Hochschule RheinMain
Fachbereich DCSM - Informatik
Dr. Oliver Hahm

Verteilte Systeme

SS 2020
LV 4132

Übungsblatt 7
Praktische Übungen
Bearbeitungszeit: 2 Wochen
Abgabe: 14.06.2021, 04:00 Uhr MESZ

Aufgabe 7.1 (Projekt „Eine IoT-Anwendung für das Hamsterasyl aka. Hamster-Instrumentierung“):

Das neue Management des westhessischen Hamsterverwahrungsunternehmens war sehr zufrieden mit der neuen SunRPC-basierten Schnittstelle und hat niemanden gefeuert. Stattdessen wurde das inzwischen recht große Unternehmen von einem kleinen Zwei-Mann-Start-Up aufgekauft.

Das Start-Up ist (bzw. soll nach eigener Überzeugung werden) der weltweit führende Anbieter für Kleintier-Management und macht natürlich in IoT („The Internet of Things“). Die beiden Gründer haben BWL und Marketing studiert (einer hat sogar einen Schein in „Informatik“ und schon mal Google gepingt, weswegen er natürlich der CTO ist) und viel Investorengeld eingesammelt nur leider noch keinen Anwendungsfall (und kein Produkt, keine Entwickler, keine Technologie …). Da kommt ihnen das uns bekannte westhessische Hamsterverwahrungsunternehmen und die dort arbeitenden Entwickler gerade recht, insbesondere nachdem sie feststellen mussten, dass Embedded-Entwickler (die braucht man für richtiges IoT wohl) stark nachgefragt, schwer zu bekommen sind und dann oftmals sogar mehr Geld verdienen, als ein CEO eines weltweit führenden Start-Ups. Mit dem Geld der zweiten Finanzierungsrunde das eigentlich für Billardtische und Bällchenbäder vorgesehen war, wird die alteingesessene Firma und damit auch Sie als deren Mitarbeiter übernommen.

Natürlich kommt auf Sie jetzt die nach Meinung der neuen Geschäftsleitung recht triviale Aufgabe zu, ein IoT-Produkt zu entwickeln, bzw. genauer ein richtiges IoH aufzubauen („Internet of Hamsters“). Jeder Hamster soll jetzt an das IoH angebunden werden und 24/7 überwacht und gemanagt werden können. Dazu soll die typische IoT-Technologie MQTT verwendet werden. Da das ganze jedem Hamster auf den Rücken geschnallt werden soll, muss es klein und leicht sein. Gut, dass Sie schon Erfahrungen mit C haben! Das ist die kommende Programmiersprache im IoT wenn es um Endgeräte geht!

In Ihrem SVN-Repository finden Sie den neuen Ordner „7“ und darin die Ordner „HamsterIoT“ und „configs“.

Aufgabe 7.1.1 Informieren über MQTT

Informieren Sie sich zuerst wieder zu MQTT. Für die Anwendung soll die Client-Bibliothek des Eclipse Paho-Projekts verwendet werden.

MQTT ist ein Publish-Subscribe-basiertes Nachrichtenprotokoll, das die Übertragung von Daten in Form von Nachrichten ermöglicht. Ein zentraler Broker stellt dabei die Verbindung zu auch ressourcenarmen Knoten her. Die Nachrichten werden unter „Topics“ veröffentlicht.

- Arbeiten sie folgenden Artikel durch:
  `https://www.heise.de/developer/artikel/MQTT-Protokoll-fuer-das-Internet-der-Dinge-2168152.html?seite=all`
- der Artikel bezieht sich zwar auf die Java-Schnittstelle, gibt aber einen guten Überblick über den Aufbau und die Verwendung von MQTT.
- Machen Sie sich dann mit der API-Dokumentation der Paho-C-API vertraut `https://www.eclipse.org/paho/files/mqttdoc/MQTTClient/html/`
- In der Dokumentation sind Beispiele aufgeführt. Probieren Sie diese aus.
- Für die Aspekte der Verschlüsselung und Authentifizierung schauen Sie sich die API-Doku für die Struktur `MQTTClient_SSLOptions` an und wie sie in den Testprogrammen des Paho-Projekts verwendet wird (`https://github.com/eclipse/paho.mqtt.c/`).

