Verteilte Systeme

Aufgabe 2

Beispiel: Verteilter Algorithmus

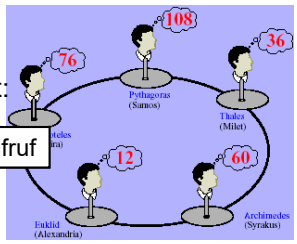
- **Satz von Euklid**: Der grösste gemeinsame Teiler (ggT) zweier positiver ganzer Zahlen x, y (mit x≥y>0) ist gleich dem ggT von y und dem Rest, der bei ganzzahliger Division von x durch y entsteht.
- **Eigenschaften**:
  - Man setzt nun noch **ggT(x,0) := x** für al
  - Rekursive Realisierung: **ggt(x,y) := ggt**
  - Offenbar ist **ggT(x,x) = x** für alle x
- **Erweiterung**: **mod*(x,y) := mod(x-1,y)+1**
- **Verteilter Algorithmus**:
  - Jeder Prozeß P_i hat seine eigene Variable M_i.
  - ggT aller am Anfang bestehender M_i wird berechnet:

Noch nicht geklärt:
- Wie wird der Algorithmus gestartet ?
- Wie erkennt man die Terminierung ?
- Wo steht das Ergebnis ?

Modifikation für die Verteilung

```
{Eine Nachricht <y> ist eingetroffen}
if y < Mi
   then Mi := mod(Mi-1,y)+1;
        send <Mi> to all neighbours;
fi
```

Entspricht rekursivem Aufruf



Beispiel: Verteilter Algorithmus

- Iterative Implementierung (für 2 Zahlen):

```
While b <> 0 do
   Ersetze (a,b) durch (b,(a mod b))
return a
```

(76,12) → (12,4)

- Rekursive Implementierung (für 2 Zahlen):

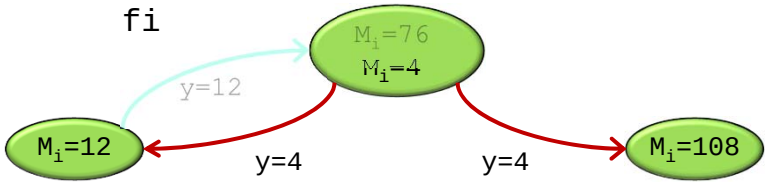
```
Procedure GGT (a,b)
   if b == 0 return a
      else return GGT(b,(a mod b))
```

GGT(76,12) → GGT(12,4)

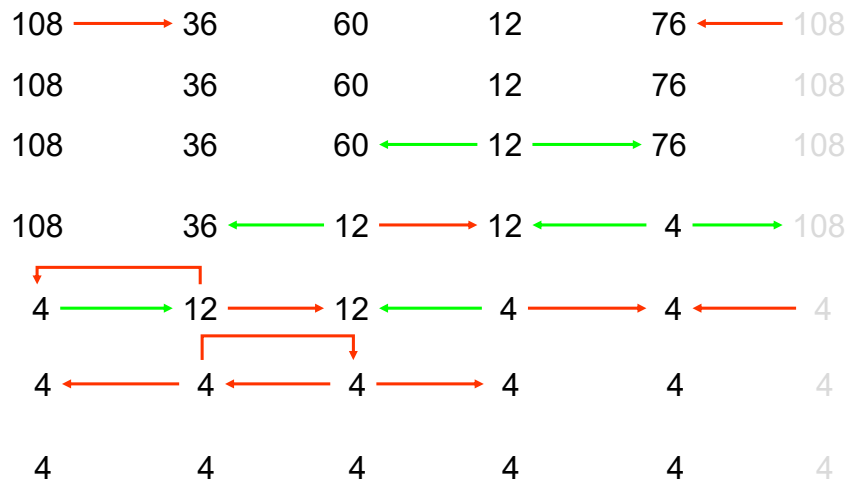
Beispiel: Verteilter Algorithmus

- Verteilte Implementierung (für n Zahlen):

