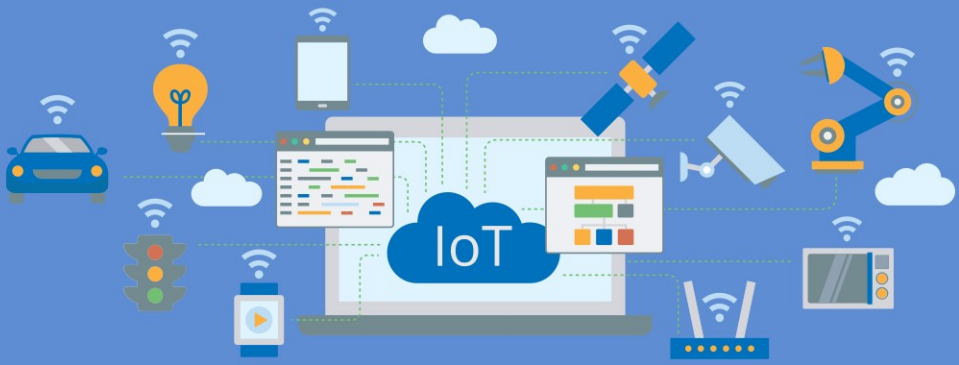
Dessyanto Boedi P

Protokol Jaringan dalam Internet of Things



Penerbit LPPM UPN "Veteran" Yogyakarta

Protokol Jaringan Dalam Internet of Things

Oleh
Dessyanto Boedi Prasetyo

Penerbit LPPM UPN "Veteran" Yogyakarta

Halaman ini sengaja dikosongkan

Kata Pengantar

Puji syukur kehadirat Allah SWT yang senantiasa memberikan rahmat, hidayah serta ridhonya sehingga penulis dapat menyelesaikan buku yang berjudul "Protokol Jaringan dalam Internet of Things". Salawat serta salam diberikan kepada jujungan Nabi Muhammad SAW yang telah membawa risalah yang benar sebagai pedoman menuju jalan yang lurus yang di ridhoi Allah SWT. Buku ini merupakan salah satu instrument output penelitian. Buku ini berisi tentang pengetahuan protokol jaringan komputer dalam Internet of Things. Terlepas dari semua itu, penulis menyadari sepenuhnya bahwa masih ada kekurangan baik dari segi susunan kalimat maupun tata bahasanya. Oleh karena itu dengan tangan terbuka penulis menerima segala saran dan kritik dari pembaca agar penulis dapat memperbaiki makalah ilmiah ini. Akhir kata penulis berharap semoga buku ini dapat memberikan manfaat maupun inpirasi terhadap pembaca. Semoga Allah SWT selalu meridhoi semua umatnya, Amiin Ya Robbal Alamin.

Yogyakarta, Agustus 2020

Penulis

Daftar Isi

1. Pengenalan Internet of Things

1.1. Apa itu Internet of Things?

Internet of Things adalah teknologi terkini yang menciptakan jaringan global mesin dan perangkat yang mampu berkomunikasi dan bertukar data satu sama lain melalui Internet. Ada perbedaan antara Internet of Things dan Internet. Internet of Things dapat membuat informasi tentang objek yang terhubung, menganalisisnya, dan membuat keputusan; dengan kata lain, orang dapat mengatakan bahwa Internet of Things lebih pintar daripada Internet[1]. Kamera keamanan, sensor, kendaraan, gedung, dan perangkat lunak adalah contoh benda yang dapat saling bertukar data.

1.2. Tantangan Internet of Things

Seiring bertambahnya jumlah aplikasi (perangkat) waktu nyata yang membutuhkan koneksi cerdas satu sama lain, tantangan Internet of Things juga akan meningkat. Tantangan tersebut adalah sebagai berikut.

1.3. Konektivitas pintar

Sensor dan perangkat yang terhubung dan dikomunikasikan bersama melalui infrastruktur Internet of Things mungkin perlu memperbarui tren atau fiturnya agar sesuai dengan perubahan lingkungan sekitarnya. Internet of Things adalah infrastruktur cerdas yang dapat memproses data yang dikumpulkan dan membuat keputusan yang diperlukan untuk meningkatkan dirinya sendiri dan untuk mengubah tren atau fitur perangkat yang terhubung untuk mengakomodasi

perubahan lingkungan sekitarnya. Internet of Things adalah teknologi pintar yang membantu semua perangkat yang terhubung untuk memperbarui diri mereka sendiri sesuai dengan perubahan di lingkungan sekitarnya dan untuk dapat diadopsi dan bekerja di lingkungan aneh lainnya dengan akurasi tinggi [2]. Hasilnya, sistem yang terhubung dengan cerdas dapat diproduksi jika infrastruktur cerdas dirancang dengan baik untuk menangani data yang dikumpulkan dari perangkat dengan benar, untuk membuat keputusan yang diperlukan.

1.4. Menjaga privasi dan keamanan yang tinggi dari semua perangkat yang terhubung

Gagasan utama menggunakan Internet of Things adalah memiliki sistem yang cerdas dan menghubungkan miliaran perangkat di seluruh dunia. Perangkat yang diharapkan untuk dihubungkan bersama diharapkan menjadi 50 miliar melalui perangkat Internet of Things pada tahun 2020 [3]. Data menunjukkan pertumbuhan populasi dunia dan perangkat yang terhubung pada tahun 2020. Menghubungkan sejumlah besar perangkat memerlukan tingkat keamanan yang tinggi untuk mencegah penipuan dan memungkinkan perlindungan data tingkat tinggi. Akibatnya, mencapai tingkat keamanan yang tinggi merupakan tantangan besar untuk mendapatkan kepercayaan yang dibutuhkan baik dari industri maupun individu untuk berbagi data mereka menggunakan Internet of Things.

1.5. Mengolah data besar

Tantangan paling penting dalam menggunakan Internet of Things adalah pertumbuhan luar biasa dari data yang dikirimkan antar perangkat yang terhubung. Tiga sumber data dasar adalah (1) database yang digunakan dalam proses bisnis; (2) aktivitas manusia sehari-hari seperti email, Facebook, dan weblog; dan (3) koneksi perangkat fisik seperti kamera dan mikrofon. Perlu disebutkan bahwa 90% penuh dari semua data di dunia telah dihasilkan selama 2 tahun terakhir [4]. Ini membuatnya lebih menantang bagi perancang infrastruktur Internet of Things untuk menangani pertumbuhan data yang dihasilkan seperti itu.

1.6. Mengurangi latensi data keseluruhan di antara interaksi mesin-ke-mesin

Saat menghubungkan banyak perangkat melalui infrastruktur Internet of Things, data bersama di antara mereka juga akan sangat meningkat. Ini akan menyebabkan penundaan atau latensi pengiriman data di antara perangkat yang terhubung. Ini membuka tantangan baru yang harus dihadapi oleh Internet of Things untuk mengurangi latensi tersebut untuk memastikan infrastruktur Internet of Things yang kuat akan diperoleh [5].

1.7. Mengurangi bandwidth dan konsumsi daya

Baik bandwidth dan konsumsi daya dari sejumlah besar perangkat yang terhubung, berkomunikasi, dan berbagi data satu sama lain melalui Internet of Things meningkat pesat. Inilah sebabnya mengapa saat merancang infrastruktur

Internet of Things, tantangan bandwidth dan konsumsi daya harus dipertimbangkan. Tren utama saat ini adalah memperkecil ukuran perangkat yang terhubung, dan akibatnya konsumsi daya akan berkurang. Kecepatan data yang ditransmisikan masih menjadi masalah yang harus dipecahkan karena besarnya data bersama di antara perangkat.

1.8. Kompleksitas

Berbagi data dan menghubungkan perangkat bersama melalui Internet of Things dapat diimplementasikan melalui beberapa tingkatan dan lapisan perangkat lunak / perangkat keras dan beberapa protokol standar. Dengan peningkatan luar biasa dalam data bersama dan perangkat yang terhubung, perangkat lunak / perangkat keras yang digunakan dan protokol standar akan menjadi lebih rumit. Akibatnya, terdapat tantangan untuk mengurangi kompleksitas teknologi Internet of Things seiring dengan meningkatnya jumlah perangkat yang terhubung.

1.9. Aplikasi Internet of Things

Internet of Things diakui sebagai salah satu bidang terpenting dari teknologi masa depan dan mendapatkan pengakuan luas dalam berbagai aplikasi dan bidang yang terkait dengan kota pintar, militer, pendidikan, rumah sakit, sistem keamanan dalam negeri, transportasi dan mobil yang terhubung secara otonom, pertanian, sistem belanja cerdas, dan teknologi modern lainnya. Rumah pintar adalah salah satu aplikasi utama yang menggunakan infrastruktur Internet of Things untuk menghubungkan beberapa sensor. Sensor dapat

merasakan dan mengumpulkan informasi di sekitar yang digunakan untuk mengontrol sepenuhnya berbagai sistem rumah seperti pencahayaan dan keamanan.

Ada banyak aplikasi lain yang menggunakan infrastruktur Internet of Things seperti smart bridge dan smart tunnel. Sensor suhu dan getaran, serta kamera pengintai video, dapat dipasang di jembatan untuk mendeteksi aktivitas tidak normal dan mengirim peringatan melalui SMS [6]. Analisis pemrosesan video juga dapat dilakukan untuk mengontrol kepadatan lalu lintas di jembatan. Terowongan pintar dapat menggunakan beberapa sensor untuk memantau kelembapan, perpindahan, dan suhu untuk meminta perawatan yang tepat jika masalah terdeteksi. Semua aplikasi ini menggunakan sensor untuk mendeteksi dan mengumpulkan data yang digunakan untuk memberikan keputusan yang tepat yang menjaga keamanan tingkat tinggi dari instalasi.

2. Protokol HTTP

2.1. HTTP dan WWW

Internet dapat menelusuri akarnya ke proyek-proyek penelitian yang dimulai pada 1960-an oleh Departemen Pertahanan Amerika Serikat. Seorang fisikawan Inggris yang bekerja di Swiss, bagaimanapun, telah mempengaruhi Internet saat ini lebih dari orang lain. Pada tanggal 1 Maret 989, Tim Berners-Lee pertama kali menguraikan keuntungan dari sistem informasi terkait berbasis hypertext. Dan pada akhir tahun 1990, Berners-Lee, bersama dengan Robert Cailliau, membuat browser dan server Web pertama. Browser tersebut membutuhkan protokol untuk mengatur komunikasi mereka; untuk itu Berners-Lee dan Cailliau merancang http versi pertama.

Sejak saat itu, lalu lintas Web telah mendominasi Internet. Pada tahun 1998, http menyumbang lebih dari 75 persen dari lalu lintas di tulang punggung Internet yang mengecilkan protokol lain seperti email, transfer file, dan login jarak jauh. Saat ini, setidaknya dalam bahasa sehari-hari, World Wide Web adalah Internet. Dan Web terus berkembang. Pada musim gugur tahun 2000, karena buku ini hampir selesai, Proyek Censorware melaporkan bahwa Web memiliki kira-kira:

- 2.700.000.00 halaman
- 50.700.000.000.000 byte text
- 600.000.000 gambar
- 10.100.000.000.000 btye gambar

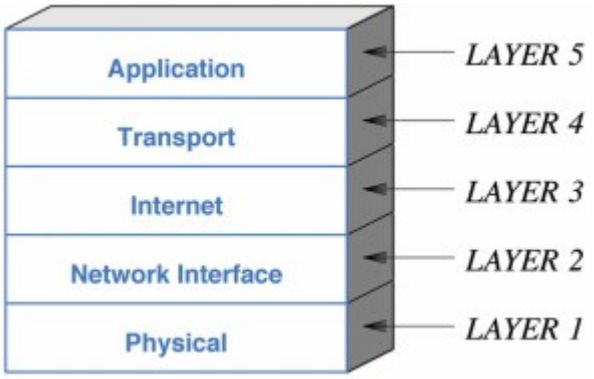
Hypertext Transfer Protocol telah berkembang seiring dengan Web. Spesifikasi asli untuk http cocok dengan nyaman di satu halaman dan, dengan panjang 656 kata, dapat dibaca dan dipahami hanya dalam beberapa menit. Sebaliknya, spesifikasi untuk http versi 1.1 mencakup beberapa dokumen. Dokumen inti saja berisi hampir 60.000 kata dalam 176 halaman spesifikasi http inti, bersama dengan dokumen lain yang membentuk standar http, menentukan aturan yang digunakan browser Web, server Web, proxy, dan sistem Web lainnya untuk membangun dan memelihara komunikasi satu sama lain [7]. Standar http tidak menentukan informasi apa yang dipertukarkan sistem setelah mereka menjalin komunikasi. Memang, salah satu kekuatan terbesar http adalah kemampuannya untuk mengakomodasi hampir semua jenis pertukaran informasi. Halaman web, misalnya, seringkali dibuat sesuai dengan aturan Hypertext Markup Language, atau html (juga ditemukan oleh Berners-Lee). Tetapi http juga mahir dalam mentransfer instruksi pencetakan jarak jauh, file program, dan objek multimedia. Dengan adanya browser Web di mana-mana, Internet yang mudah menyebar, dan kekuatan serta fleksibilitas http, protokol yang dikembangkan Berners-Lee dan Cailliau pada akhirnya dapat menjadi fondasi untuk semua komputasi berbasis jaringan.

2.2. Lapisan Protokol

Untuk memahami http, ada baiknya mengetahui sedikit tentang arsitektur Internet. Kita dapat melihat arsitektur Internet dari dua perspektif. Dari satu pandangan, Internet adalah sekumpulan jaringan yang terhubung secara longgar dari semua ukuran dan jenis yang bekerja sama untuk bertukar

informasi. Namun, alih-alih mempertimbangkan sistem fisik, kami akan fokus pada perangkat lunak yang mengontrol sistem tersebut. Dari perspektif itu, Internet adalah kumpulan protokol komunikasi yang berbeda; protokol ini bekerja sama untuk menyediakan layanan. Menyediakan layanan melalui Internet sebenarnya merupakan pekerjaan yang sangat kompleks. Untuk membuat tantangan lebih dapat dikelola, perancang Internet membagi pekerjaan menjadi komponen yang berbeda dan menetapkan komponen tersebut ke beberapa protokol komunikasi yang berbeda. Para desainer selanjutnya mengatur protokol tersebut menjadi beberapa lapisan.

Gambar 2.2.1 menunjukkan empat lapisan protokol dalam sistem komputer. Protokol lapisan terendah mengontrol teknologi jaringan tertentu, baik itu jaringan Ethernet, modem dial-up, tautan serat optik, atau teknologi lainnya. Salah satu kekuatan terbesar Internet adalah kemampuannya untuk beradaptasi dengan semua jenis teknologi jaringan. Mengisolasi protokol untuk teknologi itu di dalam lapisannya sendiri adalah salah satu alasan untuk fleksibilitas ini; mendukung teknologi jaringan baru hanyalah masalah penerapan protokol lapisan rendah yang sesuai [2].



Gambar 2.2.1: TCP/IP Layer

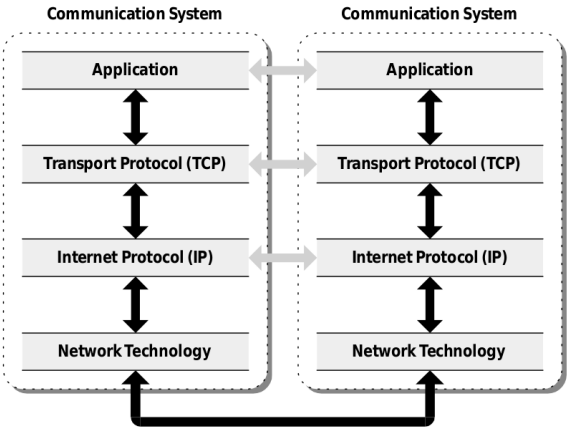
Lapisan protokol tepat di atas teknologi jaringan adalah Internet Protocol, atau IP. Dan meskipun ip mungkin tidak setenar protokol lain di Internet, IP dapat dengan mudah membenarkan namanya sebagai Protokol Internet. Tidak setiap sistem di Internet menggunakan teknologi jaringan yang sama, dan sistem yang berbeda bergantung pada transportasi dan protokol aplikasi yang berbeda. Setiap sistem di Internet, bagaimanapun, menggunakan IP. Tanggung jawab utama Protokol Internet adalah mengambil paket informasi individu dan meneruskannya ke tujuannya. Sebagian besar komunikasi antar sistem memerlukan pertukaran banyak paket, dan IP bertanggung jawab untuk setiap paket [8].

Protokol selanjutnya adalah protokol transport. Internet secara umum menggunakan tiga protokol transport yang berbeda, tetapi komunikasi Web secara khusus menggunakan satu: Transmission Control Protocol (tcp). Sementara ip memiliki tanggung jawab untuk memindahkan paket dari satu sistem ke

sistem lainnya, tcp membuat transfer informasi tersebut dapat diandalkan. Ini memastikan bahwa paket tiba dalam urutan yang benar, tidak ada yang hilang saat transit, dan tidak ada kesalahan yang muncul. Lapisan protokol terakhir adalah aplikasi. Protokol ini sebenarnya melakukan sesuatu yang berarti dengan informasi yang dipertukarkan, termasuk mengatur pertukaran menjadi percakapan. Protokol aplikasi yang paling menarik bagi kami di sini, tentu saja, http, tetapi ada banyak protokol aplikasi lainnya di Internet. Ada protokol aplikasi untuk bertukar surat elektronik, untuk mengatur panggilan telepon, untuk mengotorisasi sesi dialup, dan sebagainya. Tentu saja, seperti yang kami catat sebelumnya, lalu lintas http adalah lalu lintas massal di Internet saat ini. Organisasi protokol internal dari satu sistem bukanlah hal yang penting untuk komunikasi. Lagi pula, dibutuhkan lebih dari satu sistem untuk memiliki komunikasi yang bermakna. Gambar 2.2.2 memperluas gambar sebelumnya dengan memasukkan sistem kedua ke dalam diagram. Sekarang kita dapat mulai melihat bagaimana komunikasi sebenarnya terjadi. Gambar tersebut menunjukkan panah hitam di antara berbagai lapisan protokol dalam suatu sistem. Panah tersebut mewakili interaksi langsung. Protokol aplikasi dalam satu sistem berinteraksi langsung dengan protokol transport.

Protokol itu, pada gilirannya, berinteraksi langsung dengan ip, dan ip berinteraksi dengan protokol yang mengendalikan teknologi jaringan [8]. Sistem yang berbeda dapat berinteraksi secara langsung satu sama lain hanya melalui teknologi jaringan. Gambar 2.2.2 juga menunjukkan bentuk interaksi lain. Panah abu-abu mewakili interaksi logis,

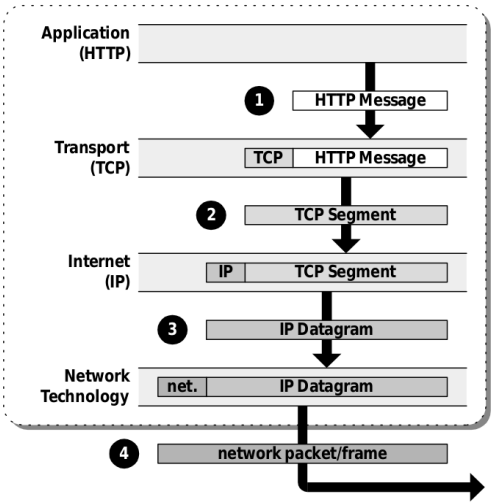
dan, seperti yang ditunjukkan gambar, setiap lapisan protokol secara logis berinteraksi dengan rekannya di sistem yang jauh. Jadi meskipun aplikasi dalam satu sistem hanya berinteraksi langsung dengan tcp, hasil interaksi tersebut adalah komunikasi logis dengan aplikasi di sistem lain. Dalam kasus http, implementasi http dalam satu sistem (misalnya, browser Web) secara efektif berkomunikasi dengan implementasi http di sistem lain (server Web, mungkin). Untuk melihat proses ini secara lebih rinci, mari kita lihat bagaimana pesan http berpindah dari browser Web Anda ke server Web di Internet. Gambar 2.2.3 menunjukkan empat langkah pertama dalam proses tersebut. Pertama, proses http membangun pesan yang ingin dikirim; kemudian, pada langkah 1, ini menyerahkan pesan itu ke proses tcp. Proses tcp menambahkan informasi



Gambar 2.2.2: Komunikasi dua entitas menggunakan TCP/IP

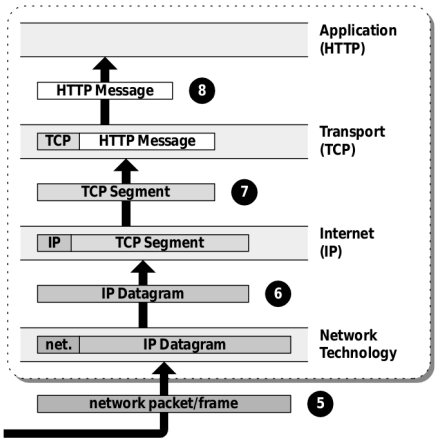
sepesifik tcp ke pesan, membuat segmen tcp. Penambahan ini

sangat mirip dengan amplop untuk surat biasa. Surat itu sendiri memuat informasi yang sebenarnya, tetapi kami melampirkannya dalam amplop untuk kepentingan layanan pos. Layanan pos menggunakan informasi alamat pada amplop untuk mengirim surat, tanpa peduli tentang isi surat. Pada langkah 2, proses tcp meneruskan segmen ke proses ip. Proses ip dibangun di atas segmen ini dengan menambahkan lebih banyak informasi, yang pada dasarnya menambahkan amplop lain. Hasilnya adalah datagram ip yang, pada langkah 3, mencapai implementasi protokol yang mengendalikan teknologi jaringan sistem. Hanya pada langkah 4, setelah lebih banyak informasi ditambahkan ke pesan asli, apakah informasi tersebut benar-benar meninggalkan sistem komputer. Daunnya dalam bentuk paket atau bingkai.



