آموزش پروتکل

MQTT

استاندارد پیام‌رسانی اینترنت اشیاء

www.sasadra.com

فهرست مطالب

معرفی MQTT ، استاندارد پیام‌رسانی اینترنت اشیاء

MQTT یک پروتکل پیام‌رسانی استاندارد OASIS برای اینترنت اشیاء (IoT) است. این پروتکل، یک پیام‌رسان انتشار/اشتراک (publish/subscribe) است که بسیار سبک طراحی شده و برای اتصال دستگاه‌های راه دور با استفاده از یک کد کوچک و حداقل پهنای باند شبکه، ایده‌آل محسوب می‌شود. امروزه MQTT در صنایع مختلفی مانند خودروسازی، تولید، مخابرات، نفت و گاز و غیره استفاده می‌شود. آخرین نسخه منتشر شده این پروتکل v5.0 است.

چند نکته:

- نسخه‌های v5.0 و v3.1.1 اکنون از استانداردهای OASIS هستند (v3.1.1 نیز توسط ISO تأیید شده است).
- پورت TCP/IP 1883 برای استفاده MQTT و پورت TCP/IP 8883 برای استفاده MQTT از طریق SSL ثبت شده‌اند.
- در V3.1 پروتکل برای امنیت، می‌توان نام کاربری و رمز عبور را با یک بسته MQTT ارسال کرد. رمزگذاری در سراسر شبکه را می‌توان با SSL، مستقل از خود پروتکل MQTT انجام داد (شایان ذکر است که SSL سبک‌ترین پروتکل نیست و سربار شبکه قابل توجهی را اضافه می‌کند). امنیت بیشتر را می‌توان توسط برنامه‌ای که داده‌های ارسالی و دریافتی را رمزگذاری می‌کند اضافه کرد، و البته این چیزی نیست که در پروتکل تعبیه شده باشد تا آن را ساده و سبک نگه دارد.
- پلتفرم HiveMQ منبع‌باز است. نسخه Community آن، کارگزار MQTT را پیاده‌سازی کرده و با MQTT 3.1، 3.1.1 و MQTT 5 سازگار است. HiveMQ MQTT Client یک پیاده‌سازی مبتنی بر جاوا است که با MQTT 3.1.1 و MQTT 5 سازگار است. هر دو پروژه، تحت مجوز آپاچی در GitHub در دسترس هستند.

مزایای MQTT

- سبک و کارآمد

از آنجا که مشتریان MQTT (مشتری‌ها) بسیار کوچک هستند و به حداقل منابع نیاز دارند، بنابراین می‌توان از میکروکنترلرهای کوچک استفاده کرد. هدرهای پیام MQTT برای بهینه‌سازی پهنای باند شبکه، کوچک طراحی شده‌اند.

- ارتباطات دوطرفه

MQTT امکان ارسال پیام بین دستگاه و ابر و ابر به دستگاه را فراهم ساخته است. این امکان باعث می‌شود پیام‌های broadcast به راحتی به گروه‌هایی از اشیاء ارسال شود.

- مقیاس‌پذیری به میلیون‌ها شیء

MQTT می‌تواند برای اتصال میلیون‌ها دستگاه اینترنت اشیاء مقیاس‌پذیر باشد.

- تحویل پیام قابل اعتماد

قابلیت اطمینان از تحویل پیام (Reliability)، برای بسیاری از موارد استفاده در اینترنت اشیاء مهم است. به همین دلیل است که MQTT دارای 3 سطح کیفیت خدمات تعریف شده است:

0	حداکثر یک بار
1	حداقل یک بار
2	دقیقا یک بار

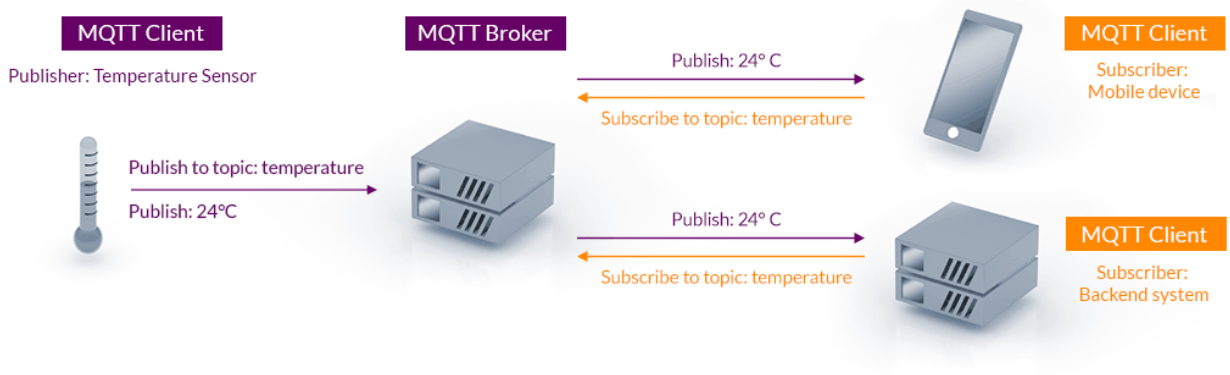
- پشتیبانی از شبکه‌های غیر قابل اعتماد

بسیاری از دستگاه‌های اینترنت اشیاء از طریق شبکه‌های سلولی غیرقابل اعتماد، متصل می‌شوند. پشتیبانی MQTT از جلسات مداوم و پایدار، زمان اتصال مجدد مشتری به کارگزار را کاهش می‌دهد.

- امنیت

رمزگذاری پیام‌ها با استفاده از TLS و احراز هویت مشتریان با استفاده از پروتکل‌های احراز هویت جدید مانند OAuth در MQTT آسان شده است.

معماری MQTT



موارد استفاده

MQTT در صنایع مختلفی مانند خودروسازی، تدارکات و آمایش (لجستیک)، کارخانه‌جات، خانه‌های هوشمند، صنایع نفت و گاز، حمل‌ونقل و لوازم خانگی و کاربردی استفاده می‌شود. نمونه‌های پیاده‌سازی شده این موارد در وب سایت مرجع MQTT قابل دیدن است.

برخی اصطلاحات و کلمات اختصاری

Broker

کارگزار، سروری است که پیام‌های منتشر شده را به مشترکین هدایت می‌کند.

Bridge

ارتباط بین دو کارگزار MQTT

(Really Small Message Broker) RSMB

متعلق به IBM بوده و اکنون بخشی از پروژه Eclipse Mosquitto است.

Last Will and Testament (LWT)

از آنجایی که MQTT اغلب در سناریوهایی استفاده می‌شود که شامل شبکه‌های غیرقابل اعتماد است، منطقی است که فرض کنیم برخی از مشتریان MQTT در این سناریوها گهگاه به‌طور ناخوشایند قطع می‌شوند. این قطع ناخوشایند ممکن است به دلیل قطع اتصال، خالی بودن باتری و یا بسیاری از دلایل دیگر رخ دهد. دانستن اینکه آیا یک مشتری به طور خوشایند (با پیام قطع اتصال MQTT) قطع شده است و یا به‌طور ناخوشایند (بدون پیام قطع)، به شما کمک می‌کند تا به درستی پاسخ دهید. ویژگی آخرین وصیت و عهد (Last Will and Testament) راهی را برای مشتریان فراهم می‌کند تا به قطع‌های ناخوشایند به روشی مناسب پاسخ دهند. در MQTT، از ویژگی

LWT استفاده می‌شود تا به مشتریان دیگر در مورد یک مشتری قطع شده به طور ناخوشایند اطلاع دهند. هر مشتری می‌تواند آخرین پیام خود را هنگام اتصال به یک کارگزار مشخص کند. آخرین پیام می‌تواند یک پیام معمولی MQTT با موضوع خاص، retained message flag، QoS و payload باشد. کارگزار تا زمانی که تشخیص دهد مشتری به طور ناخوشایند قطع شده است پیام را نگهداری می‌کند. در پاسخ به قطع ناخوشایند، کارگزار، پیام آخرین را برای همه مشتریان مشترک موضوع پیام آخر ارسال می‌کند. اگر مشتری با یک پیام DISCONNECT صحیح، ارتباط خود را به خوبی قطع کند کارگزار، پیام LWT ذخیره شده را دور می‌ریزد.LWT به شما کمک می‌کند تا زمانی که اتصال یک مشتری قطع می‌شود، راهبردهای مختلفی را پیاده‌سازی کنید و یا حداقل، سایر مشتریان را در مورد این وضعیت آفلاین مطلع کنید. در بخش‌های بعدی درباره LWT بیشتر صحبت شده است.

Machine to Machine (M2M)

ارتباط ماشین به ماشین، ارتباط مستقیم بین دستگاه‌ها با استفاده از هر کانال ارتباطی، از جمله سیمی و بی‌سیم است. این ارتباط می‌تواند شامل ابزار دقیق صنعتی باشد که حسگر را قادر می‌سازد تا با اطلاعاتی مانند دما، سطح موجودی، و غیره که ثبت می‌شوند ارتباط برقرار کند. چنین ارتباطی در اصل با داشتن یک شبکه راه دور از ماشین‌ها، اطلاعات را برای تجزیه و تحلیل به یک هاب مرکزی باز می‌گرداند تا پس از آن به سامانه‌ای مانند یک رایانه شخصی ارجاع داده شود.

M2M Industry Working Group (M2M IWG)

اعضای اولیه این گروه IBM، Sierra Wireless، Eurotech و Axeda هستند و این گروه برای تمرکز بر مسائل حوزه ارتباطات ماشین به ماشین تشکیل شد.

ملزومات MQTT

بخش 1 ملزومات ـ تاریخچه

پروتکل MQTT بسیار سبک است و به دلیل حداقل سربار بسته آن، هنگام انتقال داده‌ها از طریق سیم، بر پروتکل‌هایی مانند HTTP برتری دارد. یکی دیگر از جنبه‌های مهم پروتکل این است که پیاده‌سازی MQTT در سمت مشتری بسیار آسان است. قبلا سهولت استفاده، یکی از دغدغه‌های کلیدی در توسعه MQTT بود و امروزه آن را برای دستگاه‌های محدود با منابع محدود مناسب می‌کند.

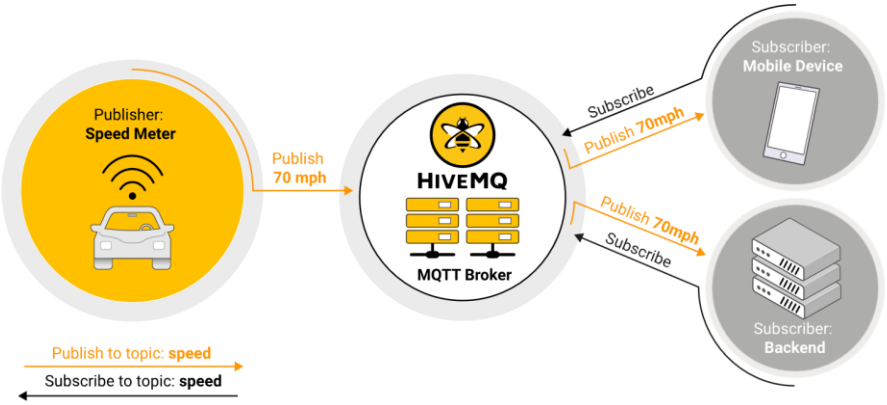
این پروتکل در سال 1999 اختراع شد. مخترعین آن به پروتکلی با حداقل تلفات باتری و حداقل پهنای باند برای اتصال خطوط لوله نفت از طریق ماهواره نیاز داشتند. آنها الزاماتی را برای پروتکل مشخص کردند:

- پیاده‌سازی ساده
- کیفیت ارائه داده‌های تحویلی (Quality of Service)
- سبک بودن و نیاز به حداقل پهنای باند
- استقلال از داده
- آگاهی از جلسات مستمر

این اهداف هنوز هم در نهاد MQTT هستند. با این حال، تمرکز اصلی پروتکل از سامانه‌های اختصاصی، به موارد استفاده در اینترنت اشیاء (IoT) تغییر یافته است. این تغییر در تمرکز، سبب سردرگمی زیادی در مورد آنچه MQTT مخفف آن است ایجاد کرده است. پاسخ کوتاه این است که MQTT دیگر به صورت مخفف در نظر گرفته نمی‌شود و به سادگی، MQTT نام یک پروتکل است.

بخش 2 ملزومات ـ الگوی انتشار/اشتراک یا publish/subscribe

پیش از بررسی MQTT ، ابتدا مدل انتشار/اشتراک به طور کلی بررسی می‌شود. الگوی انتشار/اشتراک (که به عنوان pub/sub هم شناخته می‌شود) جایگزینی برای معماری سنتی مشتری-سرور (client-server) است. در مدل مشتری-سرور، یک مشتری مستقیماً با یک نقطه پایانی ارتباط برقرار می‌کند. در مدل pub/sub ، مشتری ارسال کننده پیام (ناشر یا publisher) از مشتری یا مشتری‌های گیرنده پیام (مشترکین یا subscribers) جدا شده‌اند. ناشران و مشترکین هرگز مستقیماً با یکدیگر تماس نمی‌گیرند و عملا حتی از وجود یکدیگر اطلاعی ندارند. ارتباط بین آنها توسط یک جزء سوم با نام کارگزار یا broker انجام می‌شود. وظیفه کارگزار، فیلتر کردن تمامی پیام‌های دریافتی و توزیع صحیح آنها برای مشترکین است.



