Kerangka Kerja Arsitektur IoT: Kerangka Kerja Koneksi dan Integrasi untuk Sistem IoT

Onoriode Uviase Gerald Kotonya
Fakultas Ilmu Komputer dan Komunikasi Universitas Lancaster
Lancaster, Inggris
o.uviase@lancaster.ac.uk g.kotonya@lancaster.ac.uk

Berkembangnya Internet of Things (IoT) telah meningkatkan minat terhadap desain arsitektur dan kerangka kerja adaptif untuk mendorong hubungan antara perangkat IoT yang heterogen dan sistem IoT. Arsitektur perangkat lunak yang paling banyak disukai dalam IoT adalah Service Oriented Architecture (SOA), yang bertujuan untuk menyediakan sistem yang digabungkan secara longgar untuk meningkatkan penggunaan dan penggunaan kembali layanan IoT di lapisan middle-ware, untuk meminimalkan masalah integrasi sistem. Namun, terlepas dari fleksibilitas yang ditawarkan oleh SOA, tantangan untuk mengintegrasikan, menskalakan, dan memastikan ketahanan dalam sistem IoT tetap ada. Salah satu penyebab utama integrasi yang buruk dalam sistem IoT adalah kurangnya kerangka kerja yang cerdas dan sadar akan koneksi untuk mendukung interaksi dalam sistem IoT. Makalah ini mengulas kerangka kerja arsitektur yang ada untuk mengintegrasikan perangkat IoT dan mengidentifikasi area-area utama yang memerlukan perbaikan penelitian lebih lanjut. Makalah ini diakhiri dengan mengusulkan solusi yang memungkinkan berdasarkan microservice. Kerangka kerja integrasi IoT yang diusulkan mendapat manfaat dari lapisan API cerdas yang menggunakan *perakit layanan* eksternal*, auditor layanan, monitor layanan, dan* komponen *router layanan* untuk mengoordinasikan penerbitan layanan, langganan, pemisahan, dan *kombinasi layanan* dalam arsitektur.

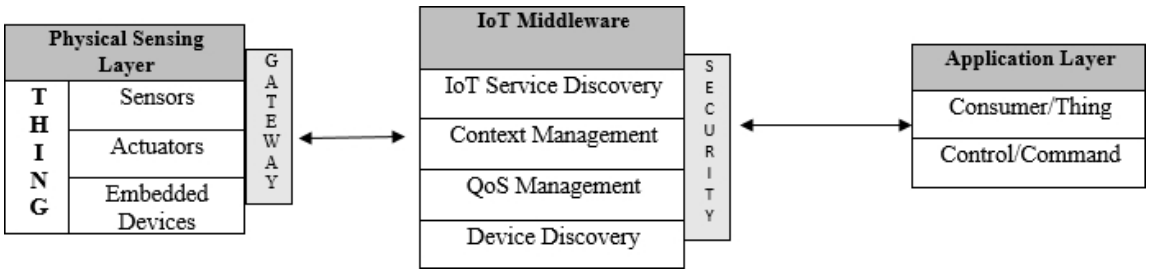
1 Pendahuluan

Istilah Internet of Things (IoT) mengacu pada jaringan heterogen objek fisik dan virtual yang tertanam dengan elektronik, perangkat lunak, sensor, dan konektivitas untuk memungkinkan objek mencapai nilai dan layanan yang lebih besar dengan bertukar data dengan objek lain yang terhubung melalui internet [16]. "Benda" dalam hal IoT, dapat berupa seseorang dengan implan monitor jantung, hewan ternak dengan transponder biochip, robot operasi lapangan yang membantu misi pencarian dan penyelamatan, atau benda alami atau buatan manusia lainnya yang dapat diberi alamat IP dan dilengkapi dengan kemampuan untuk mentransfer data dan beroperasi dalam infrastruktur Internet yang ada [21].

Contoh lingkungan yang mendukung IoT adalah sistem transportasi terintegrasi yang dapat dirutekan dan direorganisasi secara dinamis sebagai respons terhadap perubahan kebutuhan dan kondisi lalu lintas [26]. Dalam bidang kesehatan, IoT telah digunakan untuk menindaklanjuti pemulihan pasien dan untuk menilai hal tersebut terhadap sejumlah parameter yang unik bagi pasien dengan menggunakan perangkat yang mendukung IoT [8]. Data yang dikumpulkan juga dapat digunakan untuk membandingkan respons pasien terhadap pengobatan dalam konteks lingkungan yang berbeda dalam skala global. Perangkat IoT pintar juga dapat digunakan untuk memantau dan mengontrol penggunaan energi. Di bidang pertanian dan produksi pangan, IoT dapat digunakan untuk mengelola produksi dengan memantau dan melacak variabel-variabel yang memengaruhi produksi pangan seperti cuaca, indikator ekonomi-politik, bencana alam, konsumsi, penyakit tanaman dan hewan, dll. [14]. Dalam kehidupan berbantuan, keberadaan perangkat dan layanan IoT di mana-mana dapat membantu memenuhi kebutuhan hidup mandiri bagi semakin banyak orang yang hidup dengan disabilitas fisik, kondisi jangka panjang, masalah sosial, dan masalah yang berkaitan dengan usia [23].

Gambar 1 menunjukkan arsitektur sistem IoT dasar. Lapisan penginderaan fisik berisi perangkat tertanam yang memanfaatkan sensor untuk mengumpulkan data dunia nyata. Lapisan gateway menyediakan mekanisme

D. Pianini dan G. Salvaneschi (Eds.): Lokakarya pertama tentang Arsitektur, Bahasa, dan Paradigma untuk IoT
EPTCS 264, 2018, hal. 1-17, doi: 10.4204/EPTCS.264.1



Gambar 1: Arsitektur Dasar IoT

dan protokol untuk perangkat agar dapat mengekspos data yang diindera ke Internet (misalnya Wi-Fi, Ethernet, GSM, dll.). Lapisan middle-ware memfasilitasi dan mengelola komunikasi antara aktivitas penginderaan dunia nyata dan lapisan aplikasi. Lapisan aplikasi memetakan ke aplikasi yang dapat digunakan oleh konsumen untuk mengirim perintah ke objek dunia nyata melalui Internet melalui aplikasi seluler, aplikasi web, dll.

IoT adalah tren yang berkembang pesat dalam industri besar. Pada tahun 2015, Samsung mengantisipasi bahwa 90% dari produknya akan mendukung IoT pada tahun 2017 [3] dan seluruh produknya pada tahun 2020. Gartner juga membayangkan percepatan peningkatan perangkat IoT dengan memperkirakan sekitar 21 miliar perangkat yang mendukung IoT pada tahun 2020 [2]. Ukuran dan heterogenitas pertumbuhan yang diprediksi menambah kompleksitas dan urgensi ekstra pada masalah koneksi dan integrasi yang ada [7]. Selain itu, sistem IoT kemungkinan besar akan didistribusikan di berbagai domain aplikasi dan lokasi geografis yang menciptakan ketergantungan tersembunyi di seluruh domain, platform, dan layanan. Hal ini memiliki implikasi serius tentang bagaimana sistem IoT dikembangkan dan berevolusi. Dengan demikian, kebutuhan akan kerangka kerja yang cerdas dan sadar akan koneksi telah menjadi suatu keharusan.

Service Oriented Architecture (SOA) menawarkan kerangka kerja yang kuat untuk mendukung konektivitas, interoperabilitas, dan integrasi dalam sistem IoT, yang menjadi tulang punggung kerangka kerja IoT saat ini. Meskipun tujuan utama SOA adalah untuk meningkatkan interoperabilitas aplikasi IoT, penggunaan monolitiknya dalam kerangka kerja IoT saat ini semakin memperkuat masalah skalabilitas, terutama dengan banyaknya "*hal*" yang diprediksi. Sistem IoT cenderung berkembang dan seiring berjalannya waktu, kerangka kerja SOA yang mumpuni menjadi tidak mampu lagi menangani perluasan sistem. Microservice bertujuan untuk memecah sistem IoT yang berbeda berdasarkan paradigma System of Systems (SOS) untuk memenuhi evolusi dan ekstensibilitas sistem secara memadai. Makalah ini mengulas kerangka kerja integrasi IoT saat ini dan mengusulkan solusi yang sebagian telah dieksplorasi, tetapi menjanjikan untuk masalah integrasi layanan yang buruk, skalabilitas, perluasan, dan toleransi kesalahan dalam IoT.

Makalah ini disusun sebagai berikut: Bagian 2 akan berisi motivasi dan tantangan dalam arsitektur IoT. Pada Bagian 3, kami akan menyajikan beberapa persyaratan untuk desain kerangka kerja IoT dan mengulas beberapa kerangka kerja IoT yang terkemuka. Bagian 4 akan membahas pendekatan kami dalam merancang kerangka kerja IoT untuk mendukung konektivitas perangkat, integrasi, interoperabilitas, toleransi terhadap kesalahan, dan skalabilitas. Di Bagian 5, kami menyimpulkan makalah kami.

