STARS ユーザーズマニュアル

高エネルギー加速器研究機構 小菅 隆

1 はじめに

大規模な計測・制御システムの世界では、オブジェクト分散の技術を利用したシステム構築が日常のように行われていますが、ネットワークを利用したシステム構築に際しては、小規模システムにおいてもプログラムの分散化は有益です。たとえば、実際の機器を制御するドライバー部、GUI(Graphical User Interface) 部やメイン制御部などのように分散して開発を行う事ができれば、各部分のプログラムのサイズを縮小でき、保守時の見通しも良くなります。また、そうする事でプログラム開発を複数のスタッフにより共同で行ったり、プログラムの再利用が可能となります。

CORBA や DCOM によるオブジェクト分散の機能を利用する事は、小規模なシステムにおいても有効ですが、様々なオペレーティングシステム（以下 OS）や開発言語を利用する場合には、システム構築作業の大部分が CORBA、DCOM 導入のための作業となる事が予想されます。これらの要求や問題点を考慮した結果、我々は簡単にアプリケーション間のメッセージ配信を行える小規模システム向けの通信機構が必要であるという結論に達し、STARS(Simple Transmission and Retrieval System) の開発に至りました。

STARS におけるアプリケーション間通信は、全て TCP/IP Socket を利用したテキストベースのコマンドの送受により行われます。この事でシステムの扱いが非常に容易となり、それと同時に、開発言語や OS 選択の幅が広がります。また、デバッグを行う場合にも TELNET などのツールを利用する事が可能です。さらに、コアの部分となるプログラムは Perl を使って開発されており、STARS は様々なプラットフォーム上で動作可能です。

2 STARS の開発に至るまで

STARS の開発に着手する前、我々は様々な形態の制御プログラムを作成してきました。ここでは STARS の概念をより深く説明するために、STARS 開発に至るまでの経緯を示します。

2.1 ネットワーク対応型アプリケーション

開発の出発点はネットワーク対応型のアプリケーション作成でした。計算機ネットワークの普及に伴い、比較的容易にネットワーク対応型のプログラムを作成する事ができるようになり、リモートからの管理が可能となりました。また、ネットワーク機器の低価格化や Free の UNIX 系 OS が利用しやすくなった事は、ここでのネットワーク対応型アプリケーション [1] 開発を進める大きな力となりました。しかし、この段階ではアプリケーションプログラムの再利用や複数のスタッフでのアプリケーション開発に関する概念はまだありませんでした。

2.2 複数クライアントへの対応

ネットワーク対応型のプログラムが完成し、リモートでの管理が出来るようになると、「複数の場所から、機器の状態を監視したい」等の要求が発生してきました。ネットワーク対応型アプリケーションの第二段階として、我々は複数のクライアントプログラムに対応可能なシステム [2] を開発しました。このシステムにおいてプログラムは、機器の制御を行うサーバプログラムとユーザインターフェースであるクライアントプログラムに分割されていて、それぞれは TCP/IP Socket によって接続されます。サーバープログラムは複数のクライアントプログラムに対応可能であり、ユーザは複数の場所からシステムの監視が可能です。また、汎用的なユーザインターフェースプログラムを作成すれば、ある程度のアプリケーションプログラムの再利用が可能です。しかし、このタイプのシステムには、新しいハードウエアに対応するためには結局サーバプログラム自体を改造しなければならない事や、機能を増強してゆくとサーバープログラム自体が肥大化してしまうなどの問題がありました。

2.3 プログラムの分散化

これまでの問題点を解決するために開発を開始したのが STARS です。プログラムを分散化するための手法としてオブジェクト分散を用いたプログラム開発もひとつの手ですが、STARS では機能ごとにアプリケーショ

ンプログラムを作成する事でプログラムの分散化を行っています。また、各クライアントプログラム間の通信をTCP/IP Socke を用いたテキストベースのコマンド送受により行うなど、可能な限り単純化する事で汎用化を実現しています。

また、STARS の開発が進行中であった頃、比較的大規模なシステム向けの制御カーネルである COACK(Component Oriented Advanced Control Kernel)[3, 4] の開発も進行していました。COACK はもともと Windows ベースのシステムですが、COACK が持つ TCP/IP Socket インターフェースと STARS を利用して Non-Windows システムへのインターフェースの一つとして使おうとするアイデアが生まれ、COACK の Non-Windows システム用インターフェースとしての STARS 開発が進みました。このため STARS 自体、COACK と連携して動作する事が可能です。

3 STARS の構成

STARS はユーザが作成するクライアントプログラム、システムの中心的存在となりメッセージの配信を行う TAK（Transferring Agent Kernel）サーバおよび他のシステムとの接続を行う場合や、複数の TAK サーバを接続する場合に使用する Bridge から構成されます。(図 1 参照)TAK サーバは Perl によって記述されています。

