



دانشکده مهندسی کامپیوتر

ارائه راهکاری برای یکپارچه سازی دستگاه‌های اینترنت اشیاء با سکوی کوبرنیتز

پروژه کارشناسی مهندسی کامپیوتر گرایش مهندسی نرم افزار

سینا شعبانی کومله

استاد راهنما

دکتر محسن شریفی

تابستان ۱۴۰۲

تأییدیه‌ی هیأت داوران جلسه‌ی دفاع از پروژه

نام دانشکده: دانشکده مهندسی کامپیوتر

نام دانشجو: سینا شعبانی کومله

عنوان پروژه: ارائه راهکاری برای یکپارچه سازی دستگاه‌های اینترنت اشیا با سکوی کوبرنیتز

تاریخ دفاع: تابستان ۱۴۰۲

رشته: مهندسی کامپیوتر

گرایش: مهندسی نرم‌افزار

ردیف	سمت	نام و نام خانوادگی	مرتبه دانشگاهی	دانشگاه یا مؤسسه	امضاء
۱	استاد راهنما	دکتر محسن شریفی	استاد تمام	دانشگاه علم و صنعت ایران	
۲	داور نهایی	دکتر TODO	دانشیار	دانشگاه علم و صنعت ایران	

تأییدیه‌ی صحت و اصالت نتایج

باسمه تعالی

اینجانب سینا شعبانی کومله به شماره دانشجویی ۹۷۵۲۱۳۵۱ دانشجوی رشته مهندسی کامپیوتر مقطع تحصیلی کارشناسی تأیید می‌نمایم که کلیه‌ی نتایج این پروژه حاصل کار اینجانب و بدون هرگونه دخل و تصرف است و موارد نسخه‌برداری شده از آثار دیگران را با ذکر کامل مشخصات منبع ذکر کرده‌ام. در صورت اثبات خلاف مندرجات فوق، به تشخیص دانشگاه مطابق با ضوابط و مقررات حاکم (قانون حمایت از حقوق مؤلفان و مصنفان و قانون ترجمه و تکثیر کتب و نشریات و آثار صوتی، ضوابط و مقررات آموزشی، پژوهشی و انضباطی ...) با اینجانب رفتار خواهد شد و حق هرگونه اعتراض در خصوص احقاق حقوق مکتسب و تشخیص و تعیین تخلف و مجازات را از خویش سلب می‌نمایم. در ضمن، مسئولیت هرگونه پاسخگویی به اشخاص اعم از حقیقی و حقوقی و مراجع ذیصلاح (اعم از اداری و قضایی) به عهده‌ی اینجانب خواهد بود و دانشگاه هیچ‌گونه مسئولیتی در این خصوص نخواهد داشت.

نام و نام خانوادگی: سینا شعبانی کومله

تاریخ و امضا:

مجوز بهره‌برداری از پایان‌نامه

بهره‌برداری از این پایان‌نامه در چهارچوب مقررات کتابخانه و با توجه به محدودیتی که توسط استاد راهنما به شرح زیر

تعیین می‌شود، بلامانع است:

☐ بهره‌برداری از این پایان‌نامه برای همگان بلامانع است.

☐ بهره‌برداری از این پایان‌نامه با اخذ مجوز از استاد راهنما، بلامانع است.

☐ بهره‌برداری از این پایان‌نامه تا تاریخ ممنوع است.

استاد راهنما: دکتر محسن شریفی

تاریخ:

امضا:

چکیده

در حال حاضر، مدیریت و نظارت بر دستگاه‌های اینترنت اشیا^۱ به یک چالش عمده تبدیل شده است. راه حل‌های موجود برای کنترل و نظارت بر این دستگاه‌ها اغلب ناسازگاری‌ها و محدودیت‌هایی دارند که موجب کاهش کارایی و پیچیدگی مدیریت در مقیاس بالا می‌شوند. به منظور حل این مسئله، این پروژه تلاش می‌کند تا با استفاده از کوبرنیتز^۲ و پروژه کوبلت مجازی^۳ یک سازوکار جامع برای نظارت و مدیریت دستگاه‌های اینترنت اشیا ارائه دهد. انگیزه اصلی پروژه متمرکز کردن کنترل و نظارت بر دستگاه‌های اینترنت اشیا به صورت یکپارچه و موثر است. راه حل‌های کنونی اغلب ناسازگاری‌هایی با استانداردها و فناوری‌های مختلف دستگاه‌های اینترنت اشیا دارند و به تنهایی قادر به ارائه یک محیط یکپارچه برای مدیریت و نظارت نیستند. این پروژه شامل سه بخش اصلی، یعنی تامین‌کننده^۴، کنترل‌کننده^۵ و دستگاه‌ها^۶ این بخش‌ها با یکدیگر ارتباط برقرار میکنند تا اطلاعات مفیدی درباره دستگاه‌های اینترنت اشیا مورد کنترل ارائه دهند و این اطلاعات را در دسترس خوشه کوبرنیتز قرار دهند.

واژگان کلیدی: اینترنت اشیا، کوبرنیتز، کوبلت مجازی، نظارت یکپارچه، پایش مقیاس پذیر

^۱ Internt Of Things (IoT)

^۲ Kubernetes

^۳ Virtual Kubelet

^۴ Provider

^۵ Controller

^۶ Device

فهرست مطالب

ح فهرست تصاویر

د فهرست جداول

۱ فصل ۱: مقدمه

۱-۱ شرح مسأله ۱

۲-۱ اهداف پروژه ۲

۳-۱ ساختار گزارش ۲

۳ فصل ۲: مفاهیم پایه

۱-۲ مقدمه ۳

۲-۲ بستری ۳

۳-۲ سکوی کوبرنیتز [۱] ۴

۱-۳-۲ کانتینر ۵

۲-۳-۲ پاد ۶

۳-۳-۲ گره ۶

۴-۳-۲ کوبلت ۷

۵-۳-۲ خوشه کوبرنیتز ۸

۴-۲ قرارداد انتقال فرا متن ۸

۱-۴-۲ توابع قرارداد انتقال فرا متن ۹

۲-۴-۲ درخواست قرارداد انتقال فرا متن ۱۰

۱۱	پاسخ قرارداد انتقال فرا متن	۳-۴-۲
۱۲	انتقال حالت نمایشی	۴-۴-۲
۱۳	رابط کاربردی قابل برنامه‌ریزی	۵-۴-۲
۱۴	معماری فن‌آوت	۵-۲
۱۵	کوبلت مجازی	۶-۲
۱۶	معماری کوبلت مجازی	۱-۶-۲
۱۷	اینترنت اشیاء	۷-۲
۱۸	معماری بازخوانی	۸-۲
۱۹	جمع‌بندی	۹-۲
۲۰	کارهای مرتبط	فصل ۳:
۲۰	مقدمه	۱-۳
۲۰	روش‌های متفاوت حل مسئله کنترل و پایش دستگاه‌های اینترنت اشیاء	۲-۳
۲۰	کنترل و پایش دستگاه‌های اینترنت اشیاء بر پایه زنجیره‌ی بلوکی	۱-۲-۳
۲۱	گسترش کوپرنیتز بر روی دستگاه‌های لبه	۲-۲-۳
۲۲	پایش خانه هوشمند به کمک دستگاه‌های اینترنت اشیاء	۳-۲-۳
۲۲	پایش دستگاه‌های اینترنت اشیاء بر پایه کوپرنیتز	۴-۲-۳
۲۴	روش پیشنهادی	فصل ۴:
۲۴	مقدمه	۱-۴
۲۴	معماری سامانه	۲-۴
۲۵	پیاده‌سازی تامین‌کننده کوبلت مجازی	۱-۲-۴
۲۷	رابط برقراری ارتباط با کنترل‌کننده‌ها	۲-۲-۴
۲۷	شبیه‌ساز	۳-۲-۴
۲۸	رابط گرافیکی	۴-۲-۴
۳۰	نحوه کارکرد	۳-۴

فصل ۵: ارزیابی روش پیشنهادی ۳۵

۱-۵ مقدمه ۳۵

۱-۱-۵ ارزیابی و مقایسه معماری‌های ارجح ۳۵

فصل ۶: نتیجه‌گیری و کارهای آینده ۳۷

۱-۶ نتیجه‌گیری ۳۷

۲-۶ دستاوردها ۳۷

۳-۶ کارهای آینده ۳۷

کتاب‌نامه ۳۸

واژه‌نامه انگلیسی به فارسی ۴۰

فهرست تصاویر

۱-۲	معماری کلی کوبرنیتز	۵
۲-۲	نمای کلی درخواست و پاسخ قرارداد انتقال فرا متن	۹
۳-۲	نمای کلی از انتقال حالت نمایشی	۱۳
۴-۲	نمای کلی از رابط کاربردی قابل برنامه ریزی	۱۴
۵-۲	نمای کلی از معماری فن آوت	۱۵
۶-۲	کوبلت مجازی در یک خوشه کوبرنیتز	۱۶
۷-۲	نمای کلی از اینترنت اشیاء	۱۸
۱-۳	پایش مبتنی بر زنجیره ی بلوکی [۲]	۲۱
۲-۳	پایش مبتنی بر کوبرنیتز [۳]	۲۳
۱-۴	نمای کلی معماری	۲۵
۲-۴	گره مجازی	۲۷
۳-۴	رابط کاربردی قابل برنامه نویسی شبیه ساز	۲۸
۴-۴	صفحه اصلی رابط گرافیکی	۲۹
۵-۴	صفحه گره های کوبرنیتز	۲۹
۶-۴	صفحه کنترل کننده های کوبرنیتز کوبرنیتز	۳۰
۷-۴	دستگاه های اینترنت اشیاء ساخته شده در شبیه ساز	۳۲
۸-۴	پاد ساخته شده از روی مستند در وضعیت عدم آمادگی	۳۳
۹-۴	پاد ساخته شده از روی مستند در وضعیت آماده	۳۳
۱۰-۴	تغییر وضعیت قفل مرکزی در رابط گرافیکی	۳۴

۴-۱۱ تغییر وضعیت پاد بدلیل تغییر وضعیت قفل مرکزی ۳۴

فهرست جداول

- ۱-۲ جدول کد پاسخ‌های متداول قرارداد انتقال فرا متن ۱۲
- ۱-۵ نتایج معماری‌های متفاوت بر مجموعه داده FSVQA. اعداد موجود در نام کدگشا نشانگر تعداد لایه‌های آن است. ۳۶

فصل ۱

مقدمه

۱-۱ شرح مسأله

یکی از مسائلی که امروزه در زمینه کنترل و پایش دستگاه‌های اینترنت اشیا وجود دارد، عدم یکپارچگی و هماهنگی میان دستگاه‌های مختلف است. این دستگاه‌ها از فناوری‌ها، پروتکل‌ها و استانداردهای متنوعی برای ارتباط و عملکرد استفاده میکنند، که این تنوع باعث پیچیدگی و مشکلاتی در کنترل و پایش مرکزی آنها میشود. به عنوان مثال، در یک بستر اینترنت اشیا ممکن است دستگاه‌هایی با پروتکل‌های ارتباطی مختلف، مانند HTTP^۱، MQTT^۲ و CoAP^۳ داشته باشند که هر کدام نیازمند روش‌ها و فناوری‌های جداگانه برای کنترل و پایش خود هستند. همچنین، دستگاه‌های اینترنت اشیا ممکن است از نظر تکنولوژی و نوع عملکرد با هم تفاوت داشته باشند. برای مثال، یک سنسور دما و یک قفل هوشمند دارای نیازهای کنترل و پایش متفاوتی هستند. این تنوع در دستگاه‌ها باعث پیچیدگی در توسعه و اجرای یک سیستم کنترل یکپارچه می‌شود. محدودیت منابع نیز یک چالش اساسی در محیط‌های اینترنت اشیا است. این دستگاه‌ها منابع محدودی نظیر پردازشگر، حافظه و پهنای باند شبکه دارند که توان محاسباتی آنها را به شدت کاهش می‌دهد. بنابراین، ضرورت بهره‌برداری بهینه از این منابع و مدیریت آنها به منظور افزایش کارایی و بهره‌وری دستگاه‌ها مطرح میشود. همچنین، امنیت و حفاظت از اطلاعات حساس در محیط‌های اینترنت اشیا نیز از اهمیت بالایی برخوردار است، زیرا این دستگاه‌ها اطلاعاتی حساس را در محیط شبکه منتقل میکنند که به تهدیدات امنیتی از جمله نفوذ، جاسوسی و دسترسی

^۱ Hypertext Transfer Protocol
^۲ Message Queuing Telemetry Transport
^۳ Constrained Application Protocol

غیرمجاز معرض هستند.

