



**دانشگاه پیام نور تهران**

**واحد شهر ری**

**سمینار**

**عنوان:**

**رویکرد مبتنی بر مدل برای پردازش داده ها در بستر اينترنت اشياء IoT**

**استاد راهنما:**

**جناب آقای دکتر سید علی رضوی ابراهیمی**

**نگارش:**

**مجيد لطفي**

**پاييز 1400**

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 دربيايد.

با این حجم فزاینده داده ای که به طور مداوم پردازش می شود، چالش های متعددی چون جلوگيري از تداخل و آسيب در فرآيند هاي گذرا به وجود می آید كه نيازمند بررسي پردازش مبتنی بر جریان داده در محیط های IoT مي باشد و از سوي ديگر با شبكه توزيع شده ناهمگون مواجه هستيم متشكل از انواع سخت افزار ها و سنسور ها كه بايد بتوان اطلاعات آنها را پردازش كرد كه بهترين محيط براي پردازش، محيط ابري مي باشد.

اما با وجود حجم بالاي اطلاعات و پردازش هاي طولاني، نمي توان تمامي اطلاعات در فضاي ابري متمركز پردازش نمود و پيشنهاد مناسب انجام اين پردازش ها در محل نزديك توليد اطلاعات مي باشد (گره پردازشي) و ارائه مدل اجراي اين ساختار بصورت جريان داده و سيستم ناهمگن و توزيع شده، هدف اصلي اين پايان نامه دكتري مي باشد.

**کلمات کلیدی:** اينترنت اشياء ، داده ، جريان داده ، پردازش ابري ، سرویس‌دهنده ابري

فهرست مطالب

[عنوان](#_Toc413125201) صفحه

# **فصل اول**

**آشنايي و معرفي**

## **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، اين فناوري امکان توسعه برنامه‌های کاربردی پیچیده مانند شهرها ، خانه‌ها يا کارخانه‌های هوشمند را فراهم می‌کند.

در محيط 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، چالش های متعددی براي پردازش کارآمد داده های اینترنت اشیا وجود دارد. به عنوان مثال، چگونه می توان پردازش داده های اینترنت اشیا را بدون تأثیر بر واکنش پذیری برنامه های 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 کلي جدید برای پردازش برنامه های کاربردی مبتنی بر جریان داده در محیط های IoT ارائه شده است كه تمرکز آن بر قرار دادن کارآمد اپراتورهای برنامه های 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 می شود، به طوری که کل چرخه حیات محیط های اینترنت اشیا و برنامه های مبتنی بر جریان داده را می توان به راحتی مدیریت کرد.

در این رویکرد، یکی از اهداف اصلی پردازش داده‌های اینترنت اشیا تا حد امکان نزدیک به منابع داده است، به طوری که 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 داده های برنامه های کاربردی اینترنت اشیا را می توان برای موارد استفاده خاص، پشتیبانی از نیازهای خاص دامنه ها، و علاوه بر این، کاربران برنامه های 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) كه یک پلت‌فرم منبع باز اینترنت اشیا، مي باشد پشتیبانی می‌شود.

### 1-1 تعريف مسئله و سؤال

چنانچه در مقدمه بيان شد موضوع تبادل اطلاعات در محيط IoT نيازمند جريان داده و محيط ابري مي باشد و پردازش جريان داده نيز چالش خاص را داشته و نيازمند سيستم هاي توزيع شده مي باشد كه ما در اينجا با توجه به تنوع دستگاه ها، با يك سيستم ناهمگن روبرو هستيم. در اين نوع سيستم ها با توجه به حجم بالاي اطلاعات بهترين راهكار براي پردازش اطلاعات، قراردادن پردازش در اپراتورهاي پردازش (گره هاي پردازشي) نزديك به محل ايجاد اطلاعات و ارسال پردازش نهايي به سيستم ابري است. لذا سؤالاتی كه تا پايان اين تحقيق پاسخ داده می‌شود را این‌گونه بيان می‌کنیم.

1. چگونه بايد پردازش مناسب براي انواع جريان داده را تشخيص داد؟
2. چگونه مي توان اپراتورهاي پردازش جريان داده را در محيط هاي ناهمگن و پويا اجرا كرد؟
3. چگونه مي توان مدل مناسب را طراحي، اجرا و كارائي اين مدل را تضمين كرد؟

### 1-2 ضرورت تحقيق

ما در اينترنت اشياء با تنوع اشياء و داده هاي آنها مواجه هستيم كه در يك جريان داده در حال اجرا بوده و هر دستگاه و شيء در اين شبكه نيازمند پردازش هايي مي باشد كه بايد پردازش داده ها به موقع و کارآمد باشد . اين موضوع نيازمند بررسي تمام جوانب و ايجاد مدل مناسب مي باشد كه بتواند الزامات IoT را پشتيباني كرده و مدلي مناسب و آسان براي کاربران باشد.

### 1-3 اهداف

با توجه به موضوعات مطرح شده در مقدمه و ضروريات تحقيق مي توان اين اهداف را براي اين پايان نامه متصور بود:

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

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

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

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

ايجاد چنين مدل هايي با دو روش خودكار و دستي امكان پذير بوده كه در اين پايان نامه به روش دستي كه در آن خود تحليل گر با توجه به الزامات و ساختار IoT تصميم مي گيرد پردازش ها و نوع ارتباط داده ها به چه صورت باشد.

### 1-4 جمع‌بندی

اینترنت اشیاء به زبان ساده، ارتباط حسگرها و دستگاه‌ها با شبکه‌‌ای است که از طریق آن می‌توانند با یکدیگر و با کاربرانشان تعامل کنند. هر كدام از اين دستگاها يا اشياء داراي داده هايي مي باشند كه مداوم و آنلاين در حال ساخت مي باشد كه خود باعث ايجاد يك جريان داده مي شود. در اين محيط از يك سو جريان داده و از سوي ديگر پردازش اين داده ها را داريم لذا بايد بدانيم اين ساختار را چگونه و با چه مدلي اجرا نماييم.

این مدل بايد منابع داده، سینک های داده، اپراتورهای پردازش و جریان داده را در میان آنها توصیف کند و علاوه بر این، الزامات اپراتورهای پردازش برای اشیاء IoT را شرح دهد.

# **فصل دوم**

**پيشينه و سابقه**

## 2-1 اينترنت اشياء

اصطلاح اینترنت اشیا (IoT) برای اولین بار در اواخر دهه 90 ظاهر شد، با ایده اشتون که به رایانه ها اجازه می دهد همه چیز را در مورد چیزها بدانند، اين ایده ابتداعا تقویت رایانه‌هایی با فناوری‌های شناسایی فرکانس رادیویی (RFID) و حسگر برای جمع‌آوری اطلاعات، مشاهده و شناسایی یک محیط بدون نیاز به کمک انسانی بود. به این ترتیب، امکان ردیابی و نظارت بر موارد به منظور کاهش هزینه ها و علاوه بر آن، اطلاع از زمان نیاز به تعمیر یا تعویض وجود دارد. ورمسان و همکاران اینترنت اشیا را به عنوان الگویی تعریف می کند که در آن چیزهای مختلفی وجود دارد.

در این محیط ، این اشياء 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 حتي شهر هاي هوشمند توسعه یافته‌اند.

## 2-2 پردازش جريان داده و داده پيچيده

الزامات برنامه های کاربردی اینترنت اشیا نمی توانند با به کارگیری سیستم های مدیریت پایگاه داده سنتی (DBMS) كه با استفاده از فرآیندهای استخراج-تبدیل بار (ETL)، که نیاز به ذخیره و فهرست بندی داده ها برای پردازش دارند، برآورده شوند. بنابر اين یکی از نیازهای مهم برنامه های اینترنت اشیا، توانایی پردازش جریان‌های داده‌ای چندگانه، پیوسته، سریع و متغیر مي باشد که اين 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، پرس‌وجوهای پیوسته را بر روی جریان‌های داده ورودی اجرا می‌کند 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 دنباله‌ای از تبدیل‌های مبتنی بر عملگرهای SQL، مانند انتخاب، تجمیع، یا پیوستن، که توسط جبر رابطه‌ای تعریف می‌شود، پردازش می‌کند. از سوی دیگر، پردازش رویدادهای پیچیده (CEP) شامل مجموعه‌ای از اصول و تکنیک‌ها برای تجزیه و تحلیل مجموعه‌هایی از رویدادها است که تا حدی بر اساس زمان با رسیدن این رویدادها مرتب شده‌اند. 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 می‌شوند، با این حال، می‌توانند با رویدادهای نامرتبط دیگر مخلوط شوند. ویژگی مهم CEP توانایی تشخیص الگوها (به عنوان مثال، روابط) بین رویدادها است به عنوان مثال، موقعیت‌هایي 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.

## 2-3 مشكل قراردادن اپراتورها

پردازش جریان های داده می تواند از طریق یک سيستم 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 مي 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.

## 2-4 تكنولوژي TOSCA

فناوري 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 می‌تواند در عين هزینه‌های پایین برای استقرار کل محیط‌های اینترنت اشیا ، قابليت راه‌اندازی و ادغام سریع اشیاء جدید و برنامه‌های کاربردی اینترنت اشیا را ايجاد کند.

فناوري TOSCA (Topology and Orchestration Specification for Cloud Applications) یک استاندارد OASIS تایید شده برای مدل سازی، استقرار و مدیریت برنامه های کاربردی ابری است كه داراي دو بخش اصلی است: (1) مدل توپولوژی یک برنامه کاربردی كه شامل اجزای نرم افزار ، پلت فرم و زیرساخت آن مي باشد، و (2) 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. تصوير زير يك نمونه از مدل TOSCA براي يك نرم افزار كاربردي انتشار و اشتراك اطلاعات مي باشد.

****

شکل 1

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 . همچنين 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 .

# **فصل سوم**

**بررسي اجمالي پايان نامه**

## 3-1 دستاوردها

دستاورد ها و پژوهش هاي ارائه شده در اين پايان نامه بر اساس استانداردهای تعیین شده (مانند MQTT، TOSCA، XML) است تا از کاربرد طولانی مدت آنها اطمینان حاصل شود. اين دستاوردها به شرح ذيل مي باشند:

* مدل پردازش جريان داده در محيط اينترنت اشياء :

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

* استقرار مدل پردازش جريان داده (DSPM) در محيط اينترنت اشياء (IoTEM) :

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

1. یک رویکرد خودکار، که در آن یک طرح نقشه‌برداری حاوی حداقل یک مجموعه از اشیاء IoT که الزامات DSPM را برآورده می‌کنند، به‌طور خودکار تولید می‌شود كه برای محيط هاي بزرگ‌تر مناسب‌ است، به عنوان مثال، در حوزه کارخانه هوشمند، که در آن تعداد زیادی از اشیاء در دسترس هستند.
2. یک رویکرد دستی، كه در آن تحلیلگران خودشان تصمیم می گیرند که کدام اشیاء IoT باید کدام اپراتورها را اجرا کنند. برای این کار، تحلیلگران به صورت دستی یک مدل استقرار ایجاد می کنند كه برای موارد کوچک توصیه می شود، به عنوان مثال، برای خانه های هوشمند.

* استقرار اپراتورها در محیط های IoT :

اساس این تحقيق از استاندارد TOSCA كه مورد تایید سازمان پیشرفت استانداردهای اطلاعات ساختاریافته (OASIS) استفاده می‌کند كه در دو رويكرد خودكار و دستي الزامي مي باشد.

* نظارت بر DSPM های مستقر شده :

پس از استقرار اپراتورهای پردازش، پردازش کلی جریان داده یک برنامه IoT آغاز می شود و برای اطمینان از اینکه این پردازش تا زمانی که برنامه IoT به آن نیاز دارد درست می ماند بايد ابزاری را برای تشخیص اختلالات مؤثر بر مدل پردازش جریان داده مستقر شده (DSPM) فراهم كرد. چنین اختلالاتی می تواند از طریق تغییرات در محیط اینترنت اشیا (به عنوان مثال، یک دستگاه معیوب) یا تغییر در DSPM (به عنوان مثال، با افزودن یک نیاز جدید) رخ دهد. برای تشخیص چنین اختلالاتی، MBP به طور مداوم اشیاء IoT و DSPM ها را نظارت می کند. این نظارت عمدتاً با استفاده از تکنیک‌های پردازش رویداد پیچیده (CEP) انجام می‌شود. چنین تکنیک هایی به خوبی تثبیت شده اند و برای پردازش مستمر مقادیر زیادی از داده ها، به عنوان مثال، برای تشخیص به موقع موقعیت های بحرانی (یعنی تغییراتی که نیاز به اقدامات اصلاحی دارند) استفاده می شوند.

## 3-2 رويكرد تخصصي

متخصصان دامنه (Domain experts) 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. این دانش فنی شامل اطلاعاتی در مورد قابلیت های محاسباتی (حافظه اصلی موجود، اتصال، قدرت پردازش ) است.

همچنين، اين کارشناسان در مورد نحوه دسترسی به این اشیاء برای مثال، استخراج داده‌های حسگر یا ارسال دستورات کنترلی به یک محرک، دانش دارند. وظیفه اصلی اين متخصصان ، ایجاد IoTEM ها و ثبت خودکار در MBP و ایجاد DSPM ها مي باشد زيرا دانش كافي در مورد پردازش داده های تولید شده در محیط اینترنت اشیاء را دارند، به عنوان مثال، دانش لازم را برای مدل سازی برنامه های کاربردی اینترنت اشیا مختلف برای موارد استفاده خاص.