مهمترین جنبه pub/sub ، جدا شدن ناشر پیام از گیرنده یا همان مشترک است. این جداسازی چند بعد دارد:

- جداسازی فضا: ناشر و مشترک نیازی به شناخت یکدیگر ندارند. مثلاً به تبادل IP و پورت نیازی نیست.
- جداسازی زمان: ناشر و مشترک نیازی به اجرای همزمان ندارند.
- جداسازی همگام‌سازی (synchronization): ناشر و مشترک، نیازی به قطع شدن در حین انتشار یا دریافت ندارند.

به طور خلاصه، مدل pub/sub ارتباط مستقیم بین ناشر پیام و گیرنده/مشترک را حذف می‌کند. عمل فیلترینگ کارگزار این امکان را فراهم می‌کند تا مشخص شود کدام مشتری/مشترک چه پیامی را دریافت کند. جداسازی سه بعد دارد: فضا، زمان و همگام‌سازی.

مقیاس‌پذیری

روش Pub/Sub بهتر از رویکرد سنتی مشتری- سرور مقیاس‌پذیر است. این بدان جهت است که بسیاری از وظایف کارگزار را می‌توان به روش موازی کاری انجام داد و پیام‌ها را می‌توان به روش رویداد-محور یا event-driven پردازش کرد. حافظه پنهان پیام‌ها (cache) و مسیریابی هوشمند، اغلب از عوامل تعیین کننده برای بهبود مقیاس‌پذیری هستند. با این وجود، همچنان افزایش میلیون‌ها اتصال، یک چالش است. چنین سطح بالایی از اتصالات را می‌توان با گره‌های کارگزار خوشه‌ای (clustered broker) به دست آورد تا بار بر روی سرورهای منفرد بیشتری توزیع شود.

فیلتر کردن پیام

واضح است که نقش کارگزار در فرآیند pub/sub بسیار مهم است. اما چگونه کارگزار می‌تواند تمام پیام‌ها را فیلتر کند تا هر مشترک، تنها پیام‌های مورد علاقه خود را دریافت کند؟ پاسخ این است که کارگزار، چندین گزینه فیلتر دارد:

گزینه 1: فیلتر بر اساس موضوع

این فیلتر بر اساس موضوع یا موضوعی است که بخشی از هر پیام است. بدین ترتیب مشتری، تنها موضوعات مورد علاقه خود را مشترک می‌شود. از اینجا به بعد، کارگزار تضمین می‌کند که تمام پیام‌های منتشر شده با موضوعات مشترک شده، به دست مشتری می‌رسد. این موضوعات، متن‌هایی با ساختار سلسله مراتبی هستند و می‌توان بر اساس تعداد محدودی از عبارات، آنها را فیلتر کرد.

گزینه 2: فیلتر مبتنی بر محتوا

کارگزار در روش مبتنی بر محتوا، پیام را بر اساس یک زبان فیلتر خاص، فیلتر می‌کند تا پیام‌هایی که مشتریان به آنها علاقه‌مند هستند به دست‌شان برسد. نکته منفی مهم این روش این است که محتوای پیام باید از قبل شناخته شده باشد و نمی‌توان آن را رمزگذاری کرد و یا به راحتی تغییر داد.

گزینه 3: فیلتر بر اساس نوع

هنگامی که از زبان‌های شیءگرا استفاده می‌شود، فیلتر کردن بر اساس نوع/کلاس پیام (رویداد) یک روش معمول است. به عنوان مثال، یک مشترک می‌تواند به تمام پیام‌هایی که از نوع Exception و یا هر نوع فرعی دیگری هستند گوش دهد.

البته باید توجه داشت که روش pub/sub برای هر موردی مناسب نیست و قبل از استفاده از این مدل باید چند نکته را در نظر داشت. جداسازی ناشر و مشترک که پایه و اساس این روش است، چند چالش خاص خود را دارد. به عنوان مثال، شما باید ساختار داده‌های منتشر شده را از قبل بدانید. برای فیلتر مبتنی بر موضوع، هم ناشر و هم مشترک باید بدانند از چه موضوعاتی باید استفاده کنند. یکی دیگر از موارد مهم، تحویل پیام است. ناشر نمی‌تواند تظاهر کند که کسی پیام‌های ارسال شده را شنیده است. در برخی موارد، این امکان وجود دارد که یک پیام خاص را هیچ مشترکی نخواند.

MQTT

حال که به طور کلی مدل انتشار/اشتراک را بررسی کردیم، به طور خاص بر مبحث MQTT متمرکز می‌شویم. بسته به آنچه می‌خواهید به دست بیاورید، MQTT تمام جنبه‌های ذکر شده pub/sub را در بر می‌گیرد:

- MQTT ناشر و مشترک را جدا می‌کند. برای ارسال یا دریافت پیام، ناشرین و مشترکین فقط باید نام میزبان یا IP و پورت کارگزار را بدانند.
- اگرچه بیشتر مواقع، MQTT پیام‌ها را تقریبا بدون درنگ تحویل می‌دهد، در صورت تمایل، کارگزار می‌تواند پیام‌ها را برای مشتریانی که برخط نیستند ذخیره کند. برای ذخیره پیام‌ها دو شرط باید رعایت شود: مشتری باید با یک جلسه پایدار یا persistent وصل شده باشد و موضوعی با کیفیت خدمات (QoS) بزرگتر از 0 را مشترک شده باشد.
- MQTT به صورت ناهمزمان یا asynchronous کار می‌کند. از آنجا که بیشتر کتابخانه‌های سرویس گیرنده، به صورت ناهمزمان کار می‌کنند و بر اساس callback و یا مدلی مشابه هستند، در زمان انتظار برای پیام، و یا انتشار پیام، وظایف متوقف نمی‌شوند. البته در موارد استفاده خاص، همگام‌سازی، لازم و ممکن است. مثلا

برای انتظار یک پیام خاص، برخی از کتابخانه‌ها دارای APIهای همزمان هستند اما جریان کلی، معمولاً ناهمزمان است.

نکته دیگری که باید به آن اشاره کرد این است که به خصوص در سمت مشتری، استفاده از MQTT آسان است. بیشتر سامانه‌های pub/sub ، منطق مدیریتی‌شان در سمت کارگزار است، اما طراحی MQTT باعث می‌شود که برای دستگاه‌های کوچک و محدود، پروتکلی سبک باشد.
MQTT از فیلتر موضوعی پیام‌ها استفاده می‌کند. هر پیام یک موضوع دارد که کارگزار می‌تواند از آن برای تعیین اینکه آیا مشترک، پیام را دریافت می‌کند یا خیر استفاده کند.

برای رسیدگی به چالش‌های یک سامانه pub/sub، MQTT دارای سه سطح کیفیت خدمات (QoS) است. به راحتی می‌توان تعیین کرد که یک پیام با موفقیت از مشتری به کارگزار یا برعکس تحویل داده شود. با این حال، این احتمال وجود دارد که هیچ کس در یک موضوع خاص مشترک نشود. اگر این مورد، یک مشکل حساب می‌شود کارگزار باید بداند که چگونه با این وضعیت برخورد کند. به عنوان مثال، کارگزار HiveMQ MQTT یک افزونه دارد که می‌تواند چنین مواردی را حل کند. می‌توانید از کارگزار بخواهید که اقدامی خاص انجام دهد و یا پیام‌ها را برای تجزیه و تحلیل تاریخچه‌شان در پایگاه داده ذخیره کند. برای منعطف نگه داشتن درخت سلسله مراتبی موضوعی، مهم است که آن را با دقت طراحی کنید و فضایی را برای موارد استفاده آینده ایجاد کنید. اگر از این راهبردها استفاده شود، MQTT عالی است.

تمایز از صف پیام

در مورد نام MQTT و اینکه آیا این پروتکل به عنوان صف پیام پیاده‌سازی می‌شود یا خیر، سردرگمی زیادی وجود دارد. MQTT به محصول MQseries از IBM اشاره دارد و ربطی به "صف پیام" ندارد. صرف نظر از اینکه این نام از کجا آمده است، درک تفاوت بین MQTT و صف پیام سنتی مفید است:

- یک صف پیام، تا زمانی که پیام‌ها توسط مشتری استفاده می‌شوند ذخیره باقی می‌مانند (مشتری را اغلب مصرف کننده نیز می‌نامند). اگر هیچ کدام از مشتری‌ها پیامی را دریافت نکنند، آن پیام همچنان در صف گیر می‌کند و منتظر می‌ماند تا استفاده شود. در MQTT این مسئله حل شده است.
- یک پیام، تنها توسط یک مشتری استفاده و پردازش می‌شود. در MQTT برعکس است: هر مشترکی که در موضوعی مشترک شود پیام‌های مرتبط را دریافت می‌کند.

- روند ایجاد صف سخت‌تر از موضوع است. قبل از اینکه بتوان از یک صف استفاده کرد، باید آن را با یک دستور جداگانه ساخت و نامگذاری کرد. تنها پس از این مرحله است که امکان انتشار و یا استفاده پیام‌ها وجود دارد. در مقابل، موضوعات MQTT بسیار منعطف هستند و می‌توانند در لحظه ایجاد شوند.

بخش 3 ملزومات – مشتری، کارگزار و برقراری ارتباط

مشتری یا client

هنگام صحبت درباره یک مشتری، تقریباً همیشه مشتری MQTT مد نظر است. هم ناشرین و هم مشترکین، مشتریان MQTT هستند. اصطلاحات ناشر و مشترک به این اشاره دارند که آیا مشتری، در حال انتشار پیام‌ها است و یا برای دریافت پیام مشترک شده است (عملکرد انتشار و اشتراک نیز می‌تواند در همان مشتری MQTT پیاده‌سازی شود). هر دستگاهی می‌تواند مشتری MQTT باشد; از یک میکروکنترلر تا یک سرور کامل که کتابخانه MQTT را اجرا کند و از طریق شبکه به یک کارگزار MQTT متصل شود. برای مثال، مشتری MQTT می‌تواند یک دستگاه بسیار کوچک و با منابع محدود باشد که از طریق یک شبکه بی‌سیم وصل می‌شود و یک کتابخانه حداقلی دارد. مشتری MQTT همچنین می‌تواند یک رایانه معمولی باشد که یک مشتری گرافیکی MQTT را برای اهداف آزمایشی اجرا می‌کند. اصولاً، هر دستگاهی که با MQTT از طریق پشته TCP/IP صحبت کند، می‌تواند مشتری MQTT نامیده شود. پیاده‌سازی مشتری MQTT بسیار سرراست و ساده است. این سادگی یکی از دلایلی است که چرا MQTT برای دستگاه‌های کوچک مناسب است. کتابخانه‌های مشتری MQTT برای انبوهی از زبان‌های برنامه نویسی مانند Android، Arduino، C، C++، C#، Go، iOS، Java، JavaScript و .NET در دسترس هستند.

کارگزار یا broker

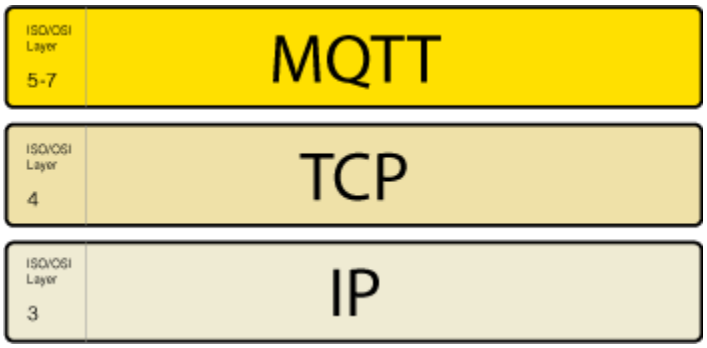
نقطه مقابل مشتری MQTT، کارگزار MQTT است. قلب هر پروتکل pub/sub ، کارگزار آن است. بسته به پیاده سازی، یک کارگزار می‌تواند تا میلیون‌ها مشتری متصل MQTT را همزمان مدیریت کند.

کارگزار، مسئول دریافت همه پیام‌ها، فیلتر کردن پیام‌ها، تعیین مشترکین هر پیام و ارسال پیام به این مشترکین است. کارگزار همچنین داده‌های جلسه تمام مشتریانی را که جلسات مداوم (persistent session) دارند، شامل اشتراک‌ها و پیام‌های از دست رفته نگهداری می‌کند. یکی دیگر از وظایف کارگزار، احراز هویت و مجوز مشتریان است. معمولاً کارگزار قابل توسعه است تا احراز هویت و مجوز سفارشی و یکپارچه‌سازی با سامانه‌های backend را ساده کند. یکپارچه‌سازی از اهمیت ویژه‌ای برخوردار است چرا که کارگزار، اغلب مؤلفه‌ای است که مستقیماً از طریق اینترنت در معرض دید همه است، مشتریان زیادی را مدیریت می‌کند و باید پیام‌ها را به سامانه‌های تحلیل

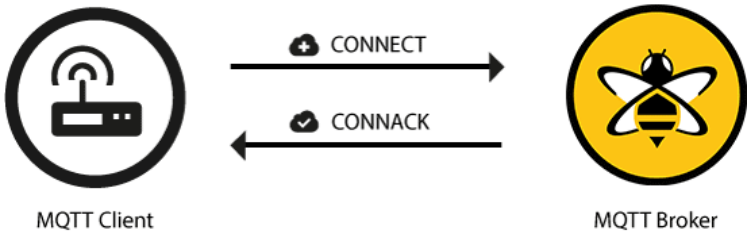
و پردازش پایین‌دستی ارسال کند. به طور خلاصه، کارگزار، مرکزی است که هر پیامی باید از آن عبور کند. بنابراین، مهم است که کارگزار شما بسیار مقیاس‌پذیر، قابل یکپارچه‌سازی با سامانه‌های backend، آسان برای نظارت، و البته مقاوم در برابر شکست باشد. HiveMQ این الزامات را با استفاده از پیشرفته ترین پردازش شبکه‌ای مبتنی بر رویداد، در یک سامانه متن‌باز فراهم آورده است.