2 Motivasi dan Tantangan

Interoperabilitas mengacu pada kemampuan perangkat dan layanan IoT yang berbeda untuk bertukar informasi dan menggunakan informasi yang telah dipertukarkan. Inisiatif IoT saat ini sebagian besar berfokus pada aplikasi dan perangkat yang memenuhi kebutuhan yang berbeda-beda, tetapi memberikan sedikit ruang lingkup untuk interoperabilitas dan koneksi. Mereka tidak memiliki seperangkat istilah yang sama untuk menggambarkan layanan, batasan layanan, dan strategi layanan. Hal ini mengurangi visibilitas sistem secara keseluruhan, menciptakan fungsionalitas yang berlebihan, layanan yang berlebihan, dan menghadirkan masalah untuk perubahan

manajemen dan membatasi ruang lingkup untuk koordinasi dan penggunaan ulang sumber daya. Solusi IoT perlu mendukung koneksi dan interoperabilitas sebagai *blok bangunan* untuk menyediakan mekanisme, proses, dan keamanan yang memungkinkan perangkat dan layanan yang berbeda untuk digabungkan. Manfaatnya adalah perangkat dan layanan IoT dapat berinteraksi satu sama lain dan menciptakan efisiensi yang lebih besar, meningkatkan kemudahan penggunaan, memberikan kemampuan yang lebih baik, pilihan yang lebih banyak, mengarah pada skala ekonomi dan biaya unit yang [13]. lebih rendah Interoperabilitas dan koneksi yang buruk diperparah oleh sejumlah faktor termasuk:

- *Kurangnya deskripsi layanan yang terstandarisasi*. Saat ini, masih ada kekurangan standar untuk konvensi penamaan layanan perangkat IoT, informasi data dan deskripsi. Hal ini menjadi tantangan besar dalam masalah integrasi. Tren terbaru dalam kesadaran konteks dan manajemen data telah menerapkan teknologi web semantik dari Ontology Web Language (OWL) tetapi tetap saja, tidak ada metode standar untuk menggambarkan layanan IoT baik pada tingkat semantik tinggi maupun rendah [10]. Dengan demikian, kita telah melihat konvensi penamaan ini berbeda sebagai akibat dari latar belakang budaya, sosial, atau kebiasaan pengkodean di mana istilah yang berbeda digunakan untuk mewakili entitas yang serupa atau sama. Solusi untuk mengintegrasikan IoT yang heterogen akan sangat diuntungkan dengan teknik pemodelan terintegrasi untuk memberikan akuisisi pengetahuan dan representasi yang tepat dari domain IoT.
- *Kesadaran konteks* yang *buruk untuk layanan.* Kurangnya semantik yang memadai untuk deskripsi teks layanan sesuai masih terlihat dalam desain sistem IoT saat ini. Sistem IoT saat ini mengalami kekurangan kesadaran konteks yang memadai untuk layanan karena semantik yang dimodelkan secara tidak tepat yang [24]. menghasilkan berbagai ontologi yang tidak terdistribusi secara merata dan semantik yang tidak koheren untuk layanan Untuk pemrosesan data yang sadar konteks yang tepat dalam domain IoT, pendekatan baru harus digunakan untuk memodelkan dan merancang mesin aturan untuk layanan.
- *Klasifikasi layanan perangkat yang buruk*. Beberapa sistem IoT menggunakan katalog layanan perangkat berdasarkan kategorisasi perangkat. Jenis praktik ini mengaitkan layanan ke perangkat berdasarkan pengenal unik dari perangkat dan layanan [25]. Layanan-layanan ini dapat ditemukan jika dilakukan penyelidikan terhadap layanan atau pengenal perangkat. Dalam sistem IoT, praktik ini telah lama membantu penemuan perangkat dan penemuan layanan, tetapi dari sudut pandang M2M, penemuan layanan masih belum meyakinkan sehingga, jika perangkat gagal dan tidak ada dalam sistem, sistem akan mencatatnya sebagai layanan yang tidak tersedia meskipun layanan yang sama dengan pengidentifikasi unik yang berbeda disediakan oleh perangkat lain. Harus ada pendekatan baru untuk klasifikasi layanan untuk mengakomodasi penemuan layanan yang dinamis.
- *Visualisasi dan analisis informasi* yang *buruk.* Inisiatif yang ada saat ini hanya memberikan sedikit ruang lingkup untuk pengumpulan dan visualisasi data yang disesuaikan. Visualisasi dan analisis data yang [9].disesuaikan dari aktivitas dan lingkungan objek dapat memberikan wawasan yang tak ternilai tentang kesejahteraan dan kecukupan sistem berkelanjutan

3 Kerangka Kerja IoT

Agar kerangka kerja IoT dapat diandalkan dan dapat diandalkan, beberapa langkah minimal harus dipenuhi untuk mencapai integrasi dan interoperabilitas dalam IoT. Kerangka kerja ini menjangkau seluruh komunitas penelitian IoT mulai dari penelitian akademis hingga penelitian organisasi yang berfokus pada integrasi berbagai hal dalam IoT. Karena paradigma IoT sendiri masih dalam tahap pengembangan, kami mengusulkan seperangkat ukuran minimal yang harus dipenuhi oleh kerangka kerja IoT untuk integrasi. Hal-hal tersebut adalah:

- *Pemisahan kontrak*: Sistem IoT berisi perangkat heterogen dengan protokol komunikasi yang berbeda. Kerangka kerja integrasi harus cukup kompeten untuk menangani pemisahan kontrak secara efisien. Pemisahan kontrak adalah kemampuan konsumen layanan dan produsen layanan untuk

berkembang secara independen tanpa mengakhiri kontrak di antara mereka [19]. Sebagai contoh, sebuah layanan mungkin dalam format JSON dan konsumen layanan membutuhkan input dalam XML. Kerangka kerja harus menyediakan dukungan untuk mengubah pesan ke format yang memenuhi kontrak di antara mereka.

- *Skalabilitas*: Mengingat sifat IoT yang terus berkembang dan prediksi serta perhitungan oleh [3] dan [2], kerangka kerja integrasi yang efisien harus dapat diskalakan dan cukup berevolusi untuk mendukung miliaran hal yang akan segera terhubung ke internet.
- *Kemudahan pengujian*: Kerangka kerja integrasi harus mendukung kemudahan pengujian dan debugging. Kerangka kerja ini harus menyediakan dukungan untuk men-debug cacat dan kegagalan, pengujian integrasi, pengujian komponen, pengujian sistem, pengujian kompatibilitas, pengujian instalasi, pengujian fungsional dan non-fungsional, pengujian kinerja, dan pengujian keamanan.
- *Kemudahan pengembangan*: Kerangka kerja integrasi IoT harus menyediakan sarana pengembangan yang mudah bagi pengembang. Kerangka kerja harus mengecualikan semua kerumitan dan menyediakan dokumentasi yang tepat untuk non-pengembang dan pengembang dengan pengetahuan pemrograman dasar untuk dengan mudah memahami bagian dalam kerangka kerja.
- *Toleransi terhadap kesalahan*: Sistem IoT harus dapat diandalkan dan tangguh. Kerangka kerja integrasi yang cerdas harus secara efektif menangani kesalahan karena perangkat IoT pada akhirnya dapat beralih antara kondisi offline dan online. Kerangka kerja ini harus menyediakan mekanisme penyembuhan sendiri untuk kesalahan transien (kesalahan jaringan, kesalahan tingkat node, dll.), kesalahan akses yang tidak sah, kegagalan crash server, kegagalan kelalaian (ketika server tidak menerima permintaan yang masuk dari klien), kesalahan pengaturan waktu, dll.
- *Implementasi* yang *ringan*: Kerangka kerja integrasi harus memiliki overhead yang ringan baik dalam tahap pengembangan maupun penerapan. Kerangka kerja ini harus ringan dan mudah dipasang, dicopot, diaktifkan, dinonaktifkan, diperbarui, dibuat versi, dan mudah beradaptasi.
- *Koordinasi layanan*: Koordinasi layanan adalah orkestrasi dan koreografi layanan. Orkestrasi layanan adalah koordinasi beberapa layanan oleh mediator yang bertindak sebagai komponen terpusat. Koreografi layanan di sisi lain, adalah rantai layanan bersama-sama untuk melaksanakan transaksi tertentu. Kerangka kerja integrasi harus mendukung setidaknya salah satu atau keduanya untuk mencapai keandalan.
- *Operabilitas antar domain*: Kerangka kerja harus dapat diperluas untuk mendukung komunikasi antar domain. Sebagai contoh, dalam domain mobil pintar, kerangka kerja integrasi juga harus menyediakan dukungan untuk komunikasi dan interaksi dengan lampu lalu lintas, penutupan jalan, dll. yang termasuk dalam domain kota pintar.

Terlepas dari komunitas penelitian atau perbedaan dalam penelitian, semuanya bertujuan untuk mencapai ekstensibilitas, fleksibilitas, skalabilitas, penggunaan ulang desain, dan penggunaan ulang implementasi. Sub-bagian berikutnya akan menyajikan gambaran umum dari beberapa kerangka kerja IoT.