各クライアントプログラムは TCP/IP Socket により TAK サーバに接続され、各クライアントプログラム及び TAK サーバ間の通信はテキストベースのメッセージの授受により行われます。各クライアントはユニークなターミナル名を持ち、それぞれのメッセージの送り先を指定する際に使用されます。また、それぞれは同一コンピュータ上にあってもネットワークを介したコンピュータ上に分散してもかまいません。各クライアントプログラムの起動、停止は STARS のシステム自体に特に影響を与えないので、システムの運転中に機能の追加や保守などを任意に行う事ができます。

3.1 TAK サーバ

STARS の中心的役割を果たすのが TAK サーバです。TAK サーバは接続されたクライアントプログラムから送られてくるメッセージを目的のクライアントプログラムへ配信します。また、イベント配信の機能等も有しています。実質的に STARS の機能そのものを実現するのが TAK サーバであるため、STARS サーバと呼ぶ事もあります。

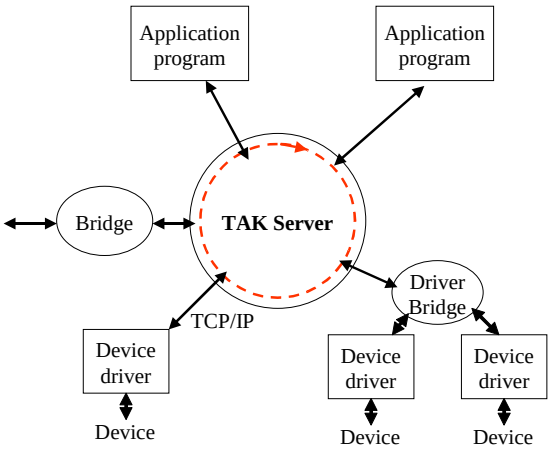

図 1: STARS の構成

TAK サーバは Perl によって記述されている為、あらかじめサーバが動作する PC には Perl（Version 5）がインストールされている必要があります。Perl は Linux、FreeBSD、Windows のような様々な OS で利用可能であり、TAK サーバも様々な OS 上で動作可能です。なお、TAK サーバ自体は一つのプログラムですが、後述する aliase の設定ファイル、セキュリティーチェック用の host 名一覧ファイル、キーワードファイル、ターミナル名+ホスト名チェック用ファイル等のファイルを使用します。これらのファイルは TAK サーバのライブラリ用のディレクトリにまとめて収納されます。

3.2 STARS クライアント

各クライアントプログラムは TCP/IP Socket を使用して TAK サーバに接続します。クライアントプログラムに関しては TCP/IP Socket を利用できれば OS や開発言語に特別な制限はありません。

各クライアントプログラムは STARS のシステムにおいてユニークなターミナル名を持たなければなりません。STARS クライアントは TAK サーバに接続する際、はじめに自分のターミナル名を自ら名乗る必要があり、接続が確立した後、メッセージの授受はこのターミナル名をもとに行われます。また、STARS には特別なターミナル名、“System” と “Debugger” が存在するので、クライアントプログラム作成においては注意が必要です。

3.3 Bridge

STARS では他のシステムへのインターフェース、或いは TAK サーバ同士を接続するクライアントプログラ

ムを “Bridge” と呼んでいます。STARS では各クライアントへ宛先を指定してメッセージを送信しますが (詳細は 4.1 で述べます)、宛先において “.”(ピリオド) は特別な意味を持ち、宛先に “.” を使用すると、以降の文字列に関係なく常に “.” より前の文字列で示されるクライアントにメーッセージが送出されます。STARS ではこの事を利用し “Bridge” の機能を実現しています。

4 STARS の機能

STARS は基本的にメッセージの配信を行う非常にシンプルなシステムですが、いくつかの機能の追加及びメッセージ配信に対して幾つかの決まりを設けて、様々なシステムにも対応可能にしています。ここでは STARS の持つ機能について紹介します。

4.1 メッセージの配信

TAK サーバに接続された全ての STARS クライアントプログラムは、データ転送に関して同等な権利を有します。もしアクセス制限などの機能が必要な場合は、各クライアントで独自に用意する必要があります。

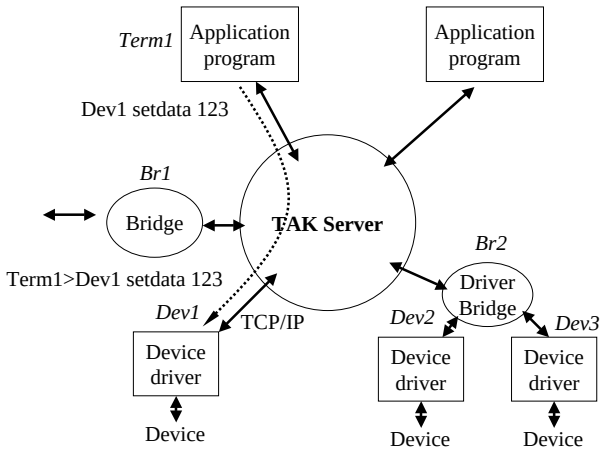

図 2: メッセージ配信

授受されるデータは全てキャラクターベースでデリミタは LF（0x0A）です。実際のデータ転送の方法としては、たとえば Term1 と名づけられたクライアントプログラムから Dev1 と言う名のクライアントプログラムにデータを転送する場合を例にすると図 2 に示す通りになります。はじめに Term1 は “Dev1 setdata 123” のようなコマンドを TAK サーバに送出します。ここで “Dev1” は宛先であり、“setdata 123” はメッセージです。その後、Term1 よりメッセージを受け取った TAK サーバは Dev1 に対して “Term1>Dev1 setdata 123” のような送り主を示す文字列を行頭に付加したメッセージを送信します。