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

بر اساس قابلیت‌های کلی IoTEM و الزامات مشخص‌شده توسط DSPM بايد تصمیم‌گیری كرد که اپراتورها بر روی کدام اشیاء IoT مستقر شوند و این کار توسط الگوریتم هایی انجام می شود كه هم قابلیت‌های اشیاء IoT، و هم قابلیت‌های اتصالات شبکه را در نظر می‌گیرند.

قرار دادن اپراتورها بايد تا حد امکان به منابع داده نزدیک باشد تا احتمالاً ترافیک شبکه و حجم داده مبادله شده در یک محیط IoT کاهش یابد. تحلیلگران دامنه به صورت دستی یک نقشه ایجاد می کنند و با توجه به طرح نقشه، اپراتورهای پردازش می توانند بر روی اشیاء IoT مربوطه خود مستقر شوند و اجرای DSPM می تواند آغاز شود. اشیاء IoT ابتدا برای اجرای اپراتورهای پردازشی آماده می شوند، به عنوان مثال، نرم افزار طراحي شده مورد نیاز یک اپراتور پردازشگر نصب و در صورت لزوم پیکربندی می شوند. نرم افزارهاي طراحي شده بیشتری برای جمع آوری مقادیر تجربی و نظارت بر اپراتورهای پردازش و اشیاء اینترنت اشیا نیز نصب و پیکربندی می شوند. پس از آن، اپراتورهای پردازش بر روی اشیاء IoT متناظر خود مستقر می شوند و سپس راه اندازی می شوند، که منجر به یک نمونه در حال اجرا از DSPM می شود كه dDSPM ناميده مي شود. آماده سازی اشیاء IoT و استقرار متعاقب آن اپراتورهای پردازش با استفاده از استاندارد TOSCA محقق می شود.

برای اطمینان از اینکه پردازش کلی در محیط اینترنت اشیا درست می‌ماند، اشیاء IoT و dDSPM به‌منظور تشخیص اغتشاشات به‌طور مداوم نظارت می‌شوند. اگر اختلالی بر پردازش اصلی تأثیر منفی بگذارد، الگوریتم‌های استقرار مجدداً راه‌اندازی می‌شوند و یک طرح نقشه‌برداری جدید ایجاد می‌شود.

اپراتورهای پردازش و نرم افزارهاي طراحي شده که دیگر مورد نیاز نیستند متوقف و حذف می شوند و پردازش جریان داده برنامه IoT می تواند كنار گذاشته شود و اشیاء IoT پاک شوند.

## 3-3 معماري كلي

معماری کلی طبق تحقيقات اين پايان نامه 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 محیط های فیزیکی IoT و برنامه های کاربردی IoT را پر می کند. تصوير زير اين نماي كلي را نشان ميدهد :

شكل 1

لایه محیط فیزیکی اینترنت اشیا

استقرار اپراتورهای پردازش

همگام سازی

استخراج داده های حسگر، محرک های کنترل

داشبورد كلي

تشخیص اختلال Disturbance recognizer

مديريت استقرار Deployment manager

ارتباط دهنده بين DSPM و IoTEM

مدل ساز و مدیر DSPM

مدل ساز و مدیر IoTEM

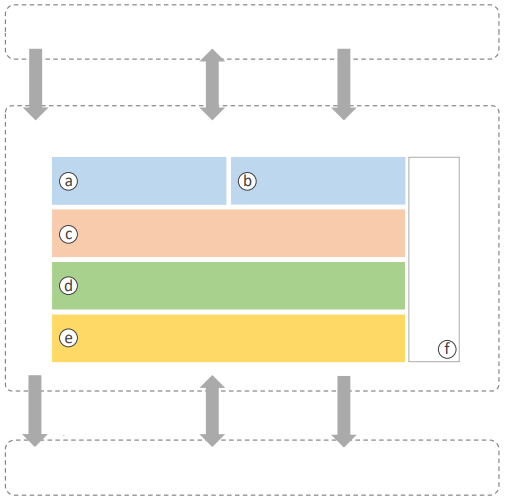
تحليل نتايج پردازش

ارسال/ دريافت

اطلاعات سنسورها

ارسال دستورات

لايه هاي برنامه محيط اينترنت اشياء



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

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

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

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

بخش مديريت 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 طراحي شده مورد نیاز دیگر برای نظارت بر اشیاء اینترنت اشیا، می تواند از طریق این مؤلفه نیز مستقر شود.

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

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

## 

# **فصل چهارم**

**مدل محيط اينترنت اشياء و پردازش جريان داده**

## 4-1 مدل محيط اينترنت اشياء

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

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

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

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

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **عنوان مدل** | **تكامل** | **سلسله مراتبي** | **دسترس بودن** | **پياده سازي** | **موقعيت جغرافيايي** | **كاربرد** |
| homeML | Non-standard | ✘ | ✘ | ✘ | ✓ | خانه هاي هوشمند |
| IEEE 1451 | standard | ✘ | ✓ | ✓ | ✘ | متمركز بر روي سنسور ها |
| IoT ARM | Non-standard | ✓ | ✘ | ✘ | ✓ | مدل مرجع عمومي |
| IoT-Lite | submitted | ✓ | ✓ | ✓ | ✓ | حسگرها و SSN ontology |
| IoT MC | standard | ✓ | ✓ | ✓ | ✘ | IoTivity |
| IoT-O | standard ext | ✓ | ✘ | ✘ | ✘ | حسگرها و SSN ontology |
| Nexus | Non-standard | ✓ | ✓ | ✓ | ✓ | جغرافیایی سازی |
| oneM2M | standard | ✓ | ✓ | ✓ | ✓ | تمرکز بر خدمات دستگاه های IoT |
| OPC-UA | standard | ✓ | ✓ | ✓ | ✘ | کارخانه های هوشمند |
| SenML | standard | ✘ | ✓ | ✓ | ✓ | تمرکز بر حسگرها و مقادير حسگر |
| SensorML | standard | ✘ | ✓ | ✓ | ✓ | پشتیبانی از پردازش ها |
| SSN | standard | ✓ | ✓ | ✓ | ✓ | مورد استفاده IoT-Lite / IoT-O |
| TDLIoT | Non-standard | ✘ | ✓ | ✓ | ✓ | نمونه اولیه تحقیق |
| Vorto | Non-standard | ✘ | ✓ | ✓ | ✘ | زبان برنامه نويسي |

معیار تكامل نشان مي دهد آیا مدل يك مدل استاندارد تایید شده یک سازمان، مانند OASIS، W3C یا OGC است یا خیر و مي توان فرض گرفت سازمان‌ها تحت یک فرآیند بررسی کامل قرار گرفتند و یک استاندارد مزایایی را ارائه می‌کند.

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

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

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

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

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

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

### 4-1-1 تعيين IoTEM

تعريف IoTEM یک گراف بدون جهت است که شامل اشیاء IoT به عنوان گره و اتصالات شبکه آنها به عنوان يالهاي گراف است كه بر اساس فاصله اتصالات شبكه وزن دهي شده است. ساختار آن به شرح ذيل مي باشد :

مدل در محیط IoT متشكل از چند بخش است : اشیاء (Object)، اتصالات بين اشياء (Connection)، فاصله (Distance) (تابع براي وزن دهي اتصالات شبكه بر اساس فاصله)، قابلیت‌های شی (ObjectCapabilities)، قابلیت هاي اتصال (ConnectionCapabilities)، تخصیص قابلیت شی (objectCapabilityAssignment)، تخصیص قابلیت اتصال (connectionCapabilityAssignment) كه در تصوير زير اين مدل نمايش داده شده.



شکل 2

### 4-1-2 قابليت هاي اشياء 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، ESP8226و Raspberry Pi هستند.

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

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

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

* حافظه (Memory): این قابلیت با حافظه کاری موجود (یعنی RAM) دستگاه IoT مطابقت دارد.
* نوع شبکه (Networking type). این قابلیت نحوه اتصال یک دستگاه اینترنت اشیا به دستگاه های دیگر در محیط اینترنت اشیا را توضیح می دهد. نمونه هایی از انواع شبکه عبارتند از: اترنت (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، دستگاه‌های وابسته به باتری باید بتوانند در حالت‌های مصرف کم مصرف کار کنند تا بتوانند با برنامه‌های طولانی مدت اینترنت اشیا همگام شوند. این قابلیت حالت‌های مصرف انرژی را که یک دستگاه اینترنت اشیا می‌تواند پشتیبانی کند، توصیف می‌کند. به عنوان مثال، Raspberry Pi چهار حالت قدرت را ارائه می دهد، از جمله یک حالت اجرا (پیش فرض) و یک حالت آماده به کار .
* قدرت پردازش (Processing power). این قابلیت با سرعت CPU دستگاه اینترنت اشیا مطابقت دارد. به عنوان مثال، Raspberry Pi3 مدل B دارای چهار پردازنده با فرکانس 1.2 گیگاهرتز است.
* گنجایش دخيره سازي (Storage capacity). این قابلیت با مقدار ذخیره سازی موجود مطابقت دارد. این برای اپراتورهایی که داده‌های حسگر را جمع‌آوری می‌کنند یا نتایج میانی را ذخیره می‌کنند، مهم است.
* پشتیبانی از زمان اجرا (Supported runtime): این قابلیت مربوط به لیستی از محیط‌های زمان اجرا نرم‌افزار پشتیبانی شده و در دسترس (مانند موتورهای جاوا، پایتون، CEP) یک دستگاه اینترنت اشیا است.

همانطور که در بخش 4-1 بيان 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.

### 4-1-3 مولفه هاي معماری و پیاده سازی بخش مدل ساز و مدیر IoTEM

در معماری کلی (در بخش 3-3)، بخش مدل‌ساز و مدیر IoTEM با پشتيباني متخصصان دامنه امكان ارائه ابزاری برای ایجاد و ذخیره IoTEM و علاوه بر این، مدیریت IoTEM‌ های ثبت‌شده در MBP را فراهم مي كند.

بخش مدل ساز و مدیر IoTEM در شکل زير نشان داده شده است.



شکل 3

این مدل شامل ابزار مدل‌سازی گرافیکیIoTEM (بخش a) است که برای مدل‌سازی محیط اینترنت اشیا با کشیدن و رها کردن انواع شیIoT (بخش b) در یک محيط مدل‌سازی استفاده می‌شود.

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 مدل‌سازی IoTEM به متخصصان حوزه امکان می‌دهد، علاوه بر ذخیره و بازیابی IoTEM، تمام اشیاء IoT یک IoTEM را به‌طور هم‌زمان در MBP ثبت کنند. این ثبت، مدیر اشیاء اینترنت اشیا (بخش d) را فعال می کند تا همتایان دیجیتالی (به عنوان مثال، دوقلوهای دیجیتال) اشیاء IoT مدل شده را نمونه برداری کند.

این همتاهای دیجیتالی در حافظه IoTEM ذخیره می 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. چگونگی تشخیص اختلالات به تفصیل در فصل 7 توضیح داده خواهد شد.

### 4-1-4 كارهاي مرتبط

چنانچه قبلا بيان شد رویکردهای زیادی برای مدل‌سازی محیط‌های IoT وجود دارد و مدل‌ها و ابزارهای IoT را برای ایجاد و مدیریت نمونه‌های این مدل‌های IoT ارائه شده اند. 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 مي باشند و نه از طریق استقرار و یک مفهوم کل نگر مانند این پایان نامه برای مدل سازی، پیکربندی و نظارت بر محیط های IoT ارائه نمی کنند.

مک دونالد و همکاران یک ابزار 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 می کند که عمومی است و می تواند در سایر حوزه های اینترنت اشیا مانند کارخانه هوشمند یا شهر هوشمند نیز اعمال شود.

مایر و همکاران رویکردی را نشان می‌دهد که ابرداده‌های 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 اتاق 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 OpenMTC و Microsoft Azure IoT .

با این حال، آنها ابزاری برای مدل‌سازی و مدیریت محیط‌های کامل اینترنت اشیا فراهم نمی‌کنند، به‌عنوان مثال، اشیاء 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 پایان نامه، بررسی ادبیات کاملی در مورد مدل های پیشرفته اینترنت اشیا انجام شده و در آن مقایسه ای مبتنی بر معیار از مدل های اینترنت اشیا ارائه شده است. در نتیجه این مقایسه، هستی شناسی IoT-Lite بهترین مدل مناسب برای اهداف این پایان نامه نشان داده شده است.

علاوه بر این، می‌توان از مدل‌های عمومی‌تری که به‌طور خاص برای محیط‌های IoT طراحی نشده‌اند نیز استفاده کرد. برای مثال، 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.

## 4-2 مدل سازی پردازش جریان داده

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

4-2-1 تعریف DSPM

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

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

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

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

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

بر اساس نیازهای محاسباتی اپراتورها، می توان به دنبال قرارگیری اپراتور مناسب در محیط IoT بود که تمام الزامات یک DSPM مدل شده را برآورده می کند. الزامات محاسباتی احتمالی آنهایی هستند که قابلیت‌های محاسباتی نمونه دستگاه‌های IoT را که در بخش 4-1-2 توضیح داده شده است، برآورده می‌کنند. برای برخی نیازها، مانند حافظه اصلی مورد نیاز، تحلیلگران دامنه معمولاً نمی دانند از چه مقادیر مشخصی باید استفاده شود.

این دانش به 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 تکنیک‌های تحلیلی پردازش می‌شوند تا توصیه‌هایی در مورد الزامات در زمان مدل‌سازی به تحلیل‌گران حوزه ارائه شود.

