BAI5-VS	Praktikum Verteilte Sicherheit – Aufgabenblatt 2	MDE/KSS/ZGR
WiSe 2024/25	Voll-vermascht und weiterer Kleinkram	

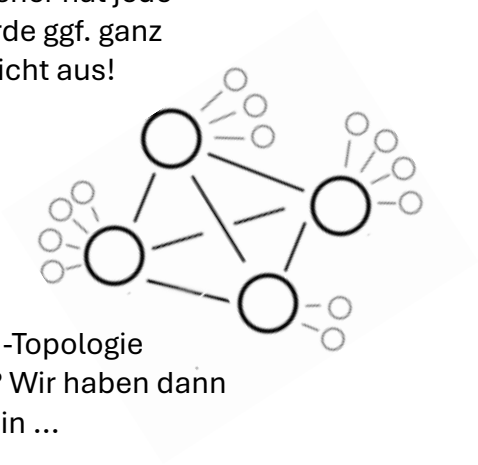
Zur Erinnerung:

Sie programmieren eine Peer-To-Peer-Anwendung, d.h. alle Komponenten sind erstmal potenziell gleich, manchmal wird eine oder werden mehrere mit der Zeit besonderer als andere, aber eins nach dem anderen ;) Ihre Peers sollen, wenn sie aktiviert werden, sich als Teil eines sogenannten Systems-of-Systems von gleichartigen Peers integrieren, hier also Teil eines verteilten System in einem lokalen (Computer-) Netzwerk werden.

Dafür müssen Sie verschiedene Herausforderungen lösen – hatten wir bereits gesagt – wobei ein paar noch offen sind:

- Wenn wir evtl. mehrere finden: in welches integrieren wir uns?
- Kommen wir auch wieder raus, wenn wir reingekommen sind? Und dürfen wir das überhaupt?
- Wie kommen die anderen damit zurecht, dass eine Komponente rausgeht?
- Und wer macht am Ende das Licht aus?

Bisher gab es nur einen Stern, alles andere wurde ignoriert. Und bisher hat jede Komponente auch selbst hinter sich aufgeräumt und der Stern wurde ggf. ganz aufgegeben. Das ändert sich jetzt … und das Licht geht natürlich nicht aus!



Und das mit den Nachrichten ist auch noch nicht ganz gelöst …

Wie soll sich die Anwendung ändern?

Wir wollten ja ein Messaging-System entwickeln, das in einer Stern-Topologie funktioniert! Aber warum eigentlich nun in einem Sternen-System? Wir haben dann doch ganz viele Sterne, und natürlich wollen Nachrichten überall hin …

Doch nicht nervös werden … Sie werden gar nicht so viele Methoden / Verfahren neu entwickeln müssen, vielmehr geht es um kleine Veränderungen mit großer Wirkung.

Gibt es auch neue Rahmenbedingungen für Ihre Programmierung?

Ja, natürlich … aber nicht so viele. Wir führen eine neue Standard-Portnummer ein – den <GALAXYPORT>. Auf dem dürfen sich nur die SOL-Komponenten unterhalten! Und alle Arbeitsgruppen, die erfolgreich einen Stern aufgebaut haben, dürfen diesen Port auch verwenden – erst dann 😉

< GALAXYPORT > ::= 8000 + <GALAXY-ID>

Damit auch dieser Port variabel bleibt – und wir ggf. mehrere Galaxien nebeneinander (also isoliert) betreiben können – bitte die <GALAXY-ID> beim Aufruf übergeben!

Auch zwischen Galaxien gibt es BROADCAST- und UNICAST-Kommunikation, die wie gesagt nur die Komponenten verwenden dürfen, die in ihrem System gerade SOL sind:

1. Die BROADCAST-Kommunikation per UDP „mit allen aktiven Komponenten im Netzwerk“ (auf das lokale Netzwerk beschränkt), wobei Pakete an den <GALAXYPORT >/udp gesendet werden müssen.

   Sofern nichts anderes beschrieben wird, enthalten die Pakete für die BROADCAST-Kommunikation nur eine Zeichenkette von max. 1024 Zeichen in UTF-8. Diese Zeichenkette muss durch Null-Byte terminiert sein.[1]

2. Die UNICAST-Kommunikation per TCP „mit Einzelnen aber potenziell allen aktiven Komponenten im Netzwerk“, wobei Verbindungen zu einem gegebenen Port <GALAXYPORT >/tcp aufgebaut werden.

