

Zeitsynchronisation in Netzwerken mit SNTP

Alle ticken richtig

Netzwerkrechner sollten auf die gleiche Zeit eingestellt sein, denn in bestimmten Situationen kann es sonst zu Problemen kommen. Die Aufgabe der Synchronisierung löst das Simple Network Time Protocol (SNTP). dotnetpro zeigt, wie es funktioniert und wie Sie einen eigenen SNTP-Server für Ihr Netzwerk aufsetzen können.

Die Zeitsynchronisation hat in Netzwerken einen hohen Stellenwert. Weicht die Zeit von zwei Servern in einer Active-Directory-Domäne zu stark ab, klappt die Replikation nicht mehr. Somit sind Benutzer- und Computerkonten nicht mehr synchron, und im schlimmsten Fall kann sich ein Benutzer nicht mehr anmelden. Innerhalb eines LANs, aber auch über WAN-Strecken und das Internet, kommt hier das Simple Network Time Protocol (SNTP) zum Einsatz.

In allen Windows-Versionen dient dazu der Befehl `net time`. Ein Aufruf von

```
net time /set <servername>
```

setzt Datum und Uhrzeit des Clients auf die Zeitinformationen des angegebenen Servers. Ab Windows 2000 kann zudem der Parameter

```
setsntp:<servername>
```

verwendet werden, der den Zeitgeberdienst anweist, sich regelmäßig mit dem angegebenen Server abzugleichen. Weitere Parameter von `net time` finden Sie in der Hilfe.

Bei Problemen mit der Synchronisierung bietet sich – wieder ab Windows 2000 – der Einsatz des Befehls `W32TM` an. Auch hier findet sich in der Hilfe eine ausführliche Beschreibung der verfügbaren Parameter.

Als Zeitquelle können Sie dabei nicht nur jeden beliebigen PC im LAN angeben, sondern natürlich auch SNTP-Server im Internet. Davon gibt es mehr als genug, zum Beispiel `ptbtime1.ptb.de` von der physikalisch-technischen Bundesanstalt oder `ntp1.t-online.de` vom „rosa Riesen“. Eine umfassende Liste finden Sie auf www.ntp.org. Dort wird unterschieden zwischen Poolservern, Primary Servers und Secondary Servers. Die Poolserver sind ausfallsicher konfiguriert, das heißt, die Anfrage wird mittels Round-Robin-Verfahren auf mehrere Server ver-

teilt. Sollte ein Server einmal ausfallen, kann die Anfrage von seinen Kollegen bedient werden. Die Primary Servers beziehen ihre Zeit in der Regel direkt von einer offiziellen Zeitquelle, zum Beispiel einer Atomuhr, die Secondary Servers gleichen sich mit bei den Primary Servers ab. Um herauszufinden, ob die Zeitquelle ein Primary oder Secondary Server ist, dient das Byte 2 (Stratum) der NTP-Datenstruktur.

Aber wir sind ja keine Administratoren, wir sind Entwickler! Deshalb soll hier nicht der Einsatz der Bordmittel beschrieben werden, sondern ein eigener Dienst entwickelt werden. Im Weiteren wird das SNT-Protokoll beschrieben und im Anschluss daran als .NET-Programm umgesetzt.

Die Theorie

Die Beschreibung für das SNT-Protokoll ist in RFC 2030 zu finden [2]. Standardmäßig läuft SNTP auf UDP-Port 123. Der Server erwartet hier ein Datenpaket mit dem Inhalt `0x1B` im ersten Byte. Er antwortet mit einem Datenstrom aus 48 oder 68 Byte. Die letzten 20 Bytes sind optional und können entfallen.

Das vorliegende Beispiel verwendet sie nicht. Tabelle 1 führt den Inhalt des Datenstroms auf.

Timestamp

Die Timestamp-Felder (beschrieben in RFC 1305, das auch das erweiterte NTP beschreibt [3]) erhalten die vergangenen Sekunden seit dem 01.01.1900, 00:00 Uhr. Diese können einfach auf Datum und Uhrzeit umgerechnet werden:

Die Daten sind in einer 64-Bit-Zahl (8 Bytes) gespeichert. Die ersten vier Bytes enthalten die Sekunden, die folgenden vier Bytes den Fraktalanteil, also die Stellen „nach dem Komma“. Es sind somit auch millisekunden- und noch genauere Anga-

ben möglich. Näheres hierzu finden Sie in der RFC 1305 [3]. Die Umrechnung der Sekunden in Datum und Uhrzeit erfolgt gemäß dem Algorithmus in Listing 1.

Damit aber stoßen Sie bereits auf ein Problem: SNTP wäre in der vorliegenden Form nur bis zum Jahr 2036 nutzbar, da dann ein Überlauf der Sekunden stattfindet. Ja, vier Milliarden Sekunden seit dem 1.1.1900 sind schnell vergangen. Die Entwickler haben sich aber hierzu bereits Gedanken gemacht: Ist das Bit 0 gesetzt (=true), bezieht sich die Zeitangabe auf die Jahre 1968 bis 2036. Ist es nicht gesetzt, wäre es eigentlich 1900 bis 1968; per Definition ist aber 2036 bis 2104 gemeint.

