1C-4

# XIM プロトコルのトレース・プログラムの開発

瀧口 一郎，土井 健史，中村 俊雄，高橋 伸行
日本アイ・ビー・エム株式会社 統合システム技術センター

## 1. はじめに

Linux/UNIX のデスクトップ環境を構成する X Window System, Version 11 (X11)上には、いくつかの Input Method (IM)が存在し[1]、標準として X Input Method (XIM)[2]が実装されている。XIM は 1998 年に X11R6.4 をリリースした後、開発は終了しており、いくつかの欠点なども指摘されている[3]が、今日まで使用されている[1]。

XIM は IM サーバおよび X クライアント間のプロトコルとして定義されている。XIM の使用方法が複雑であるため、IM サーバ、X クライアントの組み合わせでいろいろな問題が発生し[4]、有効なデバック法が必要とされる。XIM のデバッグには XIM パケットを即座に解析するトレース・プログラムが有用と考えられる。XIM は X プロトコル上に実装されているため、一般的な X プロトコルのトレース・プログラムである xmon[5]などを利用することもできる。しかし、XIM が複数のイベントやリクエストなどで構成されているため、XIM 自体のパケットの解析は容易ではない。

本論文では新規に作成した XIM プロトコルをトレースする専用のプログラムの実装方法と使用事例を紹介する。画面操作やキー入力操作が即座に解析表示するように実装した XIM プロトコルのトレース・プログラムの有用性を明らかにする。

## 2. XIM プロトコルのトレース・プログラムの実装

今回作成した XIM プロトコルのトレース・プログラム (imtrace) は、XIM パケットを中継する IM サーバとして振舞うことで、X クライアントと IM サーバ間に入り込み、XIM パケットの捕捉を行う。

### 2.1 コミュニケーション・ウィンドウ

コミュニケーション・ウィンドウとは X11 のイベントを介して通信を行うウィンドウのことである。XIM を使用する X クライアントは X サーバから入手した IM サーバのコミュニケーション・ウィンドウを用いて通信を開始する。imtrace のコミュニケーション・ウィンドウは図 1のように X クライアントと IM サーバの間に入り込む。２つのウィンドウを使用したのは XIM プロトコルの転送先を判断しやすくし、プログラムの実装を容易にするためである。

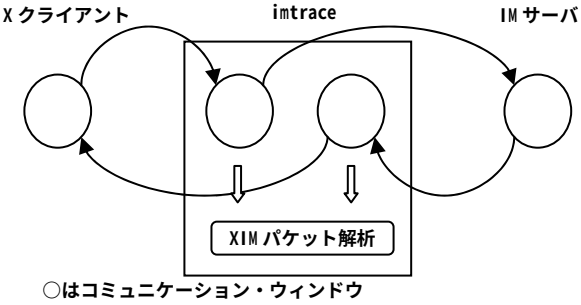


図 1. imtrace の動作

### 2.2 エンディアン

XIM プロトコルでは IM サーバ側でエンディアンの違いを吸収する仕組みになっている。imtrace は XIM プロトコルの XIM_CONNECT に含まれているバイトオーダのデータによりエンディアンを判断する。

## 3. XIM パケットの解析

XIM プロトコルではデータ転送量を少なくするために、属性の ID 化などを行い、パケットを構成し、データ通信を行う。XIM パケットを解析する場合、ID をそのまま表示しても解析が複雑になる。これを解決するため、imtrace はサーバ固有のデータおよびクライアント固有のデータを記憶し、ID から復元して表示する。

