Non-Windows システム用 COACK インターフェース

高エネルギー加速器研究機構 技術部　○小菅 隆
&　COACK 開発チーム

1　はじめに

実験装置用制御システムとして共同で開発されてきた COACK(Component Oriented Advanced Control Kernel)[1, 2, 3, 4] は非常に強力かつ柔軟なシステムであり、様々な機能を有している。COACK ではサーバ内に仮想的な実験装置を構築することが可能であり、COACK サーバに接続するクライアントアプリケーションはサーバ上に構築された仮想的な実験装置にアクセスするだけでよいので、プログラム開発者は実験装置の一部が未完成であってもクライアントプログラムの作成に取り掛かる事が可能である。

COACK は Windows をベースとしたシステムであり、クライアントとサーバ間では通常 DCOM を使った通信が行われているが、他のオペレーティングシステム (以下 OS) の機器をを接続することを考慮し TCP/IP Socket のインターフェースを有している。TCP/IP Socket インターフェースを利用したクライアントとサーバとのコマンドのやり取りは XML を使用して行われるため、DCOM で接続されたクライアントプログラムと同様なことが行える。しかし、逆の言い方をすれば TCP/IP Socket インターフェースを利用するクライアントプログラムは XML を扱わなければならないため、プログラム開発に対する労力が大きくなる事が予想される。これまで、この問題を解決するための一つの方法として STARS(Simple Transmission and Retrieval System)[5, 7] の利用と COACK Bridge の開発を行ってきたが、今回 COACK Bridge について機能向上を目指した改造を行った。ここでは、COACK お STARS 及び COACK Bridge を使った Non-Windows システム用インターフェースの詳細と COACK Bridge の機能向上等について報告する。

2　COACK の TCP/IP Socket インターフェース

TCP/IP Socket インターフェースが使用するポート番号は COACK サーバ管理者が必要に応じて設定することが出来る。ポート番号の設定は COACK の Registry Tool を使って行う。

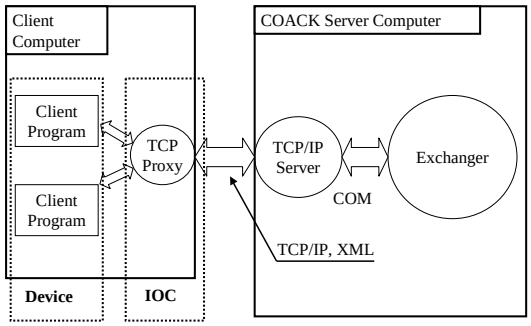

図 1: COACK の TCP/IP インターフェース

2.1　コネクションとセッション

COACK において TCP/IP Socket のポートを Open し、接続を行うことをコネクションという。コネクションは IOC(I/O Controller) 毎に行い、クライアント側ではプロトコルアダプタ (Proxy) を作成する (図 1 参照)。実際の機器などを直接制御するクライアントプログラムは Proxy を通してサーバへの接続 (セッション) を行う。COACK の TCP/IP サーバはシングルスレッドサーバであるため、同一のサーバで扱うクライアントはなるべく少なくする必要がある。実際の通信は DCOM サーバと同様に IOC 単位に行われるので、TCP/IP サーバにおいても IOC 単位にサーバを立ち上げ、TCP/IP コネクションを共有する方式がとられている。もし、複数の TCP/IP コネクションを使用する必要がある場合は Kernel Manager により複数の TCPServer を起動する。この時、各 TCPServer で使用される Socket のポート番号は Registry Tool で設定したポート番号をもとに、1 ずつインクリメントされる。

2.2　COACK のコマンドと *XML*

COACK のサーバとクライアントの間で交換されるコマンドは、ネストした繰り返し構造をもっている。そのため CVS 形式等で単純に表現することは難しく、構造を表現するための何らかの手立てが必要となる。COACK の TCP/IP Socket インターフェースではデータ形式として XML のサブセットを定義し、利用することでコマンドを表現するようにしている。