اتصال MQTT

پروتکل MQTT مبتنی بر TCP/IP است. هم مشتری و هم کارگزار باید یک پشته TCP/IP داشته باشند.



همیشه اتصال MQTT بین یک مشتری و کارگزار است. مشتریان هرگز مستقیماً به یکدیگر متصل نمی‌شوند. برای شروع یک اتصال، مشتری یک پیام CONNECT به کارگزار ارسال می‌کند. کارگزار با یک پیام CONNACK و یک کد وضعیت پاسخ می‌دهد. هنگامی که اتصال برقرار شد، کارگزار آن را تا زمانی که مشتری، دستور قطع اتصال را ارسال کند یا اتصال قطع شود باز نگه می‌دارد.



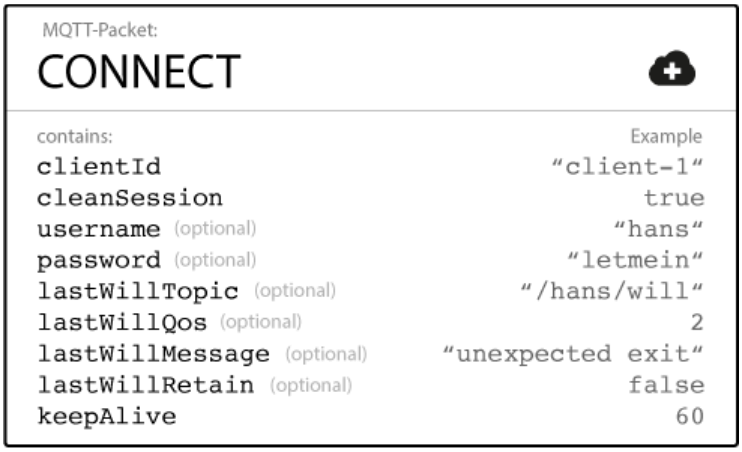
اتصال MQTT از طریق NAT

در بسیاری از موارد استفاده رایج، مشتری MQTT بعد از مسیریابی (router) قرار دارد که از NAT برای ترجمه از یک آدرس شبکه خصوصی به آدرس عمومی استفاده می‌کند. این مسئله، مشکلی ایجاد نمی‌کند چون همانطور که قبلاً اشاره شد، مشتری MQTT با ارسال یک پیام CONNECT به کارگزار، اتصال را آغاز می‌کند. از آنجایی که

کارگزار، یک آدرس عمومی دارد و اتصال را باز نگه می‌دارد تا ارسال و دریافت دوطرفه پیام‌ها ممکن باشد (بعد از اتصال اولیه)، هیچ مشکلی با مشتری‌هایی که پشت NAT قرار دارند، وجود ندارد.

برقراری ارتباط با ارسال پیام CONNECT

مشتری برای شروع یک ارتباط، یک پیام فرمان CONNECT به کارگزار ارسال می‌کند. اگر این پیام، مطابق مشخصات MQTT نباشد و یا زمان زیادی بین باز کردن سوکت شبکه و ارسال پیام اتصال بگذرد، کارگزار اتصال را می‌بندد. این یک رفتار بازدارنده برای مشتریان مخربی است که می‌توانند کارگزار را کند کنند. یک پیام درست CONNECT به شکل زیر است:



پاسخ کارگزار با پیام CONNACK

هنگامی که یک کارگزار پیام CONNECT را دریافت کرد موظف است با یک پیام CONNACK پاسخ دهد. پیام CONNACK شامل دو بخش است:

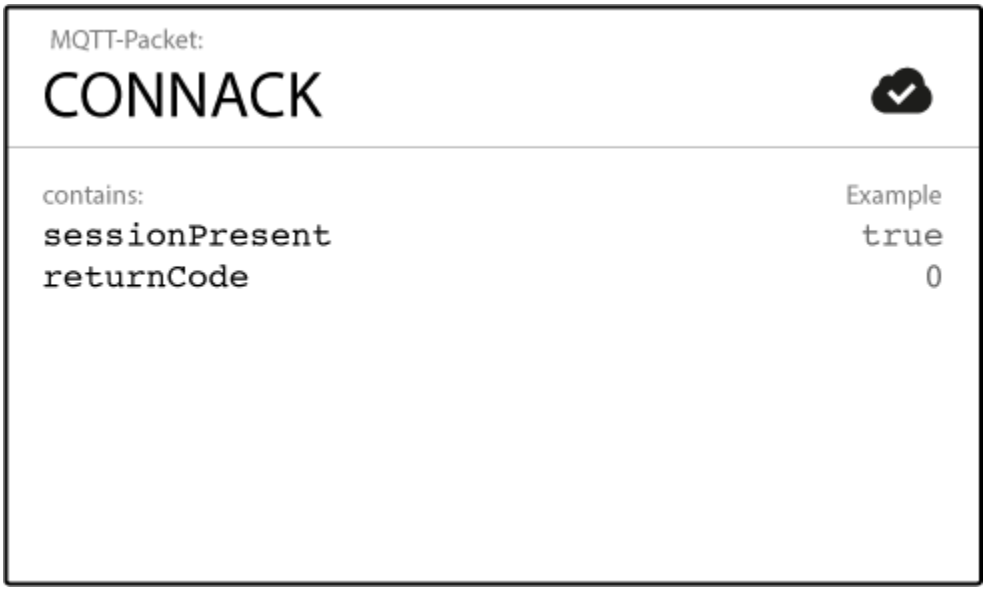
Session Present flag

این پرچم به مشتری می‌گوید که آیا کارگزار قبلاً یک جلسه مداوم در دسترس، از تعاملات قبلی با مشتری دارد یا خیر. اگر یک مشتری با Clean Session با مقدار true متصل شود، این پرچم همیشه نادرست است زیرا هیچ جلسه‌ای در دسترس نیست و اگر مقدارش false باشد، دو احتمال وجود دارد: اگر اطلاعات جلسه برای clientId در دسترس باشد و اطلاعات جلسه را کارگزار ذخیره کرده باشد، پرچم true است. در غیر این صورت، اگر کارگزار هیچ اطلاعات جلسه‌ای برای ClientId نداشته باشد، پرچم false است. این پرچم در MQTT 3.1.1 اضافه شد تا به

مشتریان کمک کند که آیا باید در موضوعاتی مشترک شوند و یا اینکه آن موضوعات هنوز در یک جلسه مداوم ذخیره شده‌اند یا خیر.

Connect return code

این پرچم دوم، پرچم تأیید اتصال است که حاوی یک کد بازگشتی بوده و به مشتری می‌گوید آیا تلاش برای اتصال، موفقیت آمیز بوده است یا خیر.



جدول کدهای بازگشتی در زیر آمده است:

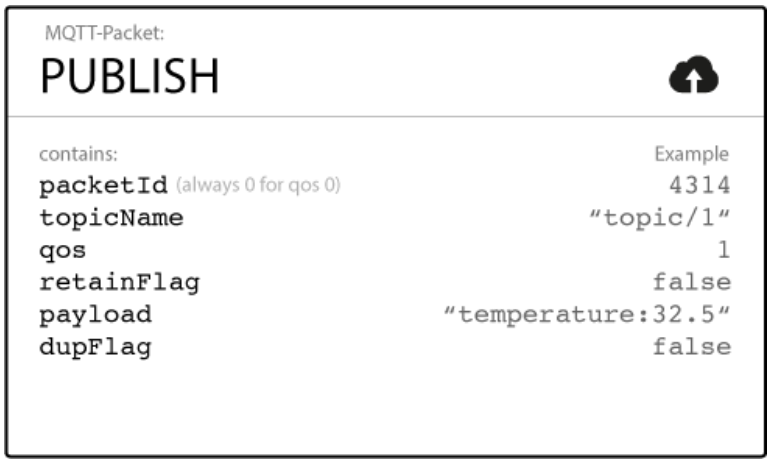
Return Code	Return Code Response
0	Connection accepted
1	Connection refused, unacceptable protocol version
2	Connection refused, identifier rejected
3	Connection refused, server unavailable
4	Connection refused, bad user name or password

بخش 4 ملزومات – انتشار، اشتراک و لغو آن

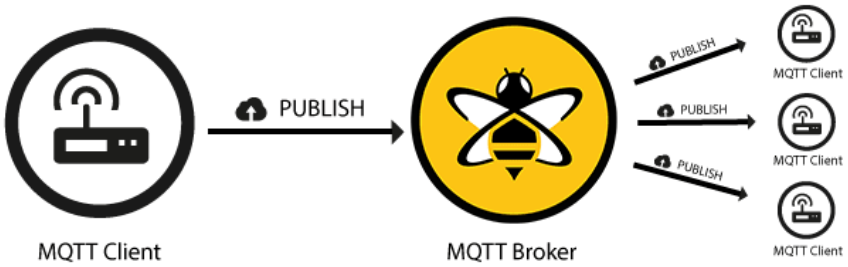
انتشار یا Publish

یک مشتری MQTT می‌تواند به محض اتصال به کارگزار، پیام‌ها را منتشر کند. MQTT از فیلتر موضوعی پیام‌های موجود در کارگزار استفاده می‌کند. هر پیام باید حاوی موضوعی باشد که کارگزار بتواند از آن برای ارسال پیام به مشتریان علاقه‌مند استفاده کند. به طور معمول، هر پیام حاوی داده‌هایی مانند تصویر، متن و یا هر چیزی است (payload) که باید در قالب بایت، ساختار یافته و منتقل شوند. مشتری ارسال کننده یا همان ناشر تصمیم می‌گیرد

که آیا می‌خواهد داده‌های باینری، متنی یا حتی XML و JSON ارسال کند. یک پیام PUBLISH در MQTT چندین بخش دارد که در شکل زیر آمده است.



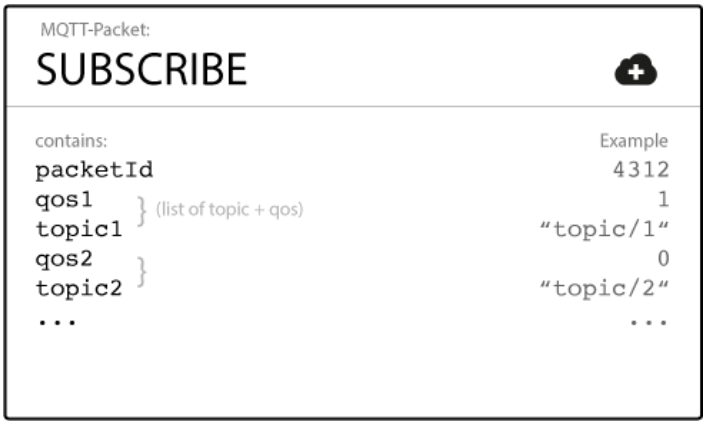
هنگامی که مشتری پیامی را برای انتشار به یک کارگزار MQTT ارسال می‌کند، کارگزار پیام را می‌خواند، تأیید می‌کند (طبق سطح QoS)، و پردازش می‌کند. پردازش شامل تعیین این است که کدام مشتریان، مشترک موضوع شده‌اند و پیام باید برای‌شان ارسال شود.



هر مشتری که ابتدا پیام را منتشر می‌کند فقط علاقه‌مند است بداند آیا پیام PUBLISH به کارگزار تحویل شده است یا خیر. هنگامی که کارگزار، پیام PUBLISH را دریافت کرد، مسئولیت تحویل پیام به همه مشترکین بر عهده خود کارگزار است. در مورد اینکه آیا کسی به پیام منتشر شده، علاقه‌مند است و یا اینکه چند مشتری، پیام را از کارگزار دریافت کرده‌اند، به ناشر پیام اطلاعی داده نخواهد داشت.

پیام SUBSCRIBE

انتشار پیامی که به دست کسی نرسد (کسی مشترک موضوع پیام نباشد) منطقی نیست. برای دریافت پیام در زمینه موضوعات مورد علاقه، مشتری یک پیام SUBSCRIBE به کارگزار MQTT ارسال می‌کند. این پیامی بسیار ساده بوده و حاوی یک شناسه منحصر به فرد و فهرستی از اشتراک‌ها است.



شناسه بسته

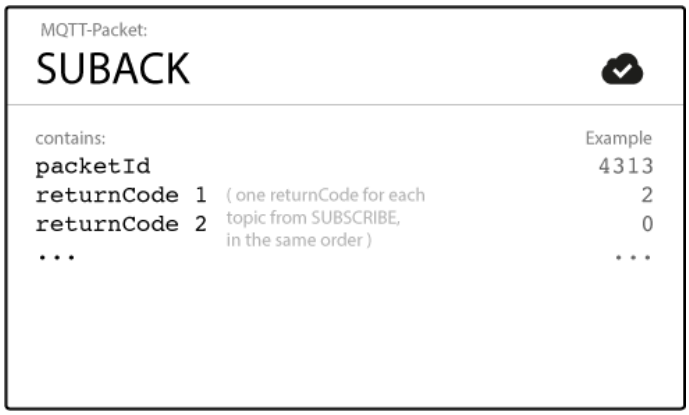
شناسه بسته یا Packet Identifier به طور منحصربه‌فرد، پیامی را که بین مشتری و کارگزار جریان دارد شناسایی می‌کند. برای ساختن آن از کتابخانه مشتری و/یا کارگزار استفاده می‌شود.

