yarpによるアプリケーション連携

著者	Watanabe Tetsuyou
雑誌名	日本ロボット学会誌 = Journal of the Robotics Society of Japan
巻	30
号	9
ページ	849-852
発行年	2012-01-01
URL	http://hdl.handle.net/2297/35220

doi: 10.7210/jrsj.30.849

解　説

yarpによるアプリケーション連携

Communication between applications with *yarp*

渡　辺　哲　陽*　*金沢大学理工研究域
Tetsuyou Watanabe*　*College of Science and Engineering, Kanazawa University

1. は　じ　め　に

Yarp [1] は Yet Another Robot Platform の略で，ロボットシステム用のライブラリの一つである．端的に述べれば，プログラム（アプリケーション）同士の通信を行うためのライブラリである．ロボットを動作制御するプログラム（C言語による作成を想定）を作成中に Matlab の ”ある ”関数を使いたい，と思ったことは無いであろうか?少なくとも著者にはある．当時は ”諦めて ”自作したが…．Yarp はこのような場合に有用なライブラリである．例えば，Windows 上で走る Maltab で作られたプログラムと Linux 上で走る C言語で作られたプログラムをインターネットを介してつなげることができる．そんなライブラリである．

もともと，Yarp はヒューマノイドロボット用のライブラリとして作成させたミドルウェアであり，Robocup などでは比較的使用されている．もっとも,「Yarp」で検索しても日本語のページがあまり出てこないところをみるに，日本ではあまり普及していないのではと考える．この解説が普及の一助になるのであれば，幸いである．

Yarp は以下の三つのコンポーネントからなる．

libYARP_OS Linux，Windows，Mac，Solaris 上で動くアプリケーション通信用

libYARP_sig 画像や音声などの信号処理データ通信用

libYARP_dev モータ制御用ボード，カメラなどのデバイス上データ通信用

libYARP_OS は先に述べたような様々な OS 上にて走るプログラム同士をつなげるためのコンポーネントである．libYARP_sig は画像や音声などの信号処理で用いられるデータを通信するためのコンポーネントである．libYARP_dev はデバイス間の通信のためのコンポーネントである．

今回は，Yarp のメインとなっている libYARP_OS に焦点を当てて解説を行う．具体例をおりまぜながら，インストールから簡単な使用法まで述べようと思う．

原稿受付
キーワード：Yarp, Communication between applications
*金沢市角間町
*Kakuma-machi, Kanazawa

2. Yarp のインストール

Ubuntu の場合，下記のようにターミナル上で打ち込んでいけばインストールできる．

```
# sudo apt-get install cmake cmake-gui libace-dev subversion
# svn co https://yarp0.svn.sourceforge.net/svnroot/yarp0/trunk/yarp2 yarp2
# cd yarp2
# mkdir build
# cd build
# cmake ../.
# make
# sudo make install
```

他の LinuxOS の場合も上記に順じて CMake [2]，Cmake-gui，ACE（ADAPTIVE Communication Environment）ライブラリ（様々な環境下で使用できるようにするため）[3] をインストールした後で Yarp をインストールすればよい．上記では，subversion を使っているが，ソースをダウンロードしてもよい．なお，ccmake が使えないと後で困るので，CMake だけでなく Cmake-gui もインストールした方がよい．

Windows の場合，まず，CMake をインストールする．Cmake のホームページ `http://www.cmake.org/` からバイナリファイルがダウンロードできるので，これを実行すればよい．その後，`http://sourceforge.net/projects/yarp0/files/yarp2/` から必要なライブラリを含んだバイナリファイルが得られるので，これをダウンロードして実行すればよい．なお，使用する Visual Studio のヴァージョンに応じて使用すべきファイルが異なるので注意する．例えば，Visual Studio 2008 なら yarp_**_v9_x86**.exe である．Matlab とC言語を接続するなどに必要なソースコード群が含まれてないので，同じヴァージョンのソースコード群 yarp-**.zip も合せてダウンロードすることを勧める．