4.2 メッセージに関する取り決め

TAK サーバは、受け取ったメッセージを単に指定された宛て先に送信します。STARS ではこれらのメッセージに、ある程度取り決めを設けて効率的な動作を行うようにしています。実際には “@”（アットマーク）で始まるメッセージは何かのコマンドに対する回答 (Reply message) である事を示し、“_”（アンダースコア）で始まるメッセージは何かのイベント (Event message) が発生した事を示します。それ以外のメッセージはコマンド (Command message) です。

STARS クライアントを作成する際は、作成するクライアントプログラムが他のクライアントに通常の Command message を送出した場合、必ず “@” で始まる Reply message を受け取る事を期待するようにします。また、逆に作成したクライアントプロがラムが通常の Command message を受け取った場合は必ず “@” から始まる Reply message を Command message の送り主に返送します。この際、たとえば Command message が “hello” であったとすると Reply message は “@hello Nice to meet you.” のように “@”+“受け取った Command message” から始まるようにする事が推奨されます。なお、クライアントプログラムは “@” から始まるメッセージを受け取った場合、そのメッセージに対する返事を返してはなりません。もし、知らない Reply message を受け取った場合は単に無視するようにします。

“_” で始まる Event message についても Reply message 同様、決して返事を返してはなりません。そして、理解可能な Event message についてはそれなりに処理を行い、それ以外の Event message に関しては無視をするようにします。

TAK サーバは、受け取った Command message の宛先が存在しない場合、エラーを示す Reply message を返しますが、Event message 及び Reply message の場合エラーメッセージは返しません。現在のバージョンの TAK サーバにおける、宛先不明の Reply message と Event message に対する扱いは同じで単に破棄されるだけです。

4.3 イベント配信機能

前述の通り STARS において “_” から始まるメッセージはイベントメッセージですが、TAK サーバはイベント配信先が不確定の場合や配信先が複数の場合に有用

なイベント配信機能を有しています。まず、イベントを発信するクライアントは宛先を System にしてイベントメッセージを送出します。その後 TAK サーバは、あらかじめ flgon コマンドにより配信の依頼を行っているクライアントに対してイベントメッセージを送出します。

4.4 alias 機能

STARS のクライアントはそれぞれユニークなターミナル名を有しますが、TAK サーバに別名を登録する事ができます。この機能を利用すると、クライアントの構成が変わった場合などのプログラムの書き換えを最小限に抑える事ができます。なお、別名の情報は TAK サーバのライブラリ用ディレクトリ内の “aliases.cfg” ファイルに記録されています。別名の登録等はこのファイルをテキストエディタ等で編集する事により行います。ファイル書き換え後は STARS クライアントから TAK サーバに対して “loadaliases” コマンドを送出する必要があります。

4.5 システムコマンド

TAK サーバはイベント配信要求のためのコマンドを含め幾つかのコマンドを有しています。TAK サーバは “System” という名前を有しているので、クライアントプログラムからは宛先を “System” にして Command message を送出すれば OK です。以下はシステムコマンドの一覧です。

- flgon: TAK サーバにイベント配信の要求を行います。以下はクライアント “term2” のイベント受取を要求した場合の例です。

```
System flgon term2
System>term1 @flgon Node term2 has
 been registered.
```

- flgoff: TAK サーバに要求したイベント配信を解除します。以下はクライアント “term2” のイベント受取を取り消した場合の例です。

```
System flgoff term2
System>term1 @flgoff Node term2 has
 been removed.
```

- help: システムコマンドの一覧をスペース区切りで返します。
- hello: STARS の動作をチェックするために “hello” コマンドが用意されています。単純に以下のような答えを返してくるだけです。

```
System hello
System>term1 @hello Nice to meet you.
```

- loadaliases: TAK サーバはターミナル名のエイリアシング機能を備えていて、システム構成が変わった場合にも対応できるよう考慮されています。また、エイリアスの変更時にシステムを停止する必要はありません。loadaliases コマンドは変更されたエイリアスファイルをシステムに再び読み込みます。
- listaliases: エイリアシングの状況をスペース区切りで返します。
- listnodes: TAK サーバに接続されている全てのノード名をスペース区切りで返します。
- gettime: TAK サーバが動作しているコンピュータの現在の時刻を返します。
- getversion: TAK サーバのバージョンを返します。
- quit: TAK サーバとの接続を解除します。
- disconnect: 指定したクライアントと TAK サーバとの接続を解除します。

4.6 Debugger

TAK サーバはデバッグの為の機能を有していて、“Debugger” という特別なターミナル名で接続すると、TAK サーバが各クライアントに送出するメッセージを全て受け取る事ができます。なお詳細については 8.1 で述べます。