Gambar 2.2.3: Aliran data dalam aplikasi http di satu sisi entitas

Gambar 2.2.4 melengkapi contoh dengan menunjukkan apa yang terjadi ketika paket mencapai server Web. Ini mungkin telah melakukan perjalanan melalui banyak sistem lain dan di berbagai teknologi jaringan untuk sampai ke sana, tetapi langkah-langkah perantara tersebut tidak penting untuk browser atau server. Proses yang ditunjukkan Gambar 2.2.4 sebenarnya hanyalah kebalikan dari empat langkah pertama. Setiap lapisan protokol menerima pesan tersebut, memprosesnya sesuai kebutuhan, dan meneruskan informasi yang diekstrak ke protokol tertinggi berikutnya. Akhirnya, di langkah 8, pesan http asli tiba di aplikasi server Web. Dalam buku ini, kita terutama akan memusatkan perhatian pada protokol lapisan aplikasi — terutama http. Karena http bergantung pada tcp untuk membawa pesannya, bagaimanapun, kami akan sesekali mendiskusikan interaksi antara http dan tcp; interaksi tersebut dapat memberikan pengaruh yang signifikan pada kinerja http, dan telah menyebabkan pengembangan banyak fitur penting di http.

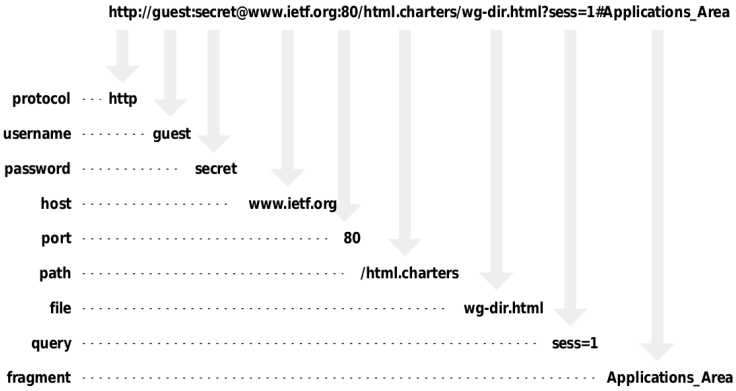


Gambar 2.2.4: Aliran data dalam aplikasi http di satu sisi entitas yang lain

2.3. Uniform Resource Identifiers

Kemungkinan besar, Anda sudah terbiasa dengan Uniform Resource Locators, atau url. Mereka adalah alamat yang kami gunakan untuk menamai situs Web; http://www.waterscreek.com adalah contohnya. Anda mungkin akan sedikit terkejut, meskipun, ketika Anda melihat bahwa http terus-menerus mengacu pada Uniform Resource Identifiers, atau uris. Sebenarnya, tidak ada banyak perbedaan antara kedua konsep tersebut. Secara teknis, url hanyalah salah satu jenis uri. Bagaimanapun, salah satu cara untuk mengidentifikasi suatu objek adalah dengan mendeskripsikan lokasinya. Namun, dalam praktiknya, kedua istilah itu setara. Buku ini biasanya menggunakan uri karena itulah istilah dalam spesifikasi http. Jika, setiap kali Anda melihat "uri", Anda

menerjemahkannya secara mental sebagai "url", Anda tidak akan menderita efek buruk apa pun. Bagaimanapun, uri sebenarnya dapat berisi cukup banyak informasi, dan pemahaman yang menyeluruh tentang struktur uri akan membantu dalam mengapresiasi beberapa aspek http. Gambar 2.3.1 menunjukkan sampel uri dengan hampir semua elemen yang memungkinkan [9]. (Memasukkan uri ini di browser Web benar-benar berfungsi saat buku ini ditulis; tentu saja, tidak ada jaminan akan tetap demikian setelah diterbitkan.)



Gambar 2.3.1: URI

2.4. Clients and Servers

Seperti banyak protokol komunikasi, http membuat perbedaan utama antara dua pihak yang berkomunikasi. Dalam pertukaran http apa pun, satu sistem mengasumsikan peran klien sementara yang lain adalah server. Perbedaan ini sangat penting, karena http mengharuskan klien dan server

untuk mengikuti aturan dan prosedur yang sangat berbeda. Dalam sesi Web sederhana, pc penjelajahan Web adalah klien http, sedangkan sistem hosting situs Web bertindak sebagai server http. Meskipun kedua sistem ini sama-sama berkomunikasi menggunakan http, mereka jelas memiliki tanggung jawab yang sangat berbeda dalam komunikasi tersebut. Seperti yang akan kita lihat di bagian ini, klien, yang selalu memulai komunikasi http, mengontrol beberapa karakteristik penting dari sesi tersebut, termasuk koneksi tcp yang mendasarinya, persistensi, dan pipelining [10].

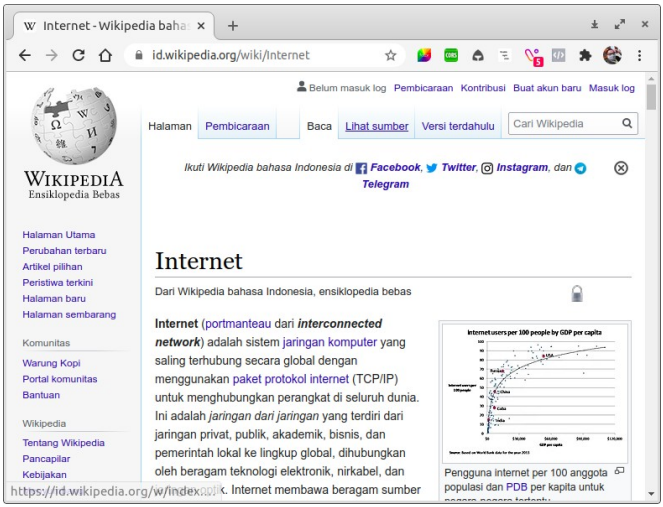
2.5. Operasi Pengguna

Protokol http mendefinisikan empat operasi dasar—GET, POST, PUT, dan DELETE. Kami menganggap ini sebagai operasi pengguna karena, setidaknya dalam konteks penjelajahan Web, masing-masing merupakan hasil langsung dari tindakan pengguna. Seperti yang akan kita lihat di bagian selanjutnya, tindakan pengguna dapat menyebabkan pertukaran http lain, dan pengguna akhir tidak perlu memulai salah satunya. Namun, empat operasi pada bagian ini tetap merupakan operasi http paling dasar [11].

2.5.1 Pengambilan web – GET

Operasi http paling sederhana dari semuanya adalah GET. Ini adalah cara klien mengambil objek dari server. Di Web, browser meminta halaman dari server Web dengan GET. Misalnya, mengklik link di tengah Gambar 2.5.1 akan memaksa browser untuk mengeluarkan permintaan GET ke server meminta halaman Web baru untuk ditampilkan. Seperti yang

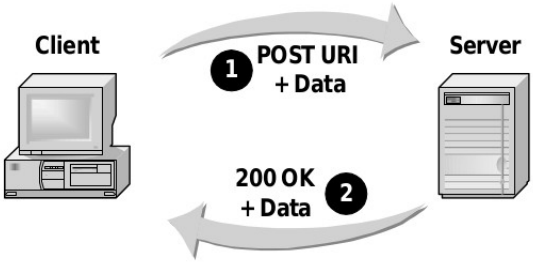
ditunjukkan Gambar 2.5.1, GET adalah pertukaran dua pesan sederhana. Klien memulainya dengan mengirimkan pesan GET ke server. Pesan tersebut mengidentifikasi objek yang diminta klien dengan Uniform Resource Identifier (uri). Jika server dapat mengembalikan objek yang diminta, ia melakukannya sebagai tanggapannya. Seperti yang ditunjukkan gambar tersebut, server menunjukkan keberhasilan dengan status yang sesuai; 200 OK adalah kode status untuk respons yang berhasil. Bersama dengan kode status, server menyertakan objek itu sendiri dalam responsnya. Jika server tidak dapat mengembalikan objek yang diminta (atau memilih untuk tidak), maka itu dapat mengembalikan sejumlah kode status lainnya.



Gambar 2.5.1: Contoh penggunaan GET

2.5.2 Web Forms – POST

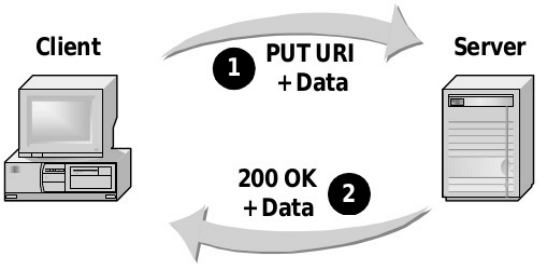
Meskipun penjelajahan Web sebagian besar dimulai sebagai cara untuk melihat halaman informasi, namun segera berkembang menjadi interaksi dua arah. Sementara GET memungkinkan server mengirim informasi ke klien, operasi POST menyediakan cara bagi klien untuk mengirim informasi ke server. Browser web paling umum menggunakan operasi POST untuk mengirim formulir ke server Web [12]. Gambar 2.9 menunjukkan contoh bentuk seperti itu. Ini adalah halaman Web yang memungkinkan pengguna untuk mencari standar Internet. Saat pengguna mengklik tombol "Cari Database", browser mengirimkan permintaan POST ke server; permintaan tersebut mencakup informasi yang diberikan pengguna dalam formulir. Seperti yang ditunjukkan Gambar 2.5.2, operasi POST hampir sesederhana GET. Klien mengirim pesan POST dan menyertakan informasi yang ingin dikirim ke server. Seperti pesan GET, bagian dari pesan POST adalah Uniform Resource Identifier (uri). Dalam hal ini, uri mengidentifikasi objek di server yang dapat memproses informasi yang disertakan. Di server Web, uri ini sering kali berupa program atau skrip. Seperti halnya operasi GET, server dapat mengembalikan informasi itu sendiri sebagai bagian dari respons. Untuk penjelajahan Web, informasi ini biasanya berupa halaman Web baru untuk ditampilkan, sering kali merupakan halaman yang mengakui masukan pengguna; dalam kasus formulir pencarian, halaman Web baru sering kali menampilkan hasil pencarian.



Gambar 2.5.2: Respon server pada operasi GET

2.5.3 File Upload – PUT

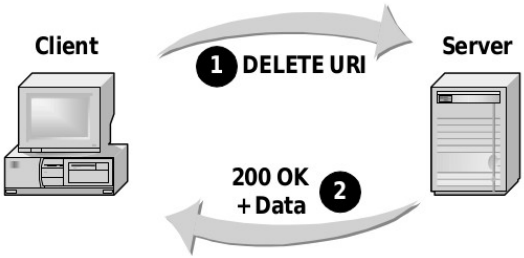
Operasi PUT juga menyediakan cara bagi klien untuk mengirimkan informasi ke server. Ini sangat berbeda dari operasi POST, meskipun, seperti yang ditunjukkan Gambar 2.5.3, keduanya terlihat sangat mirip. Seperti halnya POST, klien mengirimkan metode, uri, dan data. Server mengembalikan kode status dan, secara opsional, data. Perbedaan antara POST dan PUT terletak pada cara server menafsirkan Uniform Resource Identifier. Dengan POST, uri mengidentifikasi objek di server yang dapat memproses data yang disertakan. Dengan PUT, di sisi lain, uri mengidentifikasi objek di mana server harus menempatkan data. Sementara uri POST umumnya menunjukkan program atau skrip, PUT uri biasanya merupakan jalur dan nama untuk file. Gambar 2.5.3 menunjukkan contoh pengoperasian PUT. Pada halaman ini pengguna telah mengidentifikasi file lokal. Dengan mengklik tombol Unggah, pengguna meminta browser untuk mengirim permintaan PUT ke server.



Gambar 2.5.3: Operasi PUT

2.5.4 File Deletion – DELETE

Dengan operasi GET dan PUT, http menjadi protokol yang dapat diservis untuk transfer file sederhana. Operasi DELETE menyelesaikan fungsi ini dengan memberi klien cara untuk menghapus objek dari server. Pertukaran pesan tidak mengandung kejutan. Seperti yang ditunjukkan Gambar 2.5.4, klien mengirim pesan DELETE bersama dengan uri dari objek yang harus dihapus server. Server merespons dengan kode status dan, secara opsional, lebih banyak data untuk klien.



Gambar 2.5.4: Operasi DELETE

3. Protokol UPNP

3.1. Apa itu UPnP ?

Teknologi UPnPTM mendefinisikan arsitektur untuk konektivitas jaringan peer-to-peer yang tersebar luas dari peralatan cerdas, perangkat nirkabel, dan PC dari semua faktor bentuk. Ini dirancang untuk menghadirkan konektivitas berbasis standar yang mudah digunakan, fleksibel, ke jaringan ad-hoc atau tidak terkelola baik di rumah, dalam bisnis kecil, ruang publik, atau terhubung ke Internet. Teknologi UPnP menyediakan arsitektur jaringan terbuka yang terdistribusi yang memanfaatkan teknologi TCP / IP dan Web untuk memungkinkan jaringan kedekatan yang mulus selain untuk mengontrol dan mentransfer data antar perangkat jaringan [13].

Arsitektur Perangkat UPnP (UDA) lebih dari sekadar perpanjangan sederhana dari model periferal plug and play. Ini dirancang untuk mendukung konfigurasi nol, jaringan "tak terlihat", dan penemuan otomatis untuk berbagai kategori perangkat dari berbagai vendor. Ini berarti perangkat dapat secara dinamis bergabung dengan jaringan, mendapatkan alamat IP, menyampaikan kemampuannya, dan mempelajari keberadaan serta kapabilitas perangkat lain. Terakhir, perangkat dapat meninggalkan jaringan dengan lancar dan otomatis tanpa meninggalkan keadaan yang tidak diinginkan.

Teknologi yang dimanfaatkan dalam arsitektur UPnP mencakup protokol Internet seperti IP, TCP, UDP, HTTP, dan

XML. Seperti Internet, kontrak didasarkan pada protokol kawat yang bersifat deklaratif, diekspresikan dalam XML, dan dikomunikasikan melalui HTTP. Menggunakan protokol Internet adalah pilihan yang kuat untuk UDA karena kemampuannya yang telah terbukti untuk menjangkau media fisik yang berbeda, untuk memungkinkan interoperasi multi-vendor dunia nyata, dan untuk mencapai sinergi dengan Internet dan banyak intranet rumah dan kantor. Arsitektur UPnP telah dirancang secara eksplisit untuk mengakomodasi lingkungan ini. Lebih lanjut, melalui penghubung, UDA mengakomodasi media yang menjalankan protokol non-IP ketika biaya, teknologi, atau warisan mencegah media atau perangkat yang terpasang padanya untuk menjalankan IP [13].

Apa yang "universal" tentang teknologi UPnP? Tidak ada driver perangkat; protokol umum digunakan sebagai gantinya. Jaringan UPnP adalah media independen. Perangkat UPnP dapat diimplementasikan menggunakan bahasa pemrograman apa pun, dan pada sistem operasi apa pun. Arsitektur UPnP tidak menentukan atau membatasi desain API untuk aplikasi; Vendor OS dapat membuat API yang sesuai dengan kebutuhan pelanggan mereka.

1.1. Forum UPnP

Forum UPnP adalah inisiatif industri yang dirancang untuk memungkinkan konektivitas yang mudah dan kuat di antara perangkat dan PC yang berdiri sendiri dari banyak vendor yang berbeda. Forum UPnP berupaya mengembangkan

standar untuk menjelaskan protokol perangkat dan skema perangkat berbasis XML untuk tujuan memungkinkan interoperabilitas perangkat-ke-perangkat dalam lingkungan jaringan yang dapat diskalakan [11].

UPnP Implementers Corporation (UIC) terdiri dari perusahaan anggota Forum UPnP di banyak industri yang mempromosikan penerapan standar interkonektivitas perangkat teknis yang seragam serta pengujian dan sertifikasi perangkat ini. UIC mengembangkan dan mengelola proses pengujian dan sertifikasi, mengelola program logo UPnP, dan memberikan informasi kepada anggota UIC dan pihak lain yang berkepentingan terkait sertifikasi perangkat UPnP. Proses sertifikasi perangkat UPnP terbuka untuk semua vendor yang merupakan anggota Forum UPnP dan UIC, telah membayar iuran UIC, dan memiliki perangkat yang mendukung fungsionalitas UPnP. Untuk informasi lebih lanjut, lihat http://www.upnp-ic.org.

Forum UPnP telah membentuk komite kerja di bidang keahlian domain tertentu. Komite kerja ini ditugaskan untuk membuat standar perangkat yang diusulkan, membangun implementasi sampel, dan membangun rangkaian pengujian yang sesuai. Dokumen ini menunjukkan keputusan teknis khusus yang menjadi lingkup komite kerja Forum UPnP.

Vendor UPnP dapat membangun perangkat yang sesuai dengan kepercayaan interoperabilitas dan manfaat dari kekayaan intelektual bersama dan program logo. Terpisah dari program logo, vendor juga dapat membuat perangkat yang

mematuhi Arsitektur Perangkat UPnP yang ditentukan di sini tanpa prosedur standar formal. Jika vendor membuat perangkat non-standar, mereka menentukan keputusan teknis yang akan ditentukan oleh panitia kerja Forum UPnP.

Arsitektur Perangkat UPnP (sebelumnya dikenal sebagai Kerangka DCP) yang terkandung di sini mendefinisikan protokol untuk komunikasi antara pengontrol, atau titik kontrol, dan perangkat. Untuk penemuan, deskripsi, kontrol, eventing, dan presentasi, Arsitektur Perangkat UPnP menggunakan tumpukan protokol berikut (warna yang ditunjukkan dan gaya tipe digunakan di seluruh dokumen ini untuk menunjukkan di mana setiap elemen protokol didefinisikan).

Pada lapisan tertinggi, pesan secara logis hanya berisi informasi khusus vendor UPnP tentang perangkat mereka. Bergerak ke bawah tumpukan, konten vendor dilengkapi dengan informasi yang ditentukan oleh komite kerja Forum UPnP. Pesan dari lapisan di atas dihosting dalam protokol khusus UPnP seperti Simple Service Discovery Protocol (SSDP), General Event Notification Architecture (GENA) dan protokol peristiwa multicast yang ditentukan dalam dokumen ini, dan lainnya yang direferensikan. SSDP dikirimkan melalui multicast atau unicast UDP. Acara multicast dikirim melalui multicast UDP. GENA dikirim melalui HTTP. Pada akhirnya, semua pesan di atas dikirim melalui IP. Bagian selanjutnya dari dokumen ini menjelaskan konten dan format untuk masing-masing lapisan protokol ini secara rinci. Sebagai referensi, warna dalam [tanda kurung siku] di atas

menunjukkan protokol mana yang mendefinisikan komponen pesan tertentu di seluruh dokumen ini.

Dua klasifikasi umum perangkat ditentukan oleh arsitektur UPnP: perangkat yang dikontrol (atau hanya "perangkat"), dan kontrol poin. Perangkat yang dikontrol berfungsi dalam peran server, menanggapi permintaan dari titik kontrol. Baik titik kontrol dan perangkat yang dikendalikan dapat diimplementasikan pada berbagai platform termasuk komputer pribadi dan sistem tertanam. Beberapa perangkat, titik kontrol, atau keduanya mungkin beroperasi pada titik akhir jaringan yang sama secara bersamaan [14].

Landasan untuk jaringan UPnP adalah pengalamatan IP. Dalam lingkungan IPv4, setiap perangkat atau titik kontrol harus memiliki klien Protokol Konfigurasi Host Dinamis (DHCP) dan mencari server DHCP saat perangkat atau titik kontrol pertama kali dihubungkan ke jaringan. Jika server DHCP tersedia, yaitu jaringan dikelola; perangkat atau titik kontrol HARUS menggunakan alamat IP yang ditetapkan untuk itu. Jika server DHCP tidak tersedia, mis., Jaringan tidak dikelola; perangkat atau titik kontrol HARUS menggunakan IP Otomatis untuk mendapatkan alamat. Singkatnya, Auto IP mendefinisikan cara perangkat atau titik kontrol secara cerdas memilih alamat IP dari sekumpulan alamat yang dicadangkan dan dapat berpindah dengan mudah antara jaringan yang dikelola dan tidak dikelola. Jika selama transaksi DHCP, perangkat atau titik kontrol memperoleh nama domain, misalnya melalui server DNS atau melalui penerusan DNS, perangkat atau titik kontrol harus

menggunakan nama itu dalam operasi jaringan selanjutnya; jika tidak, perangkat atau titik kontrol harus menggunakan alamat IP-nya.