3. Yarp の基本的な使い方

Yarp は，インターネットを介してプログラム間のデータのやりとりを可能にする．サポートされている通信方式は以下の通りである．

Tcp，Udp，Multicast，Shared memory

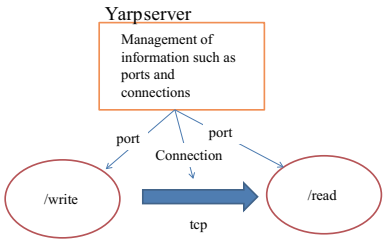

図 1 Simple example of Yarp connection

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\watanabe>yarp write /write
yarp: Port /write active at tcp://127.0.0.1:10002
yarp: Sending output from /write to /read using tcp
test
1 2 3 -4.56
```

/write

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\watanabe>yarp read /read
yarp: Port /read listening at tcp://127.0.0.1:10012
yarp: Receiving input from /write to /read using tcp
test
1 2 3 -4.56
```

/read

図 2 Output example for Yarp connection in FIg.1

特に明示しなければ Tcp が使用される．データ通信の窓口としてポートが提供されている．Fig.1 に Yarp で接続された二つのポートの接続例を示す．Yarp はこのようなポートを接続することで，プログラム間のデータのやりとりを行う．具体的な構成方法は次の通りである．以下は Windows 環境を想定している．

(1) コマンドプロンプトを立ち上げ yarp server と打ち込む
(2) コマンドプロンプトを立ち上げ yarp write /write と打ち込む
(3) コマンドプロンプトを立ち上げ yarp read /read と打ち込む
(4) コマンドプロンプトを立ち上げ yarp connect /write /read と打ち込む

コマンドプロンプトは，合計４つ起動することになる．ポート（の数や名前）はユーザが自由に定めることができる．名前を付ける時の書式は/***である．複数重ねて /robot/leftarm/motor としてもよい．このポート同士をつなぐことでアプリケーション間の通信を実現する．上述の例では，yarp write /write により/write という名前のポートが生成される．このポートは，データ出力用のポートで，コマンドプロンプト上にて数字か文字を直接入力することで入力を行う．yarp read /read により/read という名前の，データ受け取り用ポートが生成される．上述の例では，/write を通して入力されたデータが/read にて受信される．なお，ポートを作っただけでは通信はできない．これらをつなぐ必要がある．そのためのコマンドが yarp connect /write /read である．これは/write を送信，/read を受信としてつなぐ場合のつなぎ方である．もし，送受信のポートを逆にしたいなら，yarp connect /read /write とすればよい．ポートの名前やどのポートとどのポートがつながっているかといった情報は yarp server により管理される．このため，事前に server（最初のコマンド「yarp server」）を先に立ち上げておく必要がある．

この接続を使ってテキストデータと数値データを送った結果を Fig.2 に示す．以上の例は単一のＰＣ上で実行した例であるが，yarp server，/read，/write がそれぞれ別のＰＣ上で走っていたとしても問題は無い．別々のＰＣ上で走っているアプリケーション同士をつなぐ場合にはもう一手間必要である．

ＰＣ１にて：
(1) コマンドプロンプトを立ち上げ yarp namespace /rsj と打ち込む
(2) yarp server と打ち込む

ＰＣ２にて：
(1) コマンドプロンプトを立ち上げ yarp namespace /rsj と打ち込む
(2) yarp write /write と打ち込む

ＰＣ３にて：
(1) コマンドプロンプトを立ち上げ yarp namespace /rsj と打ち込む
(2) yarp read /read と打ち込む
(3) コマンドプロンプトを立ち上げ yarp connect /write /read と打ち込む

