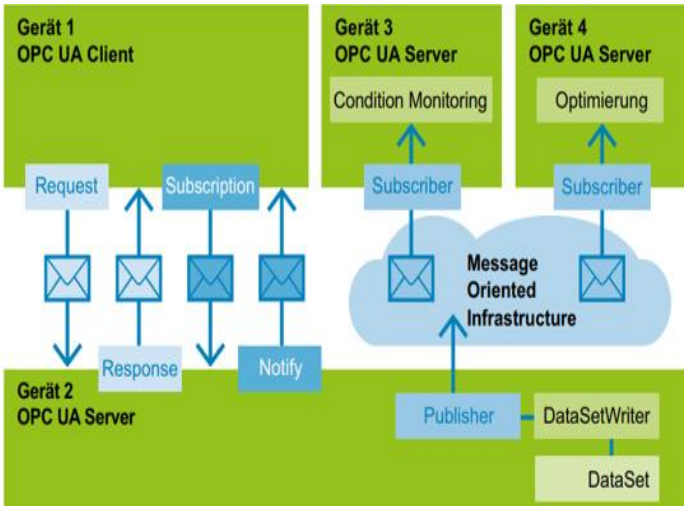
OPC UA vereint mehrere **Übertragungsmechanismen** in sich. Drei wichtige sollen nun beschrieben werden:

- „Request and Response“ - Prinzip
- „Subscription and Notify“ - Prinzip
- „Publish and Subscriber“ - Prinzip
- …

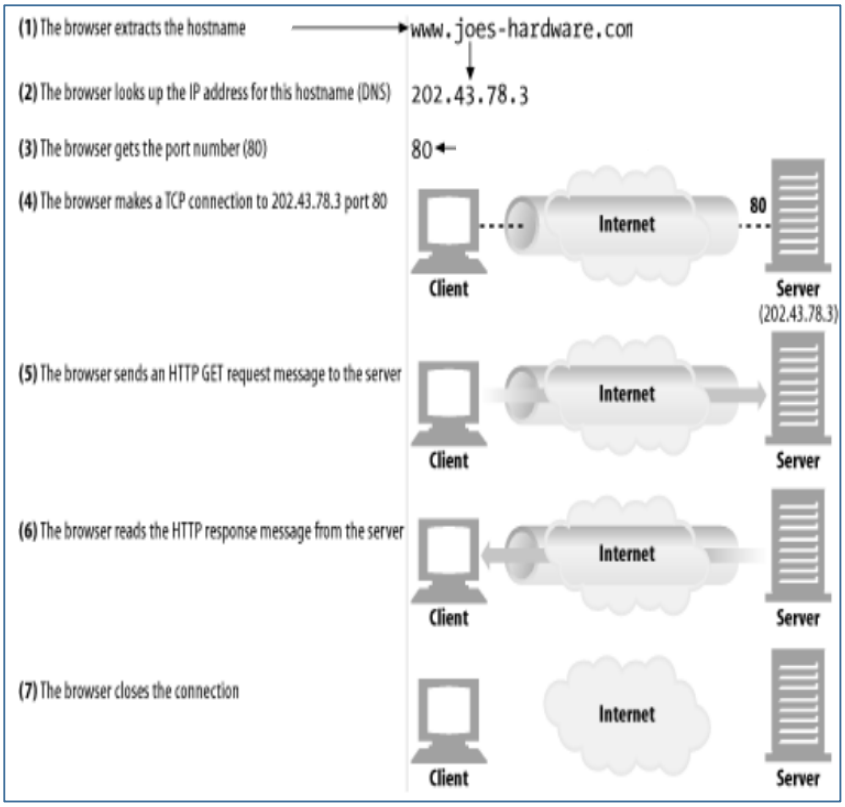
Neben dem aus der Netzwerktechnik bei z. B. TCP/ HTTP bekannten **„Request and Response“ (1)** existieren noch mehrere Mechanismen, die ständig ausgebaut werden. Bei der **Subscription (2)** werden nur im Falle von Änderungen (z. B. an Sensoren), benachrichtigt der Server den Client über solche Änderungen. Dieser Mechanismus reduziert die Menge der übertragenen Daten immens. **„Publish and Subscribe“ (3)** arbeitet zusätzlich noch mit dem sogenannten Broker.



