サーバー構築/運用術

第6回 VNC

VNC(Virtual Network Computing)を使うと，PCのデスクトップをGUI(グラフィカル・ユーザー・インタフェース)で遠隔操作できます。2001年5月号で紹介した仮想端末マネージャscreenのGUI版とも言えるソフトウエアで，時間・場所を問わず同じデスクトップ環境を継続することができます。

（ケイ・ラボラトリー　仙石 浩明）

皆さんはインスタント・メッセンジャー(IM)を使ってますか？　私は，遠隔地の人と一緒に仕事をするようになって，IMを手放せなくなってしまいました＊1。電話だと，文字として残らないので確実性に欠けますし，そもそも電話代がバカになりません。電子メールだと，いつ相手が読むか分からないので即時性に欠けますし，読みやすく理解しやすい文章を書くべきなので，それなりに時間がかかります。

IMだと，しゃべった内容をログに残したり，コピー・アンド・ペーストすることによって会話に参加しなかった人へ知らせることもできます。また，自分が暇なのか，忙しいのか，あるいは席を外しているのか，といった状態を相手に知らせる「プレゼンス」という機能があるので，相手の都合を考慮して話しかけることが可能です。メールが基本的に一方向の話しっぱなしであるのに対し，IMは相手の反応を見ながら話を進めていけるので，気楽に話し始めることができます＊2。

このように大変便利なIMですが，残念なことにAOL IM，MSNメッセンジャー，ICQ，Yahoo! メッセンジャーなど，相互にメッセージをやりとりできない規格が乱立しています。それぞれのユーザーと会話するには，それぞれのクライアント・ソフトウエア＊3を使わなければならず，不便なことこの上ありません。

こんな不便な状況を解決してくれるのが，Jabberです。Jabberサーバーは，主要なIMサーバーへのゲートウエイ機能を備えている＊4ので，Jabberクライアントのユーザー同士だけでなく，その他のIMのユーザーとも会話できます。しかもJabberはオープン・ソースなので，だれでも開発に参加できます。

現時点では，完全に日本語に対応しているとは言いがたいのですが，日本でJabberコミュニティが立ち上がれば，早晩解決できるでしょう。日本語のメーリング・リストがまだ無いようだったので作りました＊5。

また，Jabberサーバーも立ち上げました。MSN，ICQ，Yahoo!へのゲートウエイだけでなく，独自に日本語対応＊6を行った Yahoo! サーバーへのゲートウエイを組み込んでいますので，Yahoo! JAPANへ接続することもできます。詳しくは http://jabber.jp/を参照してください。

＊1　いろんな場所にいる何人もの人と，別々にそれぞれ異なる会話を同時進行させることもあります。
＊2　とりあえず「もしもし」と呼び掛けておいて，相手の反応を待ちながら，どう話そうか考えをまとめることがよくあります。
＊3　MSNメッセンジャーのクライアントは，広告が表示され続けるので，使いたくありません。
＊4　AOLの場合，プロトコルを解析することによって接続できるようにしても，AOL側がそれを察知してプロトコルを変更してしまうので，ある日突然接続できなくなったりします。
＊5　参加するには，jabber-subscribe@jabber.jpあてにメールを送ってください（本文は空で構いません）。折り返し確認のメールが送られて来ますので，そのメールに対して返信すれば参加手続き完了です。
＊6　接続先をYahoo!サーバーからYahoo! JAPANサーバーへ変更したほか，Jabberクライアントから送られてくるUTF-8文字列をShift JISの漢字コードへ変換するルーチンを付加しています。オリジナルのソースからの差分をhttp://jabber.jp/で公開しています。

写真1　Linux上のVNCクライアントからWindowsデスクトップを操作

VNC

現時点では，サーバー管理はキャラクタ・ユーザー・インタフェース（CUI）を利用する方が便利*7ですし，まして細い帯域の通信路を介してリモート・ログインする場合は，GUI は使い物にならないのですが，Windowsマシンや Macintosh*8が相手では，そうも言ってられません。

例えば，Windowsマシン/Macintoshのディスクに置いたファイルをリモートから参照したい場合や，TVチューナ内蔵のWindowsマシンでの録画をリモートからスタートさせたい場合など，GUIもリモートから操作できれば便利なのは間違いありません。