Jaringan UPnP tertentu memiliki konfigurasi yang lebih kompleks seperti beberapa jaringan fisik dan / atau beberapa jaringan logis untuk mengakomodasi beberapa skema pengalamatan yang tidak tumpang tindih. Perangkat dan titik kontrol juga dapat memiliki dua atau lebih antarmuka jaringan, dan / atau dua atau lebih alamat IP yang ditetapkan untuk setiap antarmuka. Dalam konfigurasi seperti itu, satu perangkat atau titik kontrol dapat diberikan beberapa alamat IP dari jaringan logis yang berbeda dalam jaringan UPnP yang sama, mengakibatkan perangkat muncul ke titik kontrol beberapa kali dalam jaringan. Perangkat dan titik kontrol yang memiliki beberapa alamat IP pada jaringan UPnP yang sama disebut sebagai multi-homed [15]. Di seluruh dokumen ini, istilah "antarmuka berkemampuan UPnP" digunakan untuk merujuk ke antarmuka yang diberi alamat IP milik jaringan UPnP. Perilaku tambahan khusus untuk perangkat multi-rumah dan titik kontrol akan dicakup dalam bagian yang berlaku di seluruh dokumen. Namun, sebagai prinsip umum, interaksi terkait antara titik kontrol dan perangkat (mis. Permintaan kontrol tindakan dan pesan respons, langganan acara, dan pesan acara) HARUS terjadi menggunakan pasangan antarmuka yang mengaktifkan UPnP keluar dan masuk yang sama. Diberikan alamat IP, Langkah 1 di jaringan UPnP adalah penemuan. Saat perangkat ditambahkan ke jaringan, protokol penemuan UPnP memungkinkan perangkat tersebut mengiklankan layanannya untuk mengontrol titik di

jaringan. Demikian pula, ketika titik kontrol ditambahkan ke jaringan, protokol penemuan UPnP memungkinkan titik kontrol tersebut untuk mencari perangkat yang diminati di jaringan. Pertukaran mendasar dalam kedua kasus tersebut adalah pesan penemuan yang berisi beberapa spesifik penting tentang perangkat atau salah satu layanannya, misalnya, jenisnya, pengenal, dan penunjuk ke informasi yang lebih detail. Bagian Penemuan di bawah ini menjelaskan bagaimana perangkat beriklan, bagaimana titik kontrol mencari, dan berisi rincian tentang format pesan penemuan.

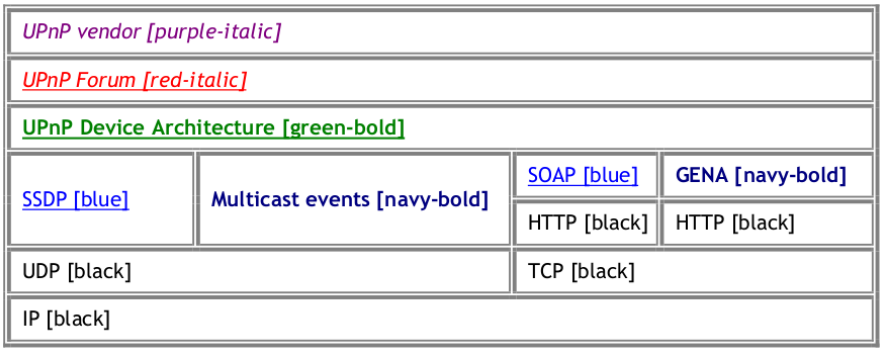
Langkah 2 di jaringan UPnP adalah deskripsi. Setelah titik kontrol menemukan perangkat, titik kontrol tersebut masih sangat sedikit mengetahui tentang perangkat tersebut. Untuk titik kontrol untuk mempelajari lebih lanjut tentang perangkat dan kemampuannya, atau untuk berinteraksi dengan perangkat, titik kontrol harus mengambil deskripsi perangkat dari URL yang disediakan oleh perangkat dalam pesan penemuan. Perangkat mungkin berisi perangkat logis lainnya, serta unit fungsional, atau layanan. Deskripsi UPnP untuk perangkat diekspresikan dalam XML dan mencakup informasi pabrikan khusus vendor seperti nama dan nomor model, nomor seri, nama pabrikan, URL ke situs Web khusus vendor, dll. Deskripsi juga mencakup daftar setiap perangkat atau layanan yang disematkan, serta URL untuk kontrol, acara, dan presentasi. Untuk setiap layanan, deskripsinya menyertakan daftar perintah, atau tindakan, yang ditanggapi oleh layanan, dan parameter, atau argumen untuk setiap tindakan; deskripsi layanan juga menyertakan daftar variabel; Variabel ini memodelkan status layanan pada waktu proses, dan dijelaskan

dalam istilah jenis data, rentang, dan kejadiannya karakteristik. Bagian Deskripsi di bawah ini menjelaskan bagaimana perangkat dideskripsikan dan bagaimana titik kontrol mengambilnya deskripsi.

Langkah 3 dalam jaringan UPnP adalah kontrol. Setelah titik kontrol mengambil deskripsi perangkat, titik kontrol dapat mengirim tindakan ke layanan perangkat. Untuk melakukan ini, titik kontrol mengirimkan pesan kontrol yang sesuai ke URL kontrol untuk layanan (disediakan dalam deskripsi perangkat). Pesan kontrol juga diekspresikan dalam XML menggunakan Simple Object Access Protocol (SOAP) [16]. Seperti panggilan fungsi, dalam menanggapi pesan kontrol, layanan mengembalikan nilai spesifik tindakan apa pun. Efek tindakan, jika ada, dimodelkan oleh perubahan dalam variabel yang menjelaskan status waktu proses layanan. Bagian Kontrol di bawah ini menjelaskan deskripsi tindakan, variabel status, dan format pesan kontrol.

Langkah 4 di jaringan UPnP sedang berlangsung. Deskripsi UPnP untuk layanan menyertakan daftar tindakan yang ditanggapi layanan dan daftar variabel yang memodelkan status layanan pada waktu proses. Layanan menerbitkan pembaruan ketika variabel ini berubah, dan titik kontrol mungkin berlangganan untuk menerima informasi ini. Layanan menerbitkan pembaruan dengan mengirim pesan acara. Pesan peristiwa berisi nama satu atau beberapa variabel status dan nilai variabel tersebut saat ini. Pesan-pesan ini juga diekspresikan dalam XML. Pesan peristiwa awal khusus dikirim ketika titik kontrol pertama kali berlangganan; pesan

peristiwa ini berisi nama dan nilai untuk semua variabel peristiwa dan memungkinkan pelanggan untuk menginisialisasi model status layanannya. Untuk mendukung skenario dengan beberapa titik kontrol, eventing dirancang untuk menjaga semua titik kontrol mendapat informasi yang sama tentang efek dari tindakan apa pun. Oleh karena itu, semua pelanggan dikirimi semua pesan acara, pelanggan menerima pesan acara untuk semua var acara dan / atau melihat status perangkat. Sejauh mana masing-masing hal ini dapat dicapai tergantung pada kemampuan spesifik dari halaman dan perangkat presentasi. Bagian Presentasi di bawah ini menjelaskan protokol untuk mengambil halaman presentasi.



Gambar 3.1.1: Protokol UPnP

3.2. Pengalamatan

Pengalamatan adalah Langkah 0 dari jaringan UPnPTM. Melalui pengalamatan, perangkat dan titik kontrol mendapatkan alamat jaringan. Pengalamatan memungkinkan penemuan (Langkah 1) di mana titik kontrol menemukan perangkat yang menarik, deskripsi (Langkah 2) di mana titik

kontrol belajar tentang kemampuan perangkat, kontrol (Langkah 3) di mana titik kontrol mengirimkan perintah ke perangkat, eventing (Langkah 4) di mana titik kontrol mendengarkan perubahan status di perangkat, dan presentasi (Langkah 5) di mana titik kontrol menampilkan antarmuka pengguna untuk perangkat.

Landasan untuk jaringan UPnP adalah pengalamatan IP. Perangkat UPnP atau titik kontrol MUNGKIN mendukung IP versi 4 saja, atau keduanya IP versi 4 dan IP versi 6. Bagian ini, dan contoh yang diberikan di seluruh bagian 1 hingga 5 dokumen ini, mengasumsikan implementasi IPv4. Lampiran A dari dokumen ini menjelaskan operasi IPv6. Setiap perangkat UPnP atau titik kontrol yang tidak mengimplementasikan server DHCP itu sendiri HARUS memiliki klien Dynamic Host Configuration Protocol (DHCP) dan mencari server DHCP ketika perangkat atau titik kontrol pertama kali terhubung ke jaringan (jika perangkat atau titik kontrol) sendiri mengimplementasikan server DHCP, MUNGKIN mengalokasikan sendiri alamat dari kumpulan yang dikontrolnya). Jika server DHCP tersedia, yaitu jaringan dikelola; perangkat atau titik kontrol HARUS menggunakan alamat IP yang ditetapkan untuk itu. Jika server DHCP tidak tersedia, mis., Jaringan tidak dikelola; perangkat atau titik kontrol HARUS menggunakan alamat IP otomatis (Auto-IP) untuk mendapatkan alamat.

Auto-IP (didefinisikan dalam RFC 3927) mendefinisikan bagaimana perangkat atau titik kontrol: (a) menentukan apakah DHCP tidak tersedia, dan (b) dengan cerdas memilih

alamat IP dari sekumpulan alamat IP link-local. Metode penetapan alamat ini memungkinkan perangkat atau titik kontrol untuk dengan mudah berpindah antara jaringan yang dikelola dan tidak dikelola. Bagian ini memberikan gambaran umum tentang operasi dasar IP Otomatis. Operasi yang dijelaskan di bagian ini dirinci dan diklarifikasi dalam dokumen referensi yang tercantum di bawah ini. Jika ada konflik antara dokumen ini dan dokumen referensi, dokumen referensi selalu diutamakan [17].

3.3. Auto IP

Perangkat atau titik kontrol yang mendukung Auto-IP dan dikonfigurasi untuk penetapan alamat dinamis dimulai dengan meminta alamat IP melalui DHCP dengan mengirimkan pesan DHCPDISCOVER. Jumlah waktu Klien DHCP ini mendengarkan DHCPOFFER bergantung pada implementasi. Jika DHCPOFFER diterima selama waktu ini, perangkat atau titik kontrol HARUS melanjutkan proses penetapan alamat dinamis. Jika tidak ada DHCPOFFER yang valid yang diterima, perangkat atau titik kontrol HARUS kemudian mengkonfigurasi alamat IP secara otomatis menggunakan IP Otomatis.

3.4. Pemilihan Alamat

Untuk mengkonfigurasi alamat IP secara otomatis menggunakan Auto-IP, perangkat atau titik kontrol menggunakan algoritma yang bergantung pada implementasi untuk memilih alamat dalam rentang 169.254 / 16. 256 alamat pertama dan terakhir dalam rentang ini dicadangkan dan

TIDAK HARUS digunakan. Alamat yang dipilih HARUS kemudian diuji untuk menentukan apakah alamat tersebut sudah digunakan. Jika alamat sedang digunakan oleh perangkat atau titik kontrol lain, alamat lain HARUS dipilih dan diuji, hingga jumlah percobaan ulang yang bergantung pada implementasi. Pemilihan alamat HARUS diacak untuk menghindari benturan ketika beberapa perangkat atau titik kontrol mencoba mengalokasikan alamat. Perangkat atau titik kontrol memilih alamat menggunakan algoritma pseudo-random (didistribusikan ke seluruh rentang alamat dari 169.254.1.0 hingga 169.254.254.255) untuk meminimalkan kemungkinan bahwa perangkat atau titik kontrol yang bergabung dengan jaringan pada saat yang sama akan memilih alamat yang sama dan kemudian memilih alamat alternatif dalam urutan yang sama saat tabrakan terdeteksi. Algoritme acak-semu ini HARUS diunggulkan menggunakan alamat MAC perangkat keras Ethernet dari perangkat atau titik kontrol.

3.5. Pengujian Alamat

Untuk menguji alamat yang dipilih, perangkat atau titik kontrol HARUS menggunakan probe Address Resolution Protocol (ARP). Probe ARP adalah permintaan ARP dengan alamat perangkat keras atau titik kontrol yang digunakan sebagai alamat perangkat keras pengirim dan alamat IP pengirim disetel ke 0s. Perangkat atau titik kontrol HARUS mendengarkan tanggapan ke probe ARP, atau probe ARP lain untuk alamat IP yang sama. Jika salah satu dari paket ARP ini terlihat, perangkat atau titik kontrol HARUS mempertimbangkan alamat yang digunakan dan mencoba alamat yang berbeda. Probe ARP DAPAT diulangi untuk

kepastian yang lebih besar bahwa alamat tersebut belum digunakan; DIANJURKAN bahwa probe dikirim empat kali dengan interval dua detik. Setelah berhasil mengonfigurasi alamat tautan-lokal, perangkat atau titik kontrol HARUS mengirim dua ARP serampangan, berjarak dua detik, kali ini mengisi alamat IP pengirim. Tujuan dari ARP serampangan ini adalah untuk memastikan bahwa host lain di internet tidak memiliki entri cache ARP basi yang tersisa dari beberapa host lain yang sebelumnya mungkin telah menggunakan alamat yang sama.

Perangkat dan titik kontrol yang dilengkapi dengan penyimpanan persisten MUNGKIN merekam alamat IP yang telah mereka pilih dan pada boot berikutnya menggunakan alamat tersebut sebagai kandidat pertama mereka saat menyelidiki, untuk meningkatkan stabilitas alamat dan mengurangi kebutuhan untuk menyelesaikan konflik alamat. Deteksi tabrakan alamat tidak terbatas pada fase pengujian alamat, saat perangkat atau titik kontrol mengirim probe ARP dan mendengarkan balasan. Deteksi tabrakan alamat adalah proses berkelanjutan yang berlaku selama perangkat atau titik kontrol menggunakan alamat tautan-lokal. Kapan pun, jika perangkat atau titik kontrol menerima paket ARP dengan alamat IP-nya sendiri yang diberikan sebagai alamat IP pengirim, tetapi alamat perangkat keras pengirim yang tidak cocok dengan alamat perangkat kerasnya sendiri, perangkat atau titik kontrol HARUS memperlakukan ini sebagai tabrakan alamat dan HARUS merespons seperti yang dijelaskan di (a) atau (b) di bawah ini:

a) Segera konfigurasikan alamat IP link-local baru seperti yang dijelaskan di atas; atau,

b) Jika perangkat atau titik kontrol saat ini memiliki koneksi TCP aktif atau alasan lain untuk memilih menyimpan alamat IP yang sama, dan belum melihat paket ARP lain yang berkonflik baru-baru ini (misalnya, dalam sepuluh detik terakhir) maka MUNGKIN memilih untuk mencoba mempertahankan alamatnya satu kali, dengan mencatat waktu saat paket ARP yang bentrok diterima, dan kemudian menyiarkan satu ARP yang tidak beralasan, memberikan alamat IP dan perangkat kerasnya sendiri sebagai alamat sumber ARP. Namun, jika paket ARP lain yang berkonflik diterima dalam waktu singkat setelah itu (mis., Dalam sepuluh detik) maka perangkat atau titik kontrol HARUS segera mengkonfigurasi alamat IP Otomatis baru seperti yang dijelaskan di atas.

Perangkat atau titik kontrol HARUS merespons paket ARP yang bertentangan seperti yang dijelaskan dalam (a) atau (b) di atas; itu TIDAK HARUS mengabaikan paket ARP yang saling bertentangan. Jika alamat baru dipilih, perangkat atau titik kontrol HARUS membatalkan iklan sebelumnya dan membacanya kembali dengan alamat baru. Setelah berhasil mengonfigurasi alamat IP Otomatis, semua paket ARP berikutnya (balasan serta permintaan) yang berisi alamat sumber IP Otomatis HARUS dikirim menggunakan siaran tingkat tautan alih-alih unicast tingkat tautan, untuk memfasilitasi deteksi duplikat tepat waktu alamat.

3.6. Aturan Forwarding

Paket IP yang alamat sumber atau tujuannya berada dalam kisaran 169.254 / 16 TIDAK HARUS dikirim ke router mana pun untuk diteruskan. Sebagai gantinya, pengirim HARUS ARP untuk alamat tujuan dan kemudian mengirim paket langsung ke tujuan di link yang sama. Datagram IP dengan alamat tujuan multicast dan alamat sumber Auto-IP TIDAK HARUS diteruskan dari tautan lokal. Perangkat dan titik kontrol MUNGKIN mengasumsikan bahwa semua 169.254 / 16 alamat tujuan terhubung dan langsung dapat dijangkau. Rentang alamat 169.254 / 16 TIDAK HARUS di-subnet.

3.7. Cek ketersediaan IP

Perangkat atau titik kontrol yang memiliki konfigurasi otomatis alamat IP HARUS memeriksa keberadaan server DHCP secara berkala. Ini dilakukan dengan mengirim pesan DHCPDISCOVER. Seberapa sering pemeriksaan ini dilakukan bergantung pada implementasi, tetapi pemeriksaan setiap 5 menit akan menjaga keseimbangan antara bandwidth jaringan yang diperlukan dan pemeliharaan konektivitas. Jika DHCPOFFER diterima, perangkat atau titik kontrol HARUS melanjutkan alokasi alamat dinamis. Setelah alamat yang ditetapkan DHCP ditempatkan, perangkat atau titik kontrol MUNGKIN melepaskan alamat yang dikonfigurasi secara otomatis, tetapi MUNGKIN juga memilih untuk mempertahankan alamat ini selama jangka waktu tertentu (atau tanpa batas) untuk menjaga konektivitas.

Untuk beralih dari satu alamat IP ke yang baru, perangkat HARUS, jika memungkinkan, membatalkan iklan yang belum selesai dibuat pada alamat sebelumnya dan HARUS menerbitkan iklan baru di alamat baru. Bagian Penemuan menjelaskan iklan dan pembatalannya. Selain itu, langganan acara apa pun dihapus oleh perangkat (lihat bagian tentang Acara). Untuk perangkat multi-rumah dengan beberapa alamat IP, untuk mengalihkan salah satu alamat IP ke yang baru, perangkat HARUS membatalkan semua iklan yang beredar yang dibuat pada alamat IP sebelumnya, dan HARUS menerbitkan iklan baru pada alamat IP baru. Selain itu, HARUS juga mengeluarkan iklan pembaruan yang sesuai pada semua alamat IP yang tidak terpengaruh. Bagian Penemuan menjelaskan iklan, pembatalan dan pembaruannya. Bagian Eventing menjelaskan efek pada langganan acara.

3.8. Penamaan alat dan interaksi DNS

Setelah perangkat memiliki alamat IP yang valid untuk jaringan tersebut, perangkat tersebut dapat ditemukan dan direferensikan di jaringan itu melalui alamat itu. Mungkin ada situasi di mana pengguna akhir perlu mencari dan mengidentifikasi perangkat. Dalam situasi ini, nama yang bersahabat untuk perangkat jauh lebih mudah digunakan manusia daripada alamat IP. Jika perangkat memilih untuk memberikan nama host ke server DHCP dan mendaftar dengan server DNS, perangkat HARUS memastikan nama host yang diminta unik atau menyediakan cara bagi pengguna untuk mengubah nama host yang diminta. Paling sering, perangkat tidak memberikan nama host, tetapi menyediakan URL menggunakan alamat IP literal (numerik) [18].

Selain itu, nama jauh lebih statis daripada alamat IP. Klien yang merujuk perangkat dengan nama tidak memerlukan modifikasi apa pun ketika alamat IP perangkat berubah. Pemetaan nama DNS perangkat ke alamat IP-nya dapat dimasukkan ke dalam basis data DNS secara manual atau dinamis sesuai dengan RFC 2136. Sementara perangkat yang mendukung pembaruan DNS dinamis dapat mendaftarkan catatan DNS mereka langsung di DNS, juga memungkinkan untuk mengkonfigurasi server DHCP untuk mendaftarkan catatan DNS atas nama klien DHCP ini.

4. Protokol CoAP

4.1. Apa itu CoAP

Penggunaan layanan web (API web) di Internet telah ada di mana-mana di sebagian besar aplikasi dan bergantung pada arsitektur dasar Representational State Transfer [REST] dari Web. Pekerjaan pada Constrained RESTful Environments (CoRE) bertujuan untuk mewujudkan arsitektur REST dalam bentuk yang sesuai untuk node yang paling terbatas (misalnya, mikrokontroler 8-bit dengan RAM dan ROM terbatas) dan jaringan (misalnya, 6LoWPAN, [RFC4944]). Jaringan terbatas seperti 6LoWPAN mendukung fragmentasi paket IPv6 menjadi bingkai lapisan tautan kecil; Namun, hal ini menyebabkan penurunan yang signifikan dalam kemungkinan pengiriman paket. Salah satu tujuan desain CoAP adalah menjaga overhead pesan tetap kecil, sehingga membatasi kebutuhan untuk fragmentasi [19].

Salah satu tujuan utama CoAP adalah merancang protokol web generik untuk persyaratan khusus dari lingkungan terbatas ini, terutama mempertimbangkan energi, otomatisasi gedung, dan aplikasi mesin-ke-mesin (M2M) lainnya. Tujuan CoAP bukanlah untuk mengompresi HTTP [RFC2616] secara membabi buta, melainkan untuk mewujudkan subset REST yang umum dengan HTTP tetapi dioptimalkan untuk aplikasi M2M. Meskipun CoAP dapat digunakan untuk merombak antarmuka HTTP sederhana menjadi protokol yang lebih kompak, yang lebih penting CoAP juga menawarkan fitur untuk M2M seperti penemuan bawaan, dukungan multicast,

dan pertukaran pesan asinkron. Dokumen ini menetapkan Constrained Application Protocol (CoAP), yang dengan mudah diterjemahkan menjadi HTTP untuk integrasi dengan Web yang ada sambil memenuhi persyaratan khusus seperti dukungan multicast, overhead yang sangat rendah, dan kesederhanaan untuk lingkungan terbatas dan aplikasi M2M.

