی‌کند. حسن و همکاران پیشنهاد می‌کند از CEP همراه با غنی‌سازی پویا از داده‌های حسگر به منظور تحقق آگاهی از موقعیت استفاده شود. در این رویکرد، موقعیت‌های مورد علاقه مستقیماً در موتور CEP تعریف می‌شوند، یعنی کاربر موقعیت‌های مورد علاقه را با استفاده از زبان‌های پرس و جو CEP فرموله می‌کند.

یک جزء غنی‌سازی پویا، داده‌های حسگر را قبل از ارزیابی موتور CEP پردازش و غنی می‌کند. این رویکرد رویکرد موجود در این پایان‌نامه به شرح زیر است: هیچ غنی‌سازی دینامیکی پیچیده‌ای از داده‌های حسگر در این پایان‌نامه انجام نشده است. اطلاعات لازم در مورد سنسور برای تشخیص موقعیت (به عنوان مثال، شناسایی سنسور) در حداقل نگه داشته می‌شود. این اطلاعات همراه با خواندن سنسور مستقیماً در اختیار موتور CEP قرار می‌گیرد و هر مرحله پردازش بیشتر داده‌های حسگر را انجام می‌دهد. علاوه بر این، به جای تعریف موقعیت‌ها به طور مستقیم به عنوان پرس و جوهای CEP، که بسته به موقعیت می‌تواند طولانی و پیچیده باشد،

این پایان‌نامه موقعیت‌های مورد علاقه را به عنوان الگوهای موقعیت تعریف می‌کند. انتزاع ارائه شده از طریق الگوهای موقعیت، استفاده از موتورهای CEP و همچنین استفاده از فناوری‌های دیگر را برای تشخیص موقعیت ممکن می‌سازد. علاوه بر این، استفاده از الگوهای موقعیت نیز مرحله مدل‌سازی موقعیت‌ها را برای کاربر تسهیل می‌کند، به طوری که کاربر مجبور نیست با پیچیدگی فرمول‌بندی پرس و جوهای CEP سروکار داشته باشد.

فرمول پرس و جویهای CEP توسط تبدیل‌ها مراقبت می‌شود، که به طور خودکار پرس و جویهای CEP لازم را برای یک الگوی موقعیت معین ایجاد می‌کند. گلوبیفیکاسیون و همکاران رویکرد مشابهی را ارائه می‌دهد که زمینه را از طیف گسترده‌ای از منابع برای شناسایی موقعیت با استفاده از فناوری‌های پردازش رویداد یکپارچه می‌کند. با این حال، آنها هیچ گونه انتزاعی ارائه نمی‌دهند، به عنوان مثال، کاربران تشخیص موقعیت باید خودشان جستارهای CEP را ایجاد کنند. این امر به خصوص برای کارشناسان حوزه، به عنوان مثال، در کارخانه‌ها که دانش گسترده‌ای در زمینه علوم کامپیوتر ندارند، دشوار است.

فصل چهارم

۴. تجزیه و تحلیل موضوع انتخابی

۴-۱ مقدمه

پیشرفت‌های اخیر در زمینه‌های مختلف، از جمله فناوری‌های حسگر، شبکه و پردازش داده‌ها، چشم‌انداز اینترنت اشیا (IoT) را قادر ساخته است که هر روز بیشتر و بیشتر 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 درآمده‌اند و با این حجم فزاینده داده‌ای که به طور مداوم پردازش می‌شود، چالش‌های متعددی برای پردازش کارآمد داده‌های اینترنت اشیا وجود دارد. به عنوان مثال، چگونه می‌توان پردازش داده‌های اینترنت اشیا را بدون تأثیر بر واکنش پذیری برنامه‌های آن تحقق بخشید. علاوه بر این، چگونه می‌توان از طریق پردازش داده‌های IoT، نیازمندی‌های مختلف عملکردی، غیرعملکردی و تعریف‌شده توسط کاربر برنامه‌ها را برآورده کرد.

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

این پایان نامه دکتری توسط مدل های اطلاعاتی مختلف و تکنیک های قرار دادن اپراتور پشتیبانی می‌شود، به طوری که کل چرخه حیات محیط های اینترنت اشیا و برنامه های مبتنی بر جریان داده را می‌توان 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 زیرساخت‌های ابری تنها در مواردی استفاده می‌شوند که محیط‌های اینترنت اشیا منابع پردازش کافی را برای کاربرد اینترنت اشیا ارائه نمی‌دهند.

از طریق رویکرد این پایان نامه دکتری، پردازش داده‌های برنامه‌های کاربردی اینترنت اشیا را می‌توان برای موارد استفاده خاص، پشتیبانی از نیازهای خاص دامنه‌ها، و علاوه بر این، کاربران برنامه‌های اینترنت اشیا، تنظیم کرد. پس از تعیین مکان‌های امکان‌پذیر، اپراتورهای پردازش با استفاده از استانداردهایی مانند TOSCA بر روی اشیاء

IoT مربوطه مستقر 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 اشیا مستقر شده، به‌طور مداوم نظارت می‌شود.

رویکرد این پایان‌نامه دکترا توسط پلتفرم چند منظوره *Binding and Provisioning (MBP)* که یک پلت‌فرم منبع باز اینترنت اشیا، می‌باشد پشتیبانی می‌شود.

۴-۲ مدل محیط اینترنت اشیا

پیشرفت مستمر در فناوری‌های حسگر و شبکه، وجود دستگاه‌های اینترنت اشیا را امکان‌پذیر کرده است که به هم پیوسته هستند و به‌طور 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 می‌شود.

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

به‌طور معمول، اشیا IoT توسط پلتفرم‌های IoT مدیریت می‌شوند که دسترسی به برنامه‌های IoT را از طریق API‌های سطح بالا فراهم می‌کنند و در سال‌های اخیر، بسیاری از پلتفرم‌های IoT توسعه یافته‌اند. در بسیاری از رویکردها، مانند FIWARE, IBM Watson IoT, OpenMTC, Microsoft Azure IoT اشیا IoT به صورت دستی و جداگانه ثبت و با این پلتفرم‌های IoT پیکربندی می‌شوند. با این حال، این یک کار پیچیده و زمان‌بر است. علاوه بر این، محیط‌های IoT بسیار پویا هستند، به عنوان مثال، دستگاه‌ها 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. IoTEM‌ها در پلتفرم چند منظوره *Binding and Provisioning (MBP)* ادغام شده و توسط آن برای ثبت و پیکربندی خودکار محیط‌های IoT به عنوان یک کل به جای پیکربندی هر شی اینترنت اشیا به صورت جداگانه استفاده می‌شود. 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	X	X	X	✓	خانه های هوشمند
IEEE ۱۴۵۱	standard	X	✓	✓	X	متمرکز بر روی سنسورها
IoT ARM	Non-standard	✓	X	X	✓	مدل مرجع عمومی
IoT-Lite	submitted	✓	✓	✓	✓	حسگرها و SSN ontology
IoT MC	standard	✓	✓	✓	X	IoTivity
IoT-O	standard ext	✓	X	X	X	حسگرها و SSN ontology
Nexus	Non-standard	✓	✓	✓	✓	جغرافیایی سازی
oneM2M	standard	✓	✓	✓	✓	تمرکز بر خدمات دستگاه های IoT
OPC-UA	standard	✓	✓	✓	X	کارخانه های هوشمند
SenML	standard	X	✓	✓	✓	تمرکز بر حسگرها و مقادیر حسگر
SensorML	standard	X	✓	✓	✓	پشتیبانی از پردازش ها
SSN	standard	✓	✓	✓	✓	مورد استفاده IoT-Lite / IoT-O
TDLIoT	Non-standard	X	✓	✓	✓	نمونه اولیه تحقیق
Vorto	Non-standard	X	✓	✓	X	زبان برنامه نویسی

معیار تکامل نشان می دهد آیا مدل یک مدل استاندارد تایید شده یک سازمان، مانند OASIS، C3W یا OGC است یا خیر و می توان فرض گرفت سازمان ها تحت یک فرآیند بررسی کامل قرار گرفتند و یک استاندارد مزایایی را ارائه می کند.

معیار سلسله مراتبی یک عامل مهم هنگام مدل سازی محیط ها 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 گروه بندی، باید بتوان سیستم های پیچیده را مدل سازی کرد، مانند ماشین های

تولیدی در یک کارخانه هوشمند، که حاوی مقدار بالایی از دستگاه‌ها، حسگرها و محرک‌ها هستند و از طریق انتزاع، می‌توان انواع عمومی را تعریف کرد. به عنوان مثال، مازول‌های سنسور مختلف اندازه‌گیری دما را می‌توان 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.

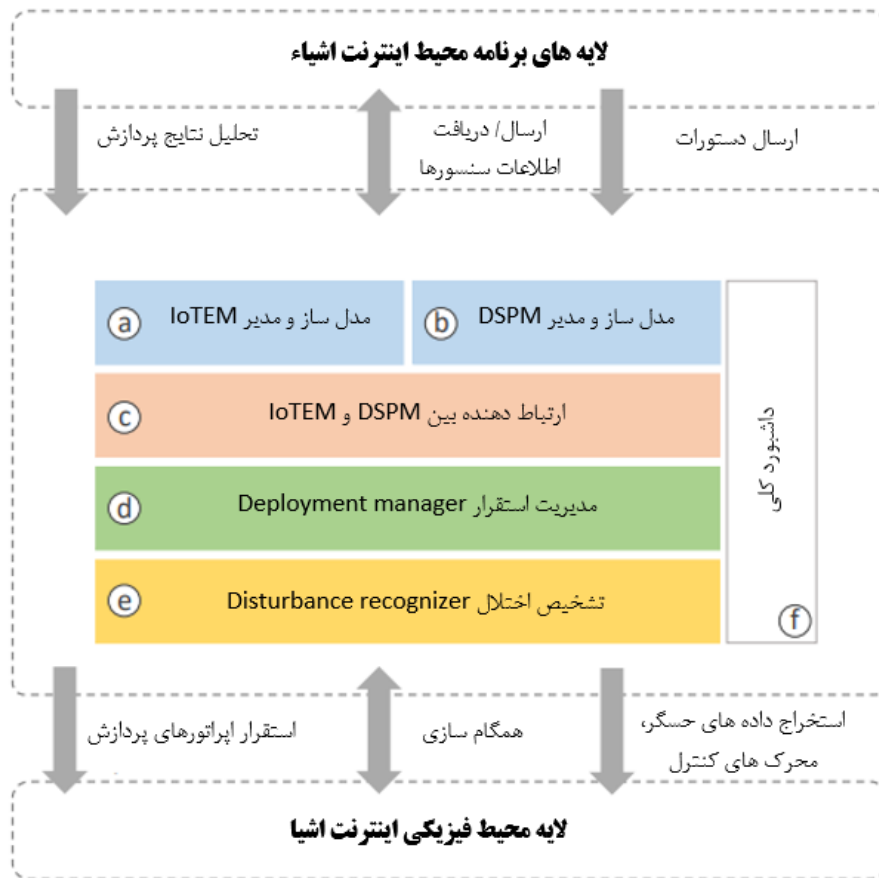
برای درک این موضوع، مدل باید یا منبع باز در دسترس باشد، یا اگر منبع بسته است، باید توسط یک سازمان بزرگتر توسعه و استفاده شود.

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

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 می‌باشد، به عنوان مثال، رویدادهایی که در یک خانه هوشمند رخ می‌دهد.

۳-۴ معماری کلی

معماری کلی طبق تحقیقات این پایان‌نامه از سه لایه اصلی تشکیل شده است: لایه محیط فیزیکی IoT، لایه برنامه کاربردی IoT و لایه پلتفرم اتصال و تامین چند منظوره (MBP) که شکاف بین محیط‌های فیزیکی IoT و برنامه‌های کاربردی IoT را پر می‌کند. تصویر زیر این نمای کلی را نشان می‌دهد:



شکل شماره 4

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 می‌کند 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 می‌کند. علاوه بر این، الگوریتم هایی را برای تصمیم‌گیری خودکار در مورد قرار دادن اپراتور، بر اساس الزامات اپراتورهای پردازش و قابلیت‌ها ارائه می‌دهد.

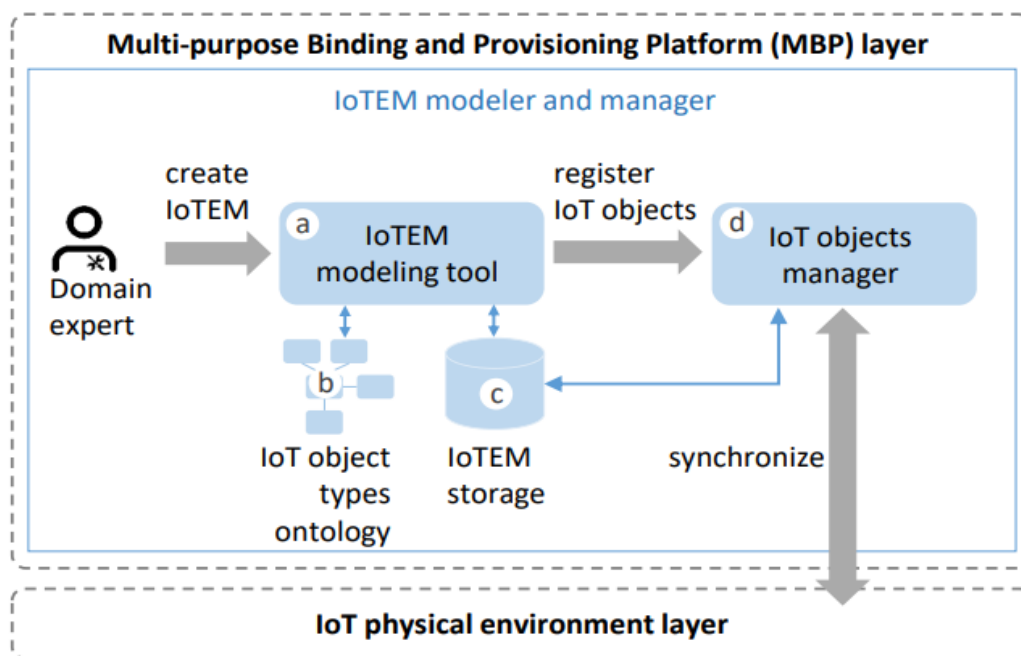
بخش مدیریت استقرار (Deployment manager) مسئول استقرار اپراتورهای پردازش بر روی اشیاء IoT است، به طوری که اجرای DSPM ها می تواند آغاز شود. علاوه بر این، هر نرم افزار طراحی شده مورد نیاز دیگر برای نظارت بر اشیاء اینترنت اشیا، می تواند از طریق این مؤلفه نیز مستقر شود.

