**Comparison of Asynchronous Messaging Technologies: JMS, AMQP, and MQTT**

Messaging has been the fundamental communication mechanism which has been succeeded all over the world. Either it is human to human, machine to human or machine to machine, messaging has been the single common method of communication. There are 2 fundamental mechanisms we used to exchange messages between 2(or more) parties.

1. Synchronous messaging
2. Asynchronous messaging

مقایسه فن آوری های پیام رسانی ناهمزمان: JMS ، AMQP و MQTT

پيام رساني سازوکار اساسي ارتباطاتي است که در سرتاسر جهان موفق شده است. چه از نظر انسان به انسان و چه از نظر دستگاه به انسان و یا از دستگاه به دستگاه دیگر ، پیام رسانی یک روش متداول ارتباطی است. 2 مکانیسم اساسی وجود دارد که ما برای تبادل پیام بین 2 (یا بیشتر) طرف استفاده کردیم.

1. پیام رسانی همزمان

2. پیام رسانی ناهمزمان

Synchronous messaging is used when the message sender expects a response to the message within a specified time period and waiting for that response to carry out his next task. Basically he “blocks” until he receives the response.

Asynchronous messaging means that sender does not expect an immediate response and does not “blocks” waiting for the response. There can be a response or not, but the sender will carry out his remaining tasks.

از پیام رسانی همزمان استفاده می شود که فرستنده پیام در یک بازه زمانی مشخص پاسخی به پیام بدهد و منتظر انجام آن پاسخ برای انجام وظیفه بعدی خود باشد. اصولاً او "مسدود" می کند تا زمانی که پاسخ را دریافت کند.

پیام رسانی ناهمزمان به این معنی است که فرستنده انتظار پاسخ فوری را ندارد و در انتظار جواب "بلوکه" نمی شود. پاسخی وجود دارد یا نه ، اما فرستنده وظایف باقی مانده خود را انجام می دهد.

Out of the above mentioned technologies, Asynchronous messaging has been the widely used mechanism when it comes to machine to machine communication where 2 computer programs talk to each other. With the hype of the micro services architecture, it is quite evident that we need an asynchronous messaging model to build our services.

از بین فن آوری های ذکر شده ، پیام رسانی هم زمان مکانیزم بسیار مورد استفاده در ارتباط دستگاه به دستگاه است که در آن 2 برنامه رایانه ای با یکدیگر صحبت می کنند. با استفاده از نکات برجسته در معماری میکروسرویس ، کاملاً مشهود است که برای ساخت خدمات خود به یک مدل پیام رسانی ناهمزمان نیاز داریم.

This has been a fundamental problem in software engineering and different people and organizations have come up with different approaches. I will be describing about 3 of the most successful asynchronous messaging technologies which are widely used in the enterprise IT systems.

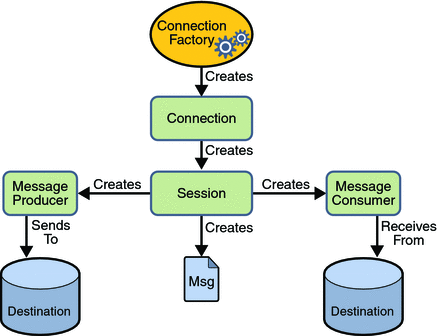
این یک مشکل اساسی در مهندسی نرم افزار بوده و افراد و سازمانهای مختلف با رویکردهای مختلفی روبرو شده اند. من در مورد 3 از موفق ترین فن آوری های پیام رسانی ناهمزمان که به طور گسترده در سیستم های IT شرکت بکار می رود ، توضیح خواهم داد.

## Java Messaging Service (JMS)

JMS has been one of most successful asynchronous messaging technology available. With the growth of the Java adoption of large enterprise applications, JMS has been the first choice for enterprise systems. It defines the API for building the messaging systems.

سرویس پیام رسانی جاوا (JMS)

JMS یکی از موفق ترین فناوری پیام رسانی ناهمزمان در دسترس است. با رشد پذیرش جاوا از برنامه های سازمانی بزرگ ، JMS اولین انتخاب برای سیستم های سازمانی بوده است. این API را برای ساختن سیستم های پیام رسانی تعریف می کند.



