Simple Network 

Handbuch

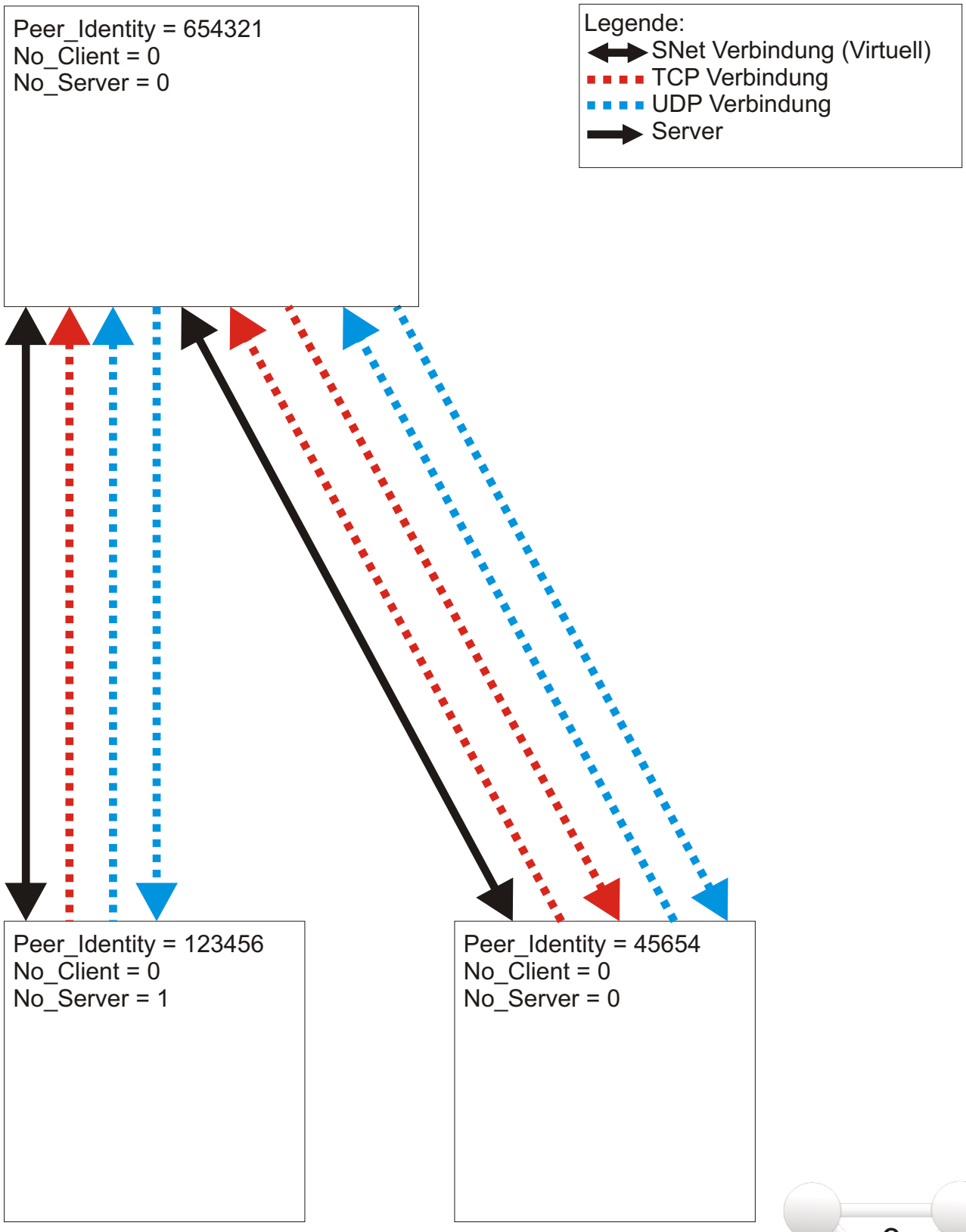
by David Vogel

Inhalt

Voraussetzungen

Übersicht über das P2P Netzwerk

Hier eine Grafik um das Prinzip zu verdeutlichen:



Vorbereitung

Nachdem die Sie die Bibliothek inkludiert haben, beginnen wir mit den Grundeinstellungen:

Alle Einstellungen für die Bibliothek befinden sich in der Struktur “SNet_Options”.
Zu erst muss dem eigenen Peer eine Id (Identity) zugeordnet werden. Sie sollte einmalig sein, da zwei Peer’s mit gleichen Id’s die Verbindung verweigern.
Folgender Code ist dafür meistens ausreichend:

```
SNet_Options\Own_Identity = Random(2147483647)
```

Wenn der Befehl “RandomSeed()” benutzt wird, sollte die eigene Identity vor dem Befehl “RandomSeed()” erstellt werden.

Weitere Einstellungen sind zum Beispiel:

TCP-Port:

```
SNet_Options\Port_TCP = 8240
```

UDP-Port:

```
SNet_Options\Port_UDP = 8241
```

Erstellen eines Servers (TCP) sperren:

```
SNet_Options\No_Server = 0
```

Verbindung zu anderen Server (TCP) für den eigenen Peer sperren:

```
SNet_Options\No_Client = 0
```

Nachdem alle Einstellungen vorgenommen wurden, kann die Bibliothek über SNet_Init() initialisiert werden.