شکل زير 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 4

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 می شوند، در حالی که سینک های داده توسط داشبورد (Si1، Si2، Si3) نشان داده می شوند. 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 می شود و در طول زمان به منظور تشخیص افزایش ها تجزیه و تحلیل می شود.

4-2-1 اپراتورهای پردازش

در این پایان نامه، یک اپراتور پردازش یک جزء نرم افزاری است که عملیاتی (به عنوان مثال، فیلتر کردن داده ها، تابع میانگین) را بر روی یک یا چند جریان داده ورودی اجرا می کند. این اجزای نرم افزار می توانند به عنوان مثال، اسکریپت های پایتون یا شل، فایل های 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 به‌طور پیوسته اجرا می‌شوند و براساس پنجره‌ها، یعنی فواصل زمانی داده‌ها بر اساس زمان یا تعداد عناصر، به دلیل ماهیت بی‌نهایت بودن جریان داده هستند.

علاوه بر این، عملیات مبتنی بر پنجره ضروری است، زیرا داده‌ها در محیط‌های IoT می‌توانند 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 رویه‌های قرار دادن اپراتور آماده هستند، یعنی مکانیسم‌های لازم برای توزیع پردازش داده‌ها قبلاً به کار گرفته شده است. رویکردهایی که پردازش جریانی را در محیط های توزیع شده و محیط های IoT انجام ميدهند را می توان در لیست زير نشان داد:

* تجمع (Aggregation): این عملگر یک تابع تجمیع را به مجموعه ای از مقادیر (به عنوان مثال، یک محيط اجرايي window) در یک جریان داده اعمال می کند و یک مقدار واحد را برمی گرداند. نمونه هایی از تجمیع 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 داده‌ها را از منابع داده استخراج می‌کند، برای مثال، می‌تواند کد نرم‌افزاری باشد که به رابط فیزیکی یک حسگر (مثلاً از طریق GPIO) برای دریافت اندازه‌گیری‌های واقعی حسگر دسترسی دارد.
* فیلتر (Filter). این عملگر داده ها را بر اساس پارامترها فیلتر می کند.
* پیوستن (Join). این اپراتور دو جریان داده را پیوند مي 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 تبدیل ها را متوجه می شود، به عنوان مثال، در بین قالب های مختلف داده یا طرحواره های داده.

4-2-2 ساختار معماری و پیاده سازی - مدل ساز و مدیر DSPM

در معماری کلی (به بخش 3.3 رجوع کنید)، مؤلفه مدلساز و مدیر DSPM از تحلیلگران دامنه با ارائه ابزاری برای ایجاد و ذخیره DSPMها و علاوه بر این، مدیریت DSPMهای مستقر در محیط های IoT پشتیبانی می کند. این مشخصات در شکل زير نشان داده شده است:



شکل 5

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 می‌کند و مدل زیربنایی مبتنی بر JSON آن را می‌توان برای فعال کردن حاشیه‌نویسی نیازمندی‌های محاسباتی در گره‌ها و لبه‌ها گسترش داد، که مقدمه‌ای برای مفاهیم این پایان‌نامه است. ابزار مدل‌سازی DSPM با کشیدن و رها کردن منابع داده، سینک‌های داده و اپراتورهای پردازش به منطقه مدل‌سازی، ایجاد DSPM را امکان‌پذیر می‌سازد. در حالی که اپراتورها در مخزن نوع اپراتور (بخش b) موجود هستند،

منابع داده و 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 مدل‌سازی‌شده در ذخیره‌سازیDSPM (بخش c) ذخیره می‌شود. 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 6 مراجعه کنید)، مدیر dDSPM از موفقیت آمیز بودن استقرار مطلع می شود و یک dDSPM مربوطه ایجاد می کند که حاوی اطلاعاتی است در مورد اینکه کدام شی IoT در حال پردازش کدام اپراتور است.

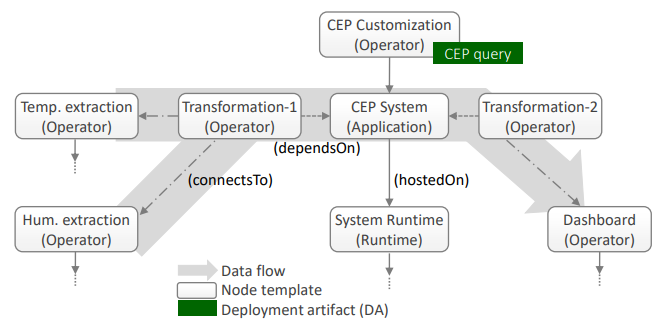
مدیر dDSPM دائماً اطلاعات وضعیت را از مدیر اشیاء IoT بازیابی می کند تا مثلاً بینشی از سلامت اپراتورهای مستقر و اشیاء موجود IoT بدست آورد. این اطلاعات را می توان در داشبورد MBP مشاهده کرد. در نهایت، ابزار مدل‌سازی تحلیلگران دامنه را قادر می‌سازد تا dDSPM‌ها را توسط مدیر dDSPM متوقف نمايد.

### 4-2-4 كارهاي مرتبط

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

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

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



شکل 6

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

# **فصل پنجم**

**ارتباط بين DSPM ها در مدل هاي اينترنت اشياء (IoTEM)**

## 5-1 رويكرد خودكار

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

اجزاي IoTEM :Objects, Connections, Distance, Object Capabilities, Connection Capabilities, Object Capability Assignment, Connection Capability Assignment كه پيش تر بيان شد.

اجزايDSPM : Operators, Sources, Sinks, Edges, Operator Requirements, Edge Requirements, Operator Requirement Assignment, edge Requirement Assignment نيز پيش تر بيان شد.

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

Source Mapping: Sources → Objects

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

Sink Mapping: Sinks → Objects

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

Operator Mapping: Operators → Objects

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

Edge Mapping: Edges → Paths (IoTEM):

ارتباطي 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 كه 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، الگوريتم هاي 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 مجموعه‌ای از اشیاء IoT است که قابلیت‌های آنها نیازهای اپراتورها را برآورده می‌کند. قراردادن اشیاء IoT نزدیک به منابع داده ترجیح داده می شوند، زيرا تشخیص زودهنگام موقعیت های بحرانی و اقدامات به موقع مربوطه را ممکن می سازد. علاوه بر این، حجم داده‌های مبادله شده در محیط اینترنت اشیا را می‌توان با جمع‌آوری و فیلتر کردن داده‌ها در نزدیکی منابع آنها و در اولین فرصت ممکن در زنجیره پردازش داده کاهش داد.

برای هر دو الگوریتم، فرض بر این است که اشیاء IoT که از طریق رابط‌های سخت‌افزاری به حسگرها یا محرک‌ها متصل هستند، قابلیت‌های محاسباتی کافی را برای اجرای استخراج داده‌های 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.

### 5-1-1 الگوریتم تطبیق (حريصانه)

الگوریتم در شبه کد نشان داده شده كه IoTEM و DSPM را به عنوان ورودی نیاز دارد. خروجی نقشه ای از عملگرها و لبه های DSPM را بر روی اشیاء IoT و اتصالات آنها به IoTEM برمی گرداند. در ادامه الگوریتم به صورت گام به گام شرح داده شده است.

1. function GreedyMatching((Sources, Sinks,Operators, Edges, ...), (Objects, Connections, ...))
2. DSPM ← (Sources, Sinks,Operators, Edges, ...)
3. IoTEM ← (Objects, Connections, ...)
4. operatorMapping : Operators → Objects
5. edgeMapping : Edges → Paths(IoTEM)
6. order ← GetTopologicalOrder(Operators, Edges)
7. sourceMapping ← FillSourceMapping(IoTEM, DSPM)
8. sinkMapping ← FillSinkMapping(IoTEM, DSPM)
9. i ← 0
10. while i < |Operators| do
11. opj ← order(i)
12. if (opj .isConnectedToSource()) then
13. opObject ← sourceMapping(opj)
14. ConsumeCaps(operatorRequirementAssignment(opj), objectCapabilityAssignment(opObject))
15. operatorM apping(opj ) ← opObject
16. else if (opj .isConnectedToSink()) then
17. opObject ← sinkMapping(opj )
18. ConsumeCaps(operatorRequirementAssignment(opj), objectCapabilityAssignment(opObject))
19. operatorMapping(opj ) ← opObject
20. else
21. PreOperators ← {opi ∈ Operators : ∃(opi , opj ) ∈ Edges}
22. PreObjects ← operatorMapping(PreOperators)
23. closestOrder ← GetClosestObjects(IoTEM, PreObjects)
24. foundObject ← false
25. j ← 0
26. while ( j < |Objects| & foundObject) do
27. opObject ← closestOrder(j)
28. if TryMapOperator(opj , opObject, operatorMapping, edgeMapping, DSPM, IoTEM) then
29. foundObject ← true
30. end if
31. end while
32. if foundObject then
33. return "No solution found"
34. end if
35. end if
36. end while
37. return (operatorM apping, edgeMapping)
38. end function

در خط 6 تابع GetTopologicalOrder فراخوانی می شود که DSPM را به همراه Operators و Edge هایش به عنوان ورودی می گیرد. این تابع یک ترتیب نقشه برداری را برمی گرداند: {0، 1، . . . ، که با مرتب سازی توپولوژیکی عملگرها مطابقت دارد.

در خط 7، تابع FillSourceMapping نامیده می شود که DSPM و IoTEM را به عنوان ورودی می گیرد و نقشه ای از منابع انتزاعی شده از حسگرها و اشیاء اختصاصی اینترنت اشیا متصل به این حسگرها را برمی گرداند. به طور مشابه، تابع FillSinkMapping در خط 8 نقشه‌ای از سینک‌های انتزاعی از محرک‌ها و اشیاء اختصاصی اینترنت اشیا متصل به این محرک‌ها را برمی‌گرداند.

در مرحله بعد، لیست مرتب شده اپراتورها پیمایش می شود. اگر اپراتور فعلی به یک گره منبع داده متصل باشد، شی IoT متصل به منبع داده (یعنی حسگر) بازیابی می شود (خط 13). سپس قابلیت های این شی اینترنت اشیا با کم کردن منابع مورد نیاز اپراتور تنظیم می شود (خط 14).

در نهایت، اپراتور بر روی شی IoT (خط 15) نگاشت می شود. سپس، حلقه با عملگر بعدی ادامه می یابد. اگر اپراتور فعلی به یک گره سینک داده (خط 16) متصل باشد، مانند منبع داده همانطور که در بالا توضیح داده شد، مدیریت می شود. در خط 21، لیستی از اپراتورهای پیشین اپراتور فعلی در DSPM تعیین می شود و متعاقبا، اشیاء اینترنت اشیا میزبان این اپراتورهای قبلی نیز بازیابی می شوند (خط 22).

تابع GetClosestObjects همه اشیاء IoT را در IoTEM (خط 23) بر اساس فاصله شبکه آنها با اشیاء IoT قبلی مرتب می کند. نتیجه این است که نقشه به ترتیب بسته می‌شود: {0، 1، . . . ، } در حالی که closestoOrder(0) به شی IoT نزدیک ترین به اشیاء IoT قبلی اشاره دارد. اشیاء قبلی IoT نیز بخشی از این مرتب سازی هستند و ابتدا ظاهر می شوند زیرا فاصله شبکه آنها 0 است. پس از آن، لیست نزدیکترین اشیاء IoT طی می شود (خط 26) و بررسی می شود که آیا می توان اپراتور را روی یکی از آنها نگاشت کرد یا خیر. سپس چك كردن نزدیک ترین اشیاء اینترنت اشیا بسته به قابلیت های موجود آنها.

اگر نگاشت امکان پذیر باشد، به عنوان مثال، تابع TryMapOperat مقدار true را برمی گرداند (خط 28)، یک شی IoT پیدا شد و اپراتور با موفقیت بر روی شی IoT یافت شده نگاشت شد. در غیر این صورت، نزدیکترین شی IoT بعدی بررسی می شود. اگر هیچ شیء IoT قابلیت های لازم را نداشته باشد، هیچ راه حلی نمی توان یافت. در این حالت، الگوریتم از یک زیرساخت ابری به عنوان یک بازگشت استفاده می‌کند تا از یک راه‌حل اطمینان حاصل شود. در نهایت، الگوریتم مجموعه ای از نگاشت عملگرها و اشیاء را برمی گرداند (خط 37).

نوع حریصانه راه حلی را ارائه می دهد، با این حال، ممکن است بهترین راه حل در مورد حداقل فاصله شبکه بین اپراتورها نباشد.

## 5-1-2 الگوریتم تطبیق (روش عقبگرد)

هدف این الگوريتم نه تنها یافتن یک راه حل، بلکه بهترین راه حل است. ورودی آن مانند الگوریتم قبلي است، یعنی IoTEM و DSPM. نمونه شبه كد آن به اين صورت است:

1. function BACKTRACKING MATCHING(Sources, Sinks, Operators, ...), (Objects, Connections, ...))
2. DSPM ← (Sources, Sinks, Operators, ...)
3. IoTEM ← (Objects, Connections, ...)
4. operator Mapping : Operators →Objects
5. edgeMapping: Edges → Paths(IoTEM)
6. solution operator Mapping, edgeMapping)
7. source Mapping ← FILLSOURCEMAPPING(IOTEM, DSPM)
8. sink Mapping ← FILLSINKMAPPING(IOTEM, DSPM)
9. RestOperators ← Operators
10. for op € RestOperators do
11. if (op.isConnectedToSource() then
12. opObject ← sourceMapping(op)
13. CONSUMECAPS(operator RequirementAssignment(op) objectCapability Assignment(opObject))
14. operator Mapping(op) ← opObject
15. RestOperators ← RestOperators \ op
16. else if (op.isConnected ToSink) then
17. opObject ← sink Mapping(op)
18. CONSUMECAPS(operator RequirementAssignment(op) objectCapability Assignment(opObject))
19. operator Mapping(op) ← opObject
20. RestOperators ← Rest Operators \ op
21. end if
22. end for
23. Solutions ← Ф
24. found Solution ← FINDSOLUTION(solution, RestOperators, Edges, Solutions, DSPM, IOTEM)
25. if (Solutions ← found Solution) then
26. return "No solution found"
27. end if
28. return GETBESTSOLUTION(Solutions, Edges, distance)
29. end function

در خطوط 11-15، تمام اپراتورهای متصل به منابع DSPM پیمایش می شوند، اشیاء مربوط به IoT بازیابی می شوند و عملگرهای استخراج داده بر روی این اشیاء IoT نگاشت می شوند. پس از آن عملگرهای استخراج از لیست عملگرها حذف می شوند.