5 接続時のセキュリティ機能

STARS はクライアントの接続に対して簡単なセキュリティー機能を有しています。実際には、ホスト名によるチェック、キーワードによるチェック、ターミナル名及びホスト名によるチェックです。

5.1 ホスト名によるチェック

TAK サーバは STARS クライアントから接続の要求があると、はじめにそのクライアントプログラムが動作するコンピュータが、接続可能ホストのリスト内に存在するかどうかをチェックします。接続可能ホストのリストは TAK サーバライブラリ用ディレクトリ内の “allow.cfg” ファイルです。ホスト名の登録はこのファイルをテキストエディタなどで編集する事により行います。なお、各ホスト名は改行で区切られます。書き換えを行った場合、TAK サーバの再起動などの必要はなく、

書き換え後はその設定がすぐに反映されるようになります。

ホスト名が接続可能ホストのリスト内に存在しない場合、TAK サーバへの接続は拒否されます。以下はテスト的に telnet を使用して client-host から、server-host 上で動作する TAK サーバに接続を試みた場合で、client-host は接続可能リストに登録されていない為に接続が拒否された例です。

```
% telnet server-host 6057
Trying 192.168.0.10...
Connected to server-host.
Escape character is '^]'.
Bad host. client-host
Connection closed by foreign host.
```

上記の例で “Bad host. ” に続く部分が TAK サーバが認識しているクライアントのホスト名 (場合によっては IP アドレス) です。システムの環境等により、認識するクライアントのホスト名が変化する場合があるので、クライアントプログラムがうまく接続できない場合は、telnet 等を使用して TAK サーバの認識している実際のホスト名を確認します。そして、この時表示されるホスト名を “allow.cfg” ファイルに登録すれば OK です。

5.2 キーワードによるチェック

STARS の接続時における第二段階のセキュリティーチェックはキーワードによるチェックであり、TAK サーバには予め STARS クライアント毎にキーワードのリストが登録されている必要があります。キーワードのリストは実際にはテキストファイルで “1 番目のキーワード”[改行]“2 番目のキーワード” ・・・ のように改行文字 (Unix の場合は LF、Windows の場合は CR+LF) で区切られています。以下は 4 つのキーワードが登録されているファイルの例です。

```
Keyword1
Keyword2
Keyword3
Keyword4
```

このキーワードリストのファイルはクライアントプログラムごとに “ターミナル名”+.key(例: term1.key) のような名前で TAK サーバのライブラリ用ディレクトリに保存されます。なお、各クライアント毎に設定可能なキーワードの数は 10,000 個です。また、ファイル登録の際、TAK サーバの再起動は必要ありません。

実際のキーワードチェック (図 3 参照) は次の手順で行われます。まずはじめにホスト名によるチェックが成功すると、TAK サーバは 0 から 9,999 までの数をランダムに発生し、接続を試みる STARS クライアントに送出します。次にクライアントはこの数をもとに “自分のターミナル名 要求するキーワード” のメッセージを TAK サーバに送信します。この時の要求されるキーワードは、

(TAK サーバからの数をキーワードの数で割った余り+1) 番目のキーワード

です。上記のキーワードリストを例にすると、TAK サーバからの数が “0” の場合は “Keyword1” を “10” の場合は “Keyword3” を自分のターミナル名と共に TAK サーバに送信すれば OK です。以下は telnet を使用してテスト的に接続を試みた例です。この場合 TAK サーバはクライアントと同一のコンピュータ上で動作しています。

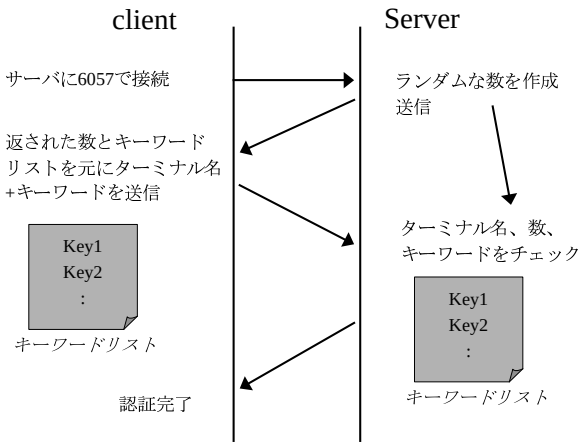

図 3: キーワードチェックの手順

```
[接続に成功した例]
%telnet localhost 6057
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
3392
term1 Keyword1
System>term1 Ok:

[不正なキーワードで接続に失敗]
%telnet localhost 6057
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
4767
term1 Keyword1
System> Er: Bad node name or key
Connection closed by foreign host.
```

通常クライアントプログラムを作成する場合、これらのキーワードリストはソースファイルに埋め込まれたり、別途キーワードファイルとして登録する事が予想されますが、複数のユーザが使用するコンピュータではキーワードの漏洩に対して注意が必要です。そのようなシステムでは TAK サーバのライブラリ用ディレクトリに保存されているキーワードリストファイルも含め、漏