Zur Identifikation von Komponenten werden nicht überraschend UUIDs benutzt, aber jetzt müssen wir etwas aufpassen. Die bisherigen UUIDs bleiben auf jeden Fall erhalten und behalten ihre Funktion:

<STAR-UUID> :	für die Identifizierung eines bestimmten Sterns
<COM-UUID> :	für die Identifizierung einer bestimmten Komponente, eindeutig innerhalb eines bestimmten Sterns

UUID für die Galaxien brauchen wir nicht (d.h. Sie können beruhigt sein, wir bauen dieses Mal keine „höheren“ Topologien mehr auf. Und wenn SOL-Komponenten miteinander kommunizieren, verwenden diese einfach die <STAR-UUID> um sich zu identifizieren.

3.1 SOL ist aufgegangen ...

Sie ahnen sicherlich, was jetzt kommt ... wie baut sich die Galaxy zusammen? Auf eine sehr ähnliche Art und Weise wie der Stern sich aufbaut. Sie bekommen deswegen auch nicht mehr lang und ausführlich beschrieben, wie es geht!

Jeder neue Stern (und wenn ein Stern was „machen“ soll, dann ist immer SOL als die ausführende Komponente gemeint, außer es wird etwas anderes spezifiziert) fragt per BROADCAST an, ob eine Galaxy aktiv ist! Hierzu wird die Zeichenkette „HELLO?“ ergänzt um die identifizierende Angabe der <STAR-UUID> in einem UDP-Paket gesendet.

```
Aufruf:         "HELLO? I AM <STAR-UUID>"
```

Alle aktiven Sterne müssen auf Broadcasts an <GALAXYPORT>/upd lauschen und im Falle einer „HELLO?“-Nachrichten reagieren. Wir können bei diesem Vorgehen nicht 100% garantieren, dass eine <STAR-UUID> in der Galaxy wirklich eindeutig ist. Deswegen müssen alle empfangenden SOLs prüfen, ob die angegebene <STAR-UUID>

[1] Achtung! Wenn Nachrichten an die Broadcast-Adresse 255.255.255.255 gesendet werden, so wird diese NICHT auf der IP empfangen, an die der sendende Socket selbst gebunden ist. Wenn also ein eigener BROADCAST nicht ankommt, so hilft evtl. an eine „kleinere“ Broadcast-Domain zu senden wie z.B. 127.255.255.255. (https://stackoverflow.com/a/74943642) Da Sie ja aber die richtige NETMASK nutzen, kommt das Problem in der Praxis hoffentlich nicht vor ...

schon bekannt ist, und wenn ja, ob das UDP-Paket von der dann schon bekannten IP-Adresse kommt – oder einer ganz anderen IP-Adresse. Anders als beim Stern wird aber im positiven Fall nicht per BROADCAST reagiert, sondern gleich per UNICAST an <GALAXYPORT>/tcp. Es gibt zwei Fälle …

Ist die angegebene <STAR-UUID> noch nicht bekannt, gibt es per REST-Aufruf eine Reaktion von allen empfangenden SOLs:[2]

```
Aufruf:         POST /vs/v1/star

Übergebene Daten:
                {
                     “star”          : <STAR-UUID>,
                     “sol”           : <COM-UUID>,
                     “sol-ip”        : <IP>,
                     “sol-tcp”       : <PORT>,
                     “no-com”        : <NUMBER OF COMPONENTS>,
                     “status”        : <STATUS>
                }

Ergebnis:       application/json

Antworten:      200 ok

                {
                     “star”          : <STAR-UUID>,
                     “sol”           : <COM-UUID>,
                     “sol-ip”        : <IP>,
                     “sol-tcp”       : <PORT>,
                     “no-com”        : <NUMBER OF COMPONENTS>,
                     “status”        : <STATUS>
                }
```

Die gleiche Reaktion gibt es im Prinzip auch, wenn die angegebene <STAR-UUID> bereits bekannt ist und die IP-Adresse noch übereinstimmt. Dann wird allerdings kein neues Objekt angelegt, sondern „`PATCH /vs/v1/star/<STAR-UUID>`“ wird statt „`POST`“ angewandt. Damit übermittelt quasi die SOL, die den Broadcast verarbeitet, ein Update des eigenen Status.