4.2. Fitur CoAP

CoAP memiliki beberapa fitur utama berikut:

- Protokol web yang memenuhi persyaratan M2M dalam lingkungan terbatas
- UDP [RFC0768] mengikat dengan keandalan opsional yang mendukung permintaan unicast dan multicast.
- Pertukaran pesan asinkron.
- Overhead header rendah dan kompleksitas parsing.
- URI dan dukungan tipe konten.
- Proxy sederhana dan kemampuan caching.
- Pemetaan HTTP stateless, yang memungkinkan pembuatan proxy yang menyediakan akses ke sumber daya CoAP melalui HTTP dengan cara yang seragam atau untuk antarmuka sederhana HTTP untuk direalisasikan sebagai alternatif melalui CoAP.
- Keamanan mengikat Datagram Transport Layer Security (DTLS) [RFC6347].

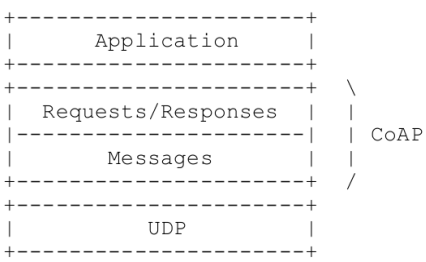
4.3. Constrained Application Protocol

Model interaksi CoAP mirip dengan model klien / server HTTP. Namun, interaksi mesin-ke-mesin biasanya menghasilkan implementasi CoAP yang bertindak di peran

klien dan server. Permintaan CoAP setara dengan HTTP dan dikirim oleh klien untuk meminta tindakan (menggunakan Kode Metode) pada sumber daya (diidentifikasi oleh URI) di server. Server kemudian mengirimkan respons dengan Kode Respons; tanggapan ini mungkin termasuk representasi sumber daya [7] [20].

Tidak seperti HTTP, CoAP menangani pertukaran ini secara asinkron melalui transportasi berorientasi datagram seperti UDP. Ini dilakukan secara logis menggunakan lapisan pesan yang mendukung keandalan opsional (dengan kemunduran eksponensial). CoAP mendefinisikan empat jenis pesan: Dapat Dikonfirmasi, Tidak Dapat Dikonfirmasi, Pengakuan, Setel Ulang. Kode Metode dan Kode Respon termasuk dalam beberapa pesan ini membuat mereka membawa permintaan atau tanggapan. Pertukaran dasar dari empat jenis pesan agak ortogonal dengan interaksi permintaan / tanggapan; permintaan dapat dibawa dalam pesan yang Dapat Dikonfirmasi dan Tidak Dapat Dikonfirmasi, dan tanggapan dapat dilakukan dalam pesan ini serta didukung dalam pesan Pengakuan.

Seseorang dapat menganggap CoAP secara logis sebagai menggunakan pendekatan dua lapisan, lapisan pesan CoAP yang digunakan untuk menangani UDP dan sifat asinkron dari interaksi, dan interaksi permintaan / tanggapan menggunakan Metode dan Kode Tanggapan (lihat Gambar 1). Namun CoAP adalah satu protokol, dengan pesan dan permintaan / tanggapan hanya sebagai fitur dari header CoAP.



Gambar 4.3.1: Layer CoAP

4.4. Model pertukaran pesan

Model pesan CoAP didasarkan pada pertukaran pesan melalui UDP antara titik akhir. CoAP menggunakan header biner panjang tetap yang pendek (4 byte) yang dapat diikuti oleh opsi biner kompak dan muatan. Format pesan ini dibagikan oleh permintaan dan tanggapan. Format pesan CoAP ditentukan di Bagian 3. Setiap pesan berisi ID Pesan yang digunakan untuk mendeteksi duplikat dan untuk keandalan opsional. (ID Pesan ringkas; ukuran 16-bitnya memungkinkan hingga sekitar 250 pesan per detik dari satu titik akhir ke titik lain dengan parameter protokol default.)Keandalan diberikan dengan menandai pesan sebagai Dapat Dikonfirmasi (CON). Pesan yang Dapat Dikonfirmasi dikirim ulang menggunakan batas waktu default dan mundur eksponensial antara pengiriman ulang, hingga penerima mengirim pesan Pengakuan (ACK) dengan ID Pesan yang sama (dalam contoh ini, 0x7d34) dari titik akhir yang sesuai; lihat Gambar 2. Ketika penerima sama sekali tidak dapat memproses pesan yang Dapat Dikonfirmasi (yaitu, bahkan tidak dapat memberikan respons kesalahan yang sesuai),

itumembalas dengan pesan Reset (RST) dan bukan dengan (ACK).

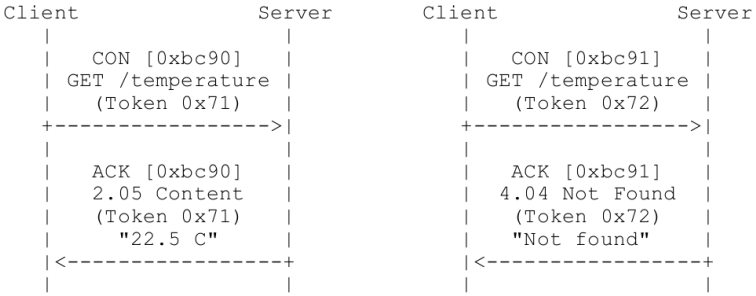
Pesan yang tidak memerlukan transmisi yang andal (misalnya, setiap pengukuran keluar dari aliran data sensor) dapat dikirim sebagai pesan Non-confirmable (NON). Ini tidak diakui, tetapi masih memiliki ID Pesan untuk deteksi duplikat (dalam contoh ini, 0x01a0). Ketika penerima tidak dapat memproses pesan Non-confirmable, ia mungkin membalas dengan pesan Reset (RST).

4.5. Request/Response Model

Permintaan CoAP dan semantik respons dibawa dalam pesan CoAP, yang masing-masing menyertakan Kode Metode atau Kode Respons. Informasi permintaan dan respons opsional (atau default), seperti URI dan jenis media payload dibawa sebagai opsi CoAP. Token digunakan untuk mencocokkan tanggapan terhadap permintaan secara independen dari pesan yang mendasarinya. (Perhatikan bahwa Token adalah konsep yang terpisah dari ID Pesan.)

Permintaan dibawa dalam pesan yang Dapat Dikonfirmasi (CON) atau Tidak Dapat Dikonfirmasi (NON), dan, jika segera tersedia, tanggapan atas permintaan yang dibawa dalam pesan Dapat Dikonfirmasi dibawa dalam pesan Pengakuan (ACK) yang dihasilkan. Ini disebut respons piggyback (Tidak perlu mengakui respons piggyback secara terpisah, karena klien akan mengirimkan ulang permintaan tersebut jika pesan Pengakuan yang membawa respons piggyback hilang.) Dua contoh untuk permintaan GET dasar

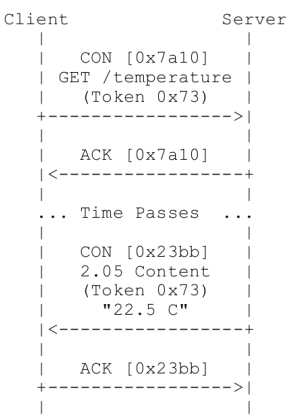
dengan respons piggyback ditampilkan pada Gambar 4.5.1, satu berhasil, satu menghasilkan respons 404 (Tidak Ditemukan).



Gambar 4.5.1: Request GET

Jika server tidak dapat segera menanggapi permintaan yang dibawa dalam pesan yang Dapat Dikonfirmasi, itu hanya merespons dengan pesan Pengakuan Kosong sehingga klien dapat berhenti mengirim ulang permintaan tersebut.

Ketika respon sudah siap, server mengirimkannya dalam pesan baru yang Dapat Dikonfirmasi (yang kemudian perlu diakui oleh klien). Ini disebut "respons terpisah", seperti yang diilustrasikan pada Gambar 4.5.2.



Gambar 4.5.2: GET Request dengan respon terpisah

CoAP menggunakan metode GET, PUT, POST, dan DELETE dengan cara yang mirip dengan HTTP. (Perhatikan bahwa semantik rinci metode CoAP "hampir, tetapi tidak sepenuhnya berbeda" [HHGTTG] metode HTTP: intuisi yang diambil dari pengalaman HTTP umumnya berlaku dengan baik, tetapi ada cukup banyak perbedaan yang membuatnya bermanfaat untuk benar-benar membaca saat ini spesifikasi.)

Metode di luar empat dasar dapat ditambahkan ke CoAP dalam spesifikasi terpisah. Metode baru tidak harus menggunakan permintaan dan tanggapan secara berpasangan. Bahkan untuk metode yang ada, satu permintaan dapat menghasilkan banyak tanggapan, misalnya, untuk permintaan multicast atau dengan opsi Amati [OBSERVE]. Dukungan URI di server disederhanakan karena klien sudah mengurai URI dan membaginya menjadi komponen host, port, jalur, dan kueri,

menggunakan nilai default untuk efisiensi. Kode Respon berhubungan dengan subset kecil dari kode status HTTP dengan beberapa kode khusus CoAP yang ditambahkan.

4.6. Intermediaries dan Caching

Protokol mendukung caching respons untuk memenuhi permintaan secara efisien. Caching sederhana diaktifkan menggunakan informasi kesegaran dan validitas yang dibawa dengan tanggapan CoAP. Cache dapat ditemukan di titik akhir atau perantara.

Proxy berguna dalam jaringan terbatas karena beberapa alasan, termasuk untuk membatasi lalu lintas jaringan, untuk meningkatkan kinerja, untuk mengakses sumber daya perangkat tidur, dan untuk alasan keamanan. Proksi permintaan atas nama titik akhir CoAP lain didukung dalam protokol. Saat menggunakan proxy, URI sumber daya yang akan diminta disertakan dalam permintaan, sedangkan alamat IP tujuan disetel ke alamat proxy.

Karena CoAP dirancang sesuai dengan arsitektur REST [REST], dan dengan demikian menunjukkan fungsionalitas yang mirip dengan protokol HTTP, sangat mudah untuk memetakan dari CoAP ke HTTP dan dari HTTP ke CoAP. Pemetaan seperti itu dapat digunakan untuk mewujudkan antarmuka HTTP REST menggunakan CoAP atau untuk mengubah antara HTTP dan CoAP. Konversi ini dapat dilakukan oleh proxy lintas protokol ("proxy silang"), yang mengubah Metode atau Kode Respons, jenis media, dan opsi ke fitur HTTP yang sesuai.

4.7. Pengiriman Pesan

Pesan CoAP dipertukarkan secara asinkron antara titik akhir CoAP. Mereka digunakan untuk mengangkut permintaan dan tanggapan CoAP. Karena CoAP terikat pada pengangkutan yang tidak dapat diandalkan seperti UDP, pesan CoAP mungkin tiba rusak, tampak digandakan, atau hilang tanpa pemberitahuan. Untuk alasan ini, CoAP mengimplementasikan mekanisme keandalan yang ringan, tanpa mencoba membuat ulang set fitur lengkap dari sebuah transport seperti TCP. Ini memiliki beberapa fitur berikut:

- Kehandalan pengiriman ulang stop-and-wait sederhana dengan back-off eksponensial untuk pesan Dikonfirmasi.
- Deteksi duplikat untuk pesan yang Dapat Dikonfirmasi dan Tidak Dapat Dikonfirmasi.

4.8. Messages dan Endpoints

Titik akhir CoAP adalah sumber atau tujuan pesan CoAP. Definisi spesifik dari titik akhir bergantung pada pengangkutan yang digunakan untuk CoAP. Untuk pengangkutan yang ditentukan dalam spesifikasi ini, titik akhir diidentifikasi tergantung pada mode keamanan yang digunakan (lihat Bagian 9): Tanpa keamanan, titik akhir hanya diidentifikasi oleh alamat IP dan nomor port UDP. Dengan mode keamanan lainnya, titik akhir diidentifikasi seperti yang ditentukan oleh mode keamanan.

Ada berbagai jenis pesan. Jenis pesan ditentukan oleh bidang Jenis Header CoAP. Terlepas dari jenis pesannya, sebuah pesan

mungkin membawa permintaan, tanggapan, atau Kosong. Ini ditandai dengan kolom Request / Response Code di CoAP Header dan relevan dengan model request / response. Nilai yang mungkin untuk bidang tersebut disimpan dalam Daftar Kode CoAP (Bagian 12.1). Pesan Kosong memiliki bidang Kode yang disetel ke 0,00. Bidang Panjang Token HARUS disetel ke 0 dan byte data TIDAK HARUS ada setelah bidang ID Pesan. Jika ada byte, mereka HARUS diproses sebagai kesalahan format pesan.

4.9. Pesan Dikirim dengan Andal

Transmisi pesan yang andal dimulai dengan menandai pesan sebagai Dapat Dikonfirmasi di header CoAP. Pesan yang dapat dikonfirmasi selalu membawa permintaan atau respons, kecuali jika hanya digunakan untuk mendapatkan pesan Reset, dalam hal ini Kosong. Penerima HARUS (a) mengakui pesan yang Dapat Dikonfirmasi dengan pesan Pengakuan atau (b) menolak pesan jika penerima tidak memiliki konteks untuk memproses pesan dengan benar, termasuk situasi di mana pesan Kosong, menggunakan kode dengan kelas yang dicadangkan (1 , 6, atau 7), atau memiliki kesalahan format pesan [21]. Menolak pesan yang Dapat Dikonfirmasi dilakukan dengan mengirimkan pesan Reset yang cocok dan sebaliknya mengabaikannya. Pesan Pengakuan HARUS menggemakan ID Pesan dari pesan yang Dapat Dikonfirmasi dan HARUS memberikan tanggapan atau menjadi Kosong. Pesan Reset HARUS menggemakan ID Pesan dari pesan yang Dapat Dikonfirmasi dan HARUS Kosong. Menolak pesan Pengakuan atau Atur Ulang (termasuk kasus di mana Pengakuan membawa permintaan atau kode dengan kelas yang

dicadangkan, atau pesan Atur Ulang tidak Kosong) dilakukan dengan mengabaikannya secara diam-diam. Secara lebih umum, penerima pesan Pengakuan dan Atur Ulang TIDAK HARUS merespons dengan Pengakuan atau Atur Ulang pesan.

Pengirim mentransmisikan ulang pesan yang Dapat Dikonfirmasi pada interval yang meningkat secara eksponensial, hingga menerima pengakuan (atau Atur ulang pesan) atau percobaan habis.

Transmisi ulang dikendalikan oleh dua hal yang HARUS dilacak oleh titik akhir CoAP untuk setiap pesan yang Dapat Dikonfirmasi yang dikirimnya sambil menunggu pengakuan (atau reset): batas waktu dan penghitung transmisi ulang. Untuk pesan baru yang Dapat Dikonfirmasi, waktu tunggu awal disetel ke durasi acak (seringkali bukan merupakan jumlah integral dari detik) antara ACK_TIMEOUT dan (ACK_TIMEOUT * ACK_RANDOM_FACTOR) (lihat Bagian 4.8), dan penghitung pengiriman ulang disetel ke 0. Ketika batas waktu dipicu dan penghitung pengiriman ulang kurang dari MAX_RETRANSMIT, pesan ditransmisikan ulang, penghitung transmisi ulang bertambah, dan batas waktu habis digandakan. Jika penghitung pengiriman ulang mencapai MAX_RETRANSMIT pada waktu habis, atau jika titik akhir menerima pesan Reset, maka upaya untuk mengirimkan pesan dibatalkan dan proses aplikasi diberitahu tentang kegagalan. Di sisi lain, jika titik akhir menerima pengakuan tepat waktu, transmisi dianggap berhasil.

Spesifikasi ini tidak membuat persyaratan yang kuat tentang keakuratan jam yang digunakan untuk mengimplementasikan algoritma back-off eksponensial biner di atas. Secara khusus, titik akhir mungkin terlambat untuk transmisi ulang tertentu karena jadwal tidurnya dan mungkin menyusul yang berikutnya. Namun, jarak minimum sebelum transmisi ulang lainnya adalah ACK_TIMEOUT, dan seluruh urutan transmisi (ulang) HARUS berada dalam amplop MAX_TRANSMIT_SPAN (lihat Bagian 4.8.2), meskipun itu berarti pengirim mungkin kehilangan kesempatan untuk mengirim.

Titik akhir CoAP yang mengirim pesan yang Dapat Dikonfirmasi MUNGKIN menyerah dalam upaya mendapatkan ACK bahkan sebelum nilai penghitung MAX_RETRANSMIT tercapai. Misalnya, aplikasi telah membatalkan permintaan karena tidak lagi membutuhkan tanggapan, atau ada indikasi lain bahwa pesan CON memang tiba. Secara khusus, pesan permintaan CoAP mungkin telah menimbulkan respons terpisah, dalam hal ini jelas bagi pemohon bahwa hanya ACK yang hilang dan pengiriman ulang permintaan tidak akan berguna. Namun, responder TIDAK HARUS bergantung pada perilaku lintas-lapisan dari pemohon, yaitu, HARUS mempertahankan status untuk membuat ACK untuk permintaan tersebut, jika diperlukan, bahkan jika respons yang Dapat Dikonfirmasi sudah diakui oleh pemohon.

Alasan lain untuk menghentikan pengiriman ulang MUNGKIN adalah diterimanya kesalahan ICMP. Jika diinginkan untuk memperhitungkan kesalahan ICMP, untuk

mengurangi potensi serangan spoofing, implementasi HARUS berhati-hati untuk memeriksa informasi tentang datagram asli dalam pesan ICMP, termasuk nomor port dan informasi header CoAP seperti jenis dan kode pesan, ID Pesan , dan Token; jika ini tidak memungkinkan karena keterbatasan API layanan UDP, kesalahan ICMP HARUS diabaikan. Kesalahan Paket Terlalu Besar [RFC4443] ("diperlukan fragmentasi dan set DF" untuk IPv4 [RFC0792]) tidak dapat terjadi dengan benar dan HARUS diabaikan jika catatan implementasi di Bagian 4.6 diikuti; jika tidak, mereka HARUS dimasukkan ke dalam algoritma penemuan MTU jalur [RFC4821]. Sumber Quench and Time Exceeded Pesan ICMP HARUS diabaikan. Kesalahan host, jaringan, port, atau protokol yang tidak dapat dijangkau atau kesalahan masalah parameter MUNGKIN, setelah pemeriksaan yang sesuai, digunakan untuk menginformasikan aplikasi tentang kegagalan dalam pengiriman.

5. Protokol MQTT

5.1. Pengenalan

Konsep yang muncul dari Internet of Things adalah fondasi penting di mana visi untuk Planet yang Lebih Cerdas akan terwujud. Selain itu, mendukung Internet of Things adalah pendekatan telemetri baru yang lebih canggih yang memungkinkan untuk menghubungkan semua jenis perangkat, di mana pun mereka berada, satu sama lain, ke Internet, dan ke perusahaan bisnis. Salah satu kemajuan ini adalah protokol perpesanan MQTT. Ini sangat ringan sehingga dapat didukung oleh beberapa perangkat pengukur dan pemantauan terkecil, dan mengirimkan data melalui jaringan yang berjauhan, terkadang terputus-putus. Ini juga open source, yang membuatnya mudah untuk beradaptasi dengan berbagai kebutuhan perpesanan dan komunikasi. Sebelum masuk ke detail MQTT, sebaiknya lihat sekilas dunia yang sedang berkembang yang sedang dikerjakan oleh pengembang yang menggunakan MQTT untuk dihubungkan [22].

5.2. MQTT

Siapa pun yang telah mengarahkan browser web ke mesin pencari atau situs media sosial mengetahui kekuatan Internet untuk menghubungkan orang ke informasi atau ke orang lain. Namun, dengan munculnya berbagai perangkat pintar, Internet akan berkembang untuk memasukkan apa yang beberapa orang sebut sebagai Internet of Things: Miliaran perangkat pintar yang saling berhubungan mengukur, bergerak, dan bertindak atas, terkadang secara independen, semua bit data

yang menyusun kehidupan sehari-hari Untuk membayangkan apa yang mungkin dihasilkan Internet of Things dalam 10 atau 20 tahun, pikirkan tentang hal-hal luar biasa yang sudah dapat kita lakukan dari jarak jauh:

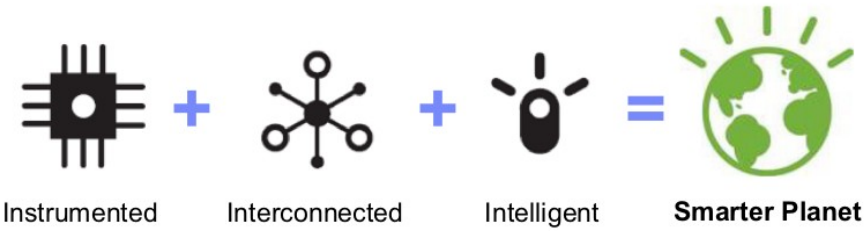
- Seorang dokter dapat memeriksa pasien di kota yang jauh dan melihat informasi status kesehatan secara real-time, seperti tekanan darah, detak jantung, dan sebagainya.
- Perusahaan energi dapat memantau jaringan pipa minyak atau gas yang panjangnya ratusan mil dan memutus aliran dari jarak jauh jika masalah terdeteksi.
- Pemilik rumah dapat melihat rumahnya di halaman web, lengkap dengan status perangkat interior seperti alarm keamanan, sistem pemanas, dan lainnya.

