

Inhaltsverzeichnis

1	Einleitung	2
1.1	Begriffs-Definition	2
2	Verbindungs-Management	2
2.1	Netz-Anmeldung	4
2.2	Verzögerungstaktik	5
3	Daten-Synchronisation	5
3.1	Neues Posting/neuer Thread	5
3.2	Archivierung von Threads	5
3.2.1	Erzwungene Archivierung durch einen Administrator	6
3.3	Löschung von Threads	6
3.4	Synchronisation von Plugin-Spezifischen Daten	6
4	Datentypen	6
4.1	Thread-ID	6
4.2	Posting-ID	6
4.3	Servertoken	6
4.4	Unique-ID	7
5	Angedachte Einsatzumgebung	7
6	Protokoll-Definition	7
6.1	Verbindungsaufnahme	7
6.2	Verbindungs-Abbau	8
6.3	Neuer Thread	8
6.4	Neues Posting	8
6.5	Vereinbarung einer ID	8

1 Einleitung

Die zunehmende Last des SELFHTML Servers legt es nahe, ein verteiltes Foren-System zu entwickeln. Prinzipiell sind derartige Möglichkeiten bereits vorgesehen, lediglich die Protokoll-Spezifikation und die Implementation warten.

1.1 Begriffs-Definition

Broadcast Mit einem Broadcast ist hier kein IP-Broadcast gemeint, sondern ein Netzwerk-interner Vorgang, bei dem jeder Peer im Netzwerk befragt wird.

Peer Ein Peer ist ein Teilnehmer des Netzwerk-Verbunds.

Route Hier ist nicht die IP-Route gemeint, sondern ein Pfad zu einem Peer in dem Netzwerk-Verbund.

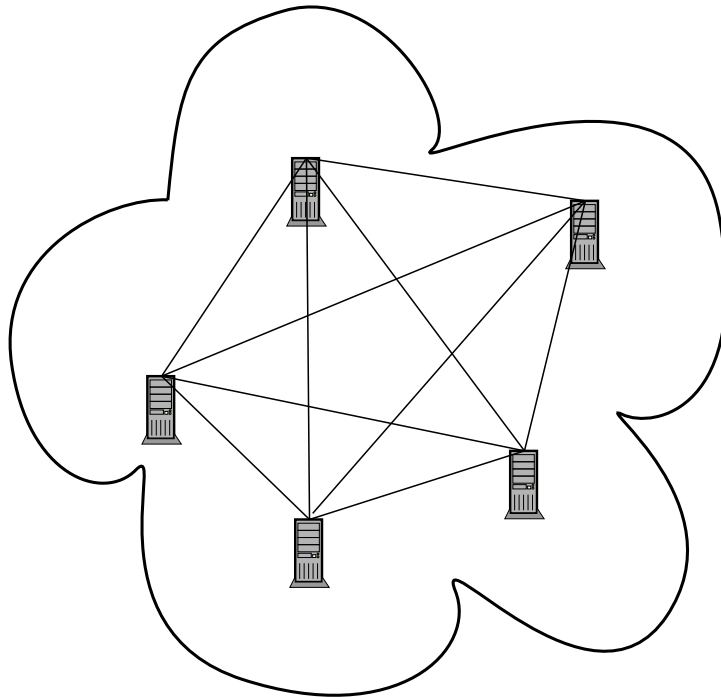
Netzverbund, Verbund Hiermit ist die Menge aller Peers gemeint.

Servertoken Ein Servertoken ist ein Peer-spezifisches Text-Element mit mindestens 8 Byte Länge. Ein Servertoken muß im Verbund eindeutig sein!

Byte Ein Byte ist hier definiert als 8 Bit im Intel-Format (little endian).

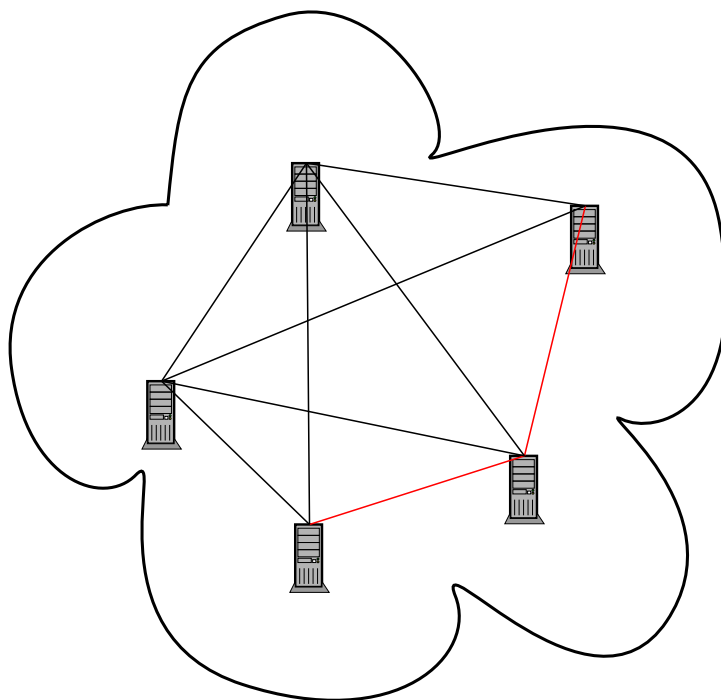
2 Verbindungs-Management

Aufgrund der Wahrscheinlichkeit von Netzausfällen halte ich eine Dezentralisierung für sinnvoll, deshalb ist erstmal jeder Peer mit jedem Peer verbunden:



Sollte nun eine Verbindung ausfallen, so wird erst versucht, den Server erneut zu erreichen. Zufällig verzögert¹ wird ein Broadcast losgeschickt, in dem gefragt wird, ob der Server über einen anderen Host erreicht werden kann. Ist dies der Fall, so wird bis zur erfolgreichen Wiederherstellung der direkten Verbindung diese Route zur Kommunikation gewählt. Stehen mehrere Routen zur Verfügung, wird die direkteste Route ausgewählt. Gibt es mehrere Routen derselben Direktheitsstufe, so wird **zufällig** eine der Routen ausgewählt:

¹Siehe: Verzögerungstaktik



Sollte ein Peer komplett aus dem Verbund gefallen sein, so muß auch der Broadcast im Netzwerk fehlschlagen. In diesem Fall wird ein Eintrag in der Log-Datei gemacht und der Peer ansonsten komplett ignoriert. Sind die Netzprobleme behoben, so muß sich dieser Peer bei **jedem** Peer im Netzwerk-Verbund erneut anmelden.

2.1 Netz-Anmeldung

Ist ein Peer aus dem Verbund gefallen (weil er neu gestartet wurde, weil es einen Netzausfall gab, weil er zum ersten mal gestartet wird, etc, pp), so versucht er direkt beim Startup eine Verbindung zu allen anderen Peers herzustellen. Gelingt die Verbindung nicht, muss er das Ereignis loggen und es erneut versuchen. Zwischen jedem Verbindungsversuch müssen mindestens drei, höchstens jedoch fünf Sekunden liegen. Kommen die Verbindungen von zwei Peers zueinander gleichzeitig zustande, so muss vereinbart werden, welche der beiden TCP-Verbindungen benutzt werden soll. Dazu wird in der Konfiguration eine Prioritäten-Zahl vergeben. Der Peer muss diese nun an seinen Gegenüber senden. Die Verbindung des Peers mit der höheren Priorität wird verwandt. Haben zwei Peers die gleiche Priorität, so werden beide Verbindungen geschlossen und nicht wieder aufgebaut, da hier ein Fehler in der Konfiguration vorliegen muss!

Ist nun erfolgreich eine Verbindung zu einem der anderen Peers zustande gekommen, so muss er eine sog. Synchronisations-Phase einleiten. Sollte

der Sonderfall der gleichzeitigen Verbindung aufgetreten sein, so muss der Client mit der niedrigeren Priorität diese Phase einleiten. Er muss sicherstellen, dass der Peer denselben Datenbestand hat wie er. Dazu werden alle Postings, die einer der beiden Peers nicht hat, in den Datenbestand desjenigen mit aufgenommen. Sollten Differenzen auftreten, so muss im Falle einer Wieder-Aufnahme in den Verbund der Kontakt aufnehmende Peer sich nach dem richten, was von den anderen kommt. Sollte vorher der Fall einer gleichzeitigen Verbindung eingetreten sein, so werden die Differenzen des Peers mit der höheren Priorität übernommen.

2.2 Verzögerungstaktik

Bei Netz-Problemen kann es sein, dass ein Host komplett aus dem Verbund ausfällt. Dieser Fall kann dazu führen, dass jeder Peer im Verbund gleichzeitig einen Broadcast losschickt. Um derartige Netzspitzen zu vermeiden, wird eine solche Massnahme in einem Zeitraum zwischen 0 und 5 Sekunden verzögert.

3 Daten-Synchronisation

3.1 Neues Posting/neuer Thread

Wird ein neues Posting oder ein neuer Thread (und damit gleichzeitig ein neues Posting) geschrieben, so muß zuerst die Posting- (und Thread-ID) festgelegt werden. Hier gibt es zwei Möglichkeiten: entweder, es wird ein Servertoken in die Posting- und Thread-ID mit einbezogen, so dass die ID eindeutig wird, oder die ID muß im Verbund vereinbart werden. Die Wahl ist hier der Implementation freigestellt: beides hat Vor- und Nachteile. Eine Absprache würde den Verwaltungs-Overhead noch weiter erhöhen, wohingegen ein Servertoken keine rein numerischen IDs mehr ermöglicht.

Nach Erteilung einer ID muß im Verbund nachgefragt werden, ob das Posting bereits existiert. Dies passiert über die Unique-ID, die jedem Posting zugeteilt wird. Ist die Unique-ID bereits vergeben, so darf das Posting nicht angenommen werden und muß mit einer entsprechenden Fehlermeldung dem User reklamiert werden.

Sind die obigen Vereinbarungen getroffen, so wird das neue Posting an den kompletten Verbund verteilt. Die Peers im Verbund sind **verpflichtet**, dieses Posting/diesen Thread anzunehmen. Eventuelle Rückweisungen müssen bereits vorher geschehen sein.