1. „Request und Response“

Das aus der **Netzwerktechnik** bekannte „Request und Response“-Modell kommt z.B. bei der Datenübertragung bei HTTP zum Einsatz. Nachdem über den Port 80 eine TCP-Verbindung zum HTTP-Server mit einer bestimmten IP aufgebaut worden ist, fordert der Browser (HTTP-Client) über einen Request die Daten der entsprechenden Webseite an. Diese erhält er über den entsprechenden HTTP-Response vom Webserver (HTTP-Server).

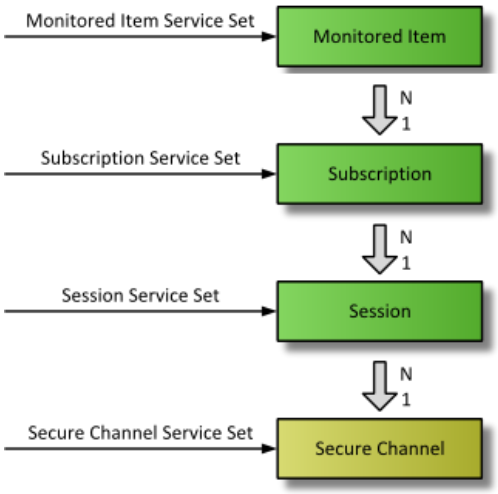
Quelle: oreilly.com



2. „Subscription and Notify“ (Abonnement)

Im Gegensatz zum permanenten Lesen von Informationen (Polling) bietet OPC UA eine elegantere Funktionalität, ein sogenanntes Abonnement (**Subscription**). Ein UA-Client kann eine Auswahl von Knoten von Interesse abonnieren und den Server diese Elemente überwachen lassen. Nur im Falle von Änderungen, benachrichtigt der Server den Client über solche Änderungen. Dieser Mechanismus reduziert die Menge der übertragenen Daten immens. Neben der Reduzierung der Bandbreite bietet dieser Mechanismus weitere Vorteile und ist der empfohlene Mechanismus zum „Lesen“ von Informationen von einem UA-Server.

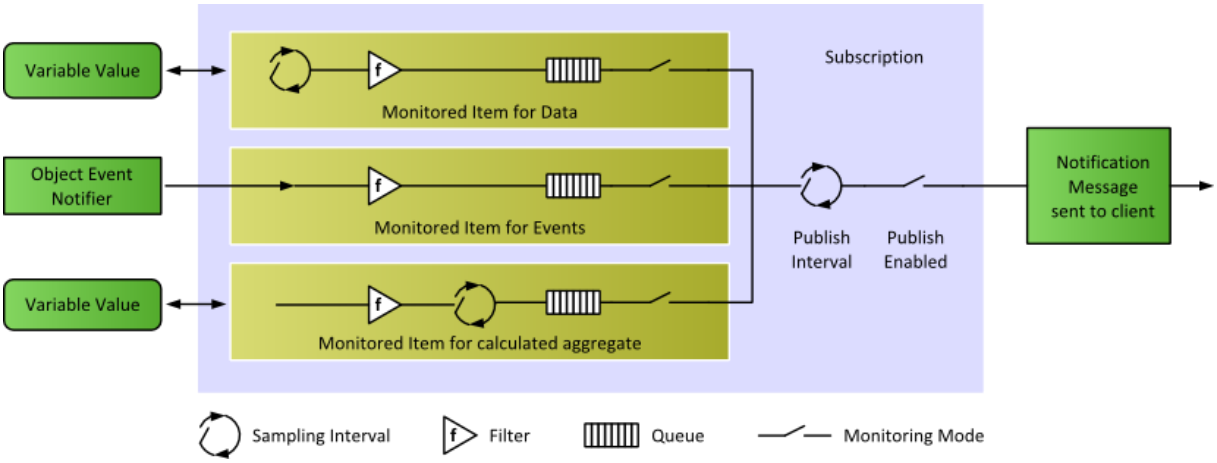
Ein Client kann verschiedene Arten von Informationen abonnieren, die von einem OPC UA-Server bereitgestellt werden. Der Zweck eines Abonnements besteht darin, diese Informationsquellen, die als überwachte Elemente **(Monitored Items**) bezeichnet werden, zu einer Information zusammenzufassen, die als Benachrichtigung bezeichnet wird.
Die folgende Abbildung zeigt die Dienste, die beteiligt sind, wenn ein Client Datenänderungen und Ereignisse abonniert.
Ein Abonnement besteht aus mindestens einem überwachten Element, muss im Rahmen einer Sitzung (**Session**) erstellt werden und kann auf eine andere Sitzung übertragen werden. Um eine Sitzung zu erstellen, muss ein sicherer Kanal zwischen dem Client und dem Server eingerichtet werden.