در خطوط 16-21، تمام اپراتورهای متصل به سینک ها نیز پیمایش می شوند، اشیاء IoT مربوطه بازیابی می شوند و اپراتورهای ارائه دهنده داده بر روی این اشیاء IoT نگاشت می شوند. پس از آن اپراتورهای سرویس دهنده نیز از لیست اپراتورها حذف می شوند. سپس تابع FindSolution فراخوانی می شود (خط 24) که شامل منطق تطابق نیازمندی ها و قابلیت ها است. كد اين تابع به صورت زير مي باشد.

1. function FindSolution(solution, Opera tors, Edges, Solutions, DSPM, IoTEM)
2. if Operators ̸= Ф then
3. foundSolution ← false
4. nextOperator ← GetOneElement(Operators)
5. for currObject ∈ Objects do
6. if SatisfiesReqs(operatorRequirementAssignment(nextOperator), objectCapabilityAssignment(currObject)) then
7. ConsumeCaps(operatorRequirementAssignment(nextOperator), objectCapabilityAssignment(currObject))
8. operatorMapping(nextOperator) ← currObject
9. Operators ← Operators \ {nextOperator}
10. if FindSolution(solution,Operators, Edges, Solutions, DSPM, IoTEM) then
11. foundSolution ← true
12. return true
13. else
14. Undo(operatorRequirementAssignment(nextOperator), objectCapabilityAssignment(currObjec t))
15. operatorMapping(nextOperator) ← null
16. Operators ← Operators ∪ {nextOperator}
17. end if
18. end if
19. end for
20. return foundSolution
21. else
22. if Ed ges ̸= Ф then
23. // similar to operators
24. else
25. Solutions ← Solutions ∪ {solution}
26. return true
27. end if
28. end if
29. end function

اين تابع FindSolution، یک جستجوی عمقی را برای راه حل های ممکن با استفاده از بازگشت انجام می دهد. در خط 5 بررسی می شود که الگوریتم تا چه حد پیشرفت کرده است. اگر مجموعه اپراتورها همچنان حاوی عناصر باشد، ابتدا این عناصر بر روی اشیاء IoT نگاشت می شوند.

در خط 3، متغیر foundSolution تعریف شده است که به خاطر می‌سپارد که آیا راه‌حلی پیدا شده است یا خیر. در خط 4، تابع GetOneElement عملگر فعلی را برمی گرداند. پس از آن، اشیاء IoT که برای اجرای این اپراتور مناسب هستند با تکرار تمام اشیاء IoTEM جستجو می شوند. توجه به این نکته ضروری است که ترتیب پردازش اشیاء IoT برای هر تکرار یکسان است. در غیر این صورت، دیگر امکان عقب نشینی وجود ندارد، یعنی بعداً مراحل نقشه برداری را لغو کنید و به دنبال جایگزین باشید.

در خط 6، بررسی می شود که آیا دستگاه فعلی نیازهای اپراتور nextOperat را برآورده می کند یا خیر. اگر اینطور نباشد، شیء IoT بعدی در نظر گرفته می شود. اگر یک شی IoT مطابقت داشته باشد، اپراتور از مجموعه اپراتورهایی که هنوز نیاز به نقشه برداری دارند حذف می شود (خط 12). سپس تابع FindSolution به صورت بازگشتی فراخوانی می شود.

با انجام این کار، ضمن ارتباط هر اپراتور با هر دستگاه اینترنت اشیا، تمام راه حل های ممکن پیدا می شود. این منجر به زمان اجرا بالاتر در مقایسه با الگوریتم حریصانه می شود. در نتیجه، برای ايجاد نقشه برداری در زمان اجرا یا برای برنامه هایی که نیاز به به روز رسانی مداوم نقشه برداری از اپراتورها دارند، الگوریتم عقبگرد توصیه نمی شود. نگاشت لبه ها مشابه نگاشت اپراتور است که در بالا توضیح داده شد. با این حال، در این مورد، تمام لبه‌ها به عنوان راه‌حل‌های ممکن بر اساس نگاشت موجود از اپراتورها روی دستگاه‌ها بررسی می‌شوند. اگر مجموعه عملگرها و مجموعه یال ها هر دو خالی باشند، همه آنها نگاشت شده و یک راه حل پیدا شد. در این حالت راه حل پيدا شده به مجموعه Solutions (خط 25) اضافه می شود.

پس از خاتمه تابع FindSolution، تابع GetBestSolution در خط 28 در الگوریتم قبلي بهترین راه حل مجموعه کل راه حل ها را تعیین می کند. این تابع تمام راه حل های یافت شده را طی می کند و مجموع فواصل مسیرهای موجود را محاسبه می کند و راه حل با کمترین فاصله بهترین راه حل در نظر گرفته می شود.

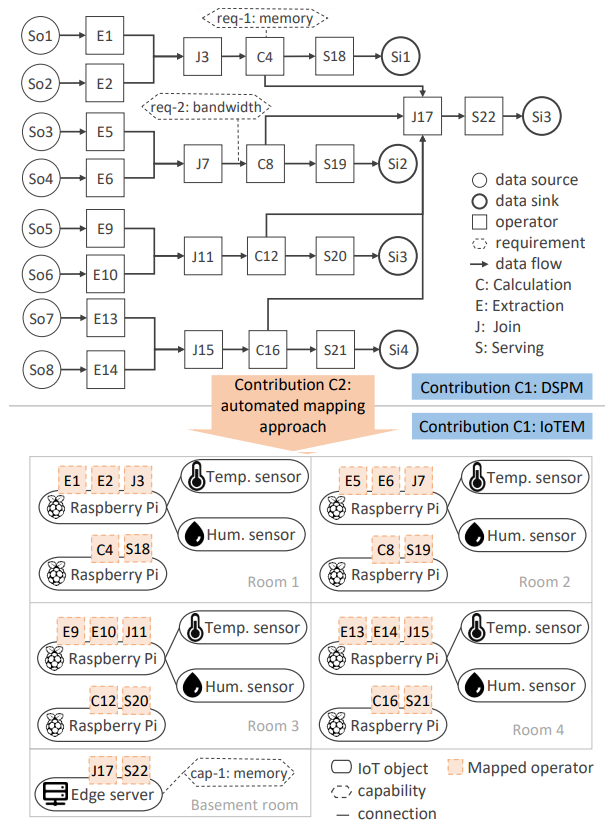
از آنجایی که الگوریتم عقبگرد تمام راه حل های ممکن را محاسبه می کند، زمان اجرا آن نمایی می شود. بنابراین، الگوریتم عقبگرد برای محیط های کوچک اینترنت اشیا و مدل های پردازش جریان داده توصیه می شود. اگر یک شی IoT از زیرساخت ابری در IoTEM مدل شود، الگوریتم از آن به عنوان راه حل بازگشتی استفاده می کند و میتواند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

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

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

## 5-1-3 سناریوی موردی: نظارت بر سطوح قالب در ساختمان های هوشمند

در ادامه، یک سناریوی در حوزه ساختمان هوشمند و نحوه استفاده از مفاهیم نگاشت برای اجرای آن ارائه مي كنيم. 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 محدوده‌های ناسالم افزایش مي رسند. اين سناریو در شکل زير نشان داده شده است.



شكل 7

مطابق شكل برای هر اتاق، سطح قالب به طور مداوم بر اساس مقادیر دما و رطوبت اندازه گیری شده زنده محاسبه می شود. سطوح قالب در طول زمان تجزیه و تحلیل می شوند تا هر گونه افزایش را تشخیص دهند.

ساختمان هوشمند این مطالعه از چهار اتاق و یک اتاق دیگر در زیرزمین تشکیل شده است. هر اتاق دارای دو دستگاه Raspberry Pis به عنوان IoT است که یکی از آنها به یک سنسور دما و یک سنسور رطوبت متصل است. Raspberry Pi دوم هر اتاق در صورت لزوم منابع محاسباتی اضافی را فراهم می کند. علاوه بر این، اتاق زیرزمین مجهز به یک سرور لبه به عنوان یک دستگاه اینترنت اشیا است که نسبت به دیگر Raspberry Pis در ساختمان هوشمند دارای قابلیت های محاسباتی بیشتری است. دستگاه های IoT درگیر به یک شبکه متصل هستند، یعنی میتوانند از طریق پروتکل های استاندارد اینترنت با یکدیگر ارتباط برقرار کنند. در این سناریو، اپراتورهای DSPM باید بین دستگاه‌های IoT موجود در ساختمان توزیع شوند. نوع حریصانه الگوریتم تطبیق به منظور تصمیم گیری برای استقرار عملگرها استفاده می شود.

در مرحله اول، اپراتورهای استخراج، E1 تا E14، بر روی دستگاه‌های IoT مستقر می‌شوند، که حسگرها به طور فیزیکی به آن متصل می‌شوند (همانطور که در شکل نشان داده شده است). این عملگرها شامل اسکریپت هایی برای استخراج داده های حسگر و ارسال آنها به اپراتور پردازشگر بعدی هستند. بنابراین، این اپراتورهای استخراج باید مستقیماً روی دستگاه اینترنت اشیا متصل به سنسورها مستقر شوند. بنابراین، فرض بر این است که منابع کافی در این دستگاه ها موجود است.

پس از نگاشت اپراتورهای استخراج، منابع مورد نیاز آنها از قابلیت های دستگاه های IoT منتخب کم می شود تا در مرحله بعدی تصمیم گیری شود که آیا عملیات اضافی را می توان بر روی این دستگاه های IoT نگاشت یا خیر. پس از آن، مشابه عملگرهای استخراج، اپراتورهای سرویس دهنده 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 Join J7، 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 می‌شود. این دستگاهی است که قبلاً شامل عملگرهای نقشه‌برداری 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 می‌شود 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 اتاق‌های باقی‌مانده، عملگرها به طور مشابه نگاشت می‌شوند.

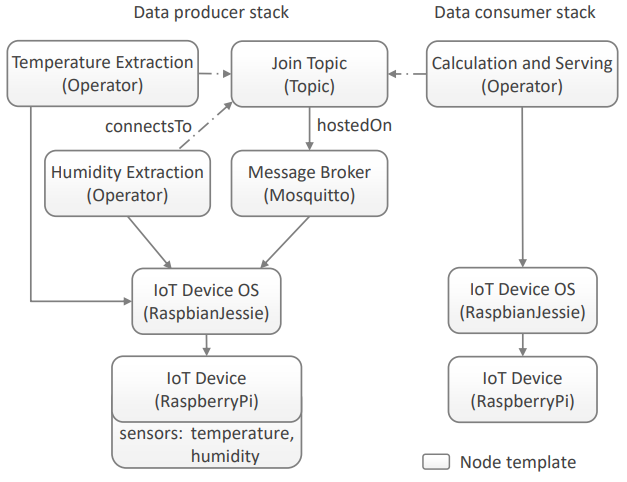
در نهایت، سطح قالب محاسبه‌شده هر چهار اتاق با استفاده از عملگر J17 جمع‌آوری می‌شود، که داده‌های خروجی مناسب برای تجسم داشبورد برای کاربران تولید می‌کند. در این مثال، دستگاه‌های اینترنت اشیا در اتاق‌های مختلف فاصله شبکه یکسانی با یکدیگر دارند. در نتیجه، الگوریتم آنها را به طور تصادفی انتخاب می کند. در مرحله بعد، بررسی می شود که آیا یک دستگاه IoT می تواند الزامات اپراتور 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 به عنوان مثال، مدل های عمومی به کار گرفته شده اند (IoTEM، DSPM)، و علاوه بر این، مجموعه ای از الگوریتم های نگاشت قابل توسعه ارائه شده است، این رویکرد قابل انتقال به روش های دیگر است. دامنه ها نیز مانند دامنه کارخانه هوشمند. این رویکرد همچنین برای چنین سناریوهای بزرگ مقیاس می شود، یعنی الگوریتم ها در نهایت یک راه حل را بر می گردانند.

## 5-2 رويكرد دستي

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 اينجا نشان داده‌ایم که چگونه برنامه‌های نگاشت دستی مبتنی بر TOSCA برای سناریوهای اینترنت اشیا در حوزه خانه هوشمند می‌توانند محقق شوند، و علاوه بر این، چگونه می‌توان آنها را در مرحله بعدی با استفاده از یک موتور استقرار مبتنی بر TOSCA به کار برد. شکل زير یک نگاشت دستی با توپولوژی TOSCA را نشان می دهد



شكل 8

در اين رويكرد اشیاء IoT و منطق پردازش برنامه IoT برای نظارت بر سطوح قالب در یک اتاق تک نشان داده شده است. به عنوان مثال، اشیاء IoT یک IoTEM به عنوان الگوهای گره TOSCA انتزاع می شوند. طرح نگاشت مبتنی بر TOSCA برای سناریوی موردی ارائه شده در بخش 5-1-3 ایجاد شده است که هدف آن نظارت بر سطوح قالب در ساختمان های هوشمند است.