3.2 Archivierung von Threads

Soll ein Thread archiviert werden, so muß auch dies vorher im Verbund bekannt gegeben werden. Jeder Peer hat dann zu prüfen, ob er im Falle einer

Archivierung auf denselben Schluss kommen würde. Ist dies nicht der Fall, so darf der Thread nicht archiviert werden. Die Archivierung wird dann verschoben auf einen späteren Zeitpunkt. Die Ablehnung eines Archivierungsvorgangs muß geloggt werden, so dass bei eventuellen Fehlern der Administrator eingreifen kann.

3.2.1 Erzwungene Archivierung durch einen Administrator

Bei einer erzwungenen Archivierung hat kein Peer Mitspracherecht und der Thread **muß** archiviert werden.

3.3 Löschung von Threads

Wird ein Thread gelöscht, so wird dies dem Verbund bekannt gegeben. Es besteht kein Widerspruchsrecht. Eine Löschung ist verbindlich und muß von jedem Peer im Verbund akzeptiert werden.

3.4 Synchronisation von Plugin-Spezifischen Daten

Jedes Plugin **muß** die Möglichkeit der Synchronisation haben. Plugins wie `flt_visited` und dergleichen müssen die Möglichkeit besitzen, ihre Datenbestände abzugleichen. Dafür muß im Multiserver-Betrieb ein weiterer Handler-Hook eingeführt werden.

4 Datentypen

4.1 Thread-ID

Bei rein numerischen IDs muß die ID **mindestens** 4 Byte gross sein, **höchstens** jedoch 8 Byte. Im ersten Fall muß die ID Vorzeichenlos sein, im zweiten Fall darf die ID ein Vorzeichen enthalten. Sollte die Wahl der Implementation auf Text-IDs fallen, so muß eine ID mindestens 8 Byte gross sein, abzüglich des Servertokens. Die Maximal-Größe für Text-IDs ist nicht definiert, sollte jedoch unter 255 Byte bleiben.

4.2 Posting-ID

Für eine Posting-ID gelten dieselben Regeln wie für eine Thread-ID.

4.3 Servertoken

Das Servertoken muß mindestens 8 Byte gross sein, die maximale Größe ist jedoch auch hier nicht festgelegt, sollte jedoch unterhalb von 255 Byte liegen.

4.4 Unique-ID

Eine Unique-ID muß 50 vorzeichenlose Byte betragen. Ihre Eindeutigkeit innerhalb des Verbunds muß durch einen angemessenen Algorithmus sichergestellt sein. Dies darf jedoch **nicht** durch ein Servertoken passieren.

5 Angedachte Einsatzumgebung

Die Umgebung, für die dieses Konzept gedacht ist, ist sicher nicht das Internet. Das Protokoll ist einfach zu "geschwätzig". Tatsächlich wird ein internes Netz angedacht, das mindestens 100MBit switched ist. Dort ist auch keine verschlüsselte Kommunikation notwendig. Und damit vom fo_view-Prozess nicht jedesmal eine neue Verbindung zum Server aufgenommen werden muß, wird eine Persistente Verbindung entweder über ein Webserver-Modul oder über einen fastcgi-Prozess angedacht.

6 Protokoll-Definition

Das Protokoll dieses Konzepts ist für eine reine Intranet-Umgebung gedacht. Es wäre äußerst unklug, die Verbindung extern offen zu halten: jeder Hans und Franz könnte Problemlos so tun als sei er ein Partner aus dem Verbund und hat damit vollen Zugriff auf die Postings.

6.1 Verbindungsaufnahme

```
P1: 100 Hello, this is <codename> speaking.
P2: 200 Hi there. This is <codename>.
P1: 100 Got another connection for you. My Priority is 10.
P2: 100 Got another connection for you. My priority is 11.
P2: 201 Using this channel no. Closing other channel.
P2: 100 Initializing synchronization phase. Go and tell me what you've got.
P1: 200 Ok, here you are:
P1: THREAD t1
P1: MSG m1
P1: ...
P2: 400 t1 is different for me. Taking my version:
P2: THREAD t1
P2: MSG m1
P2: ...
P1: 200 Ok, got it
P2: 400 Additionally I've got t15
P2: THREAD t15
P2: MSG m10
P2: ...
```

P1: 200 Ok, got it.
P2: 200 Request phase finished.
P1: 200 Ok, you're right.

6.2 Verbindungs-Abbau

P1: 100 Going down now
P2: 200 Ok, bye then

6.3 Neuer Thread

P1: 100 Got new thread
P1: THREAD t1
P1: MSG m1
P1: ...
P2: 200 Got it

6.4 Neues Posting

P1: 100 Got new Posting
P1: THREAD t1
P1: ANSWER TO m1
P1: MSG m2
P1: ...
P2: 200 Got it

6.5 Vereinbarung einer ID

P1: 100 Need new ID, suggesting <id>
P2: 200 Ok
P3: 400 No, I've got <id> already. Take <id1>
P1: 100 Need new ID, suggesting <id1>
P2: 200 Ok
P3: 200 Ok