۱-۲ اهداف پروژه

هدف اصلی پروژه امکان‌سنجی، طراحی، پیاده‌سازی و ارزیابی سیستمی برای کنترل و پایش دستگاه‌های اینترنت اشیا بر سکوی کوبنیتز است.

۱-۳ ساختار گزارش

در این پروژه هدف ارائه روشی نو برای حل مسئله کنترل و پایش دستگاه‌های اینترنت اشیا بر سکوی کوبنیتز است. در ابتدا به معرفی مفاهیم پایه استفاده شده در این پروژه و سپس به معرفی روش‌ها و کارهای مرتبط پرداخته خواهد شد. پس از آن به معرفی روش و ارزیابی آن پرداخته شده است. در انتها نتیجه‌گیری و کارهای آینده معرفی می‌شوند.

فصل ۲

مفاهیم پایه

۱-۲ مقدمه

در این بخش به معرفی مفاهیم پایه درباره سکوی کوبرنیتز، پروژه کوبلت مجازی، اینترنت اشیاء و برخی معماری‌هایی که در این پروژه استفاده شده‌اند به مانند استخر کارگران^۱ پرداخته شده است.

۲-۲ بستر ابری

بستر ابری^۲ به محیطی اشاره دارد که منابع محاسباتی، شبکه و ذخیره سازی را برای ارائه خدمات به صورت ابری و توسط یک ارائه دهنده ابری فراهم می کند. در این محیط، خدمات و برنامه‌ها بر روی خدمات دهنده‌های فیزیکی مجازی سازی شده قرار می گیرند و کاربران می توانند به آنها از طریق اینترنت وصل شوند و از آنها استفاده کنند. با استفاده از محیط ابری، امکاناتی مانند انعطاف پذیری بالا، قابلیت مقیاس پذیری، اشتراک گذاری منابع و مدیریت آسانتر برای خدمات فراهم می شود. همچنین یکی از مزیت‌های بستر ابری ساده سازی ساخت، مدیریت و انتشار یک خدمت می باشد.

^۱ Worker Pool
^۲ Cloud Environment

۳-۲ سکوی کوبرنیتز [۱]

کوبرنیتز^۳ یک سامانه مدیریت کانتینرها^۴ است که توسط گوگل توسعه داده شده است و در حال حاضر تحت نظارت و پشتیبانی مؤسسه CNCF^۵ قرار دارد. این ابزار به توسعه‌دهندگان و مدیران سامانه امکان می‌دهد برنامه‌ها و خدمات را در بسترهای ابری^۶ مدیریت کنند و کانتینرها را به طور موثر و مقیاس‌پذیر در محیط‌های توزیع شده^۷ مدیریت کنند.

از طریق کوبرنیتز، می‌توان کانتینرها را بر روی یک خدمت دهنده مجازی^۸ یا فیزیکی^۹ اجرا کرده و مدیریت آنها را ساده‌تر و مؤثرتر نمود. این سامانه با بهره‌گیری از روش‌هایی مانند اتوماسیون^{۱۰}، توازن بار^{۱۱} و تشخیص خودکار اشکال^{۱۲}، مدیریت و کنترل بهبود یافته‌ای در محیط‌های مبتنی بر کانتینر فراهم می‌کند. این ابزار برای حل مشکلات زیر موثر است:

۱. مقیاس‌پذیری: کوبرنیتز می‌تواند تعداد کانتینرها و پیش‌نمونه‌های برنامه را بر اساس نیازهای ترافیک و خدمت تنظیم کند. با استفاده از مدیریت منابع مبتنی بر درخواست، میزان منابع مورد استفاده توسط برنامه را به طور خودکار تنظیم می‌کند و مقیاس‌پذیری عمودی و افقی را به راحتی فراهم می‌کند.

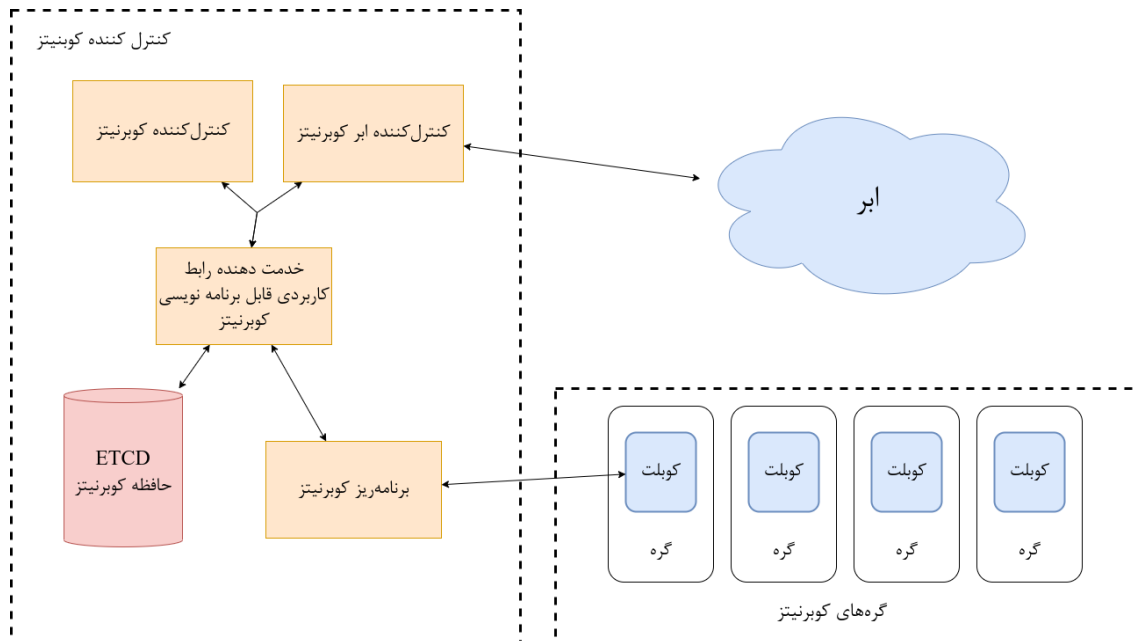
۲. توازن بار: با استفاده از کوبرنیتز، می‌توان بار کار را به طور متوازن بین نودها و خدمت دهنده‌های مختلف تقسیم کرد. این باعث بهبود عملکرد و عدم وقوع اختلال در سامانه می‌شود. همچنین، در صورتی که یک نود یا خدمت دهنده دچار مشکل شود، کوبرنیتز به طور خودکار کار را به سایر نودها منتقل می‌کند.

۳. مدیریت پیچیدگی: کوبرنیتز امکاناتی را برای مدیریت پیچیدگی سامانه‌های کانتینری فراهم می‌کند. این ابزار اجرا، مدیریت، نظارت و زندگی دوباره سازی کانتینرها را ساده می‌کند. همچنین امکاناتی برای مدیریت تنظیمات، آپدیت‌ها، و تغییرات در حال اجرا نیز در اختیار کاربران قرار می‌دهد.

۴. قابلیت انتقال: کوبرنیتز امکان انتقال برنامه‌ها و خدمات بین بسترهای مختلف را فراهم می‌کند. با استفاده از این امکان، می‌توان برنامه‌ها را بین محیط‌های توسعه، آزمون و تولید به راحتی منتقل کرد. کوبرنیتز با استفاده از

Kubernetes^۳Containers^۴Cloud Native Computing Foundation^۵Cloud Environment^۶Distributed Environment^۷Virtual Machine^۸Physical Server^۹automation^{۱۰}Load Balancing^{۱۱}Automatic Diagnostics^{۱۲}

این قابلیت‌ها و خصوصیات، به توسعه‌دهندگان و مدیران سامانه امکان می‌دهد برنامه‌ها را به صورت مؤثر، قابلیت مقیاس‌پذیری و قابل اطمینان در محیط‌های کانتینری مدیریت کنند و عملکرد سامانه را بهبود دهند.



شکل ۲-۱: معماری کلی کوبرنیتز

۲-۳-۱ کانتینر

کانتینرها یک فناوری پیشرفته در زمینه مدیریت و اجرای نرم‌افزار هستند. یک کانتینر، یک واحد نرم‌افزاری است که تمام نیازمندی‌های لازم برای اجرای یک نرم‌افزار را شامل می‌شود. در واقع، کانتینرها مجموعه‌ای از عملیات سامانه‌ای، کدها و تنظیماتی هستند که با یکدیگر در یک بستر محصور می‌شوند. از ویژگی‌های برجسته کانتینرها، می‌توان به استقلال و حمل‌پذیری آن‌ها اشاره کرد. به عبارتی دیگر، یک کانتینر می‌تواند بدون تغییر و با حفظ کارایی خود، بین بسترها و سامانه‌های عامل منتقل شود. این ویژگی باعث شده است که کانتینرها در صنعت فناوری اطلاعات بسیار محبوب شوند. برای مدیریت کانتینرها، ابزارهای مختلفی وجود دارند. یکی از محبوب‌ترین ابزارها برای مدیریت کانتینرها، داکر^{۱۳} است. داکر یک بستر توسعه نرم‌افزار مبتنی بر کانتینر است که به توسعه‌دهندگان امکان می‌دهد تا برنامه‌های خود را در یک

^{۱۳} Docker

کانتینر قرار داده و آن را در هر سامانه‌ای اجرا کنند. با استفاده از کانتینرها، عملیات توسعه، آزمون و استقرار نرم‌افزارها سریع‌تر و ساده‌تر می‌شود. با توجه به این که هر کانتینر دارای محیط مستقلی است، احتمال بروز تداخل بین برنامه‌ها به حداقل می‌رسد و تغییرات در یک کانتینر بر روی سایر کانتینرها تأثیری نمی‌گذارد. همچنین، سبک بودن کانتینرها امکان مقیاس‌پذیری بالایی را فراهم می‌آورد.

۲-۳-۲ پاد

پادها^{۱۴} در کوبرنیتز واحد اصلی اجرا و مدیریت برنامه‌ها و خدمات هستند. یک پاد شامل یک یا چند کانتینر مرتبط است که به صورت مشترک منابع شبکه و ذخیره‌سازی را به اشتراک می‌گذارند. همچنین، هر پاد دارای یک آدرس یکتا درون کلاستر است. پادها به صورت لایه‌ای مجازی شبیه سازی می‌شوند و انتزاعی از یک ماشین مجازی یا سامانه عامل فیزیکی هستند. این انتزاع به برنامه‌ها امکان می‌دهد تا بدون احتیاج به اطلاعات جزئیات بستری که برای اون اجرا می‌شوند، در محیط کنترلی کوبرنیتز اجرا شوند. بنابراین، پادها برای توسعه‌دهندگان و مدیران سامانه، یک واسطه سطح بالا و یک فضای کاری است.