3.1 Kerangka Kerja Eclipse Smarthome

Kerangka kerja Eclipse Smart Home (ESH) dirancang untuk memudahkan penyelesaian sistem dan masalah IoT oleh para pengembang yang diuntungkan oleh antarmuka, aturan otomasi, mekanisme persistensi, dan implementasi SOA [1]. ESH adalah kerangka kerja koneksi dan integrasi untuk domain rumah pintar IoT dan tidak bergantung pada fitur konektivitas perangkat keras, tetapi lebih menekankan pada implementasi *konektor* ke kerangka kerja. Konektor ini disebut *pengikatan* dan diharapkan dapat

diimplementasikan setidaknya satu kali dan hanya satu kali untuk protokol komunikasi tertentu. ESH telah menjadi sangat terkenal karena bersifat open source dan dengan demikian, diimplementasikan secara luas sebagai solusi rumah pintar oleh pasar yang besar. Hal ini berkontribusi pada banyaknya API bersama yang tersedia untuk berbagai produk komersial [1].
ESH secara eksplisit terpaku pada otomatisasi rumah dan dibangun di atas lima tumpukan utama [4]. Ini adalah:

- *Sistem Operasi*: Ini adalah dukungan inti untuk fungsi dasar komputer. ESH dapat berjalan dengan baik di Linux (Ubuntu, distribusi Linux berbasis Yocto), macOS, dan Windows.
- *Wadah Aplikasi atau Lingkungan Runtime*: Wadah ini sepenuhnya ditulis dalam bahasa Java dan menggunakan OSGi Runtimes (Eclipse Equinox) bersama dengan Apache Karaf dan membundelnya dengan server HTTP Jetty.
- *Komunikasi dan Konektivitas*: Penerimaan ESH yang luas telah membuat implementasinya sangat luas, sehingga menyediakan konektivitas dan komunikasi antara berbagai produk otomatisasi rumah yang siap pakai. Contohnya adalah, Belkin WeMo, Philips Hue, Sonos, dll. Karena fokusnya secara khusus pada otomatisasi rumah, maka ESH menyediakan dukungan untuk kemampuan komunikasi offline paradigma "Intranet of Things".
- *Manajemen Data dan Pengiriman Pesan*: Pendekatan SOA dari kerangka kerja ESH melihat implementasinya pada "*Event Bus*" internal. Bus ini diakses dan diekspos secara eksternal melalui pengikatan protokol yang diimplementasikan, misalnya SSE atau MQTT. Struktur ESH juga menyediakan mekanisme ketekunan untuk penyimpanan basis data dan juga mesin aturan untuk mengatur perilaku runtime.
- *Manajemen Jarak Jauh*: Kerangka kerja ESH dirancang untuk menyediakan pemantauan jarak jauh, pembaruan firmware, dan konfigurasi perangkat yang terhubung.

ESH terutama berfungsi sebagai perangkat lunak yang dimasukkan ke dalam perangkat keras untuk menyediakan titik kolektif untuk koordinasi konektivitas berbagai hal antara satu sama lain dan ke jaringan eksternal. Kerangka kerja ESH digabungkan dan diimplementasikan oleh openHAB untuk menyediakan perangkat lunak sumber terbuka untuk memudahkan pengembangan aplikasi IoT.

3.2 Kerangka Kerja Calvin

Kerangka kerja Calvin [18] adalah kerangka kerja hibrida dari model pemrograman IoT dan Cloud untuk menjelaskan kompleksitas komputasi terdistribusi, bahasa pemrograman yang beragam, dan protokol komunikasi. Kerangka kerja ini dikembangkan dengan mengkonsolidasikan teori-teori dari *model Actor* dan *Flow Based Computing.* Kerangka kerja ini membagi pengembangan aplikasi IoT menjadi empat aspek yang terpisah dan dieksekusi secara berurutan, yaitu:

- *Jelaskan*: Dalam pengertian primitif, Calvin mengkarakterisasi sebuah *aktor* yang terdiri dari tindakan, port komunikasi dan kondisi yang dapat memicu sebuah tindakan. Aktor-aktor ini berkomunikasi satu sama lain melalui port komunikasi dengan pertukaran token. Dalam Calvin, aktor adalah komponen perangkat lunak yang mencerminkan: perangkat, layanan, dan perhitungan. Pada aspek "describe", pengembang merinci dan mengekspresikan tindakan, hubungan input/output dan prasyarat yang diperlukan untuk memicu tindakan dari aktor. Dalam aspek ini juga diperlukan untuk memprioritaskan urutan antara tindakan sambil memisahkan aktor dari sumber daya yang diwakilinya untuk meningkatkan kecepatan pergeseran aktor dari satu runtime ke runtime lainnya.
- *Hubungkan*: Tahap selanjutnya adalah menghubungkan port-port dari aktor-aktor yang telah dijelaskan dengan menggunakan bahasa intuitif dan deklaratif yang ringan yang disebut *CalvinScript*. Parameter yang dibutuhkan oleh aktor disertakan dan pola koneksi mereka diinisialisasi dalam aspek ini

- *Menyebarkan*: Aspek ini berfokus pada penyebaran komponen yang dijelaskan dan terhubung dalam lingkungan runtime yang tidak diservis. Aktor yang dideskripsikan dan terhubung bersifat ringan sehingga memungkinkan migrasi antara jaringan mesh runtime. Aspek "deploy" mengimprovisasi pola penyebaran sederhana dengan meneruskan skrip aplikasi dari aplikasi yang dimaksudkan untuk disebarkan ke dalam lingkungan runtime. Skrip aplikasi selalu disebarkan ke runtime terdekat yang dapat diakses. Runtime kemudian di-instansiasi dan semua aktor terhubung secara lokal. Karena sifat terdistribusi dari lingkungan runtime yang diimplementasikan, aktor dapat melakukan manuver di seluruh runtime yang dapat diakses : lokalitas, persyaratan kinerja, persyaratan konektivitas, dan sumber daya. Dalam aspek ini, algoritma penerapan yang ditentukan mempertimbangkan beberapa faktor yang dapat mempengaruhi atau memengaruhi kinerja aplikasi, beban kerja pada aktor yang ada, kemacetan jaringan, dan kelebihan beban waktu proses.
- *Mengelola*: Aspek ini melibatkan pengelolaan migrasi aktor antara runtime, pemulihan kesalahan, penskalaan, penggunaan sumber daya, dan pembaruan oleh lingkungan eksekusi terdistribusi.

Kerangka kerja Calvin menggabungkan model IoT dan Cloud dengan merancang sistem IoT untuk memanfaatkan sistem Cloud untuk melakukan perhitungan dan komputasi yang kompleks yang jarang dilakukan oleh aktor yang memiliki keterbatasan sumber daya. Dengan demikian, kerangka kerja ini menetapkan API runtime untuk koneksi dan komunikasi antara runtime dan aktor. Model kerangka kerja Calvin mengusulkan lingkungan runtime terdistribusi dan menunjukkan multi-tenancy karena aktor dapat berbagi runtime dengan aktor dari aplikasi lain. Hal ini juga mendukung pembatasan aktor yang mengkonsumsi sumber daya tinggi pada aspek "*manage*". Sebagai contoh, aktor pemrosesan gambar terkadang dapat sepenuhnya dibatasi atau dibatasi dari runtime.

3.3 SOCRADES

SOCRADES [20] adalah arsitektur integrasi berbasis layanan berorientasi layanan yang menyediakan komponen umum untuk membantu pemodelan proses yang sangat terperinci. Arsitektur ini menargetkan objek-objek cerdas dalam domain manufaktur, yang merepresentasikan perilaku mereka sebagai layanan web untuk meningkatkan kemampuan mereka. SOCRADES menggabungkan pola, konsep, dan kode SIRENA [6] (proyek yang didanai oleh Uni Eropa) untuk mengusulkan dan merancang infrastruktur integrasi untuk layanan web dan kerangka kerja untuk pengawasan perangkat dan siklus hidup yang cacat dalam SIRENA.

SOCRADES terdiri dari empat bagian utama, yaitu *lapisan Perangkat*, *perangkat lunak tengah SOCRADES* (terdiri dari dua sub bagian, yaitu *bagian Aplikasi* dan *bagian Layanan Perangkat*), aplikasi *xMII* dan *Enterprise.*

- *Lapisan Perangkat*: Lapisan ini terdiri dari perangkat yang diaktifkan layanan web dan terhubung ke lapisan perangkat lunak tengah SOCRADES menggunakan model Device Profile for Web Service (DPWS).
- *SOCRADES Middle-ware*: Perangkat di lapisan perangkat dapat terhubung ke Aplikasi Perusahaan karena mendukung layanan web. Middle-ware SOCRADES berfungsi sebagai jembatan antara lapisan perangkat dan aplikasi perusahaan. Tujuan utama dari komponen ini adalah untuk menyederhanakan pengelolaan perangkat di lapisan perangkat. Beberapa fitur dan komponen lain dari SOCRADES adalah: akses ke perangkat, penemuan layanan, pengawasan perangkat, manajemen siklus hidup layanan, katalog layanan lintas lapisan, dan dukungan keamanan (opsional). Komponen middle-ware SOCRADES juga memperluas fungsionalitas komponen xMII ke komponen lain dan sebaliknya.
- *Komponen xMII*: Komponen ini adalah Sistem Aplikasi dan Produk (SAP) dalam produk pemrosesan data: SAP xApp Manufacturing Integration and Intelligence (SAP xMII). Ini memiliki fitur untuk:

perangkat layanan non-web yang memungkinkan konektivitas, layanan visualisasi, pemodelan grafis, eksekusi aturan bisnis, dan konektivitas ke perangkat lunak SAP sebelumnya melalui protokol tertentu. Komponen ini diintegrasikan ke dalam GUI Aplikasi Perusahaan dalam bentuk mash-up dengan menghasilkan konten web yang kaya sesuai dengan tujuan.