yarp server は名前をベースにして管理している ”name server ”である．この yarp server 自身に名前をつける必要がある．デフォルトでは/root と設定されており，同一ＰＣ上でのコネクションを考えるなら名前を付ける必要はない．しかし，複数のＰＣ上でコネクションを作成するなら，”どの yarp server ”を用いるのかを指定する必要がある．このためのコマンドが yarp namespace /rsj である．この場合，yarp server の ”名前 ”は/rsj である．なお,「yarp connect /write /read」はどのＰＣで行ってもよい．名前を統一しても yarp server が見つからず，接続が上手くいかない場合がある．この場合は

yarp where

と打ち込む．Yarp が yarp server を自動的に探してくれる．このコマンドの後，必要に応じて yarp detect と打ち込む．筆者の経験から言うと，このコマンドなしで接続が上手くいくことは，あまりない．なので，重宝するコマンドである．コマンドプロンプト上で実行できるコマンドは yarp help と打ち込むと一覧として確認できる．接続がおかしくなってしまったときは，yarp clean などのコマンドも有効であろう．

次に，プログラム上でどのように Yarp を用いたらよいかについて説明する．参考にすべきプログラム群は *yarproot* /example/os のフォルダ（*yarproot* は Yarp のルートフォルダを表す）にある．単純に送受信の方法を知りたいなら，sample_sender.cpp および sample_receiver.cpp が良いだろう．だが一番参考になるのは summer.cpp（Fig3 参照）かもしれない．このプログラムは/summer というポートを用いて，数値データを受け取り，まず，そのデータを文字表示した後で，合計を計算，次いで結果を「total」という文字

```
#include <yarp/os/all.h>
#include <iostream>

using namespace std;
using namespace yarp::os;

int main(int argc, char *argv[]) {
  Network yarp;
  BufferedPort<Bottle> port;
  port.open("/summer");
  while (true) {
    cout << "waiting for input" << endl;
    Bottle *input = port.read();
    if (input!=NULL) {
      cout << "got " << input->toString().c_str() << endl;
      double total = 0;
      for (int i=0; i<input->size(); i++) {
        total += input->get(i).asDouble();
      }
      Bottle& output = port.prepare();
      output.clear();
      output.addString("total");
      output.addDouble(total);
      cout << "writing " << output.toString().c_str() << endl;
      port.write();
    }
  }
  return 0;
}
```

図 3 Sample code 'summer.cpp'