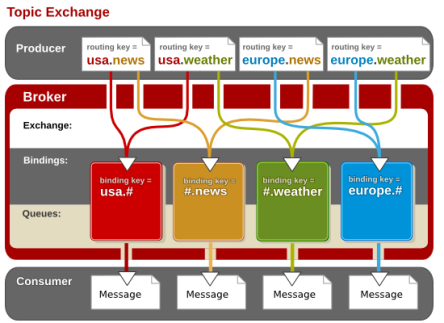
* Standard messaging API for JAVA platform
* Interoperability is only within Java and JVM languages like Scala, Groovy
* Does not worry about the wire level protocol
* Supports 2 messaging models with queues and topics
* Supports transactions
* Defines the message format (headers, properties and body)
* API پیام رسانی استاندارد برای پلت فرم JAVA
* قابلیت همکاری فقط در زبان های جاوا و JVM مانند Scala ، Groovy است
* از پروتکل سطح سیم نگران نباشید
* پشتیبانی از 2 مدل پیام رسانی با صف و موضوعات
* تراکنش ها را پشتیبانی می کند
* قالب پیام (هدرها ، خصوصیات و بدن) را تعریف می کند

## Advanced Message Queueing Protocol (AMQP)

JMS was awesome and people were happy about it. Microsoft came up with NMS (.Net Messaging Service) to support their platform and programming languages and it was working fine. But then comes the problem of interoperability. How 2 programs written in 2 different programming languages can communicate with each other over asynchronous messaging. Here comes the requirement to define a common standard for asynchronous messaging. There was no standard wire level protocol with JMS or NMS. Those will run on any wire level protocol but the API was bound with the programming language. AMQP addressed this issue and come up with a standard wire level protocol and many other features to support the interoperability and rich messaging needs for the modern applications.

پروتکل صف بندی پیام پیشرفته (AMQP)

JMS بسیار جذاب بود و مردم از آن خوشحال بودند. مایکروسافت برای پشتیبانی از سیستم عامل و زبانهای برنامه نویسی خود با NMS (سرویس پیام رسانی .net) همراه شد و خوب کار کرد. اما پس از آن مشکل تعامل پذیری پیش می آید. چگونه 2 برنامه که به 2 زبان برنامه نویسی مختلف نوشته شده اند می توانند از طریق پیام رسانی ناهمزمان با یکدیگر ارتباط برقرار کنند. در اینجا الزام به تعریف یک استاندارد مشترک برای پیام های ناهمزمان داده می شود. هیچ پروتکل سطح استاندارد سیم با JMS یا NMS وجود نداشت. این پروتکل ها با هر پروتکل سطح سیم اجرا می شوند اما API با زبان برنامه نویسی محدود شده است. AMQP به این موضوع پرداخته و با یک پروتکل استاندارد سطح سیم و بسیاری از ویژگی های دیگر برای پشتیبانی از قابلیت همکاری و نیازهای پیام رسانی غنی برای برنامه های جدید ارائه کرده است.



Here are the main features of AMQP:

* Platform independent wire level messaging protocol
* Consumer driven messaging
* Interoperable across multiple languages and platforms
* It is the wire level protocol
* Have 5 exchange types direct, fanout, topic, headers, system
* Buffer oriented
* Can achieve high performance
* Supports long lived messaging
* Supports classic message queues, round-robin, store and forward
* Supports transactions (across message queues)
* Supports distributed transactions (XA, X/Open, MS DTC)
* Uses SASL and TLS for security
* Supports proxy security servers
* Meta-data allows to control the message flow
* LVQ not supported
* Client and server are equal
* Extensible

در اینجا ویژگی های اصلی AMQP آورده شده است:

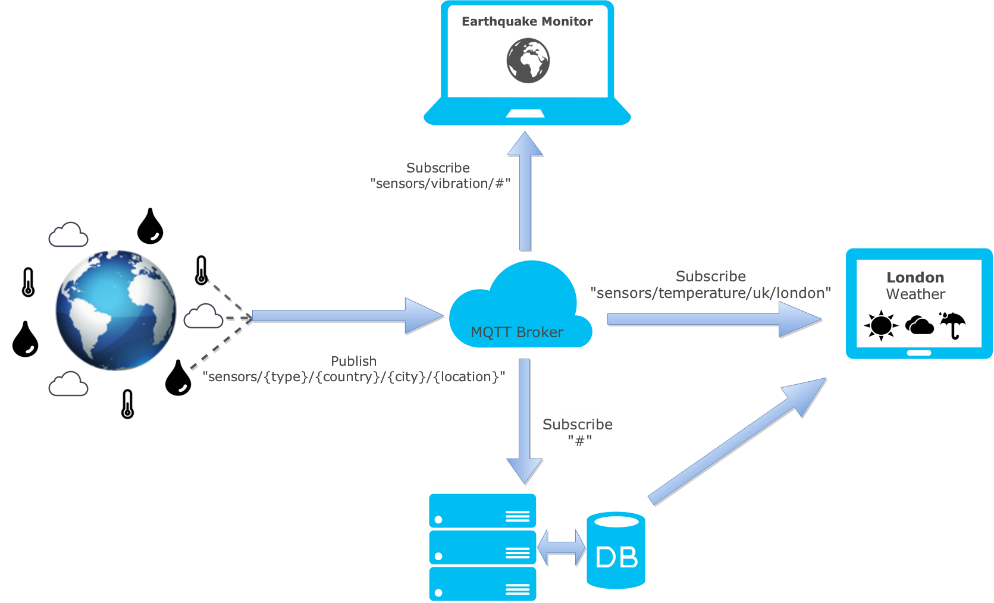
* پروتکل پیام رسان سطح پیام مستقل از پلتفرم
* پیام رسانی به مشتری
* قابلیت تعامل بین چندین زبان و سیستم عامل
* این پروتکل سطح سیم است
* دارای 5 نوع مبادله مستقیم ، fanout ، موضوع ، هدرها ، سیستم
* بافر گرا
* می تواند به کارایی بالایی دست یابد
* پشتیبانی از پیام های طولانی مدت
* پشتیبانی از صف های پیام کلاسیک ، round-robin ، store and forward
* پشتیبانی از transactions (در صف های پیام)
* پشتیبانی از transactionهای توزیع شده (XA, X/Open, MS DTC)
* برای امنیت از SASL و TLS استفاده می کند
* از سرورهای امنیتی پروکسی پشتیبانی می کند
* به داده های متا اجازه می دهد تا جریان پیام را کنترل کنید
* LVQ پشتیبانی نمی شود
* مشتری و سرور برابر هستند
* قابل توسعه

## Message Queueing Telemetry Transport (MQTT)

Now we have JMS for Java based enterprise applications and AMQP for all other application needs. Why do we need a 3rd technology? It is specifically for small guys. Devices with less computing power cannot deal with all the complexities of AMQP rather they want a simplified but interoperable way to communicate. This was the fundamental requirement for MQTT, but today, MQTT is one of the main components of Internet Of Things (IOT) eco system.

ارسال پیام از راه دور انتقال از راه دور (MQTT)

اکنون ما JMS را برای برنامه های کاربردی شرکت مبتنی بر جاوا و AMQP برای سایر نیازهای برنامه دیگر داریم. چرا ما به فناوری سوم احتیاج داریم؟ این مخصوص افراد کوچک است. دستگاه هایی که قدرت محاسبه کمتری دارند نمی توانند با تمام پیچیدگی های AMQP برقرارا کنند بلکه روشی ساده و با قابلیت ارتباط می خواهند. این نیاز اساسی برای MQTT بود ، اما امروز ، MQTT یکی از مؤلفه های اصلی اینترنت اشیاء Internet Of Things (IOT) است.



Here are the main features of MQTT:

* Stream oriented, low memory consumption
* Designed to be used for small dumb devices sending small messages over low bw networks
* No long lived store and forward support
* Does not allow fragmented messages (hard to send large messages)
* Supports publish-subscribe for topics
* No transactional support (only basic acknowledgements)
* Messaging is effectively ephemeral (short lived)
* Simple username, password based security without enough entropy
* No connection security supported
* Message is opaque
* Topic is global (one global namespace)
* Ability to support Last Value Queue (LVQ)
* Client and server are asymmetric
* Not possible to extend

در اینجا ویژگی های اصلی MQTT آورده شده است:

* جریان گرا با مصرف کم
* طراحی شده برای دستگاه های گنگ کوچک که پیام های کوچکی را از طریق شبکه های کم وزن تولید می کنند
* پشتیبانی بلند مدت از store and forward نمیکند
* پیامهای تکه تکه شده ارسال نمی کند (ارسال پیام های بزرگ دشوار است)
* پشتیبانی از publish-subscribe برای موضوعات
* عدم پشتیبانی از تراکنشها (فقط تاییدیه های اساسی)
* پیام رسانی بطور مؤثر و زودگذر (کوتاه مدت) است
* امنیت ساده مبتنی بر کلمه کاربری و گذرواژه بدون entropy کافی
* از امنیت اتصال پشتیبانی نمی شود
* پیام غیرشفاف است
* موضوع Global است (یک Global namespace)
* امکان پشتیبانی از صف ارزش آخر (LVQ)
* مشتری و سرور نامتقارن هستند
* امکان توسعه وجود ندارد

<https://dzone.com/articles/comparison-of-asynchronous-messaging-technologies>