همانطور که در شکل 5.2 در پایین نشان داده شده است. علاوه بر این، عملگرهای پردازش یک DSPM به عنوان الگوهای گره نیز انتزاع شده و در شکل 5.2 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 ديگر مصرف کنندگان داده.

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

1. 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،
2. سیستم عامل دستگاه، به عنوان مثال، یک سیستم عامل میزبانی شده بر روی دستگاه IoT،
3. یک اپراتور که متصل می‌شود. حسگرها، قادر به استخراج داده‌های خود و تحویل آنها، و
4. موضوعی که داده‌ها برای آن منتشر شده و توسط پشته مصرف‌کننده داده قابل دسترسی است.

برای دستگاه هایی که سیستم عامل ارائه نمی دهند، 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 می کند باید ارائه شود، که امکان قرار دادن اپراتور روی آن را فراهم می کند که به چنین دستگاه هایی متصل می شود و داده های حسگرهای تعبیه شده مربوطه را استخراج می کند. مثال شرح داده شده در زیر عمدتاً بر روی دستگاه‌های IoT متمرکز است که یک سیستم عامل را ارائه می‌دهند بطور مثال : Raspberry Pis.

پشته تولید کننده داده در شکل 8 شامل یک دستگاه اینترنت اشیا است که دارای سنسورهای دما و رطوبت و زیرساختی است که دسترسی به این مقادیر حسگر را فراهم می کند. 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 می شوند. 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 می کند، در این مورد، سیستم عامل Raspbian Jessie. برای مدل سازی مناسب چنین سیستم عاملی، باید اطلاعاتی در مورد نوع آن (مثلاً مبتنی بر یونیکس)، نحوه دسترسی به آن (مثلاً استفاده از اتصالات SSH) و اعتبار دسترسی آن، که می تواند از طریق احراز هویت کاربر یا بر اساس آن باشد، ارائه شود. یک کلید SSH بر اساس این سیستم عامل، اسکریپت های اپراتور را می توان به طور خودکار برای دسترسی به رابط های سخت افزاری حسگرها و استخراج داده های آنها مستقر کرد. الگوی گره استخراج دما و الگوی گره استخراج رطوبت به عنوان رابط بین دستگاه فیزیکی اینترنت اشیاء عمل می کنند. به عبارت ديگر Raspberry Pi، و بخشي 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 8 یک دستگاه اینترنت اشیا و زیرساختی برای مصرف، پردازش و نظارت بر مقادیر حسگر از پشته تولید کننده داده را در بر می گیرد. این پشته یک اپراتور محاسبه و سرویس ارائه می دهد که بر روی دستگاه IoT اجرا می شود. این اپراتور برای دریافت مقادیر حسگر و همچنین برای محاسبه سطح قالب فعلی در اتاق بر اساس مقادیر سنسور دما و رطوبت، به واسطه پیام پشته تولید کننده داده مشترک می شود.

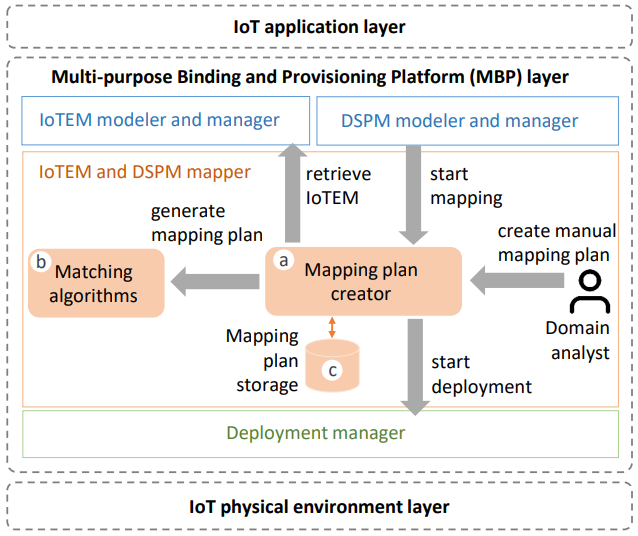
مصرف کنندگان داده به روشی مشابه با تولیدکنندگان داده کار می کنند به این معنا که بر اساس الگوی انتشار-اشتراک ایجاد می کنند. در نتیجه، برای مدل‌سازی آنها با استفاده از TOSCA، می‌توان از انواع گره‌های معرفی‌شده از قبل استفاده مجدد کرد، یعنی انواع Raspberry Pi، RaspbianJessie و Node Operator.

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

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

## 5-3 جزئيات معماری و پیاده سازی IoTEM و DSPM mapper

در معماری کلی (به بخش 3-3 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 9 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 دستي و خودكار انجام دهند:



شکل 9

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

در این مورد، یک مدل‌ساز برای مدل‌های توپولوژی TOSCA، مانند Winery می‌تواند برای ایجاد و دخيره یک نقشه تطبيقي مبتنی بر TOSCA استفاده شود. هنگامی که یک طرح تطبيقي ایجاد شد، استقرار طرح های تطبيقي مبتنی بر TOSCA را می توان 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 تطبيقي و الگوریتم های تطبیق (نوع حریصانه و عقبگرد) در سمت سرور MBP در جاوا توسط Schneider بعنوان پایان نامه کارشناسی انجام شده.

براي ذخیره سازی نقشه ، از پایگاه داده MongoDB استفاده می شود. بخش 8-1 معماری یکپارچه سازی تمام اجزای معماری را ارائه می دهد و نشان می دهد که چگونه آنها با هم تطبیق می یابند.

## 5-4 كارهاي مرتبط

Rizou و همکاران یک الگوریتم ابتكاري را ارائه می دهد که به توزیع اپراتورهای جریان داده اجازه می دهد تا کمترین تأخیر ممکن را برای پردازش داده ها فعال کند. این الگوریتم بهترین توزیع ممکن را جستجو می کند تا تاخیر شبکه را به حداقل برساند.

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

رویکرد در این پایان نامه بیشتر به دنبال یک جفت آزاد بین اشیاء IoT است. در بهترین حالت، اشیاء 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 داده ها در نظر می گیرد، بنابراین، سایر الزامات، همانطور که در این پایان نامه توضیح داده شده است، پشتیبانی نمی شوند.

Cipriani و همکاران رویکردی را برای قرار دادن اپراتور پرس‌و‌جوهای پردازش جریان ارائه می‌کند، که در آن اهداف 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 پایان نامه، M-TOP حاشیه نویسی الزامات را بر روی اپراتورهای پردازش امکان پذیر میکند. این الزامات باید توسط گره‌های محاسباتی موجود (به عنوان مثال، اشیاء اینترنت اشیا) برآورده شوند تا به یک 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 اشیاء سخت افزاری اینترنت اشیا که واجد شرایط گره های محاسباتی هستند نیز در نظر گرفته می شوند.

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

# **فصل ششم**

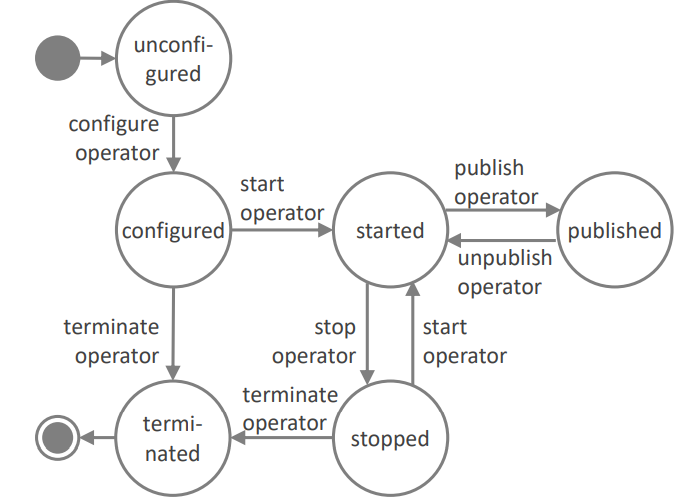
**استقرار اپراتورها در محیط های IoT**

## 6-1 رويكرد خودكار

این رويكرد شامل نمايش اپراتور است که اپراتورهای یک DSPM را به اشیاءIoT مدل اختصاص می دهد. بر اساس نمايش اپراتور، امكان تكرار مي باشد و هر یک از آنها به طور خودکار با استفاده از موتور استقرار مبتنی برTOSCA بنامOpenTOSCA مستقر می شوند. هر اپراتور می تواند به چندین حالت برسد که نشان دهنده وضعیت فعلی استقرار یک اپراتور است. این حالات در ادامه توضیح داده شده است.

### 6-1-1 حالت های استقرار یک اپراتور

این پایان نامه شش حالت استقرار یک اپراتور را تعریف می کند: (i) پیکربندی نشده، (ii) پیکربندی، (iii) شروع شده، (iv) منتشر شده، (v) متوقف شده، (vi) پایان یافته. این حالت های عملگر در شکل 10 نشان داده شده است.



شکل 10

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 طراحي شده به شی IoT اختصاص داده شده کپی می شود. علاوه بر این، وابستگی‌های نرم‌افزاری مورد نیاز (به عنوان مثال، کلاینت MQTT، مفسر پایتون) برای اپراتور بر روی شی IoT اختصاص‌یافته نصب و پیکربندی می‌شوند.

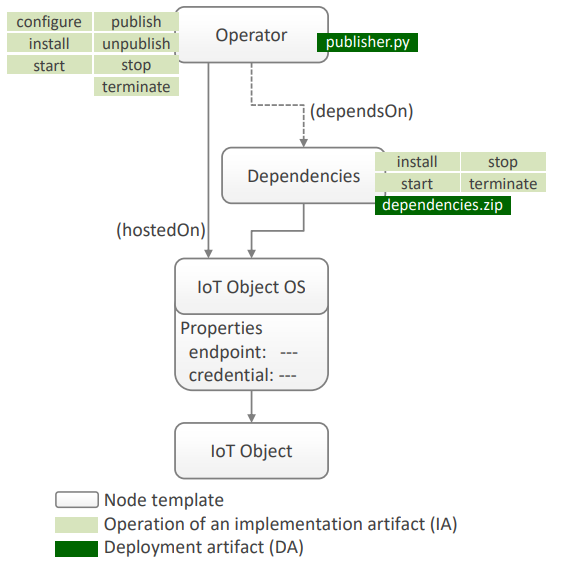
پس از پیکربندی، اپراتور به حالت پیکربندی شده (configured) منتقل می شود. یک اپراتور پیکربندی شده میتواند راه اندازی شود، که هدف اصلی استقرار خودکار است. هنگامی که همه اپراتورها در نقشه برداری اپراتور به حالت شروع رسیدند، استقرار خودکار به عنوان پایان یافته در نظر گرفته می شود و در نتیجه اپراتورهای برنامه 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 6-3 توضیح داده شده است. در نهایت، هنگامی که پردازش جریان داده كنار گذاشته می شود، هر یک از اپراتورهای قبلا مستقر شده باید توسط تحلیلگران دامنه كنار 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 افزارهاي طراحي شده که در شی IoT کپی شده اند حذف می شوند و وابستگی ها حذف می شوند.

### 6-1-2 استقرار اپراتور مبتنی بر TOSCA

شکل 11 یک مدل توپولوژی TOSCA را نشان می دهد



شکل 11

اين 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 افزارهاي طراحي 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.

## 6-2 رویکرد استقرار نیمه اتوماتیک

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

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

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

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

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

استقرار همانطور که در بخش 6-1 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.

## 6-3 زبان توضيح براي اينترنت اشياء (Topic Description Language)

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

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

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

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

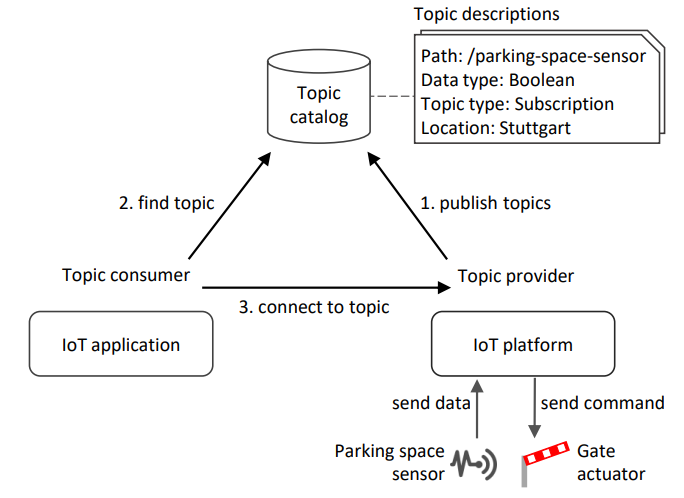
در مدل انتشار-اشتراک مبتنی بر موضوع محقق می شود، مشترکین در موضوعات مرتبط با زمینه خود ثبت نام می کنند تا زمانی که ناشران پیام هایی به این موضوعات ارسال می کنند به طور ناهمزمان مطلع شوند.