Nur wenn es keine Übereinstimmung bei der über eine bekannte <STAR-UUID> gespeicherte IP-Adresse gibt, wird keine Integration durchgeführt – und in diesem Fall wird gar nichts kommuniziert. Daran ändern auch weitere HELLO-Broadcasts der SOL-Komponente nichts![3]

---

[2] Bitte beachten: Die Felder heißen gleich, aber jede SOL gibt natürlich „ihre“ Daten an, nicht?

[3] Wer Lust hat, kann an dieser Stelle gerne weiterdenken und dann mir einen Vorschlag machen 😉

Das Feld <STATUS> kennen Sie schon. Eine Mengenbegrenzung kommt dieses Mal nicht zum Tragen. Keine Komponente muss sich beenden, wenn nichts zurückkommt, einen Stern zu managen ist sicherlich Grund genug, nicht auszufallen.

3.1 Nachgefragt ...

Jede SOL-Komponente kann jederzeit befragt werden, was sie über die Sterne in einer Galaxy weiß:

```
Aufruf:        GET /vs/v1/star/<STAR-UUID>

Ergebnis:      application/json

Antworten:     200 ok

               {
                    "star"         : <STAR-UUID>,
                    "sol"          : <COM-UUID>,
                    "sol-ip"       : <IP>,
                    "sol-tcp"      : <PORT>,
                    "no-com"       : <NUMBER OF COMPONENTS>,
                    "status"       : <STATUS>
               }

               404 not found
```

Wenn alle bei der angefragten SOL-Komponenten bekannten Sterne abgefragt werden sollen, dann gibt dieser Aufruf folgendes Ergebnis zurück. Bitte beachten, die angefragte Komponente muss sich selbst in dem Ergebnis mit aufführen, d.h. eine leere Liste ist nicht zu erwarten:

```
Aufruf:        GET /vs/v1/star

Ergebnis:      application/json

Antworten:     200 ok

               {
               "totalResults": 2,
               "stars": [
                    {
                    "star"     : <STAR-UUID-1>,
                    "sol"      : <COM-UUID-1>,
                    "sol-ip"   : <IP-1>,
                    "sol-tcp"  : <PORT-1>,
                    "no-com"   : <NUMBER OF COMPONENTS-1>,
                    "status"   : <STATUS-1>
                    },
                    {
                    "star"     : <STAR-UUID-2>,
```

```
                    "sol"       : <COM-UUID-2>,
                    "sol-ip"    : <IP-2>,
                    "sol-tcp"   : <PORT-2>,
                    "no-com"    : <NUMBER OF COMPONENTS-2>,
                    "status"    : <STATUS-2>
                    },
                ]
            }
```

Komponenten, die nicht SOL sind, unterstützen diese REST-Aufrufe nicht, weil diese ja auch gar nicht auf dem <GALAXY-PORT> lauschen.

3.2 Abmelden eines Sterns

Eine aktive SOL-Komponente, die sich nach einem „EXIT"-Befehl runterfährt, meldet sich auch bei allen anderen ihr bekannten Sternen ab. Wenn eine der Sterne nicht erreichbar ist, wird es nach 10 bzw. 20 Sekunden nochmal versucht. Danach wird es aufgegeben, diesen Stern weiter zu erreichen ... Die sich abmeldende Komponente macht wie im ersten Aufgabenblatt angegeben, evtl. auch bei einem Statuscode, der einen Fehler signalisiert, weiter und beendet sich dann irgendwann.

```
Aufruf:         DELETE /vs/v1/system/<STAR-UUID>

Ergebnis:       text/plain

Antworten:      200 ok
                401 unauthorized
                404 not found
```

Eine angesprochene SOL prüft, ob die Anfrage zur Löschung von der hinterlegten IP stammt und es sich um den richtigen Stern handelt. Ist dies nicht der Fall, wird die Abmeldung mit „401" verweigert. Soll eine Komponente abgemeldet werden, die nicht bei SOL als aktiv gespeichert ist, wird „404" geantwortet.

Stimmt alles, wird der neue Status und der Zeitpunkt dieser Meldung durch die empfangende SOL abgespeichert. Der Stern wird aus ihrer Sicht der Galaxy entfernt. Der Dateneintrag bleibt aber erhalten mit der Angabe „left".

