MobiLink と TCP/IP の Keep-Alive

MobiLink の TCP/IP ベースの通信ストリームには、"keep_alive"オプションが用意されています。ここでは、このオプションを効果的に設定する方法について説明します。

keep_alive オプションとは?

keep_alive オプションは、TCP/IP 接続での*活性チェック*を有効または無効にします。活性チェックとは、ネットワーク接続の相手側が接続を維持している(接続が生きている)状態かどうかをチェックするプロセスです。送信側では、接続の切断はデータ送信を試行するとすぐに検出されるため、この活性チェックは、データの受信を待機している接続の終端側にのみ適用されます。

活性チェックが重要である理由

活性チェックによって、接続が失われた場合に、接続を待機している側がいつまでも待ち続けるのを防ぐことができます。これによって、MobiLink などのサーバが、切断されたクライアント接続を破棄できます。活性チェックを行わなければ、サーバは失われた接続に対して、まだ接続が生きていると誤解して非常に多くのリソースを使うことになります。クライアント側では、活性チェックは接続の障害を検出するのに役立ち、ユーザが障害を認識して、後でもう一度試したり別のサーバを試すことが可能になります。
活性チェックを行わなければ、接続の切断によって、クライアントまたはクライアントが接続されていた MobiLink のワーカー・スレッドのいずれかがハングします。MobiLink が、クライアントからのデータを待機する同期接続の最中であった場合、そのワーカー・スレッドはデータを待機し続けるため、他の同期接続を利用できなくなります。クライアントが MobiLink からのデータを待機している場合、クライアントはデータを待機し続けるため、同期接続を完了できなくなります。MobiLink の同期接続では双方向の通信が行われるため、MobiLink サーバとクライアントの両方で活性チェックを有効にして、通信が切断された場合にどちらの側もハングするのを防ぐ必要があります。

MobiLink のサーバおよびクライアントで keep_alive オプションを設定する方法

MobiLink の同期接続の際に使用される TCP/IP ベースのストリームでは、クライアント側とサーバ側の両方で次の新しいパラメータを利用できるようになりました。

```
keep_alive=[0|1]
```

たとえば、7.0.x MobiLink で活性チェックを有効にするには、コマンドラインで次のように指定します。

```
dbmlsrv7 -x tcpip{keep_alive=1} ....
```

0 は TCP/IP の活性チェックをオフにし、1 はオンにします。デフォルトは 1 です(活性の検出が有効な状態)。ネットワーク・レイヤで TCP/IP の活性チェックがサポートされていない場合、このパラメータは無視されます。当初はデフォルトは 0 でしたが、当社の顧客がこのオプションをオンにして問題が解決されたことが多かったため、デフォルトは 1 に変更されました。**疑わしい場合は、keep_alive オプションを明示的に設定してください。**
MobiLink クライアントで TCP/IP ストリームのオプションを設定する方法については、『UltraLite and ASA client for MobiLink』のマニュアルを参照してください。
SQL Anywhere Studio の以下のリリース(および EBF)と、それ以降のすべてのリリースでは、keep_alive オプションが利用可能になっています。

- 7.0.0.455
- 7.0.1.1103
- 7.0.2

ただし、通常は keep_alive オプションを 1 に設定しても十分ではありません。TCP/IP の活性チェックに関するオペレーティング・システムの設定を、使用するアプリケーションに合わせて変更する必要がありますが、その前に活性チェックの詳細をある程度理解しておく必要があります。

TCP/IP の活性チェックのしくみ

OS のネットワーク・レイヤにおける活性チェックの実装は、OS ごとに微妙に異なりますが、以下の基本的なアルゴリズムが当てはまります。

```
IF(TCP/IP 接続でのデータを最低 Ti 時間は待機する) {
    //  活性チェックの開始：
    Repeat = Nm
    WHILE(相手側から応答がなく、Repeat > 0) {
        keep-alive メッセージを送信する
        Td 時間まで応答を待機する
        Repeat <- Repeat - 1
    }
    IF(応答がなかった) {
        接続の切断を宣言する
    } ELSE {
```