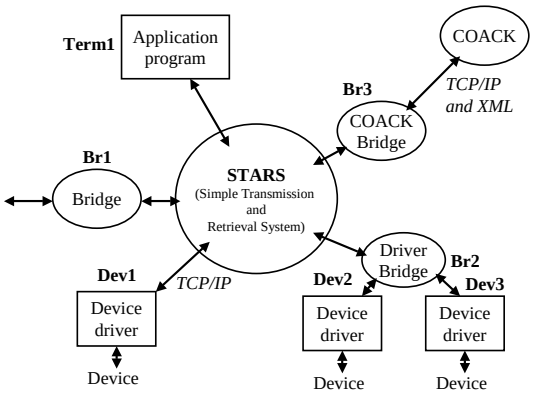

図 2: STARS の概要

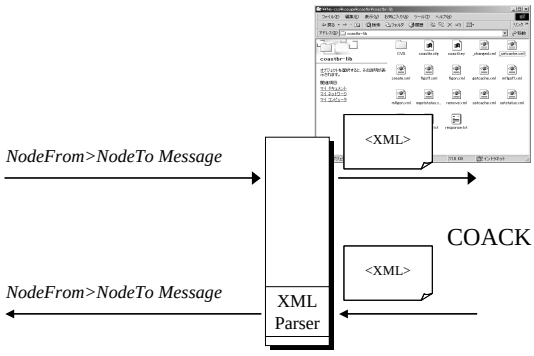

図 3: COACK Bridge

3　STARS との接続

STARS は小規模な制御システム向けに開発されたサーバ・クライアント型のシステムで、複数のクライアントが TCP/IP Socket 利用してサーバに接続することが出来る。STARS サーバは各クライアントから送られてくるメッセージを、そのメッセージ内に含まれる配信先の情報に応じて転送を行う。それぞれの送受されるメッセージはテキストベースでありシステムとしては非常にシンプルである (図 2 参照)。また、サーバ部分は Perl を使用して作成されており、様々な OS 上で動作することが可能である。さらに、TCP/IP Socket 及びテキストベースでのデータ送受を行うだけである為、様々な開発言語を利用することが出来る。

COACK の Non-Windows システム用インターフェースとして STARS を使えば、より多くの OS や開発言語から COACK を簡単に利用できるようになる。なお、もともと COACK とのインターフェースを考慮して開発された STARS は、COACK との親和性が非常に高い。

3.1　COACK Bridge

COACK と STARS を接続するためのプログラムが COACK Bridge である。開発言語としては Perl が使用されている。このことにより COACK Bridge は STARS 同様様々な OS 上で動作可能であり、COACK サーバが動作するパーソナルコンピュータ (以下 PC) 上で動作させることも、COACK サーバとは別の UNIX マシン等で動作させることもできる。また、COACK Bridge は STARS 側から見ると単なる Bridge として見えるだけであり、COACK 側からは IOC に見える。

STARS 側に接続されたクライアントプログラムが CAOCK Bridge に”create”命令を送ると COACK Bridge は COACK サーバに対し、セッション開設作業を行う。クライアントプログラムはセッションの開設後、必要に応じて”flgon”、 ”flgoff”、 ”getcache”、 ”setcache”などのコマンドを送ればよい。COACK 側からはこれらセッションの開設されたクライアントプログラムは、単に Proxy を通して XML データのやり取りが可能なクライアントプログラムに見える。なお、セッションが開設されていない STARS のクライアントについて COACK は一切認識しないので、COACK に不必要なクライアントプログラムを COACK から切り離すことができる。

3.2　メッセージの変換

図 3 に示す通り COACK Bridge は STARS メッセージと COACK 用 XML の変換を行う。まず、STARS からのメッセージを受け取ると COACK Bridge はメッセージ内に含まれるコマンドに応じた XML のテンプレートを呼び出す。XML のテンプレートについては COACK Bridge 起動時に予め読み込んでおくことで、動作速度の向上を図っている。次に STARS からのメッセージを分解しそれぞれの部分をテンプレートにはめ込んでゆく。たとえば受け取ったメッセージが”coast.ARBL.NE01.MBS getcache Status”だとすると、”getcache”の部分から”getchach.xml”を使用することを決定、”ARBL.NE01.MBS”を COACK のコマンド宛先に、”Status”の部分をプロパティーにそれぞれ当てはめてゆく (図 4 参照)。この時、”coast”の部分は STARS における COACK Bridge のターミナル名であるので無視される。その他、タイムスタンプなどの情報もそれぞれ埋め込まれてゆき、最終的にできあがった XML のメッセージは COACK の TCP/IP Socket インターフェースに送信される。

次に、COACK Bridge は COACK からの XML メッセージを受け取ると、内蔵した XML Perser を起動し、順次テキストメッセージに変換してゆく。XML メッセージにはメッセージの宛先の情報も含まれているので COACK Bridge はそれを STARS クライアントの宛先として変換されたテキストメッセージを STARS サーバに送出する。なお、COACK Bridge で使用している XML パーサは XML::Parser モジュールである。