洩の危険が無いようにパーミッションの管理を正しく行う事が重要です。

5.3 ターミナル名+ホスト名によるチェック

STARS の接続に関する第三段階のセキュリティーチェックはターミナル名とクライアントの動作するコンピュータのホスト名 (場合によっては IP アドレス) のマッチングによるチェックです。前述のホスト名によるチェックとキーワードによるチェックは必須ですが、ターミナル名及びホスト名によるチェックの導入はシステムの運用方針により決定する事ができます。

TAK サーバはライブラリ用ディレクトリ内に “接続を要求してきたクライアントのターミナル名”+.allow (例、ターミナル名 term1 の場合: term1.allow) というマッチングチェック用ファイルがあった場合、ターミナル名とホスト名のマッチングのチェックを行います。たとえばホスト名が “client-host” というコンピュータ上で動作する “term1” というターミナル名のクライアントプログラムが接続を試みてきた場合で、TAK サーバのライブラリ用ディレクトリ内に “term1.allow” というファイルがあった場合は、このファイルの中に “client-host” という名前があるかどうかのチェックが行われます。なお、マッチングチェック用ファイルは接続可能ホストのリストファイルと同様、改行で区切られたテキストファイルで、テキストエディタ等で編集や登録を行う事が出来ます。

5.4 小規模クライアントプログラムへの配慮

これまで述べたように、キーワードによるチェックを設ける事で、STARS では接続に関してある程度のセキュリティーを確保しています。しかし、これらの手順は、クライアントプログラムがシングルユーザ環境でのみ動作する場合や、小規模な機器上で動作する場合には不必要である事があります。5.3 の “ターミナル名及びホスト名によるチェック” は、これらのようなクライアントプログラムに対して大変有効です。

たとえば、組み込み用の機器で動作する非常に簡単なクライアントプログラムを例に取ります。ここではクライアントのターミナル名が “simpleclient”、ホスト名が “client-host” でシングルユーザのシステムであると仮定します。そして、ここで接続の手順を最も簡単にする方法は、TAK サーバのライブラリ用ディレクトリに置かれている “simpleclient.key” の中に含まれているキーワードを “Keyword1” のように 1 つだけにしてしまう事です。この事により “simpleclient” が接続の際に TAK サーバへ送信するキーワードは、TAK サーバが送信して来るランダムな数に関係なく常に “Keyword1” で良くなります。つまり、ここでは “simpleclient” は TAK サーバにからのランダムな数字を単に空読みして、常に “simpleclient Keyword1” の文字列を TAK サーバに送れば良い事になります。

この方法によりクライアントプログラムの接続に関する部分が非常に簡単になりますが、たった一つのキーワードが漏洩した時点で、誰でも STARS に接続できてしまうので大変危険です。ここで、大きな役割を果たすのが “ターミナル名及びホスト名によるチェック” 機構です。この例の場合は TAK サーバのライブラリ用ディレクトリに “simpleclient.allow” ファイルを作成し、その中に “client-host” のエントリーを作っておけば OK です。その事で “simplecliet” というターミナル名をもつクライアントプログラムは “client-host” 以外からは接続不可能となり、たとえキーワードが一つだけであっても、この例の “client-host” はシングルユーザのシステムであるため、接続に対するセキュリティ確保が可能となります。

6 システムを起動するまで

STARS においてクライアントプログラムに対する開発言語の制約はありませんが、サーバを動かす場合には必ず Perl(Perl5) が必要です。最近では UNIX 系の OS の場合、元々Perl がインストールされている場合が多いので、ターミナル上で “%perl -v” と入力してみるのも一つの手です。もし、Perl がインストールされていれば以下の様に表示される筈です。

```
%perl -v

This is perl, version 5.005_03
 built for i386-freebsd

Copyright 1987-1999, Larry Wall

Perl may be copied only under the terms
 of either the Artistic License or the
GNU General Public License, which may be
 found in the Perl 5.0 source kit.
```

なお、Windows においては Active Perl[1] において動作の確認を行っています。

6.1 サーバのインストール

サーバのインストールは配布パッケージ内の “takaserv” を STARS サーバ用のディレクトリにコピーし、TAK サーバのライブラリ用ディレクトリ “takaserv-lib” を作成、必要な設定ファイルを作成す

[1] http://www.activestate.com

る事により完了します。なお、詳細は STARS のパッケージ内に含まれるドキュメントファイルに記されています。

6.2 設定ファイルの準備

TAK サーバのライブラリ用ディレクトリ “takaserv-lib” 配下には、別名情報ファイル、接続可能ホストのリスト、クライアント毎のキーワードリスト、マッチングチェック用ファイルなどのファイルを用意する必要があります。

6.2.1 別名情報ファイル

別名情報ファイルは必要に応じて作成します。ファイル名は “aliases.cfg” であり、テキストエディタ等で作成可能です。別名情報ファイルの内容を表示すると以下の様になります。(1 行目は表示の為のコマンド)

```
%cat aliases.cfg
Device1 Alias1
```