データを待機する時間のカウントをリセットする

}

}

注意：

太字の値(**Ti**、**Nm**、および **Td**)は、ここでの説明用として作成した名前の例で、OS/ネットワークの設定を表します。

- Ti はアイドル時間のスレッショルドです。この時間以上アイドルになっている接続は、活性チェックの候補となります。
- **Nm** は、接続の切断が宣言されるまでに送信可能な keep-alive メッセージの最大数です(符合なし整数)。
- **Td** は、keep-alive メッセージの間隔として適用される最小遅延です。
- 接続の切断が宣言された場合は、その接続で現在または後続の通信を行おうとすると、すぐにエラーになって失敗します。
- TCP/IP の実装形態によっては、**Td** に数式を適用して幾何学的な遅延の進行を利用する場合もありますが、ここでは説明しません。
- keep-alive メッセージは、単純な低レベルのメッセージです。接続が維持されている場合、相手側は基本的な TCP/IP 機能を利用して応答し、送信側アプリケーションには影響を与えません。

活性チェックを設定する際は、接続がアイドル状態のときに、実際にチェックが開始されるまで待機するアイドル時間のスレッショルド(**Ti**)を選択して設定することが重要です。その他のパラメータは、通常は変更する必要はありません。

アイドルのスレッショルドの選択

Ti の選択は、特にネットワーク特性などの運用形態によって異なります。通常は、活性チェックのパケットがネットワークのパフォーマンスに影響を与えない程度に、間隔をかなり短めに設定します。LAN ベースの運用形態では、間隔を短めに設定することが可能です。ワイヤレスまたはモデム・ベースのネットワークなどの低速なネットワークでは、間隔を長めに設定する必要があります。

それでは、間隔はどの程度の長さに設定したらよいでしょうか? 接続の切断は時を選ばずに発生しますが、ここで関係する同期接続の遅延では 2 つの遅延が予想されます。1 つ目は、クライアントがアップロードを認められるまで待機する遅延です。2 つ目は、クライアントがダウンロードを適用して受信確認を送信するまでサーバが待機する遅延です。この 2 つ目の遅延のほう

が、1つ目の遅延よりも長い傾向があります。適用するデータの量を等しくした場合、通常は非常に高速なハードウェア上で動作するサーバの方が、デスクトップやノートパソコン、または携帯/内蔵型デバイスであるクライアントよりもアップロードを高速に処理します。
これらの遅延の間は、ネットワーク・トラフィックは予想されないため、接続はアイドル状態となります。つまり、遅延が十分に長いと動作チェックが行われることになります。そのため、これらの遅延の予想される長さが分かっている場合は、間隔を2倍または3倍くらい長めに設定する必要があります。
また、運用形態が、定期的に繰り返し実行される同期接続に依存している場合は、間隔を次の同期接続が試行される前に接続の切断を検出できる値に設定するとよいでしょう。
間隔を選択するには、同期接続の際に自然に発生する遅延を含めた、同期接続の平均時間を正確に分析することが必要です。平均時間を特定するには、1つの同期接続の時間だけでは不十分です。可能であれば、**現在の運用形態を可能な限りにシミュレートできる環境で、同じサーバに対して複数のクライアントが同時に同期する条件に基づいて、これらの間隔の値を決める**必要があります。ただし、この点についてはあまり強調できません。
間隔をどの程度の長さに設定したらよいか理解したら、さまざまなオペレーティング・システムで間隔をどのように設定できるのか調べてみましょう。MobiLink サーバのすべてのプラットフォームを取り上げて説明します。