۲-۳-۳ گره

در بستر کوبرنیتز، گره‌ها^{۱۵} از اجزای کلیدی هستند که برنامه‌ها و خدمات در آن‌ها اجرا می‌شوند. یک گره معمولاً یک خدمت دهنده فیزیکی یا ماشین مجازی است که بر روی آن کانتینرها اجرا می‌شوند. هر گره شامل عناصر زیر است:

۱. پلتفرم سخت‌افزاری: این شامل خدمت دهنده‌ها، سامانه‌های فیزیکی، یا ماشین‌های مجازی است که منابع سخت‌افزاری مانند پردازنده، حافظه، و دیسک را فراهم می‌کنند. گره‌ها بسته به نیازهای برنامه‌ها و خدمات، می‌توانند از طریق شبکه به یکدیگر متصل شوند.

۲. کوبلت^{۱۶}: مسئول مدیریت و اجرای کانتینرها در گره است. آن بر روی هر گره نصب شده و با کنترل‌کننده‌های کوبرنیتز برای دریافت توصیف کانتینرها و مدیریت آن‌ها در ارتباط است.

^{۱۴} پاد
^{۱۵} Node
^{۱۶} Kublet

۳. پروکسی^{۱۷}: یک کنترل کننده شبکه است که مسئول مدیریت ترافیک شبکه بین کانتینرها در گره است. این عملکرد به ارتباط و مسیریابی درخواست‌ها بین کانتینرها و اجزای دیگر کوبرنیتز مرتبط است.

۴. حافظه مشترک^{۱۸}: گره‌ها از یک حافظه مشترک برای ذخیره و به اشتراک گذاری اطلاعاتی مانند پیکربندی‌ها و وضعیت گره‌ها استفاده می‌کنند. این حافظه مشترک معمولاً از طریق ابزارهای ذخیره‌سازی مانند ETCD پیاده‌سازی می‌شود.

با استفاده از گره‌ها، کوبرنیتز قادر است برنامه‌ها و خدمات را بر روی یک سری از خدمت دهنده‌ها یا ماشین‌های مجازی توزیع کند و به طور همزمان و مقیاس‌پذیر اجرا کند. این باعث افزایش انعطاف‌پذیری، بهره‌وری و پایداری در محیط‌های ابری و مجازی می‌شود.

۲-۳-۴ کوبلت

کوبلت یکی از اجزای اصلی سامانه مدیریت کانتینرها کوبرنیتز است. کوبلت مسئول اجرا و مدیریت کانتینرها در یک گره می‌باشد. کوبلت در هر گره از خوشه^{۱۹} کوبرنیتز نصب شده و وظیفه‌ای اساسی را بر عهده دارد که شامل موارد زیر است:

۱. مدیریت کانتینرها: کوبلت مسئول ساخت و اجرای کانتینرها بر اساس توصیف‌هایی که از طرف کنترل کننده‌های کوبرنیتز به آن ارسال می‌شود، می‌باشد. این توصیف‌ها شامل اطلاعاتی مانند نرم‌افزار مورد نظر، تنظیمات شبکه و منابع مصرفی کانتینر می‌شوند.

۲. پایش^{۲۰} منابع: کوبلت مسئول نظارت بر منابع مصرفی کانتینرها است و اطلاعات مربوط به استفاده از پردازنده، حافظه، شبکه و دیگر منابع سامانه را جمع‌آوری کرده و گزارش می‌دهد. این اطلاعات به کنترل کننده‌های کوبرنیتز ارسال می‌شوند تا بتوانند به‌طور هوشمند منابع را تخصیص دهند و بهینه‌سازی منابع را انجام دهند.

۳. بروزرسانی و نگهداری کانتینرها: کوبلت مسئول بروزرسانی و نگهداری کانتینرها است. اگر نسخه جدیدی از نرم‌افزار موجود باشد، کوبلت قادر است آن را دریافت و کانتینرها را بروزرسانی کند. همچنین، در صورت خطا در اجرای کانتینر یا توقف آن، کوبلت تلاش می‌کند کانتینر را به‌طور خودکار مجدداً راه‌اندازی کند.

^{۱۷} Kube-proxy

^{۱۸} Shared Memory

^{۱۹} Cluster

^{۲۰} Monitoring

۴. ارتباط با سایر اجزا: کوبلت وظیفه برقراری ارتباط با اجزای دیگر کوبرنیتز را نیز دارد. به عنوان مثال، با کنترل کننده^{۲۱} برای دریافت دستورات مدیریتی، با برنامه ریز^{۲۲} برای دریافت جدول بندی پیشنهادی و با کنترل کننده شبکه^{۲۳} برای تنظیمات شبکه در ارتباط است.

به طور خلاصه، کوبلت یکی از اجزای کلیدی کوبرنیتز است که وظیفه مدیریت و اجرای کانتینرها را در گره های سامانه بر عهده دارد. این کامپوننت از طریق ارتباط با سایر اجزا و دریافت توصیف های مربوطه، به ایجاد و مدیریت یک محیط توزیع شده و مقیاس پذیر برای اجرای برنامه ها و خدمات در کوبرنیتز کمک می کند.

۵-۳-۲ خوشه کوبرنیتز

یک خوشه کوبرنیتز^{۲۴} یک بستر توزیع شده است که شامل مجموعه ای از گره ها است که برای مدیریت و اجرای برنامه ها و ارائه خدمات از طریق کوبرنیتز استفاده می شود. خوشه کوبرنیتز شامل اجزا و خدماتی متعددی است که با همکاری میان گره ها، برنامه ها را مدیریت می کنند.

۴-۲ قرارداد انتقال فرا متن

قرارداد انتقال فرا متن^{۲۵} یک پروتکل ارتباطی است که در اینترنت استفاده می شود و برای انتقال اطلاعات بین خدمت دهنده^{۲۶} و خدمت گیرنده^{۲۷} استفاده می شود. به طور کلی، قرارداد انتقال فرا متن به عنوان روشی برای انتقال اطلاعات و محتوا در وب مورد استفاده قرار می گیرد. در یک ارتباط، HTTP خدمت گیرنده درخواستی به خدمت دهنده می فرستد و سپس خدمت دهنده با پاسخ مناسب به درخواست خدمت گیرنده پاسخ می دهد. این درخواست و پاسخ در قالب پیام های متنی انجام می شود، که ممکن است شامل سرآیندها^{۲۸} و محتوای پیام^{۲۹} باشند. سرآیندها شامل اطلاعاتی مانند نوع محتوا، تاریخ، طول پیام و سایر جزئیات مربوط به ارتباط است. قرارداد انتقال فرا متن برای انتقال انواع مختلف منابع و اطلاعات در وب استفاده می شود. مثلاً می توان از قرارداد انتقال فرا متن برای دریافت صفحات وب، تصاویر، ویدیوها و

^{۲۱}kube-controller-manager

^{۲۲}kube-scheduler

^{۲۳}kube-proxy

^{۲۴}Kubernetes Cluster

^{۲۵}Hypertext Transfer Protocol (HTTP)

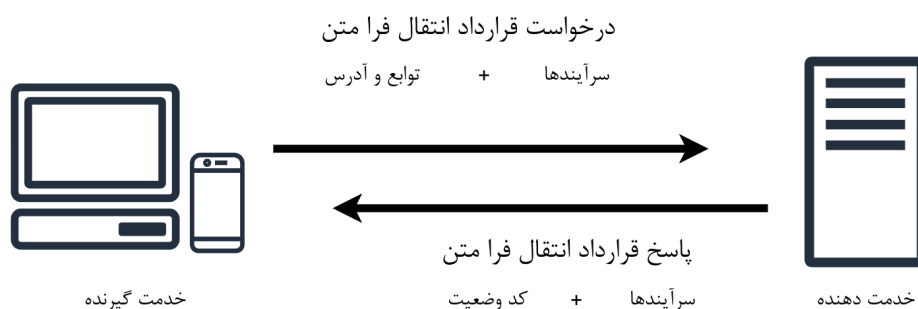
^{۲۶}Server

^{۲۷}Client

^{۲۸}Header

^{۲۹}Body

سایر منابع در خدمت دهنده استفاده کرد. همچنین، قرارداد انتقال فرا متن از مدل درخواست-پاسخ پیروی می‌کند، به این معنی که خدمت گیرنده درخواستی ارسال می‌کند و خدمت دهنده با یک پاسخ مناسب به آن پاسخ می‌دهد. قرارداد انتقال فرا متن اساسی است در عملکرد وب و تعامل بین سرویس دهنده و خدمت گیرنده. این پروتکل به صورت گسترده در نرم افزارها و سرویس های وب مورد استفاده قرار می‌گیرد و امکان انتقال اطلاعات و ارتباط بین کامپیوترها و دستگاه ها را فراهم می‌کند.



شکل ۲-۲: نمای کلی درخواست و پاسخ قرارداد انتقال فرا متن

۴-۲-۱ توابع قرارداد انتقال فرا متن

قرارداد انتقال فرا متن توابع^{۳۰} مختلفی را برای تعیین نوع عملیاتی که باید انجام شود تعریف کرده است که عبارت است از:

۱. دریافت^{۳۱}: این تابع برای دریافت (خواندن) اطلاعات از یک منبع مشخص درخواست می‌شود. مثلاً با استفاده از تابع دریافت می‌توانید صفحات وب، تصاویر یا سایر منابع را از خدمت دهنده دریافت کنید. درخواست دریافت، بدون تغییر و اثری روی منبع مورد درخواست، انجام می‌شود.

۲. ساخت^{۳۲}: این تابع برای ارسال داده‌ها به خدمت دهنده استفاده می‌شود. از تابع ساخت برای ایجاد یا ارسال داده جدید به خدمت دهنده استفاده می‌شود، مانند ارسال فرم‌ها در وب، ارسال نظرات یا انجام عملیات‌های پردازشی.

^{۳۰}Methods متن فرا انتقال قرارداد
^{۳۱}GET
^{۳۲}POST

درخواست ساخت می‌تواند تغییراتی روی منبع مورد درخواست ایجاد کند.

۳. بروزرسانی^{۳۳}: این تابع برای به‌روزرسانی (بازنویسی) یک منبع مشخص استفاده می‌شود. با استفاده از تابع بروزرسانی، می‌توانید اطلاعات موجود در خدمت دهنده را با داده‌های جدید جایگزین کنید. درخواست بروزرسانی می‌تواند تغییراتی روی منبع مورد درخواست اعمال کند یا در صورت عدم وجود، یک منبع جدید ایجاد کند.

۴. حذف^{۳۴}: این تابع برای حذف یک منبع مشخص استفاده می‌شود. با استفاده از تابع حذف می‌توانید یک منبع را از خدمت دهنده حذف کنید. درخواست حذف، منبع مورد نظر را از خدمت دهنده حذف می‌کند.

۵. بروزرسانی مقطعی^{۳۵}: این تابع برای اعمال تغییرات جزئی روی یک منبع مشخص استفاده می‌شود. با استفاده از تابع بروزرسانی مقطعی، می‌توانید تغییرات کوچکی را روی یک منبع اعمال کنید بدون ایجاد تغییرات بزرگتری در داده‌های موجود.

این توابع به عنوان پایه‌های قرارداد انتقال فرا متن عمل می‌کنند و به خدمت گیرنده و خدمت دهنده امکان ارتباط و انجام عملیات‌های مختلف را می‌دهند. درخواست‌ها و پاسخ‌های قرارداد انتقال فرا متن بر اساس این توابع تعریف می‌شوند و برای تعامل موثر بین سرویس دهنده و خدمت گیرنده استفاده می‌شوند.