AT&Tケンブリッジ研究所で開発されたVNC（Virtual Network Computing）*9は，Windowsマシン/Macintosh上のVNCサーバーが，デスクトップの画面とキーボード/マウスの操作の情報をリモートのVNCクライアントとの間で送受信することにより，リモートからGUIを操作することを可能にします。VNCクライアントを使って，リモートのデスクトップを操作している例を**写真1**に示します*10。

VNCクライアントは数多くの環境に移植されていて，Windows，Macintosh，UNIX互換システムはもちろん，Windows CE，Palm OS，EPOCなどのPDA（携帯情報端末）や，DOS，OS/2などにも移植されています*11。また，Java 仮想マシン上で動作するVNCクライアントもあるので，Java仮想マシンが動作する環境ならば利用可能ですし，JavaがサポートされているWebブラウザ上でVNCクライアントを動作させることもできます。

VNCクライアントからネットワーク経由でVNCサーバーへ接続したときの構成図を**図1**に示します。VNCクライアントは，任意にVNCサーバーから切り離したり再接続したりできますので，screen*12のGUI版として利用することができます。つまり時間や場所を問わず同じデスクトップ環境を継続することが可能になります。ただし，screenがキャラクタ・ベースなので通信路の帯域があまり太くなくても十分実用になるのに

＊7　2001年4月号の本連載第1回「キャラクタ・ユーザー・インタフェース」を参照してください。
＊8　私はかつてMacintosh用のパッケージ・ソフトを開発していたこともあった（もう10年以上昔の話ですが）のですが，現在は Macintoshを使うどころか，見る機会もほとんど無くなってしまいました。
＊9　http://www.uk.research.att.com/vnc/を参照してください。
＊10　自宅から会社のWindowsマシンを操作しています。ローカルとリモートでTeraTerm Proを使ってasao.gcd.orgへログインしてscreen -xを実行しているので，同じEmacsエディタの編集画面が表示されています。
＊11　一覧がhttp://www.uk.research.att.com/vnc/platforms.htmlにあります。
＊12　2001年5月号の本連載第2回「screen」を参照してください。

対し，VNCはグラフィックス・ベースなので，帯域が十分無いと快適に操作するというわけにはいかないでしょう。

一方，帯域が十分ある場合は，リモートのデスクトップを直接操作しているのとあまり変わらない操作感*13が得られます。PC切り替え器を使って，一組のキーボード/マウス/ディスプレイ*14で複数のマシンを操作している方も多いのではないかと思いますが，VNCを使えば同様のことが実現できます*15。しかもPC切り替え器と異なり，複数のマシンのデスクトップを同時に同じディスプレイ上に表示することもできますし，PC切り替え器はケーブルの長さに限界がありますが，VNCならば帯域さえ十分であれば距離的な制約はありません。

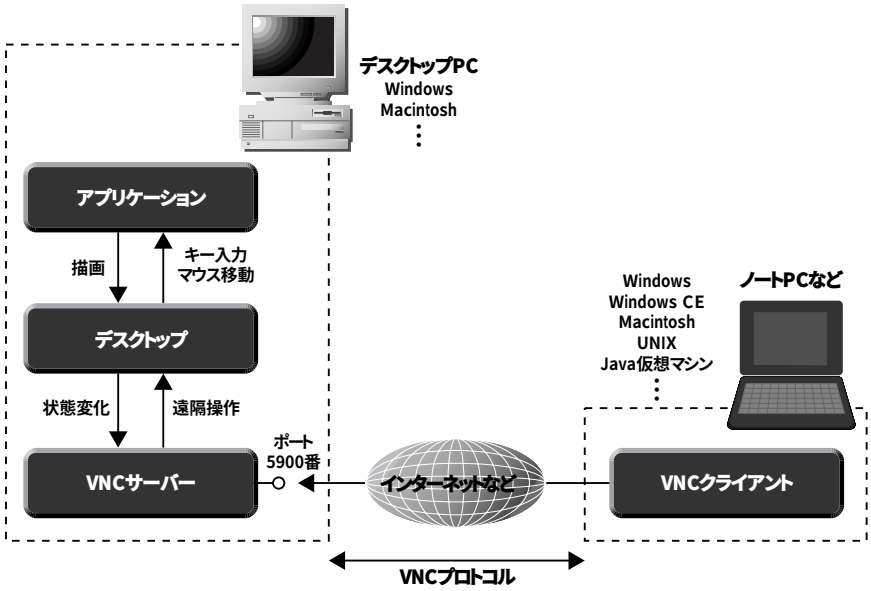

図1　Virtual Network Computing

Xvncサーバー

VNCサーバーは，Windows/Mac OS用だけでなくLinuxなどのUNIX互換システム用もあります。UNIXの場合，WindowsやMac OSのようにデスクトップが1つだけという制約はありませんから，VNCサーバーの数だけデスクトップを作れます。