Für Linux gibt es das Paket „mosquitto“, das einen MQTT-Broker sowie fertige Kommandozeilenprogramme zum manuellen „publishen“ und „subscriben“ bereitstellt.

Aufgabe 7.1.2 Vorarbeiten: Erster Test mit `mosquitto`

Hier soll zunächst mithilfe der `mosquitto`-Tools die Funktion des publish-subscribe Protokolls erprobt werden. Dieser Teil ist *nicht* abgabepflichtig.

Es gibt einen zentralen MQTT-Broker unter `hamsteriot.vs.cs.hs-rm.de`. Sie können aber auch lokal einen eigenen `mosquitto` Broker starten[1]. Die Konfiguration des Brokers, die Sie auch für Ihren eigenen Broker verwenden können,findet sich im `config`-Ordner. (*Hinweis:* Achtung ggf. müssen Pfade angepasst werden!)

Der mosquitto-MQTT-Broker bietet generische Publish- und Subscribe-Anwendungen. Diese sollten installiert sein. Mit `mosquitto_sub` können Sie beliebige Topics abonnieren und die Nachrichten auf der Konsole ausgeben und mit `mosquitto_pub` Nachrichten an Topics schicken.

Der MQTT-Broker ist so konfiguriert, dass nur bestimmte Topics erzeugt und verwendet werden können. Zum ersten Testen gibt es aber das Topic „`/Test`“.

Subscriben Sie das Topic mit `mosquitto_sub -t "/Test" -h hamsteriot.vs.cs.hs-rm.de`

In einer zweiten Konsole können Sie dann Nachrichten an das Topic schicken mit:
`mosquitto_pub -t "/Test" -h hamsteriot.vs.cs.hs-rm.de -m "NACHRICHT"`

Beginnen Sie als nächstes, ein eigenes kleines C-Programm aufzusetzen, das das Test-Topic abonniert. Damit ihr Programm die Bibliotheken des Eclipse Paho Projekts findet, müssen sie vor dem Starten die Datei `env.sh` „sourcen“ (Befehl: `source env.sh`).

Senden Sie wieder mit `mosquitto_pub` Daten an das Test-Topic und überprüfen Sie deren Empfang mit Ihrem eigenen Subscriber-Programm.

[1] Projektseite: `https://mosquitto.org`, Doku `https://mosquitto.org/man/mosquitto-8.html`

Aufgabe 7.1.3 Die eigentliche Aufgabe: Hamster-IoT

Entwickeln Sie eine IoT-Anwendung, um die abgegeben Hamster überwachen zu können. Verwenden Sie dazu das IoT-Protokoll MQTT und die Eclipse Paho Bibliothek für C.

Im Endausbau sollen die Hamster mit kleinen Sensorknoten-Rucksäcken ausgestattet werden, die verschiedene Aktivitäten des Hamsters per „publish“ an das System melden. Ausserdem gibt es einen Rückkanal, den diese Sensorknoten abonnieren sollen. Darüber ist es möglich, den Hamster zu knuddeln (*fondle*), oder ihn mit kleinen Elektroschocks zu bestrafen (*punish*), wenn er sich schlecht benimmt.

Da die Hochschule keine Mittel zum Unterhalt einer geeigneten Hamsterherde hat, soll Ihre IoT-Anwendung den Hamster aber nur simulieren, d.h. Ihr Programm soll, über Konsoleingaben gesteuert, Aktivitäten des Hamsters melden und es soll über den Rückkanal eingehende Knuddeleien und Bestrafungen einfach auf der Konsole als Meldungen ausgeben.