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، ابر داده 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.

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

در معماری کلی، بخش مدل ساز و مدیر IoTEM با پشتیبانی متخصصان دامنه امکان ارائه ابزاری برای ایجاد و ذخیره IoTEM و علاوه بر این، مدیریت IoTEM های ثبت شده در MBP را فراهم می کند. بخش مدل ساز و مدیر IoTEM در شکل زیر نشان داده شده است.



شکل شماره 5

این مدل شامل ابزار مدل سازی گرافیکی IoTEM (بخش a) است که برای مدل سازی محیط اینترنت اشیا با کشیدن و رها کردن انواع شی IoT (بخش b) در یک محیط مدل سازی استفاده می شود.

انواع شی IoT، مانند Raspberry Pis، حسگرهای دما، و ماشین های مجازی، توسط یک هستی شناسی (ontology) ارائه می شوند. IoTEM ها، که نمونه هایی از هستی شناسی هستند، در ذخیره سازی IoTEM (بخش c) ذخیره می شوند.

علاوه بر این، ابزار مدل سازی IoTEM به متخصصان حوزه امکان می دهد، علاوه بر ذخیره و بازیابی IoTEM، تمام اشیاء IoT یک IoTEM را به طور هم زمان در MBP ثبت کنند. این ثبت، مدیر اشیاء اینترنت اشیا (بخش d) را فعال می کند تا همتایان دیجیتالی (به عنوان مثال، دوقلوهای دیجیتالی) اشیاء IoT مدل شده را نمونه برداری کند.

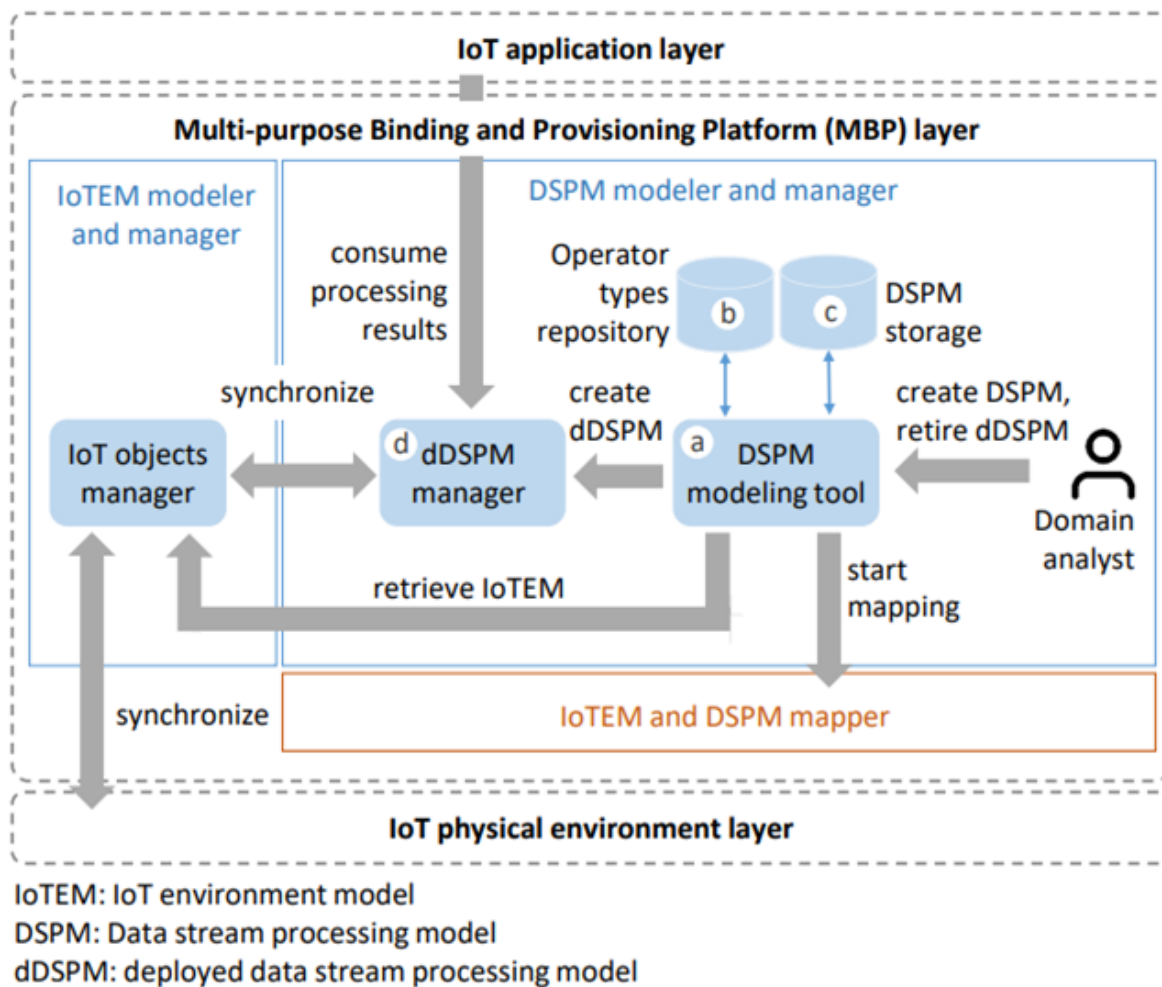
این همتهای دیجیتالی در حافظه IoTEM ذخیره می شوند. مدیر اشیاء IoT دائماً با محیط فیزیکی اینترنت اشیا هماهنگ می شود تا اختلالات موجود در محیط فیزیکی را تشخیص دهد، مانند زمانی که دستگاه ها معیوب می شوند. وضعیت فعلی اشیاء ثبت شده IoT را می توان در داشبورد MBP بررسی و تجسم کرد.

۴-۴ 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 داده ها شکل جریان های داده ای را در خود جای می دهند که پایدار نیستند، بلکه برای پردازش در جریان های متعدد، پیوسته، سریع و متغیر با زمان می آیند. رویکردهای خوبی برای پردازش جریان های داده، پردازش رویداد پیچیده و پردازش جریان است.

۴-۴-۱ ساختار معماری و پیاده سازی - مدل ساز و مدیر DSPM

در معماری کلی، مؤلفه مدلساز و مدیر DSPM از تحلیلگران دامنه با ارائه ابزاری برای ایجاد و ذخیره DSPM ها و علاوه بر این، مدیریت DSPM های مستقر در محیط های IoT پشتیبانی می کند. این مشخصات در شکل زیر نشان داده شده است:



شکل شماره ۶

این پایان نامه از ابزار FlexMash (Flexible Modeling and Execution of Data Mashups) که توسط Hirmer و همکاران، به عنوان ابزار مدل سازی DSPM (بخش a) برای ایجاد DSPMs پیشنهاد شد. این یک رابط گرافیکی ارائه می کند و مدل زیربنایی مبتنی بر JSON آن را می توان برای فعال کردن حاشیه نویسی نیازمندی های محاسباتی در گره ها و لبه ها گسترش داد، که مقدمه ای برای مفاهیم این پایان نامه است. ابزار مدل سازی DSPM با کشیدن و رها کردن منابع داده، سینک های داده و اپراتورهای پردازش به منطقه مدل سازی، ایجاد DSPM را امکان پذیر می سازد. در حالی که اپراتورها در مخزن نوع اپراتور (بخش b) موجود هستند، منابع داده و سینک ها از حسگرها و محرک های موجود در مدل های IoTEM که از مدیر اشیاء IoT بازیابی می شوند، انتزاع می شوند. علاوه بر این، DSPM مدل سازی شده در ذخیره سازی DSPM (بخش c) ذخیره می شود. از طریق ابزار مدل سازی 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 مدل سازی تحلیلگران دامنه را قادر می سازد تا dDSPM ها را توسط مدیر dDSPM متوقف نماید.

۵-۴ ارتباط بین DSPM ها در مدل های اینترنت اشیاء (IoTEM)

نگاشت مدل های پردازش جریان داده (DSPM) بر روی مدل های محیط اینترنت اشیا (IoTEMS) شامل دو رویکرد برای تحقق نگاشت است: (۱) یک رویکرد خودکار، که در آن یک طرح نقشه برداری به طور خودکار توسط الگوریتم های مختلف تولید می شود، و (ب) یک رویکرد دستی، که در آن تحلیل گران حوزه خودشان تصمیم می گیرند که کدام اپراتورها باید توسط کدام عملگر اجرا شوند.

۵-۴-۱ 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 الگوریتم هایی محقق می شود که قادر به ارزیابی و مطابقت با قابلیت های 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، الگوریتم های ابتکاری، مانند الگوریتم هایی که توسط V.M. Lo ارائه شده اند، معمولاً برای حل این نوع مسائل استفاده می شوند. 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 IoTEM اختصاص می دهد.

Edge Mapping: Edges → Paths (IoTEM)

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

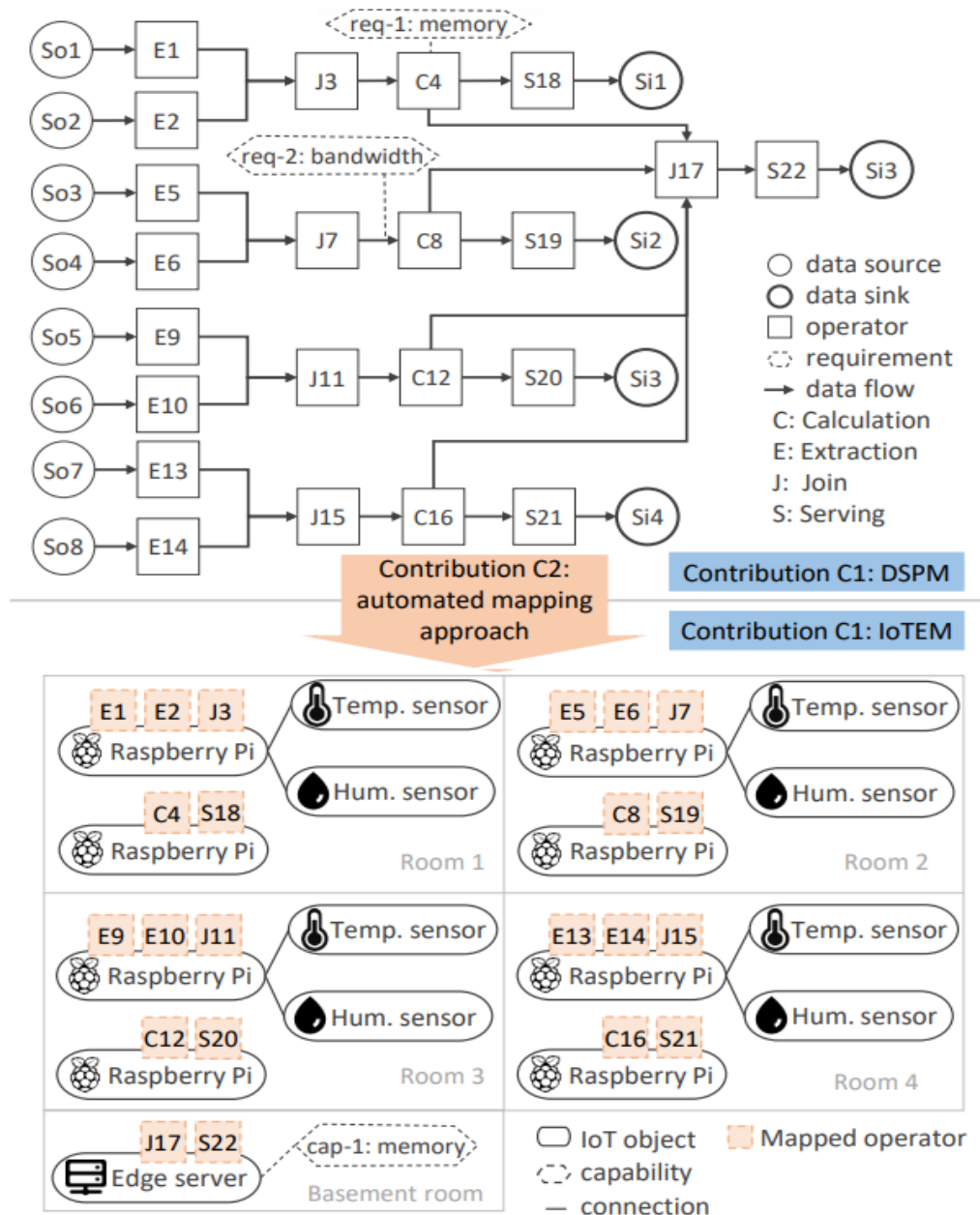
در این پایان نامه، دو الگوریتم برای حل مسئله قرارگیری اپراتور DSPM ارائه شده است: (i) یک روش حریصانه و یک (ii) روش عقبگرد که نوع حریصانه راه حل را به موقع پیدا می کند، با این حال، یافتن بهترین راه حل ممکن را تضمین نمی کند.

LO نشان می دهد که یک الگوریتم حریصانه ساده بسیار کارآمدتر است از الگوریتم های ابتکاری پیچیده تر. در مقابل، الگوریتم های عقبگرد تمام راه حل های ممکن را محاسبه 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 سناریوهای ساده ای که شامل اشیاء IoT کمتری هستند استفاده شود.