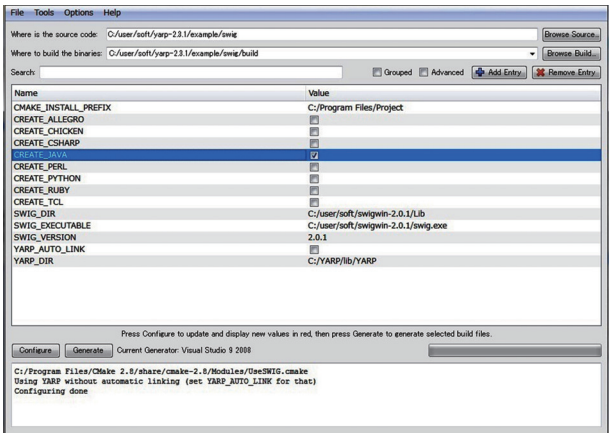

図 4 Cmake preview for using java at Yarp

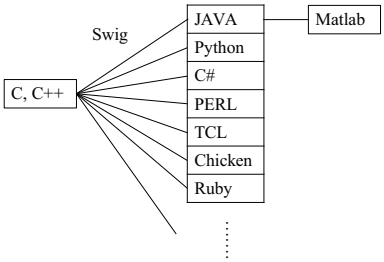

図 5 Schema for connections by different program languages

と同時に出力するというものである．実際の運用では，入力ポートと出力ポートは別のポートを使うことをお勧めするが，簡単に処理方法を学ぶ上でこのサンプルコードはとても役に立つ．難しいコードではないので，一行一行説明することは割愛し，重要箇所についてのみ述べる．まず，ポート作成についてである．ここでは BufferedPort⟨Bottle⟩port という BufferedPort が使われている．Yarp ではこれに加え，Port（使用方法は sample_sender.cpp 参照）が用意されている．違いは値を保持するか否かである．特に問題が無ければ値を保持する BufferedPort の利用を勧める．次にデータに関して述べる．データは基本，”Bottle ”という型（入れ物）を介してやりとりされる．データ受信時は通常，データが来るのを待つ設定になっている．同期のことなるプログラム同士でやりとりをしたいときなど ”データを待ちたくない ”時は Bottle *input=port.read(false) とすればよい．通信可能なデータの主な型は，String，double，int の三種類である（行列を送るための List 構造などもあるが主なものはこれら三種類である）．受け取ったデータ（Bottle *input）から実際のデータを取り出す場合，データの型に合わせて，input->toString().c_str()，input->get(i).asDouble()，input->get(i).asInt() とする．出力の際（データの入れ物：Bottle &output）は，output.addString(total)，output.addDouble(total)，output.addInt(total) を，変数 total の型に合わせて用いる．送受信の順番やその数などは合わせる必要がある．

プログラムにおいてポート同士を接続させたい場合は，Network::connect(”/write”,”/read”); というコマンドを用いる．

4. 異なる言語で作成されたアプリケーション間通信

以上はC＋＋またはC言語を用いて作成されたプログラム間の接続であった．ここでは Matlab で作成されたプログラムの接続について述べる．

追加のインストール作業が必要である．まず必要なのが Swig（Simplified Wrapper and Interface Generator）ライブラリ [4] である．Swig はC＋＋言語と様々なプログラム言語を接続するためのライブラリである．合せて jdk（Java Development Kit）が必要である．これらをインストールする．Windows の場合，Swig 関連ファイルをダウンロードし，swig.exe のパスをシステムの環境変数の Path に追加すればよい．jdk はバイナリファイルとしてダウンロードできるので，それを用いてインストールすればよい．Linux（ubuntu）の場合は sudo apt-get install swig default-jdk（jdk は sun java でも open java でもどちらでも良いが Matlab が使用している java 及び jdk に合わせることを勧める）を実行してインストールすればよい．なお，jdk フォルダの bin フォルダをシステムの環境変数の Path に追加しておくことを勧める．

さて，次に Yarp の再設定を行う．Cmake-gui を実行し，ソースコードを *yarproot* /example/swig に，ビルド場所を *yarproot* /example/swig/build に設定する．その上で一度，configure を行うと Fig.4 のような場面が出てくるので，CREATE_JAVA の項目をクリックし，再度 configure を行う．上手くいったら Generate ボタンをクリックする．すると build フォルダ内に Projet.sln というファイルが生成され

るので，これを立ち上げ，Debug と Release モードでコンパイルする．*yarproot* /example/swig/build/generated_src フォルダ内に**.java というファイルが生成される．*yarproot* /example/swig/src フォルダ内の**.java ファイルを *yarproot* /example/swig/build/generated_src フォルダにコピーする．次いでコマンドプロンプトを立ち上げ，*yarproot* /example/swig/build/generated_src フォルダに移動した後で， **javac -source 1.6 -target 1.6 *java**

と打ち込む．すると Matlab にて Yarp を用いるのに必要な**.class ファイルが生成される．*yarproot* /example/swig/build/generated_src/yarp フォルダを作成し，LoadYarp.class と YarpImageHelper.class 以外の class ファイル全てを移動する．なお，「1.6」は java のヴァージョンなので，適宜変更してもらったらよい．また，Linux 上の場合は，Cmake-gui の代わりに CCmake を使用してもらえばよい．残りの処理はほぼ同じである．