Einige Hinweise:

- Zur Identifikation der MQTT-Endpunkte wird eine eindeutige ID benötigt. Für diese Aufgabe können Sie die IDs der Hamsterlib verwenden[2], oder auch eine eigene Zufallszahl, die Sie mit der Option `-i` angeben können.
- Die Topics sind bei MQTT in „Namespaces“ gegliedert (Verzeichnisbaum/URI-artig), wobei die Hamster-ID Teil des Namespaces ist. Es gibt für diese Aufgabe zwei Topics für alle und dann mehrere Topics für jeden Hamster (siehe Tabelle 1).
- MQTT bietet die Möglichkeit, Benutzer über Namen und Passwort authentifizieren zu lassen. Um die Komplexität zu reduzieren verwenden alle IoT-Anwendungen denselben Usernamen und kein Passwort (User: „hamster“).
- Der Broker beschränkt die Nutzung von Topics. Jedes erlaubte Topic kann Lese- und Schreibrechte haben. Unter „ACL“ sind in Tabelle 1 die Rechte für die beiden User „hamster“ und „admin“ aufgeführt.
- MQTT bietet verschiedene Quality of Service (QoS) Optionen. Unter „QoS“ sind in Tabelle 1 die geforderten Eigenschaften aufgeführt.

Aufgaben:

(a) Hamster-Status publishen

Simulieren Sie das Verhalten Ihres Hamsters über Menü-Eingaben: der Hamster kann einen der Zustände RUNNING, SLEEPING, EATING oder MATEING einnehmen, und er kann sich in einem von vier Räumen (A, B, C oder D) aufhalten. Sehen sie Benutzereingaben (z.B. ein Menüprogramm) für Statusänderungen des Hamsters vor. Eine Vorlage hierfür finden Sie in der Datei `hamster_mqtt.c`. Publizieren Sie den Zustand Ihres Hamsters bei jeder Änderung unter dem Topic `state`, seinen aktuellen Aufenthaltsraum nach jedem Raumwechsel unter dem Topic `position` und die von ihm bisher gelaufenen Umdrehungen des Hamsterrads unter dem Topic `wheels`. Diese Zahl ist beim Programmstart gleich Null und sie erhöht sich um 25 Umdrehungen pro Minute solange der Hamster im Zustand RUNNING ist. Publizieren Sie den aktuellen Wert daher immer dann, wenn Ihr Hamster

[2]Kopieren Sie sich dazu einfach die Funktion `make_hash()` aus der Datei `hmstr_new()` im Verzeichnis `libsrc` der ersten Übung

aus dem Zustand RUNNING in einen der drei anderen Zustände wechselt. Überprüfen Sie die Funktionen mit dem `mosquitto`-Subscriber-Programm.

Hinweise:

- Sehen Sie vor, dass beim Start Ihres Programmes die Hamster-ID in dem geforderten Topic publiziert wird.
- Denken Sie daran, dass Sie den Usernamen mit angeben müssen!
- Überprüfen Sie die Funktionalität mit dem `mosquitto`-Subscriber-Programm.
- Setzen Sie dann die entsprechende QoS-Klasse.
- Stellen Sie sicher, dass die Benutzung ihres Programms einem Benutzer erklärt wird (Hilfetexte etc).

(b) Hamster-Control subscriben

Erweitern Sie ihr Programm so, dass Nachrichten für ihren Hamster aus den Control-Topics `fondle` und `punish` sinnvoll auf der Konsole ausgegeben werden.

Verwenden Sie das `mosquitto`-Publisher-Programm um die Funktion zu testen. Denken Sie dabei daran, dass dieses den User „admin“ verwenden muss.

Topic	Beschreibung	ACL	QoS
`/pension/livestock`	Jeder neue Hamster publiziert hier seine ID	hamster-rw, admin-rw	Ein neu hinzukommender Subscriber soll unmittelbar die Nachricht bekommen, aber nur genau einmal.
`/pension/hamster/{ID}/wheels`	Wenn der Hamster in einem Rad läuft, wird hier die Anzahl an Runden publiziert.	hamster-rw, admin-rw	Best Effort.
`/pension/hamster/{ID}/state`	Aktueller Zustand des Hamsters. Publizierung bei Änderungen. States: - RUNNING - SLEEPING - EATING - MATEING	hamster-rw, admin-rw	Gesicherte Zustellung mindestens einmal.
`/pension/hamster/{ID}/position`	Aktuelle Position des Hamsters in der Pension. Positions: - A - B - C - D	hamster-rw, admin-rw	Gesicherte Zustellung, mindestens einmal.
`/pension/hamster/{ID}/fondle`	Rückkanal zum jeweiligen Hamster. Anzahl an Belohnungen in Form von Streicheleinheiten.	hamster-r, admin-rw	Gesicherte Zustellung, genau einmal.
`/pension/hamster/{ID}/punish`	Rückkanal zum jeweiligen Hamster. Anzahl an Bestrafungen.	hamster-r, admin-rw	Gesicherte Zustellung, genau einmal.
`/pension/room/{A,B,C,D}`	Publiziert eine Liste der Hamster-IDs die in diesem Raum sind. Basiert auf der korrekten Publizierung in `/pension/hamster/{ID}/position`.	hamster-r, admin-rw	Best Effort.