۲-۴-۲ درخواست قرارداد انتقال فرا متن

در قرارداد انتقال فرا متن، وقتی که خدمت گیرنده (مانند مرورگر وب یا برنامه‌ای که درخواست ارسال می‌کند) برای دستیابی به منبع مورد نظر خود، درخواستی^{۳۶} ایجاد می‌کند، یک درخواست قرارداد انتقال فرا متن ایجاد می‌شود. درخواست‌های قرارداد انتقال فرا متن شامل اطلاعاتی درباره نوع و منبع درخواست شده هستند. در زیر توضیحی از مهمترین اجزای یک درخواست قرارداد انتقال فرا متن را می‌یابید:

۱. توابع قرارداد انتقال فرا متن: تابع مشخص می‌کند خدمت گیرنده درخواست انجام چه عملیاتی را بر روی منبع مورد نظر دارد.

PUT^{۳۳}
DELETE^{۳۴}
PATCH^{۳۵}
HTTP Request^{۳۶}

۲. درس^{۳۷}: مشخص می‌کند که منبع مورد نظر کجا قرار دارد و آدرس دقیق آن چیست. آدرس شامل پروتکل، نام دامنه خدمت دهنده^{۳۸} و مسیر^{۳۹} منبع مورد نظر است.

۳. سرآیندها: درخواست قرارداد انتقال فرا متن می‌تواند شامل سرآیندها باشد که اطلاعات اضافی درباره درخواست و مشخصات خدمت گیرنده را ارائه می‌دهند. به عنوان مثال، سرآیندها می‌توانند شامل اطلاعات مربوط به نوع محتوا، زبان مورد نظر، تاریخ و غیره باشند.

۴. بدنه درخواست^{۴۰}: بدنه درخواست، در صورتی که درخواست ارسالی دارای داده‌های ارسالی است، این داده‌ها را در خود نگه می‌دارد. بدنه درخواست به صورت متنی یا در قالب داده‌های باینری می‌تواند باشد و معمولاً در درخواست‌هایی مانند ساخت و بروزرسانی استفاده می‌شود.

هنگامی که خدمت گیرنده درخواست قرارداد انتقال فرا متن را به خدمت دهنده ارسال می‌کند، خدمت دهنده با استفاده از اطلاعات درخواست، منبع مورد نظر را پیدا کرده و به مناسبت درخواست، پاسخ مناسب را به خدمت گیرنده ارسال می‌کند. درخواست‌های قرارداد انتقال فرا متن بسته به نوع و عملیات مورد نظر، تعیین می‌کنند که چه عملیاتی باید در سمت خدمت دهنده انجام شود و اطلاعات مربوطه را برگردانند.

۲-۴-۳ پاسخ قرارداد انتقال فرا متن

هنگامی که یک درخواست از سوی خدمت گیرنده به خدمت دهنده ارسال می‌شود، خدمت دهنده با یک پاسخ^{۴۱} مناسب به آن درخواست پاسخ می‌دهد. پاسخ‌های قرارداد انتقال فرا متن حاوی اطلاعات مربوط به نتیجه درخواست و وضعیت آن است.

۱. کد وضعیت^{۴۲}: هر پاسخ قرارداد انتقال فرا متن دارای یک کد وضعیت است که نشان می‌دهد درخواست با موفقیت انجام شده است یا با مشکل مواجه شده است. برخی از کدهای وضعیت رایج عبارتند از:

^{۳۷} Uniform Resource Locator (URL)

^{۳۸} Domain Name

^{۳۹} Path

^{۴۰} Request Body

^{۴۱} HTTP Response

^{۴۲} Status Code

کد پاسخ	شرح
۲۰۰	درخواست با موفقیت انجام شده است و پاسخ داده مورد نظر در دسترس است
۲۰۱	منبع مورد نظر با موفقیت ساخته شد
۲۰۴	پاسخ بدون بدنه
۴۰۴	منبع مورد نظر پیدا نشد
۵۰۰	مشکلی در سمت خدمت دهنده رخ داده است که موجب عدم توانایی در ارسال پاسخ مورد نظر شده است

جدول ۲-۱: جدول کد پاسخ‌های متداول قرارداد انتقال فرا متن

۲. سرآیندها: پاسخ قرارداد انتقال فرا متن شامل سرآیندها است که اطلاعات اضافی در مورد پاسخ و منبع مورد نظر را ارائه می‌دهند. به عنوان مثال، سرآیندها می‌توانند شامل اطلاعات مربوط به نوع محتوا، طول پاسخ، تاریخ ارسال، و غیره باشند.

۳. محتوای پاسخ^{۴۳}: پاسخ قرارداد انتقال فرا متن ممکن است حاوی محتوای مورد نظر باشد که توسط خدمت دهنده ارسال می‌شود. محتوای پاسخ می‌تواند اطلاعاتی مانند متن، تصویر، ویدیو یا سایر منابع مورد نیاز را شامل شود.

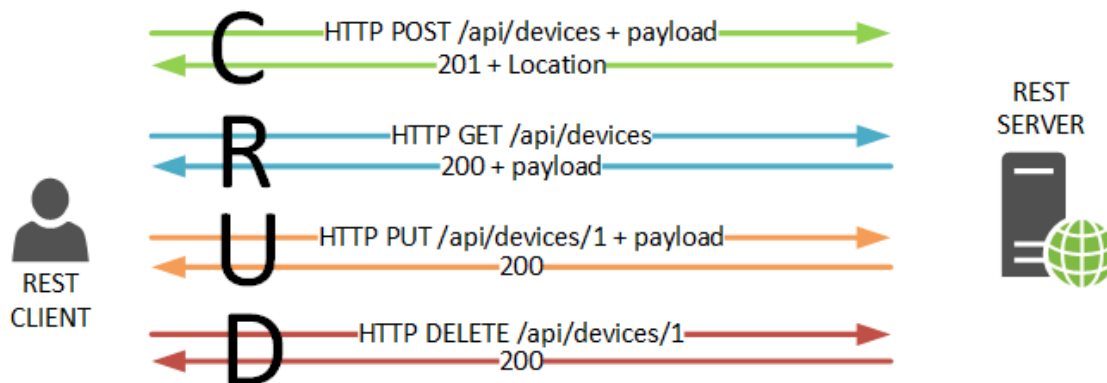
پاسخ‌های قرارداد انتقال فرا متن ارسال شده توسط خدمت دهنده معمولاً به درخواست‌های ارسالی از سوی مشتری پاسخ می‌دهند و اطلاعات مورد نیاز را در اختیار مشتری قرار می‌دهند. این پاسخ‌ها با کدهای وضعیت، سرآیندها و محتوای پاسخ کامل شده و تعیین می‌کنند که درخواست با موفقیت انجام شده است یا خطا رخ داده است.

۴-۴-۲ انتقال حالت نمایشی

انتقال حالت نمایشی^{۴۴} یک معماری نرم‌افزاری است که برای طراحی سامنه‌های توزیع‌شده و مبتنی بر وب استفاده می‌شود. انتقال حالت نمایشی بر اساس مجموعه‌ای از اصول و محدودیت‌ها ساختاردهی شده است که در تبادل اطلاعات بین خدمت دهنده و خدمت گیرنده نقش مهمی دارد. یکی از اصول اساسی انتقال حالت نمایشی، تعریف یکپارچگی^{۴۵} برای سامنه است. این به این معنی است که سامنه باید یک مجموعه‌ی مشخص از روش‌های استاندارد را برای تعاملات در نظر بگیرد. معمولاً در سامنه‌های انتقال حالت نمایشی، از متدهای قرارداد انتقال فرا متن مانند دریافت، ساخت، بروزرسانی و حذف برای تعیین عملیات مورد نیاز استفاده می‌شود. همچنین، منابع^{۴۶} در سامنه انتقال حالت نمایشی به صورت یکتا شناسایی می‌شوند و آدرس‌های مشخصی برای آنها تعیین می‌شود. با توجه به این اصول و محدودیت‌ها، سامنه‌های انتقال

^{۴۳} Response Body^{۴۴} Representational State Transfer (REST)^{۴۵} Uniform Interface^{۴۶} Resources

حالت نمایشی قابلیت‌هایی مانند قابلیت توزیع، قابلیت مقیاس‌پذیری، انعطاف‌پذیری و قابلیت استفاده مجدد را فراهم می‌کنند. از طریق روش‌های استاندارد و اصول انتقال حالت نمایشی، سامانه‌ها قادر به تعامل با یکدیگر بدون توجه به جزئیات داخلی و پیچیدگی‌های نحوه پیاده‌سازی هستند.



شکل ۲-۳: نمای کلی از انتقال حالت نمایشی

۵-۴-۲ رابط کاربردی قابل برنامه‌ریزی

رابط کاربردی قابل برنامه‌ریزی^{۴۷} به قوانین، پروتکل‌ها و دستوراتی گفته می‌شود که برای ارتباط و تعامل بین نرم‌افزارها، خدمات و برنامه‌های مختلف به کار می‌رود. به طور کلی، رابط کاربردی قابل برنامه‌ریزی نقش یک میانجی بین سامانه‌ها را بازی می‌کند و به برنامه‌نویسان اجازه می‌دهد با استفاده از آن، به منابع و امکانات موجود در سامانه دیگر دسترسی پیدا کنند. رابط کاربردی قابل برنامه‌ریزی‌ها می‌توانند در دو شکل مختلف عمل کنند: به صورت وب خدمت یا به صورت کتابخانه برنامه‌نویسی. در حالت وب خدمت، رابط کاربردی قابل برنامه‌ریزی بر روی یک خدمت دهنده میزبان شده است و از طریق پروتکل‌های اینترنتی مانند قرارداد انتقال فرامتن قابل دسترسی است. در حالت کتابخانه برنامه‌نویسی، دستورات و توابع مشخصی به برنامه اضافه می‌شوند که برنامه‌نویسان می‌توانند از آنها به عنوان قسمتی از برنامه خود استفاده کنند. استفاده از رابط کاربردی قابل برنامه‌ریزی‌ها به برنامه‌نویسان امکان می‌دهد که بخشی از دستورات را استفاده کنند و از

خدمات و قابلیت‌های ارائه شده توسط یک سامانه دیگر بهره ببرند. این راهکار می‌تواند زمان و هزینه توسعه برنامه را کاهش داده و امکان ادغام بین نرم‌افزارها را فراهم کند. به طور خلاصه، رابط کاربردی قابل برنامه‌ریزی، مانند یک پل ارتباطی است که برنامه‌نویسان می‌توانند از آن استفاده کنند تا بین نرم‌افزارها و خدمات اطلاعات را به اشتراک بگذارند و تعامل کنند.