この手法は Matlab 上で java を使用できることを利用したものである．したがって java のプログラムとの通信も同様の方法で可能となる．他に，Python，C#を試したことがあるが，いずれも問題なく動作した．Fig.4，Fig.5 から分かるように，PERL，TCL，Chicken，Ruby などとの接続も可能である．

最後に Matlab 側の設定であるが，作成した**.class ファイルと dll ファイルが使えるように Matlab の path を設定する．Matlab の path は classpath.txt，librarypath.txt にて管理されている．Matlab 上で which classpath.txt（librarypath.txt）と打ち込めばどこにあるのか見つけることができる．classpath.txt，librarypath.txt に

yarproot/example/swig/build/generated_src

yarproot/example/swig/build/Release

を追加して，Matlab を再起動すればよい．classpath.txt は起動時にのみ読み込まれるため注意が必要である．最後に，Matlab 上で LoadYarp と打ち込んで何もエラーが出なければ成功である．

Matlab で通信するためのサンプルコードは，*yarproot* /example/matlab フォルダにある．Fig.6 ではこのうち，yarp_write.m というプログラムを示す．手順はほぼ同じである．Port を作成するときの関数などに若干の違いがあるものの，基本的にはC＋＋言語と同じ関数や使用法にてプログラムを構成できる．yarp read /read で生成される/read ポートと yarp_write.m で生成される/matlab/write ポートをつなげた結果を Fig.7 に示す．

5. お わ り に

今回は，Yarp の基本的な使用方法を紹介することで，使い勝手の良さや簡便さを知って頂くことに主眼を置いた．計測データをバッファリングするのではなく他のＰＣへ飛ばす，Windows 用のドライバしかないセンサデータを用いてロボットを制御（制御は linux 上で）するなど，他にも様々な使い方も考えられる．今回は述べなかったが，OpenCV や ROS などの連携もとれているので，興味のある方は Yarp のホームページ [1] をご参照いただければと思う．

```
LoadYarp;
done=0;
port=yarp.Port;
port.close;
disp('Going to open port /matlab/write');
port.open('/matlab/write');
disp('Please connect to a bottle sink (e.g. yarp read)');
b=yarp.Bottle;
while(~done)
 reply = input('Write a string ("quit" to quit):', 's');
 b.fromString(reply);
 port.write(b);
 if (strcmp(reply, 'quit'))
       done=1;
 end
end
port.close;
```

図 6 Matlab sample code ‘yarp_write.m’

```
>> yarp_write
Yarp library already loaded and initialized, doing nothing
Going to open port /matlab/write
Please connect to a bottle sink (e.g. yarp read)
Write a string ('quit' to quit):test
Write a string ('quit' to quit):1
Write a string ('quit' to quit):-0.3
Write a string ('quit' to quit):quit
>>
```

/matlab/write

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\watanabe>yarp read /read
yarp: Port /read listening at tcp://127.0.0.1:10012
yarp: Receiving input from /matlab/write to /read using tcp
test
1
-0.3
quit
yarp: Removing input from /matlab/write to /read
```

/read

図 7 Communication between Matlab sample ‘yarp_write.m’ and /read

参 考 文 献

[1] yarp. [Online]. Available: http://eris.liralab.it/yarp/.

[2] CMake. [Online]. Available: http://www.cmake.org/.

[3] ACE library. [Online]. Available: http://www.cs.wustl.edu/~schmidt/ACE.html.

[4] Swig. [Online]. Available: http://www.swig.org/.

渡辺　哲陽 (Tetsuyou Watanabe)

2003 年京都大学大学院工学研究科博士後期課程修了．同年山口大学工学部助手，2006 年講師，2007 年金沢大学大学院自然科学研究科講師，2008 年金沢大学理工研究域講師，2011 年金沢大学理工研究域准教授となり，現在に至る．ロボットハンド，ロボット技術の医療応用，微細操作システム開発などの研究に従事．日本生体医工学会， IEEE，日本ロボット学会，日本機械学会の会員．

（日本ロボット学会正会員）