図2に示すように，VNCサーバー（図中ではXvncサーバー）は，アプリケー

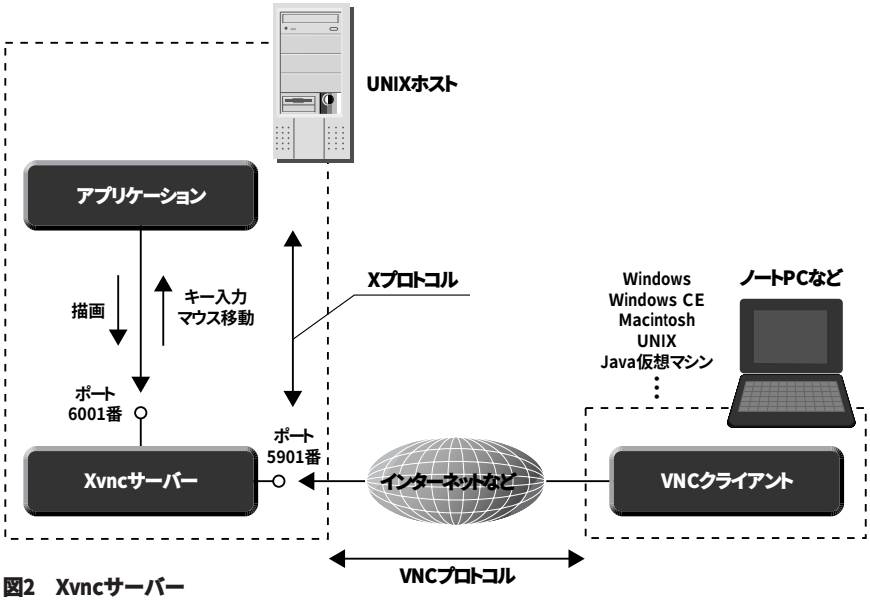

図2　Xvncサーバー

＊13　後述するように，WinVNCサーバーの場合は少々問題があります。
＊14　PC切り替え器は，キーボード（Keyboard），ディスプレイ（Video），マウス（Mouse）の頭文字をとって，KVMなどと呼ばれているようです。
＊15　もちろん，DOSやBIOS画面など，VNCサーバーが動かない環境下では使えませんから，PC切り替え器が全く不要になると言うわけではありません。

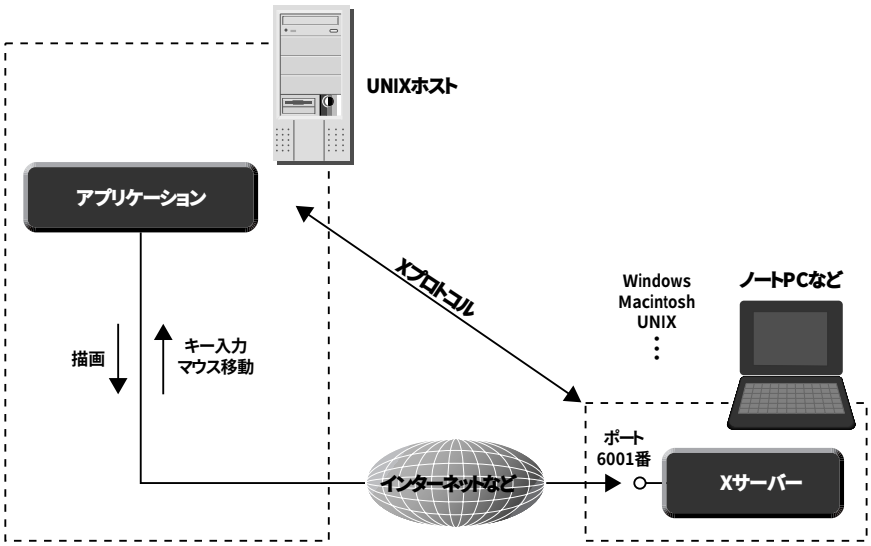

図3　Xサーバー

ション側から見るとXサーバーに見えます。つまりアプリケーション(Xクライアント)はポート6000+n(nはディスプレイ番号)番ポートへ接続してXプロトコルで描画命令をXvncサーバーへ送信したり, 逆にキーボード/マウスのイベントをXvncサーバーから受信します。