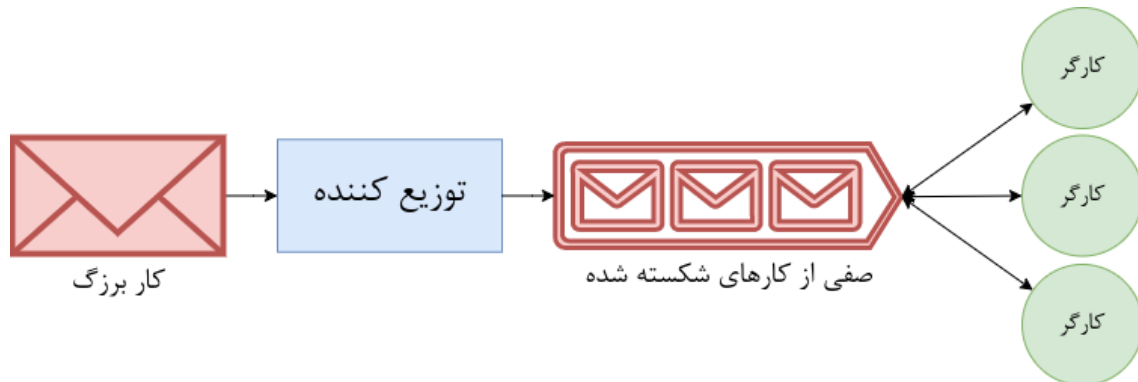
شکل ۲-۴: نمای کلی از رابط کاربردی قابل برنامه‌ریزی

۲-۵ معماری فن‌آوت

معماری فن‌آوت^{۴۸}: الگوی طراحی است که به طور معمول در سامانه‌های توزیع شده برای مدیریت همزمان یا موازی درخواست‌ها استفاده می‌شود. این معماری به توزیع درخواست‌های ورودی به چند واحد پردازش، که به عنوان کارگرها شناخته می‌شوند، برای انجام عملیات مورد نیاز می‌پردازد. معماری فن‌آوت با بهره‌گیری از پردازش موازی و توازن بار، امکان مقیاس‌پذیری و بهبود عملکرد سیستم را فراهم می‌کند. در معماری فن‌آوت، هنگامی که یک درخواست دریافت می‌شود، آن را به چندین کارگر مستقل تکرار یا ارسال می‌کنند تا هر کدام از آن‌ها قادر به پردازش درخواست به صورت

^{۴۸}Fan-out

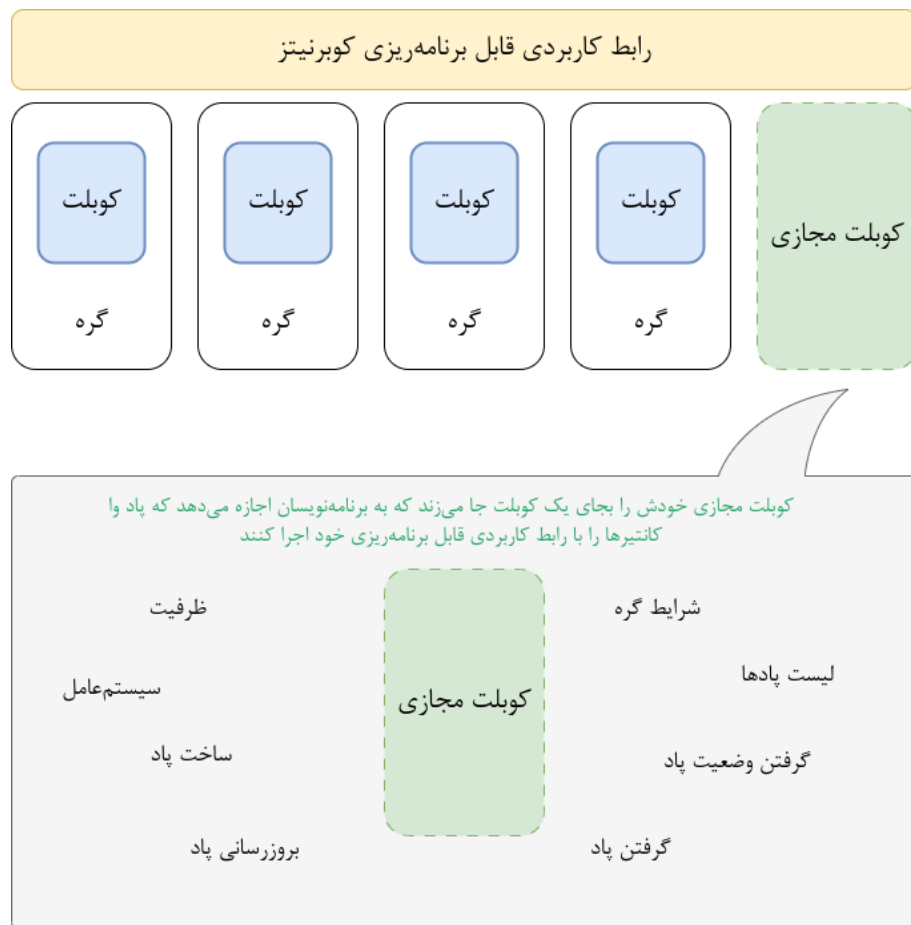
مستقل باشند. این رویکرد به چندین کارگر اجازه می‌دهد تا به صورت همزمان روی بخش‌های مختلف درخواست کار کنند و بهبود ظرفیت تولید و کاهش زمان پاسخ را به همراه داشته باشد.



شکل ۲-۵: نمای کلی از معماری فن‌آوت

۶-۲ کوبلت مجازی

کوبلت مجازی^{۴۹} یک پروژه متن باز است که با گسترش رابط کاربردی قابل برنامه‌ریزی کوبرنیتز، امکان ادغام^{۵۰} خدمات و سامانه‌های خارجی را در قالب گره‌های کوبرنیتز فراهم می‌کند. این پروژه این امکان را می‌دهد که یک گره "مجازی" بسازیم که توسط کوبرنیتز قابل مدیریت باشد و امکان ادغام سامانه‌های متنوع را در داخل خوشه کوبرنیتز به صورت یک پارچه فراهم کند.



شکل ۶-۲: کوبلت مجازی در یک خوشه کوبرنیتز

۱-۶-۲ معماری کوبلت مجازی

کوبلت مجازی از چندین بخش کلیدی تشکیل شده است:

۱. تامین‌کننده: تامین‌کننده مسئول پیاده‌سازی رابط کوبلت مجازی است و به عنوان پل ارتباطی بین سامانه یا خدمت خارجی و کوبرنیتز عمل می‌کند. این بخش، فراخوانی‌های رابط کاربردی قابل برنامه‌ریزی کوبرنیتز را به عملیات مناسب در سامانه خارجی ترجمه می‌کند.

۲. گره: گره مجازی نمایانگر یک سامانه یا خدمت خارجی در خوشه کوبرنیتز است. عملکرد آن شبیه یک گره کوبرنیتز عادی است، اما به جای اجرا در زیرساخت فیزیکی، از طریق تامین‌کننده با سامانه خارجی ارتباط برقرار می‌کند.

۳. یاد: یاد در کوبرنیتز، شامل یک یا چند کانتینر است. در کوبلت مجازی، پادها نماینده کارها^{۵۱} هستند که در گره‌های مجازی ایجاد شده توسط سامانه خارجی اجرا می‌شوند.

۴. برنامه‌ریز^{۵۲}: برنامه‌ریز کوبرنیتز مسئول تخصیص پادها به گره‌های موجود بر اساس نیازهای منابع و محدودیت‌ها است. در کوبلت مجازی، برنامه‌ریز مسئول زمانبندی پادها به گره‌های مجازی ایجاد شده توسط تامین‌کننده است. این بخش تضمین می‌کند که منابع بصورت بهینه استفاده شده و کارها به طور مناسب در سامانه خارجی توزیع شوند.

۷-۲ اینترنت اشیاء

اینترنت اشیاء^{۵۳} به مجموعه‌ای از دستگاه‌ها، سنسورها، دستگاه‌های هوشمند و شبکه‌های مرتبط که قادر به تبادل اطلاعات با یکدیگر از طریق اتصال به اینترنت هستند، اشاره دارد. این اشیاء می‌توانند شامل تلفن همراه و ساعت هوشمند تا لوازم خانگی هوشمند، خودروهای متصل و تجهیزات صنعتی باشند. اینترنت اشیاء با اتصال اشیاء و جمع‌آوری اطلاعات، امکان برقراری ارتباط و کنترل بیشتری را بین دنیای فیزیکی و دنیای دیجیتال فراهم می‌کند. مزیت اصلی اینترنت اشیاء در جمع‌آوری و تبادل داده‌ها است. سنسورها و دستگاه‌ها در اینترنت اشیاء می‌توانند اطلاعات مربوط به بستر، شرایط، موقعیت جغرافیایی، وضعیت و داده‌های دیگر را جمع‌آوری کرده و به خدمت‌دهنده‌ها یا سامانه‌های مرکزی ارسال کنند. این اطلاعات در خدمت‌دهنده‌ها تحلیل می‌شوند و می‌توانند به عنوان منبعی برای ارائه داده‌های مفید، تجزیه و تحلیل ترافیک، پیش‌بینی و اتخاذ تصمیم‌های هوشمند استفاده شوند. از جمله کاربردهای اینترنت اشیاء می‌توان به موارد زیر اشاره کرد:

۱. خانه هوشمند: اتصال لوازم خانگی مانند تلویزیون، سامانه‌های روشنایی، دستگاه‌های گرمایشی و سرمایشی، سامانه‌های امنیتی و سایر دستگاه‌ها به اینترنت به کاربران امکان می‌دهد تا این دستگاه‌ها را از راه دور کنترل و مدیریت کنند.

۲. صنعت هوشمند: در صنعت، اینترنت اشیاء می‌تواند در جمع‌آوری داده‌ها از تجهیزات و سنسورها به منظور نظارت بر فرآیندها، پیشگیری از خرابی‌ها، بهینه‌سازی استفاده از منابع و افزایش بهره‌وری مورد استفاده قرار گیرد.

^{۵۱} Workload

^{۵۲} Scheduler

^{۵۳} Internet of Things (IoT)

شکل ۲-۷: نمای کلی از اینترنت اشیاء

۸-۲ معماری بازخوانی

Callback-Based Software^{Δf}
Callback^{ΔΔ}

این الگو در پاسخ به وقوع رویدادها است. در این حالت، یک تابع بازخوانی به عنوان ورودی به یک رویداد مرتبط ارسال می‌شود. وقتی که رویداد رخ دهد، نرم‌افزار بازخوانی را فراخوانی می‌کند و عملیات مورد نظر را اجرا می‌کند. این روش به برنامه‌نویس امکان می‌دهد تا همزمان با اجرای دیگر بخش‌های برنامه، به رویدادها واکنش نشان دهد. در نرم‌افزارهای مبتنی بر بازخوانی، تعامل بین بخش‌های مختلف برنامه به صورت غیر همزمان انجام می‌شود. با این روش، برنامه‌نویس می‌تواند به طور کنترل شده تغییرات را پیگیری کند و اقدامات مناسبی را در زمان لازم انجام دهد. این الگو به برنامه‌نویسان امکان می‌دهد تا برنامه‌های پیچیده‌تر و قابل اطمینان‌تری را طراحی کنند، زیرا تعاملات غیر همزمان میان بخش‌ها را کنترل می‌کند و بهبود قابلیت اطمینان سیستم را فراهم می‌کند. بنابراین، نرم‌افزارهای مبتنی بر بازخوانی، با استفاده از تابع بازخوانی به عنوان مکانیزم اصلی برای تعامل بین بخش‌های مختلف برنامه، سبب افزایش انعطاف‌پذیری، کنترل و قابلیت اطمینان در طراحی و پیاده‌سازی نرم‌افزارها می‌شوند.

۹-۲ جمع‌بندی

فصل ۳

کارهای مرتبط

۱-۳ مقدمه

در این فصل به معرفی کارهای مشابه و شرح رویکردهای مورد استفاده برای حل مسأله‌ی کنترل و پایش دستگاه‌های اینترنت اشیا بر بستر کوبرنیتز می‌پردازیم.