これは “Device1” の別名を “Alias1” と設定した例です。実際の名前と別名はスペースで区切って記述します。

6.2.2 接続可能ホストのリスト

接続可能ホストのリストは必須です。ファイル名は “allow.cfg” であり、接続可能なホスト名を改行で区切って列挙しておきます。以下は同一コンピュータ内からのアクセスを許可した場合の例です。(1 行目は表示の為のコマンド)

```
%cat allow.cfg
localhost
127.0.0.1
```

6.2.3 クライアント毎のキーワードリスト

クライアント毎のキーワードリストファイルは、接続するクライアント毎のものを必ず用意しなければなりません。ファイル名は “クライアントのターミナル名”+“.key” となります。以下はターミナル名 “term1” を持ったクライアントのキーワードファイルの例で、“hello” というたった一つだけのキーワードを設定しています。(1 行目は表示の為のコマンド)

```
%cat term1.key
kek
```

6.2.4 マッチングチェック用ファイル

マッチングチェック用ファイルは必要に応じてクライアント毎のものを用意します。以下はターミナル名 “term1” を持ったクライアントのマッチングチェック用ファイルで同一コンピュータ内からのアクセスだけを許可した場合の例です。(1 行目は表示の為のコマンド)

```
%cat term1.allow
localhost
127.0.0.1
```

6.3 TAK サーバの起動

サーバのインストールが完了し設定ファイル等の準備が出来れば、STARS の起動が可能です。起動に際しては TAK サーバのあるディレクトリに移動し、ターミナルから (Windows 2000、XP の場合はコマンドプロンプト)“perl stars” と入力すれば OK です。なお、UNIX 系の OS では環境によって、“./stars” と入力する事で起動可能 (実行のパーミッションが必要) です。更に UNIX 系の OS では、

```
%perl stars &
```

あるいは、

```
%./stars &
```

と入力する事で、バックグラウンドでの実行が可能となります。

6.4 クライアントの接続テスト

ここではサーバの起動が成功した後のクライアント接続テストについて紹介します。はじめに TAK サーバのライブラリ用ディレクトリには 6.2 での例題ファイルと同じ物を用意します。次に telnet を使用して接続のテストを行います。以下は UNIX 系 OS での例です。

```
%telnet localhost 6057
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
7592
term1 kek
System>term1 Ok:
System hello
System>term1 @hello Nice to meet you.
quit
System>term1 @quit
Connection closed by foreign host.
```

この例では接続に成功した後、“hello” コマンドを TAK サーバに送りその後 “quit” コマンドで接続を切断しています。

7 クライアントプログラムの例

ここでは実際のクライアントプログラムの例を示します。以下は 6.2 での設定ファイルが用意されてた環境に対応したクライアントプログラムの抜粋で、他のクライアントにメッセージを送るところまでを示しています。

```
1 #! /usr/bin/perl
2 use IO::Socket;
3 $host='localhost';
4 $port=6057;
5 $|=1;
```

```
## Open TCP/IP socket
my $server =
  new IO::Socket::INET(PeerAddr=>$host,
        PeerPort=>$port, Proto=>'tcp')
or die "Socket: $!\n";
select($server);$|=1;select(STDOUT);
## Get random number from TAK server.
$buf=<$server>;
## Send terminal name and keyword.
print $server "term1 kek\n";
## Expect connection will be accepted.
$buf=<$server>;chop($buf);s/\r//;
unless($buf =~ /Ok:$/){
  close($server);
  die "$buf\n";
}
print "$buf\n";
print "Please enter destination: ";
$ds=<STDIN>;
chomp($ds);
print "Please enter commands: ";
$cm=<STDIN>;
chomp($cm);
## Send message
print $server "$ds $cm $lp\n";
```

また、以下は 8.2 の “STARS Perl モジュール” を使用したクライアントの例です。このクライアントは他のクライアントプログラムからの “hello” コマンド及び “help” コマンドに反応します。

```
#! /usr/bin/perl
use strict;
use Getopt::Long;
use stars;
## ToDo: Set parameters here.
$::NodeName   = 'testhello';
$::Server     = 'localhost';
## ToDo: You can set option switchs.
## See help 'Getopt::Long'.
GetOptions(
'node=s' => \$::NodeName,
'h'      => \&usage,
) or die "Bad switch.\n";
if($_ = shift(@ARGV)){$::Server = $_;}
## Open Stars server.
## $::tak is a Stars object.
$::tak=stars->new($::NodeName,$::Server)
or die "Could not connect Stars server";
$::tak->addcallback(\&handler);
stars->Mainloop();
exit(1);
# Print usage. -------------------------
sub usage{
## Todo: Please modify help message
##   for "-h" option.
 print "Usage: testhello [-h]".
  " [-node MyNodeName] [StarsServer]\n";
  exit(0);
}
# Command handler from Stars server ----
sub handler{
##ToDo:Please modify handler sub routine.
## (The handler sub routine will be
## called when client receives a message
##from a Stars server.)
  my ($from, $to, $mess) = @_;
  if($mess eq 'hello'){
    $::tak->Send
     ("\@hello nice to meet you.",$from);
  }elsif($mess eq 'help'){
    $::tak->Send("\@help hello", $from);
  }elsif($mess =~ /^[_@]/){
    return;
  }else{
    $::tak->Send
 ("\@$mess Er: Bad command or parameter",
     "$from");
  }
}
```