通常のXサーバーは, **図3**に示すようにディスプレイを持つPC上で動かすのですが, Xvncサーバーの場合, 表示はVNCクライアントで行いますから, Xvncサーバー自体はディスプレイの無い*16マシン上で動かすことができます。XvncサーバーはVNCクライアントから見るとVNCサーバーに見えます。

図2, 図3のどちらの方式も, 操作するユーザー(図中の「ノートPC」)から見れば, ほとんど同じように使えます。それぞれの方式の長所と短所を挙げると次のようになります。

●Xvncサーバー方式(図2)

(1)VNCクライアントが単純

VNCプロトコル(後述)は, Xプロトコルに比べると非常に単純なので, VNCクライアントはかなり単純ですし, 実行時のCPU負荷が軽く, 非力なモバイルPCでも動かすことができます(Palmでさえ実行可能*17です)。

(2)ソースが公開されている

VNCクライアントおよびサーバーのすべてのソース・プログラムはGNU 一般公有使用許諾書(GPL)のもと, 自由に再配布できます。

(3)ほとんどすべての環境でVNCクライアントが利用可能

(1), (2)の特徴から, VNCクライアントは数多くの環境へ移植されました。

(4)X端末エミュレーションが完全に行える

UNIX互換システムのコンソールでXサーバーを動かしたときと変らない操作環境となっています。例えば, Xのすべてのキーイベントが確実に通知されます。

(5)安心して利用できる

オープンソースで公開されており, また多くのユーザーが使い続けるため, これからも改良が進むことが期待できます。

＊16　サーバー・マシンの多くは, 普段ディスプレイがつながれていないか, つないであったとしても, サーバーが別の部屋にあったりして, コンソールを使うことは滅多にないのではないかと思います。
＊17　http://www.icsi.berkeley.edu/~minenko/PalmVNC/を参照してください。

●Xサーバー方式（図3）

(1)データ転送量が少ない

VNCプロトコルと比べるとXプロトコルは機能豊富なので，クライアント・サーバー間で通信するデータ量を最小化することができます。半面，機能豊富なXプロトコルをサポートするためにXサーバーはかなり複雑で，CPU負荷も高くなります。

(2)Xサーバーのソースが公開されていない

UNIX互換システム＊18以外の環境で動作するXサーバーのほとんどは商用の製品であり，ソースが公開されていません。しかも，あまり売れないためか，その種の製品を店頭で見かけることは滅多になくなってしまいました＊19。

(3)Xサーバーが利用可能な環境が限定されている

Xサーバーは複雑であり，ある程度CPUパワーが高いマシンでないと利用できません。

(4)X端末エミュレーションが不完全

UNIX互換システム以外のXサーバーは，ユーザーの数が少ないためか，本来のXサーバーと挙動が異なる点があります。例えばAltキーのイベントが通知されないなど，かなり致命的な問題がある製品もあります。

LinuxでXvnc サーバーを立ち上げるときの実行例を**図4**に示します。vncserverコマンドは，適切なオプションを設定した上でXvncサーバー・プログラムを起動するperlスクリプトです。自動的に未使用ディスプレイ番号を探して割り当てます。図4の場合，ディスプレイ番号は1が割り当てられ，環境変数DISPLAYには「:1.0」が設定されます。VNCクライアントが接続するためのXvncサーバーのポート番号は，5900 + ディスプレイ番号になるので，この例の場合は5901番になります。vncserverコマンドに未使用ディスプレイ番号を探させる代わりに，コマンドの引数として指定することもできます（**図5**）。Xvncサーバーは，**図6**のように終了させるまでは動き続けるので，作業中のデスクトップを異なるVNCクライアントから継続して利用することができます。

vncserverコマンドは，Xvncサーバーを起動すると ~/.vnc/xstartupを実

```
asao:/home/sengoku % vncserver↵
New 'X' desktop is asao.gcd.org:1

Starting applications specified in /home/sengoku/.vnc/xstartup
Log file is /home/sengoku/.vnc/asao.gcd.org:1.log
```