۲-۳ روش‌های متفاوت حل مسأله کنترل و پایش دستگاه‌های اینترنت اشیا

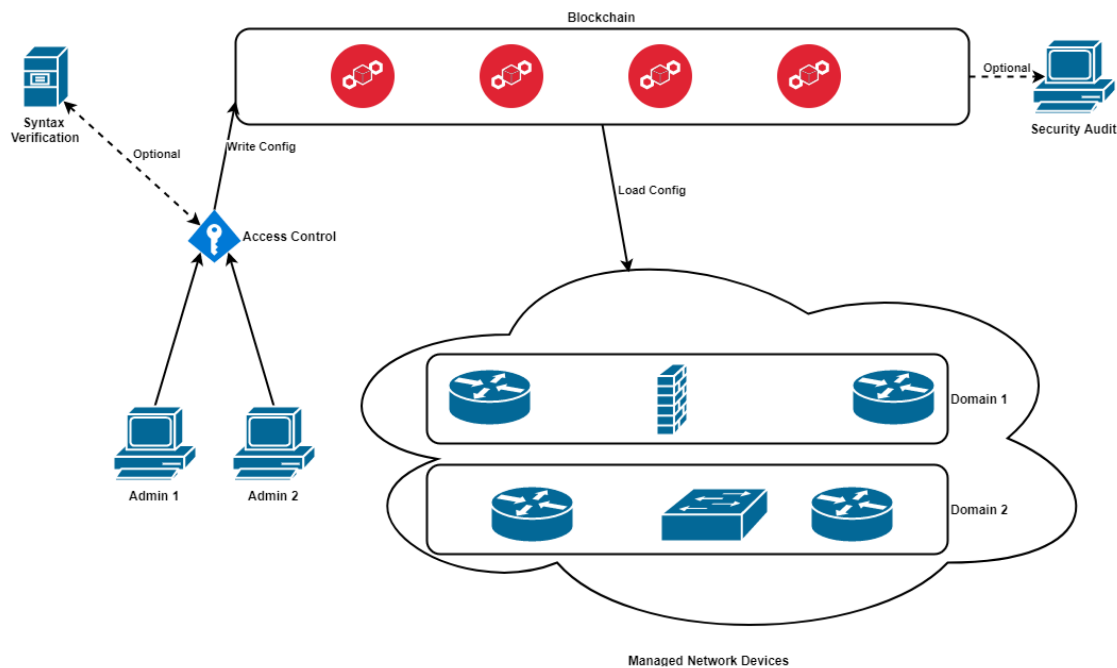
در مسأله مورد بحث، کارهای زیادی انجام نشده است. در ادامه به بررسی برخی از معدود روش‌های حل این مسأله می‌پردازیم

۱-۲-۳ کنترل و پایش دستگاه‌های اینترنت اشیا بر پایه زنجیره‌ی بلوکی

در [۲]، نویسندگان به بررسی مسائل مدیریت و نظارت بر دستگاه‌های اینترنت اشیا می‌پردازند و از تکنولوژی زنجیره‌ی بلوکی^۱ برای حل این مسائل استفاده می‌کنند. در این مقاله، نویسندگان نکاتی را در مورد مزایا و چالش‌های استفاده از زنجیره‌ی بلوکی در مدیریت و نظارت بر دستگاه‌های اینترنت اشیا بررسی می‌کنند. آنها به بررسی معماری زنجیره‌ی بلوکی

^۱Blockchain

و نحوه استفاده آن در این زمینه می‌پردازند. همچنین، روش‌های مختلف برای اعتبارسنجی و امنیت دستگاه‌های اینترنت اشیا با استفاده از زنجیره‌ی بلوکی را مورد بررسی قرار می‌دهند. در ادامه، نویسندگان به بررسی موارد کاربردی مدیریت و نظارت بر دستگاه‌های اینترنت اشیا با استفاده از زنجیره‌ی بلوکی می‌پردازند. آنها به بررسی روش‌هایی برای ثبت و ذخیره داده‌های مربوط به دستگاه‌های اینترنت اشیا در زنجیره‌ی بلوکی می‌پردازند و نحوه استفاده از قراردادهای هوشمند برای اجرای منطق کسب و کار را بررسی می‌کنند.



شکل ۳-۱: پایش مبتنی بر زنجیره‌ی بلوکی [۲]

۲-۲-۳ گسترش کوبرنیتز بر روی دستگاه‌های لبه

در [۴]، نویسندگان به بررسی نیازمندی‌های موجود برای ارتقاء خوشه‌های کوبرنیتز به دستگاه‌های لبه با منابع کم می‌پردازند و نشان می‌دهند که استفاده از کوبلت مجازی می‌تواند یک راه حل مناسب برای این امر باشد. آنها به توضیح عملکرد کوبلت مجازی، که نماینده‌های مجازی در خوشه کوبرنیتز است، می‌پردازند و نحوه تعامل آن با دستگاه‌های لبه را تشریح می‌کنند. در ادامه، نویسندگان به بررسی فناوری‌ها و پروتکل‌های مورد استفاده در کوبلت مجازی می‌پردازند و نحوه اتصال و مدیریت آن را توضیح می‌دهند. آنها به بررسی عملکرد استفاده از کوبلت مجازی در دستگاه‌های لبه، می‌پردازند.

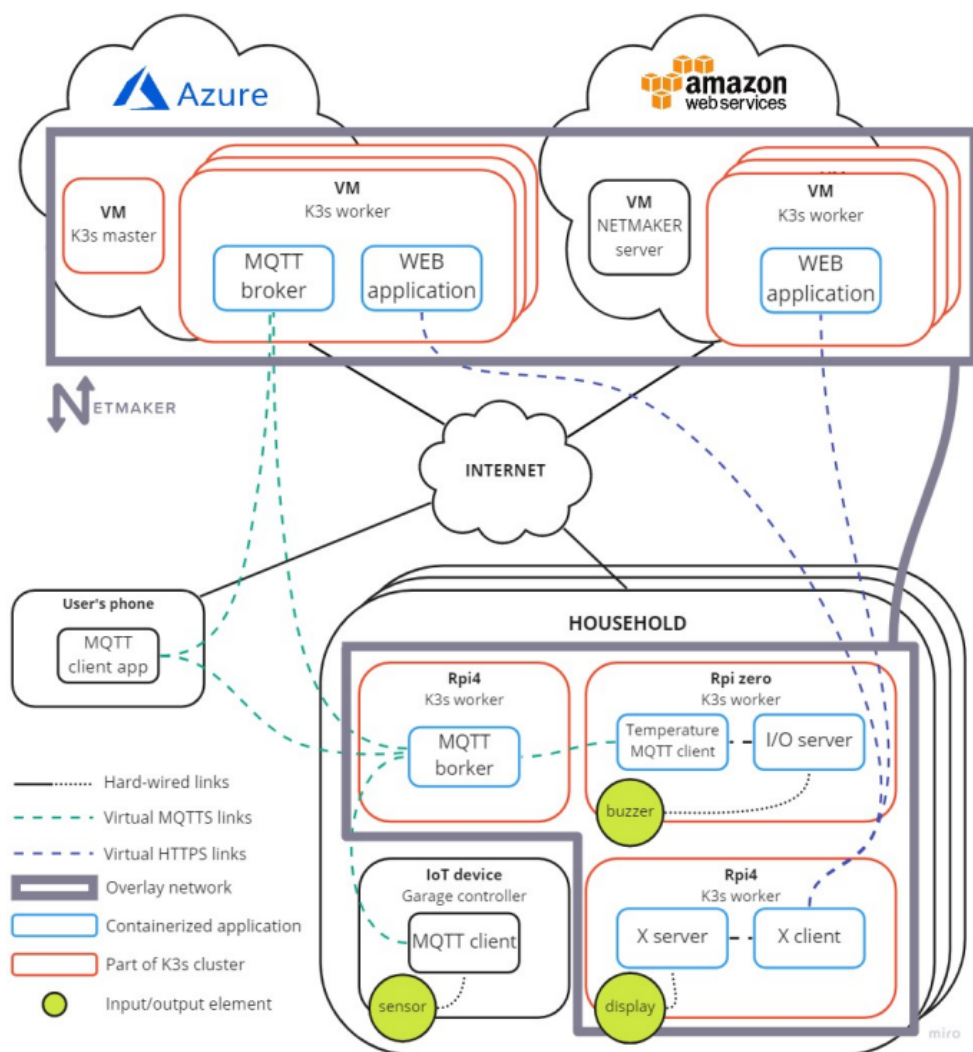
در نهایت، نویسندگان به ارزیابی و ارائه نتایج آزمایش‌هایی که در ارتقاء خوشه‌های کوبرنیتز به دستگاه‌های لبه با استفاده از کوبلت مجازی انجام شده است، می‌پردازند. آنها به بررسی کارایی و کارایی سیستم، زمان پاسخ و نیازمندی‌های منابع مورد نیاز برای استفاده از کوبلت مجازی در دستگاه‌های لبه می‌پردازند.

۳-۲-۳ پایش خانه هوشمند به کمک دستگاه‌های اینترنت اشیا

در [۵]، نویسندگان به بررسی مزایا و چالش‌های استفاده از اینترنت اشیا در خانه‌های هوشمند می‌پردازند. آنها به تشریح معماری سیستم نظارت و کنترل مبتنی بر اینترنت اشیا برای خودکارسازی خانه می‌پردازند و نحوه ارتباط و تعامل بین اجزای مختلف این سیستم را بررسی می‌کنند. در ادامه، نویسندگان به توضیح کارکردها و قابلیت‌های سیستم نظارت و کنترل مبتنی بر اینترنت اشیا برای خودکارسازی خانه می‌پردازند. آنها به بررسی امکانات نظارت بر دستگاه‌های خانگی، کنترل سیستم‌های روشنایی و الکترونیکی، کنترل دما و رطوبت، و همچنین مدیریت از راه دور این سیستم می‌پردازند. در نهایت، نویسندگان به ارائه نتایج واقعی سیستم نظارت و کنترل مبتنی بر اینترنت اشیا برای خودکارسازی خانه پرداخته و مورد استفاده قرار گرفتن این سیستم را بررسی می‌کنند. آنها به تأثیر این سیستم بر کاهش مصرف انرژی، راحتی کاربران و بهبود کیفیت زندگی در خانه اشاره می‌کنند.

۴-۲-۳ پایش دستگاه‌های اینترنت اشیا بر پایه کوبرنیتز

در این مقاله [۳]، چالش‌های فناوری‌های در توسعه معماری اینترنت اشیا بررسی می‌شود و راه‌حلی خاص بر بستر کوبرنیتز برای مدیریت میلیون‌ها دستگاه در محیط‌های مختلف پیشنهاد می‌شود. این پایان‌نامه بر تأسیس یک زیرساخت اینترنت اشیا کامل با استفاده از ابزار کوبرنیتز تمرکز دارد که به عنوان یکی از بهترین راه‌حل‌ها برای کنترل منابع مختلف شناخته شده است و در محیط‌های ابری به‌طور گسترده برای مدیریت بارکاری نرم‌افزاری استفاده می‌شود. هدف اصلی این کار، ایجاد یک معماری اینترنت اشیا انعطاف‌پذیر، مقیاس‌پذیر و گسترش‌پذیر با تأکید بر امنیت در تمام لایه‌های آن است. مقاله نیز به بررسی فناوری‌های موردنیاز برای استفاده از کوبرنیتز در طراحی معماری اینترنت اشیا می‌پردازد و تکنولوژی‌های مرتبط دیگری را نیز معرفی می‌کند که به هدف‌های مشخص شده در این تحقیق منطبق هستند.