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

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

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

رویکرد TDLIoT : (i) یک توصیف جامع از موضوعات، (ii) یک کاتالوگ موضوع برای مرور توضیحات موضوع، و (iii) یک راه موثر برای یافتن موضوعات مناسب که دسترسی به حسگرها و محرک‌ها را به عنوان منابع داده و مخزن‌ها انتزاع می‌کند، ارائه می‌کند. به این ترتیب، توسعه اپلیکیشن IoT می تواند از طریق انتزاع از پلتفرم های خاص اینترنت اشیا آسان شود. موضوعات موجود در کاتالوگ را می توان به عنوان مثال با یک مکان خاص یا نوع حسگر جستجو کرد. رویکرد TDLIoT، که در شکل 12 نشان داده شده است،



شکل 12

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

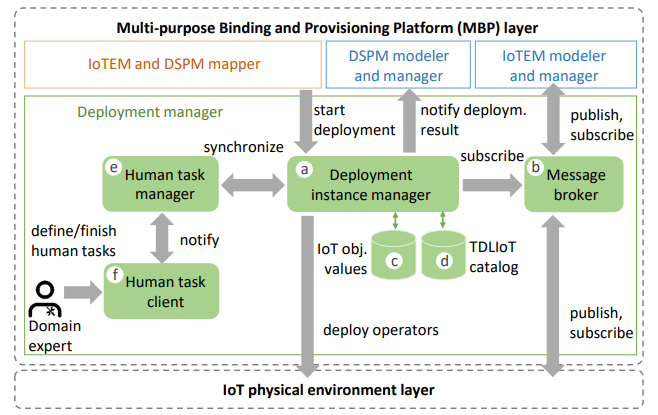
یک TDLIoT باید حداقل دارای ویژگی‌های زیر باشد، با این حال، در صورت لزوم می‌توان آن را با ویژگی‌های بیشتر برای توصیف یک موضوع با جزئیات بیشتر گسترش داد:

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

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

## 6-4 جزئيات معماری و پیاده سازی مدیر استقرار

در معماری کلی بخش 3-3، بخشDeployment manager ابزاری را برای استقرار اپراتورها بر روی اشیاء IoT فراهم می کند. این جزء در شکل 13 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 5-1) یا ایجاد دستی (ر.ک. بخش 5-2) 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 نمونه‌های در حال اجرا به مؤلفه مدل‌ساز و مدیر DSPM (به بخش 4-2-3 مراجعه کنید)، که از موفقیت آمیز بودن اطلاعات مطلع می‌شود، ارسال می‌شوند.



شکل 13

در رویکرد استقرار خودکار، اپراتورها و نرم افزارهای طراحي 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 بخشMessage broker در قسمتb .

بخش Deployment instance manager برای دریافت مداوم مقادیر شی IoT و مقادیر نظارتی که در مخزن مقادیر شیIoT بخش c ذخیره می شوند، با Message broker مرتبط مي شود. توضیحات نحوه دسترسی و تجزیه خروجی اپراتورهای منتشر شده در کاتالوگ TDLIoT 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،Human task manager بخش e اقدامات دستی (به عنوان مثال، وظایف انسانی) را از نقشه تطبيقي 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 خارجی موجود استفاده می کند. OpenTOSCA به عنوان یک مدیر نمونه استقرار خارجی استفاده می شود، که قادر است به طور خودکار اپراتورهای نقشه برداری را بر اساس استاندارد TOSCA مستقر کند.

## 6-5 كارهاي مرتبط

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

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

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

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

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

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

**فصل هفتم**

**نظارت بر DSPM های مستقر شده**

## 7-1 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 كه استخراج محتواي سطح بالا از داده‌های حسگر خام و سطح پایین، انطباق خودکار و تحقق محیط‌های اینترنت اشیاء خودسازمان‌یافته را ممکن می‌سازد. با این حال، چالش‌های زیادی در رابطه با کسب، مدل‌سازی و مدیریت اطلاعات زمینه وجود دارد.

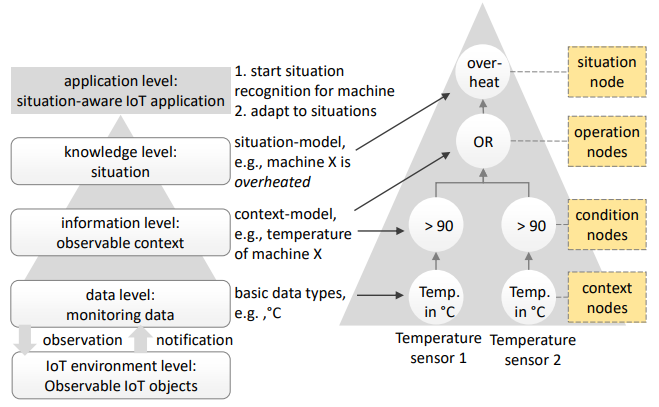
چالش دیگر این است که چگونه می توان به پردازش کارآمد مقادیر زیادی از داده های حسگر و در نتیجه کیفیت خوب دانش زمینه سطح بالا دست یافت. برای این کار، تکنیک‌های تثبیت‌شده‌ای مانند پردازش رویدادهای پیچیده (CEP)، برای پردازش حجم زیادی از داده‌ها لحظه اي معرفی شده‌اند.

این پایان نامه از تعریف ارائه شده توسط Dey و Abowdاستفاده 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 محتواي 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 مي باشد و رویکرد هیرمر و همکاران را گسترش می دهد.

با فعال کردن پردازش مداوم داده‌های حسگر از طریق رویکرد مبتنی بر جریان با استفاده از فناوری‌های CEP. بدین ترتیب، مدل‌سازی و اجرای سناریوهای پیچیده‌تر تشخیص موقعیت فعال می‌شود. این پایان نامه از SitRS XT برای تشخیص اختلالات در DSPM های مستقر در محیط های IoT استفاده می کند. SitRS XT داده های نظارتی را در سطوح مختلف پردازش می کند که در شکل 14 در سمت چپ نشان داده شده است.



شکل 14

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

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

در SATها، حسگرها با گره‌های برگ به نام گره‌های زمینه مطابقت دارند و شاخه‌ها از پایین به بالا از طریق ترکیبی از به اصطلاح گره‌های شرطی و گره‌های عملیاتی تا رسیدن به گره ریشه (یعنی گره موقعیت) جمع می‌شوند. یک مثال ساده از یک الگوی موقعیت در شکل 14 در سمت راست نشان داده شده است. شرایطی را برای تشخیص زمانی که دمای یک ماشین تولیدی از آستانه 90 درجه سانتیگراد عبور می کند، تعریف می کند. گره‌های یک الگوی موقعیت، سطوح پردازش فوق‌الذکر را منعکس می‌کنند: گره‌های زمینه نشان‌دهنده حسگرهایی هستند که یک شی IoT خاص را نظارت می‌کنند که با سطح داده مطابقت دارد. گره های زمینه به گره های شرطی متصل می شوند که روابط داده های حسگر را با اشیاء اینترنت اشیا (سطح اطلاعات) برقرار می کنند. علاوه بر این، گره های شرط، داده های نظارتی را بر اساس شرایط تعریف شده فیلتر می کنند. گره‌های شرطی را می‌توان توسط گره‌های عملیاتی با استفاده از عملیات منطقی جمع‌آوری کرد تا زمانی که به گره موقعیت رسید، که نشان‌دهنده وضعیتی است که باید شناسایی شود.

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

بنابراین، SitRS XT این پیچیدگی را با تبدیل خودکار قالب‌های موقعیت به نمایش‌های اجرایی مورد نیاز، یعنی پرس و جوهای CEP، کاهش می‌دهد. تبدیل الگوهای موقعیت به پرس و جوهای CEP، و در نتیجه، اجرای تشخیص اختلال بر اساس این نمایش های اجرایی در بخش 7-2 توضیح داده شده است. در رویکرد SitRS XT فوق الذکر برای تشخیص اختلال، فرض بر این است که پرس و جوهای CEP حاصل فقط در زیرساخت های فناوری اطلاعات یکپارچه اجرا می شوند. با این حال، برای پردازش کارآمدتر داده ها، رویکردهای بیشتری برای توزیع پرس و جوهای CEP در محیط IoT مورد نیاز است، به طوری که پردازش داده های توزیع شده با مسیرهای ارتباطی کوتاه و کاهش ترافیک شبکه فعال می شود.

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

هدف از این سناریوی موردی، نظارت بر یک مرحله تولید در کف کارگاه یک شرکت تولیدی است تا در اسرع وقت اختلالات در مرحله تولید شناسایی شود. در این مرحله تولید، یک کارگر کف مغازه قطعه فلزی را وارد دستگاه می‌کند که آن را به شکل مورد نیاز برش می‌دهد. در این فرآیند، دو مشکل ممکن است رخ دهد: (1) ابزار برش فلز با گذشت زمان پوسیده می شود، یا (ب) قسمت فلزی به اشتباه در دستگاه قرار می گیرد، به طوری که فلز را نمی توان به درستی برش داد. هر دوی این موارد منجر به یک محصول نهایی اشتباه می شود که گاهی اوقات در اواخر فرآیند تولید کشف می شود. این می تواند باعث هزینه های زیادی شود زیرا بسیاری از مراحل باید تکرار شوند یا حتی محصولات باید دور ریخته شوند. در نتیجه، این موارد خطا باید فوراً شناسایی شوند تا در هزینه ها صرفه جویی شود. در این سناریو، دو منبع داده وجود دارد، (i) یک حسگر موقعیت و (ii) یک حسگر وضعیت ابزار. سنسور موقعیت یک مقدار بولین را برمی گرداند که نشان می دهد آیا قطعه فلزی به درستی در دستگاه قرار گرفته است یا خیر. در صورت درست بودن، قطعه فلزی در موقعیت صحیح قرار دارد. در صورت کاذب، قسمت فلزی در موقعیت اشتباه قرار دارد. سنسور وضعیت ابزار، وضعیت ابزار برش را به عنوان مقدار درصد برمی‌گرداند، که در آن 0٪ به معنای پوسیدگی کامل ابزار است و 100٪ ابزار استفاده نشده است. علاوه بر این، برای تحقق این سناریوی موردی، می توان سه پرس و جو CEP مختلف ایجاد کرد تا بتوان آنها را در مکان های مختلف اجرا کرد. با این حال، موتور CEP که این درخواست‌های CEP را اجرا می‌کند، باید توزیع شود، مانند ارائه شده توسط Cugola و همکاران.

در پرسش CEP Q1 در زير نحوه زبان پردازش رویداد (EPL) نشان داده شده است. این پرس و جو از مقادیر یک سنسور موقعیت به عنوان ورودی خود استفاده می کند و اگر سنسور مقدار false را تولید کند، یک رویداد خروجی ایجاد می کند. برای این کار، طرحی برای جریان رویداد ورودی با نام PositionSensorStream در خط 1 تعریف میشود. به همین ترتیب، طرحی برای جریان رویداد خروجی با نام ProductionErrorStream در خط 2 تعریف می شود. متعاقباً، پرس و جو CEP Q1 در خطوط 4 و 5 تعریف می شود.

1. create schema PositionSensorStream (value boolean);
2. create schema ProductionErrorStream (value boolean);
4. @Name('Q1') insert into ProductionErrorStream
5. Select ∗ from PositionSensorStream (value=false);

در پرس و جوي بعدي CEP Q2 ميتوان زماني شرایط ابزار کمتر از 80 درصد شود، یک رویداد خروجی تولید شود.

1. create schema ConditionSensorStream(value integer);
2. create schema ProductionErrorStream(value boolean);
4. @Name('Q2') insert into ProductionErrorStream
5. Select ∗ from ConditionSensorStream(value<80);

در نهايت پرس و جو CEP Q3 ، به عنوان یک پرس و جو تجمعی عمل می کند که از رویدادهای خروجی تولید شده توسط پرس و جوهای Q1 و Q2 به عنوان ورودی استفاده می کند.

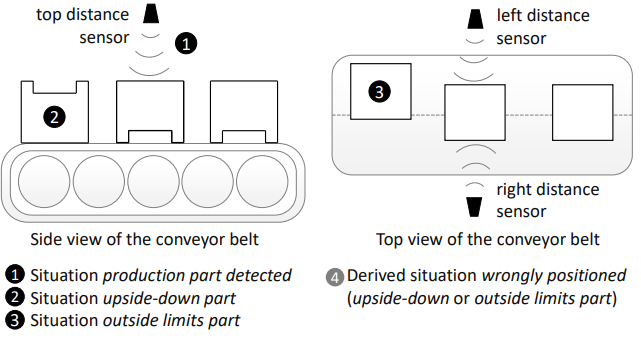
1. create schema ProductionErrorStream(value boolean);
2. @Name('Q3') select count(∗)
3. From ProductionErrorStream.win:time(10sec)
4. Having count(∗)>1

در Q3 بررسی می کند که آیا Q1 یا Q2 بیش از یک رویداد را برای یک پنجره زمانی 10 ثانیه ای ایجاد می کند. پنجره زمانی برای جلوگیری از تشخیص رویدادهای مثبت کاذب به دلیل نقاط پرت استفاده می شود. فقط اگر چندین رویداد در طول این پنجره زمانی تولید شود، می توان مطمئن شد که یک خطای تولید واقعا رخ داده است. علاوه بر این، یک سرویس اعلان وجود دارد که به عنوان سینک داده عمل می کند، که رویدادهای Q3 را دریافت می کند و به فرد مسئول برای مقابله با خطای رخ داده اطلاع می دهد، به عنوان مثال، یک مهندس تعمیر و نگهداری که می تواند قطعه تولید آسیب دیده را جایگزین کند، یا یک کارگر طبقه کارگاه که می تواند قسمت اشتباه را از فرآیند تولید حذف کند. در نهایت، نرم افزار برای اجرای پرس و جوهای CEP همانطور که در فصل 6 توضیح داده شده است، مستقر می شود که به طور خودکار تشخیص اختلال را شروع می کند.