図4　Xvncサーバーの立ち上げ

↵このマークで改行

```
asao:/home/sengoku % vncserver :4↵
New 'X' desktop is asao.gcd.org:4

Starting applications specified in /home/sengoku/.vnc/xstartup
Log file is /home/sengoku/.vnc/asao.gcd.org:4.log
```

図5　ディスプレイ番号を指定したXvncサーバーの立ち上げ

```
asao:/home/sengoku % vncserver -kill :4↵
Killing Xvnc process ID 1713
```

図6　Xvncサーバーの終了

＊18　当然，Mac OS Xを含みます。
＊19　もし仮にユーザー数が減っているのだとすると，製造中止およびサポートの打ち切りも懸念されます。

```
#!/bin/sh
xrdb
twm &
xsetroot -solid grey
xload -geometry 120x94-92-0 \
        -xrm '*overrideRedirect:on' -borderwidth 0 &
xclock -analog -update 1 -geometry 94x94-0-0 \
        -xrm '*overrideRedirect:on' -borderwidth 0 &
```

図7　~/.vnc/xstartupの例

写真2　Xvncサーバーの中でVNCクライアントを実行

行します。普段使うウインドウ・マネージャ(GNOMEやKDE)やその他のクライアント・プログラムを実行するshスクリプトを書いておくと良いでしょう。その一例を**図7**に示します。

そして，VNCクライアントからXvncサーバーへ接続します。するとX端末エミュレータと同じような感覚で使うことができます。**写真2**にWindows上のVNCクライアントからXvncサーバーに接続したときの例を示します。Emacsエディタのshellモードからvncviewer(Linux用のVNCクライアント・プログラム)を実行して，別のWindowsマシン*20のVNCサーバーへ接続しています。

VNCプロトコル

VNCプロトコル*21は，クライアント側のプログラムが，極力単純になるように設計されたプロトコルです。Xプロトコルが，データ転送量が少なくなるように設計されたのと対照的です。

どのくらい単純かというと，描画関係は次の一種類の命令しかありません。

●長方形描画命令

描画する方法として次の3種類があります

・指定した色で塗りつぶす
・サーバーが転送するビットマップ・イメージを描画する
・画面上のその他の長方形領域からコピーする

Xプロトコルには，直線や円弧などをはじめとしてさまざまな描画命令がある

＊20　ここで元のWindowsマシンへつないでしまうとループしてしまいます。
＊21　詳しくは，http://www.sra.co.jp/people/akira/vnc/等を参照してください。

ほか，文字は文字コードでアプリケーションからXサーバーへ伝えられます。従ってXサーバーはさまざまな描画命令をサポートしなければならない上に，Xで表示するすべてのフォントを持っていなければなりません。

VNCプルトコルでは，図形も文字もすべて長方形描画命令でクライアントへ伝えられます。すなわち，水平あるいは垂直の線分は，塗りつぶした長方形を描画する命令として伝えられますし，その他の図形および文字は，長方形のビットマップ・イメージ描画命令として伝えられます。

イメージとして送っているので，文字コードで送る場合に比べれば，転送するデータ量は当然大きくなります。斜めの線分や円弧などの曲線も，それを含む長方形のイメージとして送られますから，単純で大きな図形であればあるほど，Xプロトコルにくらべ，VNCプロトコルは太い帯域を必要とするでしょう。

しかし，実際のデスクトップでは，あまり大きな文字は使いませんし，単純で大きな図形を描画することもまれでしょう。しかも，画面全体を書き換えることはほとんどなく，大抵の場合ごく一部を書き換えるだけで済みます。

確かにXプロトコルに比べればデータ転送量は多いのですが，デスクトップで通常のGUI操作をする限りは，VNCプロトコルのこの欠点はあまり問題とはならないようです。実際，私はわずか128kビット/秒の細い帯域*22を通して自宅のVNCクライアントから職場のVNCサーバーへ接続する（後述）ことがあるのですが，非常用としては十分実用になります。