Der Client

Aber der Reihe nach: Nach dem Initialisieren des Daten-Arrays wird das erste Byte auf den bereits angesprochenen Wert `0x1B` und `_transmitTimestamp` auf die aktuelle Uhrzeit des Clients gesetzt:

Auf einen Blick

Autor

Andreas Müller ist Geschäftsführer der IkarusSoft GmbH, die kleinen und mittleren Unternehmen Hilfestellung rund um die EDV bietet. Sie erreichen ihn unter andreas.mueller@ikarussoft.de.



dotnetpro.code
A0506TimeServer



Sprachen C#

Technik Simple Network Time Protocol (SNTP)

Voraussetzungen .NET SDK

Listing 1

Umrechnung des Timestamps in Datum und Uhrzeit.

```
private DateTime GetTimestamp(int offset)
{
    DateTime time = ComputeDate(GetMilliseconds(offset));
    long tmpOffset =
    TimeZone.CurrentTimeZone.GetUtcOffset(DateTime.Now).Ticks;
    TimeSpan span = TimeSpan.FromTicks(tmpOffset);
    return (time + span);
} // end GetTimestamp

private DateTime ComputeDate(ulong milliseconds)
{
    TimeSpan span = TimeSpan.FromMilliseconds((double)milliseconds);
    DateTime time = new DateTime(1900, 1, 1);
    return (time + span);
} // end ComputeDate

private ulong GetMilliseconds(int offset)
{
    ulong part_int = 0;
    ulong part_fractal = 0;
    for (int i=0; i<=3; i++)
    {
        part_int = 256*part_int + _ntpData[offset+i];
        part_fractal = 256*part_fractal + _ntpData[offset+i+4];
    }
    ulong ms = (1000*part_int+(1000*part_fractal)/0x100000000L);
    return ms;
} // return GetMilliseconds
```

Listing 2

Bitshifting für die Properties Leap Indicator, Version und Mode.

```
...
// -> Leap Indicator
byte val = (byte)(_ntpData[0] >> 6);
switch (val)
{
    case 0: _leapIndicator = LeapIndicator.NoWarning; break;
    case 1: _leapIndicator = LeapIndicator.LastMinute61; break;
    case 2: _leapIndicator = LeapIndicator.LastMinute59; break;
    default: _leapIndicator = LeapIndicator.Alarm; break;
}

// -> Version number (3 or 4)
_version = (byte)((_ntpData[0] & 0x38) >> 3);

// -> Mode
val = (byte)(_ntpData[0] & 0x7);
switch (val)
{
    case 1: _mode = Mode.SymmetricActive; break;
    case 2: _mode = Mode.SymmetricPassive; break;
    case 3: _mode = Mode.Client; break;
    case 4: _mode = Mode.Server; break;
    case 5: _mode = Mode.Broadcast; break;
    default: _mode = Mode.Unknown; break;
}
...
```

```
_ntpData = new byte[SNTP_DATA_SIZE];
// version = 4, mode = 3
_ntpData[0] = 0x1B;
// init byte array with 0
for (int i=1; i<48; i++) _ntpData[i] = 0;
_transmitTimestamp = DateTime.Now;
```

Nach dem Ermitteln der IP-Adresse des Servers mittels *Dns.Resolve* wird die Anfrage per *UdpClient* gesendet und der empfangene Datenstrom wieder in *_ntpData* zurückgeschrieben:

```
IPAddress address =
Dns.Resolve(servername).AddressList[0];
IPEndPoint endPoint = new IPEndPoint(address,
portnumber);
UdpClient udpClient = new UdpClient();
udpClient.Connect(endPoint);
udpClient.Send(_ntpData, _ntpData.Length);
_ntpData = udpClient.Receive(ref endPoint);
_receptionTimestamp = DateTime.Now;
```

Die empfangenen Daten werden auf die entsprechenden Bytes aufgeteilt. Beim ers-

ten Byte ist dabei zu beachten, dass hier gleich drei Properties enthalten sind. Diese müssen mittels Bitshifting und/oder And-Verknüpfungen entsprechend extrahiert werden, wie es Listing 2 demonstriert.

Eine weitere Besonderheit ist der *ReferenceIdentifier*. Hier hängen die Daten vom Feld *Stratum* ab. Bei einer primären Referenzquelle, zum Beispiel einem Primary Server, steht hier ein String mit 4 Bytes. Bei einer sekundären Referenzquelle, etwa einem Secondary Server, wird unterschieden zwischen den Versionen 3 und 4 des Protokolls: Version 3 liefert eine IP-Adresse, Version 4 die Zeitangabe der letzten Synchronisierung mit einer primären Quelle.