4.1 Inter-Galaxy-Nachrichten kommen überall hin ...

Die SOL-Komponenten bekommen ja alle Nachrichten, die innerhalb des Sterns erzeugt werden. D.h. diese können auch weitergegeben werden. Interessanterweise können wir die bereits bekannten REST-Aufrufe aus dem ersten Aufgabenblatt weiterverwenden.

Aber damit es keine Probleme gibt – alte und neue Komponenten – ändern wir die Versionsnummer der REST-Aufrufe (hier für einen dargestellt, aber bitte alle ändern!).

```
Aufruf:         POST /vs/v2/messages/<MSG-UUID>
```

Bisher wurde anhand der <STAR-UUID> sichergestellt, dass Nachrichten auch für diesen Stern „gedacht“ sind. Ist die Komponente nicht im „richtigen“ Stern, gibt es die Antwort „401“ und die Nachricht wird nicht akzeptiert. Dies soll auch weiterhin gelten, aber wir wollen ja Nachrichten auch von Stern zu Stern wandern lassen. Dazu brauchen wir wieder die Hilfe der SOL-Komponenten.

Der gleiche REST-Aufruf, den auch die Komponenten eines Sterns verwenden, wird benutzt, um Nachrichten sozusagen von SOL zu SOL „rüberzugeben“.D.h. hierfür kommt wieder der <STARPORT> zum Einsatz! Da alle SOL-Komponenten die <STAR-UUIDs> der bekannten Sterne kennen, ist es kein Problem, den entsprechenden richtigen Wert *VOR* der Übergabe einzusetzen.

Allerdings haben wir bei Angaben zum Urheber ein kleineres Problem. Dort war eine Standard-Email-Adresse möglich, aber auch die Angabe einer Komponente als Urheber, angezeigt durch die <COM-UUID>. <COM-UUIDs> sind in Version 2 nicht mehr eindeutig, d.h. wir müssen die Urheberangabe erweitern und nehmen dafür die <STAR-UUID> hinzu, also:

<ORIGIN> ::= <COM-UUID> . „:“ . <STAR-UUID> | <EMAIL>

Spätestens bei der Auslieferung an die erste SOL-Komponente wird die <MSG-UUID> vergeben. Auch diese war bisher nur innerhalb eines Sterns eindeutig, d.h. wir passen auch hier an:

<MSG-UUID> ::= <NUMBER> . „@“ . <COM-UUID> . „:“ . <STAR-UUID>

<STAR-UUID> bezieht sich immer auf den Stern, in dem die Nachricht erzeugt wurde.

Wird eine Nachricht von SOL akzeptiert, müssen zusätzliche Angaben gespeichert werden, die Auskunft geben, von welchem Stern eine Nachricht stammt:

„from-star“ ::= <STAR-UUID>
„received“ ::= aktueller Zeitstempel in UNIX-Notation[4]

Außerdem muss SOL speichern, wann Nachrichten aus dem eigenen Stern an andere Sterne ausgeliefert wurden, jeweils:

„to-star“ ::= <STAR-UUID>
„delivered“ ::= aktueller Zeitstempel in UNIX-Notation

4.2 Nachrichten müssen überall gelöscht werden

Alle Komponenten, die in den Stern integriert sind, können Löschaufträge annehmen. Löschaufträge müssen immer an SOL weitergegeben werden. SOL sorgt dann dafür, dass der Inhalt gelöschter Nachrichten nicht mehr zur Verfügung steht, aber die Meta-Daten erhalten bleiben. Löschungen funktionieren mit folgendem Aufruf, der wieder bei SOL und anderen Komponenten identisch ist und nun auch über Stern-Grenzen hinweg funktionieren muss, auch hier über den <STARPORT> an SOL übermittelt:

---

[4] 4 Byte-Wert, der die Zahl der Sekunden seit dem 1. Januar 1970 (UTC) angibt.
siehe auch: https://www.unixtimestamp.com (und man beachte den Hinweis am Ende der Seite!)

```
Aufruf:         DELETE /vs/v2/messages/<MSG-UUID>?star=<STAR-UUID>
```

Wenn SOL einen Löschauftrag annimmt und die Nachricht löscht, wird „200“ zurückgegeben. SOL speichert den Zeitpunkt der Löschung mit der Herkunft des Löschauftrags und ändert den Status, d.h. für jeden angefragten Stern separat:

„status“ ::= „deleted by <STARUUID>“ | „deleted by us from <COM-UUID>“
„changed“ ::= Zeitstempel des Auftrags

Jetzt kommt aber erschwerend hinzu, was mit Nachrichten passiert, die von SOL bereits an andere Sterne weitergegeben wurden – die müssen doch auch von anderen Sternen gelöscht werden. Zum Glück ist das der gleiche Aufruf und der wird dann nacheinander bei allen Sternen angewandt, die die Nachricht erhalten hatten. Die Status-Informationen werden von SOL aktualisiert, es wird zusätzlich der Zeitpunkt des Entfernens vermerkt (die ersten beiden sind bereits bekannt und liegen in so einem Fall schon vor):

„to-star“ ::= <STAR-UUID>
„delivered“ ::= aktueller Zeitstempel in UNIX-Notation
„removed“ ::= aktueller Zeitstempel in UNIX-Notation

4.3 Die Liste aller Nachrichten kann von SOL geholt werden, ebenso jede Nachricht

Alle Komponenten, die in den Stern integriert sind – also auch SOL, können die Liste der Nachrichten abfragen. Bei den bekannten Abfrage ändert sich nichts, außer dass die Änderungen bei <ORIGIN> und <MSG-UUID> sichtbar werden … Die Abfrage, die durch die Angabe der „richtigen“ <STAR-UUID> kontrolliert wird, klappt auch über Stern-Grenzen hinweg. Bitte dran denken, auf „v2“ upzudaten.

Die neuen Werte, die wir oben eingeführt haben, sollen beim Abrufen der Nachrichten von einer SOL-Komponente auch mitberücksichtigt werden. Dazu werden zwei Formate angepasst:

```
{
     “msg-id”       : <MSG-UUID>,
     “status”       : <STATUS>,
     “delivered”    : [ <STAR-UUID-1>, ... ]      }
```

Das „kurze“ Format enthält eine Liste der <STAR-UUIDs>, an die die Nachricht bereits ausgeliefert wurde. Für von anderen Sternen empfangenen Nachrichten ist diese Liste leer. Die Herkunft geht aus der <MSG-UUID> hervor und muss nicht weiter ausgewiesen werden.

Das „lange“ Format enthält zusätzlich mehr Informationen pro Nachricht und natürlich die erweiterten <MSG-UUIDs> bzw. <ORIGINs>:

```
"from"          : <STAR-UUID> | "",
"received"      : 1726788854 | "",
"delivered"     : [
                      { <STAR-UUID-1>, 1726789058 },
                      { <STAR-UUID-1>, 1726712377 },
                    ]
```

Es fehlt jetzt nur noch eine Möglichkeit, nicht nur die Nachrichten eines Sterns abzurufen, sondern alle. Hierzu kann beim REST-Aufruf für den Parameter `"star"` nun das Schlüsselwort `"all"` verwendet werden. (Man kann beliebig verbesserte Sichten hier einbauen, aber das müssen wir nicht in dieser Vorlesung machen!)

```
Aufruf: GET /vs/v2/messages?star=all&scope=<SCOPE>&info=<INFO>
```

Das Ausgabeformat muss sich dabei nicht ändern, da die Herkunft der Nachrichten direkt erkennbar ist. Auch die Angabe der <STARUUID> am Anfang der Daten ist weiterhin richtig, handelt es sich ja um die Sicht auf alle Nachrichten von genau der bezeichneten SOL-Komponente.

4.4 Ach so, PUSH ist heutzutage „in“

Jetzt müssen Sie ein- oder zweimal überlegen, wie Sie das realisieren können. „Keep it easy!“ wäre ein gutes Leitmotiv, „Keep it stupid simple!“ wäre noch besser.

BITTE noch nicht umsetzen, erst drüber reden, dann programmieren!

Was ist gewünscht? Die Komponenten im Stern, sofern nicht SOL, sind ja benachteiligt, sie müssen immer und ständig „pullen“, wenn Sie die Nachrichten sofort benötigen . Damit sich das ändert, „pusht“ SOL nunmehr Nachrichten, die dort empfangen werden, an alle Komponenten.

Schreiben Sie kurz auf, wie der REST-Aufruf aussieht, und die dazugehörigen Logik. Handschriftlich ist okay, ausgedruckt ist okay, **nicht** per Email, **nicht** per Teams, **nicht** länger als eine DIN A4-Seite.