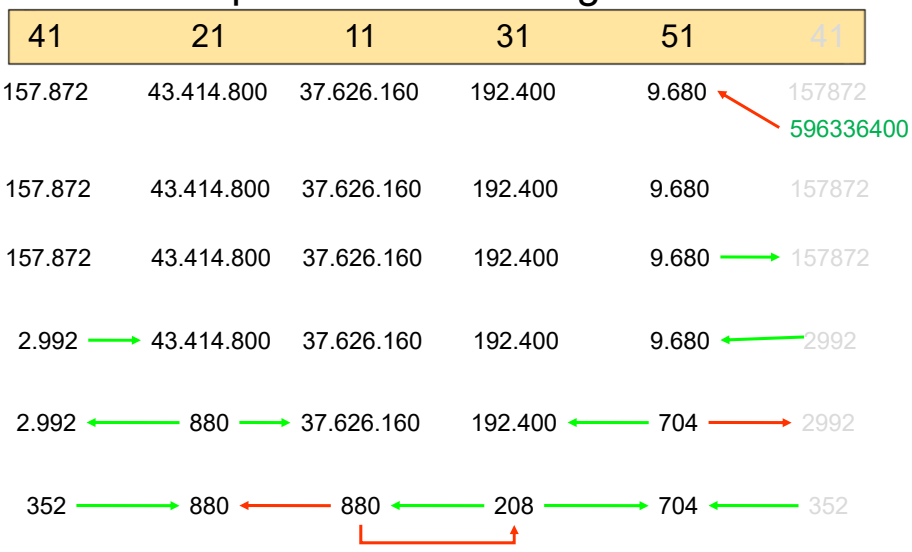
```
{Eine Nachricht <y> ist eingetroffen}
     if y < Mi
        then Mi := mod(Mi-1,y)+1;
             send <Mi> to all neighbours;
        else do something else
     fi
```