- *Aplikasi Perusahaan*: Komponen ini terdiri dari GUI yang mengekspos sistem kepada pengguna. Komponen ini menerima data dari komponen xMII melalui protokol tertentu atau dengan mengintegrasikan konten web xMII dengan GUI Aplikasi Perusahaan.

SOCRADES secara praktis merupakan arsitektur integrasi layanan web yang dirancang untuk mendukung konektivitas perangkat dan integrasi ke dalam aplikasi perusahaan seperti dalam sistem ERP. Secara ringkas, SOCRADES didasarkan pada paradigma SOA.

3.4 AllJoyn

AllJoyn [5] adalah kerangka kerja sumber terbuka yang menargetkan koneksi dan integrasi berbagai hal terlepas dari modul komunikasi, sistem operasi, dan produsennya. Kerangka kerja ini menyediakan penemuan jaringan proksimal antar perangkat dengan mengabstraksikan detail transportasi fisik dan menyediakan API yang mudah digunakan untuk menghubungkan berbagai hal. Dengan demikian, kerumitan dalam menemukan perangkat terdekat ditangani dengan membuat sesi (beberapa sesi, sesi point-to-point atau grup) antar perangkat untuk komunikasi yang aman di antara mereka. Kerangka kerja AllJoyn terdiri dari beberapa layanan umum yang diimplementasikan dan antarmuka yang digunakan oleh pengembang untuk mengintegrasikan berbagai perangkat, benda, atau aplikasi. Kerangka kerja ini secara opsional bergantung pada layanan cloud karena berjalan pada jaringan lokal. Hal ini memungkinkan perangkat dan aplikasi untuk berkomunikasi dalam jaringan hanya dengan satu agen gateway yang dirancang untuk terhubung ke internet. Hal ini pada gilirannya mengurangi ancaman keamanan dan jumlah perangkat yang terpapar ancaman internet. Kerangka kerja ini terdiri dari dua komponen utama: *Aplikasi AllJoyn* dan *Router AllJoyn*. Komponen-komponen ini dapat berada pada perangkat fisik yang sama atau berbeda.

- *Aplikasi AllJoyn*: Aplikasi adalah komponen kerangka kerja yang berkomunikasi langsung dengan AllJoyn Router dan berkomunikasi dengan Aplikasi lain melalui router. Aplikasi AllJoyn terdiri dari sub komponen : *Kode Aplikasi AllJoyn, Kerangka Kerja Layanan AllJoyn,* dan *Pustaka Inti AllJoyn*.
  - *Kode Aplikasi AllJoyn*: Ini memegang logika untuk Aplikasi AllJoyn. Kode ini menyediakan akses ke API Inti AllJoyn dengan menghubungkannya dengan kerangka kerja Layanan AllJoyn atau komponen Perpustakaan Inti AllJoyn untuk menyediakan akses tersebut.
  - *Kerangka Kerja Layanan AllJoyn*: Ini adalah komponen yang mengimplementasikan layanan umum seperti on-boarding perangkat baru untuk pertama kalinya, mengirim pemberitahuan dan mengontrol perangkat. Layanan ini memungkinkan komunikasi dan interoperabilitas antara aplikasi dan perangkat.
  - *Perpustakaan Inti AllJoyn*: Komponen ini menyediakan akses ke API untuk interaksi dengan jaringan AllJoyn. Komponen ini menyediakan dukungan untuk: pembuatan sesi, pembuatan dan penanganan objek, definisi antarmuka metode, properti dan sinyal, serta penemuan dan pengiklanan layanan/perangkat.
- *Router AllJoyn*: Komponen ini memfasilitasi komunikasi antara berbagai komponen Aplikasi yang berbeda. Kerangka kerja AllJoyn terdiri dari tiga pola umum komunikasi antara Aplikasi dan Router.

- *Router yang dibundel*: Di sini, Router dibundel dengan Aplikasi berdasarkan hubungan satu ke satu.
- *Router Mandiri*: Router dijalankan sebagai proses mandiri pada sistem dan memungkinkan koneksi ke beberapa Aplikasi pada perangkat yang sama dengan Router. Hal ini memastikan bahwa perangkat mengkonsumsi lebih sedikit sumber daya secara keseluruhan.
- *Router pada perangkat yang berbeda*: Router dijalankan pada perangkat yang berbeda yang memungkinkan koneksi dari Aplikasi. Ini digunakan untuk perangkat tertanam yang memiliki keterbatasan sumber daya dan dengan demikian, akan menggunakan versi "*Thin*" dari kerangka kerja AllJoyn untuk mengatasi keterbatasan sumber daya.

Kerangka kerja AllJoyn secara ringkas mengejar landasan bersama untuk koneksi, interaksi, dan integrasi berbagai hal, perangkat, dan aplikasi tanpa memandang OS, bahasa pemrograman, dan pembuatnya.

3.5 FRASAD

FRASAD [17] adalah kerangka kerja pengembangan yang bertujuan untuk memungkinkan pengembang merancang aplikasi IoT mereka menggunakan konsep domain node sensor. Konsep ini digerakkan oleh model dan dengan demikian kode aplikasi dihasilkan dari model yang dirancang melalui proses transformasi. Kerangka kerja FRASAD (**FRAmework** for Sensor Application Development) merupakan perluasan dari [22] dengan penambahan dan integrasi dua lapisan pada arsitektur node sensor yang sudah ada. Dua lapisan tambahan ini adalah *Application Layer* (APL) dan *Operating System Abstraction Layer* (OAL). Inti dari kedua lapisan ini adalah untuk memperbesar tingkat abstraksi dan dengan demikian menyembunyikan tingkat yang lebih rendah. Untuk mencapai hal ini, kerangka kerja ini menggunakan penggunaan Domain Specific Language (DSL) yang dirancang dengan kuat untuk memodelkan node sensor dan memisahkan sistem operasi dari aplikasi. OAL kemudian dikontrak untuk menjelaskan aplikasi yang dimodelkan berdasarkan sistem operasi tertentu untuk diimplementasikan. OAL dapat dilihat sebagai generator aplikasi: fungsi intinya adalah untuk menghasilkan kode aplikasi yang akan digunakan dalam platform yang ditargetkan. Kerangka kerja FRASAD secara inheren mengikuti pendekatan *Model Driven Architecture* (MDA) dengan mengadopsi tiga model/tingkat abstraksi. Mereka adalah:

- *Computation Independent Model* (CIM): digunakan untuk merepresentasikan informasi yang sebenarnya tanpa memperlihatkan struktur sistem atau teknologi yang digunakan untuk implementasinya.
- *Model Platform Independen* (PIM): Ini adalah model yang menampung logika dan persyaratan aplikasi.
- *Model Spesifik Platform* (PSM): PIM diterjemahkan ke PSM berdasarkan sistem operasi tertentu yang digunakan, menggunakan DSL yang dirancang khusus untuk proses pemetaan. PSM adalah sistem operasi yang spesifik dan dapat menggunakan bahasa yang didukung oleh sistem operasi.

Singkatnya, FRASAD adalah kerangka kerja yang menggunakan MDA berlapis-lapis dengan interaksi antara pengguna awam melalui beberapa antarmuka yang telah ditentukan. Frasad menggunakan model node-sentris untuk memfasilitasi pemrograman masing-masing node sensor menggunakan model pemrograman berbasis aturan.

3.6 ARIoT

Kerangka kerja ARIoT [12], mengkonsolidasikan teknologi *Augmented Reality* (AR) dengan sedikit memperluas infrastruktur IoT. Inti dari kerangka kerja ini adalah untuk: secara dinamis mengidentifikasi objek-objek yang ditargetkan IoT dalam jarak dekat dari ruang IoT, pengenalan dan pelacakan informasi fitur objek IoT (contoh

adalah, id objek, nilai atribut, produsen objek, dll.) dan menambah konten interaktif objek IoT sebagai layanan kepada klien.

Dalam kerangka kerja ini, objek IoT disiapkan untuk menyimpan informasi fitur dan konten untuk layanan. Sistem klien AR (misalnya ponsel) menggunakan teknologi AR untuk mendeteksi objek IoT di sekitar yang memenuhi kriteria pengaturan, melalui protokol penemuan layanan nirkabel standar yang ditentukan. Setelah objek IoT diidentifikasi melalui protokol standar, ada pertukaran pesan antara klien AR dan objek IoT untuk mengidentifikasi lebih lanjut dan menyaring objek berdasarkan jarak dan perspektifnya (jarak relatif dari objek lain atau arahnya relatif terhadap klien AR). Klien AR kemudian dapat menggunakan fitur dan informasi konten dari pesan yang dipertukarkan untuk melacak objek yang ditargetkan, dan menambah *konten* pada layar tampilan klien. Objek IoT individu secara independen dikaitkan dengan mode pengenalan dan pelacakan mereka sendiri, misalnya objek IoT 1 dapat mengirim informasi untuk algoritme Speeded Up Robust Features (SURF) untuk pelacakan dan objek IoT 2 dapat menggunakan model / algoritme yang berbeda untuk pelacakan.