従ってVNCプロトコルは，データ転送量が増えるという欠点よりも，VNCクライアントが単純で済むという利点の方が大きいと言えるでしょう。

ポーリング

むしろ問題なのは，Windows上のVNCサーバーであるWinVNCサーバーの現在の実装方法です。Xvncサーバーの場合ですと，アプリケーションはXvncサーバーに対して描画命令を送るので，それをそのままVNCプロトコルに変換するだけで済むのですが，Windowsは仕様がオープンでないこともあり，アプリケーションがどのような描画を行っているか，WinVNCサーバーから把握することは困難です。そこで現在のバージョンのWinVNCサーバーでは，ポーリングすなわちアプリケーションが描画した後の画面を調べることにより変化した部分を検出します。

画面全体を毎回ポーリングすると極端に遅くなってしまうので，WinVNCサーバーは画面のどの部分をポーリングするか次のオプションから選択できます（**写真3**参照）。

・全画面（Poll Full Screen）

すべての画面変化を確実に検出できますが，極端に遅くなります。

・一番手前にあるウインドウ（Poll Foreground Window）

一番手前にあるウインドウをポーリングします。ただし，このオプションのみではコマンド・プロンプト（いわゆるDOSウインドウ）の画面変化が検出できません。

・マウス・カーソルがあるウインドウ（Poll Window Under Cursor）

マウス・カーソルがあるウインドウをポーリングします。「一番手前にあるウインドウ」の場合と同様，コマンド・プロンプトの画面変化は検出できません。

＊22　OCNエコノミーを利用しています。今まで私の自宅はフレッツADSLのサービス・エリアに入っていなかったのですが，ようやく2001年7月23日からサービス受け付けが始まるようです。この記事が掲載されるころには申し込みを済ませていることでしょう。

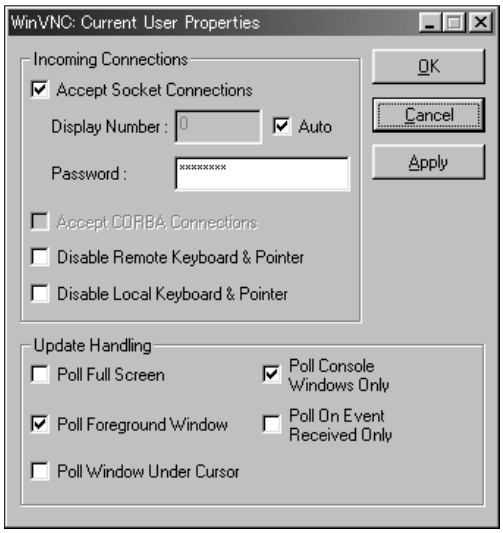

写真3　WinVNCサーバーのプロパティ設定

```
C:\>ssh -L 5910:localhost:5901 -l sengoku asao
Enter passphrase for RSA key 'sengoku@toyokawa':XXXXXXXXXX
Last login: Sun Jul 15 11:28:56 2001 from toyokawa.gcd.org
Linux 2.2.19.

Insanity is the final defense ... It's hard to get a refund when the
salesman is sniffing your crotch and baying at the moon.

asao:/home/sengoku %
```

図8　sshのポート・フォワード機能を利用
この後，VNCクライアントでlocalhost:10に接続する。

・コンソール・ウインドウ（Poll Console Windows Only）

コマンド・プロンプトの画面変化を検出します。

・イベントを受け取った時（Poll On Event Received Only）

VNCクライアントからキーボードあるいはマウスのイベントを受け取ったときのみ，ポーリングを行います。細い回線経由でVNCを使う時のためのオプションと言えるでしょう。

全画面でなく，一番手前あるいはマウス・カーソルがあるウインドウをポーリング対象とすることにより，速度低下をある程度回避できますが，ポーリングの必要がないXvncサーバーに比べれば当然遅くなりますし，ポーリングの対象外の領域で描画が行われると，それを検出することができなくなります。つまりWindowsデスクトップの表示内容とVNCクライアントの表示内容が一致しなくなるわけで，操作感の低下を招きます。

また，現行のWinVNCサーバーは，日本語変換関係のキーなどが正しく扱えないという問題もあるようです。仮に十分広い帯域が利用できて，WinVNCサーバーを十分高速なPC上で走らせたとしても，ローカル・デスクトップと同様の操作感を得ることは難しそうです。

従って，WindowsとXを同時に使いたい場合は，WindowsマシンでVNCクライアントを使ってXvncサーバーへ接続する方が，XでVNCクライアントを使ってWinVNCサーバーへ接続するより，適していると言えます。