شکل ۲-۳: پایش مبتنی بر کوبرنیتز [۳]

فصل ۴

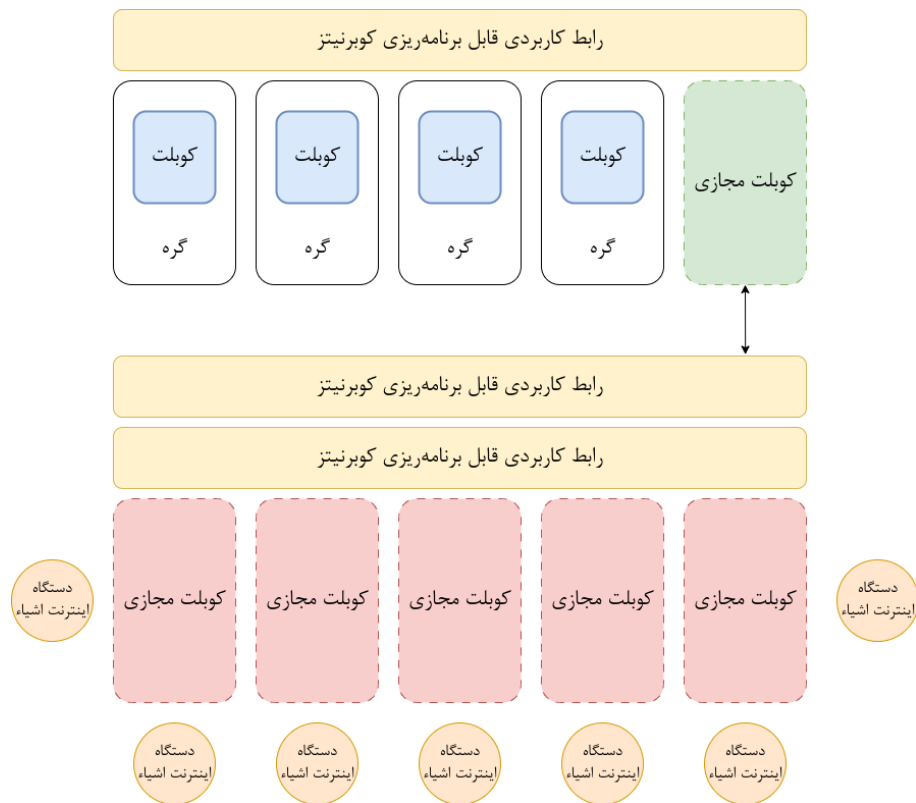
روش پیشنهادی

۴-۱ مقدمه

در این بخش، ابتدا معماری روش پیشنهاد را شرح داده و سپس اجزای مختلف آن را بررسی می‌کنیم. سپس به نحوه کارکرد این روش می‌پردازیم.

۴-۲ معماری سامانه

اجزای اصلی سامانه متشکل از پیاده‌سازی تامین‌کننده کوبلت مجازی، رابط بین تامین‌کننده و کنترل‌کننده‌ها، کنترل‌کننده‌ها و دستگاه‌ها می‌باشد که در ادامه به تعریف هرکدام پرداخته



شکل ۴-۱: نمای کلی معماری

۱-۲-۴ پیاده‌سازی تامین‌کننده کوبلت مجازی

کوبلت مجازی با در اختیار گذاشتن رابط‌هایی برای برنامه‌نویس، این امکان را می‌دهد که بتوان گره‌های کوبرنیتز با پشته‌های سفارشی‌سازی شده داشته باشیم. برای مثال رابط زیر چرخه وجودی یک پاد را نشان می‌دهد. حال با پیاده‌سازی این رابط، ما این امکان را داریم که از ساخته‌شدن، بروزرسانی شدن، حذف شدن و حتی تغییر وضعیت‌های پادهای مورد نظر خود با خبر شویم.

رابط کنترل‌کننده چرخه وجودی پاد : Listing 4-1

```

type PodLifecycleHandler interface {
    CreatePod(ctx context.Context, pod *corev1.Pod) error

    UpdatePod(ctx context.Context, pod *corev1.Pod) error

    DeletePod(ctx context.Context, pod *corev1.Pod) error

    GetPod(ctx context.Context, namespace, name string)
        (*corev1.Pod, error)

    GetPodStatus(ctx context.Context, namespace, name string)
        (*corev1.PodStatus, error)

    GetPods(context.Context) ([]*corev1.Pod, error)
}

```

پیاده‌سازی این رابط امکان این را می‌دهد که بتوانیم یک پاد بر روی کوبرنیتز اعمال کرده و سپس بوسیله کوبرنیتز فراخوانی شده تا پاد مورد نظر را بسازیم. در این پروژه یک پاد نقش یک دستگاه اینترنت اشیاء را دارد. همچنین برای ارسال وضعیت گره مجازی ساخته شده، نیاز به پیاده‌سازی رابط دیگری داریم.

رابط کنترل‌کننده وضعیت گره : Listing 4-2

```

type NodeProvider interface {
    Ping(context.Context) error

    NotifyNodeStatus(ctx context.Context, cb func(*corev1.Node))
}

```

پیاده‌سازی این رابط باعث می‌شود هنگامی که کد مربوطه در حال اجرا می‌باشد، گره مورد در نظر در خوشه کوبرنیتز بصورت آماده ظاهر شود و اگر این کد متوقف شود، بصورت غیر آماده ظاهر شود. بعد از ثبت این دو رابط و انجام چند مرحله دیگر، تامین‌کننده ما آماده استفاده می‌شود و بصورت یک گره در کوبرنیتز ظاهر خواهد شد.

```
→ examples git:(docs) x kubectl get nodes
NAME        STATUS    ROLES    AGE   VERSION
minikube    Ready    control-plane  59d   v1.26.3
vkube       Ready    agent      5d17h test-v0.0.1
→ examples git:(docs) x
```

شکل ۲-۴: گره مجازی

۲-۲-۴ رابط برقراری ارتباط با کنترل کننده‌ها

بعد از اتصال به کوبرنیتز و دریافت درخواست‌ها و بروزرسانی‌ها از سوی این سکو، باید با دستگاه‌های اینترنت اشیاء ارتباط برقرار کرده و وضعیتشان را در اختیار کوبرنیتز قرار دهیم. منطق ارتباط با کنترل کننده‌های دستگاه‌های اینترنت اشیاء به این صورت است که بصورت مداوم درخواست‌های خاصی به سمت کنترل کننده‌ها می‌فرستد تا از وضعیت خود کنترل کننده‌ها و همچنین دستگاه‌های اینترنت اشیاء تحت کنترلشان با خبر شود و در صورت نیاز کوبرنیتز را بروزرسانی کند. این درخواست‌ها چیزی نیست جز درخواست‌های قرارداد انتقال فرا متن. همچنین برای اینکه بتوان تعداد زیادی دستگاه اینترنت اشیاء را با یک کنترل کننده، کنترل کرد؛ از روش تکرار مقطعی استفاده شده است. این روش به ما کمک می‌کند تا وضعیت دستگاه‌ها را بصورت مقطعی (نه یکجا) دریافت کرده که بتوان در صورت امکان از همزمانی، برای تسریع کار، استفاده کرد.

این رابط برای اینکه داده‌های مربوط به وضعیت دستگاه‌ها و کنترل کننده‌ها را در اختیار تامین کننده قرار دهد، از معماری بازخوانی^۱ استفاده می‌کند.

۳-۲-۴ شبیه‌ساز

در این پروژه یک شبیه‌ساز هم پیاده‌سازی شده که نقش کنترل کننده دستگاه‌های اینترنت اشیاء و همچنین خود این دستگاه‌ها را ایفا می‌کند. این شبیه‌ساز یک خدمت‌دهنده قرارداد انتقال فرامتن می‌باشد که امکانات زیر را هم برای کنترل کننده و هم برای دستگاه‌های اینترنت اشیاء فراهم می‌کند:

۱. ساخت

۲. ساخت جمعی (برای ارزیابی ساده)

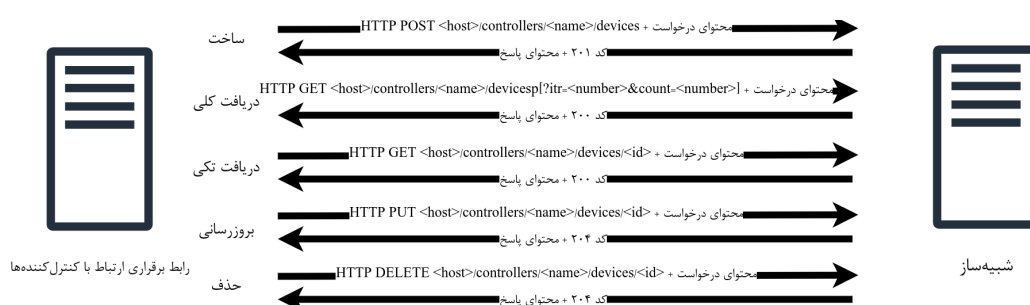
^۱ Callback

۳. بروزرسانی

۴. حذف

۵. دریافت تکی، همه و مقطعی

دستگاه‌های اینترنت اشیا شبیه‌سازی شده قفل‌های هوشمند یک ساختمان می‌باشند که امکان باز کردن و بستن قفل را دارند.



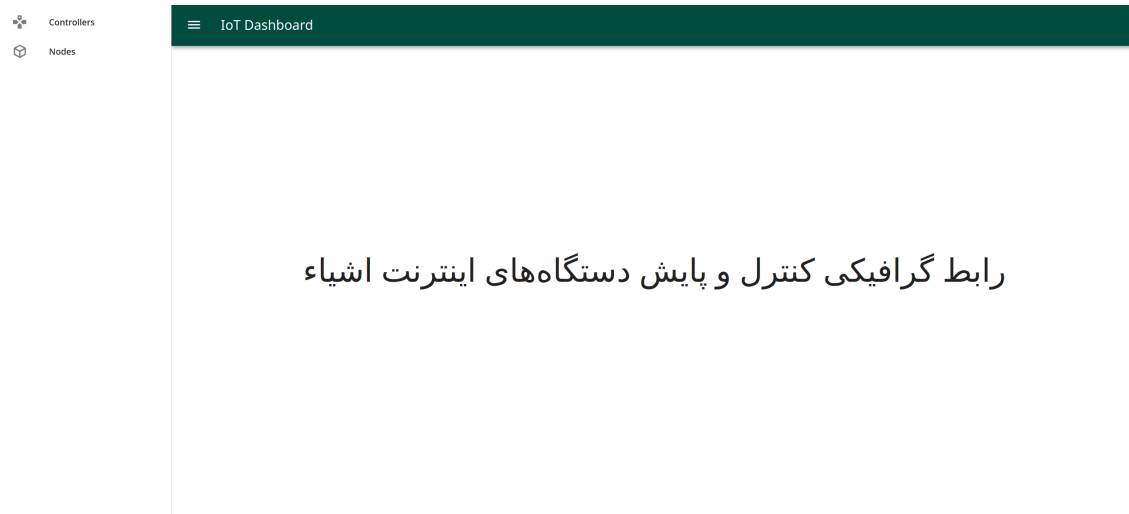
شکل ۴-۳: رابط کاربردی قابل برنامه‌نویسی شبیه‌ساز

۴-۲-۴ رابط گرافیکی

با توجه به اینکه هدف این پروژه امکان‌سنجی و پیاده‌سازی روشی برای پایش دستگاه‌های اینترنت اشیا بوسیله بستر کوبرنیتز است، اما رابط گرافیکی نیز طراحی شد برای نمایش دادن هرچه بهتر اجزای پروژه. این رابط از دو بخش تشکیل شده است.

۱. بخشی که با کنترل‌کننده دستگاه‌های اینترنت اشیا ارتباط دارد و کمک به تسهیل ساخت و نمایش دستگاه‌های اینترنت اشیا می‌کند.

۲. بخش دیگر که با تأمین‌کننده در ارتباط است و بصورت مداوم وضعیت گره‌ها و پادهای مجازی را بروزرسانی می‌کند.