Singkatnya, ARIoT mengintegrasikan teknologi AR ke dalam infrastruktur IoT untuk menyediakan lingkungan yang lebih ramah untuk memilih objek target dan menjadikannya sebagai sasaran kontrol, interaksi, dan pelacakan. Klien AR sebagian besar adalah perangkat seluler yang berisi algoritme yang diimplementasikan untuk mengidentifikasi, mengenali, dan melacak objek IoT melalui protokol standar yang telah ditentukan. Kerangka kerja ini meningkatkan skalabilitas dengan menghilangkan penggunaan server pusat melalui klien AR, yang memungkinkan objek IoT untuk secara langsung berkomunikasi dan bertukar informasi berdasarkan kebutuhan.

3.7 AVIoT

AVIoT [11] adalah kerangka kerja IoT untuk memvisualisasikan dan mengelola objek IoT yang ada di lingkungan tertentu (misalnya rumah pintar). Kerangka kerja ini bertujuan untuk memungkinkan pengguna menerapkan alat bantu penulisan visual berbasis web untuk mengabstraksikan dan memprogram perilaku benda-benda IoT. Oleh karena itu, pengguna akhir dapat dengan mudah memantau dan mendefinisikan perilaku benda-benda IoT tanpa pengetahuan internal sebelumnya tentang arsitektur atau sistem koneksi sensor. Kerangka kerja AVIoT diusulkan untuk memungkinkan konfigurasi visual dan manajemen hal-hal dalam lingkungan IoT dengan mudah. Kerangka kerja ini mengikuti proses berprinsip dari: Abstraksi Sensor dan Aktuator, Visualisasi IoT Interaktif, dan Penulisan IoT Interaktif.

- *Abstraksi Sensor dan Aktuator*: Dalam proses ini, benda-benda fisik yang ada di lingkungan IoT diabstraksikan dan divirtualisasikan untuk berinteraksi dengan virtual lainnya dengan mendefinisikannya sebagai node. Node ini akan didefinisikan berdasarkan fitur umum benda-benda seperti nama, jenis benda, jenis visualisasi, posisi, anak (jika ada) dan perilaku string fungsional (berisi kode skrip yang dapat dinilai pada saat runtime). Benda-benda diabstraksikan dan divirtualisasikan untuk berisi *sensor fisik, sensor virtual, aktuator virtual*, dan *aktuator fisik.* Sementara sensor dan aktuator fisik adalah benda-benda di dunia nyata, fungsi inti sensor virtual adalah mendeteksi peristiwa penting yang diminati oleh pengguna akhir. Fungsi inti aktuator virtual adalah menyesuaikan perilaku yang telah ditentukan sebelumnya oleh pengguna akhir dengan aktuator fisik. Aktuator virtual tidak menyimpan nilai seperti sensor virtual, melainkan memicu tindakan aktuator fisik. Interaksi antara komponen yang diabstraksikan mengikuti urutan *sensor fisik -> sensor virtual -> aktuator virtual -> aktuator fisik*.
- *Visualisasi IoT Interaktif*: Ini adalah visualisasi benda-benda virtual di dalam lingkungan IoT untuk mempromosikan pengelolaan benda-benda fisik. Fase ini menggunakan *klien* - browser web atau aplikasi web dan *server* untuk memfasilitasi visualisasi dan pengelolaan hal-hal virtual. Asumsinya adalah bahwa server telah mengidentifikasi perangkat dan menyimpan informasi tentang berbagai modul konektivitasnya. Server juga digunakan sebagai tempat penyimpanan informasi yang berkaitan dengan visualisasi

dan pembuatan berbagai hal. Fase ini memanfaatkan REST sebagai antarmuka untuk komunikasi antara klien dan server. Untuk visualisasi 3D, server menyimpan denah virtual lingkungan dalam ruangan dalam keadaan model 3D. Hal ini dapat diminta oleh klien yang sedang berjalan untuk memberikan visualisasi 3D pada saat runtime. Visual 3D direnderer seperti bingkai kawat dan ikon visual ditampilkan berdasarkan lokasi benda-benda fisik untuk mengidentifikasi sensor dan aktuator.

- *Penulisan IoT Interaktif*: Pada fase ini, benda-benda virtual dapat diedit, ditambahkan, atau dihapus dari kumpulan benda-benda fisik yang diberikan. Benda-benda juga dapat dipindahkan oleh pengguna akhir ke posisi yang berbeda dan ditemukan kembali di lokasi baru oleh server. Sensor fisik berinteraksi dengan dunia nyata dan menghasilkan data numerik yang diamati yang diterima oleh sensor virtual. Fokus utama sensor virtual adalah pada peristiwa (yang telah dikonfigurasi sebelumnya) yang menjadi perhatian pengguna akhir. Informasi ini diterjemahkan dan diteruskan ke aktuator virtual yang kemudian bertindak dengan memicu tindakan yang sesuai dari aktuator fisik.

AVIoT adalah kerangka kerja IoT untuk visualisasi dan pembuatan berbagai hal lingkungan cerdas oleh pengguna akhir. Kerangka kerja ini diimplementasikan di web dan sepenuhnya berfokus pada lingkungan IoT dalam ruangan. Kerangka kerja ini memungkinkan pengguna akhir untuk mengabstraksikan dan mendefinisikan berbagai hal sesuai urutan hierarki dengan memanfaatkan alat aplikasi web untuk visualisasi dan penulisan.

3.8 Ringkasan

Setiap kerangka kerja IoT yang dibahas, telah menerapkan berbagai arsitektur perangkat lunak (MDA, SOA, dll.). Arsitektur yang paling menonjol dan produktif adalah SOA. SOA cukup menguntungkan dalam kerangka kerja integrasi IoT karena menyediakan pemisahan kontrak dan interoperabilitas skeptis protokol heterogen Kerangka kerja yang memiliki kekurangan dan kelebihan yang disebabkan oleh arsitektur yang dipilih untuk memberikan solusi IoT atau pola implementasinya. Sebagai contoh, ESH memiliki kekuatan dalam memberikan dukungan untuk protokol komunikasi heterogen sambil menyediakan sarana untuk membuat "*binding*" untuk protokol baru. ESH mendukung berbagai macam produk siap pakai, mudah dikembangkan dan digunakan. Terlepas dari itu, ESH terbatas pada rumah pintar, ini tergantung pada bus perusahaan untuk transportasi pesan yang mungkin menjadi satu titik kegagalan, ESH tidak memiliki dukungan untuk menggabungkan layanan misalnya, jika perangkat X memerlukan layanan komposit suhu dan kelembapan, ia harus membuat permintaan individual untuk masing-masing layanan ini terutama jika layanan tersedia pada dua perangkat yang berbeda Y dan Z. Komponen ESH dirancang secara monolitik dan membuatnya cukup sulit untuk dimasukkan ke dalam sistem yang lebih besar sebagai sub sistem.

Kekuatan yang jelas dari kerangka kerja Calvin adalah fleksibilitasnya karena penggunaan sistem cloud. Terlepas dari kekuatan ini, kerangka kerja Calvin menunjukkan beberapa keterbatasan seperti, pengembang perlu mempelajari bahasa samar yang baru, tidak ada kombinasi layanan, tidak ada ketentuan untuk manajemen layanan dan ada ancaman yang tinggi terhadap keamanan sistem, misalnya ketika aktor bermigrasi antar runtime, tidak ada pertimbangan untuk faktor ancaman keamanan. Oleh karena itu, sebuah aktor yang keamanannya telah dikompromikan akan menjadi ancaman bagi beberapa aktor lain pada runtime yang berbeda.

Kerangka kerja SOCRADES menyediakan dukungan untuk sistem heterogen, komponen Aplikasi Perusahaannya menyajikan GUI yang mudah digunakan oleh pengguna sehingga mengurangi waktu penerapan dan memberikan kemudahan dalam manajemen sistem. Meskipun demikian, kegagalan titik tunggal hadir dalam penggunaan bus layanan perusahaan, tidak ada pengetahuan intrinsik tentang layanan dan tidak memiliki dukungan untuk menggabungkan layanan berbutir halus ke layanan kasar sesuai permintaan.

Yang jelas, implementasi SOA yang monolitik, dengan satu titik kegagalan yang jelas dan kompleksitas dalam arsitektur karena miliaran hal yang harus disediakan, semakin memperumit masalah skalabilitas,

toleransi kesalahan dan keandalan. Kerangka kerja koneksi dan integrasi IoT harus cukup terukur untuk mendukung miliaran hal yang terhubung, harus mentoleransi kesalahan umum seperti latensi dan menyediakan sarana untuk mengidentifikasi, menggabungkan, membuat ulang, dan memberikan akses jarak jauh yang aman ke layanan ini.