فهرست اشتراک‌ها

یک پیام SUBSCRIBE می‌تواند شامل چندین اشتراک برای یک مشتری باشد. هر اشتراک از دو بخش موضوع و سطح QoS تشکیل شده است. موضوع اشتراک می‌تواند حاوی wildcard باشد تا مجموعه‌ای از موضوع‌ها را شامل شود. اگر اشتراک‌های همپوشان برای یک مشتری وجود داشته باشد، کارگزار پیامی را تحویل می‌دهد که بالاترین سطح QoS را برای آن موضوع دارد.

Suback

برای تأیید هر اشتراک، کارگزار یک پیام تأیید SUBACK برای مشتری ارسال کند. این پیام، حاوی شناسه پیام اصلی Subscribe (برای شناسایی) و فهرستی از کدهای بازگشتی (return code) است.



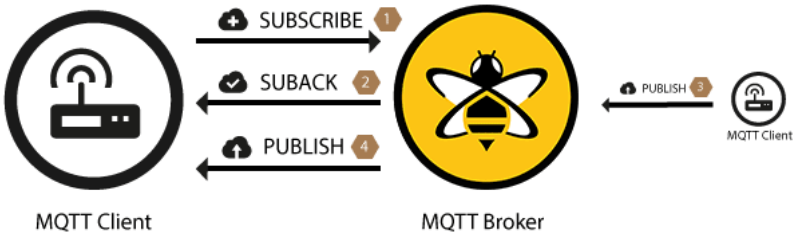
شناسه بسته

شناسه بسته یا Packet Identifier منحصر به فرد بوده و مانند پیام SUBSCRIBE است.

کد بازگشتی

کارگزار برای هر موضوع/QoS که در پیام SUBSCRIBE دریافت می‌کند، یک کد بازگشتی ارسال می‌کند. برای مثال، اگر پیام SUBSCRIBE دارای پنج اشتراک باشد، پیام SUBACK حاوی پنج کد بازگشتی است. کد بازگشتی برای تأیید موضوع است و سطح QoS ارائه شده توسط کارگزار را نشان می‌دهد. اگر کارگزار اشتراک را رد کند، پیام SUBACK حاوی کد بازگشتی خطا برای آن موضوع خاص است. به عنوان مثال اگر مشتری، مجوز کافی برای اشتراک موضوع را نداشته باشد و یا موضوع غیر معتبر باشد این کد خطا ارسال می‌شود.

Return Code Response	Return Code
Success - Maximum QoS 0	0
Success - Maximum QoS 1	1
Success - Maximum QoS 2	2
Failure	128



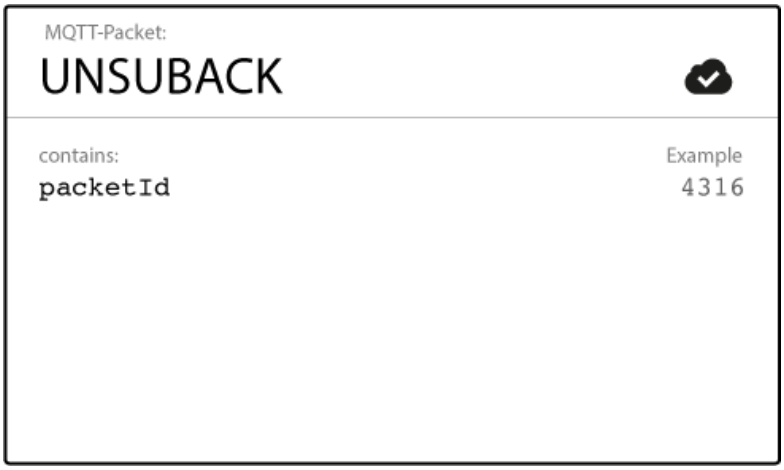
پس از اینکه مشتری با موفقیت پیام SUBSCRIBE را ارسال و پیام SUBACK را دریافت کرد، هر پیام منتشر شده‌ای که با موضوعات پیام SUBSCRIBE مطابقت دارد را دریافت می‌کند.

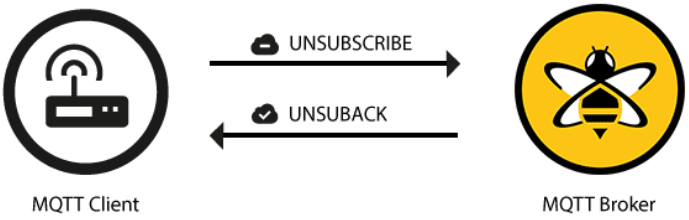
لغو اشتراک

همتای پیام SUBSCRIBE پیام UNSUBSCRIBE است که اشتراک‌های مشتری در کارگزار را حذف می‌کند. پیام UNSUBSCRIBE هم مشابه پیام SUBSCRIBE دارای شناسه بسته و فهرستی از موضوعات بدون QoS است. کارگزار، موضوع را بدون توجه به سطح QoS که در ابتدا با آن مشترک شده بود لغو می‌کند.

تأیید لغو اشتراک

برای تأیید لغو اشتراک، کارگزار یک پیام تأیید UNSUBACK برای مشتری ارسال می‌کند. این پیام فقط حاوی شناسه پیام UNSUBSCRIBE اصلی برای تشخیص پیام است.





پس از دریافت UNSUBACK از کارگزار، مشتری می‌تواند فرض کند که اشتراک‌های موجود در پیام UNSUBSCRIBE حذف شده‌اند.

بخش 5 ملزومات – موضوعات و بهترین روش‌ها

موضوعات یا Topics

در MQTT، کلمه topic یک رشته UTF-8 است که کارگزار از آن برای فیلتر کردن پیام‌ها برای هر مشتری متصل استفاده می‌کند. هر موضوع از یک یا چند سطح موضوعی تشکیل شده و هر سطح مبحث با یک / (اسلش) از بقیه جدا می‌شود. مانند:

در مقایسه با صف پیام، موضوعات MQTT بسیار سبک هستند. مشتری نیازی به ایجاد موضوع مورد نظر، قبل از انتشار یا اشتراک آن ندارد. کارگزار هر موضوع معتبر را بدون مقدار اولیه می‌پذیرد. در اینجا چند نمونه از موضوعات وجود دارد:

myhome/groundfloor/livingroom/temperature
USA/California/San Francisco/Silicon Valley
5ff4a2ce-e485-40f4-826c-b1a5d81be9b6/status
Germany/Bavaria/car/2382340923453/latitude

توجه کنید که هر موضوع باید حداقل 1 حرف داشته باشد و استفاده از حرف فاصله یا space هم مجاز است. موضوعات به حروف بزرگ و کوچک حساس هستند. برای مثال myhome/temperature و Myhome/Temperature دو موضوع متفاوت هستند. علاوه بر این، یک / تنها هم یک موضوع معتبر است.

Wildcards

هنگامی که یک مشتری، مشترک موضوعی می‌شود، می‌تواند موضوع دقیق پیام منتشر شده را مشترک شود و یا با استفاده از Wildcard برای اشتراک چندین موضوع اقدام کند. از Wildcard فقط می‌توان برای اشتراک موضوعات و نه برای انتشار پیام استفاده کرد. دو نوع Wildcard وجود دارد: تک سطحی و چند سطحی.

تک سطحی: +

همانطور که از نام آن پیداست، جایگزین یک سطح موضوعی می‌شود و نماد بعلاوه نشان دهنده آن است.

بیانگیر هر موضوعی است که در آن یک رشته دلخواه، به جای علامت + قرار گیرد.

برای مثال، اشتراک myhome/groundfloor/+/temperature می‌تواند نتایج زیر را ایجاد کند:

چند سطحی:

wildcard چند سطحی، بسیاری از سطوح موضوعی را پوشش می‌دهد و نماد # نشان دهنده آن است. برای اینکه کارگزار تعیین کند که کدام موضوعات مطابقت دارند، علامت # باید به عنوان آخرین حرف موضوع گذاشته شده و قبل از آن یک / قرار گیرد.

در این حالت، مشتری همه پیام‌های موضوعی که با الگوی قبل از wildcard شروع می‌شود را دریافت می‌کند، و مهم نیست موضوع چقدر طولانی یا عمیق باشد. اگر فقط علامت # را به کار برید، همه پیام‌هایی که به کارگزار MQTT ارسال می‌شود را دریافت می‌کنید. اگر انتظار خروجی بالایی دارید، اشتراک با wildcard چند سطحی می‌تواند نامناسب باشد.

موضوعاتی که با $ شروع می‌شوند

به طور کلی می‌توان موضوعات MQTT را به دلخواه نام‌گذاری کرد. با این حال یک استثناء وجود دارد: موضوعاتی که با نماد $ شروع می‌شوند، هدف متفاوتی دارند. موضوعات نماد $ برای آمار داخلی کارگزار MQTT محفوظ هستند و مشتریان نمی‌توانند پیام‌هایی را برای این موضوعات منتشر کنند. فعلا استانداردسازی رسمی برای چنین موضوعاتی وجود ندارد. نمونه‌هایی از آن در ویکی MQTT GitHub موجود هستند مانند:

$SYS/broker/clients/connected
$SYS/broker/clients/disconnected
$SYS/broker/clients/total
$SYS/broker/messages/sent
$SYS/broker/uptime

برخی توصیه‌ها

- هرگز از اسلش در ابتدای موضوع استفاده نکنید

اسلش ابتدایی در MQTT مجاز است. به عنوان مثال:
/myhome/groundfloor/livingroom
با این حال هیچ فایده‌ای ندارد و اغلب باعث سردرگمی می‌شود.

- هرگز از حرف فاصله در یک موضوع استفاده نکنید

حرف فاصله سبب می‌شود خواندن و اشکال‌زدایی موضوعات بسیار سخت‌تر شود. همانند دلیلی که برای / آورده شد، تنها به این دلیل که چیزی مجاز است، بدین معنی نیست که باید از آن استفاده کرد. UTF-8 انواع مختلفی از حروف فاصله دارد که باید از آنها دوری کرد.

- موضوع را کوتاه و مختصر نگه دارید

موضوعات خود را تا حد امکان کوتاه و مختصر تعریف کنید. وقتی صحبت از دستگاه‌های کوچک می‌شود، هر بایت مهم است و طول موضوع تأثیر زیادی دارد.

- فقط از حروف ASCII استفاده کرده و از حروف غیر قابل چاپ خودداری کنید

از آنجایی که حروف غیر ASCII و UTF-8 اغلب به اشتباه نمایش داده می‌شوند، یافتن اشتباهات تایپی یا مسائل مربوط به آنها بسیار دشوار است. توصیه می‌شود از استفاده از این حروف برای یک موضوع اجتناب شود، مگر اینکه کاملاً ضروری باشد.

- یک شناسه منحصر به فرد یا شناسه مشتری را در موضوع جاسازی کنید

گنجاندن شناسه منحصر به فرد مشتری در موضوع می‌تواند بسیار مفید باشد. این شناسه کمک می‌کند تشخیص دهید چه کسی پیام را ارسال کرده است. شناسه تعبیه شده می‌تواند برای الزام به احراز هویت استفاده شود. به این ترتیب فقط سرویس گیرنده‌ای مجاز به انتشار در آن موضوع است که دارای شناسه مشتری مشابه با شناسه در موضوع باشد. برای مثال، مشتری با شناسه client1 مجاز به انتشار client1/status است، اما مجاز به انتشار client2/status نیست.

- مشترک # نشوید

گاهی اوقات لازم است تمام پیام‌هایی که از طریق کارگزار منتقل می‌شود را مشترک شوید. به عنوان مثال، هنگامی که می‌خواهید همه پیام‌ها را در یک پایگاه داده ذخیره کنید. استفاده از یک مشتری MQTT و اشتراک همه پیام‌ها راه درستی نیست چرا که اغلب، مشتری قادر به پردازش این حجم از پیام نیست. توصیه این است که یک افزونه در کارگزار MQTT پیاده‌سازی شود. به عنوان مثال، با افزونه HiveMQ می‌توان یک روال ناهمزمان برای پردازش هر پیام دریافتی و حفظ آن در پایگاه داده اضافه کرد.

- توسعه‌پذیری را فراموش نکنید

موضوعات، مفاهیمی منعطف و در حال رشد هستند و به هیچ وجه به تخصیص قبلی آنها نیازی نیست. با این حال، هم ناشر و هم مشترک باید به این فکر کنند که چگونه می‌توان موضوعات را گسترش داد تا ویژگی‌ها یا محصولات جدید را اضافه کرد. به عنوان مثال، اگر راه‌حل خانه هوشمند شما حسگرهای جدیدی اضافه می‌کند، باید بدون تغییر کل سلسله مراتب موضوع، آنها را به درخت موضوعی اضافه نمود.

- از موضوعات خاص استفاده کنید نه موضوعات عمومی

موضوعات خود را تا حد امکان متمایز کنید. برای مثال، اگر سه حسگر در اتاق نشیمن دارید، موضوعاتی مانند myhome/livingroom/temperature، myhome/livingroom/brightness و myhome/livingroom/humidity را ایجاد

کنید و همه مقادیر را روی myhome/livingroom ارسال نکنید. استفاده از یک موضوع برای همه پیام‌ها یک ضد الگو است. همچنین نام‌گذاری خاص، این امکان را برای شما فراهم می‌کند تا از سایر ویژگی‌های MQTT مانند پیام‌های حفظ شده (retained messages) استفاده کنید.