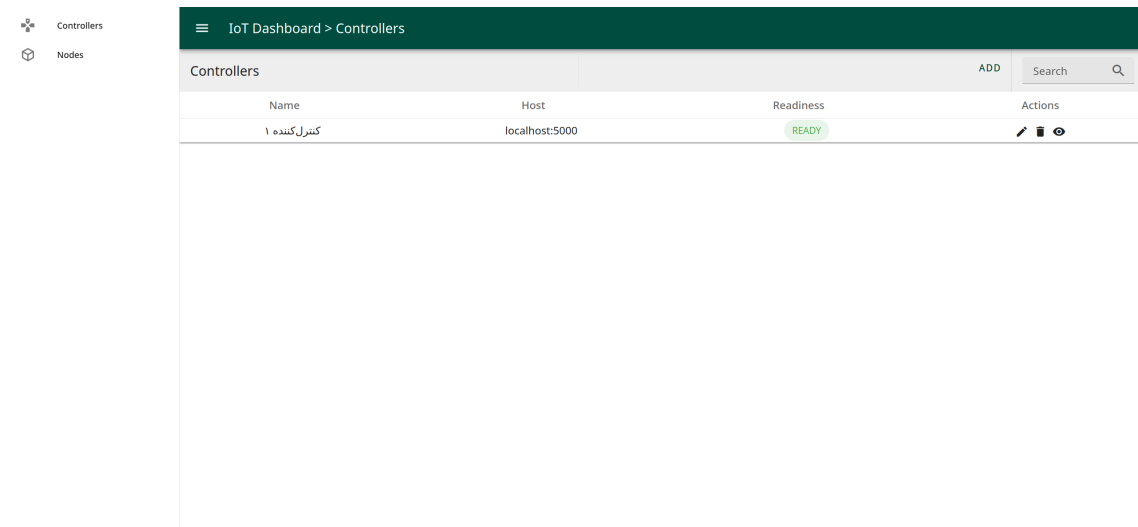





شکل ۴-۴: صفحه اصلی رابط گرافیکی

The screenshot shows the 'Nodes' page within the IoT Dashboard. The sidebar on the left is the same as in the previous image. The main area has a dark green header bar with a hamburger menu icon and the text 'IoT Dashboard > Nodes'. Below the header, there is a table with the following columns: Name, Cpu, Memory, Allocatable Pods, Max Pods, Readiness, and Actions. The table contains two rows of data: 'minikube' and 'vkube'. The 'minikube' row shows 16 Cpu, 15594756Ki Memory, 110 Allocatable Pods, 110 Max Pods, and a 'READY' status. The 'vkube' row shows 16 Cpu, 4Gi Memory, 10k Allocatable Pods, 10k Max Pods, and a 'READY' status. The 'Actions' column contains icons for each node.

Name	Cpu	Memory	Allocatable Pods	Max Pods	Readiness	Actions
minikube	16	15594756Ki	110	110	READY	
vkube	16	4Gi	10k	10k	READY	

شکل ۴-۵: صفحه گره‌های کوبرنیتز



IoT Dashboard > Controllers			
Controllers			ADD
Name	Host	Readiness	Actions
کنترل کننده ۱	localhost:5000	READY	  

شکل ۴-۶: صفحه کنترل کننده های کوبرنیتز کوبرنیتز









۴-۳ نحوه کارکرد

ابتدا باید یک مستند یاد که در ذیل آمده مهیا کرده و در کوبرنیتز اعمال کنیم. بعد از اعمال این مستند، کوبرنیتز تامین کننده را از ایجاد این مستند با خبر می کند. حال تامین کننده با بازفراخوانی رابط کنترل کننده ها منجر به شروع دریافت وضعیت این دستگاه اینترنت اشیا بصورت مداوم از کنترل کننده تعریف شده در مستند می شود. بنابراین رابط کنترل کننده ها بصورت مداوم با رابط کاربردی قابل برنامه نویسی شبیه ساز ارتباط گرفته و وضعیت دستگاه ها را بروزرسانی می کند.

مستند ساخت پاد در کوبرنیتز : Listing 4-3

```
apiVersion: v1
kind: Pod
metadata:
  name: lock-main
  annotations:
    controllerName: "controller1"
    controllerAddress: "localhost:5000"
spec:
  containers:
    # this is so that kubernetes validation will pass
    - image: doesntmatter/smart_lock
      name: lock1
  dnsPolicy: ClusterFirst
  nodeSelector:
    kubernetes.io/role: agent
    kubernetes.io/os: linux
    type: virtual-kubelet
  tolerations:
    # this will target Virtual Kubelets nodes only
    - key: itzloop.dev/virtual-kubelet
      operator: Exists
```

قبل از اعمال مستند بالا، از طریق رابط گرافیکی در شبیه‌ساز چهار دستگاه ساخته که در شکل زیر مشاهده می‌کنید.

IoT Dashboard > Controllers > Controller1		
Devices	ADD	Search
Name	Readiness	Actions
lock-main	READY	 
lock-storage	READY	 
lock-room1	READY	 
lock-room2	READY	 

Items per page: 10 1-4 of 4

شکل ۴-۷: دستگاه‌های اینترنت اشیاء ساخته شده در شبیه‌ساز

در ادامه مستند پاد را اعمال می‌کنیم. همانطور که در مستند آماده است فقط یکی از این دستگاه‌ها یعنی قفل مرکزی^۲ را از طریق کوپرنیتز رصد می‌کنیم. بعد از اعمال این مستند پادهای کوپرنیتز را مشاهده کرده که ابته در وضعیت عدم آمادگی^۳ می‌باشند.

lock-main^۲
Not-Ready^۳

IoT Dashboard > Nodes > Vcube

Controllers

Nodes

Pods

Search

Name	Namespace	Readiness
lock-main	default	NOT-READY

Items per page:

10

1-1 of 1

|<

<

>

>|

شکل ۴-۸: پاد ساخته شده از روی مستند در وضعیت عدم آمادگی

پس از گذر مدتی (مدت زمانی که طول می کشد رابط ارتباط با کنترل کننده ها وضعیت قفل مرکزی را از شبیه ساز دریافت کند) خواهیم دید که پاد مورد نظر در کوبرنیتز به وضعیت آماده^۴ در می آید.

Controllers

Nodes

IoT Dashboard > Nodes > Vcube

Pods

Search

Name	Namespace	Readiness
lock-main	default	READY

Items per page:

10

1-1 of 1

|<

<

>

>|

شکل ۴-۹: پاد ساخته شده از روی مستند در وضعیت آماده

Ready^۴

حال اگر با استفاده از رابط گرافیکی، وضعیت قفل مرکزی در شبیه‌ساز را به وضعیت عدم آمادگی تغییر دهیم خواهیم دید که بعد از مدتی وضعیت پاد نیز تغییر می‌کند.

IoT Dashboard > Controllers > Controller1		
Devices		ADD Search
Name	Readiness ↑	Actions
lock-main	NOT-READY	[Edit] [Delete]
lock-storage	READY	[Edit] [Delete]
lock-room1	READY	[Edit] [Delete]
lock-room2	READY	[Edit] [Delete]

Items per page: 10 1-4 of 4 |< < > >|

شکل ۴-۱۰: تغییر وضعیت قفل مرکزی در رابط گرافیکی

IoT Dashboard > Nodes > Vkube		
Pods		Search
Name	Namespace	Readiness
lock-main	default	NOT-READY

Items per page: 10 1-1 of 1 |< < > >|

شکل ۴-۱۱: تغییر وضعیت پاد بدلیل تغییر وضعیت قفل مرکزی

فصل ۵

ارزیابی روش پیشنهادی

۵-۱ مقدمه

۵-۱-۱ ارزیابی و مقایسه معماری‌های ارجح

پس از بدست آوردن نتایج، به مقایسه معماری‌های مورد استفاده پرداخته شده است. با توجه به جدول ۱-۲ همان طوری که انتظار می رفت، شبکه‌های عصبی Transformers همراه با شبکه LXMERT به عنوان کدگذار بهترین عملکرد را داشت. اگر اندکی نتایج را واریسی کنیم متوجه بالا بودن امتیازها می شویم که نسبت به مدل‌های مشابه در توضیح تصاویر مقادیر بسیار بالایی دارند. مطابق با آنچه در بخش؟؟ بحث شد، یکی از دلایل را می توان ساده بودن مجموعه داده در نظر گرفت. از آنجایی که مجموعه داده از قوانین از پیش تعیین شده و بصورت خودکار تولید شده اند، مدل ها و به خصوص کدگشا ممکن است این الگوها را آموزش دیده باشند. با این حال در همه حالات از دقتی که نویسندگان مجموعه داده گزارش کرده بودند عملکرد بهتری داشته ایم.

Embedding-based		Word-based			Method	
BERT Score	Average Score	ROUGE-L	METEOR	BLEU	Decoder	Encoder
-	-	-	23.3	23.9	LSTM	LSTM Q+I
79.19	86.50	56.38	56.65	32.19	1-LSTM	LXMERT
82.07	89.63	61.50	62.99	39.37	1-GRU	
91.84	95.94	85.49	86.43	79.03	1-LSTM+Bahdanau attention	
91.90	96.11	86.25	86.96	79.54	1-LSTM+Luong(general) attention	
89.32	95.42	82.62	83.26	71.40	1-GRU+Bahdanau attention	
90.37	95.73	83.84	84.71	73.92	1-GRU+Luong(dot) attention	
95.01	90.20	90.60	91.18	86.73	3-Transformer Decoder	VisualBERT
69.20	84.83	38.35	38.00	18.62	1-LSTM	
73.59	87.74	43.72	44.24	22.51	1-GRU	
93.50	97.11	87.28	88.07	84.27	1-LSTM+Bahdanau attention	
93.11	97.17	86.90	87.71	82.90	1-LSTM+Luong(concat) attention	
89.26	96.10	82.40	82.87	72.20	1-GRU+Bahdanau attention	
91.81	96.93	85.23	86.17	79.65	1-GRU+Luong(dot) attention	
94.44	91.94	89.09	89.76	85.95	3-Transformer Decoder	

جدول ۵-۱: نتایج معماری‌های متفاوت بر مجموعه داده FSVQA. اعداد موجود در نام کدگشا نشانگر تعداد لایه‌های آن است.

فصل ۶

نتیجه‌گیری و کارهای آینده

۱-۶ نتیجه‌گیری

۲-۶ دستاوردها

۳-۶ کارهای آینده

کتابنامه

- [1] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, “Borg, omega, and kubernetes,” *ACM Queue*, vol.14, pp.70–93, 2016.
- [2] K. Košťál, P. Helebrandt, M. Belluš, M. Ries, and I. Kotuliak, “Management and monitoring of iot devices using blockchain,” *Sensors*, vol.19, no.4, 2019.
- [3] D. Mlynka, “Iot device management using kubernetes [online],” master’s thesis, Masaryk University, Faculty of Informatics, Brno, 2022 [cit. 2023-07-15]. SUPERVISOR : Zdeněk Matěj.
- [4] T. Goethals, F. Turck, and B. Volckaert, “Extending kubernetes clusters to low-resource edge devices using virtual kubelets,” *IEEE Transactions on Cloud Computing*, vol.PP, pp.1–1, 10 2020.
- [5] D. Pavithra and R. Balakrishnan, “Iot based monitoring and control system for home automation,” in *2015 Global Conference on Communication Technologies (GCCT)*, pp.169–173, 2015.

واژه‌نامه فارسی به انگلیسی

واژه‌نامه انگلیسی به فارسی

گره	Node
خدمت گیرنده	Client
خدمت دهنده	Server
قرارداد	Protocol
رابط	Interface
رابط کاربردی قابل برنامه‌نویسی	Application Programming Interface
قرارداد انتقال فرا متن	Hypertext Transfer Protocol
پشتوانه	Backend
بازخوانی	Callback

Abstract

TODO

Keywords : Internet of Things, Kubernetes, Virtual Kubelet, Centralized Monitoring, Scalable Monitoring



Iran University of Science and Technology
Computer Engineering Department

A K8-Based Mechanism for Remote Monitoring and Control of IoT Devices

Bachelor of Science Thesis in Computer Engineering – Software Engineering

By:

Sina Shabani Kumeleh

Supervisor:

Dr. Mohsen Sharifi

Summer 2023