Es gibt drei verschiedene Arten von „Änderungen“, die ein Client abonnieren kann, wenn er dem Abonnement überwachte Elemente hinzufügt:

→Datenänderungen von Variablenwerten abonnieren (Wertattribut einer Variablen),
→Ereignisse von Objekten abonnieren (EventNotifier-Attribut eines Object & EventFilter-Sets),
→Aggregierte Werte, die in vom Kunden definierten Zeitintervallen basierend auf den aktuellen variablen Werten berechnet werden.

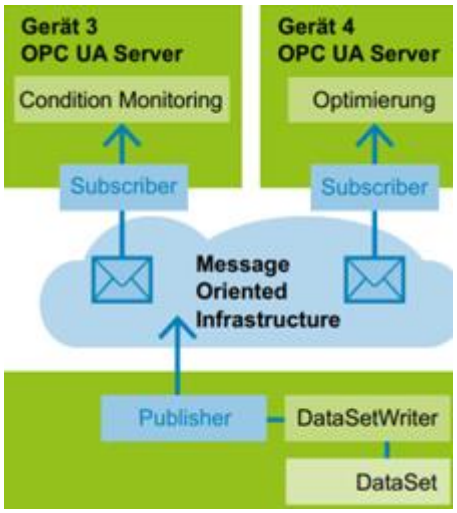
Die für überwachte Objekte und das Abonnement verfügbaren Einstellungen sind in der folgenden Abbildung dargestellt:



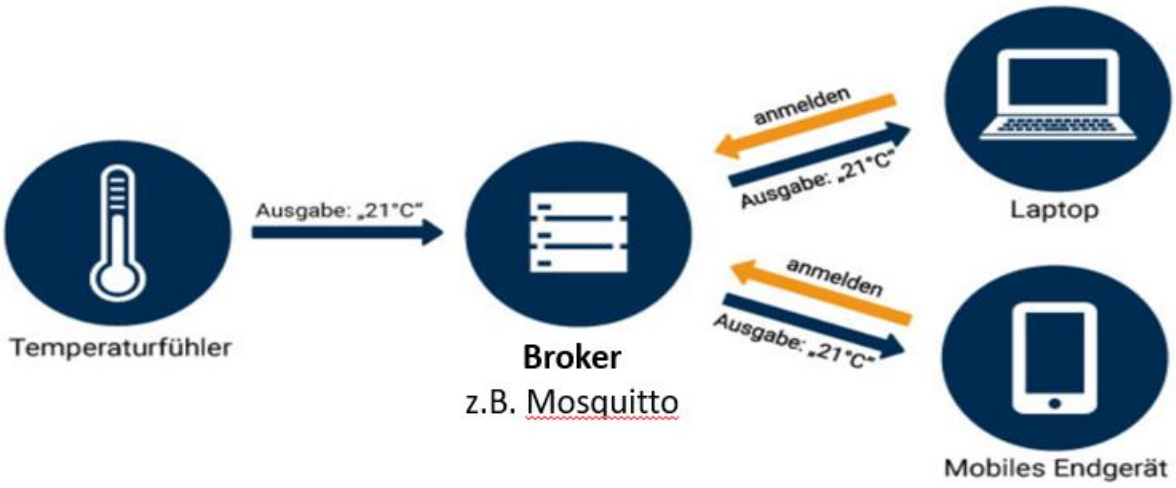
Das Abtastintervall wird individuell für jedes überwachte Element im Abonnement definiert. Dies ist die Rate, mit der der Server die Datenquelle auf Änderungen überprüft. Das Sampling (Abtastung) kann viel schneller als die Benachrichtigung an den Client. Der Server kann das Abtastintervall selbst anpassen, um Überlastungen zu vermeiden.