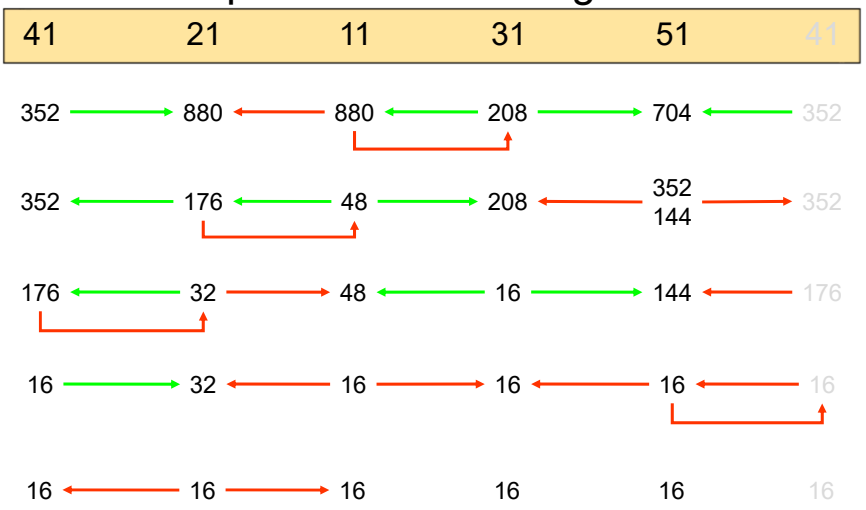
Beispiel 1: Verteilter Algorithmus



Beispiel 2: Verteilter Algorithmus



Beispiel 2: Verteilter Algorithmus



Verteilter Algorithmus

Ausschnitt aus der log-Datei

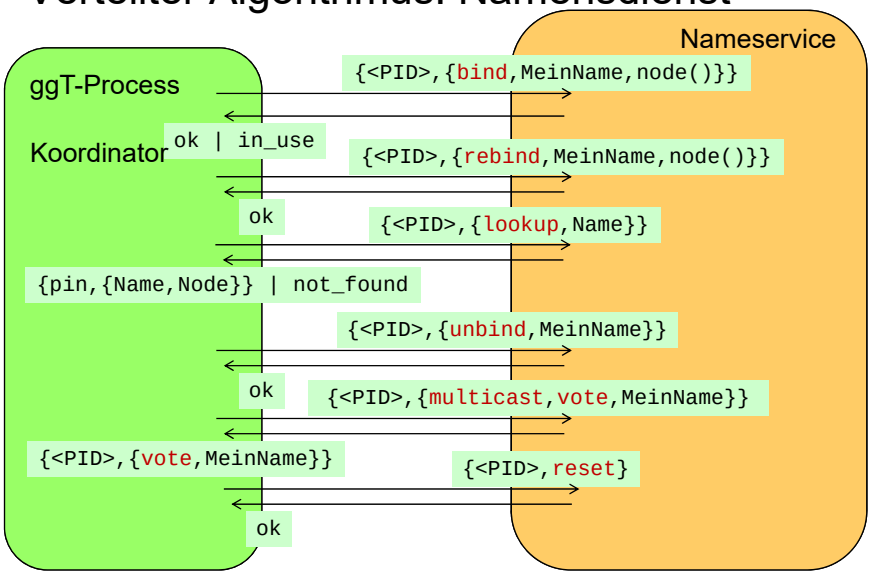
Koordinator-ko@Brummpa-KLC Startzeit: 17.06 10:13:52,969< mit PID <0.37.0>
[…] Alle ggT-Prozesse gebunden.
[…] Ring wird/wurde erstellt, Koordinator geht in den Zustand 'Bereit für Berechnung'.
[…] Beginne eine neue ggT-Berechnung mit Ziel 16.
ggT-Prozess 48851 (ggt@Brummpa) initiales Mi 9680 gesendet.
[…] ggT-Prozess 48851 (ggt@Brummpa) startendes y 596336400 gesendet.
[…] 48851 meldet Terminierung mit ggT 9680 um 17.06 10:15:52,000< (17.06 10:15:52,000<).
[…] **Per Hand** 48841 die Zahl 9680 **gesendet** `{,48841,ggt@Brummpa} ! {sendy,9680}`
48841 meldet neues Mi 2992 um 17.06 10:18:36,499< (17.06 10:18:36,499<).
48821 meldet neues Mi 880 um 17.06 10:18:39,100< (17.06 10:18:39,100<).
48851 meldet neues Mi 704 um 17.06 10:18:39,100< (17.06 10:18:39,100<).
48841 meldet neues Mi 352 um 17.06 10:18:42,700< (17.06 10:18:42,700<).
48831 meldet neues Mi 208 um 17.06 10:18:42,700< (17.06 10:18:42,700<).
48811 meldet neues Mi 880 um 17.06 10:18:42,700< (17.06 10:18:42,700<).
48811 meldet neues Mi 48 um 17.06 10:18:45,200< (17.06 10:18:45,200<).
48821 meldet neues Mi 176 um 17.06 10:18:45,200< (17.06 10:18:45,200<).
…