بخش 6 ملزومات – کیفیت خدمات 0 و 1 و 2

سطوح کیفیت خدمات

سطوح کیفیت خدمات (Quality of Service) توافقی بین فرستنده پیام و گیرنده آن است که تضمین تحویل یک پیام خاص را مشخص می‌کند. درMQTT 3 سطح QoS وجود دارد:

- حداکثر یک بار(0)
- حداقل یک بار(1)
- دقیقا یک بار.(2)

وقتی در مورد QoS در MQTT صحبت می‌شود، باید دو طرف تحویل پیام را در نظر گرفت:

- تحویل پیام از ناشر به کارگزار
- تحویل پیام از کارگزار به مشتری

ما به دو طرف تحویل پیام به طور جداگانه نگاه خواهیم کرد زیرا تفاوت‌های ظریفی بین این دو وجود دارد. ناشر پیام به کارگزار، سطح QoS آن را تعریف می‌کند. کارگزار با استفاده از سطحی که هر مشترک در طول فرآیند اشتراک تعریف می‌کند، این پیام را به مشترکین ارسال می‌کند. اگر مشترک، QoS کمتری نسبت به ناشر تعریف کند کارگزار، پیام را با کیفیت خدمات پایین‌تر منتقل می‌کند.

چرا کیفیت خدمات مهم است؟

QoS یکی از ویژگی‌های کلیدی پروتکل MQTT است .و به مشتری این قدرت را می‌دهد تا سطحی از خدمات را انتخاب کند که مطابق با قابلیت اطمینان شبکه و منطق برنامه باشد. از آنجایی که MQTT ارسال مجدد پیام‌ها را مدیریت کرده و تحویل را تضمین می‌کند (حتی زمانی که ارتباط اصلی قابل اعتماد نیست)، QoS ارتباط در شبکه‌های غیرقابل اعتماد را بسیار آسان‌تر می‌کند.

روش کار QoS

بیایید به نحوه پیاده‌سازی هر سطح QoS در پروتکل MQTT و نحوه عملکرد آن نگاهی دقیق‌تر بیندازیم:

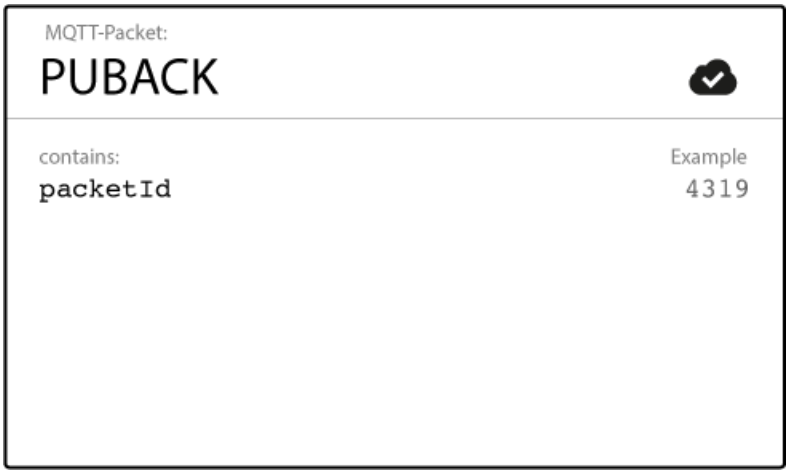
سطح QoS 0 - حداکثر یک بار

حداقل سطح QoS صفر است. در این سطح خدمات، هیچ تضمینی برای تحویل وجود ندارد. گیرنده، دریافت پیام را تأیید نمی‌کند، و فرستنده نیز پیام را ذخیره و مجددا ارسال نمی‌کند. سطح 0 را اغلب "fire and forget" می‌نامند و تضمینی مشابه پروتکل TCP زیربنایی دارد.

QoS سطح 1 - حداقل یک بار

سطح 1 تضمین می‌کند که یک پیام، حداقل یک بار به گیرنده تحویل داده می‌شود. فرستنده، پیام را تا زمانی که یک بسته PUBACK به عنوان تأیید از گیرنده دریافت کند ذخیره می‌کند. این امکان وجود دارد که یک پیام چندین بار ارسال یا تحویل داده شود.

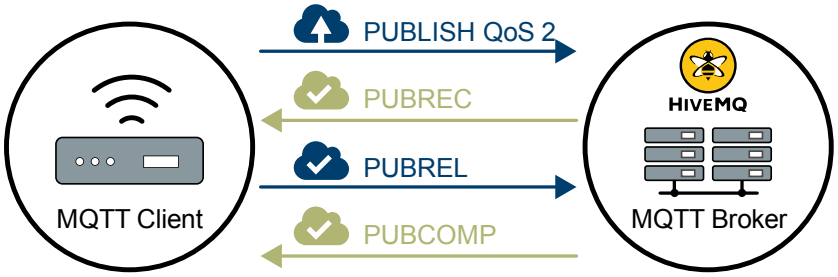
فرستنده از شناسه بسته استفاده می‌کند تا بسته PUBLISH را با بسته PUBACK مربوطه مطابقت دهد. اگر فرستنده، PUBACK را در مدت زمان معقول دریافت نکند، بسته PUBLISH را دوباره ارسال می‌کند. هنگامی که یک گیرنده، پیامی با QoS 1 دریافت می‌کند، می‌تواند بلافاصله آن را پردازش کند. به عنوان مثال، اگر گیرنده یک کارگزار باشد، کارگزار پیام را برای همه مشترکین ارسال می‌کند و سپس با یک بسته PUBACK پاسخ می‌دهد.



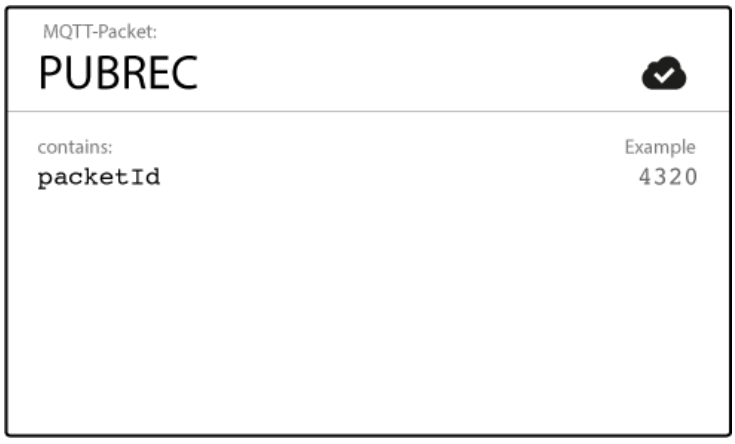
اگر ناشر، دوباره پیام را ارسال کند، یک پرچم تکراری (DUP) تعیین می‌کند. در QoS 1، این پرچم فقط برای اهداف داخلی استفاده می‌شود و توسط کارگزار یا مشتری پردازش نمی‌شود. گیرنده پیام بدون توجه به پرچم DUP یک PUBACK ارسال می‌کند.

QoS سطح 2 - دقیقا یک بار

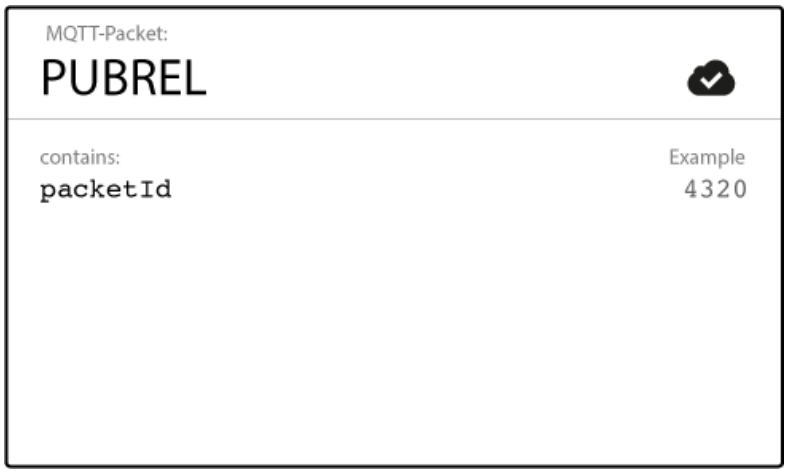
بالاترین سطح خدمات در MQTT است و تضمین می‌کند که هر پیام، تنها یک بار توسط گیرندگان مورد نظر دریافت می‌شود. QoS 2 ایمن ترین و کندترین سطح کیفیت خدمات است. تضمین با حداقل دو جریان درخواست/پاسخ (یک handshake چهار قسمتی) بین فرستنده و گیرنده تامین می‌شود. فرستنده و گیرنده از شناسه بسته پیام PUBLISH اصلی برای هماهنگ کردن تحویل پیام استفاده می‌کنند.



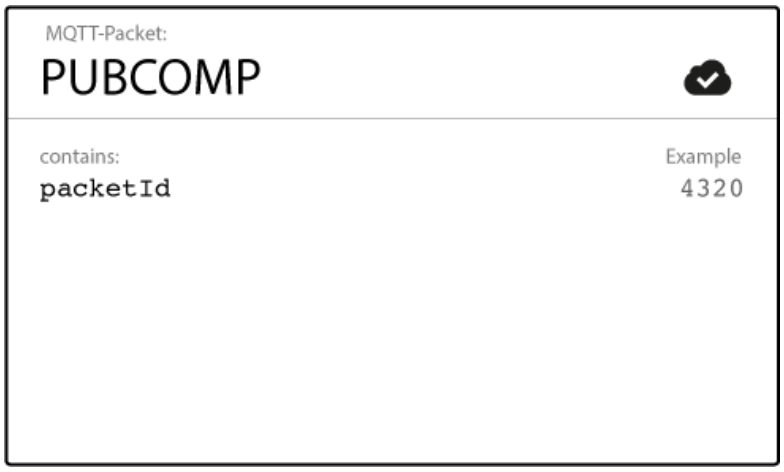
هنگامی که گیرنده، یک بسته QoS 2 PUBLISH را از فرستنده دریافت می‌کند، پیام انتشار را بر این اساس پردازش می‌کند و با یک بسته PUBREC که بسته PUBLISH را تأیید می‌کند به فرستنده پاسخ می‌دهد. اگر فرستنده، PUBREC از گیرنده دریافت نکند، PUBLISH را دوباره با یک پرچم تکراری (DUP) ارسال می‌کند تا زمانی که یک تأیید دریافت کند.



هنگامی که فرستنده، یک PUBREC از گیرنده دریافت کرد، می‌تواند با خیال راحت PUBLISH اولیه را دور بیندازد. فرستنده، PUBREC را از گیرنده ذخیره می‌کند و با یک PUBREL پاسخ می‌دهد.



پس از اینکه گیرنده، بسته PUBREL را دریافت کرد، می‌تواند تمام حالت‌های ذخیره شده را دور بیندازد و با یک بسته PUBCOMP پاسخ دهد (همانطور که فرستنده PUBCOMP را دریافت می‌کند). تا زمانی که گیرنده، پردازش را کامل کند و بسته PUBCOMP را به فرستنده بازگرداند گیرنده، یک مرجع به شناسه بسته PUBLISH اصلی را ذخیره می‌کند. این مرحله برای جلوگیری از پردازش پیام برای بار دوم مهم است. پس از اینکه فرستنده، PUBCOMP را دریافت کرد، شناسه بسته پیام منتشر شده برای استفاده مجدد در دسترس می‌شود.



هنگامی که جریان QoS 2 کامل شد، هر دو طرف مطمئن هستند که پیام تحویل داده شده است و فرستنده، تأییدیه تحویل را دارد.

اگر بسته‌ای در طول مسیر گم شود، فرستنده (چه مشتری باشد وچه کارگزار) موظف است پیام را در مدت زمان معقولی دوباره ارسال کند. گیرنده مسئولیت دارد که به هر پیام فرمان پاسخ دهد.خوب است بدانیدبرخی از

جنبه‌های QoS در نگاه اول چندان واضح نیستند. در اینجا چند نکته وجود دارد که هنگام استفاده از QoS باید در نظر داشت:

تنزل رتبه QoS

همانطور که قبلاً اشاره شد، تعریف QoS و سطوح بین مشتری که پیام را ارسال می‌کند (انتشار می‌کند) و مشتری که پیام را دریافت می‌کند دو چیز متفاوت هستند. سطوح QoS این دو تعامل نیز می‌تواند متفاوت باشد. مشتری که پیام PUBLISH را برای کارگزار ارسال می‌کند، QoS پیام را تعریف می‌کند. با این حال، هنگامی که کارگزار پیام را به گیرندگان (مشترکین) تحویل می‌دهد، کارگزار از QoSی استفاده می‌کند که گیرنده (مشترک) در هنگام اشتراک تعریف کرده است. برای مثال مشتری A فرستنده پیام است. مشتری B گیرنده پیام است. اگر مشتری B با QoS 1 در کارگزار مشترک شود و مشتری A با QoS 2 پیام را برای کارگزار ارسال کند، کارگزار پیام را به مشتری B (گیرنده/مشترک) با QoS 1 تحویل می‌دهد. پیام را می‌توان بیش از یک بار به مشتری تحویل داد. زیرا QoS 1 تحویل پیام را حداقل یک بار تضمین می‌کند و مانع از تحویل چندگانه یک پیام نمی‌شود.