Tabelle 1: Vorgesehene Topics, Zugriffsrechte und QoS-Eigenschaften

Aufgabe 7.1.4 Verschlüsselte Kommunikation mit TLS

Neben der unverschlüsselten Verbindung zu dem MQTT-Broker (std. Port 1883) gibt es auch die Möglichkeiten sich verschlüsselt über TLS zu verbinden (std. Port 8883). Damit der Sensorknoten weiß, dass er es mit dem richtigen Broker zu tun hat, wird eine PKI (public key infrastructure) basierend auf X.509 Zertifikaten verwendet. Eine CA (Certificate Authority) hat das Zertifikat des Brokers signiert. Wenn Sie dieser CA vertrauen, dann können Sie feststellen ob der Broker auch vertrauenswürdig ist.

Erweitern Sie ihr Programm so, dass eine verschlüsselte Verbindung zum Broker aufgebaut

wird (einzuschalten über das `-v` Programmparameterflag). Verwenden Sie für die verschlüsselte Verbindung das bereit gestellte Zertifikat der CA (`mqtt_ca.crt`), um sicherzustellen, dass die Verbindung sicher ist.

Überprüfen Sie mit geeigneten Mitteln die Datenkommunikation. Werden die Nachrichten verschlüsselt übertragen?

Erzeugen Sie ein eigenes CA-Zertifikat und verwenden Sie dieses. Wird erkannt, dass dem Server nicht vertraut werden kann?

Aufgabe 7.1.5 Hamster-Authentifizierung mit X.509

Mittels X.509 Zertifikation kann die IoT-Anwendung sich auch beim Server authentifizieren. Hierzu bietet unser MQTT-Broker auf Port 8884 einen weiteren Zugang.

Verwenden Sie OpenSSL, um für die IoT-Anwendung ein RSA Schlüsselpaar mit mindestens 2048 Bit Länge, sowie entsprechenden X.509 Zertifikat zu erzeugen. Füllen Sie die notwendigen Angaben sinnvoll aus. Das Setzen eines Passworts ist optional.

Erzeugen Sie dann (ebenfalls mit OpenSSL) für ihr Zertifikat ein CSR (Certificate Signing Request) und schicken Sie dieses ihrer/ihrem Lehrbeauftragten. Diese/dieser signiert es mit dem privaten CA-Schlüssel und gibt ihnen das Ergebnis. Importieren Sie dieses in ihren Schlüssel und verwenden Sie es für die Authentifizierung.

Dokumentieren Sie ihr Vorgehen: Legen Sie ihre erzeugten Zertifikatsdateien in den Ordner `certs` ab und fügen Sie diese dem SVN-Repo hinzu.

Erweitern Sie ihre Anwendung so, dass diese ihr Zertifikat für die Authentifizierung gegenüber dem Broker verwendet. Dies soll über das Programmparameterflag `-V` aktiviert werden können.

Aufgabe 7.1.6 Optionale Erweiterungsideen

Optional können Sie das Ganze noch erweitern. Mögliche Ideen wären:

- Einen Hamster-Instrumentator als realen Hardware-Aufbau (bspw. ESP8266-basiert)
- Einen richtigen Hamster-Simulator (einzelner Hamster mit MQTT-Anbindung, Hamster tut Dinge, ggf. interagiert mit anderen Hamstern)
- Hamster-(Pension)-Visualisierung (Daten aus dem MQTT)
- Hamster-Kontroll-und-Motivation-Management-Software