Verteilter Algorithmus

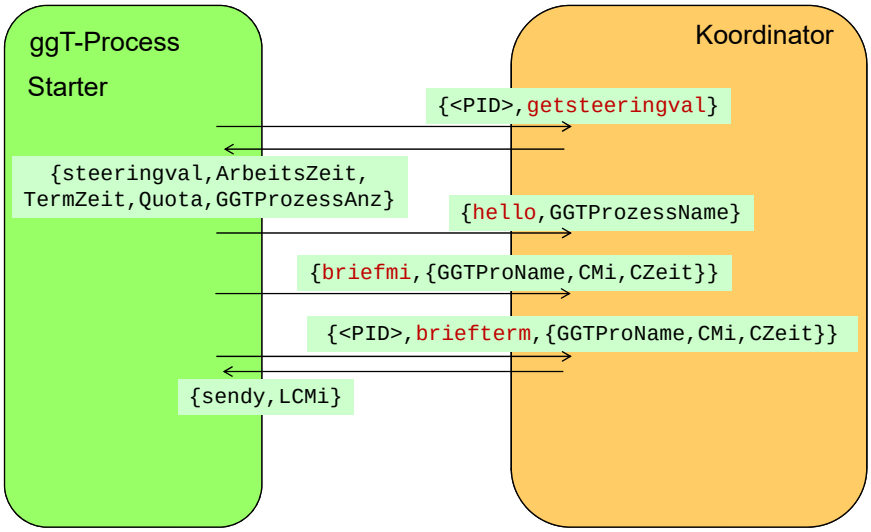
Ausschnitt aus der log-Datei

...
48851 meldet neues Mi 352 um 17.06 10:18:45,200< (17.06 10:18:45,200<).
48831 meldet neues Mi 16 um 17.06 10:18:48,800< (17.06 10:18:48,800<).
48821 meldet neues Mi 32 um 17.06 10:18:48,800< (17.06 10:18:48,800<).
48851 meldet neues Mi 144 um 17.06 10:18:48,801< (17.06 10:18:48,800<).
48841 meldet neues Mi 176 um 17.06 10:18:48,802< (17.06 10:18:48,800<).
48841 meldet neues Mi 16 um 17.06 10:18:51,300< (17.06 10:18:51,299<).
48811 meldet neues Mi 16 um 17.06 10:18:51,300< (17.06 10:18:51,300<).
48851 meldet neues Mi 16 um 17.06 10:18:51,300< (17.06 10:18:51,300<).
48821 meldet neues Mi 16 um 17.06 10:18:54,090< (17.06 10:18:54,090<).
48811 meldet Terminierung mit ggT 16 um 17.06 10:18:54,090< (17.06 10:18:54,090<).
48811 meldet Terminierung mit ggT 16 um 17.06 10:18:57,090< (17.06 10:18:57,090<).
48821 meldet Terminierung mit ggT 16 um 17.06 10:18:57,090< (17.06 10:18:57,090<).
48841 meldet Terminierung mit ggT 16 um 17.06 10:18:57,090< (17.06 10:18:57,090<).
[…]