بخش 7 ملزومات – جلسات مداوم و صف پیام‌ها

جلسات مداوم یا Persistent Sessions

ابتدا لازم است گفته شود که اگرچه MQTT یک صف پیام نیست، اما می‌تواند پیام‌ها را برای مشتریان در صف قرار دهد. برای دریافت پیام‌ها از یک کارگزار MQTT، مشتری به کارگزار وصل شده و موضوعات مورد علاقه خود را مشترک می‌شود. اگر ارتباط بین مشتری و کارگزار در طول یک جلسه غیر مداوم قطع شود، این موضوعات از بین می‌روند و مشتری باید مجدداً در اتصال جدید مشترک شود. اشتراک مجدد در هر قطعی اتصال، برای مشتریان با منابع محدود، بار سنگینی است. برای جلوگیری از این مشکل، مشتری می‌تواند هنگام اتصال به کارگزار، یک جلسه مداوم را درخواست کند. در جلسات مداوم، تمام اطلاعات مربوط به مشتری در کارگزار ذخیره می‌شود و شناسه جلسه نیز همان ClientID است که مشتری هنگام برقراری ارتباط با کارگزار دارد.

در یک جلسه مداوم چه چیزی ذخیره می‌شود؟

در یک جلسه مداوم، کارگزار (حتی اگر مشتری آفلاین باشد) اطلاعات زیر را ذخیره می‌کند. هنگامی که مشتری دوباره وصل می‌شود، این اطلاعات بلافاصله در دسترس است.

- وجود یک جلسه (حتی اگر اشتراکی وجود نداشته باشد).
- تمام اشتراک‌های مشتری
- همه پیام‌های QoS 1 یا 2 که مشتری هنوز تأیید نکرده است.

- همه پیام‌های QoS 1 یا 2 جدیدی که مشتری در حالت آفلاین از دست داده است.
- تمام پیام‌های QoS 2 دریافت شده از مشتری که هنوز به طور کامل تأیید نشده‌اند.

چگونه شروع یا پایان یک جلسه مداوم

هنگام اتصال مشتری به کارگزار، مشتری می‌تواند یک جلسه مداوم درخواست کند. مشتری از یک پرچم clean Session استفاده می‌کند تا به کارگزار بگوید چه نوع جلسه‌ای نیاز دارد:

- هنگامی که پرچم clean Session روی true تنظیم می‌شود، مشتری یک جلسه مداوم نمی‌خواهد. اگر مشتری به هر دلیلی ارتباط خود را قطع کند، تمام اطلاعات و پیام‌هایی که از یک جلسه مداوم قبلی در صف قرار گرفته‌اند از بین می‌روند.
- هنگامی که پرچم clean Session روی false تنظیم می‌شود، کارگزار یک جلسه مداوم برای مشتری ایجاد می‌کند و تمام اطلاعات و پیام‌ها تا دفعه بعد که مشتری یک جلسه تمیز درخواست کند حفظ می‌شود. اگر پرچم clean Session روی false تنظیم شده باشد و کارگزار قبلاً یک جلسه برای مشتری در دسترس داشته باشد، از جلسه موجود استفاده می‌کند و پیام‌های قبلی در صف را به مشتری تحویل می‌دهد.

چگونه مشتری متوجه می‌شود که یک جلسه قبلاً ذخیره شده است؟

از MQTT 3.1.1، پیام CONNACK از کارگزار، حاوی پرچم session present است. این پرچم به مشتری می‌گوید که آیا یک جلسه از قبل ایجاد شده هنوز در کارگزار موجود است یا خیر.

جلسه مداوم در سمت مشتری

مشابه کارگزار، هر مشتری MQTT باید یک جلسه مداوم را نیز ذخیره کند. هنگامی که مشتری از سرور درخواست می‌کند تا داده‌های جلسه را نگه دارد، مشتری مسئول ذخیره اطلاعات زیر است:

- همه پیام‌های موجود در یک QoS 1 یا 2 که هنوز توسط کارگزار تأیید نشده‌اند.
- تمام پیام‌های QoS 2 دریافت شده از کارگزار که هنوز به طور کامل تأیید نشده‌اند.

برخی توصیه‌ها

در اینجا چند دستورالعمل وجود دارد که می‌تواند به شما کمک کند تا تصمیم بگیرید که چه زمانی از یک جلسه مداوم یا یک جلسه تمیز استفاده کنید:

جلسه مداوم

- مشتری باید همه پیام‌ها را از یک موضوع خاص دریافت کند، حتی اگر آفلاین باشد. کارگزار باید پیام‌های مشتری را در صف قرار دهد و به محض اینکه مشتری برخط شد، آنها را تحویل دهد.
- مشتری منابع محدودی دارد. کارگزار باید اطلاعات اشتراک مشتری را ذخیره کرده و ارتباط قطع شده را به سرعت بازیابی کند.
- مشتری باید پس از اتصال مجدد، تمام پیام‌های QoS 1 و 2 متوقف شده را ادامه کند.

جلسه تمیز

- مشتری فقط باید پیام‌های موضوعات را منتشر کند و نیازی به اشتراک در موضوعات ندارد. کارگزار لازم نیست اطلاعات جلسه را ذخیره کند و یا دوباره ارسال پیام‌های QoS 1 و 2 را امتحان کند.
- مشتری به دریافت پیام‌هایی که به صورت آفلاین از دست می‌دهد نیازی ندارد.

کارگزار پیام‌ها را چه مدت ذخیره می‌کند؟

کارگزار تا زمانی که مشتری دوباره آنلاین شود و پیام را دریافت کند جلسه را ذخیره می‌کند. با این حال، اگر مشتری برای مدت طولانی دوباره آنلاین نشود، چه اتفاقی می‌افتد؟ معمولاً محدودیت حافظه سیستم عامل، محدودیت اصلی در ذخیره‌سازی پیام است. هیچ پاسخ استانداردی برای این سناریو وجود ندارد و راه‌حل مناسب به مورد استفاده شما بستگی دارد.

بخش 8 ملزومات – پیام‌های حفظ شده

در MQTT، هر مشتری ناشر پیام، هیچ تضمینی ندارد که مشتری مشترک، واقعاً آن پیام را دریافت کند. مشتری ناشر فقط می‌تواند مطمئن شود که پیام به طور ایمن به کارگزار تحویل داده شده است. اساساً همین امر در مورد مشتری مشترک هم صادق است. هر مشتری که مشترک موضوعاتی می‌شود، در مورد اینکه مشتری ناشر، چه زمانی پیامی را در یکی از آن موضوعات منتشر می‌کند بی‌اطلاع است. ممکن است چند ثانیه، دقیقه یا چند ساعت طول بکشد تا ناشر، پیام جدیدی با یکی از موضوعات مشترک را ارسال کند. تا زمانی که پیام بعدی منتشر شود، مشتری مشترک در مورد وضعیت فعلی موضوع کاملاً در تاریکی است. این جا همان جایی است که پیام‌های حفظ شده یا Retained Messages وارد بازی می‌شوند.

پیام‌های حفظ شده

یک پیام حفظ شده، یک پیام MQTT معمولی است که پرچم retained آن روی true تنظیم شده است. کارگزار، آخرین پیام حفظ شده و QoS مربوطه برای آن موضوع را ذخیره می‌کند. هر مشتری که در یک الگوی موضوعی

که با موضوع پیام حفظ شده مطابقت دارد مشترک شود، بلافاصله پس از اشتراک، پیام حفظ شده را دریافت می‌کند. کارگزار، برای هر موضوع، تنها یک پیام حفظ شده را ذخیره می‌کند.اگر مشتری در الگوی موضوعی دارای wildcard مشترک باشد، حتی اگر با موضوع پیام حفظ‌شده، دقیقاً مطابقت نداشته باشد، یک پیام حفظ شده را دریافت می‌کند. مثلا اگر مشتری A یک پیام حفظ شده به myhome/livingroom/temperature منتشر کند و کمی بعد، مشتری B مشترک myhome/# شود، مشتری B پیام myhome/livingroom/temperature را مستقیماً پس از اشتراک دریافت می‌کند. مشتری B (مشتری مشترک) می‌تواند ببیند که این پیام، یک پیام حفظ شده است زیرا کارگزار پیام‌های حفظ شده را با پرچم retained روی true ارسال می‌کند. مشتری می‌تواند تصمیم بگیرد که چگونه می‌خواهد پیام‌های حفظ شده را پردازش کند. پیام‌های حفظ‌شده به مشتریانی که به تازگی مشترک شده‌اند کمک می‌کنند تا بلافاصله پس از اشتراک در یک موضوع، وضعیت به‌روزرسانی را دریافت کنند. پیام حفظ شده، انتظار برای ارسال به روز رسانی بعدی ناشرین را از بین می‌برد. به عبارت دیگر، یک پیام حفظ شده در مورد یک موضوع، یک last known good value است. لازم نیست که پیام حفظ شده، آخرین مقدار باشد، اما باید آخرین پیام با پرچم retained true باشد. درک این نکته مهم است که پیام حفظ شده، هیچ ارتباطی با جلسات مداوم ندارد. هنگامی که یک پیام حفظ شده توسط کارگزار ذخیره می‌شود، تنها یک راه برای حذف آن وجود دارد که متعاقبا گفته خواهد شد.

ارسال یک پیام حفظ شده

از دیدگاه یک توسعه دهنده، ارسال یک پیام حفظ شده بسیار ساده است و فقط باید پرچم retained آن پیام true باشد. معمولا این کار در کتابخانه مرتبطش راحت است.

حذف یک پیام حفظ شده

برای حذف پیام حفظ شده یک موضوع، یک راه ساده وجود دارد: ارسال یک پیام حفظ شده با payload صفر بایت، پیام حفظ شده قبلی را حذف می‌کند. کارگزار، پیام حفظ شده را حذف کرده و مشترکین جدید دیگر پیامی برای آن موضوع دریافت نمی‌کنند. اغلب، حتی نیازی به حذف نیست، زیرا هر پیام حفظ شده جدید، پیام قبلی را بازنویسی می‌کند.چرا و چه زمانی باید از پیام‌های حفظ شده استفاده کرد؟ یک پیام حفظ شده، زمانی معنا پیدا می‌کند که بخواهید مشترکین تازه متصل شده، فوراً آن را دریافت کنند (بدون اینکه منتظر بمانند تا پیام بعدی توسط مشتری ناشر ارسال شود). این امر برای به‌روزرسانی وضعیت اجزا یا دستگاه‌ها در موضوعات جداگانه بسیار مفید است. به عنوان مثال، وضعیت device1 در موضوع myhome/devices/device1/status است. هنگامی که از پیام‌های حفظ شده استفاده می‌شود، مشترکین جدید موضوع، بلافاصله پس از اشتراک، وضعیت (آنلاین / آفلاین)

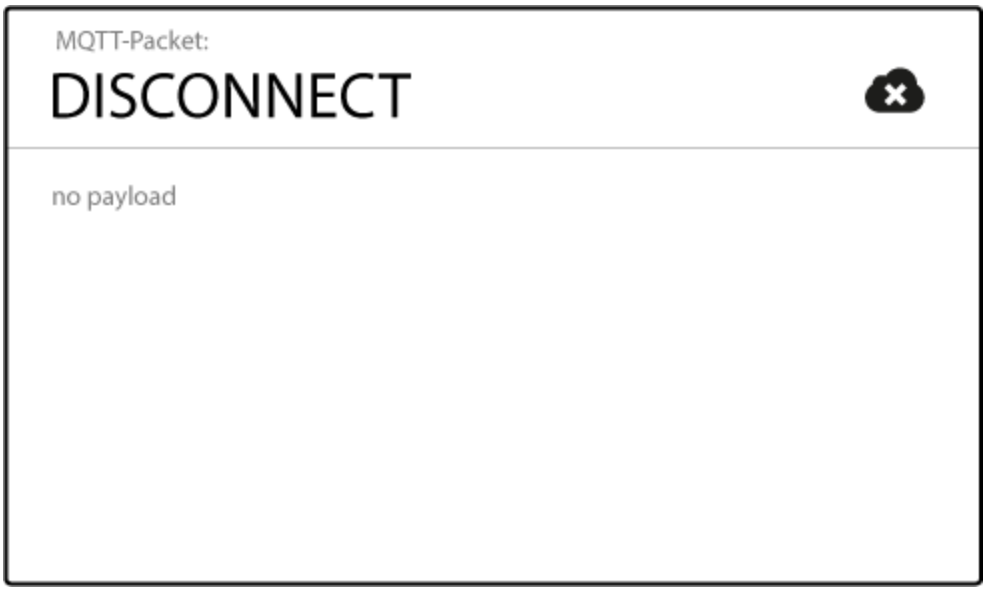
دستگاه را دریافت می‌کنند. همین امر در مورد مشتریانی که داده‌های فواصل زمانی، دما، مختصات GPS و ... را ارسال می‌کنند صادق است. بدون پیام‌های حفظ شده، مشترکین جدید، بین فواصل انتشار در بی خبری نگه داشته می‌شوند. استفاده از پیام‌های حفظ‌شده کمک می‌کند تا آخرین اطلاعات باارزش، بلافاصله برای مشتری در حال اتصال ارسال شود.

بخش 9 ملزومات – Last Will and Testament

از آنجایی که MQTT اغلب در سناریوهایی استفاده می‌شود که شامل شبکه‌های غیرقابل اعتماد هستند، منطقی است که فرض کنیم برخی از مشتریان MQTT در این سناریوها گهگاه به‌طور ناخوشایند قطع می‌شوند. قطع ناخوشایند ممکن است به دلیل قطع اتصال، خالی بودن باتری و یا بسیاری از دلایل دیگر رخ دهد. دانستن اینکه آیا یک مشتری به خوبی (با یک پیام قطع اتصال MQTT) قطع شده است یا به طور ناخوشایند (بدون پیام قطع)، به شما کمک می‌کند تا به درستی پاسخ دهید. ویژگی آخرین وصیت و عهد یا Last Will and Testament راهی را برای مشتریان فراهم می‌کند تا به قطع‌های ناخوشایند به روشی مناسب پاسخ دهند.