セキュリティ

VNCプロトコルでは，まず最初にサーバーがクライアントの認証を行います。サーバーが16バイトのチャレンジをクライアントへ送り，クライアントはユー

ザーが入力したパスワードを使ってこのチャレンジをDES*23で暗号化してレスポンスとしてサーバーへ返します。レスポンスが適正であれば，サーバーは接続を受け付けます。

パスワードがネットワークを流れることはありませんから，ほぼ安全と言えそうですが，その後のデータ転送は一切暗号化されません。すべてのキー入力および画面の状態がVNCサーバーとクライアントの間で送受信されるわけで，パスワードなどを入力すれば盗聴される恐れがあります。従ってsshのポート・フォワード機能*24などを利用してVNC通信全体を暗号化するべきです。

例えば，Windowsマシンtoyokawa上のVNCクライアントから，Linuxマシンasao上のXvncサーバー（ディスプレイ番号は1）に接続するには，まずコマンド・プロンプトで**図8**のように実行して，toyokawaのポート5910番がasaoのポート5901番へ暗号化されて転送されるように設定します。次に，VNCクライアントでlocalhost:10に接続すれば，asaoのXデスクトップを操作できます（**図9**参照）。

ファイアウオールの内側にあるVNCサーバーに対して，外部から接続するにはもう一工夫必要です。**図10**に示すように，VNCクライアント，VNCサーバーの両方がそれぞれ外部からアクセスできない内部LAN上にある場合，両方から直接アクセスできるsshサーバー（図中の要さいホスト）があれば，ポート・フォワードを2本直列につないで，VNCクライアントからVNCサーバーへ接続できます。

つまり，まずVNCクライアント側のLAN 上のマシンで**図11**のようにsshを実行して，5902番ポートを要さいホストazabu.klab.orgの12345番ポート*25へ転送されるように設定します。

-gオプションは，転送する5902番ポートをローカル・ホスト以外のホストからアクセスできるようにするための指定で，sshクライアントを実行するマシン（この例の場合は，asao）とVNCクライアントを実行するマシンが異なる場合は

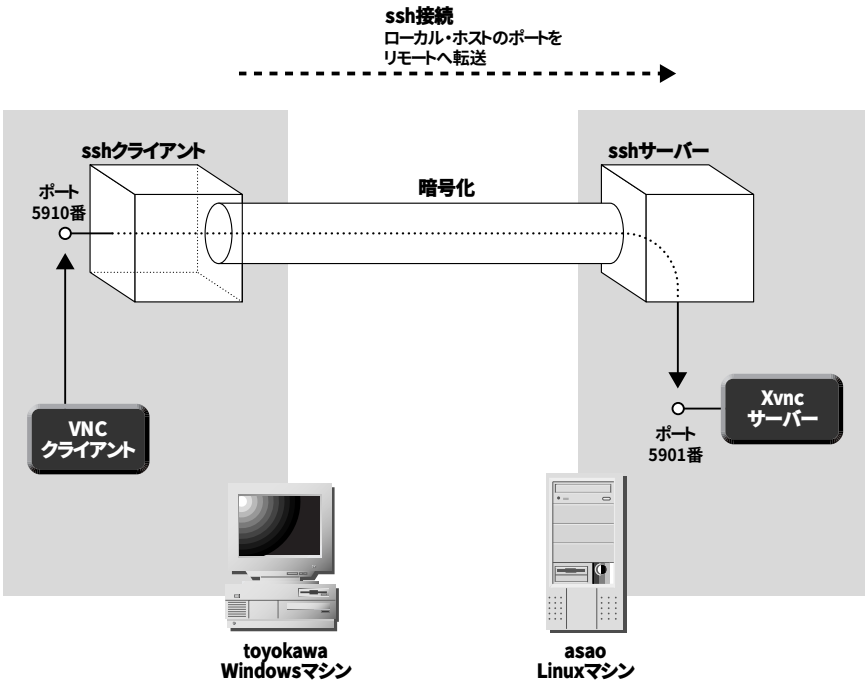

図9　sshのポート・フォワード機能を利用してVNC通信を暗号化

＊23　DES（Data Encryption Standard）は1970年代に米国商務省で制定された秘密かぎ暗号システムです。
＊24　詳しくは，2001年2月号の連載「実践で学ぶ，一歩進んだサーバー構築・運用術」の第11回「ssh（後編）」を参照してください。
＊25　未使用ポートであれば，何番を使っても構いません。