## 7-2 اجرای تشخیص اختلال

این بخش نحوه اجرای تشخیص اختلال را بر اساس مفاهیم توضیح داده شده در بخش 7-1 توضیح می دهد. برای این کار، موقعیت‌هایی که قبلاً به‌عنوان الگوهای موقعیت‌ها مدل‌سازی شده‌اند، ابتدا به نمایش‌های اجرایی و مبتنی بر رویداد، یعنی پرس‌و‌جوهای CEP تبدیل می‌شوند. سپس این نمایش‌ها توسط موتورهای CEP اجرا می‌شوند، که داده‌های نظارتی ورودی را پردازش می‌کنند و زمانی که اختلالات بر اساس نمایش‌های اجرایی شناسایی می‌شوند، اعلان‌هایی ایجاد می‌کنند.

در شکل 15، 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 اين مثال، قطعات تولیدی روی یک تسمه نقاله نظارت می‌شوند تا تشخیص داده شود که چه زمانی اشتباه قرار گرفته‌اند.

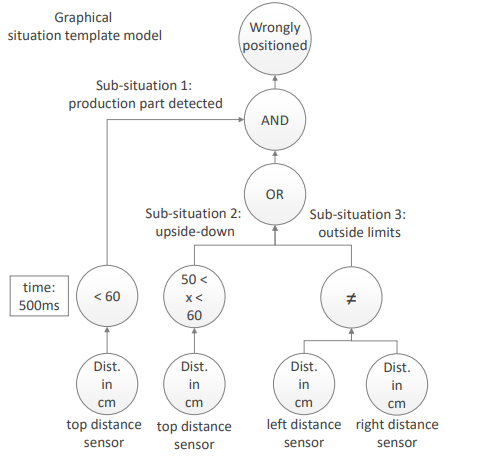


شکل 15

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.

1. وضعيت 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.
2. وضعیت وارونه نشان دهنده یک قطعه تولید وارونه است و در نتیجه اختلالی در روند تولید رخ داده است.
3. 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.
4. 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 15، 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 16

در این مثال، نمایش اجرایی با استفاده از زبان پرس و جو CEP EPL که می تواند توسط موتور Esper CEP اجرا شود، تعریف شده است. با این حال، طیف گسترده‌ای از قالب‌های اجرایی مبتنی بر CEP وجود دارد که می‌تواند توسط این رویکرد استفاده شود.

Select \* From pattern

[ (every A2\_stream=DistanceSensorStream (sensorID='distTop', distance < 60)>(timer:interval (500msec) and not DistanceSensorStream (sensorID='distTop', distance > 60))) and

(( every A1\_stream = DistanceSensorStream (sensorID='distTop',distance>50,distance<60)) or

( every A0\_stream=DistanceSensorStream (sensorID='distLeft') -> (A1\_stream=DistanceSensorStream (sensorID='distRight', A0\_stream.distance != distance) and not distanceSensorStream(sensorID='distLeft'))))]

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

conditionNode\_patternExpression=<sensor\_type\_stream>

(sensor\_id='<monitored\_sensor\_id>',

sensor\_role='<contextNode\_sensor\_role>',

<conditionNode\_condition>)

در شکل 16 در پایین، پرس و جو CEP برای تشخیص موقعیت اشتباه نشان داده شده است. این پرس و جو CEP پیچیده است و بنابراین ایجاد دستی دشوار است. علاوه بر این، پرس و جوهای CEP ممکن است بسته به پیچیدگی موقعیتی که باید تشخیص داده شود، شلوغ شوند. با ارائه تبدیل خودکار از الگوهای موقعیت به پرس و جوهای CEP، متخصصان دامنه از بار ایجاد چنین پرس و جوهای پیچیده CEP خود رها می شوند. در نهایت، تشخیص اغتشاش با استقرار جستجوهای CEP حاصل در موتور اجرای CEP آغاز می شود.

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

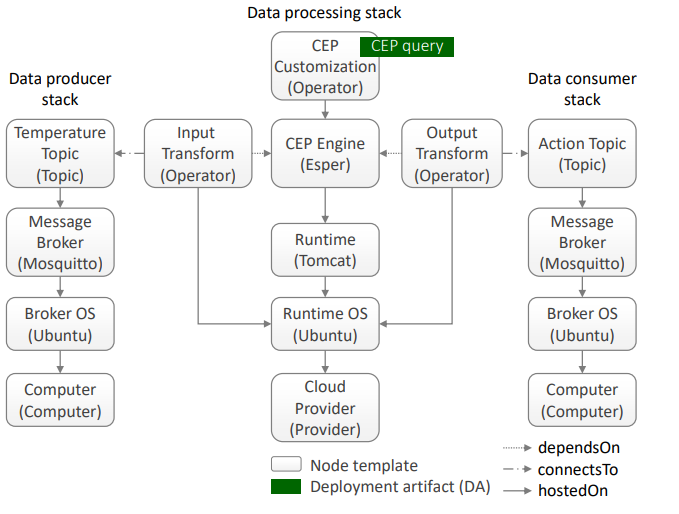
### 7-2-1 سفارشی سازی و تهیه موتورهای CEP

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

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

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

وقتی این مراحل به صورت دستی انجام شوند، خسته کننده و مستعد خطا هستند. برای مقابله با این مسائل، رویکردی را برای سفارشی‌سازی و ارائه موتورهای CEP، از جمله محیط‌های IoT، اپراتورهای مورد نیاز و جستارهای CEP ارائه می‌کنیم. این رویکرد منجر به کاهش زیادی در تلاش سفارشی‌سازی هنگام استفاده از CEP در محیط‌های IoT می‌شود. علاوه بر این، چنین رویکرد سفارشی‌سازی شده، امنیت داده‌ها را از طریق ایجاد نمونه‌های ثابت و غیرقابل تغییر موتورهای CEP، مشابه نماها در پایگاه‌های داده، افزایش می‌دهد. این رویکرد مبتنی بر یک مدل توپولوژی TOSCA مستقل است،مدل توپولوژی باید شامل تمام اجزای نرم افزاری لازم برای راه اندازی موتور CEP باشد. چنین مدل توپولوژی در شکل 17 نشان داده شده است.



شکل 17

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

(1) یک موتور CEP برای پردازش و جمع‌آوری داده‌های ورودی، (2) جستارهای CEP در یک زبان پرس و جو سازگار، که نحوه پردازش داده‌های ورودی را برای محاسبه بالاتر توسط موتور CEP تعریف می‌کند. -موقعیت‌های سطح بر اساس داده‌های سطح پایین، (3) تولیدکنندگان داده، که داده‌های سطح پایین را به موتور CEP ارائه می‌کنند، و (4) مصرف‌کنندگان داده، که موقعیت‌های سطح بالاتر به دست آمده توسط موتور CEP را دریافت و پردازش می‌کنند.

بنابراین، توپولوژی به سه پشته تقسیم می شود: پشته تولید کننده داده، پشته پردازش داده حاوی موتور CEP و پرس و جوهای CEP، و پشته مصرف کننده داده. پشته تولید کننده داده شامل یک دستگاه اینترنت اشیا، یک حسگر دما و زیرساختی است که از طریق یک واسطه پیام به مقادیر حسگر دسترسی پیدا می کند. به عنوان مثال، مقادیر دما توسط یک سنسور متصل به Raspberry Pi اندازه‌گیری می‌شود، که اپراتور را برای استخراج این مقادیر و ارسال آن‌ها به موضوع خاصی که در یک کارگزار پیام میزبانی شده است، اجرا می‌کند. در شکل 17، الگوهای گره برای شی IoT و عملگر استخراج به دلایل ساده حذف شده اند. نمونه ای از پشته تولید کننده داده شامل این الگوهای گره در شکل 8 نشان داده شده است. پشته پردازش داده نشان داده شده در وسط شکل 17 شامل زیرساخت و موتور CEP است که مقادیر حسگر را نظارت و پردازش می کند.

این پشته می‌تواند به دو صورت ایجاد شود: (1) مدل‌سازی موتور CEP که بر روی زیرساخت و اجزای پلت فرم خود میزبانی می‌شود یا (2) 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 17 نشان داده شده است. با شروع از پایین، یک الگوی گره ارائه دهنده مربوط به ارائه دهنده منابع سخت افزاری است که ماشین های مجازی را ایجاد و اجرا می کند. علاوه بر این، قالب گره سیستم عامل Runtime سیستم عامل ماشین مجازی را مدل می کند، در حالی که الگوی گره Runtime زمان اجرا لازم را مدل می کند. G

یک وب سرور برای موتور CEP که در آن مستقر می شود. در این پایان نامه، رویکرد خود میزبانی فرض شده است تا رویکردی را ارائه دهد که عمومی است و به یک ارائه دهنده ابر خاص وابسته نیست. در ادامه، قالب های گره موتور سفارشی سازی شده CEP به تفصیل شرح داده شده است. الگوی گره موتور CEP نشان دهنده خود سیستم CEP است، به عنوان مثال Esper. این الگوی گره، که در قالب گره‌های Runtime میزبانی می‌شود، باید شامل موارد زیر باشد: (1) یک مصنوع پیاده‌سازی (IA) برای نصب، پیکربندی و راه‌اندازی موتور CEP، و (2) یک ویژگی ارائه‌دهنده نقطه پایانی API موتور CEP به برای سفارشی سازی، به عنوان مثال، برای فشار دادن رویدادها، استقرار پرس و جوها، و تعریف انواع رویداد استفاده شود. معمولاً، (3) یک آرتیفکت استقرار (DA) باید ارائه شود که حاوی باینری‌های نصب‌کننده‌های موتور باشد. اگر این DA ارائه نشده باشد، این باینری ها باید به صورت پویا توسط IA بازیابی شوند.

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

به‌طور دقیق‌تر، داده‌ها بر اساس الگوی گره موضوعی مدل‌سازی‌شده ارائه یا مصرف می‌شوند. مثلا. الگوهای گره Temperature Topic و Action Topic در مدل توپولوژی نشان داده شده در شکل 17.

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.

شبیه 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 17 متفاوت است.

قالب‌های گره سیستم‌عامل 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 می توانند 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 می کند، زیرا بر اساس الگوی انتشار-اشتراک نیز ساخته می شود.

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 می توان 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 سیستم های مدیریت پایگاه داده رابطه ای، این رویکرد قادر است درجه قرار گرفتن در معرض داده های اساسی را از طریق این رویکرد سفارشی سازی محدود کند.

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

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

### 7-2-2 كلاس هاي اختلال (Disturbance classes)

با توجه به ماهیت پویای محیط های اینترنت اشیا، ممکن است در صورت بروز اختلال در محیط اینترنت اشیا، پیکربندی مجدد اپراتورهای DSPM مستقر شده نیاز باشد. بخش 7-1 ابزارهای مدل سازی تشخیص اختلال را ارائه می کند. این بخش یک نمای کلی از اینکه کدام اختلالات مربوط به مدل‌سازی و شناسایی در محدوده این پایان‌نامه است را ارائه می‌کند.

این پایان نامه دو کلاس اصلی را برای اختلالات در نظر می گیرد: (1) تغییرات در زیرساخت شبکه محیط IoT، و (2) تغییرات در DSPM اجرا شده در این محیط IoT.

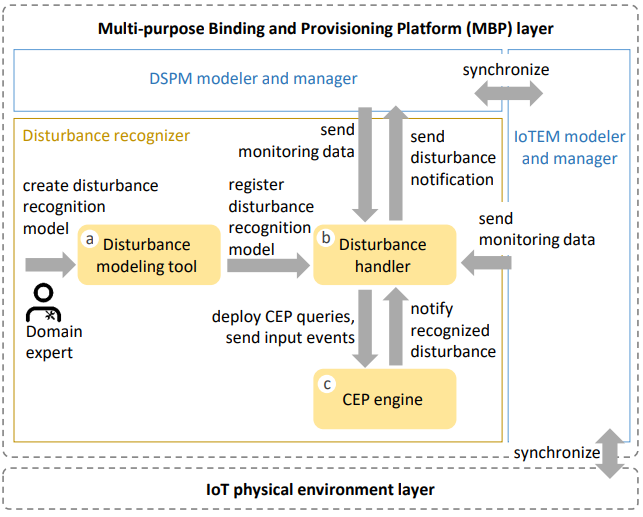
فهرست زیر ليستي از اختلالات را نشان می‌دهد که باید شناسایی شوند، ولي کامل نیست. این اختلالات می تواند از طریق تغییرات در محیط اینترنت اشیا یا در DSPM یک برنامه اینترنت اشیا رخ دهد.