### 3.1 サーバ固有のデータ

サーバ固有のデータには IM の属性に関連するものや IM の操作を行うキーリストなどがある。図 2 に IM サーバから得られたサーバ固有のデータを復元した例を示す。

```
###### Client <- Server ######
Major-Opcode: 31 (XIM_OPEN_REPLY) ← IM サーバから文字列と ID の関連付けを通知
XIM
    id: 0, type: XIMStyles, string: queryInputStyle
XIC
    id: 0, type: CARD32, string: inputStyle
    id: 1, type: Window, string: clientWindow
    id: 2, type: Window, string: focusWindow
    id: 3, type: CARD32, string: filterEvents
    id: 4, type: NEST, string: preeditAttributes
    id: 5, type: NEST, string: statusAttributes
    id: 6, type: XRectangle, string: area
    id: 7, type: XRectangle, string: areaNeeded
                                        (中略)
###### Client -> Server ######
Major-Opcode: 54 (XIM_SET_IC_VALUES)
id: 5, value: statusAttributes, type: NEST ← IM サーバから入手した固有データを
                                             復元
    id: 7, value: areaNeeded, type: XRectangle
        x=0, y=0, width=400, height=0
実際のデータ
       +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
----: -----------------------------------------------
0000: 36 00 06 00 14 00 14 00 10 00 00 00 05 00 0c 00  6...............
0010: 07 00 08 00 00 00 00 00 90 01 00 00              ............
```

図 2. サーバ固有データを ID から復元した例

XIM Protocol tracer
Ichiro Takiguchi, Kenshi Doi, Toshio Nakamura, Nobuyuki Takahashi
Integrated System Engineering Laboratory, IBM Japan, Ltd.

### 3.2 クライアント固有のデータ

imtrace は IM サーバとしてクライアントに対して振舞っており、複数のクライアントからアクセスされる可能性がある。このためクライアント固有のデータはクライアントを識別する ID と共に記憶する必要がある。クライアント固有のデータには X クライアント側で利用可能なエンコーディングのリストがある。

## 4. imtrace の使用例

imtrace は画面操作やキー入力操作により生成された XIM の通信データのトレース結果を即座に imtrace のウィンドウに表示する。X11 の標準的な X クライアントである xedit を使用し、２種類の IM サーバ A, B に対して imtrace を使用したトレース結果の例を下記に示す。

### 4.1 IM サーバの同期／非同期

同期／非同期通信のトレース結果を図 3に示す。XIM_SET_EVENT_MASK に対して転送および同期イベントマスクに同じイベントマスクを指定した場合、同期 IM サーバとなる。このことから IM サーバ A が非同期通信、IM サーバ B が同期通信を行っていることがわかる。非同期通信は余分な通信時間がかからないなどメリットも考えられるが、不安定なリモート・アクセス環境ではイベントの順序が崩れたりする可能性があり注意が必要である。同期通信の場合、XIM_FORWARD_EVENT によって転送されたキーイベントに同期フラグが設定されており、このプロトコルを処理した後、XIM_SYNC_REPLY を送り返す。

同期／非同期通信では XIM_COMMIT による変換文字列処理にも特徴がある。同期通信の場合は IM サーバ側から、非同期通信の場合は X クライアント側から、XIM_SYNC_REPLY が送られる。非同期通信では XIM_SYNC_REPLY が送られることがあまりないため、なんらかの理由で XIM_SYNC_REPLY を正常に受け取れない場合、文字入力が続けられない可能性もある。

| IM サーバ A | IM サーバ B |
|---|---|
| ###### Client <- Server ######<br>Major-Opcode: 37 (XIM_SET_EVENT_MASK)<br>forward-event-mask: KeyPressMask<br>synchronous-event-mask: KeyReleaseMask,<br>ButtonPressMask, ButtonReleaseMask, ...<br>(中略)<br>###### Client -> Server ######<br>Major-Opcode: 60 (XIM_FORWARD_EVENT)<br>flag: asynchronous<br>(中略) | ###### Client <- Server ######<br>Major-Opcode: 37 (XIM_SET_EVENT_MASK)<br>forward-event-mask: KeyPressMask<br>synchronous-event-mask: KeyPressMask<br>(中略)<br>###### Client -> Server ######<br>Major-Opcode: 60 (XIM_FORWARD_EVENT)<br>flag: synchronous<br>(中略)<br>###### Client <- Server ######<br>Major-Opcode: 62 (XIM_SYNC_REPLY)<br>(中略) |
| ###### Client <- Server ######<br>Major-Opcode: 63 (XIM_COMMIT)<br>flag: synchronus, LookupChars<br>(中略)<br>###### Client -> Server ######<br>Major-Opcode: 62 (XIM_SYNC_REPLY) | ###### Client <- Server ######<br>Major-Opcode: 63 (XIM_COMMIT)<br>flag: asynchronus, LookupChars<br>(中略)<br>###### Client <- Server ######<br>Major-Opcode: 62 (XIM_SYNC_REPLY) |