4 Pendekatan yang Diusulkan

Dalam membangun kerangka kerja yang dapat diskalakan, adaptif, dan toleran terhadap kesalahan untuk IoT, gaya arsitektur sistem terdistribusi merupakan pendekatan yang menguntungkan. Dengan cara ini, komponen perangkat lunak dapat diakses dari jarak jauh, melalui berbagai protokol akses jarak jauh. Gaya arsitektur ini terdiri dari beberapa pola yang meliputi: *Klien-server (2-tier, 3-tier atau n-tier), Peer-to-Peer, Perantara permintaan objek, Arsitektur berbasis ruang, Arsitektur tidak ada yang dibagikan, REST, Berorientasi layanan, Pola komputasi awan,* dll. Untuk mencapai kerangka kerja IoT yang dapat diskalakan, dapat dikembangkan, dan toleran terhadap kesalahan, pola arsitektur perangkat lunak yang saat ini digunakan, harus ditingkatkan untuk mengakomodasi paradigma IoT dan memenuhi persyaratan minimal yang disajikan. Kami mengusulkan adopsi *Microservices* yang merupakan arsitektur berbasis layanan (pola berorientasi layanan). Microservices dicirikan oleh penggunaan API sederhana yang berlapis tipis (ringan dibandingkan dengan SOA). Meskipun orang-orang yang skeptis mungkin berpendapat bahwa layanan mikro sama dengan SOA, kesamaannya adalah bahwa keduanya merupakan arsitektur berbasis layanan dengan penekanan pada penggunaan dan penggunaan ulang layanan. Keduanya berbeda dalam hal *gaya arsitektur, karakteristik arsitektur, karakteristik* dan *kemampuan layanan.* Istilah layanan dalam konteks SOA, didefinisikan oleh [15], sebagai "*mekanisme untuk memungkinkan akses ke satu atau lebih kemampuan, di mana akses disediakan menggunakan antarmuka yang ditentukan dan dilakukan konsisten dengan batasan dan kebijakan seperti yang ditentukan oleh deskripsi layanan*". Definisi ini mendefinisikan layanan berdasarkan kepemilikan, aksesibilitas, dan atribut kualitatifnya. Namun, layanan dapat didefinisikan lebih lanjut berdasarkan *atomitas* (ukuran layanan) dan interaksinya (dalam hal penyatuan layanan). Atomitas layanan dan klasifikasi, menyajikan fitur khas yang besar antara SOA dan layanan mikro. Interaksi antara layanan yang berbeda dapat ditangani dengan mudah dalam arsitektur layanan mikro dan dengan demikian akan menguntungkan dalam sistem IoT yang secara intrinsik tidak sinkron.

4.1 Layanan mikro untuk IoT

Tujuan utama dari layanan mikro adalah untuk mengoordinasikan aplikasi terdistribusi sebagai implementasi kolektif layanan yang berjalan secara otonom dan independen berdasarkan konteks proses yang dijelaskan [19]. Pertumbuhan teknologi telah membuat perangkat IoT berevolusi dengan protokol komunikasi yang berbeda, sehingga menyulitkan untuk mencapai konektivitas, skalabilitas, dan integrasi terutama dalam kerangka kerja yang dibuat secara monolitik. Tantangan lainnya adalah transparansi dukungan untuk protokol lama dengan atau tanpa peningkatan. Dengan kebutuhan untuk mengurangi konsumsi daya untuk mencapai lingkungan yang ramah lingkungan, perangkat IoT telah dirancang untuk mengkonsumsi lebih sedikit energi dan dengan demikian, membatasi sumber dayanya (penyimpanan, kapasitas proses, dll.). Hal ini juga menjadi tantangan dalam IoT dari sisi perangkat. Perangkat diharapkan dapat berjalan dalam jangka waktu yang lama, memproduksi dan mengkonsumsi layanan, bertukar data dan muatan yang tinggi, dll. Semua fungsi inti ini pada gilirannya akan mengganggu masa pakai baterai mereka.

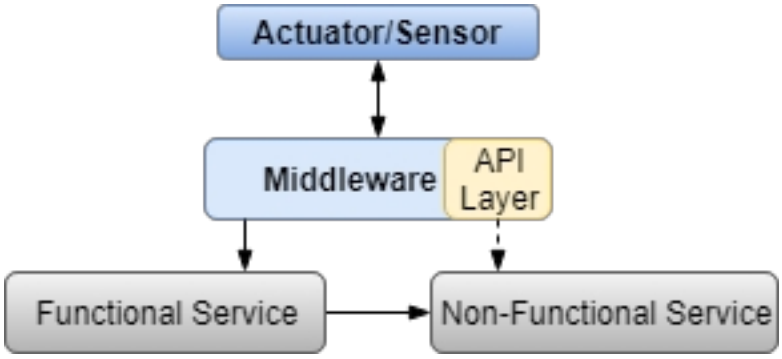
Arsitektur layanan mikro (berbagi sesedikit mungkin) menghadirkan kemungkinan untuk menyelesaikan masalah-masalah ini karena arsitektur ini mendorong penyebaran layanan secara independen, atomisasi layanan, mencegah kegagalan pada satu titik, meningkatkan manajemen transaksi keamanan dan koreografi layanan. Sub-bagian berikutnya menyajikan desain yang dibayangkan dari sistem IoT dalam arsitektur layanan mikro, berdasarkan pada beberapa masalah yang menantang yang kompleks dalam IoT.

4.1.1 Identifikasi dan Klasifikasi Layanan

Layanan mikro mengklasifikasikan layanan menjadi

- *Layanan Fungsional*: layanan yang mendukung fungsi operasional sistem pintar dalam domain IoT. Dalam IoT, layanan ini sebagian besar bersifat literal (yaitu angka, huruf, dll.). Layanan ini terbuka untuk digunakan oleh sistem atau perangkat eksternal.
- *Layanan Non-Fungsional*: layanan yang terkait dengan tugas non-operasional (seperti: Pencatatan, otentikasi, pemantauan, pembuatan, audit, dll.) tetapi harus digunakan untuk pengoperasian sistem yang andal. Layanan ini tidak terekspos dan digunakan oleh sistem atau perangkat yang ingin berintegrasi ke dalam sistem IoT yang sudah ada.

Konsep taksonomi layanan (deskripsi layanan) seperti yang terlihat dalam arsitektur layanan mikro, diadopsi ke dalam domain IoT untuk mereproduksi kerangka kerja arsitektur IoT untuk memfasilitasi identifikasi dan klasifikasi layanan.

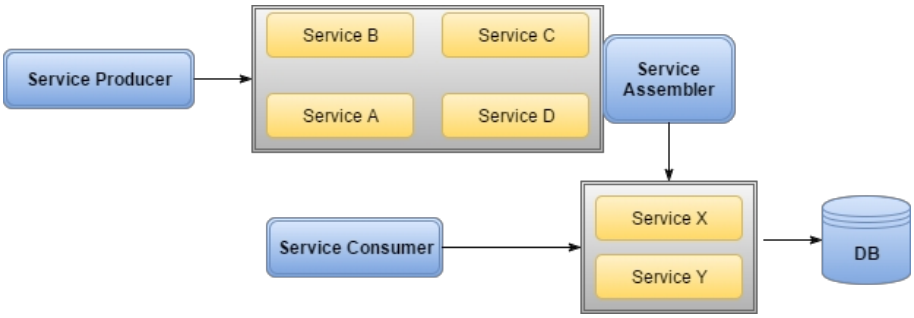


Gambar 2: Layanan Mikro untuk Identifikasi dan Klasifikasi Layanan IoT

Layanan fungsional dan non-fungsional diidentifikasi dan dipisahkan dengan jelas. Lapisan API menerima permintaan dan dapat memicu layanan non-fungsional dengan sendirinya atau melalui layanan fungsional untuk menyembunyikan layanan non-fungsional dari dunia nyata.

4.1.2 Atomitas Layanan

Konsep dasar dari layanan mikro adalah layanan berbutir halus yang memiliki tujuan tunggal dan dengan demikian bekerja secara optimal dalam menjalankan satu fungsi. Dalam IoT, layanan berbutir halus adalah hal yang umum terutama di antara perangkat dengan protokol komunikasi seperti MQTT, AMQP, dll. Salah satu tantangan dalam IoT adalah masalah latensi yang terkadang muncul sebagai konsekuensi dari layanan besar dan berbutir kasar yang dikonsumsi oleh perangkat IoT dengan sumber daya yang terbatas. Dalam layanan mikro, dampak dari layanan berbutir halus membantu dalam pengembangan, pengujian, penerapan, dan pemeliharaan perangkat lunak. Sistem IoT dapat memperoleh banyak manfaat dengan mengadopsi pola layanan yang dikabutkan ini. Membangun layanan kecil secara akurat dalam arsitektur layanan mikro agak rumit dan masalah serupa dapat dihadapi ketika mengadaptasinya ke dalam IoT.
Gambar 3 diperkuat agar sesuai dengan IoT dan menghilangkan masalah yang terkait dengan layanan granular. Dalam layanan mikro, masalah perincian layanan menghambat kinerja dan manajemen transaksi. Jika sebuah transaksi membutuhkan layanan A, B, dan C, maka dibutuhkan jumlah waktu T_A, T_B, dan T_C (di mana T_N adalah waktu untuk memproses layanan N) untuk diproses. Dengan demikian, jika dibutuhkan 200ms untuk memproses setiap layanan, maka dibutuhkan 600ms untuk memproses transaksi. Perakit *Layanan* diperkenalkan untuk menangani permintaan konsumen dan penggabungan layanan



Gambar 3: Layanan mikro untuk Atomitas Layanan IoT

(menggabungkan dua atau lebih layanan A, B dan/atau C untuk menghasilkan layanan fungsional baru X yang sesuai dengan kontrak layanan konsumen berdasarkan detail implementasi). Dengan diperkenalkannya komponen perakit layanan, permintaan konsumen untuk transaksi layanan komposit diproses dan disingkirkan oleh , sehingga mengurangi waktu yang dihabiskan untuk panggilan akses jarak jauh. Perakit layanan pada dasarnya menggabungkan layanan berbutir halus ke layanan komposit yang dapat diiklankan sebagai layanan terpisah tergantung pada jumlah permintaan untuk layanan atau transaksi serupa.