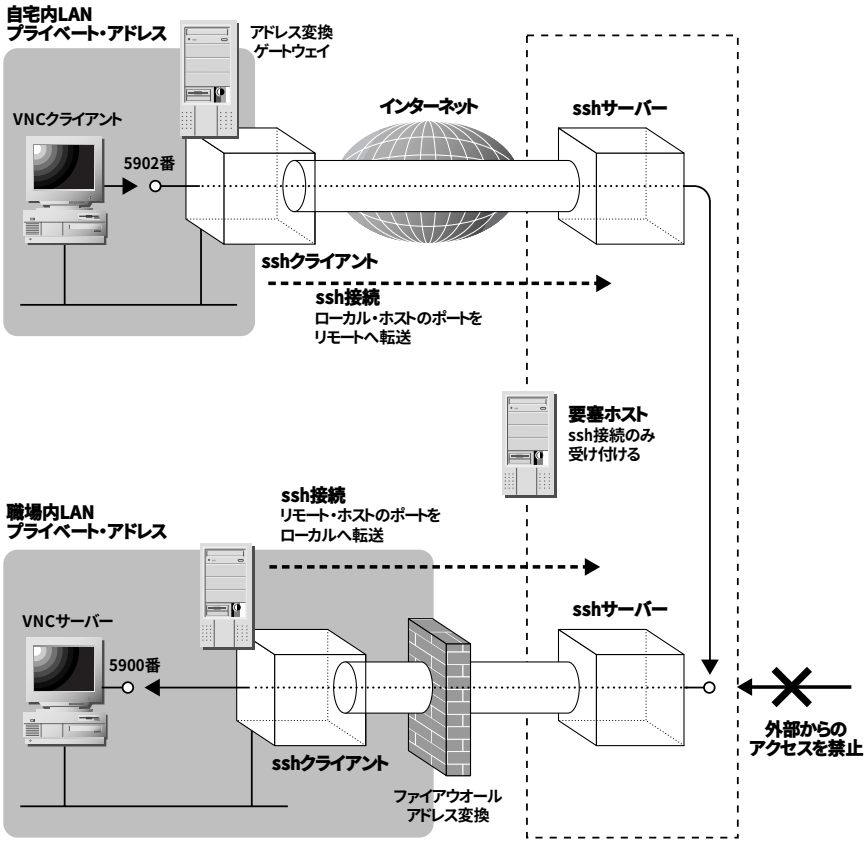

図10　sshを使ったプライベートLAN同士の接続

```
asao:/home/sengoku % ssh -g -L 5902:localhost:12345 azabu.klab.org↵
Last login: Sun Jul 15 15:00:33 2001 from asao.gcd.org
azabu:/home/sengoku %
```

図11　VNC クライアント側の ssh 接続

```
kamiya:/home/sengoku % ssh -R 12345:10.10.1.2:5900 azabu.klab.org↵
Last login: Sun Jul 15 15:01:25 2001 from asao.gcd.org
azabu:/home/sengoku %
```

図12　VNCサーバー側のssh接続

必要です。

次に，VNCサーバー側のLAN上のマシンで**図12**のようにsshを実行して，azabu.klab.orgの12345番ポートを，LAN内にあるVNCサーバーが動作中のマシン10.10.1.2の5900番ポートに転送されるように設定します。

すると，VNCクライアントでasao:2へ接続すれば，VNCサーバーに接続できます。この例では，VNCクライアント側のLAN内ではVNC通信は暗号化されません。LAN内のどのマシンからもasao:2へ接続してVNCサーバーに接続できるという利点があるのですが，LAN内で盗聴される恐れが少しでもあるなら，図9で示した例のようにVNCクライアントとsshクライアントは同一のマシン上で実行すべきです。VNCサーバー側のLANについても同様です。

盗聴される可能性が（ほとんど）無いLANとは，例えばスイッチング・ハブのみで構成されていて，2台のマシンをつなぐケーブルおよびハブのいずれにも細工される可能性が無い場合などです。家庭内LANなど，1台程度のスイッチング・ハブで構成される小規模のLANの場合は，マシン自体が侵入されない限り安全とみなして構わないでしょう。