図 3. 非同期通信と同期通信のトレース結果

### 4.2 変換文字列の確定

IM サーバ A を使用した変換文字列の確定処理のトレース結果を図 4に示す。この例では、”nihon”と入力し、『日本』と変換・確定している。XIM_FORWARD_EVENT によってキーイベントが処理され、XIM_COMMIT によって IM サーバから確定文字列が送られる。この際、XIM_ENCODING_NEGOTIATION_REPLY によって指定されたエンコーディングを使用して確定文字列を送る。X クライアントは受け取った文字列をロケール・データに基づき、再度コード変換を行う。ここでコード変換に失敗した場合、文字が表示されない場合や予期しない文字が表示された場合には、トレースの内容を解析することにより、IM サーバ側の問題か、X クライアント側の問題かを容易に判別できる。

```
###### Client -> Server ######
Major-Opcode: 38 (XIM_ENCODING_NEGOTIATION)
encoding(name): eucJP, COMPOUND_TEXT
###### Client <- Server ######
Major-Opcode: 39 (XIM_ENCODING_NEGOTIATION_REPLY)
category: name, encoding: default_encoding ← COMPOUND_TEXT と同じ意味
                                          (中略)
###### Client -> Server ######
Major-Opcode: 60 (XIM_FORWARD_EVENT) ← キー“n”を押している
flag: asynchronous
event: KeyPress, serial: 2164, window: 0x3000082, root-window: 0x41
child-window: 0x0, time: 893609, event-window: (487,217), root-window: (490,354)
state: none, keycode: 57, XLookupString: n, keysym: n, same_screen: yes
                                          (中略)
###### Client <- Server ######
Major-Opcode: 63 (XIM_COMMIT)←“1b 24 28 42”は Japanese Character Set JIS C 6226-1983
                              を意味し、“467c 4b5c”は JIS コードで『日本』を意味する
committed-string:
      +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F
----: -----------------------------------------------
0000: 1b 24 28 42 46 7c 4b 5c                  .$(BF|K\
```

図 4. 変換文字列の確定処理のトレース結果

## 5. まとめ

今回作成した XIM プロトコルのトレース・プログラムにより、IM サーバおよび X クライアント間の通信をトレースできるようになった。即座にトレースが表示されるため、操作とイベント、イベントと IM の表示との関係や IM サーバの属性も容易に確認できる。XIM で予期しない動作が発生した際、トレースの内容を確認することで問題判別が容易になる。複雑な XIM の機能を使用した場合や X クライアントと IM サーバの連携が緊密になっている場合であっても、XIM パケットの解析結果を有効に利用することにより、プログラムのデバックを容易にすることが期待できる。

### 参考文献


[1] 徳永拓之, 最適な日本語入力環境を発掘せよ, UNIX USER 2004/5 (http://kodou.net/unixuser/200405/), ソフトバンク パブリッシング

[2] The Input Method Protocol Version 1.0, X Consortium Standard, http://www.xfree86.org/snapshot/xim.pdf

[3] Hideki Hiura, Next Generation Input Method Technology, http://www.openi18n.org/iiimf/Future-IM.pdf

[4] Japanese XIM Server, 問題があるクライアントの対策, http://sanchi.appi.keio.ac.jp/~syl/Imserver/feature.html

[5] XMON - An interactive X protocol monitor, ftp://ftp.x.org/contrib/devel_tools/xmon.1.5.6.README


### 商標