LWT

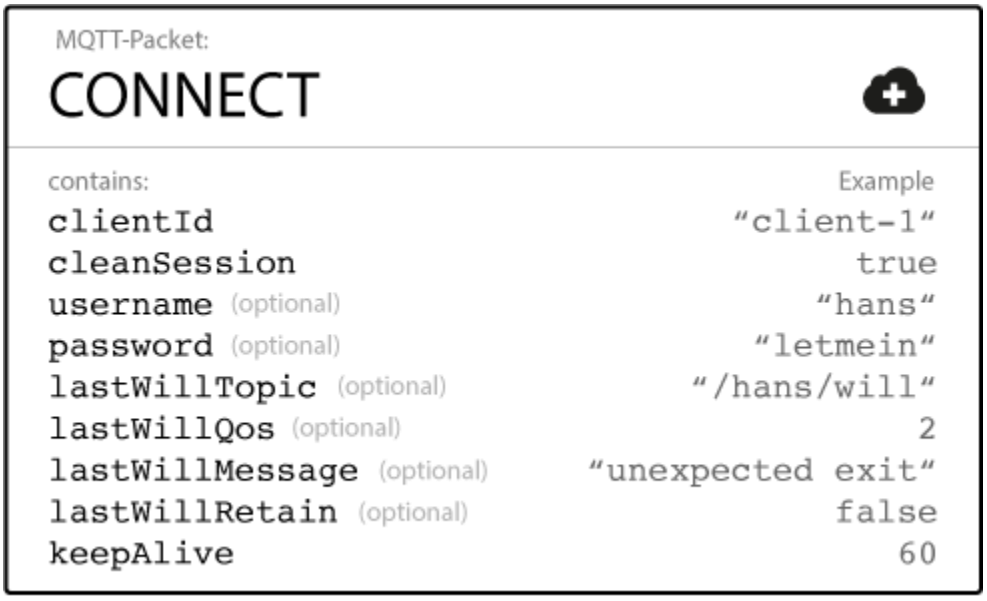
در MQTT، از ویژگی LWT استفاده می‌شود تا در مورد یک مشتری قطع شده به طور ناخوشایند، به مشتریان دیگر اطلاع داده شود. هر مشتری می‌تواند آخرین پیام Will خود را هنگام اتصال به یک کارگزار مشخص کند. آخرین پیام Will ، یک پیام معمولی MQTT با یک موضوع، پرچم پیام حفظ شده، QoS و payload است. کارگزار تا زمانی که تشخیص دهد که مشتری به طور ناخوشایند قطع شده است پیام را ذخیره می‌کند. در پاسخ به قطع ناخوشایند، کارگزار، آخرین پیام Will را برای همه مشترکین موضوع پیام ارسال می‌کند. اگر مشتری با یک پیام DISCONNECT صحیح، ارتباط خود را به خوبی قطع کند، کارگزار، پیام LWT ذخیره شده را دور می‌ریزد.



LWT به شما کمک می‌کند تا زمانی که اتصال یک مشتری قطع می‌شود، راهبردهای مختلفی را پیاده‌سازی کنید (یا حداقل، سایر مشتریان را در مورد وضعیت آفلاین مطلع کنید).

چگونه یک پیام LWT برای یک مشتری مشخص می‌شود؟

مشتریان می‌توانند یک پیام LWT را در پیام CONNECT که ارتباط بین مشتری و کارگزار را آغاز می‌کند تعیین کنند.



چه زمانی یک کارگزار، پیام LWT را ارسال می‌کند؟

طبق مشخصات MQTT 3.1.1، کارگزار باید LWT یک مشتری را در شرایط زیر ارسال کند:

- کارگزار، خطای I/O یا خرابی شبکه را تشخیص دهد.
- مشتری نتواند در مدت زمان تعریف شده Keep Alive ، ارتباط برقرار کند.
- مشتری، قبل از بستن اتصال شبکه، بسته DISCONNECT را ارسال نکند.
- کارگزار به دلیل خطای پروتکلی، اتصال شبکه را ببندد.

چه زمانی باید از LWT استفاده کرد؟

LWT یک راه عالی برای اطلاع سایر مشترکین در مورد قطع غیرمنتظره اتصال یک مشتری دیگر است. در سناریوهای دنیای واقعی، LWT اغلب با پیام‌های حفظ شده ترکیب می‌شود تا وضعیت یک مشتری در یک موضوع خاص را ذخیره کند. به عنوان مثال، ابتدا client1 یک پیام CONNECT با یک lastWillMessage که دارای payload با مقدار آفلاین است به کارگزار ارسال می‌کند، پرچم lastWillRetain روی true و lastWillTopic روی client1/status تنظیم شده است. در مرحله بعد، مشتری یک پیام PUBLISH که payload آن مقدار آنلاین دارد و پرچم حفظ شده نیز در همان موضوع (client1/status) روی true تنظیم شده است ارسال می‌کند. تا زمانی که client1 متصل بماند، مشتریانی که به تازگی در موضوع client1/status مشترک شده‌اند، پیام حفظ شده "آنلاین" را دریافت می‌کنند. اگر client1 به طور غیر منتظره قطع شود، کارگزار، پیام LWT را با payload "آفلاین" به عنوان پیام حفظ شده جدید منتشر می‌کند. مشتریانی که در حالی که client1 آفلاین است در موضوع مشترک می‌شوند، پیام حفظ شده LWT ("آفلاین") را از کارگزار دریافت می‌کنند. این الگوی پیام‌های حفظ شده، سایر مشتریان را در مورد وضعیت فعلی client1 در یک موضوع خاص به‌روز نگه می‌دارد.

بخش 10 ملزومات – Keep Alive و Client Take Over

در این بخش در مورد ویژگی Keep Alive و اینکه چرا این ویژگی برای شبکه‌های تلفن همراه مهم است صحبت می‌شود.

مشکل اتصالات TCP نیمه باز

MQTT بر اساس پروتکل TCP است. این پروتکل تضمین می‌کند که بسته‌ها از طریق اینترنت به روشی "قابل اطمینان، مرتب و بررسی شده" منتقل می‌شوند. با این وجود، هر از چند گاهی، انتقال بین طرفین ارتباط می‌تواند ناهماهنگ باشد; مثلا هنگامی که یکی از طرفین از کار بیفتد و یا خطاهای انتقال داشته باشد. در TCP به این حالت اتصال ناقص، اتصال نیمه باز یا half-open connection می‌گویند. نکته مهمی که باید به خاطر داشت این است که یک طرف ارتباط به کار خود ادامه می‌دهد و از خرابی طرف دیگر مطلع نمی‌شود. طرفی که هنوز متصل است به ارسال پیام ادامه می‌دهد و منتظر تأیید باقی می‌ماند. همانطور که اندی استنفورد کلارک (مخترع پروتکل MQTT) اشاره می‌کند، مشکل اتصالات نیمه باز در شبکه‌های تلفن همراه افزایش می‌یابد:

اگرچه در تئوری، زمانی که سوکت خراب می‌شود TCP/IP به شما اطلاع می‌دهد، اما در عمل، به‌ویژه در مواردی مانند ارتباطات تلفن همراه و ماهواره، کاملاً محتمل است که یک جلسه TCP به یک "سیاه چاله" تبدیل شود و بسته‌ها را هدر دهد.

MQTT Keep Alive

MQTT شامل یک تابع زنده نگه داشتن است که راه‌حلی برای مشکل اتصالات نیمه باز ارائه می‌دهد و یا حداقل، ارزیابی اینکه آیا اتصال هنوز باز است یا خیر را ممکن می‌سازد.Keep alive تضمین می‌کند که ارتباط بین کارگزار و مشتری همچنان باز است و کارگزار و مشتری از اتصال به هم آگاه هستند. هنگام اتصال مشتری به کارگزار، مشتری در یک بازه زمانی ثانیه‌ای با کارگزار ارتباط برقرار می‌کند. این فاصله، حداکثر مدت زمانی را که کارگزار و مشتری ممکن است با یکدیگر ارتباط برقرار نکنند مشخص می‌کند.

مشخصات MQTT می‌گوید:

مقدار Keep Alive ، حداکثر فاصله زمانی بین نقطه‌ای که مشتری، ارسال یک بسته کنترلی را تمام می‌کند و نقطه‌ای که ارسال بسته بعدی را شروع می‌کند است. این مسئولیت مشتری است که اطمینان حاصل کند فاصله زمانی بین ارسال بسته‌های کنترلی از مقدار Keep Alive بیشتر نشود. در صورت عدم ارسال بسته‌های کنترلی، مشتری باید یک بسته PINGREQ ارسال کند.

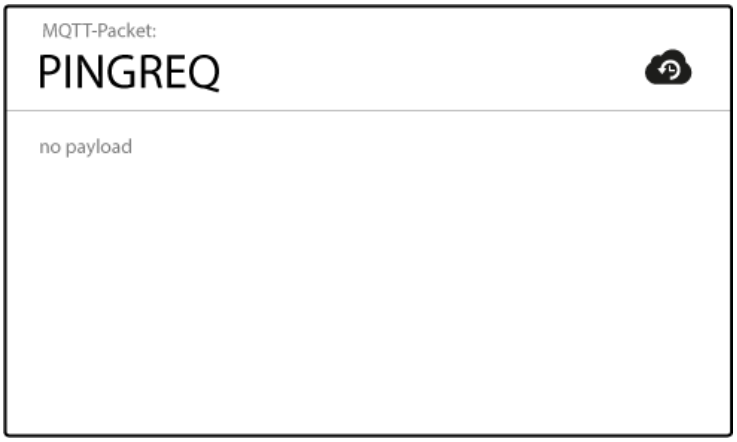
تا زمانی که پیام‌ها به طور مکرر ردوبدل می‌شوند و فاصله زمانی keep-alive رعایت می‌شود، نیازی به ارسال پیام اضافی برای تعیین اینکه آیا اتصال هنوز باز است یا خیر وجود ندارد. اگر مشتری در طول دوره keep-alive پیامی

ارسال نکند، باید یک بسته PINGREQ را برای کارگزار ارسال کند تا تأیید کند که در دسترس است و مطمئن شود که کارگزار نیز همچنان در دسترس است.

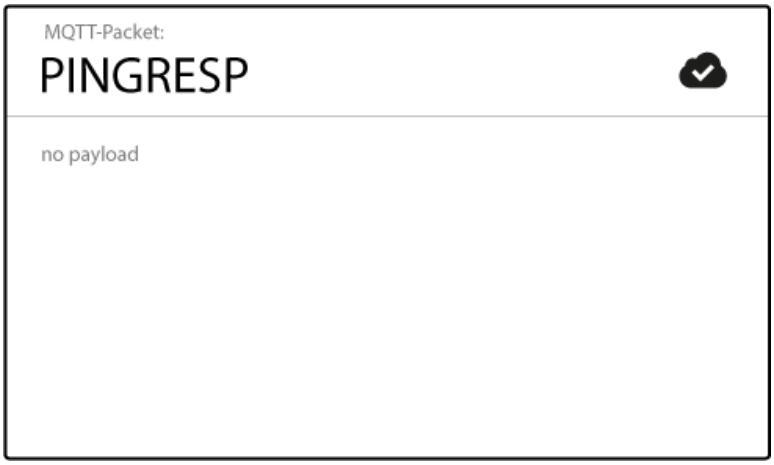
کارگزار باید آن مشتری را که پیام یا بسته PINGREQ در یک و نیم برابر فاصله زمانی keep alive ارسال نمی‌کند قطع کند. به همین ترتیب، انتظار می‌رود مشتری در صورت عدم دریافت پاسخ از طرف کارگزار در مدت زمان معقول، اتصال را ببندد.

Keep Alive Flow

بیایید به پیام‌های keep alive نگاهی دقیق‌تر بیندازیم. ویژگی keep alive از دو بسته استفاده می‌کند:



PINGREQ توسط مشتری ارسال می‌شود و به کارگزار نشان می‌دهد که مشتری هنوز زنده است. اگر مشتری، هیچ نوع بسته دیگری (مثلاً بسته PUBLISH یا SUBSCRIBE) ارسال نکند، مشتری باید بسته PINGREQ را برای کارگزار بفرستد. مشتری می‌تواند هر زمان که بخواهد یک بسته PINGREQ ارسال کند تا تأیید کند که اتصال شبکه هنوز زنده است. بسته PINGREQ حاوی payload نیست.



هنگامی که کارگزار، یک بسته PINGREQ را دریافت می‌کند، کارگزار باید با یک بسته PINGRESP پاسخ دهد تا به مشتری نشان دهد که هنوز در دسترس است. بسته PINGRESP نیز حاوی payload نیست. خوب است بدانید

- اگر کارگزار PINGREQ یا هر بسته دیگری را از مشتری دریافت نکند، کارگزار اتصال را بسته و آخرین پیام LWT را ارسال می‌کند (اگر مشتری یک LWT را مشخص کرده باشد).
- این مسئولیت مشتری MQTT است که یک مقدار مناسب keep alive را تعیین کند. به عنوان مثال، مشتری می‌تواند keep alive را با قدرت سیگنال فعلی خود تنظیم کند.
- حداکثر مقدار keep alive ، 18 ساعت و 12 دقیقه و 15 ثانیه است.
- اگر keep alive 0 باشد، عملا سازوکار آن غیرفعال می‌شود.