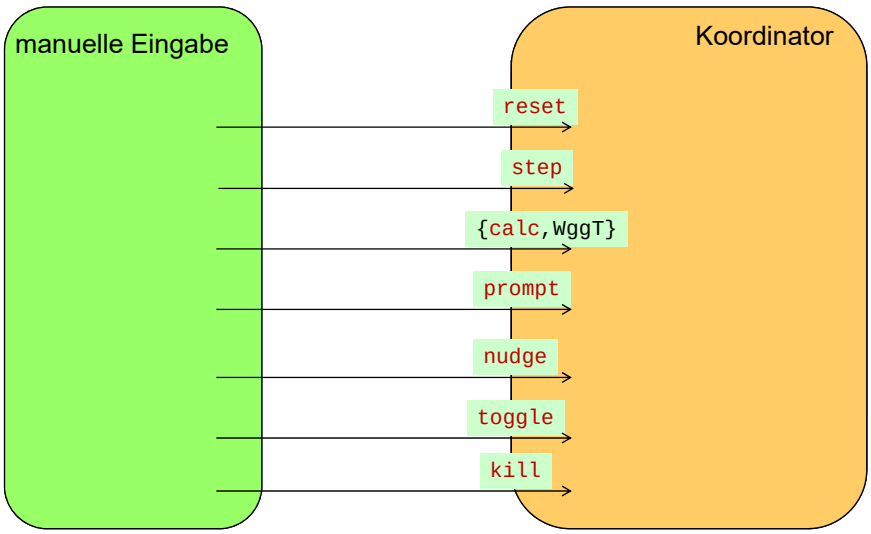
Verteilter Algorithmus: Namensdienst



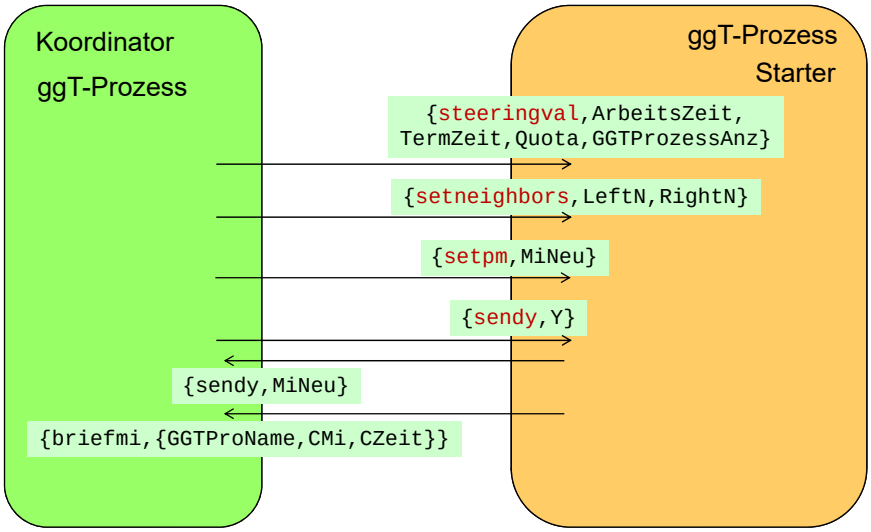
Verteilter Algorithmus: Koordinator



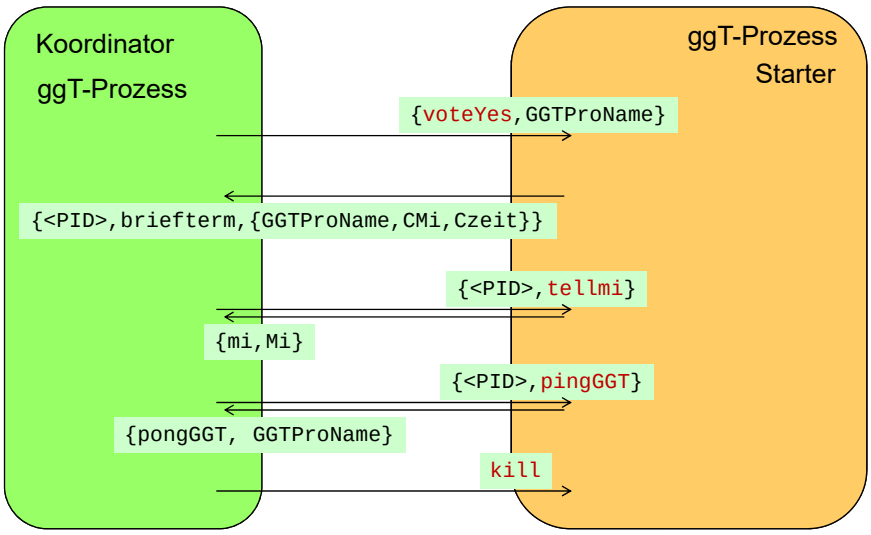
Verteilter Algorithmus: Koordinator

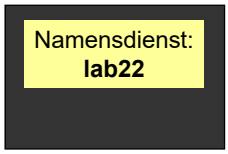


Verteilter Algorithmus: ggT-Prozess



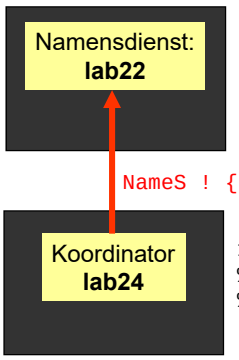