Ein einfacher Server

Auch die Implementierung eines einfachen SNTP-Servers, wie ihn Abbildung 1 zeigt, ist kein Hexenwerk. Nach dem Star-

Abbildung 1
Immer pünktlich mit dem Simple Network Time Protocol.



ten des Servers holt er sich die Zeit erst einmal von einem Primary Server. Dabei wird nicht berücksichtigt, dass die Quelle auch einmal offline sein kann, es wird also kein Timeout abgefangen. Listing 3 zeigt den Code für den UDP-Listener.

Danach wird die Systemzeit der Maschine mittels der Win32-API-Funktion *SetSystemTime* gesetzt. Anschließend wird ein Socket auf Port 124 mit einem UDP-Protokoll erzeugt. Port 124 wird deshalb benutzt, weil Windows den Port 123 bereits für NTP belegt. Nun wird auf dem Socket so lange gehorcht, bis eine Anfrage kommt, deren erstes Byte 0x1B

Tabelle I

Aufbau der SNTP-Sequenz.

Byte / bit	Inhalt	Beschreibung
1 / 7-8	Leap Indicator	Anzeige für die „Schaltsekunde“ des aktuellen Tages: 0 = keine Schaltsekunde 1 = die letzte Minute hat 61 Sekunden 2 = die letzte Minute hat 59 Sekunden 3 = die Uhr hat eine Störung
1 / 4-6	Version Number	Versionsnummer des Protokolls (3 oder 4)
1 / 1-3	Mode	Modus der Quelle 1 = aktiv symmetrisch 2 = passiv symmetrisch 3 = Client 4 = Server 5 = Broadcast (Rundruf) 0, 6, 7 = reserviert
2	Stratum	Referenzquelle 0 = nicht verfügbar bzw. nicht spezifiziert 1 = primäre Referenz, z. B. eine Atomuhr 2-15 = sekundäre Referenz, z. B. ein anderer SNTP-Server 16-255 = reserviert
3	Poll	Maximales Poll-Intervall
4	Precision	Präzision der zugrunde liegenden Uhr
5-8	Root Delay	Verzögerung zu einer primären Zeitquelle
9-12	Root Dispersion	Fehler relativ zu einer primären Zeitquelle
13-16	Reference Identifier	Referenz-Kennung; entweder ein 4 Zeichen langer String oder eine IP-Adresse
17-24	Reference Timestamp	Die Zeit, zu der die lokale Quelle zuletzt synchronisiert wurde
25-32	Originate Timestamp	Die Zeit, zu der die Anfrage vom Client gesendet wurde
33-40	Receive Timestamp	Die Zeit, zu der die Anfrage vom Server empfangen wurde
41-48	Transmit Timestamp	Die Zeit, die vom Server gesendet wurde (die Referenzzeit)

Listing 3

Der UDP-Listener für den Server.

```
private void ServerMode()
{
    // Zeit holen
    int port = Convert.ToInt32(txtPort.Text);
    Sntp sntp = new Sntp(txtServer.Text, port);
    txtTime.Text = sntp.TransmitTimestamp.ToString();

    // Listener starten
    Socket udpServer = new Socket(AddressFamily.InterNetwork,
        SocketType.Dgram, ProtocolType.Udp);
    IPEndPoint localhost = Dns.GetHostByName(Dns.GetHostName());
    IPEndPoint localIp = new IPEndPoint(localhost.AddressList[0], 124);
    udpServer.Bind(localIp);

    while (serverActive)
    {
        byte[] received = new byte[48];
        IPEndPoint tmpLocalIp = new IPEndPoint(localhost.AddressList[0], 124);
        // IP vom Client holen
        IPEndPoint remoteEP = (tmpLocalIp);
        int bytesReceived = udpServer.ReceiveFrom(received, ref remoteEP);

        if (received[0] == 0x1b)
        {
            // Daten an Client senden
            byte[] sendData = sntp.GetActualTime();
            udpServer.SendTo(sendData, remoteEP);
        }
    }
}
```

beinhaltet. Das zuvor erzeugte SNTP-Objekt wird mit der aktuellen Uhrzeit versehen und einfach an den Client geschickt.

Fazit

Natürlich liefert der Server bei der Antwort nicht die richtigen Stratum-Werte

und ein Server sollte eigentlich als Dienst laufen. Die Entwicklung von Windows-Diensten wurde bereits in [4] gezeigt. Aber in erster Linie ging es hier darum, die Verwendung des SNT-Protokolls aufzuzeigen. Erweiterungen sind selbstverständlich möglich und – für einen Praxis-einsatz – auch notwendig. |||||

[1] ntp.isc.org/bin/view/Servers/NTPPool-Servers

[2] www.eecis.udel.edu/~mills/database/rfc/rfc2030.txt

[3] www.eecis.udel.edu/~mills/database/rfc/rfc1305/rfc1305a.pdf

[4] Andy Kafourous, Fünf-Sterne-Service, dotnetpro 6/2004, Seite 96 ff.