* + تجزیه شی IoT (IoT object breakdown): در صورتی که یک شی IoT کار خود را متوقف کند، اگر شی IoT مربوطه به DSPM مستقر شده (dDSPM) تعلق داشته باشد، مثلا، اگر شی IoT در حال اجرای یک اپراتور باشد یا اگر یک منبع داده باشد، می تواند باعث اختلال شود. برای این، نظارت بر وضعیت اشیاء IoT انجام می شود. چنین مکانیزم‌های نظارتی اعلان‌هایی درباره رسیدن مقادیر بحرانی به آستانه یا خرابی‌های شرایط در اشیاء اینترنت اشیا ایجاد می‌کنند.
  + به روز رسانی شی IoT (IoT object update). اگر قابلیت های به روز شده دیگر هیچ نیاز DSPM را برآورده نکنند، تغییرات در قابلیت ها و ویژگی های یک شی IoT ممکن است منجر به اختلال شود. علاوه بر این، اگر یک شی اینترنت اشیا، به عنوان مثال، یک ویژگی مهم، مانند آدرس IP آن، تغییر کند، ممکن است شی اینترنت اشیا غیر قابل دسترس شود یا جریان داده به خطر بیفتد.
  + کم شدن منابع در اجرا (IoT object running low on resources). اگر یک شی IoT در طول زمان بیش از حد بارگذاری شود یا اگر وضعیت باتری آن کم باشد، باعث شه این گره پردازشی از کار بیفتد، این وضعیت ممکن است منجر به اختلال شود. در این حالت، در صورت امکان، می‌توان به شی اینترنت اشیا منابع بیشتری داد، یا اینکه اپراتور باید به یک شی اینترنت اشیا دیگر نگاشت شود.
  + شی IoT به IoTEM اضافه شده است. هنگامی که یک شی IoT جدید به IoTEM اضافه می شود که در آن یک DSPM قبلاً مستقر شده است، در نتیجه، محیط IoT منابع اضافی را فراهم می کند که می تواند دفعه بعد که نقشه برداری از اپراتورها بر روی اشیاء IoT انجام شود در نظر گرفته شود.
  + پردازش حذف اپراتور. در صورتی که یک اپراتور پردازش، منبع داده یا سینک داده از یک DSPM حذف شود، باید منابع تخصیص یافته آزاد شود و جریان داده باید بررسی تا مطمئن شود که به خطر نیفتاده است.
  + پردازش به روز رسانی اپراتور. تغییرات در الزامات یک اپراتور پردازش می‌تواند منجر به اختلال شود اگر نیازهای جدید یا به‌روزرسانی شده دیگر توسط شی IoT تطبيق شده برآورده نشود.
  + اپراتور پردازش به DSPM اضافه شد. هنگامی که یک اپراتور پردازش جدید، منبع داده یا سینک داده به یک DSPM مستقر شده اضافه می شود، عنصر جدید باید بر روی یک شی IoT مناسب نگاشت شود.

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

## 7-3 جزئيات 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 كه این بخش در شکل 18 نشان داده شده است.



شکل 18

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 16)، که روی موتور بخش (CEP (c مستقر شده‌اند. علاوه بر این، کنترل کننده اختلال داده های نظارتی را در مورد محیط اینترنت اشیا و اپراتورهای پردازش دریافت می کند. این داده های نظارتی به رویدادهای ورودی CEP تبدیل شده و به موتور CEP ارسال می شوند.

هنگامی که موتور CEP یک اختلال را تشخیص داد، به کنترل کننده اختلال اطلاع می‌دهد، که اعلان‌های مربوط به اختلالات را به مؤلفه مدل‌ساز و مدیر DSPM (به بخش 4-2-3 مراجعه کنید) و به داشبورد MBP ارسال می‌کند. جزء تشخیص اختلال به طور نمونه اولیه به عنوان بخشی از MBP پیاده سازی شده است.

بخش 8-2 مروری بر عملکردهای MBP می دهد. 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. کنترل کننده Disturbance در جاوا در سمت سرور MBP پیاده سازی شده است. بخش 8-1 معماری یکپارچه سازی تمام اجزای معماری را ارائه می دهد و نشان می دهد که چگونه آنها با هم تطبیق می یابند.

## 7-4 كارهاي مرتبط

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

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

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

سیستم CEP بر جریان‌های داده مشخص شده نظارت می‌کند و اعلان‌هایی را تولید می‌کند که می‌توانند هنگام وقوع رویداد به مشتریان تحویل داده شوند. در این مقاله، رویدادهای پیچیده به طور مستقیم مشخص نمی شوند، بلکه به عنوان موقعیت ها انتزاع می شوند. رویکرد در این پایان نامه همچنین تبدیل مشخصات رویداد (به عنوان مثال، موقعیت های مدل شده) به پرس و جوهای CEP برای یک سیستم CEP را درک می کند. تفاوت این است که این پایان نامه از هستی شناسی برای مدل سازی موقعیت های مورد علاقه استفاده نمی کند. حسن و همکاران پیشنهاد می کند از CEP همراه با غنی سازی پویا از داده های حسگر به منظور تحقق آگاهی از موقعیت استفاده شود. در این رویکرد، موقعیت های مورد علاقه مستقیماً در موتور CEP تعریف می شوند، یعنی کاربر موقعیت های مورد علاقه را با استفاده از زبان های پرس و جو CEP فرموله می کند.

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

این پایان نامه موقعیت های مورد علاقه را به عنوان الگوهای موقعیت تعریف می کند. 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 جوهای CEP توسط تبدیل ها مراقبت می شود، که به طور خودکار پرس و جوهای CEP لازم را برای یک الگوی موقعیت معین ایجاد می کند. گلومبیفسکی و همکاران رویکرد مشابهی را ارائه می دهد که زمینه را از طیف گسترده ای از منابع برای شناسایی موقعیت با استفاده از فناوری های پردازش رویداد یکپارچه می کند. با این حال، آنها هیچ گونه انتزاعی ارائه نمی دهند، به عنوان مثال، کاربران تشخیص موقعیت باید خودشان جستارهای CEP را ایجاد کنند. این امر به خصوص برای کارشناسان حوزه، به عنوان مثال، در کارخانه ها که دانش گسترده ای در زمینه علوم کامپیوتر ندارند، دشوار است.

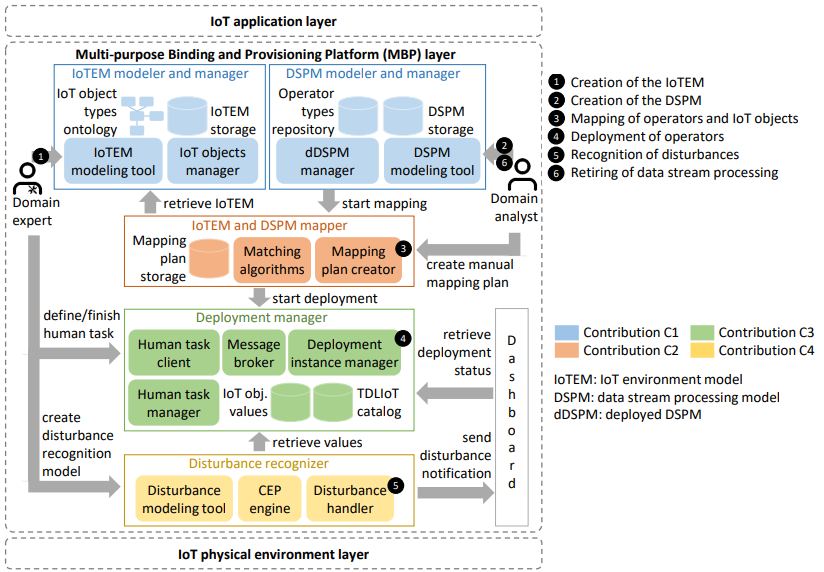
در این پایان نامه، یک انتزاع بر اساس الگوهای موقعیت و یک رابط گرافیکی ارائه شده است که امکان استفاده توسط متخصصان دامنه را بدون دانش لازم از جزئیات فنی، مانند پرس و جوهای پردازش رویداد، فراهم می کند.

**فصل هشتم**

**ارزيابي**

## 8-1 معماری یکپارچه سازی و نمونه اولیه

شكل 19 معماری دقیق حاصل از این پایان نامه را به تصویر می کشد، که اجزای هر دستاورد را ادغام می کند و نشان می دهد که چگونه آنها با هم تطبیق می یابند. دستاورد ها با رنگ برجسته می شوند و مراحل رویکرد روشمند نیز نشان داده شده است.



شكل 19

اجزای معماری و پیاده سازی های مربوط به هر دستاورد به ترتیب در فصل های 4 تا 7 به تفصیل توضیح داده شده است. در معماری کلی (مراجعه کنید به بخش 3-3)، دستاورد C1 شامل دو جزء معماری، مدل ساز و مدیر IoTEM (به بخش 4-1-3 مراجعه کنید)، و مدل ساز و مدیر DSPM (به بخش 4-2-3 مراجعه کنید) را شامل می شود. این اجزا بیشتر به اجزای کوچکتر تقسیم می شوند و در شکل 19 با رنگ آبی نشان داده شده اند. بخش مدلساز و مدیر IoTEM نقطه ورود رویکرد روشمند (مرحله 1) است و از متخصصان دامنه در طول ایجاد و مدیریت IoTEM پشتیبانی می کند.

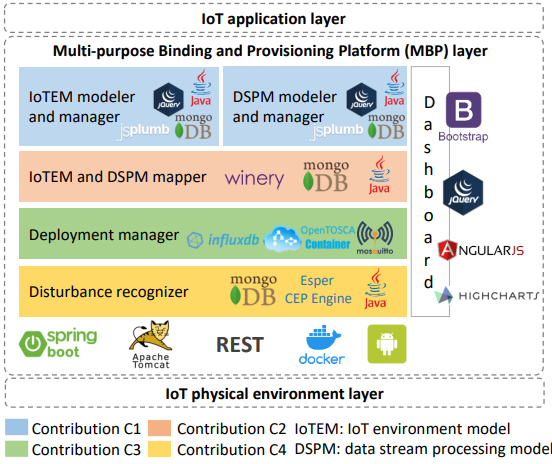
بخش مدل‌ساز و مدیر DSPM با ارائه ابزاری برای ایجاد و مدیریت DSPM (مرحله 2)، و بعداً برای توقف DSPM مستقر در محیط اینترنت اشیا (مرحله 6) از تحلیلگران دامنه پشتیبانی می‌کند. دستاورد C2 شامل اجزای معماری IoTEM و DSPM mapper است که از تحلیلگران دامنه در تصمیم گیری درباره محل استقرار اپراتورهای پردازش پشتیبانی می کند.

این جزء معماری بیشتر به اجزای کوچکتر تقسیم می شود و در شکل 19به 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 3) 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 4).

این جزء با رنگ سبز در شکل 19 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 6-2 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 DSPMهای مستقر شده به منظور شناسایی اختلالات در طول پردازش داده ها (مرحله 5) فراهم می کند. این ابزار یک ابزار مدل‌سازی اختلال را فراهم می‌کند که کارشناسان حوزه می‌توانند آن را مدل‌سازی کرده و شناسایی اختلالات را آغاز کنند. این مؤلفه به طور مداوم مقادیر خروجی اپراتورهای مستقر و مقادیر نظارت بر اشیاء IoT را از مؤلفه معماری مشارکت C3 بازیابی می کند. این مقادیر به موتور CEP ارسال می‌شوند، جایی که بر اساس مدل‌های تشخیص اختلال پردازش می‌شوند. در صورت شناسایی اختلالات، اعلان‌های مربوط به اختلالات به داشبورد ارسال می‌شود، جایی که می‌توان آن‌ها را برای کارشناسان دامنه یا تحلیلگران دامنه نشان داد.

در شکل 20، 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.



شكل 20

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 مدل‌ساز و مدیر IoTEM، ابزار مدل‌سازی IoTEM در جاوا اسکریپت پیاده‌سازی شده است و از کتابخانه JavaScript jQuery استفاده می‌کند. jsPlumb Toolkit برای پیاده‌سازی یک ویرایشگر کشیدن و رها کردن برای مدل‌سازی محیط‌های 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 IoTEM های مدل شده 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 IoTEM گسترش یافت. رابط گرافیکی FlexMash در جاوا اسکریپت 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 نقشه‌بردار IoTEM و DSPM، Winery به‌عنوان ایجادکننده طرح نقشه‌برداری خارجی برای رویکرد نقشه‌برداری دستی استفاده می‌شود.

در این مورد، نقشه های تطبيق 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 اشنایدر پیاده سازی شدند. در این مورد، نقشه های تطبيقي 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 های 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 می شود، مقادیر را به MBP ارسال می کنند. سپس MBP می تواند 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 داده های اندازه گیری، یعنی داده های حسگر و داده های مبتنی بر مهر زمانی، از پایگاه داده سری زمانی InfluxDB استفاده می شود.

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 می کند. این یک رابط وب پیاده سازی شده در جاوا اسکریپت و یک API REST برای تعامل با کاتالوگ TDLIoT فراهم می کند. در رابطه با مولفه تشخیص اختلال، ابزار مدل‌سازی اختلال، در جاوا اسکریپت پیاده‌سازی شده است و از کتابخانه jQuery استفاده می‌کند. مدل‌های تشخیص اختلال ایجاد شده در پایگاه داده MongoDB ذخیره می‌شوند. کنترل کننده اختلال، مدل های تشخیص اختلال را به جستارهای CEP تبدیل می کند، که می تواند 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. می تواند اشیاء IoT را ثبت و حذف کند و مقادیر حسگر فعلی و تاریخی را تجسم کند.

مدل‌سازی اشیاء اینترنت اشیا با MBP2Go می‌تواند با اسکن الگوهای کد QR ارائه شده در مخزن MBP2Go GitHub1 خودکار شود. با اسکن چنین الگوهای کد QR، MBP2Go به طور خودکار ویژگی هایی را پر می کند که امکان استنباط برای نوع شی IoT در حال مدل سازی وجود دارد.

**فصل ششم**

## **مقدمه**

بحث

.

### **6-1 بحث و پیشنهاد‌ها:**

.

**6-1-1 بحث** :

می‌شود .

**6-1-2 پيشنهادها** :

هست.

### **6-2** نتیجه‌گیری :

رایانش

.

# **منابع و مراجع**

**فهرست منابع غیرفارسی و اینترنتی**

1. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, “CryptDB: Processing queries on an encrypted database.” Commun. ACM, vol. 55, no. 9, pp. 103–111, 2012.

**ABSTRACT**

tion.