Internet of Things bahkan akan melampaui contoh-contoh ini, tidak hanya orang yang berinteraksi dengan perangkat tetapi juga perangkat yang berinteraksi satu sama lain, menciptakan apa yang pada akhirnya mungkin menjadi sesuatu dari sistem saraf pusat global.

Konsep IBM tentang Planet yang Lebih Cerdas dibangun di atas serangkaian pilar berikut yang disebut Tiga Is [23], seperti yang diilustrasikan pada Gambar 5.2.1:

- Terinstrumentasi: Informasi ditangkap di mana pun ada, seperti melalui penggunaan sensor jarak jauh.
- Terinterkoneksi: Informasi dipindahkan dari tempat pengumpulan ke mana pun yang dapat digunakan secara berguna.

- Cerdas: Informasi diproses, dianalisis, dan ditindaklanjuti untuk memperoleh nilai dan pengetahuan maksimum.



Gambar 5.2.1: Aplikasi Smart Planet

Dunia sudah semakin terinstrumentasi, dengan contoh mulai dari sensor kecil dan tag RFID dalam produk yang berdiri sendiri, melalui ponsel cerdas dan perangkat GPS yang sadar lokasi hingga PC notebook dan sistem tertanam. Perangkat ini biasanya memiliki daya komputasi yang cukup untuk mengumpulkan dan mengirimkan data, dan beberapa memiliki cukup daya untuk menanggapi permintaan untuk mengubah perilakunya [24].

Perangkat ini juga hampir semuanya terhubung sampai batas tertentu. Sebagian besar memiliki, atau akan memiliki, alamat Internet mereka sendiri, yang dengannya mereka dapat berkomunikasi secara langsung melalui jaringan lokal atau secara tidak langsung melalui cloud. Jadi konsep Internet of Things sudah mulai muncul.

Langkah selanjutnya adalah mengumpulkan semua data yang dikumpulkan oleh perangkat kecil, menengah, atau bahkan besar ini, merutekan data tersebut ke tempat yang paling tepat untuk menafsirkannya, dan menggunakan sumber daya komputasi yang sangat luas di dunia untuk memahami apa yang terjadi dan merespons. seperlunya untuk membuat hidup lebih baik.

Teknologi telemetri memungkinkan hal-hal diukur atau dipantau dari kejauhan. Selain itu, saat ini, peningkatan teknologi telemetri memungkinkan untuk menghubungkan perangkat pengukur dan pemantauan di lokasi yang berbeda dan untuk mengurangi biaya pembuatan aplikasi yang dapat berjalan di perangkat pintar ini untuk membuatnya lebih berguna [25].

Orang, bisnis, dan pemerintah semakin beralih ke perangkat pintar dan telemetri untuk berinteraksi lebih cerdas dengan dunia. Seorang pria yang berbelanja bahan makanan ingin tahu apa yang saat ini ada di dapur rumahnya di rumah. Seorang wanita yang menuju ke Barcelona ingin tahu apakah penerbangan ke kota itu saat ini tertunda karena cuaca. Seorang pengendara mobil yang mengemudi melintasi kota ingin tahu apakah jalan raya utama menuju ke sana masih terhalang oleh kecelakaan mobil yang dilaporkan pada berita pagi. Seorang dokter dengan pasien akan tiba di kantor pada jam 3 sore. ingin tahu, di pagi hari, apakah tekanan darah pasien cukup stabil untuk melakukan perjalanan dengan selamat. Informasi untuk membantu setiap keputusan ini dapat datang, atau suatu hari nanti, dari berbagai pengukur dan

perangkat pintar. Namun tantangannya terletak pada menyampaikan informasi dari perangkat kepada orang-orang dan aplikasi yang ingin menggunakannya, tepat waktu bagi mereka untuk menggunakannya secara efektif dan, idealnya, dengan kemampuan tambahan bagi mereka untuk membalas perangkat dengan instruksi atau permintaan baru.

Jika perangkat didistribusikan secara luas secara geografis, atau jika mereka memiliki penyimpanan atau kemampuan komputasi terbatas, tantangan akan meningkat pesat, seperti halnya biaya. Untungnya, tantangan ini sedang diatasi melalui penggunaan teknologi telemetri dan protokol komunikasi yang ditingkatkan yang memungkinkan untuk mengirim dan menerima informasi ini dengan andal melalui Internet, bahkan jika jaringan tidak stabil atau perangkat pemantau memiliki daya pemrosesan yang kecil. MQTT menyediakan teknologi telemetri untuk memenuhi tantangan informasi pengguna Internet saat ini.

MQTT adalah protokol perpesanan yang sangat sederhana dan ringan. Arsitektur publish / subscribe-nya dirancang agar terbuka dan mudah diterapkan, dengan hingga ribuan klien jarak jauh yang mampu didukung oleh satu server. Karakteristik ini membuat MQTT ideal untuk digunakan di lingkungan terbatas di mana bandwidth jaringan rendah atau di mana terdapat latensi tinggi dan dengan perangkat jarak jauh yang mungkin memiliki kemampuan pemrosesan dan memori terbatas [26].

Protokol MQTT mencakup manfaat berikut:

- Memperluas konektivitas di luar batasan perusahaan ke perangkat pintar.
- Menawarkan opsi konektivitas yang dioptimalkan untuk sensor dan perangkat jarak jauh.
- Memberikan data yang relevan ke aset pengambilan keputusan yang cerdas yang dapat menggunakannya.
- Memungkinkan skalabilitas besar-besaran penerapan dan pengelolaan solusi.

MQTT meminimalkan bandwidth jaringan dan kebutuhan sumber daya perangkat sambil berusaha memastikan keandalan dan pengiriman. Pendekatan ini membuat protokol MQTT sangat sesuai untuk menghubungkan mesin ke mesin (M2M), yang merupakan aspek penting dari konsep Internet of Things yang muncul.

Protokol MQTT mencakup hal-hal berikut:

- Terbuka dan bebas royalti untuk adopsi yang mudah. MQTT terbuka untuk memudahkan adopsi dan adaptasi untuk berbagai macam perangkat, platform, dan sistem operasi yang digunakan di tepi jaringan.
- Model pesan terbitkan / langganan yang memfasilitasi distribusi satu-ke-banyak. Aplikasi atau perangkat pengirim tidak perlu tahu apa-apa tentang penerima, bahkan alamatnya.
- Ideal untuk jaringan terbatas (bandwidth rendah, latensi tinggi, batas data, dan koneksi rapuh). Header pesan MQTT dibuat sekecil mungkin. Header tetap hanya dua byte, dan distribusi pesan gaya dorong sesuai

permintaan membuat pemanfaatan jaringan tetap rendah.

- Berbagai tingkat layanan memungkinkan fleksibilitas dalam menangani berbagai jenis pesan. Pengembang dapat menetapkan bahwa pesan akan dikirim paling banyak sekali, setidaknya sekali, atau tepat satu kali.
- Dirancang khusus untuk perangkat jarak jauh dengan sedikit memori atau daya pemrosesan. Header minimal, footprint klien kecil, dan ketergantungan terbatas pada perpustakaan membuat MQTT ideal untuk perangkat terbatas.
- Mudah digunakan dan diterapkan dengan satu set pesan perintah sederhana. Banyak aplikasi MQTT dapat diselesaikan hanya dengan CONNECT, PUBLISH, SUBSCRIBE, dan DISCONNECT.
- Dukungan bawaan untuk kehilangan kontak antara klien dan server. Server diberi tahu ketika koneksi klien terputus secara tidak normal, memungkinkan pesan untuk dikirim kembali atau disimpan untuk pengiriman nanti.

5.3. Konsep Dasar MQTT

Protokol MQTT dibangun di atas beberapa konsep dasar, semuanya bertujuan untuk memastikan pengiriman pesan sambil menjaga pesan itu sendiri seringan mungkin.

Publikasikan / berlangganan

Protokol MQTT didasarkan pada prinsip penerbitan pesan dan berlangganan topik, yang biasanya disebut sebagai

model terbitkan / berlangganan. Klien dapat berlangganan topik yang berkaitan dengan mereka dan dengan demikian menerima pesan apa pun yang diterbitkan untuk topik tersebut. Alternatifnya, klien dapat mempublikasikan pesan ke topik, sehingga membuatnya tersedia untuk semua pelanggan topik tersebut.

Topik dan langganan

Pesan di MQTT dipublikasikan ke topik, yang dapat dianggap sebagai area subjek. Klien, pada gilirannya, mendaftar untuk menerima pesan tertentu dengan berlangganan topik. Langganan bisa eksplisit, yang membatasi pesan yang diterima ke topik tertentu yang ada atau bisa menggunakan penanda wildcard, seperti tanda nomor (#) untuk menerima pesan untuk berbagai topik terkait.

Kualitas tingkat layanan

MQTT mendefinisikan tiga tingkat kualitas layanan (QoS) untuk pengiriman pesan, dengan setiap tingkat menunjukkan tingkat upaya yang lebih tinggi oleh server untuk memastikan bahwa pesan terkirim. Tingkat QoS yang lebih tinggi memastikan pengiriman pesan yang lebih andal, tetapi mungkin menghabiskan lebih banyak bandwidth jaringan atau menyebabkan pesan tertunda karena masalah seperti latensi.

Pesan yang dipertahankan

Dengan MQTT, server menyimpan pesan bahkan setelah mengirimnya ke semua pelanggan saat ini. Jika langganan baru

dikirimkan untuk topik yang sama, pesan yang dipertahankan kemudian dikirim ke klien baru yang berlangganan.

Sesi bersih dan koneksi tahan lama

Ketika klien MQTT terhubung ke server, itu menetapkan bendera sesi bersih. Jika bendera disetel ke true, semua langganan klien akan dihapus ketika terputus dari server. Jika flag disetel ke false, koneksi dianggap tahan lama, dan langganan klien tetap berlaku setelah pemutusan hubungan apa pun. Dalam hal ini, pesan berikutnya yang datang dengan tujuan QoS tinggi disimpan untuk pengiriman setelah koneksi dibuat kembali. Menggunakan bendera sesi bersih adalah opsional.

Dengan perluasan yang cepat dari sensor, monitor, dan bentuk instrumentasi jarak jauh lainnya, kebutuhan untuk mengintegrasikan perangkat pintar ini ke dalam perusahaan dan sistem berbasis web menjadi lebih penting. Sampai saat ini, pendekatan integrasi tipikal telah menggabungkan, sejauh mungkin, model komunikasi standar industri dengan protokol perpesanan pribadi atau non-standar apa pun yang mungkin ada di perangkat atau sudah digunakan dalam perusahaan.

Namun, integrasi seperti itu sering kali didasarkan pada standar point-to-point yang memerlukan penggandengan yang erat antara perangkat atau aplikasi pengirim dan penerima, dan batasan protokol, seperti SOAP atau HTTP, membuat persyaratan pemrograman menjadi cukup ketat. Tantangan tambahan terletak pada penerapan solusi untuk perangkat terbatas dengan penyimpanan minimal dan kemampuan

menghitung atau pada jaringan data yang tidak stabil dan berkinerja rendah. Diperlukan perpesanan sumber terbuka yang dapat bertukar data dan peristiwa antara perangkat di dunia fisik dan dunia virtual perusahaan atau sistem berbasis web. Ini harus mendukung model pesan terbuka yang semakin lazim dan juga memenuhi kebutuhan khusus perangkat dan jaringan yang dibatasi. Perpesanan sumber terbuka memungkinkan peralihan dari protokol yang ada dengan gaya 1-ke-1 ke model yang lebih longgar digabungkan dan menjembatani arsitektur middleware baru dengan arsitektur perangkat tertanam dan nirkabel yang terkait dengan komunikasi mesin-ke-mesin (M2M). Lihat Gambar 1-3. IBM dan Eurotech bergabung dengan Sierra Wireless dan Eclipse Foundation untuk menyediakan alat dan protokol open source untuk proyek Eclipse Paho untuk menyederhanakan pengembangan solusi M2M. Proyek Eclipse Paho ditujukan untuk mengembangkan protokol perpesanan open source, skalabel, dan standar dari jenis yang diperlukan untuk meningkatkan komunikasi dan integrasi M2M dengan web, middleware perusahaan, dan aplikasi. Ini mencakup tujuan utama berikut:

- Pesan dua arah
- Pengiriman pesan yang dapat ditentukan
- Kopling longgar
- Kegunaan platform terbatas

Cakupan proyek Eclipse Paho mencakup perangkat lunak klien untuk digunakan pada perangkat jarak jauh bersama dengan dukungan server yang sesuai. Proyek Eclipse Paho berfokus pada kerangka kerja, contoh praktik terbaik, dan alat

plug-in bagi pengembang untuk mengintegrasikan dan menguji konektivitas ujung-ke-ujung komponen perpesanan. IBM menyumbangkan implementasi kode sisi klien Java dan C dari protokol MQTT, dan Eurotech menyumbangkan implementasi kerangka kerja dan contoh aplikasi untuk digunakan pengembang saat mengintegrasikan dan menguji komponen perpesanan Paho. Pustaka klien Java MQTT berjalan pada banyak variasi Java, termasuk Konfigurasi / Foundation Perangkat Terhubung (CDC), Platform Java, Edisi Standar (J2SE), dan Platform Java, Edisi Perusahaan (J2EE). Implementasi referensi C, bersama dengan klien asli prebuilt untuk sistem operasi Windows dan Linux, memungkinkan MQTT untuk porting ke berbagai perangkat dan platform [27].

Alat Eclipse memfasilitasi perancangan dan pengembangan solusi konektivitas antara perangkat dan aplikasi, sehingga memungkinkan dan mendorong integrasi M2M yang lebih inovatif. Pengguna dapat menulis API mereka sendiri untuk berinteraksi dengan protokol MQTT dalam bahasa pemrograman pilihan mereka pada platform pilihan mereka.

Untuk menyederhanakan penulisan aplikasi klien MQTT, pengembang dapat menggunakan pustaka klien Telemetri WebSphere MQ dan kit pengembangan perangkat lunak pengembangan (SDK). Pustaka klien dan SDK pengembangan dapat diimpor ke lingkungan pengembangan (misalnya, WebSphere Eclipse Platform). Setelah aplikasi yang relevan dikembangkan, aplikasi dan pustaka klien kemudian dapat disebarkan bersama ke sistem yang sesuai. SDK mencakup

perpustakaan klien WebSphere MQ Telemetri C dan Java yang merangkum protokol MQTT V3 untuk sejumlah platform.

5.4. Keuntungan Penggunaan MQTT

Menggunakan protokol MQTT memperluas WebSphere MQ ke sensor kecil dan perangkat telemetri jarak jauh lainnya yang mungkin tidak dapat berkomunikasi dengan sistem pusat atau yang mungkin dicapai hanya melalui penggunaan jaringan khusus yang mahal. Batasan jaringan dapat mencakup bandwidth terbatas, latensi tinggi, batasan volume, koneksi rapuh, atau biaya yang mahal. Masalah perangkat dapat mencakup memori atau kemampuan pemrosesan yang terbatas, atau pembatasan penggunaan perangkat lunak komunikasi pihak ketiga. Selain itu, beberapa perangkat bertenaga baterai, yang memberikan batasan tambahan pada penggunaannya untuk pengiriman pesan telemetri. MQTT dirancang untuk mengatasi keterbatasan dan masalah ini dan mencakup prinsip-prinsip dasar berikut ini:

- Kesederhanaan: Protokol dibuat terbuka sehingga dapat diintegrasikan dengan mudah ke dalam solusi lain.
- Penggunaan model publish / subscribe: Pengirim dan penerima dipisahkan. Dengan demikian, penerbit tidak perlu mengetahui siapa atau apa yang berlangganan pesan dan sebaliknya.
- Pemeliharaan minimal: Fitur, seperti penyimpanan pesan otomatis dan transmisi ulang, meminimalkan kebutuhan administrasi saat berjalan
- Jejak on-the-wire terbatas: Protokol menjaga overhead data seminimal mungkin pada setiap pesan.

- Kesadaran sesi berkelanjutan: Dengan menyadari saat sesi telah dihentikan, protokol dapat mengambil tindakan yang sesuai, sebagian berkat fitur kemauan.
- Pemrosesan pesan lokal: Protokol mengasumsikan bahwa perangkat jauh memiliki kemampuan pemrosesan yang terbatas.
- Ketekunan pesan: Melalui penetapan QoS tertentu, penerbit dapat memastikan pengiriman pesan yang paling penting.
- Agnostik terkait tipe data: Protokol tidak mengharuskan konten pesan dalam format tertentu.

Protokol pesan MQTT dirancang untuk perangkat dalam lingkungan terbatas, seperti sistem tertanam dengan kemampuan pemrosesan dan memori terbatas atau sistem yang terhubung ke jaringan yang tidak dapat diandalkan. Ini menyediakan fitur olahpesan yang kuat yang diperlukan untuk berkomunikasi dengan sistem dan perangkat jarak jauh sambil menghabiskan hanya sebagian kecil dari bandwidth jaringan.

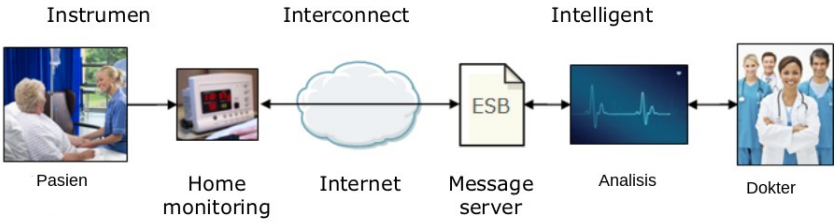
Sebelum membahas, secara umum, jenis-jenis komunikasi yang dapat dibuat MQTT lebih efisien, berikut adalah sekilas skenario di mana MQTT telah berhasil digunakan.

Kesehatan

Sebuah organisasi medis ingin membuat solusi pemantauan alat pacu jantung di rumah. Solusi yang diperlukan untuk menangani aspek perawatan pasien berikut:

- Memantau pasien jantung setelah mereka meninggalkan rumah sakit

- Meningkatkan efisiensi pemeriksaan nanti
- Memenuhi standar pengambilan data industri baru



Gambar 5.4.1: IoT untuk monitoring kesehatan

Perusahaan bekerja dengan IBM untuk membuat solusi (diilustrasikan dalam Gambar 5.4.1) di mana klien MQTT tertanam dalam alat pemantauan rumah yang mengumpulkan diagnostik kapan pun pasien berada di dekat unit dasar. Unit dasar mengirimkan data diagnostik melalui Internet ke server perpesanan pusat, di mana data tersebut diserahkan ke aplikasi yang menganalisis pembacaan dan memberi tahu staf medis jika ada tanda-tanda pasien mungkin mengalami kesulitan.

Solusinya memungkinkan rumah sakit untuk memberikan tingkat perawatan pasien pasca rumah sakit yang lebih tinggi dan diagnosis dini untuk masalah tindak lanjut. Ini juga menghemat uang baik untuk rumah sakit maupun pasiennya, karena lebih sedikit kebutuhan perjalanan oleh salah satu pihak dan karena pasien yang

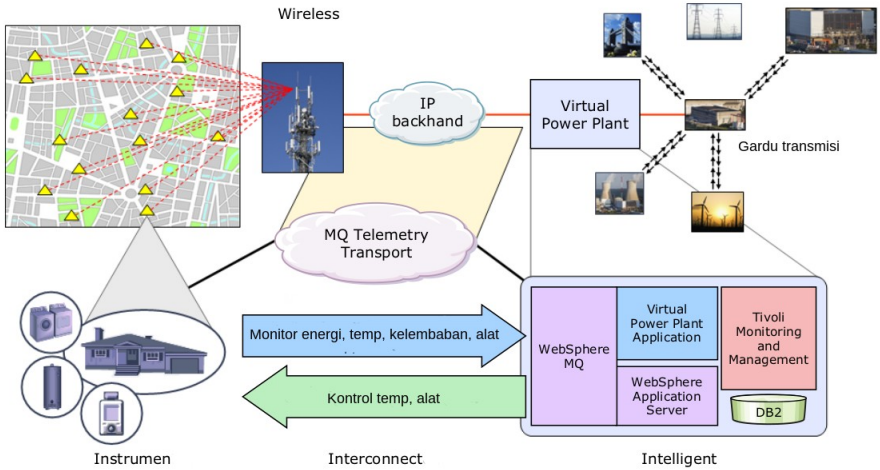
melakukan dengan baik mungkin diizinkan untuk lebih jarang datang untuk pemeriksaan.

Energi dan utilitas

Sebuah perusahaan utilitas dihadapkan pada kenaikan biaya untuk memproduksi listrik dan meningkatnya permintaan listrik dari basis pelanggannya, yang pada umumnya tidak mampu membayar tarif yang terus meningkat. Jadi, daripada langsung membebankan biaya produksi yang kemungkinan besar tidak dapat dibayar oleh pelanggannya, perusahaan tersebut pertama-tama mencari solusi untuk mengurangi permintaan listrik secara keseluruhan dengan menempatkan pengukur pintar di rumah pelanggan untuk mengontrol penggunaan perangkat tertentu yang menghabiskan daya dari jarak jauh. Namun, solusi yang dibutuhkan untuk meminimalkan penggunaan jaringan data yang tersedia, dimana perusahaan membayar sesuai dengan volume data yang dikirimkan.