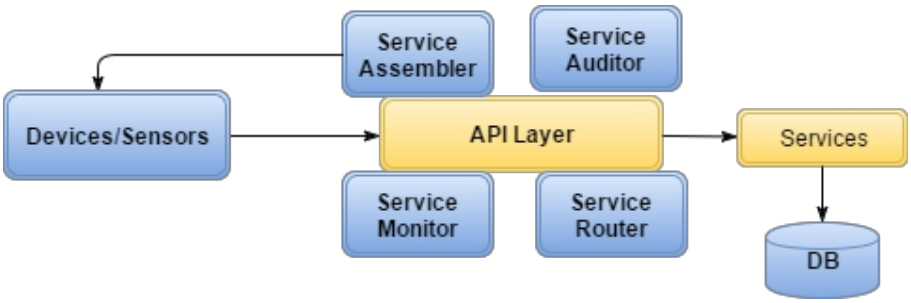
4.1.3 Koordinasi Layanan

Koordinasi layanan dapat dilihat sebagai cara sistematis dimana layanan diatur, dialokasikan dan dianalisis. Hal ini didasarkan pada dua konsep komunikasi layanan utama yaitu *Koreografi Layanan* dan Orkestrasi *Layanan.* Orkestrasi layanan adalah pengorganisasian berbagai layanan kolektif oleh komponen terpusat yang dapat dilihat sebagai *mediator. Mediator* mengkoordinasikan berbagai layanan yang diperlukan untuk menyelesaikan transaksi tertentu. Orkestrasi layanan sebagian besar dominan dalam SOA, oleh karena itu, kerangka kerja IoT yang mengadopsi paradigma SOA menggunakan pusat integrasi atau perangkat lunak tengah dalam arsitektur keseluruhan untuk menyediakan fungsionalitas ini. Kekurangan dari pendekatan ini adalah bahwa sistem dapat mengalami kelambatan dalam beberapa permintaan layanan yang diperlukan untuk menyelesaikan satu transaksi. Dalam SOA, sistem sebagian besar bergantung pada kombinasi beberapa layanan komposit (berbutir kasar) untuk menyelesaikan satu permintaan bisnis. Dalam sistem IoT, layanan biasanya berbutir halus dan dengan demikian, koreografi layanan akan cukup untuk sistem IoT skala kecil.

Seperti yang biasa digunakan dalam layanan mikro, koreografi layanan adalah pengorganisasian beberapa layanan untuk menjalankan satu transaksi tanpa mediator pusat. Hal ini memulai reaksi berantai atau service chaining, di mana satu layanan memanggil layanan lain dan layanan lain memanggil layanan lain hingga layanan terakhir yang dibutuhkan dipanggil. Meskipun hal ini tidak efisien karena akan meningkatkan waktu panggilan akses jarak jauh untuk beberapa permintaan layanan, penambahan komponen perakit layanan otonom akan bermanfaat dalam mengurangi waktu panggilan. Rantai layanan tidak rumit dalam IoT mengingat layanan berbutir halus yang sebagian besar terdiri dari literal. Perakit layanan berfungsi dengan cara menghilangkan kebutuhan akan kontrak layanan jarak jauh baru setelah *kombinasi layanan.* Komponen lain seperti auditor layanan, monitor layanan, dll. Terlibat dalam operasi runtime dari perakit layanan seperti yang terlihat pada Gambar 4

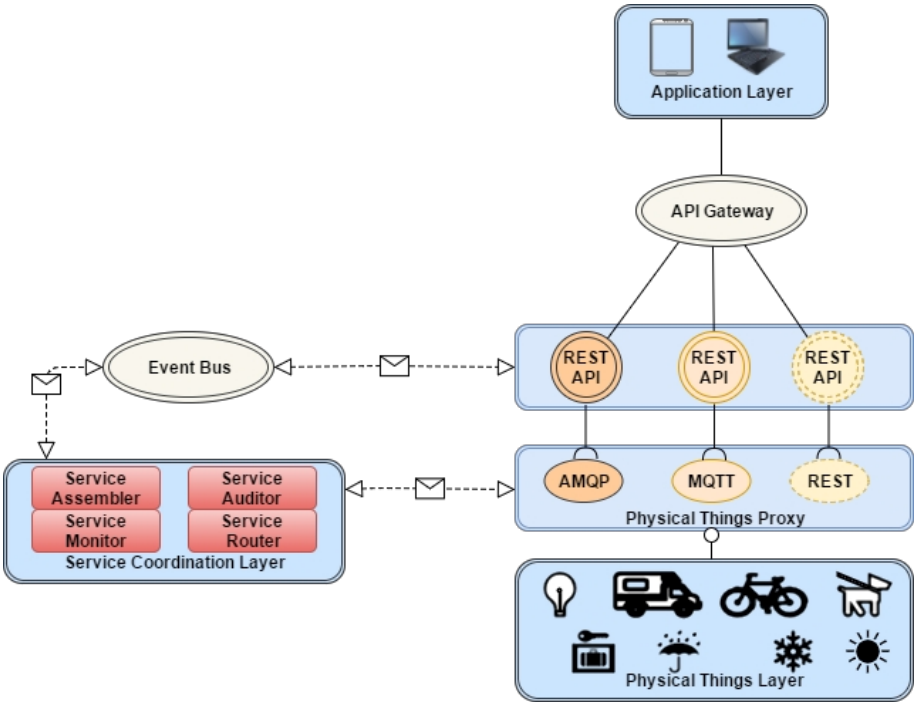
4.1.4 Interoperabilitas dalam IoT

Paradigma IoT dibangun di sekitar pembagian layanan dan untuk mencapai interoperabilitas, kerangka kerja IoT harus memungkinkan penggabungan dan penggabungan layanan. Dalam layanan mikro, Lapisan API digunakan daripada Enterprise



Gambar 4: Topologi Layanan Konsolidasi IoT

Service Bus (ESB) yang biasa digunakan dalam SOA. Manfaat dari pendekatan API untuk IoT ini adalah memungkinkan tingkat perincian layanan diubah tanpa mempengaruhi konsumen. Sebagai contoh, layanan berbutir kasar dapat dibongkar menjadi butiran yang lebih halus atau sebaliknya untuk meningkatkan kinerja, skalabilitas, dan kemudahan penerapan sistem. Dengan komponen API, konsumen tidak perlu mengubah kontrak layanan atau menyesuaikan diri. Sebaliknya, API tahu untuk memanggil layanan terpisah ketika ada panggilan ke layanan kasar. IoT akan lebih baik jika memanfaatkan lapisan API cerdas yang diimplementasikan. Tidak seperti middle-ware SOA yang menggunakan mediator untuk orkestrasi layanan, peningkatan pesan dan transformasi protokol yang diimplementasikan di dalam ESB, lapisan API cerdas untuk IoT akan menggunakan komponen yang lebih kecil. Komponen-komponen ini tidak tertutup dalam lapisan API, tetapi akan tersedia pada saat runtime untuk menjalankan fungsionalitas yang serupa, misalnya kombinasi layanan, *pemisahan kontrak*, dll. Komponen-komponen ini akan mengimbangi dukungan protokol heterogen yang tidak ada dalam arsitektur layanan mikro standar.



Gambar 5: Layanan Mikro untuk Arsitektur IoT

Gambar 5 merupakan arsitektur layanan mikro untuk IoT. Pengenalan *lapisan koordinasi layanan* adalah untuk memfasilitasi identifikasi, klasifikasi, kombinasi, dan koreografi layanan. Gambar tersebut menunjukkan beberapa layanan mikro yang mengimplementasikan berbagai antarmuka yang diperlukan (misalnya MQTT, REST, dll.) yang disediakan oleh penginderaan fisik. Sebagai contoh, jika laporan cuaca diminta dari lapisan aplikasi, lapisan ini melakukan panggilan ke API Gateway yang kemudian memanggil REST API yang sesuai dan *perintah* yang dikirim melalui Event Bus. Setelah menerima perintah, event bus mengirimkan permintaan ke lapisan koordinasi layanan (di mana identifikasi layanan yang diperlukan, penemuan, pengikatan, dan perutean dilakukan). Perintah disalurkan melalui protokol yang relevan dan perangkat fisik bertindak sesuai dengan itu. Dalam kasus aktivitas yang dirasakan secara fisik, pembaruan peristiwa dikirim melalui protokol yang disediakan ke lapisan koordinasi layanan. Pembaruan peristiwa ini didorong melalui bus peristiwa ke API REST yang sesuai dan pembaruan tercermin pada perangkat seluler.