هدف اصلی هر دو الگوریتم یافتن مجموعه ای از اشیاء IoT است که قابلیت های آنها نیازهای اپراتورها را برآورده می کند. قراردادن اشیاء IoT نزدیک به منابع داده ترجیح داده می شوند، زیرا تشخیص زودهنگام موقعیت های بحرانی و اقدامات به موقع مربوطه را ممکن می سازد. علاوه بر این، حجم داده های مبادله شده در محیط اینترنت اشیا را می توان با جمع آوری و فیلتر کردن داده ها در نزدیکی منابع آنها و در اولین فرصت ممکن در زنجیره پردازش داده کاهش داد.

برای هر دو الگوریتم، فرض بر این است که اشیاء IoT که از طریق رابط های سخت افزاری به حسگرها یا محرک ها متصل هستند، قابلیت های محاسباتی کافی را برای اجرای استخراج داده های حسگر یا اپراتورهای ارائه داده را دارند. به این معنا که یک اپراتور استخراج داده همیشه می تواند مستقیماً روی شی IoT قرار گیرد که به طور فیزیکی به حسگر مربوطه متصل است.

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



شکل شماره 7

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 دستگاه های 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، E1 تا E14، بر روی دستگاه های IoT مستقر می شوند، که حسگرها به طور فیزیکی به آن متصل می شوند (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 S18، S19، S20 و S21 نگاشت می شوند.

بعد از اینکه عملگرهای سرویس نگاشت شدند، عملگرهای Join در مرحله بعد نگاشت می شوند. برای J3 که به داده های خروجی از اپراتورهای استخراج E1 و E2 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 J3 را برآورده می کند و قابلیت های کافی باقی مانده است، بنابراین، اپراتور Join J3 می تواند بر روی همین دستگاه مستقر شود.

همین رویه برای بقیه عملگرهای J7 Join J11 و J15 انجام می شود. پس از آن بار دیگر منابع مورد نیاز از قابلیت های دستگاه ها کم می شود. سپس عملگرهای محاسباتی C4، C8، C12 و C16 نگاشت می شوند. برای C4، دوباره، نزدیکترین دستگاه در مدل محیط اینترنت اشیا به اپراتور قبلی خود (یعنی اپراتور نگاشت شده J3) جستجو می شود. این دستگاهی است که قبلاً شامل عملگرهای نقشه برداری شده E1، E2 و J3 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 C4 مطابقت ندارد. در نتیجه نزدیکترین دستگاه بعدی جستجو می شود که در همان اتاق دستگاه حاوی E1، E2 و J3 است. با فرض اینکه این دستگاه قابلیت محاسباتی کافی برای محاسبه سطح قالب را دارد، اپراتور C4 بر روی این دستگاه نگاشت شده و منابع آن مصرف می شود. در اتاق های باقی مانده، عملگرها به طور مشابه نگاشت می شوند.

در نهایت، سطح قالب محاسبه شده هر چهار اتاق با استفاده از عملگر J17 جمع آوری می شود، که داده های خروجی مناسب برای تجسم داشبورد برای کاربران تولید می کند. در این مثال، دستگاه های اینترنت اشیا در اتاق های 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 J17 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 J17 را برآورده کنند. در نتیجه، J17 باید روی یک دستگاه اینترنت اشیا دیگر نگاشت شود.

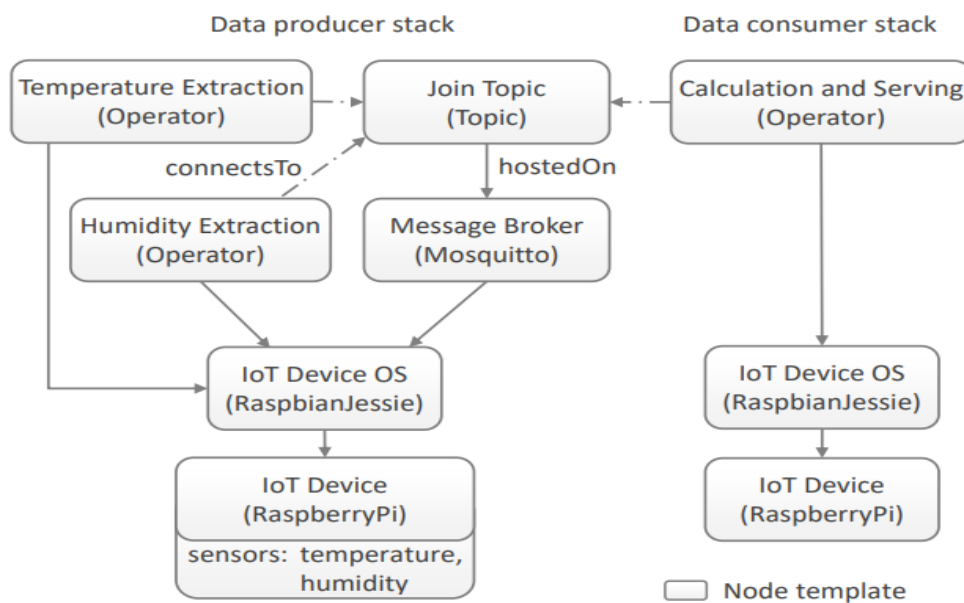
در این حالت سرور لبه در اتاق زیرزمین ساختمان برای اجرای این عملگر علامت گذاری شده است. از آنجایی که مفاهیم توضیح داده شده برای نگاشت بر اساس یک روش عمومی است، مدل های عمومی به کار گرفته شده اند (DSPM، IoT-EM)، و علاوه بر این، مجموعه ای از الگوریتم های نگاشت قابل توسعه ارائه شده است، این رویکرد قابل انتقال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

۴-۵-۲ 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 داده ایم که چگونه برنامه های نگاشت دستی مبتنی بر TOSCA برای سناریوهای اینترنت اشیا در حوزه خانه هوشمند می توانند محقق شوند، و علاوه بر این، چگونه می توان 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 شماره ۸

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

همانطور که در شکل ۵.۲ در پایین نشان داده شده است. علاوه بر این، عملگرهای پردازش یک DSPM به عنوان الگوهای گره نیز انتزاع شده و در شکل ۵.۲ در بالا نشان 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 موضوع میزبانی شده 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 دستگاه‌هایی که سیستم عامل ارائه نمی‌دهند، مانند ساعت هوشمند، یک الگوی گره برای سیستم عامل مورد نیاز نیست. در مقابل، یک الگوی گره زمان اجرا از راه دور (به عنوان مثال، یک سرور لبه) که به عنوان یک دروازه عمل می‌کند باید ارائه شود، که امکان قرار دادن اپراتور روی آن را فراهم می‌کند که به چنین دستگاه‌هایی متصل می‌شود و داده‌های حسگرهای تعبیه شده مربوطه را استخراج می‌کند. مثال شرح داده شده در زیر

عمدتاً بر روی دستگاه‌های **IoT** متمرکز است که یک سیستم عامل را ارائه می‌دهند مثال: **Raspberry Pis**.

پشته تولید کننده داده در شکل ۸ شامل یک دستگاه اینترنت اشیا است که دارای سنسورهای دما و رطوبت و زیرساختی است که دسترسی به این مقادیر حسگر را فراهم می‌کند. به عنوان مثال، مقادیر دما و رطوبت توسط حسگرهای 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 ارسال هر دو مقدار به یک موضوع در کارگزار پیام، به یکدیگر متصل می‌شوند. با استفاده 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 استفاده را مشخص می‌کند، در این مورد، سیستم عامل **Raspbian Jessie**. برای مدل‌سازی 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 دسترسی آن، که می‌تواند از طریق احراز هویت کاربر یا بر اساس آن باشد، ارائه شود. یک کلید **SSH** بر اساس این سیستم عامل، اسکریپت‌های اپراتور را می‌توان به طور خودکار برای دسترسی به رابط‌های سخت افزاری حسگرها و استخراج داده‌های آنها مستقر کرد. الگوی گره استخراج دما و الگوی گره استخراج رطوبت به عنوان رابط بین دستگاه فیزیکی اینترنت اشیا عمل می‌کنند. 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 و 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.

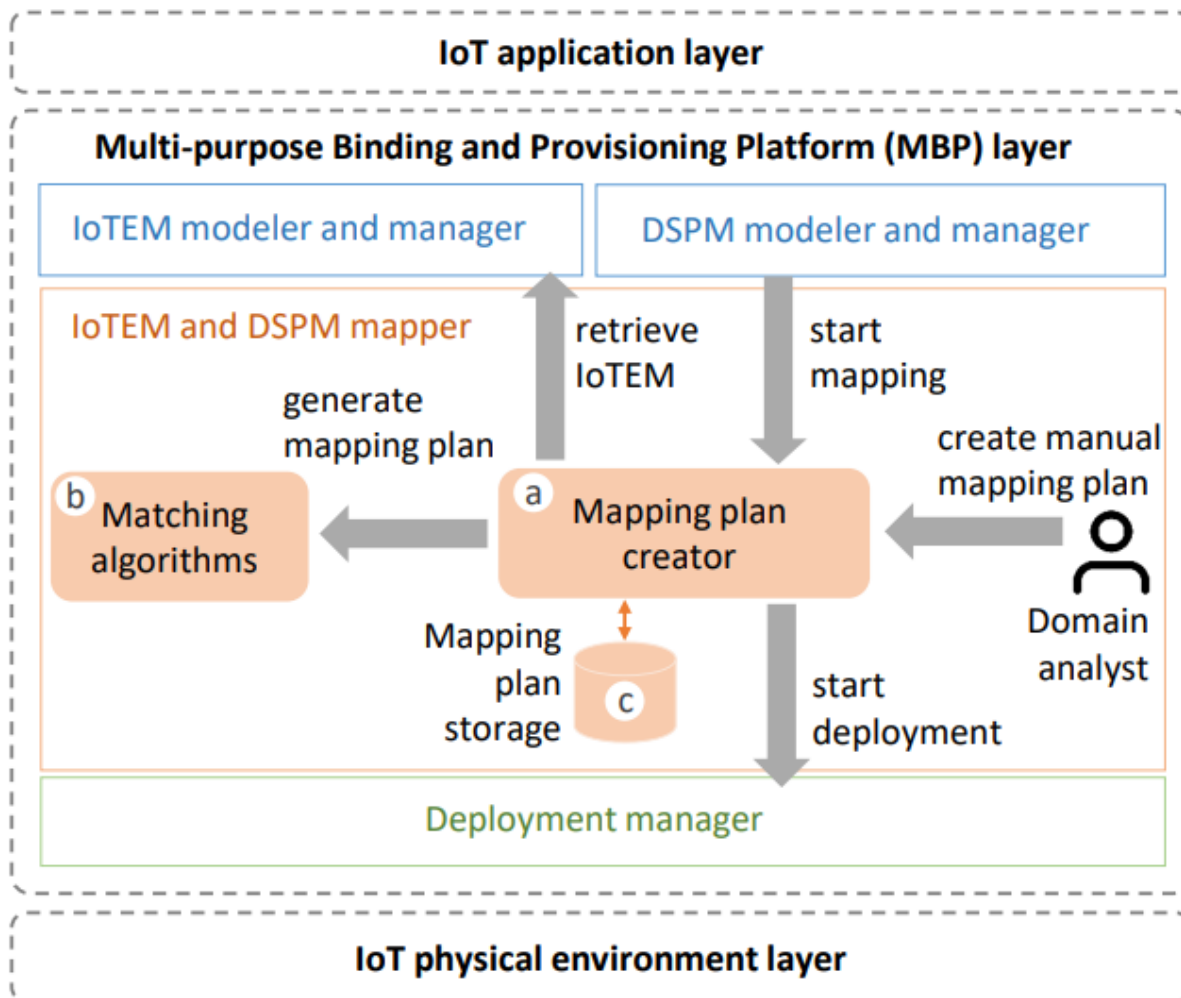
مصرف کنندگان داده به روشی مشابه با تولیدکنندگان داده کار می کنند به این معنا که بر اساس الگوی انتشار-اشتراک ایجاد می کنند. در نتیجه، برای مدل سازی آنها با استفاده از TOSCA، می توان از انواع گره های معرفی شده از قبل استفاده مجدد کرد، یعنی انواع Raspberry Pi، Raspbian Jessie و Node Operator.

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 مدل سازی 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.

۴-۵-۳ 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 شماره ۹

در رویکرد خودکار، خالق طرح تطبیقی بخش a الگوریتم‌های تطبیق b را شروع می‌کند که یک طرح تطبیقی را برمی‌گرداند. در رویکرد دستی، تحلیل‌گران دامنه می‌توانند خودشان از طریق ایجادکننده نقشه تطبیقی طرح‌های تطبیقی ایجاد کنند.

در این مورد، یک مدل‌ساز برای مدل‌های توپولوژی TOSCA، مانند Winery می‌تواند برای ایجاد و ذخیره یک نقشه تطبیقی مبتنی بر TOSCA استفاده شود. هنگامی که یک طرح تطبیقی ایجاد شد، استقرار طرح های تطبیقی مبتنی بر TOSCA را می‌توان با استفاده از یک موتور استقرار مبتنی بر TOSCA، مانند OpenTOSCA، با فعال کردن بخش Deployment Manager شروع کرد. برای رویکرد نگاشت خودکار، خالق طرح تطبیقی و الگوریتم های تطبیق (نوع حریصانه و عقبگرد) در سمت سرور MBP در جاوا توسط Schneider بعنوان پایان نامه کارشناسی انجام شده.

برای ذخیره سازی نقشه، از پایگاه داده MongoDB استفاده می‌شود. بخش ۸-۱ معماری یکپارچه سازی تمام اجزای معماری را ارائه می‌دهد و نشان می‌دهد که چگونه آنها با هم تطبیق می‌یابند.