Windows でアイドルのスレッショルドを設定する方法

Windows のデフォルト設定は次のとおりです。

Ti = 2 時間
Nm = 5
Td = 5 秒

Windows は、システムワイドなレジストリ設定に基づいて、マシン全体に対する単一のチェック間隔を設定します。さらに不都合なのは、デフォルトでは接続の切断をチェックするまで2時間も待機することです。通常、これは敏速な活性チェックには不向きです。
重要：この設定は、どのような値を選択してもすべてのアプリケーションで共有され、同じマシン上で実行されている他のプログラムに悪影響を与えないようになっています。ASA Studio の将来のリリースでは、他のアプリケーションに影響を与えずに keep-alive の動作をコントロールできるようになる予定です。

Windows 95/98

以下に示すいくつかのレジストリ・エントリは、まだ登録されていない可能性があり、これらのいくつかを作成する必要があるかもしれません。修正が終わったら、新しい設定を有効にするためにマシンを再起動する必要があります。

警告：レジストリの修正は危険を伴います。レジストリの修正は自己責任で行ってください。

Ti のレジストリ・エントリは、ミリ秒単位の DWORD **値です。**

¥HKEY_LOCAL_MACHINE
¥System
¥CurrentControlSet
¥Services
¥VxD
¥MSTCP
¥KeepAliveTime

Nm のレジストリ・エントリは、文字列の値です。

¥HKEY_LOCAL_MACHINE
¥System
¥CurrentControlSet
¥Services
¥VxD
¥MSTCP
¥MaxDataRetries

Td のレジストリ・エントリは、ミリ秒単位の DWORD **値です。**

¥HKEY_LOCAL_MACHINE
¥System
¥CurrentControlSet
¥Services
¥VxD
¥MSTCP
¥KeepAliveInterval

Windows NT/2000

以下に示すいくつかのレジストリ・エントリは、まだ登録されていない可能性があり、これらのいくつかを作成する必要があるかもしれません。修正が終わったら、新しい設定を有効にするためにマシンを再起動する必要があります。

警告：レジストリの修正は危険を伴います。レジストリの修正は自己の責任で行ってください。

Ti のレジストリ・エントリは、ミリ秒単位の DWORD **値です。**

¥HKEY_LOCAL_MACHINE
¥System
¥CurrentControlSet
¥Services
¥Tcpip
¥Parameters
¥KeepAliveTime

Nm のレジストリ・エントリは、文字列の値です。

¥HKEY_LOCAL_MACHINE
¥System
¥CurrentControlSet

¥Services
¥Tcpip
¥Parameters
¥MaxDataRetries

Td のレジストリ・エントリは、ミリ秒単位の DWORD **値です。**

¥HKEY_LOCAL_MACHINE
¥System
¥CurrentControlSet
¥Services
¥Tcpip
¥Parameters
¥KeepAliveInterval

Sun Solaris で活性チェックの間隔を設定する方法

ndd ユーティリティを使用して、tcp_keepalive_interval (**Ti**)**をミリ秒単位で設定します。次のように指定すると、値がデフォルトの 2 時間に設定されます。**

ndd -set /dev/tcp tcp_keepalive_interval 7200000

現在の値は次のようにして調査できます。

ndd -get /dev/tcp tcp_keepalive_interval

この値を設定するには、適切なパーミッション(root **が適当**)**が必要です。詳細はシステム管理者にご相談ください。**

Linux で活性チェックの間隔を設定する方法

システム・パラメータを設定するには、sysctl 関数を使用するプログラムを記述するかユーティリティを探す必要があります。詳細は長くなるためここでは省略しますが、関連するパラメータは、ミリ秒単位の tcp_keepalive_time (**Ti**)**、および** tcp_keepalive_probes (**Nm**)**です。**

出典

- *The Microsoft Developers Network* (MSDN), October 2000.

- *Sybase Technical News, Volume 7, Number 8, August 1998* (from ISUG):
http://www.isug.com/Sybase_FAQ/ASE/Section10/5/Q10.5.8.html