8　ライブラリ及びユーティリティ

STARS をより使いやすくするために、現在様々な機能やライブラリ、ユーティリティが開発されています。ここではこれらの内、一般的に利用可能であると思われる機能について紹介します。

8.1　Debugger の利用

クライアントプログラムを新しく作成した場合や、STARS の動作テストを行う場合、Debugger 機能は非常に有効です。STARS において “Debugger” というターミナル名を持つクライアントは特別です。クライアントが “Debugger” と言う名前で TAK サーバに接続すると、TAK サーバは Debugger に対し、各クライアントに送信するすべてのメッセージのコピーを送出してきます。この機能を使えば各クライアントプログラム間の通信をモニターする事が可能となり、クライアントプログラムのデバッグなどに大いに役立てる事ができます。以下は telnet を用いて “Debugger” 機能を使い、モニタリングを行っている例です。

```
%telnet localhost 6057
Trying 127.0.0.1...
```

```
Connected to localhost.kek.jp.
Escape character is '^]'.
6296
Debugger KEK
System>Debugger Ok:
System>Debugger Ok:
watcher>keyterm07 gettemp
keyterm07>watcher @gettemp 32.5
watcher>gatekeeper01 isClose
gatekeeper01>watcher @isClose 1
watcher>gatekeeper02 isClose
gatekeeper02>watcher @isClose 1
```

8.2 STARS Perl モジュール

STARS Perl モジュールを利用すると Perl での STARS クライアントの開発効率が向上します。STARS Perl モジュールを使用するためには、はじめに “stars.pm” をモジュールのサーチパスにコピーするか、クライアントプログラムのカレントディレクトリのコピーします。あるいは、“stars.pm” を適当なディレクトリにコピーした上でクライアントプログラム起動の際に “-I *directory*” オプションを指定したり、プログラム内で “use lib;” を使用してもかまいません。

次にクライアントプログラムと同一のディレクトリにキーワードファイル (TAK サーバのライブラリ用ディレクトリ内と同じもの) をコピー、クライアントプログラム内に “use stars;” の一文を付け加える事で STARS Perl モジュールが利用可能となります。

以下は実際に STARS Perl モジュールを使用したクライアントプログラムの抜粋です。

```
1 use stars;
2 $svr = stars->new('term1') or die;
3 print $svr->act('System hello')."\n";
```

1 行目では STARS Perl モジュールの利用を宣言、2 行目で STARS オブジェクトを作成、TAK サーバに接続を行っています。この時、2 つめ及び 3 つめの引数は省略可能で、それぞれ “localhost”、“ターミナル名”+“.key” がデフォルトとなります。ここでキーワードチェックに関する手順は STARS Perl モジュールが自動的に行ってくれます。3 行目では TAK サーバに hello コマンドを送出した後、Reply message を受信、表示を行っています。なお、STARS Perl モジュール使用法の詳細については、付属のドキュメントファイルに記されています。

8.3 STARS ActiveX Control

Windows 用の STARS ActiveX Control を Visual Basic 等で利用すると Windows での STARS クライアントプログラムの作成が容易になります。STARS ActiveX Control のインストールは単にインストーラ (パッケージ内の setup.exe) を起動するだけで OK です。インストールが終了した後は Visual Basic なら、プログラム作成の際に [プロジェクト] → [コンポーネント] で “StarsInterface” にチェックをした後、Form 上に StarsInterface のアイコンを配置します。STARS Perl モジュール同様キーワードファイルをクライアントプログラムと同じディレクトリ内に用意 (デリミタに注意: CR+LF) し、プログラム内で “StarsControl1.Connect” のように Connect メソッドを呼び出すだけで利用可能です。

8.4 キーワード発生プログラム

STARS においてクライアント毎のキーワードファイルは単純なテキストファイルです。そのためキーワードファイルは適当なテキストエディタで編集を行う事が出来ますが、数十や数百のキーワードを作るとなると容易な事ではありません。STARS に付属のキーワード発生プログラム (createkey) を使用すると、ランダムなキーワードを任意の数含んだキーワードファイルを作成する事ができます。但し、本プログラムは Perl で記述されているため、予め Perl が実行可能でなければなりません。

以下はキーワード発生プログラム “createkey” をカレントディレクトリにコピーして “perl createkey キーワードの数” とした場合の例です。

```
%perl createkey 5
Wyghv'U2GZ
Et3|aXo;crAv
2-NJ6EVs>UnUx44-i
:&nPjW2D%~ja=\}M
W;ln?h/gJfn
```