Verteilter Algorithmus: ggT-Prozess





Anfang

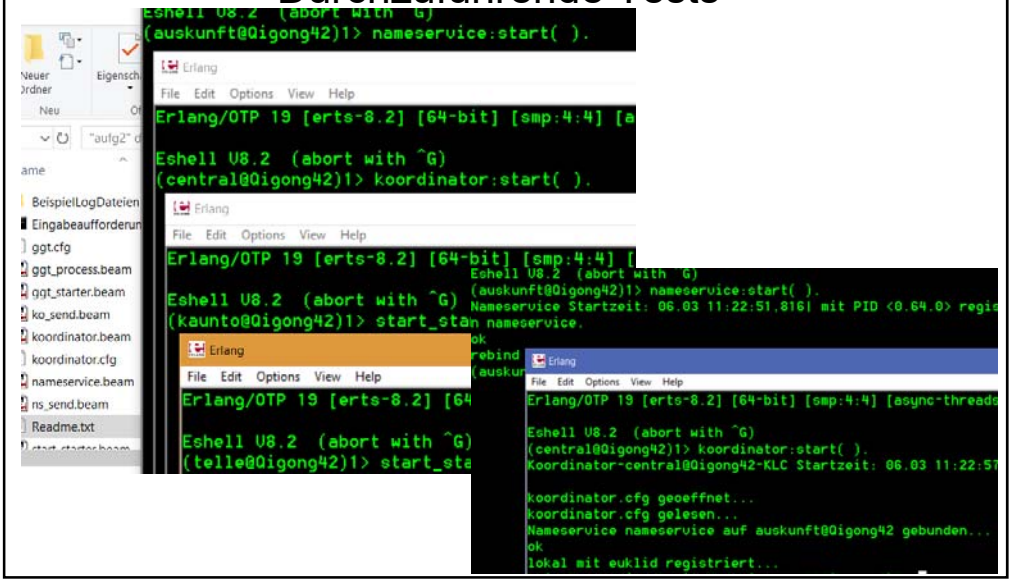
```
(w)erl -(s)name ns (-setcookie zummsel)
1>nameservice:start( ).
%global:register_name(nameservice,NServerPid)
```

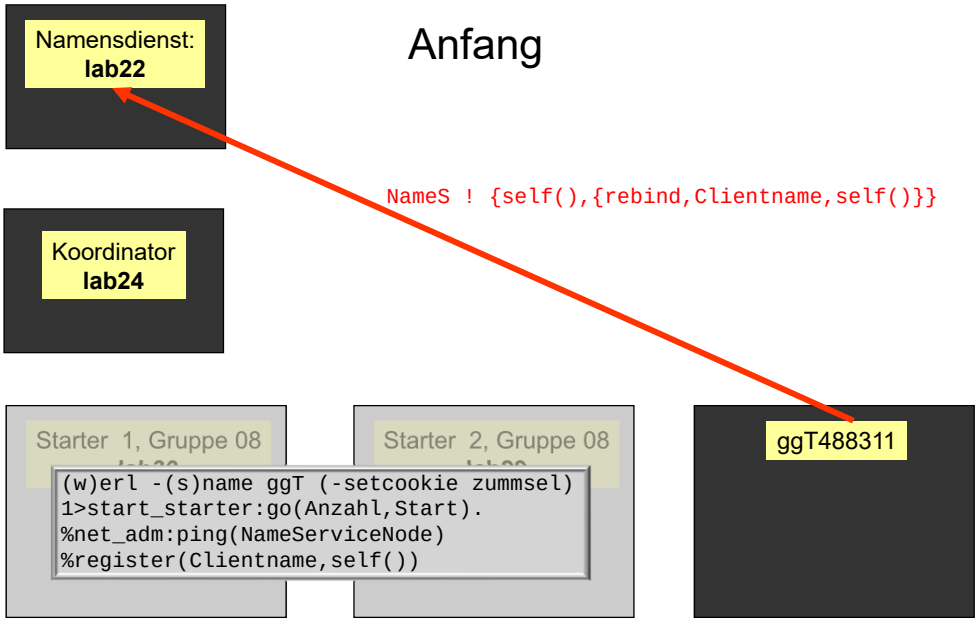
Anfang

```
(w)erl -(s)name ko (-setcookie zummsel)
1>koordinator:start( ).
%net_adm:ping(NameServiceNode)
%register(KoName,KoPID)
```