Solusinya adalah membuat pembangkit listrik virtual (VPP) yang berada di antara sumber pembangkit perusahaan dan pelanggannya. Pengukur pintar di rumah mengumpulkan data penggunaan untuk berbagai peralatan yang digunakan di sana. Kemudian, monitor gerbang rumah, dilengkapi dengan klien MQTT tingkat lanjut, mempublikasikan data penggunaan ke VPP secara berkala melalui jaringan telepon seluler lokal. Seperti yang diilustrasikan pada Gambar 5.4.2, VPP memantau konsumsi energi secara real time, memprediksi kebutuhan konsumsi yang akan datang, dan bila perlu menurunkan permintaan secara keseluruhan dengan mengambil

kendali perangkat yang menggunakan listrik di rumah pelanggan. Ketika instruksi dikirim ke perangkat yang menggunakan listrik di rumah, perintah tersebut dikirim ke kotak gateway rumah menggunakan MQTT.

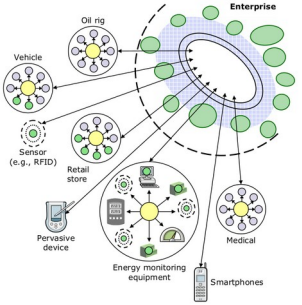


Gambar 5.4.2: MQTT dalam pembangkit listrik

5.5. Mesin ke mesin

Ketika kebanyakan orang memikirkan Internet, mereka memikirkan individu yang menggunakan browser web untuk mengumpulkan informasi dari mesin pencari, untuk berkomunikasi dengan orang lain menggunakan media sosial, atau untuk terhubung ke perangkat tampilan, seperti kamera web. Namun seiring perkembangan teknologi, semakin umum perangkat terhubung satu sama lain. Jenis koneksi ini telah menciptakan kebutuhan akan protokol komunikasi mesin-ke-mesin (M2M) yang efisien. Protokol MQTT sangat ideal untuk

digunakan dalam komunikasi M2M. Ini memungkinkan konektivitas yang melampaui perangkat pintar ke beberapa perangkat dan sensor jarak jauh terkecil, termasuk perangkat dengan kemampuan pemrosesan atau jaringan terbatas. Ekstensi ini menjadikan MQTT sebagai komponen penting dalam jaringan M2M yang dikelola sendiri dan menjadi bagian penting untuk mewujudkan visi Smarter Planet. Selain itu, karena MQTT sangat skalabel, dimungkinkan untuk membuat sistem yang melibatkan ratusan atau bahkan ribuan sensor atau perangkat jarak jauh.



Gambar 5.5.1: MQTT
untuk komunikasi antar mesin

Protokol MQTT untuk Sensor (MQTT-S) memungkinkan penyertaan mesin yang biasanya tidak dapat menggunakan MQTT karena kurangnya kemampuan jaringan TCP / IP. MQTT-S memperluas protokol MQTT ke perangkat sensor dan aktuator berbiaya rendah yang dioperasikan dengan baterai yang ada di jaringan non-TCP / IP. Ini dapat berfungsi di jaringan apa pun yang memungkinkan transfer

data dua arah. MQTT-S ideal untuk jaringan sensor nirkabel (WSN) dari sensor otonom yang didistribusikan secara spasial. WSN menarik karena kesederhanaan, biaya rendah dan kemudahan penerapannya. Klien MQTT-S terhubung ke gateway yang melakukan terjemahan protokol antara MQTT dan MSTT-S. Klien MQTT-S sering digunakan untuk memantau kondisi fisik atau lingkungan seperti suhu, suara, getaran, tekanan, atau gerakan.

5.6. MQTT dan sensor

Sensor

Sensor mengukur atau menentukan, atau merasakan, parameter tertentu dari suatu perangkat atau sistem dan melaporkannya dengan cara yang dapat dimengerti oleh manusia atau perangkat atau sistem lain [28]. Contoh paling sederhana mungkin termometer klinis gaya lama, yang dapat merasakan panas tubuh manusia dan melaporkannya dengan kolom naik atau turun merkuri, atau pengukur tekanan yang melaporkan tekanan udara interior ban melalui gerakan jarum melintasi dial berskala. Sensor modern dapat melaporkan statusnya dengan cara yang lebih canggih, seperti pada panel digital atau, dengan bantuan protokol pengiriman pesan, seperti MQTT, dengan mengirimkan informasinya melalui jaringan data.

Sensor perangkat atau sistem

Satu sensor dapat mengukur parameter tertentu pada perangkat atau sistem tertentu. Atau beberapa sensor dapat

ditempatkan untuk melaporkan beberapa parameter, memberikan gambaran lengkap tentang kondisi operasi perangkat atau sistem saat ini. Pandangan ini bisa menjadi penting bagi supervisor karena memungkinkan dia untuk menentukan, misalnya, apakah segala sesuatu beroperasi dalam batas keamanan. Pertimbangkan sistem untuk memindahkan gas bertekanan di pabrik kimia. Pengawas perlu mengetahui antara lain status pengoperasian kompresor, laju aliran pada inlet dan outlet, suhu gas dan tekanan gas di dalam sistem. Jadi, sangat penting bahwa semua faktor ini diukur dengan sensor yang dikalibrasi dengan benar yang ditempatkan di lokasi yang benar dan mampu melaporkan statusnya dengan andal dan dalam waktu nyata.

Sensor suatu perusahaan

Ketika diterapkan ke seluruh perusahaan, sifat kelompok sensor yang saling terkait menjadi semakin rumit, dan lebih kritis. Ambil contoh pabrik kimia di atas. Mungkin ada kompresor atau mesin lain yang saling bergantung, seperti saat gas terkompresi dimasukkan ke dalam ruang reaksi. Jadi selain sensor yang melaporkan sistem gas bertekanan, sensor lain mengukur aktivitas di dalam ruang reaksi. Setiap sensor individu melaporkan parameter khususnya, tetapi bersama-sama, mereka memberikan status ujung ke ujung dari keseluruhan operasi. Aspek lain dari suatu perusahaan adalah antarmuka antar departemen. Departemen produksi mungkin terutama tertarik pada sensor yang melaporkan kompresor dan ruang reaksi, sedangkan departemen pengiriman sangat tertarik pada seberapa banyak produk yang perlu dikirim ke pelanggan. Jumlah produk yang akan dikirim pada hari

tertentu bergantung pada kinerja kompresor dan ruang reaksi. Jadi, sensor satu per satu itu penting, dan kemampuannya untuk mengkomunikasikan bacaan saat ini juga sangat penting.

Aktuator

Aktuator adalah jenis perangkat khusus yang mengambil tindakan berdasarkan perilaku sistem. Sedangkan sensor melaporkan status parameter tertentu dari suatu sistem, aktuator dapat bertindak untuk mempengaruhi parameter itu atau bagian lain dari sistem. Sebuah analogi yang disederhanakan adalah operasi input-output file. Pekerjaan sebuah sensor dapat dilihat mirip dengan operasi pembacaan file; pekerjaan aktuator dapat dilihat mirip dengan operasi penulisan file. Contoh lain adalah katup aliran yang dipasang ke pipa, yang dapat mengubah laju aliran berdasarkan parameter tertentu yang dirasakan. Aktuator lain mungkin disetel untuk melacak sekumpulan parameter dan kemudian memicu alarm jika ambang tertentu tercapai.

Mengirimkan data sensor dengan MQTT

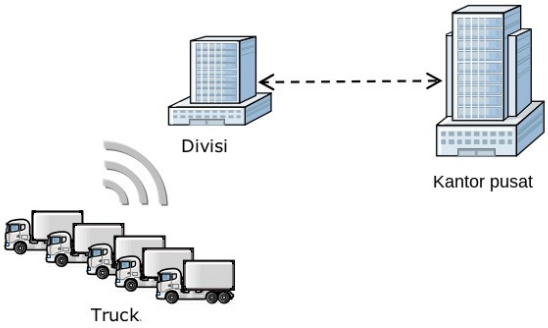
Persyaratan lebih lanjut dari sebuah sensor, seperti mentransmisikan bacaannya melalui jaringan atau menaikkan peringatan, umumnya tidak berada dalam cakupan sensor itu sendiri. Jenis persyaratan ini berada di luar cakupan sensor karena sensor biasanya berupa perangkat keras khusus kecil yang tidak dirancang untuk memuat daya komputasi yang diperlukan untuk fungsi yang lebih canggih. Beberapa sensor dilengkapi dengan kemampuan komunikasi, atau perangkat keras tambahan dapat ditambahkan untuk mengaktifkan

kemampuan ini. Namun, opsi ini umumnya terbatas karena masih tidak dapat menyediakan daya komputasi yang diperlukan untuk protokol transmisi data tradisional. Model pesan publikasi / langganan MQTT dapat membantu memberikan kemampuan untuk mengirimkan data sensor.

5.7. Skenario MQTT

Contoh program dan aplikasi terkait MQTT yang diperlihatkan dalam buku ini didasarkan, jika memungkinkan, pada skenario yang melibatkan Perusahaan Transportasi B, perusahaan logistik fiktif yang mengangkut material ke, dari, dan di antara pelanggannya. Kebutuhan perusahaan untuk secara teratur memantau lokasi kendaraannya, dan status kargo di dalam setiap kendaraan, membuatnya ideal untuk menggambarkan jenis solusi yang dapat disediakan oleh pesan berbasis MQTT [29].

Perusahaan Transportasi B ingin meningkatkan sistem pemantauan truknya. Di bawah arsitektur saat ini (lihat Gambar 5.7.1), setiap truk perusahaan menggunakan teknologi nirkabel untuk menginformasikan kantor pusat divisi secara berkala tentang muatan yang diangkut, posisinya saat ini (lokasi, kecepatan, bahan bakar yang tersedia, dan sebagainya), dan status HVAC-nya sistem (pemanas, ventilasi, dan AC). Setiap divisi menggunakan data ini untuk memantau truknya, dan terkadang pesan yang berisi rute perjalanan baru atau instruksi lain dikirim kembali ke truk. Selain itu, data yang dipilih dikirim ke kantor pusat perusahaan untuk analisis tambahan.



Gambar 5.7.1: Sistem monitoring armada konvensional

Arsitektur perpesanan saat ini tidak memadai untuk memenuhi kebutuhan perusahaan karena alasan berikut:

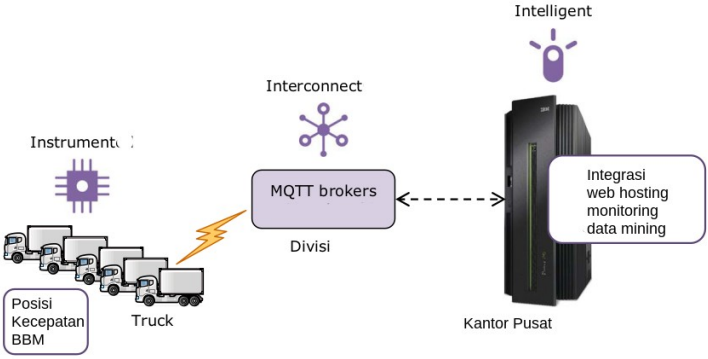
- Tingginya biaya menggunakan beberapa protokol yang berbeda untuk berkomunikasi dengan truk dalam armada, yang seringkali memiliki kemampuan dan persyaratan pengiriman pesan yang berbeda.
- Bandwidth yang tersedia terbatas, yang membuat setiap pesan ke atau dari truk harus sekecil mungkin.
- Gangguan jaringan yang sering terjadi yang menyebabkan pesan hilang atau kacau saat transmisi.
- Kurangnya mekanisme distribusi pesan yang efektif, yang membatasi komunikasi antar truk dan memaksa penggunaan sistem distribusi yang terpisah dan terpusat.
- Terlalu banyak variasi perangkat pesan telemetri internal, masing-masing dengan kemampuan pemrosesan dan sumber daya memori yang berbeda.
- Perlunya kode yang disesuaikan untuk diintegrasikan dengan persyaratan pengiriman pesan dari berbagai

sistem backend yang digunakan untuk pemantauan, analisis, dan pemrosesan.

Untuk menghilangkan kelemahan arsitektur saat ini, Perusahaan Transportasi B berencana untuk menggunakan protokol MQTT dan produk perpesanan terkait untuk menciptakan sistem yang lebih kompatibel secara universal yang menempatkan lebih sedikit permintaan pada jaringan dan dapat bekerja dengan perangkat telemetri yang sudah ada di banyak truknya. . Arsitektur perpesanan baru menggunakan konsep Smarter Planet untuk menyelesaikan banyak tantangan yang dihadapi Perusahaan Transportasi B:

- Informasi Terinstrumen disimpan di tempat yang ada: di truk itu sendiri. Menggunakan protokol MQTT yang ringan namun kuat membantu mengatasi kenyataan bahwa banyak perangkat di truk memiliki memori atau daya pemrosesan yang terbatas, atau terhubung melalui jaringan yang tidak dapat diandalkan.
- Informasi yang saling berhubungan dikirim jika diperlukan: ke kantor divisi perusahaan dan kantor pusat, dan, bila perlu, kembali ke truk. Dengan menghubungkan klien MQTT di truk ke serangkaian pialang pesan yang mendukung MQTT, informasi telemetri penting dari truk dikirim ke tempat yang dibutuhkan dengan cepat dan efisien, dengan biaya infrastruktur yang sangat berkurang dan tantangan arsitektur.
- Informasi Cerdas dianalisis dan keputusan dibuat dalam waktu nyata. Semua pesan dikirim ke divisi perusahaan dan kantor pusat menggunakan format MQTT umum,

memungkinkan sistem backend tunggal digunakan untuk pemantauan, analisis, dan pemrosesan.



Gambar 5.7.2: Sistem monitoring armada menggunakan MQTT

6. Protokol XMPP

6.1. Pengenalan

Extensible Messaging and Presence Protocol (XMPP) adalah teknologi terbuka untuk komunikasi realtime, menggunakan Extensible Markup Language (XML) sebagai format dasar untuk bertukar informasi. Intinya, XMPP menyediakan cara untuk mengirim potongan kecil XML dari satu entitas ke entitas lain secara hampir waktu nyata [30]. XMPP digunakan dalam berbagai aplikasi, dan mungkin juga tepat untuk aplikasi Anda. Untuk membayangkan kemungkinan, akan sangat membantu jika alam semesta XMPP dipecah menjadi tingkat tinggi ke dalam layanan dan aplikasi. Layanan didefinisikan dalam dua spesifikasi utama yang diterbitkan oleh Internet Engineering Task Force (IETF) di http://ietf.org/ (seri "RFC"), dan dalam lusinan spesifikasi ekstensi yang diterbitkan oleh XMPP Standards Foundation di http: //xmpp.org/ (seri "XEP"); aplikasi adalah program perangkat lunak dan skenario penyebaran yang merupakan kepentingan umum bagi individu dan organisasi, meskipun layanan inti memungkinkan Anda untuk membangun banyak jenis aplikasi lainnya juga.

Layanan

Dalam konteks ini, layanan adalah fitur atau fungsi yang dapat digunakan oleh aplikasi apa pun. Implementasi XMPP biasanya menyediakan layanan inti berikut:

Enkripsi saluran

Layanan ini, didefinisikan dalam [RFC 3920], menyediakan enkripsi koneksi antara klien dan server, atau antara dua server. Meskipun enkripsi saluran tidak selalu menarik, itu adalah blok bangunan penting untuk membangun aplikasi yang aman.

Autentikasi

Layanan ini merupakan bagian lain dari fondasi untuk pengembangan aplikasi yang aman. Dalam hal ini, layanan otentikasi memastikan bahwa entitas yang mencoba berkomunikasi melalui jaringan terlebih dahulu diautentikasi oleh server, yang bertindak sebagai semacam penjaga gerbang untuk akses jaringan.

Kehadiran

Layanan ini memungkinkan Anda untuk mengetahui tentang ketersediaan jaringan entitas lain. Pada tingkat paling dasar, layanan kehadiran menjawab pertanyaan, "Apakah entitas online dan tersedia untuk komunikasi, atau offline dan tidak tersedia?" Data kehadiran juga bisa menyertakan informasi yang lebih detail (seperti apakah seseorang sedang rapat). Biasanya, berbagi informasi kehadiran didasarkan pada langganan kehadiran eksplisit antara dua entitas untuk melindungi privasi informasi pengguna.

Daftar kontak

Layanan ini memungkinkan Anda untuk menyimpan daftar kontak, atau daftar nama, pada server XMPP. Penggunaan yang paling umum untuk layanan ini adalah "daftar teman" pesan instan, tetapi entitas apa pun yang

memiliki akun di server dapat menggunakan layanan tersebut untuk mengelola daftar entitas yang dikenal atau tepercaya (mis., dapat digunakan oleh bot).

Perpesanan satu-ke-satu

Layanan ini memungkinkan Anda untuk mengirim pesan ke entitas lain. Penggunaan klasik perpesanan satu-ke-satu adalah IM pribadi, tetapi pesan dapat berupa XML sewenang-wenang, dan dua entitas mana pun di jaringan dapat bertukar pesan — bisa berupa bot, server, komponen, perangkat, layanan web berkemampuan XMPP, atau entitas XMPP lainnya.

Perpesanan multi-pihak

Layanan ini memungkinkan Anda bergabung dengan ruang obrolan virtual untuk bertukar pesan antara banyak peserta, mirip dengan Internet Relay Chat (IRC). Pesan dapat berupa teks biasa, atau dapat berisi ekstensi XML untuk fungsionalitas yang lebih canggih, seperti konfigurasi ruang, pemungutan suara dalam ruang, dan berbagai pesan kontrol sesi.

Notifikasi

Layanan ini memungkinkan Anda untuk membuat pemberitahuan dan mengirimkannya ke banyak pelanggan. Layanan ini mirip dengan perpesanan multipihak, tetapi dioptimalkan untuk pengiriman satu ke banyak dengan langganan eksplisit ke saluran atau topik tertentu (disebut "node").

Penemuan layanan

Layanan ini memungkinkan Anda untuk mengetahui fitur mana yang didukung oleh entitas lain, serta entitas tambahan apa pun yang terkait dengannya (misalnya, ruangan yang diselenggarakan di layanan chat room).

Iklan kemampuan

Layanan ini merupakan ekstensi dari layanan keberadaan yang menyediakan notasi singkatan untuk data penemuan layanan sehingga Anda dapat dengan mudah menyimpan cache fitur yang didukung oleh entitas lain di jaringan.

Formulir data terstruktur

Layanan ini memungkinkan Anda untuk bertukar formulir terstruktur tetapi fleksibel dengan entitas lain, mirip dengan formulir HTML. Ini sering digunakan untuk konfigurasi dan tugas lain di mana Anda perlu mengumpulkan informasi ad-hoc dari entitas lain.

Manajemen alur kerja

Layanan ini memungkinkan Anda untuk terlibat dalam interaksi alur kerja terstruktur dengan entitas lain, dengan dukungan untuk tindakan alur kerja yang khas, seperti berpindah ke tahap berikutnya dari proses bisnis atau menjalankan perintah. Ini sering digunakan bersama dengan formulir data.

Sesi media peer-to-peer

Layanan ini memungkinkan Anda untuk bernegosiasi dan mengelola sesi media dengan entitas lain. Sesi semacam itu

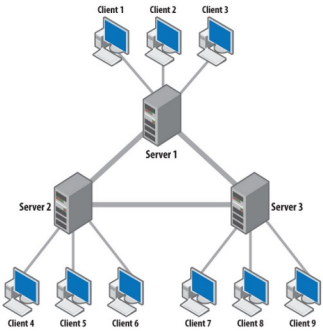
dapat digunakan untuk tujuan obrolan suara, obrolan video, transfer file, dan interaksi waktu nyata lainnya.

6.2. Arsitektur

Semua teknologi Internet yang baik memiliki "arsitektur" —suatu cara di mana berbagai entitas cocok, terhubung, dan berkomunikasi. Misalnya, World Wide Web terdiri dari jutaan server web yang menjalankan perangkat lunak seperti Apache, dan lebih banyak lagi jutaan klien web (browser) yang menjalankan perangkat lunak seperti Firefox, semua menggunakan protokol standar dan format data seperti HTTP dan HTML. Sebagai contoh lain, infrastruktur email terdiri dari jutaan server email yang menjalankan perangkat lunak seperti Postfix, dan jutaan klien email lainnya yang menjalankan perangkat lunak seperti Thunderbird, semuanya menggunakan protokol standar seperti SMTP, POP, dan IMAP. Demikian pula, infrastruktur Internet untuk pesan instan, kehadiran, dan bentuk komunikasi waktu nyata lainnya semakin banyak terdiri dari ratusan ribu server Jabber yang menjalankan perangkat lunak seperti ejabberd dan Openfire, dan jutaan klien Jabber yang menjalankan perangkat lunak seperti Adium, Gajim, Pidgin, dan Psi , semua menggunakan protokol standar yang kami sebut XMPP.

