Netzwerkprogrammierung – Network Programming

Abschlußprojekt
Client/Server Lösung zum (halb)automatischen Software Update von Clients

Jan Krüger
jkrueger@cebitec.uni-bielefeld.de

Motivation : ThinClient

- ThinClient
- Dedizierte Hardware / Energiesparend
- Robust
  - keine beweglichen Teile (Lüfter, HDD)
  - externes Netzteil
- Kaum Erweiterbar
- Standard IO Anschlüsse
- OS : Minimalistisches Linux (Tiny Core, Igel OS, ...)

Motivation : typische Hardware

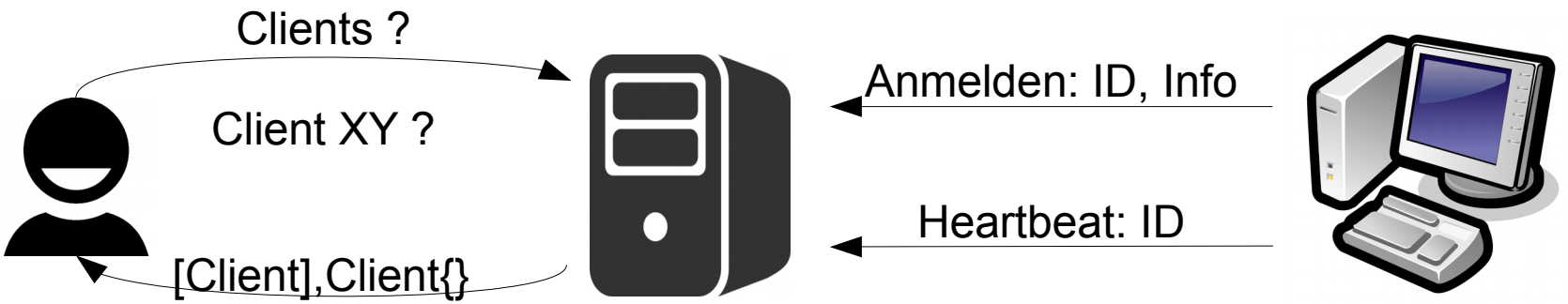
- **Igel UD3-M310C**
  - VIA Nano U3400 (SC, 800 Mhz, 64 Bit, 3.5 Watt TDP)
  - 1GB RAM (max. 2 GB), 1 GB CF (max. 16 GB)
  - Via UniChrome9 (128 MB, DVI-I)
  - 2010 → EOL : 2015
- **Igel UD5-M810C**
  - Intel Celeron 847 (DC, 1.1 GHz, 64 Bit, 17 Watt TDP)
  - 2 GB RAM (max. 4 GB), 1GB CF (max. 16 GB)
  - Intel HD 2000 (384 MB, DVI-I + DP)
  - 2012 → EOL: 2017

Projektbeschreibung

Im Rahmen des Abschußprojekts soll eine Client-/Server Lösung entwickelt werden, welche die Verwaltung und Administration eines (Thin-)Clients implementiert. Für einen erfolgreichen Abschluß müssen folgende Teilaufgaben implementiert werden.

1) Heartbeat / Administration
2) Update
3) Upgrade

Heartbeat



Client := Hostname,IP,
Alive, Datum,
Info
Info := CPU, GPU, RAM, ...

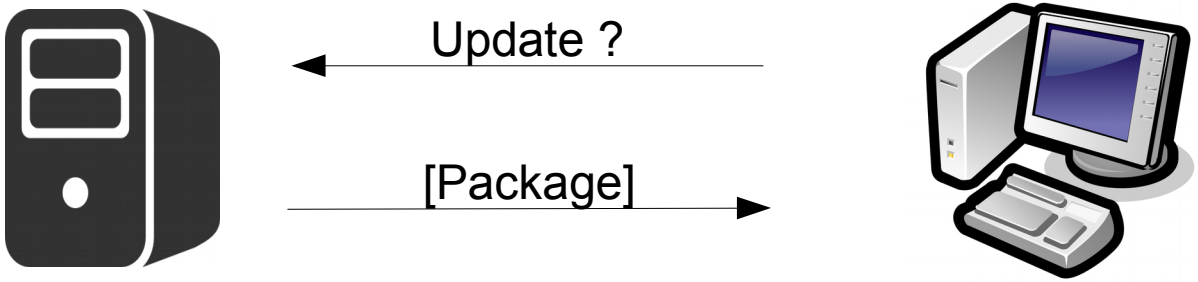
Heartbeat

- `list :: list of client ids`
- `show (id) :: client`

- `hello (id, client info) ::`
- `alive (id) ::`

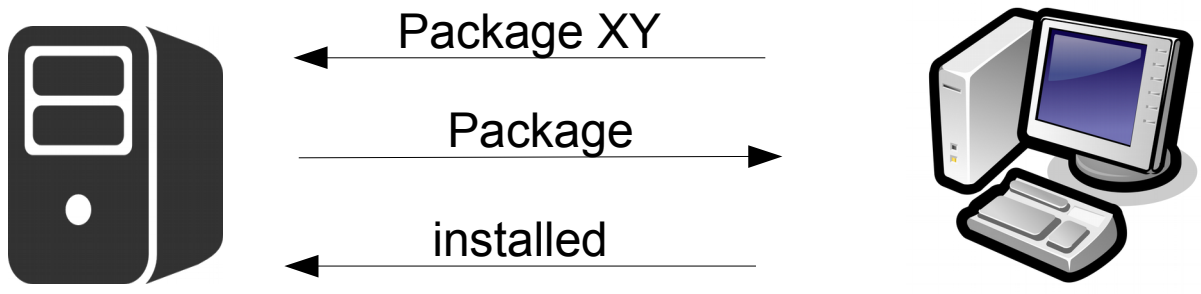
<id> muß über alle Clients eindeutig sein !

Update (verfügbar ?)



Package := Name,
Version,
[Prüfsumme],
URL,
[Command]

Projektbeschreibung : Upgrade



Package := Name,
Version,
[Prüfsumme],
URL,
[Command]

Update / Upgrade

- `update(id) :: list of packages`
- `Upgrade(id, package(_id)) ::`

<id> muß über alle Clients eindeutig sein !

Pakete

- Pakete sind Archive (tar, tgz, zip, …)
- Metainformationen : URL, Prüfsumme, Datum, Version, Abhängigkeiten
- Beispiel:

  `MyClient`

  - URL → `http://localhost:5000/resources/myclient_1.0.tgz`
  - Version → `1.0`
  - Date → `4.6.2018`
  - Dependency → `Python3, Bash`

Erfolgreiche Teilnahme

- GitHub (public)
  - Commit Messages !
  - *Branches*
- Python 3
- *API build Tools (z.B. Swagger, Thrift, GRPC, ...)*
- *JSON,YAML (, XML)*
- Dokumentation
  - PyDoc (Sourcecode)
  - Projektbeschreibung (Markdown)
    - Setup,Usage
    - Beispiel

Erweiterungen

- Sicherheit (Idee + evtl. Umsetzung)
- UI (Browser) für den Server
- Paketlisten nach Abhängigkeit des anfragenden Clients

- Abgabe (symbolisch) am 16.7.2018, 23:59 Uhr
- Vorstellen der Projekte am 17.7.2018 (letzter Termin)