Client Take-Over

معمولاً یک مشتری قطع شده سعی می‌کند دوباره وصل شود. گاهی اوقات، کارگزار هنوز یک اتصال نیمه باز برای مشتری دارد. در MQTT، اگر کارگزار، یک اتصال نیمه باز را تشخیص دهد، "تصرف مشتری" یا Client Take-Over را انجام می‌دهد. کارگزار، اتصال قبلی به همان مشتری (که توسط شناسه مشتری تعیین می‌شود) را می‌بندد و یک ارتباط جدید با مشتری برقرار می‌کند. این رفتار تضمین می‌کند که اتصال نیمه باز، مانع از برقراری مجدد اتصال مشتری قطع شده نمی‌شود.

تغییرات بنیادی MQTT 5

نسخه جدید پروتکل، تمام ویژگی‌هایی را که در موفقیت آن نقش داشته‌اند حفظ کرده است: سبک بودن، ارتباطات push، ویژگی‌های منحصر به فرد، سهولت استفاده، مقیاس‌پذیری بالا، مناسب بودن برای شبکه‌های تلفن همراه و جداسازی شرکت‌کنندگان در ارتباط. در ادامه، تغییرات جدید MQTT 5 بررسی شده‌اند.

هدرهای سفارشی و کدهای دلیل

یکی از هیجان‌انگیزترین و منعطف‌ترین ویژگی‌های جدید MQTT 5، امکان افزودن ویژگی‌های key-value سفارشی در هدر MQTT است. مشابه پروتکل‌هایی مانند HTTP، مشتریان و کارگزاران MQTT می‌توانند تعداد دلخواهی از هدرهای سفارشی (یا از پیش تعریف شده) را برای حمل آبَرداده اضافه کنند. این ابرداده‌ها را می‌توان برای داده‌های خاص برنامه استفاده کرد. هدرهای از پیش تعریف شده برای پیاده‌سازی بیشتر ویژگی‌های جدید MQTT استفاده می‌شوند.

بسیاری از بسته‌های MQTT اکنون شامل کدهای دلیل یا Reason Codes نیز هستند. یک کد دلیل نشان می‌دهد که یک خطای پروتکل از پیش تعریف شده رخ داده است. این کدهای دلیل معمولاً روی بسته‌های تأیید

(acknowledgement packets) حمل می‌شوند و به مشتری و کارگزار اجازه می‌دهند تا شرایط خطا را تفسیر کنند. کدهای دلیل را گاهی اوقات Negative Acknowledgements می‌نامند. بسته‌های MQTT زیر می‌توانند کدهای دلیل را حمل کنند:

CONNACK
PUBACK
PUBREC
PUBREL
PUBCOMP
SUBACK
UNSUBACK
AUTH
DISCONNECT

کدهای دلیل برای Negative Acknowledgements از «Quota Exceeded» تا «خطای پروتکل» متغیر هستند. مشتریان و کارگزاران، مسئول تفسیر این کدهای جدید هستند.

کدهای بازگشتی CONNACK برای ویژگی‌های پشتیبانی نشده

با محبوبیت MQTT، بسیاری از پیاده‌سازی‌های MQTT توسط شرکت‌ها ایجاد و ارائه شده است. باید توجه داشت که همه این پیاده‌سازی‌ها کاملاً با مشخصات MQTT سازگار نیستند، زیرا گاهی اوقات ویژگی‌هایی مانند QoS 2، پیام‌های حفظ شده (retained messages) و یا جلسات مداوم (persistent sessions) پیاده‌سازی نمی‌شوند. البته پیاده‌سازی HiveMQ کاملاً مطابق با مشخصات MQTT است و از همه ویژگی‌ها پشتیبانی می‌کند.5 MQTT راهی برای پیاده‌سازی ناقص MQTT (همان طور که اغلب در پیشنهادات SaaS یافت می‌شود) ارائه می‌دهد تا نشان دهد که کارگزار از ویژگی‌های خاصی پشتیبانی نمی‌کند. این وظیفه مشتری است که مطمئن شود هیچ یک از این ویژگی‌های پشتیبانی نشده، استفاده نمی‌شوند. در پیاده‌سازی کارگزار، از هدرهای از پیش تعریف شده در بسته CONNACK (که توسط کارگزار پس از ارسال بسته CONNECT توسط مشتری ارسال می‌شود) استفاده می‌شود تا نشان دهد که ویژگی‌های خاصی پشتیبانی نمی‌شوند. البته می‌توان از این هدرها برای ارسال اعلان به مشتری، مبنی بر عدم وجود مجوز استفاده از ویژگی‌های خاصی نیز استفاده کرد.

جلسات تمیز یا Clean Sessions و شروع پاک یا Clean Start

یک عملکرد محبوب MQTT 3.1.1 استفاده از جلسات تمیز (clean sessions) توسط مشتریان MQTT است که اتصال موقت دارند یا اصلاً مشترک پیام‌ها نیستند. هنگام اتصال به کارگزار، مشتری باید انتخاب می‌کرد که یک بسته CONNECT با پرچم cleanSession فعال یا غیرفعال ارسال کند. با یک جلسه تمیز، یک مشتری MQTT

نشان می‌دهد که باید به محض اینکه اتصال TCP اصلی قطع شد یا اگر مشتری تصمیم به قطع ارتباط با کارگزار داشته باشد، کارگزار باید هر گونه داده برای مشتری را دور بریزد. همچنین، اگر یک جلسه قبلی مرتبط با شناسه مشتری در کارگزار وجود داشته باشد، یک بسته clean Session CONNECT ، کارگزار را مجبور به حذف داده‌های قبلی می‌کند. با MQTT v5، یک مشتری می‌تواند از یک شروع پاک (Clean Start) استفاده کند (که با پرچم شروع پاک در پیام CONNECT نشان داده می‌شود). هنگام استفاده از این پرچم، کارگزار هرگونه داده جلسه قبلی را حذف می‌کند و مشتری با یک جلسه جدید شروع می‌کند. پس از بسته شدن اتصال TCP بین سرویس گیرنده و سرور، جلسه به طور خودکار پاک نمی‌شود. برای شروع حذف جلسه پس از قطع ارتباط مشتری، یک فیلد هدر جدید به نام "Session Expiry Interval" می‌باید با 0 مقداردهی شود. از آنجا که Clean Start جدید، انعطاف‌پذیری بیشتری را امکان‌پذیر ساخته و اجرای آن آسان‌تر از مفهوم جلسه clean Session/persistent است، مدیریت جلسات MQTT ساده شده است. با MQTT 5 همه جلسات پایدار هستند مگر اینکه Session Expiry Interval یا "فاصله انقضای جلسه" 0 باشد. حذف یک جلسه، زمانی رخ می‌دهد که پس از مهلت زمانی باشد یا مشتری مجدداً با شروع پاک وصل شود.

بسته MQTT اضافی

MQTT 5 یک بسته جدید MQTT را معرفی کرده است: بسته AUTH. این بسته جدید برای پیاده‌سازی مکانیسم‌های احراز هویت غیر پیش‌پاافتاده (non-trivial authentication mechanisms) بسیار مفید است و ما انتظار داریم که این بسته در محیط‌های تولید، استفاده زیادی داشته باشد. درک این نکته مهم است که این بسته جدید هم می‌تواند توسط کارگزاران و مشتریان، پس از برقراری اتصال برای استفاده از روش‌های پیچیده احراز هویت چالش/پاسخ (challenge/response) مانند SCRAM یا Kerberos ارسال شود، و هم می‌تواند برای روش‌های احراز هویت پیشرفته مانند Oauth استفاده شود. این بسته همچنین امکان احراز هویت مجدد مشتریان MQTT را بدون بستن اتصال فراهم می‌کند.

نوع داده جدید: جفت رشته UTF-8

ظهور هدرهای سفارشی، همچنین مستلزم معرفی یک نوع داده جدید بود. جفت رشته UTF-8. این جفت رشته در اصل، یک ساختار کلید-مقدار است که هر دوی کلید و مقدار از نوع داده رشته‌ای هستند. این نوع داده در حال حاضر فقط برای هدرهای سفارشی استفاده می‌شود. با این نوع داده جدید، MQTT از 7 نوع داده مختلف استفاده می‌کند:

Bit
Two Byte Integer

Four Byte Integer
UTF-8 Encoded String
Variable Byte Integer
Binary Data
UTF-8 String Pair

اکثر کاربران برنامه‌ها معمولاً از داده‌های باینری و رشته‌های رمزگذاری شده UTF-8 در APIهای کتابخانه MQTT خود استفاده می‌کنند. با MQTT 5، جفت رشته‌های UTF-8 نیز می‌توانند به طور مکرر استفاده شوند. همه انواع داده‌های دیگر از دید کاربر پنهان هستند، اما برای ایجاد بسته‌های معتبر MQTT توسط کتابخانه‌های مشتری و کارگزاران MQTT استفاده می‌شوند.

بسته‌های دوطرفه DISCONNECT

با MQTT 3.1.1، مشتری می‌تواند نشان دهد که می‌خواهد با ارسال یک بسته DISCONNECT قبل از بستن اتصال TCP زیربنایی و اصلی، اتصال را به خوبی قطع کند. قبلا هیچ راهی برای کارگزار MQTT وجود نداشت که به مشتری MQTT اطلاع دهد که اتفاق بدی افتاده است و کارگزار قصد دارد اتصال TCP را ببندد. این امکان با نسخه جدید پروتکل ایجاد شده است. اکنون کارگزار مجاز است قبل از بستن سوکت، یک بسته MQTT DISCONNECT ارسال کند. مشتری نیز می‌تواند دلیل قطع شدن را تفسیر کرده و اقدام مربوطه را انجام دهد. گرچه یک کارگزار، نیازی به اعلام دلیل دقیق ندارد (مثلاً به دلایل امنیتی)، اما این برای توسعه برنامه‌ها کمک زیادی می‌کند تا حداقل بدانیم چرا یک اتصال، توسط کارگزار بسته شده است.همچنین بسته‌های DISCONNECT می‌توانند حاوی کد دلیل یا Reason Code باشند، بنابراین به راحتی می‌توان دلیل قطع اتصال را مشخص کرد (مثلاً در صورت وجود مجوزهای نامعتبر).

عدم ارسال مجدد پیام‌های QoS 1 and 2

مشتری‌های MQTT از اتصالات TCP (یا پروتکل‌های مشابه با ضمانت‌های یکسان) به عنوان اتصال اصلی و زیربنایی استفاده می‌کنند. یک اتصال TCP سالم، ارتباط دوطرفه با ضمانت‌های دقیقاً یک‌بار و مرتب را ضمانت می‌کند، بنابراین تمام بسته‌های MQTT ارسال شده توسط مشتریان یا کارگزاران، به سمت دیگر می‌رسند. در صورتی که اتصال TCP قطع شود، در حالی که پیام در حال ارسال است، QoS 1 and 2 تحویل پیام را از طریق چندین اتصال TCP تضمین می‌کنند.MQTT 3.1.1 اجازه تحویل مجدد پیام‌های MQTT را در حالی که اتصال TCP سالم است، می‌دهد. البته در عمل، این ایده خوبی نیست، چرا که ممکن است مشتریان MQTT بیش از حد اطلاعات دریافت کنند. مثلا حالتی را تصور کنید که در آن یک مشتری MQTT پیامی را از یک کارگزار MQTT دریافت می‌کند و برای پردازش آن به 11 ثانیه زمان نیاز دارد (و پس از پردازش بسته می‌خواهد آن را تأیید کند). حال اگر کارگزار،

پیام را پس از 10 ثانیه مجددا ارسال کند، هیچ مزیتی برای این رویکرد وجود ندارد و فقط پهنای باند، هدر رفته و مشتری MQTT نیز بیش از حد اطلاعات دریافت است. در MQTT 5، کارگزاران و مشتریان، مجاز به ارسال مجدد پیام‌های MQTT برای اتصالات سالم TCP نیستند. هرچند هنگامی که اتصال TCP بسته شد، کارگزاران و مشتریان باید بسته‌های تأیید نشده را دوباره ارسال کنند. بنابراین تضمین‌های QoS 1 and 2 به‌اندازه MQTT 3.1.1 مهم هستند. اگر در موارد استفاده خود، به بسته‌های ارسال مجدد نیاز دارید (مثلاً چون پیاده‌سازی شما بسته‌ها را در موارد خاص تأیید نمی‌کند)، پیشنهاد می‌کنیم قبل از ارتقا به MQTT v5 در این تصمیم تجدید نظر کنید.

استفاده از رمزهای عبور بدون نام کاربری

MQTT 3.1.1 هنگام استفاده از رمز عبور در بسته CONNECT ، مشتری MQTT را ملزم به ارسال یک نام کاربری می‌کرد. برای موارد استفاده خاص که نام کاربری وجود نداشته باشد، این امر بسیار ناخوشایند بود. مثلا هنگام استفاده از OAuth که از توکن وب JSON به عنوان تنها اطلاعات احراز هویت و مجوز استفاده می‌شود این امر، مشکل‌ساز است. هنگام استفاده از چنین توکنی در MQTT 3.1.1، اغلب نام‌های کاربری ثابت استفاده می‌شد، چرا که تنها اطلاعات مرتبط در بخش رمز عبور بود. در MQTT 5 راه‌های ظریف‌تری برای حمل توکن‌ها وجود دارد (مثلاً از طریق بسته AUTH)، و البته هنوز هم می‌توان از فیلد رمز عبور بسته CONNECT استفاده کرد. اکنون کاربران می‌توانند از رمز عبور استفاده کنند و دیگر نیازی به تعیین نام کاربری ندارند.

منابع:

https://mqtt.org

https://www.hivemq.com