Teknologi XMPP menggunakan arsitektur klien-server terdesentralisasi yang mirip dengan arsitektur yang digunakan untuk World Wide Web dan jaringan email [30]. Diagram pada Gambar 6.2.1 adalah representasi sederhana yang menunjukkan tiga server, masing-masing dengan tiga klien. Keindahan menggunakan arsitektur server-klien terdesentralisasi adalah

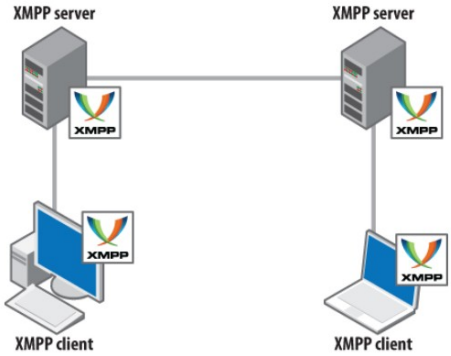
memungkinkan pemisahan perhatian yang cerdas (pengembang klien dapat fokus pada pengalaman pengguna, dan pengembang server dapat fokus pada keandalan dan skalabilitas), jauh lebih mudah bagi organisasi untuk mengelola daripada murni teknologi peer-to-peer, ini cukup kuat karena sistem penuh tidak memiliki satu titik kegagalan (siapa pun dapat menjalankan server XMPP mereka sendiri dan dengan demikian bergabung dengan jaringan), dan server dapat memberlakukan kebijakan keamanan penting seperti otentikasi pengguna , enkripsi saluran, dan pencegahan spoofing alamat. Terakhir, komunitas XMPP selalu bekerja untuk menjaga klien tetap sederhana dan mendorong kerumitan sebanyak mungkin ke server, yang selanjutnya memungkinkan adopsi teknologi secara luas. Namun, ada beberapa perbedaan arsitektural penting antara Web, email, dan Jabber.



Gambar 6.2.1: Arsitektur client-server menggunakan XMPP

Saat Anda mengunjungi situs web, browser Anda terhubung ke server web, tetapi server web biasanya tidak terhubung satu sama lain untuk menyelesaikan transaksi. Sebaliknya, HTML halaman web mungkin merujuk ke server web lain (mis., Untuk memuat gambar atau skrip), dan browser Anda membuka sesi dengan server web tersebut untuk memuat halaman penuh. Jadi, Web biasanya tidak melibatkan koneksi antar domain (sering disebut federasi, dan ditunjukkan pada Gambar 6.2.1 oleh garis ganda). Ketika Anda mengirim email ke salah satu kontak Anda di domain yang berbeda, klien email Anda terhubung ke server email "rumah" Anda, yang kemudian berusaha untuk merutekan pesan ke kontak Anda. Jadi, tidak seperti Web, sistem email terdiri dari jaringan server federasi. Namun, pesan Anda mungkin dirutekan melalui beberapa server email perantara sebelum mencapai tujuan akhirnya. Dengan demikian, jaringan email menggunakan banyak lompatan antar server untuk mengirim pesan.

Seperti email, tetapi tidak seperti Web, sistem XMPP melibatkan banyak koneksi antar-domain. Namun, ketika Anda mengirim pesan XMPP ke salah satu kontak Anda di domain yang berbeda, klien Anda terhubung ke server "rumah" Anda, yang kemudian terhubung langsung ke server kontak Anda tanpa lompatan perantara (lihat Gambar 2-4). Model federasi langsung ini memiliki perbedaan penting dari model federasi tidak langsung yang digunakan dalam email (khususnya, model ini membantu mencegah spoofing alamat dan bentuk spam tertentu).



Gambar 6.2.2: XMPP menggunakan beberapa server

Meskipun klien dan server adalah entitas fundamental pada jaringan XMPP, entitas lain juga berperan. Klien otomatis yang disebut bot menyediakan berbagai layanan komunikasi, termasuk bantuan di ruang obrolan dan antarmuka yang ramah manusia ke layanan non-XMPP seperti aplikasi jejaring sosial. Selain itu, sebagian besar server XMPP dibuat dengan cara modular yang memungkinkan administrator menambahkan layanan khusus atau komponen server, seperti ruang obrolan multi-pengguna, topik terbitkan-langganan, penengah permainan, dan sejenisnya.

Alamat

Karena komunikasi XMPP terjadi di jaringan, setiap entitas XMPP membutuhkan alamat, yang disebut JabberID (JID). XMPP biasanya mengandalkan Domain Name System (DNS) untuk menyediakan struktur yang mendasari pengalamatan, daripada menggunakan alamat Internet Protocol (IP) mentah. Lagipula, jauh lebih mudah untuk

mengingat bahwa ada layanan XMPP yang berjalan di jabber.org daripada mengingat 208.68.163.220. Demikian pula, JabberID untuk pengguna terlihat seperti alamat email (mis., Stpeter@jabber.org) karena format pengguna@domain.tld sudah dikenal orang; Selain itu, format ini menggunakan infrastruktur DNS lengkap sebagai ruang alamatnya, tidak seperti sistem IM lama yang menggunakan nomor atau nama tanpa pengenal domain.

Domain

Setiap JabberID berisi bagian domain, yang biasanya dipetakan ke nama domain yang memenuhi syarat (FQDN) [31]. Saat Anda menginstal perangkat lunak server XMPP favorit, Anda memilih nama domain untuk penerapan, seperti jabber.org atau gmail.com. Menggunakan catatan label layanan DNS, nama domain Anda dipetakan ke satu atau beberapa mesin tertentu, seperti hermes.jabber.org atau talk1.l.google.com. Nama-nama mesin tersebut pada gilirannya memetakan ke alamat IP tertentu, seperti 208.68.163.220 atau 72.14.253.125. (Kami membahas skenario penerapan lebih lanjut dalam lampiran.) Namun, untuk tujuan pengalamatan di jaringan, yang perlu kita perhatikan adalah nama domain itu sendiri (misalnya, jabber.org atau gmail.com), bukan yang lebih rendah- nama mesin tingkat dan alamat IP. Terakhir, untuk karakter ASCII, bagian domain JID tidak membedakan huruf besar / kecil (sehingga JABBER.ORG sama dengan jabber.org); seperti yang kami jelaskan nanti, aturan untuk karakter non-ASCII sedikit lebih kompleks.

Pengguna

Saat Anda membuat akun di layanan XMPP seperti jabber.org, Anda memilih JabberID yang berfungsi sebagai identitas virtual Anda di jaringan. Atau, JabberID Anda mungkin ditetapkan untuk Anda secara otomatis. JabberID Anda terlihat seperti alamat email (mis., Stpeter@jabber.org). Bergantung pada kebijakan penerapan, bahkan mungkin sama dengan alamat email Anda di layanan atau perusahaan (mis., Alamat Google Talk Anda di jaringan XMPP terlihat sama dengan alamat Gmail Anda di jaringan email). Sedangkan untuk bagian domain JabberID, bagian nama pengguna JID tidak membedakan huruf besar / kecil untuk karakter ASCII (sehingga StPeter@jabber.org sama dengan stpeter@jabber.org). Pengembang XMPP biasanya menyebut alamat dalam bentuk user@domain.tld sebagai JID kosong.

Sumber daya

Saat Anda menyambungkan klien ke server XMPP, Anda memilih (atau server menetapkan kepada Anda) pengenal sumber daya untuk koneksi tertentu itu. Sumber daya ini digunakan untuk merutekan lalu lintas ke koneksi itu alih-alih koneksi lain yang mungkin Anda buka saat ini. Sumber daya ditambahkan ke akhir alamat akun Anda, seperti stpeter@jabber.org/roundabout atau remko@eltramo.be/home. Ini memungkinkan seseorang untuk menanyakan atau bertukar pesan dengan perangkat tertentu yang dikaitkan dengan akun Anda; ini juga berarti bahwa setiap perangkat adalah "titik kehadiran" yang terpisah, dengan status ketersediaan, kemampuan yang berbeda, dll. Sumber daya sering kali berupa

nama komputer Anda, lokasi Anda, atau perangkat lunak klien yang Anda gunakan, tetapi dapat berupa string apa pun (termasuk spasi dan karakter khusus lainnya). Berlawanan dengan bagian lain dari JID, bagian sumber daya peka huruf besar / kecil (mis., Remko@eltramo.be/home berbeda dari remko@eltramo.be/Home). Pengembang XMPP biasanya menyebut alamat dalam bentuk user@domain.tld/resource JID lengkap.

6.3. Dasar komunikasi

"Bait" XML ini terdengar agak puitis, tapi apa artinya dalam praktiknya? Dalam XMPP, sebuah bait dapat dianggap sebagai unit dasar komunikasi, mirip dengan paket atau pesan dalam protokol jaringan lain (istilah ini disarankan oleh Lisa Dusseault, yang memimpin Kelompok Kerja XMPP IETF bersama dengan Pete Resnick). Beberapa faktor menentukan arti dari sebuah bait:

- Nama elemen bait, yaitu pesan, keberadaan, atau iq. Setiap jenis bait dirutekan secara berbeda oleh server dan ditangani secara berbeda oleh klien.
- Nilai atribut type, yang bervariasi tergantung pada jenis bait yang dimaksud. Nilai ini semakin membedakan bagaimana setiap jenis bait diproses oleh penerimanya.
- Elemen anak, yang menentukan muatan stanza. Payload mungkin ditampilkan kepada pengguna atau diproses secara otomatis seperti yang ditentukan oleh spesifikasi yang menentukan namespace dari payload.

Bagian berikut memberikan pengenalan singkat tentang faktor-faktor ini, dan kami akan menjelajahinya di seluruh buku ini

saat kami mengungkap arti dari berbagai jenis bait, nilai atribut tipe, dan definisi muatan.

Pesan

Stanza XMPP <message /> adalah metode "push" dasar untuk mendapatkan informasi dari satu tempat ke tempat lain. Karena pesan biasanya tidak dikenali, pesan itu semacam mekanisme "tembak dan lupakan" untuk mendapatkan informasi dengan cepat dari satu tempat ke tempat lain. Pesan digunakan untuk IM, obrolan grup, tanda dan pemberitahuan, dan aplikasi sejenis lainnya. Bait pesan hadir dalam lima bentuk, dibedakan menurut atribut type:

normal

Jenis pesan normal paling mirip dengan pesan email, karena merupakan pesan tunggal yang responsnya mungkin atau mungkin tidak akan datang.

Chat

Pesan jenis chat dipertukarkan dalam "sesi" waktu nyata antara dua entitas, seperti obrolan pesan instan antara dua teman.

groupchat

Pesan jenis groupchat dipertukarkan di chat room multi-pengguna, mirip dengan Internet Relay Chat.

headline

Pesan dengan jenis headline digunakan untuk mengirim peringatan dan pemberitahuan, dan tanggapan tidak

diharapkan sama sekali (klien yang menerima headline tidak boleh memungkinkan pengguna untuk membalas)

error
Jika terjadi kesalahan dalam kaitannya dengan pesan yang dikirim sebelumnya, entitas yang mendeteksi masalah akan mengembalikan pesan jenis kesalahan.

Selain atribut type, bait pesan berisi alamat ke dan dari, dan dapat berisi atribut id untuk tujuan pelacakan (kita membahas ID secara lebih rinci dalam kaitannya dengan bait IQ, di mana mereka digunakan secara lebih luas). Cukup wajar, alamat ke adalah JabberID penerima yang dituju, dan alamat asal adalah JabberID pengirim. Alamat dari tidak diberikan oleh klien pengirim, tetapi dicap oleh server pengirim untuk menghindari spoofing alamat.

Ekstensibilitas

Sebuah stanza XML dapat berisi sejumlah elemen turunan lainnya, termasuk badan pesan berformat XHTML, penunjuk ke URL, pemberitahuan RSS atau Atom, formulir yang harus diisi (atau formulir yang dikirimkan), data XML-RPC atau SOAP untuk layanan web, geografis lokasi, dan berbagai muatan lainnya. ("X" dalam XML dan XMPP adalah singkatan dari "extensible," jadi jenis payload hanya dibatasi oleh imajinasi Anda!) Karena XMPP adalah teknologi XML murni, XMPP menggunakan namespace XML secara ekstensif sebagai cara untuk "menjangkau" muatan stanza . Anda dapat menganggap namespace ini sebagai padanan XML dari paket dan namespace dalam pemrograman. Sejauh ini, komunitas

pengembang XMPP telah menetapkan lusinan ekstensi ke lapisan stanza XMPP inti. Ekstensi ini paling sering dipublikasikan oleh XMPP Standards Foundation di http: // xmpp .org /, tetapi Anda juga dapat menentukan ekstensi pribadi Anda sendiri untuk fitur kustom. Ekstensi cocok pada nama elemen dan namespace. Pada masa-masa awal proyek open source Jabber, pengembang menggunakan elemen <x /> untuk ekstensi yang akan ditempatkan dalam pesan atau pesan kehadiran dan elemen <query /> untuk ekstensi yang akan ditempatkan dalam stanza IQ. Anda akan melihat contoh ekstensi awal ini di buku ini (mis., <Query xmlns = "jabber: iq: roster" />), tetapi penting untuk disadari bahwa penggunaan ini hanya konvensional; ekstensi yang dikembangkan kemudian tidak mengikuti praktik yang sama.

Asynchronicity

Di XMPP, Anda bertukar stanza secara asinkron dengan entitas lain di jaringan. Model ini berbeda dari HTTP, di mana klien Anda mengirim permintaan ke server dan kemudian menunggu balasan sebelum membuat permintaan lain. Sebaliknya, di XMPP klien Anda dapat "menyalurkan" permintaan ke server Anda atau ke entitas lain dan kemudian menerima balasan saat mereka kembali. Peristiwa tertentu juga dapat memicu informasi yang didorong ke klien Anda (mis., Ketika salah satu perangkat Anda menambahkan item ke daftar Anda, item tersebut didorong ke semua perangkat Anda sehingga tetap sinkron). Pendekatan yang dipicu oleh peristiwa yang berlangsung cepat ini dapat membingungkan pada awalnya bagi pengembang yang lebih terbiasa dengan pengembangan web tradisional, tetapi memiliki sejumlah

keunggulan, seperti pemberitahuan waktu nyata dan kemampuan untuk mengatasi kebutuhan untuk terus melakukan polling untuk informasi terbaru.

Penanganan Error

Tidak seperti beberapa teknologi komunikasi, XMPP tidak mengakui setiap paket atau pesan yang dikirim melalui kabel. Biasanya, Anda berasumsi bahwa pesan atau stanza kehadiran telah terkirim jika Anda tidak melihat error. Bait IQ lebih terstruktur: Anda harus selalu menerima hasil IQ atau kesalahan IQ dalam menanggapi IQ-get atau IQ-set. Error dilaporkan dengan menyetel atribut type stanza ke nilai error, bersama dengan elemen turunan <error /> yang memenuhi syarat oleh urn: ietf: params: xml: ns: xmpp-stanzas namespace. Atribut type dari elemen <error /> adalah salah satu dari auth, cancel, lanjutkan, modifikasi, atau tunggu (nilainya mengisyaratkan bagaimana menangani kesalahan). Namun, arti utama kesalahan ditentukan oleh elemen turunan asli, misalnya, <item-not-found /> atau <Forbidden />. Kondisi kesalahan XMPP umumnya mengikuti model kesalahan dari HTTP dan SMTP, kecuali bahwa keduanya terstruktur bukan sebagai kode numerik, seperti 404, tetapi sebagai elemen XML (coba tebak!). Daftar lengkap kondisi kesalahan stanza dapat ditemukan di [RFC 3920].

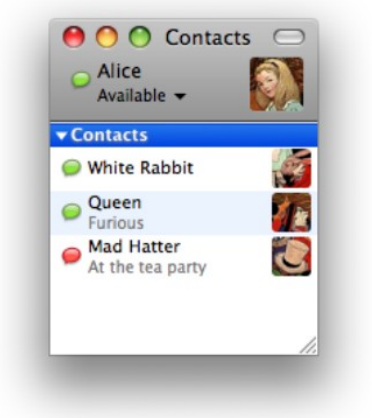
6.4. Perangkat XMPP

Presence

Bayangkan Anda ingin menghubungi teman atau kolega. Di masa lalu, Anda mungkin telah mengirim surat kepada

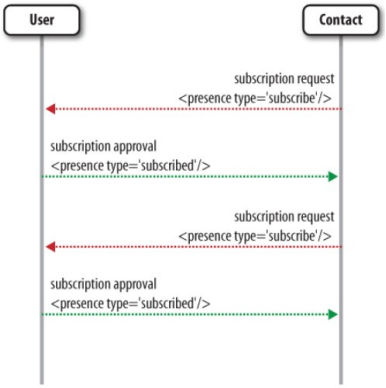
orang tersebut (Anda tahu, salah satu dari potongan kertas yang dikirim ke rumah atau kantor Anda), dan kemudian menunggu balasan. Atau Anda mungkin telah menelepon orang tersebut dan berharap dia ada (jika tidak, Anda akan meninggalkan pesan dengan seseorang atau sistem otomatis, mungkin memainkan "tag telepon" selama beberapa hari). Atau, baru-baru ini, Anda mungkin telah mengirim email, mungkin menerima balasan dalam 10 menit atau lebih, tetapi mungkin beberapa hari kemudian.

Di XMPP, Anda dapat mengetahui saat kontak Anda online dan tersedia untuk komunikasi, menggunakan teknologi yang disebut kehadiran. Jadi, alih-alih menunggu dan bertanya-tanya, atau hanya beruntung, klien Jabber Anda akan menunjukkan ketersediaan jaringan kontak Anda, biasanya dengan indikator seperti ikon bola lampu (dengan teori bahwa jika ada orang di rumah, lampu akan menyala). Gambar 6.4.1 menunjukkan contoh daftar kontak dengan kehadiran aktif di klien IM. Namun, kehadirannya tidak terbatas pada ikon-ikon kecil yang cantik; ini memungkinkan Anda menyelesaikan pekerjaan nyata. Dalam bab ini, kami mempelajari lebih dalam tentang keberadaan, dan menjelajahi bagaimana Anda dapat menggunakannya untuk membangun aplikasi yang lebih cerdas dan lebih interaktif.



Gambar 6.4.1: Contoh user dalam aplikasi XMPP

Keputusan kepercayaan atau akses di balik kehadiran terjadi secara alami dalam sistem IM, karena orang yang Anda setujui secara otomatis ditambahkan ke daftar kontak Anda (disebut daftar di XMPP), yang biasanya merupakan "basis awal" untuk pesan instan atau aplikasi komunikasi waktu nyata. . Selain itu, akses kehadiran biasanya dua arah: Anda mengizinkan kontak untuk melihat kehadiran Anda, dan kontak Anda memungkinkan Anda untuk melihat kehadirannya. Ini terjadi melalui langganan "jabat tangan", seperti yang ditunjukkan pada Gambar 6.4.2. Jika jabat tangan berhasil diselesaikan, hasilnya adalah langganan kehadiran dua arah antara kedua pihak. (Server XMPP juga menambahkan kontak ke daftar pengguna dan menambahkan pengguna ke daftar kontak selama proses ini, ditambah mengelola mesin status status langganan, tetapi kami tidak perlu mengkhawatirkan detail tersebut di sini; lihat [RFC 3921] untuk penjelasan lengkap.)



Gambar 6.4.2: Handshake dalam XMPP

Status Ketersediaan

Sejauh ini, contoh stanza pemberitahuan kehadiran kami sangat sederhana (tersedia atau tidak tersedia). Namun bait kehadiran dapat berisi lebih banyak informasi daripada ketersediaan jaringan on-off dasar. Ada dua elemen kehadiran utama yang mengekspresikan informasi lebih detail: elemen <show /> dan elemen <status />.

Elemen <show /> dibatasi hingga empat nilai yang telah ditetapkan, yang memberikan wawasan tentang ketersediaan pengguna manusia dan minat dalam komunikasi (ini tidak ditampilkan secara langsung kepada pengguna akhir tetapi digunakan untuk memberikan petunjuk ketersediaan di antarmuka pengguna):

chat

Mengumumkan bahwa Anda tersedia untuk, dan secara aktif mencari, percakapan (mungkin Anda merasa sangat ramah).

away
Menunjukkan bahwa Anda pergi dari klien IM, komputer, atau perangkat Anda untuk waktu yang singkat; keadaan ini sering kali dipicu tanpa campur tangan manusia melalui fitur yang disebut auto-away, yang umumnya ditemukan di banyak klien IM.

xa
Menunjukkan bahwa Anda pergi untuk jangka waktu yang lebih lama (xa adalah singkatan dari "eXtended Away"); Klien IM Anda juga dapat secara otomatis menghasilkan status ini.

dnd
Mengumumkan bahwa Anda sedang sibuk dan tidak ingin diganggu sekarang (dnd adalah singkatan dari "jangan ganggu").

Selain itu, elemen <status /> memungkinkan pengguna menentukan beberapa teks bentuk bebas yang dapat dibaca manusia yang mendeskripsikan ketersediaan pengguna secara lebih detail. Misalnya, pengguna dapat menggabungkan nilai <show /> jauh dengan nilai <status /> "Minum teh dengan Kelinci Putih", atau nilai <show /> dnd dengan nilai <status /> " Di tenggat waktu. "

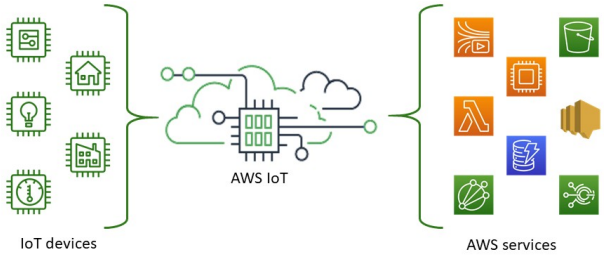
Elemen <show /> dan <status /> tidak terbatas pada pengguna manusia. Mereka juga dapat digunakan oleh proses

otomatis; sebagai contoh, unit tertentu dalam sebuah peternakan komputasi dapat dnd jika tidak dapat menerima pekerjaan baru saat ini. Namun, untuk penanganan kehadiran yang lebih canggih dari entitas otomatis, mungkin akan lebih baik jika Anda menetapkan ekstensi kehadiran kustom daripada membebani nilai teks yang ada.

