Verteilte Programmierung mit Erlang

Ein Erlang–Knoten ist ein Erlang–System, das einen Namen besitzt und mit anderen Erlang–Knoten in einem Verteilten System kommunizieren kann. Alle miteinander kommunizierenden Erlang–Knoten eines Netzes haben ein gemeinsames Geheimnis, das im File `$HOME/.erlang.cookie` abgelegt wird. Wir wählen *Verteilte Systeme* als gemeinsames Geheimnis[1]. Erlang–Knoten werden in der Form `erl -name KnotenName` gestartet. Mehrere Erlang–Knoten können (unter verschieden Namen) auf einem Computer laufen. Das Erlang–Kommando `node()` ermittelt zu einem Knoten seinen qualifizierten Namen. `nodes()` bestimmt die Liste der Knoten, mit denen der aktuelle Knoten verbunden ist. Die Verbindung wird mit dem ersten Nachrichtenaustausch aufgebaut. Mit Hilfe des Erlang–Kommandos `net_adm:ping(KnotenName@RechnerName)` kann die Verbindung etabliert werden.
Wollen wir einen Prozess auf einem anderen Knoten starten, so muss der (übersetzte) Programmcode auf diesem Knoten zur Verfügung stehen. Er kann explizit (als Programmfile) kopiert, über ein gemeinsames Filesystem bereitgestellt, mit Hilfe eines Code–Servers ausgetauscht oder aber mit dem Erlang–Kommand `nl(Modulname)` (*node-load*) auf allen verbundenen Knoten geladen werden. Wir verwenden sinnvollerweise letztere Lösung.

27. **verteiltes Erlang**

a) Definieren[1] Sie bitte das gemeinsame Geheimnis *Verteilte Systeme* in `$HOME/.erlang.cookie` und starten Sie dann einen Erlang–Knoten mit dem Namen *vs* (kurz für Verteilte Systeme) auf Ihrem Rechner. Ermitteln sie mit `node()` den qualifizierten Namen Ihres Knotens.

b) Verbinden Sie sich zu den Knoten Ihrer Kommilitonen, indem Sie mit `net_adm:ping(vs@RechnerName)`[2] die Verbindung prüfen. Bei erfolgreichem Verbindungsaufbau liefert `net_adm:ping` das Atom `pong` als Resultat; bei Fehlern `pang`. Beobachten Sie mit `nodes()`, wie die Liste der verbundenen Knoten wächst.

c) Verwenden Sie Ihre Lösung der Aufgabe 18b (oder nehmen Sie die Musterlösung `counter.erl` vom Handoutserver). Bitte nennen Sie das Modul von `counter` in `counter_gGruppenNummer` um, damit sie mit Ihrem Modul nicht in Konflikt mit den anderen Arbeitsgruppen kommen.

d) Compilieren Sie Ihr Modul. Laden Sie es mit `nl(counter_gGruppenNummer)` auf alle Knoten im Netz.

e) Starten Sie mit `spawn(vs@Rechnername,counter_gGruppenNummer,loop,[0]).` Zählerprozesse auf den anderen Knoten.

f) Schicken Sie bitte den entfernten Prozessen `reset`–, `up`–, `down`–, und `show`–Nachrichten.

g) Starten Sie bitte das Prozess–Manager–Tool mit `pman:start().` und verschaffen Sie sich bitte einen Überblick über die laufenden Prozesse der Knoten.

h) Starten Sie mit `spawn(counter_gGruppenNummer,loop,[0]).` auch lokale Zähler–Prozesse. Beobachten Sie, dass das Nachrichten–Versenden transparent im Bezug auf die Verteilung der Prozesse über Knoten ist.

i) Schreiben Sie bitte eine Erlang–Funktion `broadcast(ProzessListe, Nachricht)`, die einer Liste von Prozessen eine gegebene Nachricht senden kann. Schicken Sie mit Hilfe dieser Funktion Ihren lokalen und nichtlokalen Zähler–Prozessen `show`–, `up`– und `down`–Nachrichten.

[1] `echo Verteilte\ Systeme > $HOME/.erlang.cookie; chmod 400 $HOME/.erlang.cookie`
[2] Namen mit Bindestrich müssen in Apostroph eingefasst werden: z.B.: `'vs@RZ9PC3.fh-wedel.de'`