5 Kesimpulan dan Pekerjaan di Masa Depan

Makalah ini telah mengeksplorasi masalah yang masih ada dalam mengintegrasikan perangkat dan sistem IoT. Ini telah meninjau beberapa kerangka kerja yang ada dengan banyak fokus pada paradigma SOA yang diadaptasi. Makalah ini lebih lanjut mengusulkan kerangka kerja arsitektur untuk IoT yang juga akan menguntungkan dari sudut pandang pengembang. Dengan semakin banyaknya perangkat yang terhubung ke internet, ada kebutuhan akan kerangka kerja integrasi yang sangat skalabel, dapat diperluas, dan tahan terhadap kesalahan. Keyakinan kami adalah bahwa dengan mengadopsi arsitektur layanan mikro dalam IoT untuk merombak kerangka kerja integrasi, akan meningkatkan keandalan sistem IoT. Kerangka kerja diusulkan belum sepenuhnya diimplementasikan dan diuji, tetapi ini menyajikan arah investigasi bagi komunitas IoT tentang cara-cara untuk mengatasi masalah skalabilitas, toleransi kesalahan, kemudahan penerapan, komunikasi antar domain, kemudahan pengembangan, dan implementasi yang ringan.

Referensi

[1] *Eclipse SmartHome*. Tersedia di `https://eclipse.org/smarthome/getting-started.html#developers`.

[2] *Gartner: 6,4 Miliar Orang Telah Terhubung*. Tersedia di http://www.gartner.com/newsroom/id/3165317.

[3] *Internet of Things Membutuhkan Keterbukaan dan Kolaborasi Industri untuk Berhasil, Kata CEO Samsung Electronics, BK Yoon*. Tersedia di http://www.samsung.com/uk/news/local/the-internet-of-things-needs-openness- dan-kolaborasi-industri-untuk-berhasil-mengatakan-samsung-electronics-ceo-bk-yoon.

[4] *Pengembangan IoT* menjadi *sederhana.* Tersedia di `https://iot.eclipse.org/resources/white-papers/` Eclipse%20IoT%20White%20Paper%20-%20The%20Three%20Software%20Stacks%20Required%20for%20IoT%20Architectures`.pdf.`

[5] Allseen Alliance (2016): *Kerangka Kerja AllJoyn*. Tersedia di `https://allseenalliance.org/framework/documentation/learn/architecture.`

[6] Hendrik Bohn, Andreas Bobek & Frank Golatowski (2006): *SIRENA-Infrastruktur Layanan untuk Perangkat Jaringan Tertanam secara real time: Kerangka kerja berorientasi layanan untuk domain* yang *berbeda.* Dalam: *Jaringan, Konferensi Internasional tentang Sistem dan Konferensi Internasional tentang Komunikasi Bergerak dan Teknologi Pembelajaran, 2006. ICN/ICONS/MCL 2006. Konferensi Internasional tentang*, IEEE, hal. 43-43, doi:10.1109/icniconsmcl.2006.196.

[7] Rajkumar Buyya & Amir Vahid Dastjerdi (2016): *Internet of Things: Prinsip dan Paradigma*, hal. 79-102. Elsevier.

[8] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu & Hucheng Wang (2014): *Sebuah Visi IoT: Aplikasi, Tantangan, dan Peluang dengan Perspektif China*. *IEEE Internet of Things Journal* 1(4), hlm. 349-359, doi:10.1109/jiot.2014.2337336.

[9] Giacomo Ghidini, Vipul Gupta & Sajal K Das (2010): *SNViz: Analysis-oriented Visualiza- tion for the Internet of Things*. Tersedia di `https://pdfs.semanticscholar.org/93cf/3aca8cdfc8f48c4ad595d886079a5b72bd45.pdf`.

[10] Patrick Guillemin, F Berens, O Vermesan, P Friess, M Carugi & G Percivall (2015): *Inter- net of Things: position paper mengenai standarisasi untuk teknologi IoT*. Tersedia di `http://www.internet-of-things-research.eu/pdf/IERC_Position_Paper_IoT_Standardization_Final.pdf`.

[11] Yuna Jeong, Hyuntae Joo, Gyeonghwan Hong, Dongkun Shin & Sungkil Lee (2015): *AVIoT: Penulisan interaktif berbasis web dan visualisasi internet of things dalam ruangan. IEEE Transactions on Consumer Electronics* 61(3), hlm. 295-301, doi:10.1109/tce.2015.7298088.

[12] Dongsik Jo & Gerard Jounghyun Kim (2016): *ARIoT: kerangka kerja augmented reality yang dapat diskalakan untuk berinteraksi dengan peralatan Internet of Things* di *mana saja. IEEE Transactions on Consumer Electronics* 62(3), hlm. 334- 340, doi:10.1109/tce.2016.7613201.

[13] Jussi Kiljander, Alfredo D'elia, Francesco Morandi, Pasi Hyttinen, Janne Takalo-Mattila, Arto Ylisaukko- Oja, Juha-Pekka Soininen & Tullio Salmon Cinotti (2014): *Arsitektur interoperabilitas semantik untuk komputasi pervasive dan internet of things*. *IEEE access* 2, pp. 856-873, doi:10.1109/ACCESS.2014.2347992.

[14] Yi Liu, He Wang, Junyu Wang, Kan Qian, Ning Kong, Kaijiang Wang, Yiwei Shi & Lirong Zheng (2014): *Layanan Nama IoT Berorientasi Perusahaan untuk Manajemen Rantai Pasokan Produk Pertanian*. doi:10.1109/iiki.2014.55.

[15] C Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, Rebekah Metz & Booz Allen Hamilton (2006): *Model referensi untuk arsitektur berorientasi layanan 1.0*. Tersedia di `https://www.oasis-open.org/committees/download.php/18486/pr2changes.pdf`.

[16] Adrian McEwen & Hakim Cassimally (2013): *Merancang Internet Internet*, edisi pertama. John Wiley and Sons, West Sussex, PO19 8SQ, Inggris.

[17] Xuan Thang Nguyen, Huu Tam Tran, Harun Baraki & Kurt Geihs (2015): *FRASAD: Kerangka kerja untuk Pengembangan Aplikasi IoT berbasis model*. Dalam: *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, IEEE, pp. 387-392, doi: 10.1109/wf-iot.2015.7389085.

[18] Per Persson & Ola Angelsmark (2015): *Calvin-Penggabungan Cloud dan IoT*. *Procedia Computer Science* 52, hal. 210-217, doi: 10.1016/j.procs.2015.05.059.

[19] Mark Richards (2016): *Layanan mikro vs arsitektur berorientasi layanan*. O'Reilly Media.

[20] Luciana Moreira Sá De Souza, Patrik Spiess, Dominique Guinard, Moritz Köhler, Stamatis Karnouskos & Domnic Savio (2008): *SOCRADES: Infrastruktur Integrasi Lantai Toko Berbasis Layanan Web*. *Catatan Kuliah Internet of Things dalam Ilmu Komputer*, hal. 50-67, doi:10.1007/978-3-540-78731-0_4.

[21] John A. Stankovic (2014): *Arah Penelitian untuk Internet of Things*. *Jurnal Internet of Things IEEE* 1(1), hal. 3-9, doi:10.1109/jiot.2014.2312291.

[22] Nguyen Xuan Thang, Michael Zapf & Kurt Geihs (2011): *Pengembangan berbasis model untuk aplikasi jaringan sensor* yang *berpusat pada data.* Dalam: *Prosiding Konferensi Internasional ke-9 tentang Kemajuan dalam Komputasi Mobile dan Multimedia*, ACM, hal. 194-197, doi: 10.1145/2095697.2095733.

[23] Charalampos Tsirmpas, Athanasios Anastasiou, Panagiotis Bountris & Dimitris Koutsouris (2015): *Metode Baru untuk Pembuatan Profil dalam Lingkungan Internet of Things: Sebuah Aplikasi dalam Kehidupan Berbantuan Lingkungan*. *IEEE Internet of Things Journal* 2(6), hlm. 471-478, doi:10.1109/jiot.2015.2428307.

[24] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, Markus Eisenhauer dkk. (2011): *Peta jalan penelitian strategis Internet of things*, hal. 9-52. 1, Rivers Publishers Aalborg.

[25] Guangyi Xiao, Jingzhi Guo, Li Da Xu & Zhiguo Gong (2014): *Interoperabilitas Pengguna Dengan Perangkat IoT Heterogen Melalui Transformasi*. *IEEE Transactions on Industrial Informatics* 10(2), hal. 1486-1496, doi:10.1109/tii.2014.2306772.

[26] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista & Michele Zorzi (2014): *Internet of Things untuk Kota Cerdas*. *IEEE Internet of Things Journal* 1(1), hal. 22-32, doi:10.1109/jiot.2014.2306328.