7. Platform Layanan IoT

7.1. AWS IoT

AWS IoT menyediakan layanan cloud yang menghubungkan perangkat IoT Anda ke perangkat lain dan layanan cloud AWS. AWS IoT menyediakan perangkat lunak perangkat yang dapat membantu Anda mengintegrasikan perangkat IoT Anda ke dalam solusi berbasis AWS IoT. Jika perangkat Anda dapat terhubung ke AWS IoT, AWS IoT dapat menghubungkannya ke layanan cloud yang disediakan AWS [32].



Gambar 7.1.1: AWS IoT

AWS IoT memungkinkan Anda memilih teknologi yang paling sesuai dan terkini untuk solusi Anda. AWS IoT dapat memberikan dukungan untuk perangkat Anda yang kompatibel untuk memfasilitasi pengembangan dan integrasi perangkat Anda dengan AWS IoT. Untuk membantu Anda mengelola dan mendukung perangkat IoT Anda di lapangan, komunikasi AWS IoT mendukung MQTT (Message Queuing and Telemetry Transport) dan HTTPS (Hypertext Transfer Protocol - Secure).

Jika Anda tidak memerlukan fitur AWS IoT seperti komunikasi perangkat, aturan, atau pekerjaan, lihat Perpesanan AWS untuk informasi tentang layanan perpesanan AWS lainnya yang mungkin lebih sesuai dengan kebutuhan Anda.

Baik Anda baru mengenal IoT atau memiliki pengalaman bertahun-tahun, pastikan untuk meninjau Cara kerja AWS IoT. Topik ini membantu Anda memahami konsep dan istilah AWS IoT untuk membantu Anda memulai dengan AWS IoT secara lebih efektif.

Cara memulai dengan AWS IoT

- Lihat ke dalam AWS IoT dan komponennya di Cara kerja AWS IoT.
- Pelajari lebih lanjut tentang AWS IoT dari koleksi materi dan video pelatihan kami. Topik ini juga mencakup daftar layanan yang dapat dihubungkan ke AWS IoT, tautan media sosial, dan tautan ke spesifikasi protokol komunikasi.
- Hubungkan perangkat pertama Anda ke AWS IoT dalam Memulai dengan AWS IoT Core.
- Kembangkan solusi IoT Anda dengan Menghubungkan ke AWS IoT Core dan menjelajahi Tutorial AWS IoT.

Bagaimana perangkat dan aplikasi Anda mengakses AWS IoT
AWS IoT menyediakan antarmuka berikut untuk AWS IoT Tutorials:

- AWS IoT Device SDKs — Buat aplikasi di perangkat Anda yang mengirim pesan ke dan menerima pesan dari AWS IoT. Untuk informasi lebih lanjut, lihat AWS IoT Device and Mobile SDK.
- AWS Command Line Interface (AWS CLI) —Jalankan perintah untuk AWS IoT di Windows, MacOS, dan Linux. Perintah-perintah ini memungkinkan Anda untuk membuat dan mengelola objek benda, sertifikat, aturan, pekerjaan, dan kebijakan. Untuk memulai, lihat Panduan Pengguna Antarmuka Baris Perintah AWS. Untuk informasi selengkapnya tentang perintah untuk AWS IoT, lihat iot di Referensi Perintah AWS CLI.
- AWS IoT API — Buat aplikasi IoT Anda menggunakan permintaan HTTP atau HTTPS. Tindakan API ini memungkinkan Anda untuk membuat dan mengelola objek, sertifikat, aturan, dan kebijakan secara terprogram. Untuk informasi selengkapnya tentang tindakan API untuk AWS IoT, lihat Tindakan di Referensi API IoT AWS.
- AWS SDK — Buat aplikasi IoT Anda menggunakan API khusus bahasa. SDK ini menggabungkan API HTTP / HTTPS dan memungkinkan Anda memprogram dalam salah satu bahasa yang didukung. Untuk informasi lebih lanjut, lihat AWS SDK dan Alat.

Anda juga dapat mengakses AWS IoT melalui konsol AWS IoT, yang menyediakan antarmuka pengguna grafis (GUI) yang melaluinya Anda dapat mengonfigurasi dan mengelola objek, sertifikat, aturan, pekerjaan, kebijakan, dan elemen lain dari solusi IoT Anda.

Aturan dalam AWS IoT

Tutorial berikut menunjukkan kepada Anda cara membuat dan menguji aturan AWS IoT. Sebelum Anda memulai, pastikan untuk menyelesaikan Tutorial Memulai AWS IoT. Ini menunjukkan kepada Anda cara membuat akun AWS dan mendaftarkan perangkat di AWS IoT, yang merupakan prasyarat untuk tutorial ini.

Skenario dalam tutorial ini adalah rumah kaca dengan deretan tanaman. Setiap tanaman memiliki sensor kelembaban. Pada interval yang telah ditentukan, sensor kelembaban mengirimkan datanya ke AWS IoT. Mesin aturan AWS IoT menerima data ini dan menulisnya ke tabel DynamoDB. Anda membuat aturan untuk menulis data ke DynamoDB dan meniru sensor menggunakan klien AWS IoT MQTT.

Aturan AWS IoT terdiri dari pernyataan SQL SELECT, filter topik, dan tindakan aturan. Perangkat mengirimkan informasi ke AWS IoT dengan menerbitkan pesan ke topik MQTT. Pernyataan SQL SELECT memungkinkan Anda mengekstrak data dari pesan MQTT yang masuk. Filter topik dari aturan AWS IoT menentukan satu atau beberapa topik MQTT. Aturan dimulai saat pesan MQTT yang cocok dengan filter topik diterima pada suatu topik. Tindakan aturan memungkinkan Anda mengambil informasi yang diekstrak dari pesan MQTT dan mengirimkannya ke layanan AWS lain. Tindakan aturan ditentukan untuk layanan AWS seperti Amazon DynamoDB, AWS Lambda, Amazon SNS, dan Amazon S3. Dengan menggunakan aturan Lambda, Anda

dapat memanggil layanan web AWS atau pihak ketiga lainnya. Untuk daftar lengkap tindakan aturan, lihat AWS IoT Rule Actions.

Dalam tutorial ini, kami berasumsi bahwa Anda menggunakan klien AWS IoT MQTT dan Anda menggunakan my / greenhouse sebagai filter topik dalam aturan.

Anda juga dapat menggunakan perangkat Anda sendiri, tetapi Anda harus mengetahui topik MQTT mana yang diterbitkan perangkat Anda sehingga Anda dapat menetapkannya sebagai filter topik dalam aturan. Untuk informasi selengkapnya, lihat Aturan AWS IoT.

Mengelola perangkat dengan AWS IoT

WS IoT menyediakan registri yang membantu Anda mengelola berbagai hal. Sesuatu adalah representasi dari perangkat tertentu atau entitas logis. Ini bisa berupa perangkat fisik atau sensor (misalnya, bola lampu atau sakelar di dinding). Itu juga bisa menjadi entitas logis seperti contoh aplikasi atau entitas fisik yang tidak terhubung ke AWS IoT tetapi terkait dengan perangkat lain yang terhubung (misalnya, mobil yang memiliki sensor mesin atau panel kontrol).

Hal-hal diidentifikasi dengan sebuah nama. Things juga dapat memiliki atribut, yaitu pasangan nama-nilai yang dapat Anda gunakan untuk menyimpan informasi tentang benda tersebut, seperti nomor seri atau pabrikannya.

Kasus penggunaan perangkat pada umumnya melibatkan penggunaan nama hal sebagai ID klien MQTT default. Meskipun kami tidak menerapkan pemetaan antara nama registri sesuatu dan penggunaan ID klien MQTT, sertifikat, atau status bayangannya, kami menyarankan Anda memilih nama benda dan menggunakannya sebagai ID klien MQTT untuk registri dan layanan Device Shadow. . Ini memberikan pengaturan dan kenyamanan pada armada IoT Anda tanpa menghilangkan fleksibilitas bayangan atau model sertifikat perangkat yang mendasarinya.

Anda tidak perlu membuat apa pun di registri untuk menghubungkan perangkat ke AWS IoT. Menambahkan sesuatu ke registri memungkinkan Anda mengelola dan mencari perangkat dengan lebih mudah.

Tipe alat

Jenis benda memungkinkan Anda untuk menyimpan informasi deskripsi dan konfigurasi yang umum untuk semua hal yang terkait dengan jenis hal yang sama. Ini menyederhanakan pengelolaan hal-hal di registri. Misalnya, Anda dapat menentukan jenis benda LightBulb. Semua hal yang terkait dengan jenis benda LightBulb berbagi satu set atribut: nomor seri, pabrikan, dan watt. Saat Anda membuat sesuatu berjenis LightBulb (atau mengubah jenis benda yang ada menjadi LightBulb) Anda dapat menentukan nilai untuk setiap atribut yang ditentukan dalam jenis benda LightBulb.

Meskipun jenis benda bersifat opsional, penggunaannya membuatnya lebih mudah untuk menemukan sesuatu.

- Benda dengan jenis benda dapat memiliki hingga 50 atribut.
- Benda tanpa jenis benda dapat memiliki hingga tiga atribut.
- Suatu hal hanya dapat dikaitkan dengan satu jenis hal.
- Tidak ada batasan jumlah jenis hal yang dapat Anda buat di akun Anda.

Jenis benda tidak dapat diubah. Anda tidak dapat mengubah nama jenis hal setelah itu dibuat. Anda dapat menghentikan jenis hal kapan saja untuk mencegah hal-hal baru dikaitkan dengannya. Anda juga dapat menghapus jenis hal yang tidak terkait dengannya.

7.2. Microsoft Azure IoT

Apa itu Azure Internet of Things (IoT)?

Azure Internet of Things (IoT) adalah kumpulan layanan cloud yang dikelola Microsoft yang menghubungkan, memantau, dan mengontrol miliaran aset IoT. Dalam istilah yang lebih sederhana, solusi IoT terdiri dari satu atau lebih perangkat IoT yang berkomunikasi dengan satu atau lebih layanan back-end yang dihosting di cloud [32].

Peralatan IoT

Perangkat IoT biasanya terdiri dari papan sirkuit dengan sensor terpasang yang menggunakan WiFi untuk terhubung ke internet. Sebagai contoh:

- Sensor tekanan pada pompa oli jarak jauh.
- Sensor suhu dan kelembaban di unit AC.

- Akselerometer di lift.
- Sensor kehadiran di sebuah ruangan.

Ada berbagai macam perangkat yang tersedia dari berbagai produsen untuk membangun solusi Anda. Untuk daftar perangkat yang disertifikasi untuk bekerja dengan Azure IoT Hub, lihat Katalog perangkat Azure Certified for IoT. Untuk pembuatan prototipe, Anda dapat menggunakan perangkat seperti MXChip IoT DevKit atau Raspberry Pi. Devkit memiliki sensor bawaan untuk suhu, tekanan, kelembapan, dan giroskop, akselerometer, dan magnetometer. Raspberry Pi memungkinkan Anda memasang berbagai jenis sensor.

Microsoft menyediakan SDK Perangkat sumber terbuka yang dapat Anda gunakan untuk membangun aplikasi yang berjalan di perangkat Anda. SDK ini menyederhanakan dan mempercepat pengembangan solusi IoT Anda.

Komunikasi

Biasanya, perangkat IoT mengirim telemetri dari sensor ke layanan back-end di cloud. Namun, jenis komunikasi lain dimungkinkan seperti layanan back-end yang mengirimkan perintah ke perangkat Anda. Berikut adalah beberapa contoh komunikasi perangkat-ke-awan dan awan-ke-perangkat:

- Truk pendingin bergerak mengirimkan suhu setiap 5 menit ke IoT Hub.
- Layanan back-end mengirimkan perintah ke perangkat untuk mengubah frekuensi pengiriman telemetri untuk membantu mendiagnosis masalah.
- Perangkat mengirimkan peringatan berdasarkan nilai yang dibaca dari sensornya. Misalnya, perangkat yang

memantau reaktor batch di pabrik kimia, mengirimkan peringatan saat suhu melebihi nilai tertentu.

- Perangkat Anda mengirimkan informasi untuk ditampilkan di dasbor untuk dilihat oleh operator manusia. Misalnya, ruang kendali di kilang dapat menunjukkan suhu, tekanan, dan volume aliran di setiap pipa, memungkinkan operator memantau fasilitas tersebut.

IoT Device SDK dan IoT Hub mendukung protokol komunikasi umum seperti HTTP, MQTT, dan AMQP. Perangkat IoT memiliki karakteristik yang berbeda jika dibandingkan dengan klien lain seperti browser dan aplikasi seluler. SDK perangkat membantu Anda mengatasi tantangan dalam menyambungkan perangkat dengan aman dan andal ke layanan back-end Anda. Secara khusus, perangkat IoT:

- Seringkali sistem tertanam tanpa operator manusia (tidak seperti telepon).
- Dapat digunakan di lokasi terpencil, di mana akses fisik mahal.
- Mungkin hanya dapat dijangkau melalui back end solusi.
- Mungkin memiliki daya dan sumber daya pemrosesan yang terbatas.
- Mungkin memiliki konektivitas jaringan yang terputus-putus, lambat, atau mahal.
- Mungkin perlu menggunakan protokol aplikasi khusus, kepemilikan, atau khusus industri.

Layanan back-end

Dalam solusi IoT, layanan back-end menyediakan fungsionalitas seperti:

- Menerima telemetri dalam skala besar dari perangkat Anda, dan menentukan cara memproses dan menyimpan data tersebut.
- Menganalisis telemetri untuk memberikan wawasan, baik secara real time atau setelah fakta.
- Mengirim perintah dari cloud ke perangkat tertentu.
- Menyediakan perangkat dan mengontrol perangkat mana yang dapat terhubung ke infrastruktur Anda.
- Mengontrol status perangkat Anda dan memantau aktivitasnya.
- Mengelola firmware yang diinstal pada perangkat Anda.

Misalnya, dalam solusi pemantauan jarak jauh untuk stasiun pompa oli, back end cloud menggunakan telemetri dari pompa untuk mengidentifikasi perilaku anomali. Ketika layanan back-end mengidentifikasi anomali, secara otomatis dapat mengirim perintah kembali ke perangkat untuk mengambil tindakan korektif. Proses ini menghasilkan loop umpan balik otomatis antara perangkat dan cloud yang sangat meningkatkan efisiensi solusi.

8. Referensi

[1] M. Aazam, I. Khan, A. A. Alsaffar, and E. N. Huh, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved," *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences and Technology, IBCAST 2014*, 2014, doi: 10.1109/IBCAST.2014.6778179.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, 2015, doi: 10.1109/COMST.2015.2444095.

[3] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of Things security: A survey," *Journal of Network and Computer Applications*, 2017, doi: 10.1016/j.jnca.2017.04.002.

[4] A. Whitmore, A. Agarwal, and L. Da Xu, "The Internet of Things—A survey of topics and trends," *Information Systems Frontiers*, 2015, doi: 10.1007/s10796-014-9489-2.

[5] A. Aijaz and A. H. Aghvami, "Cognitive machine-to-machine communications for internet-of-things: A protocol stack perspective," *IEEE Internet of Things Journal*, 2015, doi: 10.1109/JIOT.2015.2390775.

[6] I. A. Aziz, I. A. Hamizan, N. S. Haron, and M. Mehat, "Cooperative flood detection using GSMD via SMS," 2008, doi: 10.1109/ITSIM.2008.4632045.

[7] M. Leggieri and M. Hausenblas, "Interoperability of two RESTful protocols: HTTP and CoAP," in *REST: Advanced*

Research Topics and Practical Applications, vol. 9781461492, 2014, pp. 27–49.

[8] A. E. Standard *et al.*, "TCP/IP Protocol Suite."

[9] J. Bai, H. Xiao, X. Yang, and G. Zhang, "Study on integration technologies of building automation systems based on web services," 2009, doi: 10.1109/CCCM.2009.5267730.

[10] S. Kalra and S. K. Sood, "Secure authentication scheme for IoT and cloud servers," *Pervasive and Mobile Computing*, 2015, doi: 10.1016/j.pmcj.2015.08.001.

[11] A. Bouguettaya, Q. Z. Sheng, and F. Daniel, "Web services foundations," *Web Services Foundations*, 2013, doi: 10.1007/978-1-4614-7518-7.

[12] G. Huang, J. He, and Y. Zhang, "Web services for things," in *Advanced Web Services*, 2014.

[13] H. G. C. Ferreira, E. Dias Canedo, and R. T. De Sousa, "IoT architecture to enable intercommunication through REST API and UPnP using IP, ZigBee and arduino," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, 2013, pp. 53–60, doi: 10.1109/WiMOB.2013.6673340.

[14] M. Aazam and E. N. Huh, "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm," *IEEE Potentials*, 2016, doi: 10.1109/MPOT.2015.2456213.

[15] R. Buyya and A. V. Dastjerdi, *Internet of Things: Principles and Paradigms*. 2016.

[16] R. P. V. Chander, "Web Services for the Internet of Things – A Feasibility Study," *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 125–126, 2016, doi: 10.1109/DCOSS.2016.47.

[17] M. Ancona, A. Dellacasa, G. Delzanno, A. La Camera, and I. Rellini, "An 'Internet of Things ' Vision of the Flood Monitoring Problem," 2015.

[18] R. Klauck and M. Kirsche, "Bonjour Contiki: A case study of a DNS-based discovery service for the internet of things," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, doi: 10.1007/978-3-642-31638-8_24.

[19] C. Bormann, A. P. Castellani, and Z. Shelby, "CoAP: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, 2012, doi: 10.1109/MIC.2012.29.

[20] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)." https://tools.ietf.org/html/rfc7252 (accessed Aug. 11, 2020).

[21] F. Van Den Abeele, J. Hoebeke, I. Moerman, and P. Demeester, "Fine-grained management of CoAP interactions with constrained IoT devices," *IEEE/IFIP NOMS 2014 - IEEE/IFIP Network Operations and Management Symposium: Management in a Software Defined World*, 2014, doi: 10.1109/NOMS.2014.6838368.

[22] A. Antonic, M. Marjanovic, P. Skocir, and I. P. Zarko, "Comparison of the CUPUS middleware and MQTT protocol for smart city services," *Proceedings of the 13th International Conference on Telecommunications, ConTEL 2015*, 2015, doi: 10.1109/ConTEL.2015.7231225.

[23] O. Expertise, "SOA Design Principles and the Internet of Things 2014 IBM SOA Architect Summit SOA on Your Terms," 2014.

[24] D. Durand, Y. Iagolnitzer, P. Krzanik, C. Loge, and J. F. Susini, "Middleware for the Internet of Things: Principles," in *RFID and the Internet of Things*, 2013, pp. 183–215.

[25] A. F. da Silva, R. L. Ohta, M. N. dos Santos, and A. P. D. Binotto, "A Cloud-based Architecture for the Internet of Things targeting Industrial Devices Remote Monitoring and Control," *IFAC-PapersOnLine*, 2016, doi: 10.1016/j.ifacol.2016.11.137.

[26] "What is MQTT and How Does it Work?," *IoT Agenda*. https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport (accessed Aug. 11, 2020).

[27] "Beyond MQTT: A Cisco View on IoT Protocols," *Cisco Blogs*, May 01, 2013. https://blogs.cisco.com/digital/beyond-mqtt-a-cisco-view-on-iot-protocols (accessed Aug. 11, 2020).

[28] J. Aponte-Luis, J. Gómez-Galán, F. Gómez-Bravo, M. Sánchez-Raya, J. Alcina-Espigado, and P. Teixido-Rovira, "An Efficient Wireless Sensor Network for Industrial Monitoring and Control," *Sensors*, vol. 18, no. 2, p. 182, Jan. 2018, doi: 10.3390/s18010182.

[29] S. M. Kim, H. S. Choi, and W. S. Rhee, "IoT home gateway for auto-configuration and management of MQTT devices," *2015 IEEE Conference on Wireless Sensors, ICWiSE 2015*, 2016, doi: 10.1109/ICWISE.2015.7380346.

[30] "Extensible Messaging and Presence Protocol (XMPP): Core." https://xmpp.org/rfcs/rfc6120.html (accessed Aug. 11, 2020).

[31] M. Sneps-Sneppe and D. Namiot, "On web-based domain-specific language for Internet of Things," *International Congress on Ultra Modern Telecommunications and*

Control Systems and Workshops, 2016, doi: 10.1109/ICUMT.2015.7382444.

[32]B. G. Rama and 08/01/2017, "Report: AWS Market Share Is Triple Azure's -," *AWSInsider*. https://awsinsider.net/articles/2017/08/01/aws-market-share-3x-azure.aspx (accessed Aug. 27, 2020).

REPUBLIK INDONESIA
KEMENTERIAN HUKUM DAN HAK ASASI MANUSIA

SURAT PENCATATAN

CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC00202033496, 16 September 2020
Pencipta		
Nama	:	**Dessyanto Boedi Prasetyo**
Alamat	:	Perum Griya Taman Asri Blok G 301, , Kabupaten Sleman, Di Yogyakarta, 55512
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**LPPM UPN "Veteran" Yogyakarta**
Alamat	:	Jl. SWK Jl. Ring Road Utara No.104, Ngropoh, Condongcatur, Kec. Depok, Kabupaten Sleman, Di Yogyakarta, 55283
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Buku**
Judul Ciptaan	:	**Protokol Jaringan Dalam Internet Of Things**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	16 September 2020, di Kabupaten Sleman
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor pencatatan	:	000202873

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM DAN HAK ASASI MANUSIA
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL

Dr. Freddy Harris, S.H., LL.M., ACCS.
NIP. 196611181994031001