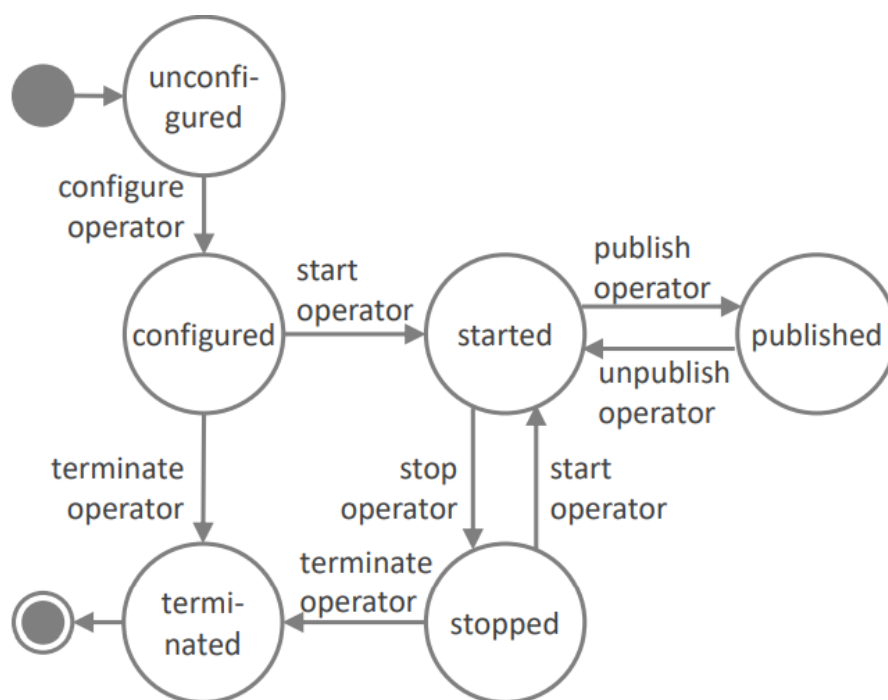
۶-۴ استقرار اپراتورها در محیط های IoT

این بخش شامل دو رویکرد برای تحقق استقرار است: (۱) یک رویکرد استقرار خودکار، که در آن اپراتورهای یک DSPM به طور خودکار بر اساس نقشه نقشه‌برداری ایجاد شده مستقر می‌شوند، و (ب) یک رویکرد استقرار نیمه خودکار، که همچنین بر اساس طرح نقشه برداری است، با این حال، اقدامات دستی، به نام وظایف انسانی پشتیبانی می‌شود.

۶-۴-۱ رویکرد خودکار

این رویکرد شامل نمایش اپراتور است که اپراتورهای یک DSPM را به اشیاء IoT مدل اختصاص می‌دهد. بر اساس نمایش اپراتور، امکان تکرار می‌باشد و هر یک از آنها به طور خودکار با استفاده از موتور استقرار مبتنی بر TOSCA بنام OpenTOSCA مستقر می‌شوند. هر اپراتور می‌تواند به چندین حالت برسد که نشان دهنده وضعیت فعلی استقرار یک اپراتور است. این حالات در ادامه توضیح داده شده است.

حالت های استقرار یک اپراتور: این پایان نامه شش حالت استقرار یک اپراتور را تعریف می‌کند: (i) پیکربندی نشده، (ii) پیکربندی، (iii) شروع شده، (iv) منتشر شده، (v) متوقف شده، (vi) پایان یافته. این حالت های عملگر در شکل ۱۰ نشان داده شده است.



شکل شماره ۱۰

وضعیت اولیه یک اپراتور (unconfigured) پیکربندی نشده است. پس از شروع استقرار، اولین کار پیکربندی شی IoT اختصاص داده شده برای اجرای اپراتور است. برای این کار، اپراتور در قالب نرم افزار طراحی شده به شی IoT اختصاص داده شده کپی می شود. علاوه بر این، وابستگی های نرم افزاری مورد نیاز (به عنوان مثال، کلاينت MQTT، مفسر پایتون) برای اپراتور بر روی شی IoT اختصاص یافته نصب و پیکربندی می شوند.

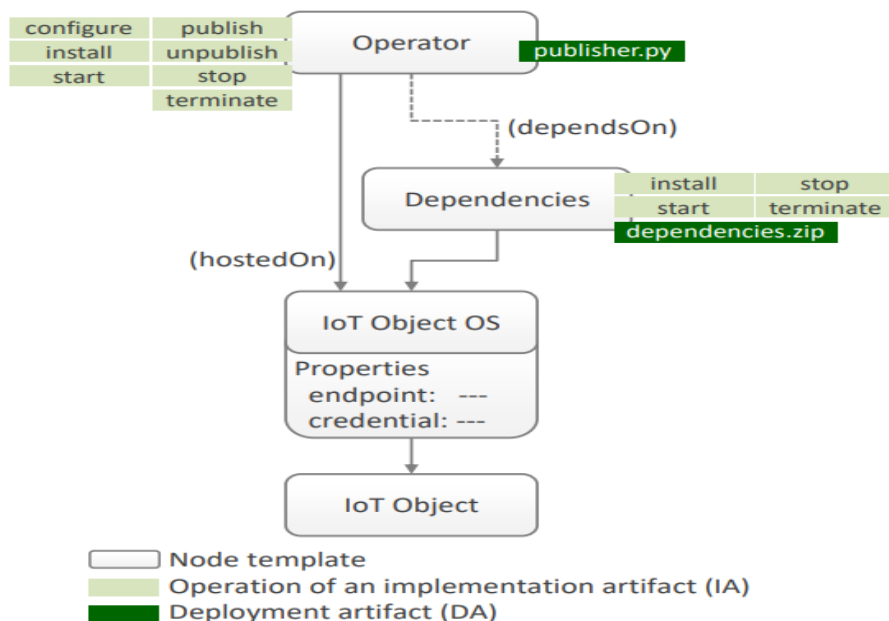
پس از پیکربندی، اپراتور به حالت پیکربندی شده (configured) منتقل می شود. یک اپراتور پیکربندی شده میتواند راه اندازی شود، که هدف اصلی استقرار خودکار است. هنگامی که همه اپراتورها در نقشه برداری اپراتور به حالت شروع رسیدند، استقرار خودکار به عنوان پایان یافته در نظر گرفته می شود و در نتیجه اپراتورهای برنامه IoT راه اندازی و اجرا می شوند. علاوه بر این، یک اپراتور پیکربندی شده را می توان مستقیماً به عنوان مکانیزم بازگشتی خاتمه داد در صورتی که اپراتور نمی تواند راه اندازی شود.

هنگامی که یک اپراتور راه اندازی می شود، خروجی آن مانند داده های حسگر استخراج شده، می تواند برای برنامه های خارجی در دسترس قرار گیرد. در این کار تحلیلگران تصمیم می گیرند آیا و کدام اپراتورها باید منتشر 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 گذاشته می شود، هر یک از اپراتورهای قبلاً مستقر شده باید توسط

تحلیلگران دامنه کنار گذاشته شوند. 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.

۴-۶-۲ استقرار اپراتور مبتنی بر TOSCA

شکل زیر یک مدل توپولوژی TOSCA را نشان می دهد



شکل شماره 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.

۴-۶-۳ رویکرد استقرار نیمه اتوماتیک

همانطور که قبلاً 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 می کنند که اشیاء IoT در محیط آن از قبل برای استقرار اپراتورها پیکربندی شده اند. 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. نمونه های بیشتر در زمینه محیط های اینترنت اشیا، وصل کردن حسگرها به دستگاه ها یا میکروکنترلرهای چشمک زن است. در نتیجه، وظیفه انسانی درخواستی از شخص برای انجام یک فعالیت خاص است.

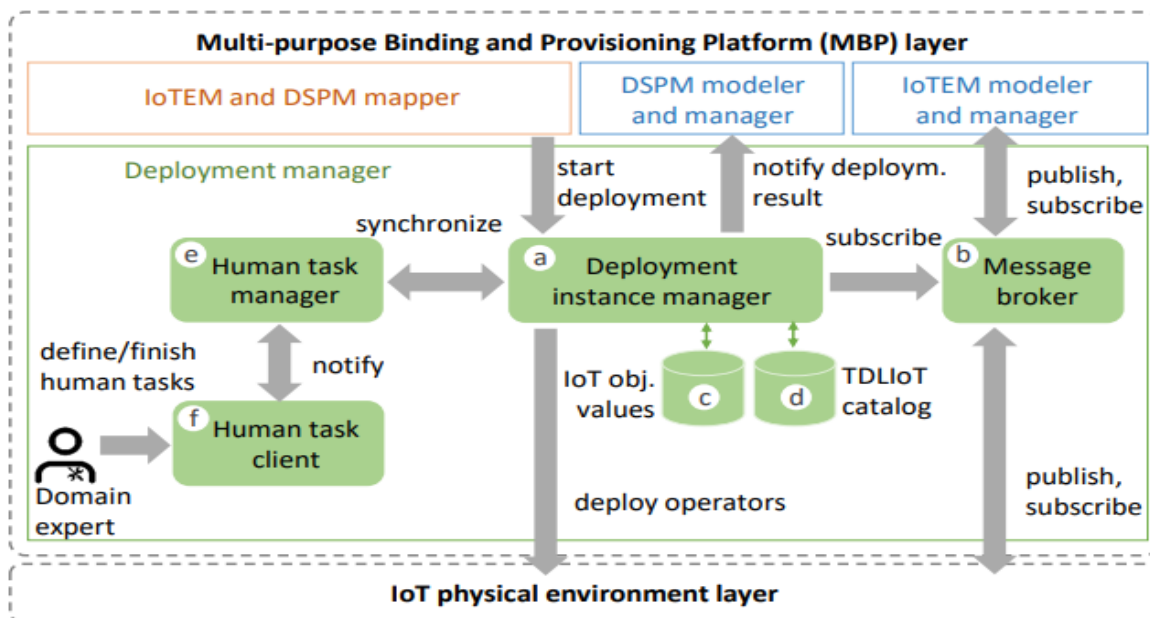
برای تطبیق بیشتر برای چالش های ارائه شده در محیط های IoT، این پایان نامه یک رویکرد استقرار نیمه خودکار را ارائه می دهد که در آن وظایف انسانی نیز برای استقرار در نظر گرفته می شود. برای این کار، این پایان نامه کارشناسان حوزه را قادر می سازد تا وظایف انسانی را تعریف کرده و قبل از شروع استقرار آنها را به 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 آدرس های IP، نیازی به مدل سازی در IoTEM ندارند و می توانند 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 عبارت 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.

۴-۷ جزئیات معماری و پیاده سازی مدیر استقرار

در معماری کلی بخش ۳-۳، بخش **Deployment 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 شوند. این نمونه های در حال اجرا به مؤلفه مدل ساز و مدیر **DSPM** (به بخش ۴-۲-۳ مراجعه کنید)، که از موفقیت آمیز بودن اطلاعات مطلع می شود، ارسال می شوند.



شکل ۱

در رویکرد استقرار خودکار، اپراتورها و نرم افزارهای طراحی شده مورد نیاز آنها بر روی اشیاء IoT نقشه برداری شده نصب، پیکربندی و شروع می شوند. هنگامی که اجرای اپراتورها بر روی اشیاء IoT آغاز می شود، 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 شی IoT و مقادیر نظارتی که در مخزن مقادیر شی IoT بخش c ذخیره می شوند، با Message broker مرتبط می شود. توضیحات نحوه دسترسی و تجزیه خروجی اپراتورهای منتشر شده در کاتالوگ TDLoT بخش d ذخیره می شود. این کاتالوگ همچنین یک REST API برای جستجوی اپراتورهای در دسترس عموم، به عنوان مثال، مقادیر حسگرهای عمومی را استخراج می کند، ارائه می دهد.

در رویکرد استقرار نیمه خودکار، بخش e اقدامات دستی (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 انسانی توسط 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 سازی شده است و همچنین از پیاده سازی های 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.

4-8 نظارت بر DSPM های مستقر شده

برای این منظور، ابزاری برای نظارت مستمر اشیاء IoT در محیط های IoT و dDSPMs به منظور شناسایی اختلالات فراهم می شود. این نظارت با استفاده از تکنیک های پردازش رویداد پیچیده (CEP)، که به خوبی برای پردازش جریان های داده برای تشخیص به موقع موقعیت ها تثبیت شده است، انجام می شود.

4-8-1 مدل سازی تشخیص اختلال

برنامه های کاربردی متن آگاه (Context-aware) در سال های اخیر توجه زیادی را به خود جلب کرده اند، به ویژه در حوزه اینترنت اشیا بخصوص که استخراج محتوای سطح بالا از داده های حسگر خام و سطح پایین، انطباق خودکار و تحقق محیط های اینترنت اشیا خودسازمان یافته را ممکن می سازد. با این حال، چالش های زیادی در رابطه با کسب، مدل سازی و مدیریت اطلاعات زمینه وجود دارد.