Eventschleife

Jetzt folgt das Abfragen der Events mit Hilfe einer Eventschleife.
Alle Events können über den Befehl SNet_Event() abgefragt werden.
Der Rückgabewert kann folgende Werte annehmen:

#SNet_Event_Peer_New:	Ein neuer Peer hat sich verbunden
#SNet_Event_Peer_Deleted:	Ein Peer hat die Verbindung unterbrochen
#SNet_Event_Data:	Ein Paket wurde empfangen

Bei einem Rückgabewert von 0 liegt kein Event vor.

Jetzt muss man nur noch wissen, welcher Peer den Event ausgelöst hat.
Dafür gibt es den Befehl SNet_Event_Peer().
Dieser gibt die Identity des gegenüberliegenden Peers zurück (SNet_Options\Own_Identity)

Hier ein Beispiel einer Eventschleife:

```
Repeat
  SNet_Event = SNet_Event()
  Select SNet_Event
    Case #SNet_Event_Peer_New
      Identity = SNet_Event_Peer()
      Debug "Neuer Peer: "+str(Identity)

    Case #SNet_Event_Data
      Identity = SNet_Event_Peer()
      Debug "Paket empfangen: "+str(Identity)

    Case #SNet_Event_Peer_Deleted
      Identity = SNet_Event_Peer()
      Debug "Peer gelöscht: "+str(Identity)
  EndSelect

  ...
  ...
Until ...
```

Herstellen einer Verbindung

Das Verbinden zu einem anderen Peer ist mit Simple Network auch sehr leicht.
Es wird nur der Befehl SNet_Connect(IP) benötigt

"IP" ist eine numerische IP-Adresse. Diese kann mit dem PB-Befehl MakeIPAddress(Feld0, Feld1, Feld2, Feld3) erstellt werden. Bei der Benutzung eines IPAddressGadget kann die Rückgabe ohne Umwandeln genutzt werden.

Die Rückgabe des Befehls sagt nur aus, ob SNet_Connect(IP) ohne Fehler ausgeführt wurde, zu diesem Zeitpunkt wird aber noch keine Verbindung hergestellt.

Die neue Verbindung wird als Event gemeldet.

Pakete erstellen und übertragen

Nun kommen wir zu dem wichtigsten Teil: Das Übertragen der Daten in Form von Paketen.

Das Erstellen und Absenden von Paketen:

Zuerst erstellen wir ein Paket, das geschieht mit dem Befehl:

```
SNet_New_Package()
```

Danach kann man das Paket mit Daten füllen, dafür stehen folgende Befehle zu verfügung:

```
SNet_Write_Byte(Value.b)
SNet_Write_Word(Value.w)
SNet_Write_Long(Value.l)
SNet_Write_Quad(Value.q)
SNet_Write_Float(Value.f)
SNet_Write_Double(Value.d)
SNet_Write_String(String.s)
SNet_Write_Data(*Pointer, Length)
```

Zum Abschicken des Pakets gibt es auch noch einen Befehl:

```
SNet_Close_Package(Identity, Priority, UDP)
```

Identity:	Die Nummer, die von SNet_Event_Peer() zurückgegeben wurde.
Priority:	Die Priorität des Pakets. von 0(Höchste) bis 2147483647(Niedrigste). Beachten Sie aber, dass die Priorität 0 für interne Aufgaben belegt ist (Ping...). Benutzen Sie daher den Bereich von 1 bis 2147483647.
UDP:	Das Paket soll (wenn möglich) über UDP gesendet werden. (Sollte 1 sein)

Pakete erstellen und übertragen

Das Empfangen und Verarbeiten von Paketen:

Nachdem der gegenüberliegende Peer ein Paket versandt hat,
wird das Event #SNet_Event_Data Gemeldet.

Das Paket wird automtisch “geöffnet” und zum lesen bereitgestellt.
Es ist jetzt nicht mehr zu tun, als folgende Befehle zu benutzen:

```
Value.b =    SNet_Read_Byte()
Value.w =    SNet_Read_Word()
Value.l =    SNet_Read_Long()
Value.q =    SNet_Read_Quad()
Value.f =    SNet_Read_Float()
Value.d =    SNet_Read_Double()
String.s =   SNet_Read_String()
Result =     SNet_Read_Data(*Pointer, Length)
```

Tips:

Damit die Kommunikation richtig funktioniert sollte am Anfang des Pakets ein “Inhalts” Byte stehen, welches Auskunft über den weiteren Inhalt des Pakets gibt. Damit weiß ein Peer wie er empfangene Daten zu interpretieren hat.
Zum Beispiel bei einem Onlinespiel:

Login:

Inhalt	Name	Passwort							
Byte	String	String							
1	“Horst”	“123”							

Spieler Bewegen:

Inhalt	X	Y	Speed						
Byte	Long	Long	Byte						
2	30	101	3						

Extras

Hier sind noch zusätzliche Befehle aufgelistet:

Ping eines Peers abfragen:

```
Ping = SNet_Peer_Get_Ping(Identity)
```

Dieser Befehl gibt die Ping des Peers zurück. Er ist schnell, weil er ein zwischengespeichertes Ergebnis verwendet.

IP eines Peers abfragen:

```
IP = SNet_Peer_Get_IP(Identity)
```

Dieser Befehl gibt die numerische IP des Peers zurück. Er ist schnell, weil er ein zwischengespeichertes Ergebnis verwendet.

Verbindung mit Peer beenden:

```
SNet_Peer_Disconnect(Peer_Identity.l)
```

Dieser Befehl bricht die Verbindung zu einem Peer ab. Es wird auf der eigenen Seite und auf der gegenüberliegenden Seite das Event #SNet_Event_Peer_Deleted ausgegeben.