3.3 STARS のターミナル名と COACK のノード名

STARS では複数の機器を一つのプログラムで制御する事も考慮しており、Bridge と呼ばれるクライアントプログラムを作成することでこれに対応している。Bridge はこの他にも複数の STARS サーバや他のプロトコルのシステムを接続する際に使用される。なお、これまで述べた COACK Bridge もその一つである。

STARS では各ターミナルの識別を行うために予めターミナル名を決めるが、Bridge を経由して接続されるターミナル名については Bridge 名とターミナル名の間にピリオドを挿入してメッセージの宛先等を表現する。たとえば、STARS サーバに対し”Br1”という名前の Bridge を経由して接続されたターミナル”Dev3”に対するメッセージの宛先は”Br1.Dev3”のように表現される。更に、Bridge ”Br1”と”Br2”と言う名前の Bridge を経由し接続されたターミナル”Dev2”は”Br1.Br2.Dev2”のように表現される。

一方、COACK では Class Builder Tool 等を使用してサーバ内に仮想の装置クラスが構築される。各ノードの名前については Device クラスとノード名をピリオドでつなげた形で表すことが出来る。たとえば”ARBL”クラスの”NE01”クラス下の”MBS”ノードは”ARBL.NE01.MBS”のようになる。図 5 左は実際のクラス例である。

COACK Bridge を使用すると、これら COACK の各 Device クラスに付属するノードは STARS 側から見た場合、COACK Bridge(ここでは coast と名前付けしてある) を経由し Device クラスと同じ名前の Bridge を経由して接続されたターミナルとして見える (図 5 右)。たとえば COACK のノード”ARBL.NE01.MBS”への STARS 上の宛先は”coast.ARBL.NE01.MBS”と言うことになる。

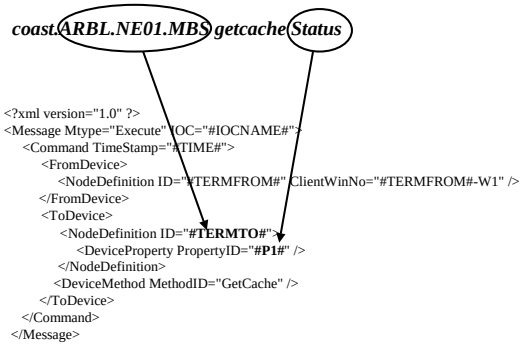

図 4: XML への変換

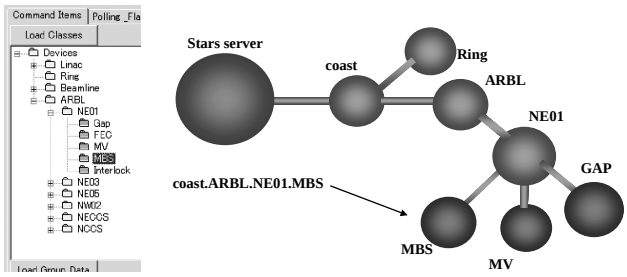

図 5: COACK のクラスと STARS

4 COACK Bridge の改造

これまでの、COACK Bridge は特に大きな問題もなく動作してきたが、幾つかの問題点があった。今回、これらの改善のために COACK Bridge に対し大幅な改造を行った。

4.1 煩雑なソースコードの改善

COACK Bridge の最初のバージョンでは、何よりも動作することが最優先であったため、ソースコード自体が非常に煩雑であった。また、STARS サーバからの入力チェックなども自前で行っていた。今回 COACK Bridge の改造に際し、最初にこの問題を解決することとしたので、後の改造が容易になった。なお、新しい COACK Bridge では STARS サーバからの入力のチェック等は旧 COACK Bridge 作成後に開発した”stars”モジュールを使用している。

4.2 COACK のセッション終了に際する問題

これまでの COACK Bridge では COACK のセッションを開設している STARS 側のクライアントが停止すると、そのままセッションが残ってしまうという問題があった。今回この部分を見直し、COACK のセッションを開設し

ている STARS クライアントが停止した場合、COACK Bridge が代行してセッションを終了するようにした。

実際には STARS クライアントからセッションを開設する”create”メッセージが届くと、STARS サーバにそのクライアントに対するイベントの要求 (flgon) 命令を送るようにしている。もし、flgon したクライアントが STARS から切り離されると、その旨を告げるメッセージが STARS サーバから届くので、それをもとにセッションの終了作業を行う。