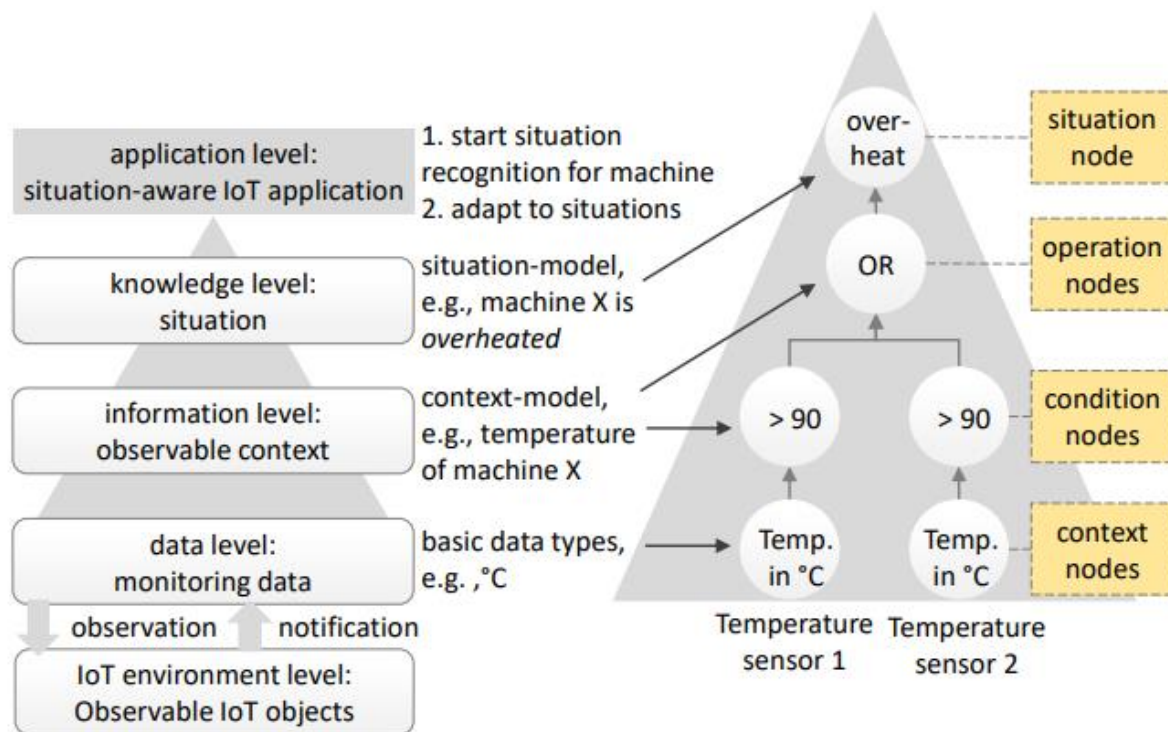
چالش دیگر این است که چگونه می توان به پردازش کارآمد مقادیر زیادی از داده های حسگر و در نتیجه کیفیت خوب دانش زمینه سطح بالا دست یافت. برای این کار، تکنیک های تثبیت شده ای مانند پردازش رویدادهای پیچیده (CEP)، برای پردازش حجم زیادی از داده ها لحظه ای معرفی شده اند.

این پایان نامه از تعریف ارائه شده توسط Dey و Abowd استفاده می کند و زمینه را به عنوان "هر اطلاعاتی که می تواند برای توصیف وضعیت یک موجودیت استفاده شود، که در آن موجودیت می تواند یک شخص، مکان یا شی باشد" 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 وضعیت در نظر گرفته می شود که به عنوان یک اتفاق که ممکن است نیاز به اقدامات اصلاحی داشته باشد تعریف می شود بنابراین در این پایان نامه از تشخیص اختلال با عنوان تشخیص موقعیت نیز یاد می شود.

هیرمر Hirmer و همکاران SitRS را پیشنهاد می کند، یک سرویس تشخیص موقعیت آماده برای ابر که تشخیص موقعیت را بر اساس داده های حسگر خام امکان پذیر می کند. در رویکرد آنها، سنسورها به صورت پویا محدود میشوند و داده های حسگر به روشی مبتنی بر کشش دریافت می شوند. تشخیص موقعیت در بازه های زمانی ثابت با کشیدن داده های حسگر و استخراج موقعیت ها اجرا می شود و بنابراین، تنها برای سناریوهای تشخیص موقعیت ساده مناسب است. SitRS XT، یک سرویس تشخیص موقعیت کارآمد و مقیاس پذیر می باشد و رویکرد هیرمر و همکاران را گسترش می دهد.

با فعال کردن پردازش مداوم داده های حسگر از طریق رویکرد مبتنی بر جریان با استفاده از فناوری های CEP. بدین ترتیب، مدل سازی و اجرای سناریوهای پیچیده تر تشخیص موقعیت فعال می شود. 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 شماره ۱۲

در سطح داده، فقط داده های نظارتی خام (به عنوان مثال، مقادیر سنسور دما) در دسترس است. این داده ها به سطح اطلاعات منتقل می شوند تا با اطلاعات مربوط به روابط داده های 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، داده های نظارت به منظور استخراج موقعیت ها جمع آوری و تفسیر می شوند که منجر به دانش در مورد محیط اینترنت اشیا می شود. 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، موقعیت ها به عنوان الگوهای موقعیت مدل سازی می شوند، یک مدل خاص دامنه که از جزئیات پیچیده و فنی انتزاع می کند. 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 SAT ها، حسگرها با گره های برگ به نام گره های زمینه مطابقت دارند و شاخه ها از پایین به بالا از طریق ترکیبی از به اصطلاح گره های شرطی و گره های عملیاتی تا رسیدن به گره ریشه (یعنی گره موقعیت) جمع می شوند. 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. گره های یک الگوی موقعیت، سطوح پردازش فوق الذکر را منعکس می کنند: گره های زمینه نشان دهنده حسگرهایی هستند که یک شی IoT خاص را نظارت می کنند 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. گره های شرطی را می توان توسط گره های عملیاتی با استفاده از عملیات منطقی جمع آوری کرد تا زمانی که به گره موقعیت رسید، که نشان دهنده وضعیتی است که باید شناسایی شود.

ترکیب گره های شرط، عملیات و موقعیت با سطح دانش مطابقت دارد، جایی که داده های حسگر تجمع، تفسیر و به موقعیت ها مشتق می شوند. 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 قلب های موقعیت به نمایش های اجرایی مورد نیاز، یعنی پرس و جوهای CEP، کاهش می دهد. 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).

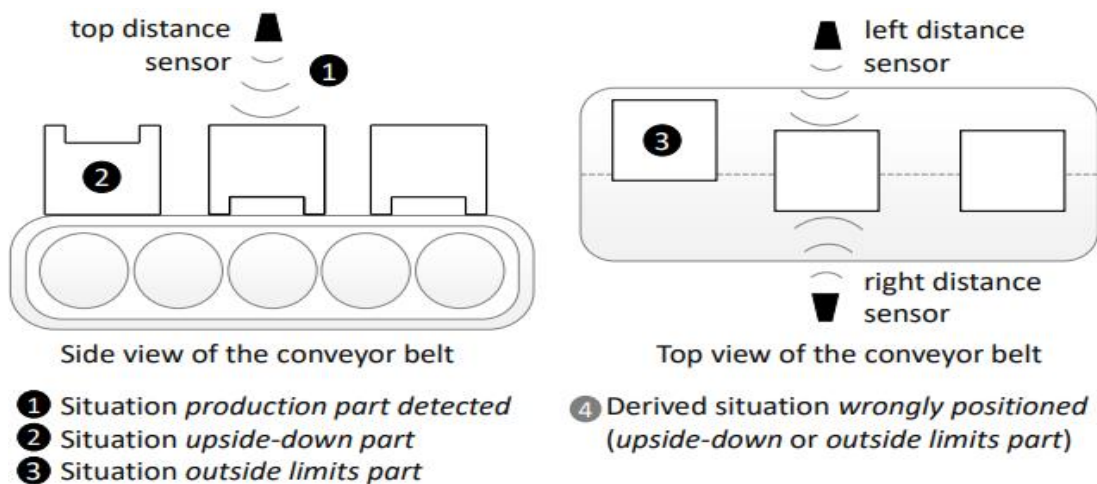
هدف از این سناریوی موردی، نظارت بر یک مرحله تولید در کف کارگاه یک شرکت تولیدی است تا در اسرع وقت اختلالات در مرحله تولید شناسایی شود. در این مرحله تولید، یک کارگر کف مغازه قطعه فلزی را وارد دستگاه می کند که آن را به شکل مورد نیاز برش می دهد. در این فرآیند، دو مشکل ممکن است رخ دهد: (۱) ابزار

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 برمی گرداند، 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 درخواست های CEP را اجرا می کند، باید توزیع شود

۴-۸-۲ اجرای تشخیص اختلال

این بخش نحوه اجرای تشخیص اختلال را بر اساس مفاهیم توضیح داده شده در بخش ۷-۱ توضیح می دهد. برای این کار، موقعیت هایی که قبلاً به عنوان الگوهای موقعیت ها مدل سازی شده اند، ابتدا به نمایش های اجرایی و مبتنی بر رویداد، یعنی پرس و جوهای CEP تبدیل می شوند. سپس این نمایش ها توسط موتورهای CEP اجرا می شوند، که داده های نظارتی ورودی را پردازش می کنند و زهانی که اختلالات بر اساس نمایش های اجرایی شناسایی می شوند، اعلان هایی ایجاد می کنند.

در شکل ۱۵، نمونه ای انتزاعی از سناریوهای تولید در دنیای واقعی نشان داده شده است که اختلالات در فرآیندهای تولید را به موقع تشخیص می دهد. در این مثال، قطعات تولیدی روی یک تسمه نقاله نظارت می شوند تا تشخیص داده شود که چه زمانی اشتباه قرار گرفته اند.

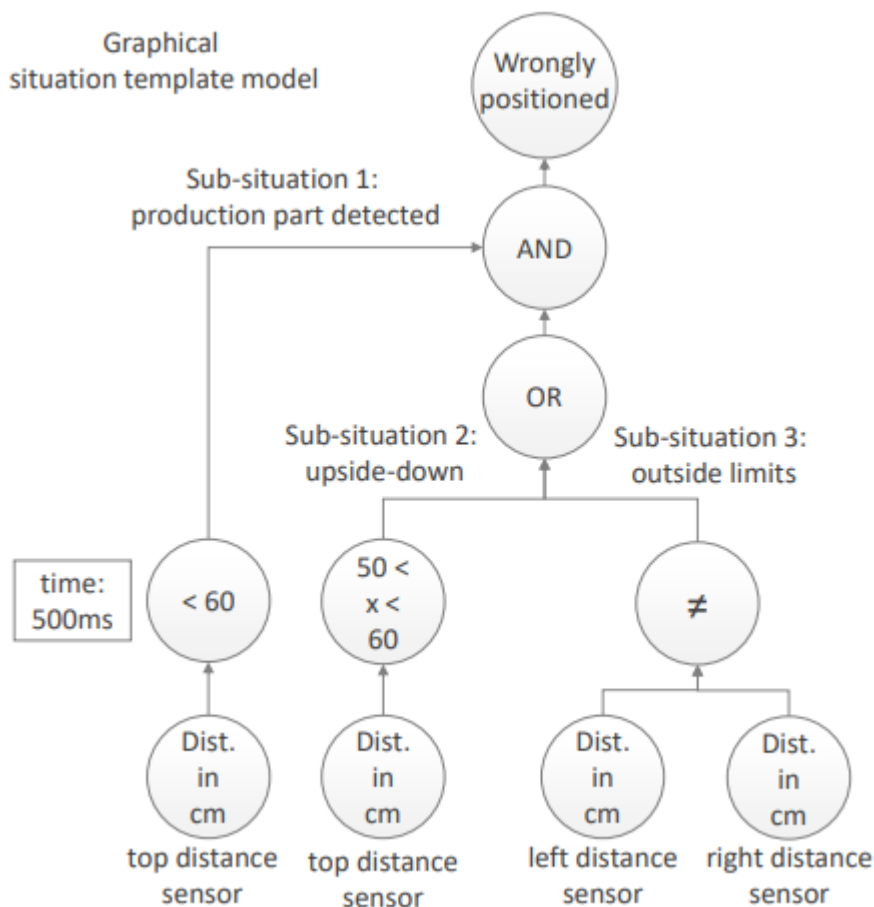


شکل شماره ۱۳

تشخیص اختلال بر اساس 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. وضعیت تولید : نشان دهنده وجود یک قطعه تولیدی بر روی تسمه 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. وضعیت خارج از محدوده نشان می دهد که قسمت تولید خارج از حدود مجاز به چپ یا راست 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 شکل 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 شماره ۱۴

برای تبدیل خودکار یک الگوی موقعیت به یک پرس و جو CEP، گره های الگوی موقعیت به منظور فرموله کردن یک الگوی رویداد پیچیده، که از عبارات الگوی ترکیب شده 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 ساخته می شود که توسط گره های عملیاتی تجمیع می شوند. هر گره شرط 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 می شوند. 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. 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 جوی پیچیده CEP خود را می شوند. در نهایت، تشخیص اغتشاش با استقرار جستجوهای CEP حاصل در موتور اجرای CEP آغاز می شود.

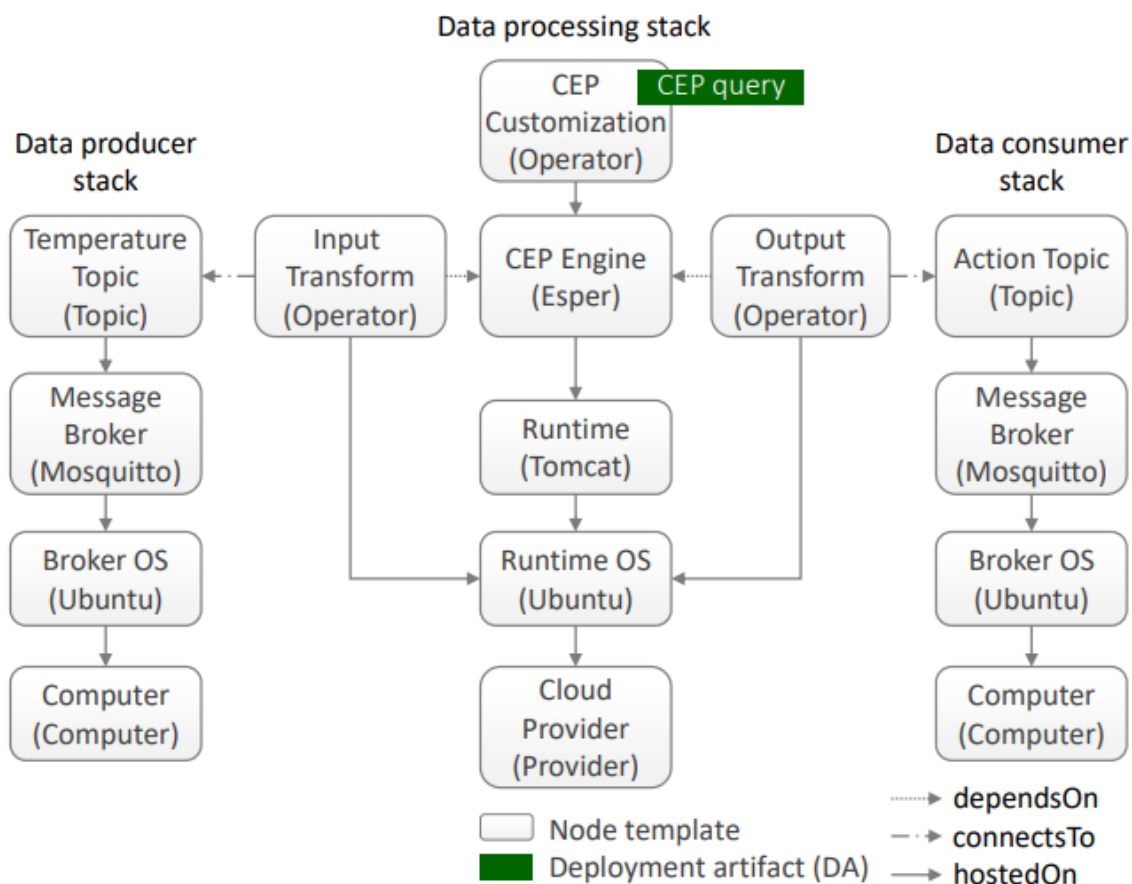
این پایان نامه کارشناسی ارشد یک زبان پرس و جو انتزاعی را ارائه می دهد که ویژگی های مشترک CEP را در بین موتورهای مختلف CEP بررسی شده بیان می کند، در نتیجه، این پایان نامه علاوه بر استفاده از الگوهای موقعیت برای تشخیص موقعیت، مدل سازی با استفاده از این زبان پرس و جو انتزاعی را نیز امکان پذیر می کند. شبیه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

نرم افزارهایی که قابلیت های CEP را ارائه می کنند، 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 Vagrant, Ansible, Amazon AWS, Docker.