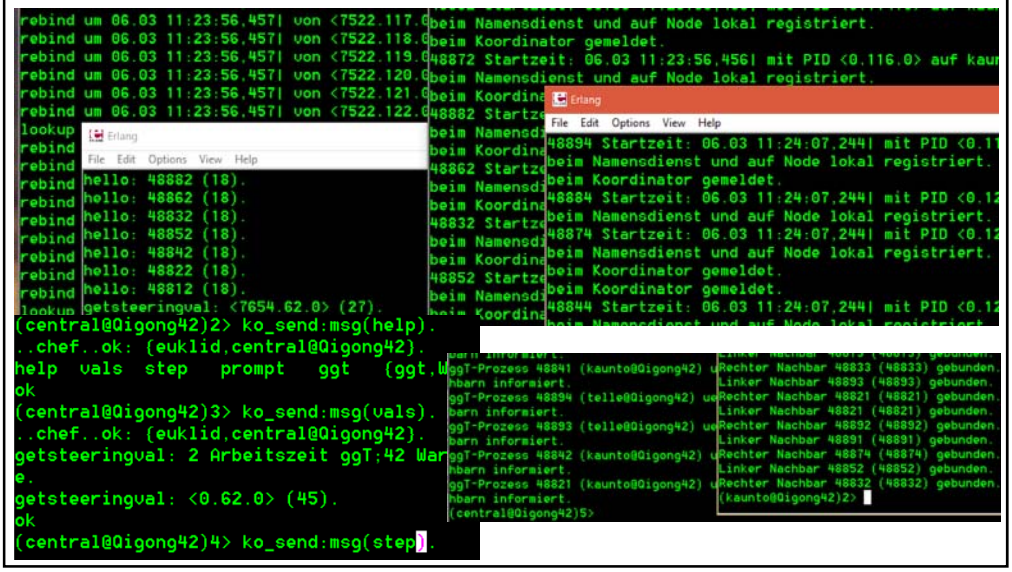
Durchzuführende Tests



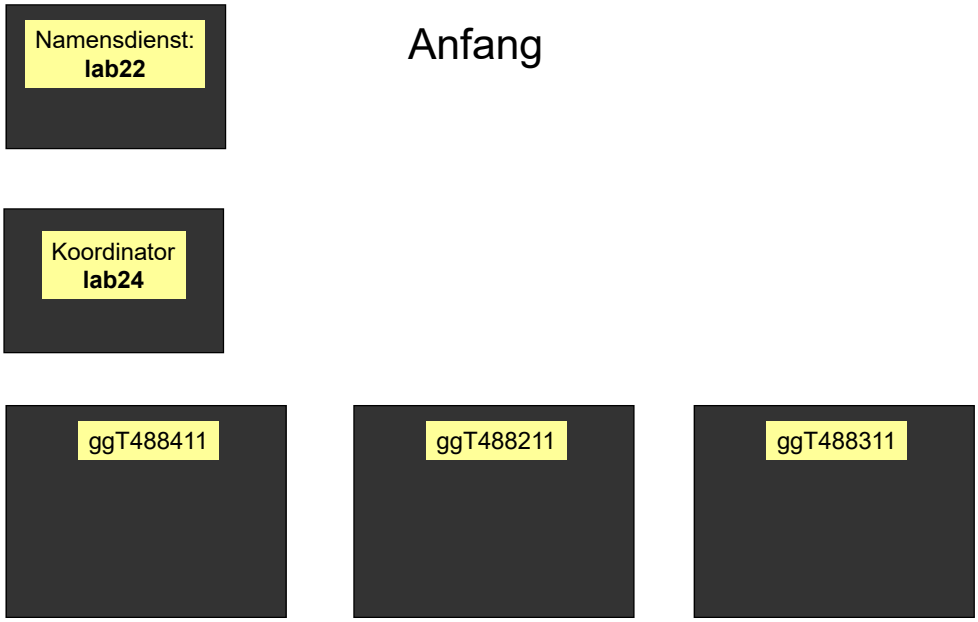
Anfang



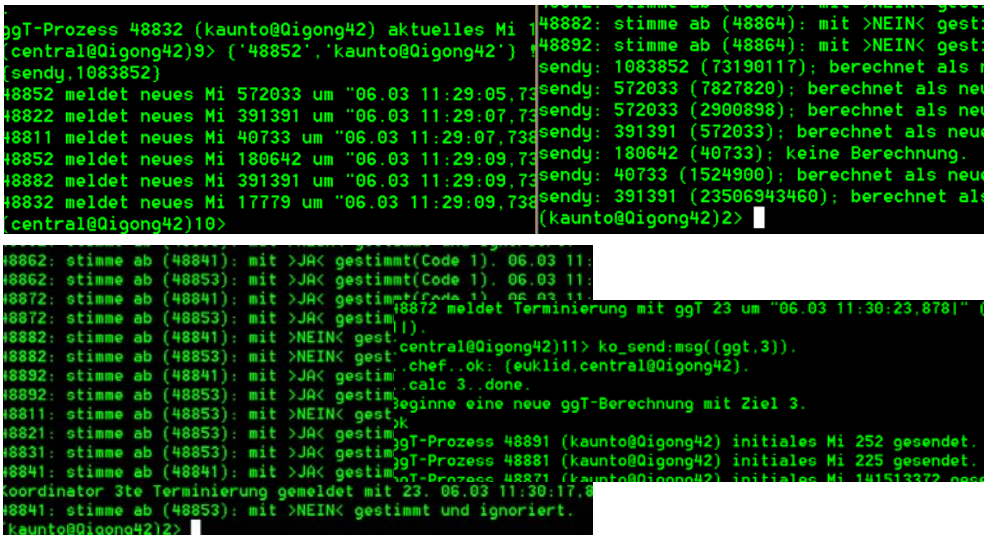
Durchzuführende Tests



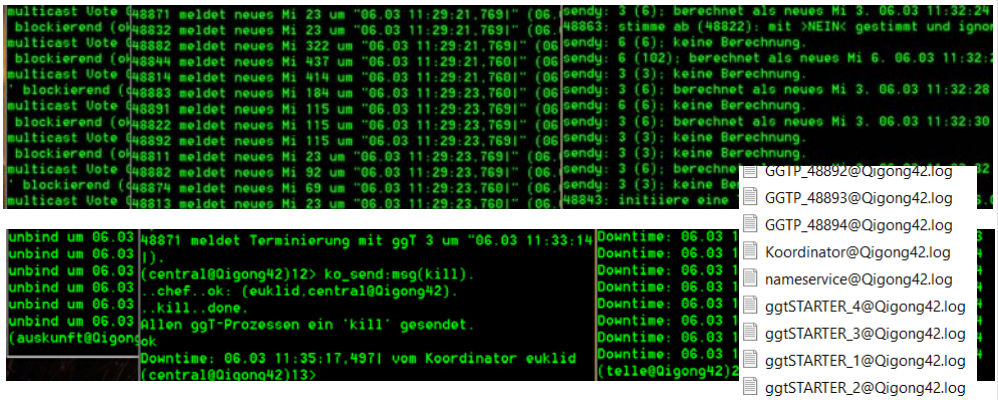
Anfang



Durchzuführende Tests



Durchzuführende Tests



Test 1: Ihr Koordinator, vorgegebener ggT Prozess
Test 2: Vorgegebener Koordinator, Ihr ggT Prozess
Test 3: ein Koordinator, vorgegebener und Ich ggT Prozess