そしてここで実際のキーワードファイルに出力するには、

```
%perl createkey 100 > term1.key
```

のようにファイルにリダイレクトすれば OK です。

8.5 Perl クライアント作成ウィザード

Perl でクライアントプログラムを作成する場合用に Perl クライアント作成ウィザードが用意されています。利用にあたっては予め Perl が実行可能となっていなければなりません。実際の利用は、Perl クライアント作成ウィザードのディレクトリに移動してから “perl newclient.pl” と入力します。以下は FreeBSD において term1 というターミナル名のクライアントプログラムを作成する場合の例です。

```
%perl newclient.pl
Make a new Stars client program in Perl.
Please enter client name.
    (null = cancel) >term1
Please enter stars server.
    (null = cancel) >localhost
Please enter directory for term1.
    (null = cancel) >/home/kosuge/term1
Create /home/kosuge/term1/stars.pm.
Create /home/kosuge/term1/term1.
Create key > /home/kosuge/term1/term1.key.
Done.
Please send "/home/kosuge/term1/term1.key"
 to localhost
 with ftp (asc mode).
Hit Enter key.
```

この時キーワードファイル (この場合は term1.key) が同時に作成されますので、このファイルを TAK サーバのライブラリ用ディレクトリに ftp 等を使って転送します。Windows と UNIX 系のシステムではテキストファイルのデリミタが違います、ftp 使用時には ascii モードを使用するとよいでしょう。

以上のように Perl 用クライアント作成ウィザードを使用すると “hello” 及び “help” コマンドのみを備えたクライアントプログラムが作成されますので、後は適宜ソースファイル内の指示に従ってプログラムを書き換えれば OK です。

8.6 Visual Basic クライアント作成ウィザード

Visual Basic によりクライアントプログラムを作成するために、Visual Basic クライアント作成ウィザードが用意されています。利用にあたっては予め Perl が実行可能となっていなければなりません。また、予め STARS ActiveX Control もインストールされていなければなりません。実際の利用は Visual Basic クライアント作成ウィザードのディレクトリに移動してから “perl newclient.pl” と入力します。以下は Windows2000 において term1 というターミナル名のクライアントプログラムを作成する場合の例です。

```
Make a new Stars client program in VB.
Please enter ClientName.
     (null = cancel) >term1
Please enter StarsServer.
     (null = cancel) >localhost
Please enter directory for term1.
     (null = cancel) >d:\term1
Create d:\term1/Form1.frm.
Create d:\term1/term1.vbp.
Create d:\term1/term1.vbw.
Create key > d:\term1/term1.key.
Done.
Please send "d:\term1/term1.key"
 to localhost
 with ftp (asc mode).
Hit Enter key.
```

この時キーワードファイル (この場合は term1.key) が同時に作成されますので、Perl 用クライアント作成ウィザードと同様に、このファイルを TAK サーバのライブラリ用ディレクトリに ftp 等を使って転送します。また、TAK サーバが UNIX 系のシステムで動作している場合はテキストファイルのデリミタが違います、ftp 使用時には ascii モードを使用するとよいでしょう。

9 STARS の今後

これまで述べた通り STARS は小規模なシステム向けに設計された非常に単純なシステムです。また、様々なプラットフォーム上で動作するなど、便利な点が多いのも特徴です。現在 STARS は放射光研究施設のビームラインインターロック集中管理システム、実験ホール入退室管理システム、ビームライン制御の一部等に導入され、安定に動作しています。

STARS は現在も開発が進行しており、様々な開発項目の検討がなされています。以下は現在進行中の開発項目です。

- タイムアウト処理: 現在、Command message を送出し Reply message が届かなかった場合のタイムアウト処理は特に行っていないため、クライアントプログラム側で処理するようになっています。しかし、この Command message の処理が Command message を送出したクライアントプログラムの予想していたものより長い場合には、タイムアウト処理を行った後に Reply message が帰ってきてしまうという不具合が発生します。STARS の次期バージョンではサーバ自体にタイムアウト処理の機能をインプリメントします。

- マルチスレッド化: 今のところ TAK サーバはシングルスレッドで動作しており、接続するクライアントプログラムが増加しメッセージ転送が増えると、そのまま処理速度に影響を受けてしまう。この対策としてマルチスレッド化などの検討を行っています。

- バイナリー転送: STARS の大きな特徴としてテキストーベースでのメッセージの送受により、通信が簡単である事が挙げられますが、バイナリーデータの授受に関する要求も多いのも現実です。今後この要求を満たすためにバイナリーデータの転送に関する機能を設ける予定です。

- XML メッセージ転送: XML の普及に伴い STARS で XML メッセージを扱う事が検討されています。そのため現在 “XML”、“@XML”、“_XML” の Command

message、Reply message、Event message は予約語となっています。

参考文献

[1] 小菅隆, 久積啓一, ワンボードマイコン及び FreeBSD マシンを使用した遠隔電圧測定システム, 平成 9 年度核融合科学研究所技術研究会, 1997

[2] Takashi Kosuge, and Yoshinori Uchida, Low-Cost Beamline Control System, PCaPAC'99, KEK, 1999

[3] I.ABE, et al., RECENT STATUS OF COACK AND IT'S FUNCTIONS, PCaPAC2000 DESY, 2000

[4] Takashi Kosuge, et al., COACK Application for the Beamline Interlock System at the Photon Factory, PCaPAC 2000, DESY, 2000