با این حال، رویکردهای عمومی که امکان ارائه موتورهای استاندارد CEP را فراهم می کنند، فاقد سفارشی سازی مورد نیاز حوزه اینترنت اشیا هستند. به طور دقیق تر، محیط های اینترنت اشیا از نظر اشیاء، برنامه های کاربردی و درخواست های پیوسته برای پردازش داده ها، 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 سفارشی سازی و ارائه موتورهای CEP، از جمله محیط های IoT، اپراتورهای مورد نیاز و جستارهای CEP ارائه می کنیم. این رویکرد منجر به کاهش زیادی در تلاش سفارشی سازی هنگام استفاده از CEP در محیط های IoT می شود. علاوه بر این، چنین رویکرد سفارشی سازی شده، امنیت داده ها را از طریق ایجاد نمونه های ثبات و غیرقابل تغییر موتورهای CEP، مشلبه نماها در پایگاه های داده، افزایش می دهد. 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 شماره ۱۵

در این مثال، یک سناریوی ساده شده سیستم HVAC مدل سازی می شود، که در آن دمای یک اتاق به طور مداوم نظارت می شود تا زمانی که دما از منطقه آسایش حرارتی فراتر می رود یا کمتر می شود، واکنش نشان دهد. سفارشی سازی و تهیه موتورهای CEP حداقل به اجزای زیر نیاز دارد:

(۱) یک موتور CEP برای پردازش و جمع آوری داده های ورودی، (۲) جستارهای CEP در یک زبان پرس و جو سازگار، که نحوه پردازش داده های ورودی را برای محاسبه بالاتر توسط موتور CEP تعریف می کند. - موقعیت های سطح بر اساس داده های سطح پایین، (۳) تولیدکنندگان داده، که داده های سطح پایین را به موتور CEP ارائه می کنند، و (۴) مصرف کنندگان داده، که موقعیت های سطح بالاتر به دست آمده توسط موتور CEP را دریافت و پردازش می کنند.

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 شده‌اند. نمونه‌ای 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 می‌کند.

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 ماشین‌های مجازی را ایجاد و اجرا می‌کند. علاوه بر این، قالب گره سیستم عامل Runtime سیستم عامل ماشین مجازی را مدل می‌کند، در حالی که الگوی گره Runtime زمان اجرا لازم را مدل می‌کند. G

یک وب سرور برای موتور CEP که در آن مستقر می‌شود. 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، قالب‌های 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 ارائه دهنده 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 نصب کننده‌های موتور باشد. اگر این DA ارائه نشده باشد، این باینری‌ها باید به صورت پویا توسط IA بازیابی شوند.

هنگامی که موتور CEP راه‌اندازی شد، سفارشی سازی موتور CEP توسط الگوی گره سفارشی سازی CEP، که انواع رویداد را تعریف می‌کند، تحقق می‌یابد. مانند، رویدادهای داده ورودی و خروجی. به‌طور دقیق‌تر، داده‌ها بر اساس الگوی گره موضوعی مدل‌سازی شده ارائه یا مصرف می‌شوند. مثلاً 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 شوند، نشان داده می شوند. علاوه بر این، الگوی گره سفارشی‌سازی CEP حاوی ویژگی‌هایی است که سفارشی‌سازی‌های اضافی خاص سیستم، مانند (غیرفعال‌سازی) بهینه‌سازی‌های پرس و جو را تعریف می‌کنند.

برای اتصال پشته تولیدکننده داده به موتور CEP از طریق الگوی انتشار اشتراک، 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 فشار دهد. داده ها را از طریق رابط ارائه شده و نقطه پایانی API به موتور CEP تبدیل کرد. برای آن، قالب گره تبدیل ورودی ارائه شده است که شامل چنین عملگر به عنوان یک مصنوع توسعه است. این الگوی گره به الگوی گره موضوع دما وصل می‌شود و به الگوی گره موتور CEP بستگی 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 سازی است که این مصنوعات استقرار را در زمان اجرا مربوطه مستقر می کند. در مثال نشان داده شده، این زمان اجرای موتور CEP است.

با این حال، اگر قالب گره تبدیل ورودی به نوع متفاوتی از زمان اجرا نیاز داشته باشد، می‌توان یک الگوی گره اضافی مدل‌سازی 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) وضعیت حاصل را که توسط موتور CEP بر اساس جستارهای CEP محاسبه شده است، 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 شده است که شامل چنین عملگر به عنوان یک مصنوع توسعه است. این الگوی گره همچنین ممکن است به چندین الگوی گره موضوعی (به عنوان مثال، موضوع اقدام) متصل شود 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.

شبهه به قالب گره تبدیل ورودی، تبدیل خروجی همچنین حاوی یک مصنوع پیاده سازی مربوطه است که استقرار این اپراتور را در زمان اجرا مناسب انجام می دهد. مدل سازی موتور CEP به عنوان یک سرویس ارائه شده توسط یک ارائه دهنده خارجی کمی با مدل در شکل ۱۷ متفاوت است.

قالب‌های گره سیستم عامل Runtime و Runtime نیازی به مدل‌سازی ندارند، با این حال، زیرساخت‌های قالب‌های گره تبدیل ورودی و تبدیل خروجی هنوز باید فراهم شود. مانند قبل، آنها می‌توانند با استفاده از نقطه پایانی API قالب گره CEP Engine با موتور CEP ارتباط برقرار کنند.

به طور خلاصه، برای ایجاد پشته پردازش داده در مدل توپولوژی TOSCA، حاوی موتور سفارشی CEP، اجزای ذکر شده در بالا باید مدل‌سازی شده و همانطور که توضیح داده شد، متصل شوند. پشته مصرف کننده داده به روشی مشابه پشته تولیدکننده داده کار می‌کند، زیرا بر اساس الگوی انتشار-اشتراک نیز ساخته می‌شود.

مصرف‌کنندگان داده می‌توانند، برای مثال، برنامه‌هایی باشند که در موقعیت‌های محاسبه‌شده توسط موتور CEP واکنش نشان می‌دهند. این پشته شامل یک دستگاه IoT، یک محرک و علاوه بر این، زیرساختی متشکل از یک واسطه پیام است که از طریق آن می‌توان محرک را کنترل کرد. به عنوان مثال، یک محرک هواکش به Raspberry Pi وصل شده است.

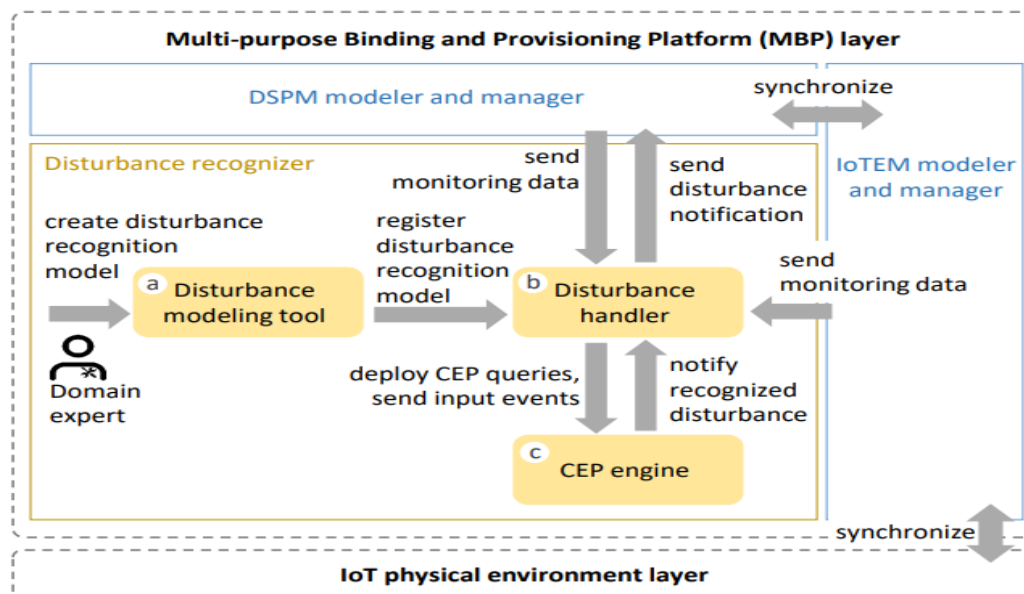
در این دستگاه اینترنت اشیا، یک اپراتور کنترل میزبانی می‌شود، که مشترک موضوع اقدام خاص یک واسطه پیام برای استخراج موتورهای CEP می‌شود و یکی دیگر از ویژگی‌های مهم را فعال می‌کند: امنیت داده‌ها. مشابه مفهوم نماهای فقط خواندنی در سیستم‌های مدیریت پایگاه داده رابطه‌ای، این رویکرد قادر است درجه قرار گرفتن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

این را می‌توان با ممنوع کردن پرس و جو و داده‌های اضافی در موتور CEP و همچنین خروجی اضافی از طریق کیسوله 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، به عنوان مثال، ساختار ملموس پرس و جو CEP پنهان می‌شود.

۴-۹ جزئیات معماری و پیاده سازی تشخیص اختلال

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



شکل شماره ۱۶

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، کنترل‌کننده اختلال داده‌های نظارتی را در مورد محیط اینترنت اشیا و اپراتورهای پردازش دریافت می‌کند. این داده‌های نظارتی به رویدادهای ورودی CEP تبدیل شده و به موتور CEP ارسال می‌شوند.

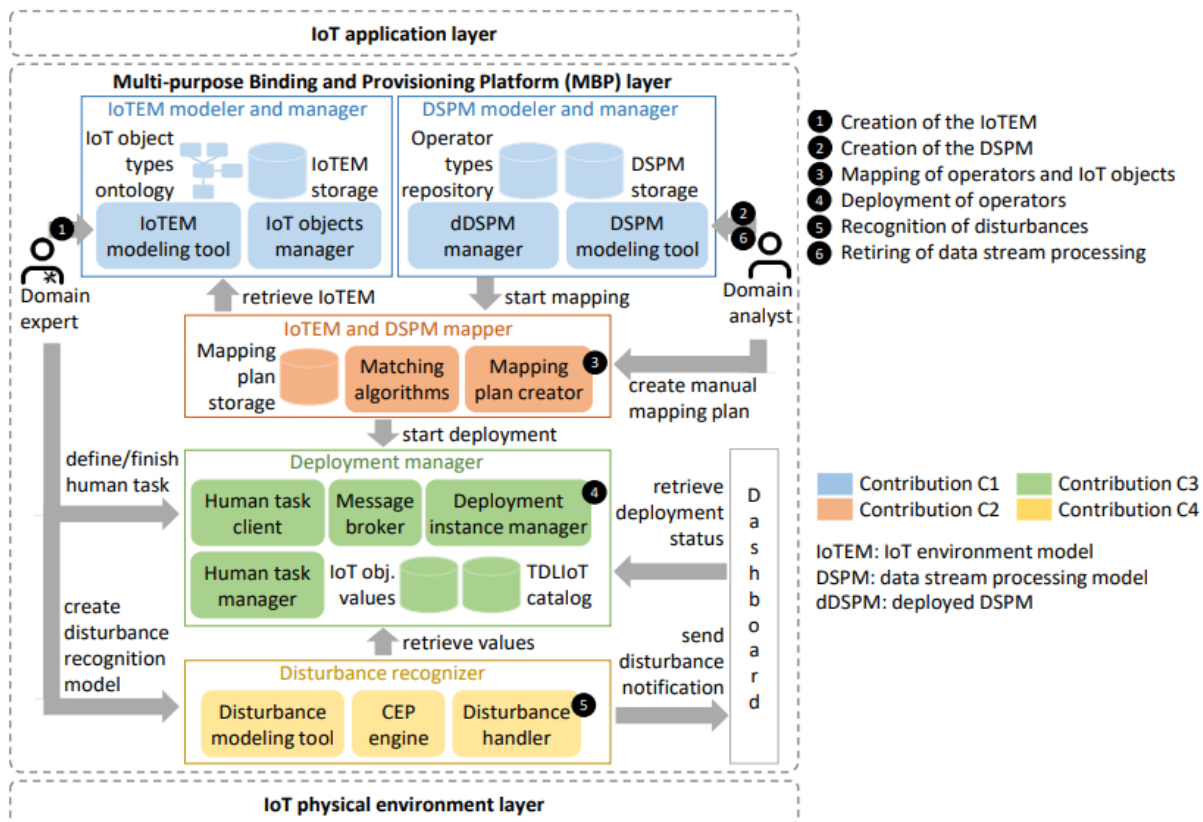
هنگامی که موتور CEP یک اختلال را تشخیص داد، به کنترل‌کننده اختلال اطلاع می‌دهد، که اعلان‌های مربوط به اختلالات را به مؤلفه مدل‌ساز و مدیر DSPM (به بخش ۴-۲-۳ مراجعه کنید) و به داشبورد MBP ارسال می‌کند. جزء تشخیص اختلال به طور نمونه اولیه به عنوان بخشی از MBP پیاده‌سازی شده است.