3. „Publish and Subscribe“

Der Mechanismus „Publish and Subscribe“ wurde kürzlich in OPCUA integriert. Das „Konkurrenz-Verfahren“ MQTT ist weit verbreitet und arbeitet danach. Im Gegensatz zur reinen Subskription (Siehe Punkt2) kommt hier noch eine dritte Komponente dazu, der sogenannte **Broker**. Im rechten Bild entspricht die blaue Wolke diesem Broker. Die Bezeichnung „Message Oriented Infrastructure“ verdeutlicht, dass es sich um eine Hardware (z. B. PC, Server, RaspberryPi, …) handelt, auf der der Broker installiert werden muss.



Wie funktioniert der Broker?



Es existiert eine Datenquelle z. B. ein Temperaturfühler, der Nachrichten an den Broker veröffentlicht (Publish). Die Abonnenten (Subscriber) müssen sich beim Broker anmelden, um die Daten vom Temperaturfühler –über den Broker- zu erhalten (Ausgabe).

Fragen:

1. Bei der Subscription (2) gibt es aggregierte Werte, bei deren Änderung OPCUA Daten veröffentlich. Überlegen Sie sich, was ein aggregierter Wert sein könnte!

……

……

……

2. Erklären Sie anhand des Analogiebeispiels des Lesezirkels, den Sie bei der Frisör*in oder Ärztin vorfinden, „Publish and Subscribe“!

……
……
……

Quellen:
https://www.oreilly.com/library/view/http-the-definitive/1565925092/ch04s01.html
https://documentation.unified-automation.com/uasdkhp/1.4.1/html/_l2_ua_subscription.html

Englische Literatur:

Request and response

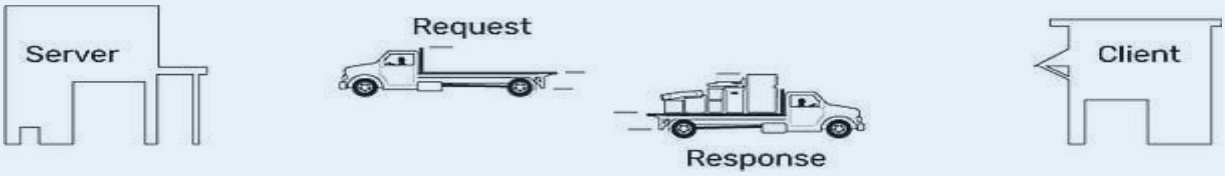
The typical model for computers communicating on a network is *request-response*. In the request-response model, a client computer or software requests data or services, and a server computer or software responds to the request by providing the data or service.

For example, when you send a spreadsheet to the printer, your spreadsheet program is the client. Its request for printer services goes to your company’s print server, which responds to the request and allocates resources for printers on the network. The print server handles all the client requests for printing, making sure your spreadsheet and your coworkers’ print jobs are all completed in an orderly way.

When you want to watch that YouTube video on your smartphone, your web browser or YouTube app is the client, requesting the video over that giant of networks, the Internet. YouTube’s web server receives the request and responds by serving the video page to you, along with the other millions of video pages going to other millions of viewers worldwide.

In automation, the client is typically a PC, and the server is a PLC or PAC. The HMI application on your PC requests data from the PLC or PAC in order to display it on the monitor.

You can imagine request-response as a client sending an empty truck out to be filled with data. The server responds by putting the data on the truck and sending it back.



Publish and subscribe

A different way for devices to communicate on a network is called *publish-subscribe,* or *pub-sub*. In a pub-sub architecture, a central source called a broker (also sometimes called a server) receives and distributes all data. Pub-sub clients can publish data to the broker or subscribe to get data from it—or both.

Clients that publish data send it only when the data changes (report by exception, or RBE). Clients that subscribe to data automatically receive it from the broker/server, but again, only when it changes.

The broker does not store data; it simply moves it from publishers to subscribers. When data comes in from a publisher, the broker promptly sends it off to any client subscribed to that data.

In our truck analogy, there are no empty trucks. A client publishing data sends a full truck to the broker. The broker sees the truck come in but doesn't unload it; it simply routes it intact to a subscriber (cloning the truck if there's more than one subscriber).

In the diagram below, the client on the left publishes data that the clients on the right subscribe to. In addition, the client in the lower right also publishes data needed by other clients not shown.