4.3 STARS メッセージに関する手直し

STARS では送られてきたイベントメッセージやリプライメッセージに関しては返事を返してはいけないが、コマンドに関しては必ず返事を返さなければならない事になっている。この仕様に関して幾つかの問題のある XML テンプレートの書き換えを行った。

5 実際の利用

COACK の Non-Windows 用インターフェースとして STARS 及び今回改造を行った COACK Bridge を使用した例を図 6 に示す。

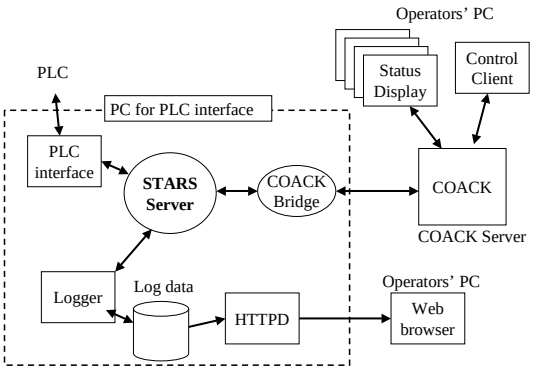

図 6: PF-AR 集中管理システムでの利用例

この例は実際に放射光研究施設 (以下 PF)AR インターロック集中管理システム [6] に利用したものである。ここでは、STARS 及び COACK Bridge は集中管理システムの PLC(Programmable Logic Controller) とのインターフェースを行う PC 上で動作している。なお、このインタフェース用 PC 上では実際に PLC とのデータの送受を行う PLC interface クライアント、ログデータを記録するための Logger クライアントが動作していて PLC のからの情報は STARS サーバと COACK Bridge を経て既存の COACK サーバに転送される。なお、COACK には強力なログデータ管理機構が備わっているが、システム構築当時の事情により非力な PC しか利用できなかったため、ログデータの記録を COACK サーバ以外で行う方式をとっている。

STARS や COACK とは無関係ではあるがインターフェース用 PC 上では Web サーバ (httpd) が動作しており、ログデータの閲覧は Web ブラウザを利用して行うことが出来る。

6 まとめ

以上ここでは、COACK Bridge の詳細及び改造、COACK の「Non-Windows システム用インターフェースとしての STARS 利用」に関する詳細を述べた。

現在 COACK Bridge 及び STARS の動作は良好で、既存の PF 2.5GeV リングビームラインインターロック集中管理システム [8, 9, 10] や今回新たに設置された PF-AR 集中管理システムで安定に動作している。

7 参考文献

[1] I. Abe, et al., ”COACK-II PROJECT ON ACCELERATOR CONTROL KERNEL DEVELOPMENT”, ICALEPCS’99, Trieste, 1999

[2] T. Kosuge, et al., ”COACK APPLICATION FOR THE BEAMLINE INTERLOCK SYSTEM AT THE PHOTON FACTORY”, PCaPAC2000, Hamburg, 2000

[3] I. Abe, et al., ”Recent status on COACK project”, PCaPAC2000, Hamburg, 2000

[4] 阿部勇 他, ”大学研究所間共同開発プロジェクト COACK”, 本技術研究会, NIFS

[5] 小菅隆, 斉藤裕樹, 伊藤健二, ”計測・制御用簡易メッセージ配信システムの開発”, 東北大学技術研究会報告 (2001) 220

[6] 斉藤裕樹 他, ”PF-AR の新しいインターロック集中管理システム”, 本技術研究会, NIFS

[7] 小菅 隆, 小山篤, ”簡易メッセージ配信システム (STARS) の入退室管理システムへの応用”, 本技術研究会, NIFS

[8] 小菅隆, 斉藤裕樹, 伊藤健二, ”放射光ビームライン・インターロック集中管理システム”, 核融合科学研究所技術研究会報告 (1991) 172

[9] 斉藤裕樹, 小菅隆, 伊藤健二, ”LAN を用いたインターロックシステムの監視”, 核融合科学研究所技術研究会報告 (1994) 228

[10] 小菅隆, 斉藤裕樹, 伊藤健二, ”放射光ビームライン・インターロックシステムとネットワーク” KEK Proceedings 95-14 (1996) 23