بخش ۸-۲ مروری بر عملکردهای MBP می‌دهد. ابزار مدل‌سازی Disturbance1 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 engine1 پردازش شوند. کنترل‌کننده Disturbance در جاوا در سمت سرور MBP پیاده‌سازی شده است. بخش ۸-۱ معماری یکپارچه‌سازی تمام اجزای معماری را ارائه می‌دهد و نشان می‌دهد که چگونه آنها با هم تطبیق می‌یابند.

۴-۱۰ ارزیابی

این بخش پلتفرم اتصال و تدارک چند منظوره (MBP) را توصیف و ارزیابی می‌کند، که به عنوان اثبات مفهوم مشارکت‌های این پایان‌نامه توسعه یافته است. MBP یک پلتفرم IoT منبع باز است و توضیح می‌دهد که چگونه می‌توان از آن برای تحقق یک سناریوی ساده اینترنت اشیا برای دامنه دفتر هوشمند استفاده کرد.

شکل ۱۹ معماری دقیق حاصل از این پایان‌نامه را به تصویر می‌کشد، که اجزای هر دستاورد را ادغام می‌کند و نشان می‌دهد که چگونه آنها با هم تطبیق می‌یابند. دستاورد ها با رنگ برجسته می‌شوند و مراحل رویکرد روشمند نیز نشان داده شده است.



شکل شماره ۱۷

اجزای معماری و پیاده سازی های مربوط به هر دستاورد به ترتیب در فصل های ۴ تا ۷ به تفصیل توضیح داده شده است. در معماری کلی (مراجعه کنید به بخش ۳-۳)، دستاورد C1 شامل دو جزء معماری، مدل ساز و مدیر IoTEM (به بخش ۳-۱-۴ مراجعه کنید)، و مدل ساز و مدیر DSPM (به بخش ۳-۲-۴ مراجعه کنید) را شامل می‌شود. این اجزا بیشتر به اجزای کوچکتر تقسیم می‌شوند 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.

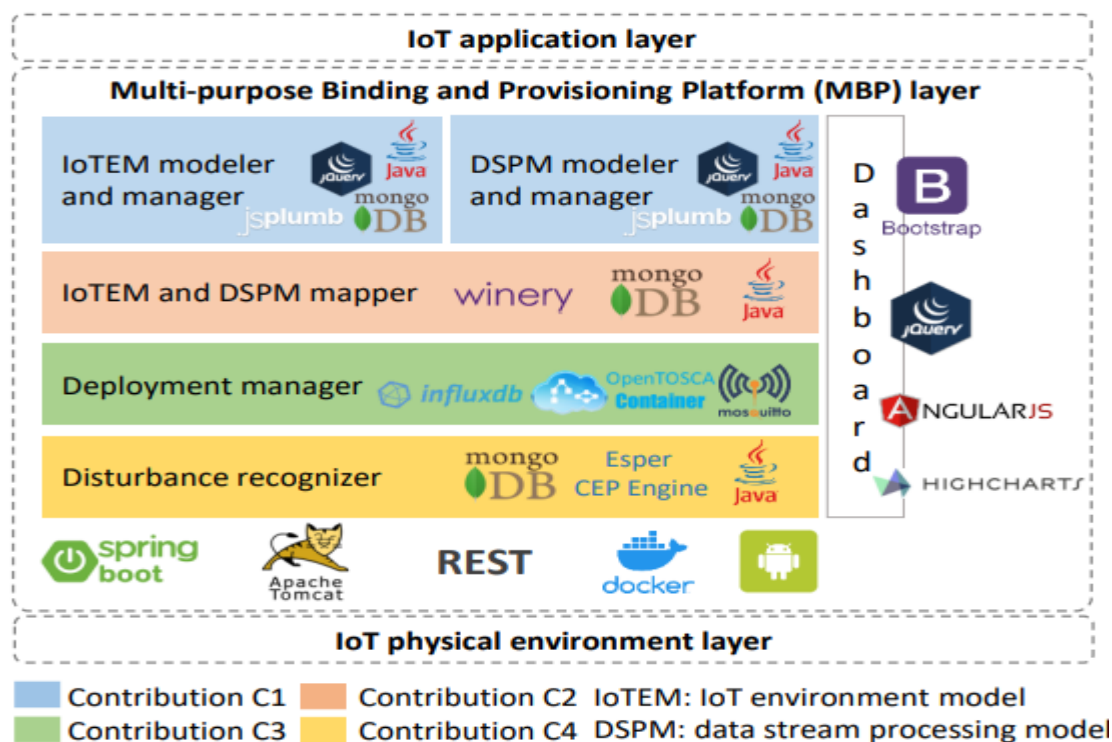
بخش مدل ساز و مدیر DSPM با ارائه ابزاری برای ایجاد و مدیریت DSPM (مرحله ۲)، و بعداً برای توقف DSPM مستقر در محیط اینترنت اشیا (مرحله ۶) از تحلیلگران دامنه پشتیبانی می کند. دستاورد C2 شامل اجزای معماری IoTEM و DSPM mapper است که از تحلیلگران دامنه در تصمیم گیری درباره محل استقرار اپراتورهای پردازش پشتیبانی می کند.

این جزء معماری بیشتر به اجزای کوچکتر تقسیم می شود و در شکل 19 به رنگ نارنجی نشان داده شده است. این جزء با اجزای معماری مشارکت C1 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. Contribution C3 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 C2 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 C4 شامل شناسایی کننده اختلال جزء معماری است که با رنگ زرد در شکل 19 نشان داده شده است. این مؤلفه ابزاری برای نظارت بر اشیاء اینترنت اشیا و DSPM های مستقر شده به منظور شناسایی اختلالات در طول پردازش داده ها (مرحله ۵) فراهم می کند. این ابزار یک ابزار مدل سازی اختلال را فراهم می کند که کارشناسان حوزه می توانند آن را مدل سازی 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 C3 بازیابی می کند. این مقادیر به موتور CEP ارسال می شوند، جایی که بر اساس مدل های تشخیص اختلال پردازش می شوند. در صورت شناسایی اختلالات، اعلان های مربوط به اختلالات به داشبورد ارسال می شود، جایی که می توان آن ها 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 شماره ۱۸

علاوه بر این، MBP را می توان به عنوان یک ظرف Docker اجرا کرد. پس از نصب، رابط کاربری از طریق یک مرورگر اینترنت (به عنوان مثال، کروم یا فایرفاکس) قابل دسترسی است. رابط کاربری MBP بر اساس الگوهای بوت استرپ است و از چارچوب جاوا اسکریپت Angular JS استفاده می کند. نمادها از کتابخانه متریال طراحی گوگل استخراج می شوند. علاوه بر این، Highcharts کتابخانه در داشبورد برای تجسم مقادیر خروجی زنده و تاریخی اپراتورهای مستقر و نمایش داده های نظارتی اشیاء اینترنت اشیا 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 مدل ساز و مدیر IoT، ابزار مدل سازی IoT در جاوا اسکریپت پیاده سازی شده است و از کتابخانه JavaScript jQuery استفاده می کند. jsPlumb Toolkit برای پیاده سازی یک ویرایشگر کشیدن و رها کردن برای مدل سازی محیط های IoT استفاده شده است. مدیر شی IoT در سرور MBP با استفاده از جاوا پیاده سازی می شود. پایگاه داده MongoDB برای ذخیره انواع شی IoT و IoT های استفاده می شود.

در مورد مؤلفه مدل ساز و مدیر DSPM، ابزار FlexMash به عنوان ابزار مدل سازی DSPM استفاده می شود که توسط Hirmer و همکاران توسعه داده شده است. در محدوده این پایان نامه، رابط گرافیکی FlexMash و مدل زیربنایی مبتنی بر JSON در پایان نامه کارشناسی ارشد انجام شده توسط Chaudhry گسترش یافت. برنامه های افزودنی شامل امکان حاشیه نویسی الزامات در اپراتورهای پردازش و لبه های بین اپراتورها است. علاوه بر این، FlexMash برای اتصال به MBP و امکان وارد کردن منابع و سینک ها از IoT گسترش یافت. رابط گرافیکی FlexMash در جاوا پیاده سازی شده و از jQuery و jsPlumb استفاده می کند.

DSPM های مدل شده در MBP در پایگاه داده MongoDB ذخیره می شوند. مدیر dDSPM با استفاده از جاوا در سرور MBP پیاده سازی می شود. مخزن انواع اپراتور مربوط به یک سیستم فایل است که در محل نصب MBP پیکربندی شده است. چندین عملگر در Python و Shell پیاده سازی شده اند که می توان آنها را در پروژه MBP GitHub¹ یافت. در مورد مؤلفه نقشه بردار IoT، DSPM و Winery به عنوان ایجادکننده طرح نقشه برداری خارجی برای رویکرد نقشه برداری دستی استفاده می شود.

در این مورد، نقشه های تطبیق بر اساس استاندارد TOSCA است و در یک سیستم فایل پیکربندی شده برای Winery ذخیره می شود. برای رویکرد نگاشت خودکار، خالق طرح نقشه برداری و دو الگوریتم تطبیق، یک نوع حریص و یک نوع عقبگرد، در سرور MBP در جاوا بصورت پایان نامه کارشناسی توسط شنایدر پیاده سازی شدند. در این مورد، نقشه های تطبیقی به طور خودکار تولید شده در پایگاه داده MongoDB ذخیره می شوند.

در مورد مولفه Deployment Manager، کانتینر OpenTOSCA به عنوان یک مدیر نمونه استقرار خارجی استفاده می شود که قادر است به طور خودکار اپراتورهای نقشه بر اساس استاندارد TOSCA را مستقر کند.

در محدوده این پایان نامه، یک پایان نامه کارشناسی توسط Kutger انجام شد 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 انسانی مبتنی بر پیاده سازی Wagner است که مربوط به یک برنامه وب جاوا است 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 سپس می‌توانند در MBP ثبت شوند و توسط MBP بر روی اشیاء IoT مستقر شوند. هنگامی که اپراتورها مستقر شدند، با اتصال و انتشار به واسطه پیام، که با استفاده از کارگزار MQTT Mosquitto پیاده سازی می‌شود، مقادیر را به MBP ارسال می‌کنند. سپس MBP می‌تواند مقادیر حسگر را به صورت نمودارهای خطی در داشبورد تجسم کند. مدیریت و پردازش داده در MBP یک معماری مبتنی بر لامبدا را پیاده‌سازی می‌کند که در آن داده‌های زنده و تاریخی را می‌توان مدیریت و پردازش کرد. برای ذخیره داده‌های اندازه‌گیری، یعنی داده‌های حسگر و داده‌های مبتنی بر مهر زمانی، از پایگاه داده سری زمانی InfluxDB استفاده می‌شود. فراداده‌های بیشتر، مدل‌های محیط اینترنت اشیا و مدل‌های پردازش جریان داده در پایگاه‌های داده NoSQL MongoDB ذخیره می‌شوند تا داده‌های اندازه‌گیری و ابرداده مربوطه را به وضوح 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 سازی شده است و از پایگاه داده MongoDB برای ذخیره توضیحات TDLIoT استفاده می‌کند. این یک رابط وب پیاده سازی شده در جاوا اسکریپت و یک API REST برای تعامل با کاتالوگ TDLIoT فراهم می‌کند. 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، مدل‌های تشخیص اختلال را به جستارهای CEP تبدیل می‌کند، که می‌تواند توسط موتور Esper CEP پردازش شود. کنترل کننده Disturbance در جاوا پیاده سازی شده است. در محدوده این پایان نامه، MBP2Go، یک برنامه مشتری موبایل مبتنی بر اندروید که در جاوا برای اتصال به MBP پیاده سازی شده است، توسعه یافته است. MBP2Go یک نمای کلی از اشیاء ثبت شده اینترنت اشیا در MBP ارائه می‌دهد. می‌تواند اشیاء IoT را ثبت و حذف کند و مقادیر حسگر فعلی و تاریخی را تجسم کند.

۴-۱۱ ملاحظات بیشتر

در سال ۲۰۱۸، معماری مرجع OpenFog به عنوان یک استاندارد رسمی برای محاسبات ابری پذیرفته شد. این استاندارد که با نام IEEE 1934-2018 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 مکانیسم های احراز هویت، مانند رمز عبور، اعتبار دیجیتال یا گواهی ها تأیید می شوند. علاوه بر این، تبادل داده بین اشیاء IoT و MBP توسط مکانیسم های امنیتی به خوبی تثبیت شده مانند HTTPS یا OAuth2 پشتیبانی می شود. 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 اصل مقیاس پذیری بهره مندی از فرصت های مقیاس پذیری ارائه شده در محیط های IoT است. این فرصت ها در ابعاد مختلفی مانند عملکرد مقیاس پذیر، ظرفیت مقیاس پذیر و نرم افزار مقیاس پذیر وجود دارند. عملکرد مقیاس پذیر به عملکرد مورد نیاز برنامه های اینترنت اشیا اشاره دارد، که باید با تأخیر کمی بین اندازه گیری های حسگر و فرمان های محرک در نتیجه ارائه شود، حتی زمانی که تعداد اشیاء متصل به پلتفرم IoT افزایش می یابد. برای این کار، MBP از فناوری های CEP استفاده می کند، که قادر به پردازش مداوم مقادیر زیادی از داده ها و پردازش کارآمد و به موقع داده های حسگر، به طوری که دستورات محرک را می توان در اسرع وقت صادر کرد.

ظرفیت مقیاس پذیر به توانایی اشیاء IoT برای افزودن (یا حذف) به محیط های اینترنت اشیا اشاره دارد. MBP مکانیسم هایی را برای به روزرسانی محیط های IoT با افزودن، به روزرسانی یا حذف اشیاء ارائه می کند، که هم شامل اشیاء سخت افزاری (مانند دستگاه ها، حسگرها، محرک ها) و هم اشیاء مجازی (مانند ماشین های مجازی) می شود.

در نهایت، نرم افزار مقیاس پذیر، در میان جنبه های دیگر، به مقیاس پذیری زیرساخت مدیریت یک پلتفرم IoT اشاره دارد، یعنی توانایی آن برای مدیریت بسیاری از اشیاء مطابق با سناریوهای IoT. با توجه به معماری ماژولار MBP و استفاده از اجزای نرم افزاری مقیاس پذیر (مانند پایگاه های داده مقیاس) همانطور که در بخش ۸-۱ توضیح داده شده است، MBP همچنین می تواند این بعد مقیاس پذیری را ارائه دهد.

MBP به عنوان یک پروژه منبع باز GitHub در دسترس است و بنابراین، نه اختصاصی است و نه تک فروشنده. علاوه بر این، MBP بر اساس استانداردهای ایجاد شده، 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 اتصال پویا، اپراتورهای هر نوع شی IoT را می توان ایجاد و در MBP ثبت کرد. MBP چندین عملکرد آماده استخراج و کنترل را در قالب اسکریپت ارائه می دهد، اما همچنین به کاربران امکان می دهد تا اپراتورهای خود را به هر زبان برنامه نویسی دلخواه ارائه دهند.

اصل خودمختاری به استقلال تصمیم گیری اشیاء اینترنت اشیا پس از وقوع خرابی اشاره دارد. MBP ابزاری را برای استقرار اپراتورها بر روی اشیاء IoT فراهم می کند، به طوری که درجه ای از استقلال اولیه اشیاء 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 عمدتاً توسط خود MBP انجام می شود. با این وجود، هنوز رویکردهای پیچیده تری برای دستیابی به اهداف این اصل مورد نیاز است و بنابراین، بخشی از کارهای آینده بر اساس مفاهیم این پایان نامه خواهد بود.

هدف اصل برنامه پذیری دستیابی به یک استقرار بسیار تطبیقی است به طوری که دادن وظایف جدید (یعنی اپراتورها) به یک شی IoT به صورت خودکار انجام می شود. این اصل در MBP از طریق مکانیابی اپراتور و مفاهیم استقرار پویا به دست می آید، که در آن یک اپراتور می تواند به طور خودکار به یک شی IoT دیگر مستقر شود.

هدف اصل 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 تصمیم 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 زود هنگام اختلالات در محیط های IoT و علاوه بر این، انجام اقدامات اصلاحی در اسرع وقت ارائه می دهد. مثال هایی برای چنین رویکردهایی ارائه اطلاعات در سطح بالا از طریق داشبورد، ابزار مدل سازی اختلال برای قوانین تعریف شده توسط کاربر، که واکنش خودکار مبتنی بر قانون را در تشخیص اختلال، و پردازش به موقع داده ها با استفاده از فناوری های CEP را امکان پذیر می سازد، است.

برای تحقق یک نظارت خودکار بر اساس داده های حسگر و فعال کردن محرک ها بر اساس اختلالات شناسایی شده، MBP ابزاری را برای تشخیص اختلالات بر اساس قوانین تعریف شده توسط کاربر فراهم می کند. چنین قوانینی، که بر اساس الگوی رویداد-شرط-عمل هستند، می توانند در ابزار مدل سازی اختلال MBP تعریف شوند. هدف اصل سلسله مراتب معماری های سلسله مراتبی متشکل از چندین لایه است که در آن هر لایه به مشکلات خاصی در سناریوهای 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 سطح متفاوتی از انتزاع مطابقت دارد. برای مثال، تشخیص دهنده اختلال داده های حسگر سطح پایین را پردازش می کند، در حالی که مدل ساز و مدیر DSPM با منطق برنامه های سطح بالا سر و کار دارد.

در نهایت، اصل RAS (قابلیت اطمینان، در دسترس بودن، قابلیت سرویس دهی) به سه جنبه تقسیم می شود. قابلیت اطمینان به عنوان توانایی پلت فرم اینترنت اشیا برای ارائه عملکردهای طراحی شده حتی در شرایط نامطلوب، به عنوان مثال، زمانی که یک شی اینترنت اشیا معیوب می شود، تعریف می شود. در دسترس بودن به مدیریت مستمر و هماهنگ سازی محیط اینترنت اشیا اشاره دارد.

قابلیت سرویس دهی به عملکرد صحیح برنامه های IoT در محیط IoT، به عنوان مثال، از طریق استقرار 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 دسترس بودن، MBP محیط های اینترنت اشیا و اپراتورهای مستقر را به طور مداوم مدیریت می کند (ر.ک. فصل ۴) و نظارت می کند. در نهایت، فصل های ۵ و ۶ قابلیت سرویس دهی را از طریق چندین مفهوم برای استقرار و استقرار مجدد اپراتورها در محیط های اینترنت اشیا پشتیبانی می کنند.

۴-۱۲ پاسخ به سوالات تحقیق

برای درک تاثیر معنایی برنامه، باید پاسخ هایی به سوالات مطرح شده در فصل اول داشته باشیم. :

۱. چگونه باید پردازش مناسب برای انواع جریان داده در محیط IoT 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 در محیط IoT را طراحی، اجرا و کارایی این مدل را تضمین کرد؟
- جواب سوال ۱-** برای پردازش مناسب جریان داده باید بتوان بخشی از عملیات را گره های ایجاد کننده اطلاعات انجام دهیم و در بخش هایی اپراتورهایی برای پردازش اولیه ایجاد نماییم تا در نهایت نتیجه پردازش ها به سرور مرکزی برای عملیاتهای نهایی ارسال شود.

جواب سوال ۲ - برای یک ارتباط مناسب در محیط های ناهمگن و پویا بهترین روش یک سامانه ابری می باشد که در آن تمامی اپراتورها حاصل پردازش خود را به یک سرور ابری متمرکز ارسال می کند و از سوی دیگر باید اشیاء را به صورت گره های IoTEm دید که توسط نرم افزارها و API ها در ارتباط می باشند.

جواب سوال ۳- یک مدل مناسب باید دارای امکانات ابری با قابلیت افزودن انواع اشیاء بدون پی‌کردن مجدد باشد و مداوم بر عملکرد اشیاء نظارت شود تا عیوب آن‌ها قبل از توقف شناسایی شود، همچنین اشیاء بلا استفاده از کنترل‌ها خارج شود تا پردازش‌ها کاهش یابد.

فصل پنجم

۵. جمع‌بندی و پیشنهادها

۵-۱ مقدمه

در این بخش نتایج حاصل از تحقیق با توجه به مباحث پژوهش شده پرداخته شده و پیشنهادات ارائه می‌شود و مزایا و معایب و نکات اصلاحی از دیدگاه شخصی و با توجه به تجربه‌ام در زمینه تحقیقاتی در خصوص بهبود طرح مطرح می‌شود.

۵-۲ نتایج حاصل از تحقیق

در این پایان نامه دکتر چهار فناوری ارائه می‌شود :

۱. مدل سازی محیط اینترنت اشیا و پردازش جریان داده

۲. نگاشت DSPM ها بر روی IoTEM

۳. استقرار اپراتورها در محیط های IoT

۴. نظارت بر DSPM مستقر شده.

در فناوری ۱ 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.

فناوری ۲ برای قرار دادن اپراتورهای پردازش در محیط های IoT، 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.

فناوری ۳ به ارائه چندین رویکرد استقرار بر اساس استاندارد TOSCA می پردازد که 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.

فناوری ۴ با نظارت مستمر محیط های اینترنت اشیا و اپراتورهای مستقر با استفاده از تکنیک های CEP به خوبی تثبیت شده می پردازد ، به طوری که اختلالات در اسرع وقت شناسایی می شوند. فناوری این پایان نامه، پلت فرم IoT Binding and Provisioning Multipurpose (MBP) به عنوان یک نمونه اولیه توسعه داده شده است.

در نهایت رویکرد این تحقیق 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،

۴- استقرار اپراتورهای پردازش بر روی اشیاء اینترنت اشیا،

۵- تشخیص اختلالات موثر بر پردازش داده ها،

۶- توقف و انفصال پردازش داده ها.

این فناوری ها بر اساس استانداردهای ایجاد شده، 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.

۳-۵ بررسی معایب پایان نامه مورد بررسی و بیان پیشنهاد

این پایان نامه با تحقیقات گسترده بر روی تمامی مدل ها و تحقیق های انجام شده در خصوص اشیاء و اپراتورهای پردازش ، به ارائه راهکاری جهت مدل مناسب می پردازد اما بخش مهمی که به آن پرداخته نشده انواع مختلف کاربردهای اینترنت اشیا می باشد چنانچه در این تحقیق بیشتر بر روی اشیاء مانند سنسورها پرداخته است اما در یک شبکه کارخانه شامل اشیاء : ماشین آلات ، سنسور ها و دوربین ها ما بیشتر نیازمند نرم افزار های قدرتمند جهت پردازش می باشیم تا مدل های استقرار.

اگر بخواهیم بر روی پردازش جریان داده در اینترنت اشیا متمرکز باشیم نیازمند نرم افزارهای قدرتمند برای کاهش پردازش ها و تنوع اشیاء می باشیم لذا مدل با توجه به نوع نرم افزار و اشیاء می تواند متفاوت شود مثلا در یک کارخانه ، شبکه ای شامل انواع ماشین آلات داریم که خود دارای بخش های گسترده جهت کنترل و پردازش می باشد و باید نرم افزارهای قدرتمند هوشمند قبل از گره ها باشیم .

۵-۴ ارائه ایده برای پایان‌نامه‌های جدید تکمیلی

- تمرکز بر پردازش‌های هوشمند در گره‌های اشیاء تا گسترش پردازش در شبکه
- تمرکز بر هوشمند سازی تغییرات در اشیاء بعثت حساسیت زمان و کارائی
- تمرکز بر ایجاد مدل‌های نرم افزاری که بتواند خلاء کنترل پردازش‌های طولانی را حل کند

۵-۵ جمع‌بندی و نتیجه‌گیری

در دهه‌های 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 محیط‌های اینترنت اشیا، مقادیر زیادی از جریان‌های داده به طور مداوم تولید می‌شود که منجر به چالش‌های بزرگی در زمینه پردازش و مدیریت آنها می‌شود. در بسیاری از رویکردها، داده‌های اینترنت اشیا به زیرساخت‌های ابری منتقل می‌شوند، جایی که پردازش به صورت متمرکز اجرا می‌شود.

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 شده‌اند. بسیاری از آنها مکان‌یابی را بر اساس برآورده شدن الزامات QoS، مانند تأخیر و پهنای باند، تعیین می‌کنند. 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 محیط‌های 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 برنامه‌های IoT مانند حریم خصوصی داده‌ها و ناشناس سازی، امنیت و قابلیت همکاری را در نظر نمی‌گیرند. بنابراین، در این پایان‌نامه دکتری، رویکردی برای قرار دادن اپراتورهای پردازش جریان داده در محیط‌های 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 و منطق تجاری یک برنامه IoT و الزامات آن توسط یک مدل پردازش جریان داده (DSPM) تعریف میشود. سپس این مدل های اطلاعاتی برای فعال کردن تصمیم گیری برای قرار دادن اپراتور برای محیط های IoT به کار می روند، یعنی تصمیم می گیرند که اپراتورهای پردازشی باید بر اساس تطابق الزامات و قابلیت ها در محیط های IoT قرار گیرند. از طریق رویکرد این پایان نامه دکتری، پردازش داده های برنامه های کاربردی اینترنت اشیا را میتوان برای موارد استفاده خاص، پشتیبانی از الزامات خاص دامنه اینترنت اشیا و علاوه بر آن، کاربران برنامه های اینترنت اشیا تنظیم کرد.

با توجه به رویکرد روش مند که از شش مرحله اصلی تشکیل شده است می توان گفت متخصص دامنه مرحله ۱ و بخشی از مرحله ۵ را انجام می دهد و تحلیلگر دامنه مرحله ۲ و بخشی از مرحله ۶ 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 داده های حسگر یا ارسال دستورات کنترلی به یک محرک تسلط دارند. بنابراین، در مرحله ۱، 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 ۲، 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 ۶ متوقف و منفصل کنند. فناوری ها در مجموع پردازش به موقع و کارآمد داده های اینترنت اشیا را در محیط های اینترنت اشیا امکان پذیر می کنند.

-
- Botta, W. de Donato, V. Persico, A. Pescapé. “Integration of Cloud computing and Internet of Things: A survey ”.
 - Buchmann, B. Koldehofe. “Complex Event Processing”.
 - C. Franco da Silva, P. Hirmer, B. Mitschang. “Model-based Operator Placement for Data Processing in IoT Environments”.
 - Asghari, Parvaneh, Rahmani, Amir Masoud, Javadi, Hamid Haj Seyyed. “Internet of Things applications: A systematic review”.
 - Babcock, S. Babu, M. Datar, R. Motwani, J. Widom. “Models and Issues in Data Stream Systems”.
 - 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.”.
 - H. Derhamy, J. Eliasson, J. Delsing, P. Priller. “A survey of commercial frameworks for the internet of things”.
 - K. Hur, S. Chun, X. Jin, K.-H. Lee. “Towards a Semantic Model for Automated Deployment of IoT Services Across Platforms”.
 - L. Atzori, A. Iera, G. Morabito. “The internet of things: A survey”.
 - M. Blackstock, R. Lea. “Toward a distributed data flow platform for the web of things (distributed node-red)”.
 - 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”.
 - T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, S. Wagner. “OpenTOSCA-A Runtime for TOSCA-based Cloud Applications”.
 - 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.



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 2020