インターネット/イントラネットの動的コンテンツ

久野　靖*

1997.7.17

1　はじめに

筑波大学公開講座「ネットワーク社会を支える技術」にようこそ。本日 4 日目は、私こと久野が「インターネット/イントラネットの動的コンテンツ」という題でお話をさせて頂きます。

最初に私自身のバックグラウンドについて述べさせて頂きます。私自身の専門は「プログラミング言語」「基本ソフトウェア」「ユーザインタフェース」などでして、ネットワークの専門家というわけではありません。ただ、インターネット/イントラネットは非常に広い範囲の技術の集合体であり、その中には私の専門である基本ソフトウェアと重なり合う部分も多くありますし、Java 言語などはプログラミング言語そのものなわけです。また、WWW の利用が広まったころにそれに関係する書籍の翻訳をさせて頂いたことなどもあり、このようなお話をさせて頂く機会を得ることとなりました。

ただし、上述のように本日のテーマの範囲はとても広いため、2 時間という時間内ですべての話題について詳しく説明していくのはまったく不可能です。しかし皆様としてはこのテーマに関係のある用語はひととおりカバーして欲しいだろうと思いますので、基本的に広く浅く概要を説明していき、いくつかの重要な/多くの人が興味を持たれるであろう箇所についてだけ重点的にやや詳しく取り上げる、という形を取りたいと思います。しかしそれ以外の箇所でも、興味があれば随時質問をして頂いて結構です。私に分かる範囲でお答えして行きたいと思います (私がすべての話題について熟知しているわけでないことはご了承ください)。

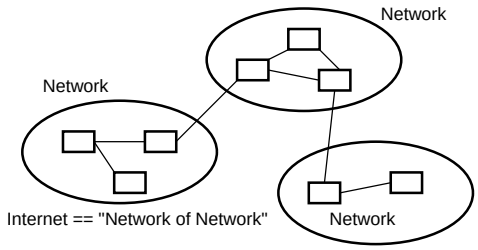
では、よろしくお願いします。

2　ネットワークと WWW

□ QUIZ: インターネットとは何だと思いますか?

- [A.] サイバースペースでネットサーフィンができるところ。
- [B.] ホームページで画像や音声のたっぷり入った情報が見られるもの。
- [C.] 世界最大のパソコン通信。
- [D.] ネットワークのネットワーク。

□ インターネットとは?　→　「各地のネットワークを相互接続することで、世界全体をカバーすることとなった、巨大なネットワークの集合体」



□ ちなみに「イソターネット」という用語もある

□ インターネットのはじまりは…

- 1960's → ARPANET。米国で軍が予算を出して研究。
- 1970's → その利便性が計算機科学者に知られ拡大。
- 1980's → 米国:　国がサポート。日本:　WIDE プロジェクト。
- 1990's → 急激に成長し、情報社会のインフラとなる。
- 2000's → ???

□ インターネットの何がすごいのか?

- A. いくら使ってもタダ???
- B. 距離や時間の概念を消滅させる。
- C. いつでも、どこでも、誰とでも → モーバイル技術
- D. 汎用の/万能のメディア → 計算機技術の効果

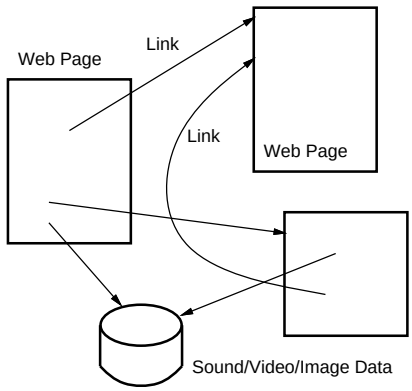
□ WWW 以前のインターネット

*筑波大学大学院経営システム科学専攻

- 最初のネットワークサービス → 電子メール == 郵便
- 電子ニュース → テキスト記事のブロードキャスト
- telnet → 遠隔ログイン
- ftp → ソフトウェアの配布 → フリーソフト文化
- gopher → メニューベースの情報システム
- wais → キーワード検索ベースの情報システム
- …で、多くのものはあったが、すべて「バラバラ」で各種のコマンドを操らなければ使いこなせなかった。

□ WWW の生まれたきっかけは…

- CERN(欧州粒子物理研究所) の Tim Berners-Lee が考案
- ハイパーテキスト → 文書の中に、他の文書へのリンクが埋め込まれている → リンクを選択すると、そのリンクにつながっている文書にジャンプする



- WWW は、ネットワーク中に「くもの巣 (Web)」のようにまたがったハイパーテキスト
- 個々のページはネットワークの各所にある WWW サーバによって提供されている。
- ネットワーク上の資源はすべて URL(Uniform Resource Locators) によって指し示すことができる

```
プロトコル 所在
    ↓      ↓
news:<5315$fw27@gssm.otsuka.tsukuba.ac.jp>
mailto:kuno@gssm.otsuka.tsukuba.ac.jp
http://www.w3.org:80/hypertext/index.html
          ↑       ↑     ↑
       サーバ   ポート ディレクトリ+ファイル
```

- この「どれでもあり」な URL が WWW のキーアイデアの 1 つ
- Web ページは HTML(HyperText Markup Language) で記述されている
- HTML では、文章に「ここは表題」「ここは箇条書」といった印 (マークアップ) を付加することで構造を規定

```
<HTML>
<HEAD><TITLE>Sample Page</TITLE></HEAD>
<BODY>
<H1>サンプルページだよーん</H1>
<UL>
<LI>なんか
<LI>かんか
<LI><A HREF="page9.html">戻る</A>
</UL>
</BODY></HTML>
```

- リンクも「ここがリンクで、選択されたらこの URL へジャンプ」と指定
- Web ブラウザ (WWW を見るためのプログラム) は HTML ファイルを整形してそれなりに表示 → どう見えるかはユーザの環境によって変化
- リンクを選択すると、Web ブラウザは対応する URL の資源 (Web ページに限らない) を取り寄せて来る → それが HTML ファイルならそれを表示

□ イントラネットの出現

- WWW の急速な広まり → その使いやすさ、非専門家への浸透性の認識 → 企業情報システムにも適用したい
- しかし、インターネットは「公道」 → 部外秘な情報は流せない
- 伝送容量も限られている → それに適合した利用しかできない→ 要するに「仕事に合わせる」ことはできない
- 半面、以前から企業は社内ネットワークを保持 (オンラインシステム、社内電話、…) → これらを TCP/IP で (インターネットと同一の技術によって) 運営 → 「イントラ」ネット
- イントラネット間の接続…社内専用線のこともあるが、これは高価 → インターネットによる中継: 「エクストラネット」 (inter : 「ネットワークをまたがる」、intra : 「内部の」extra : 「外部の」)

□ イントラネットの用途は…

- 「社内むけのインターネット」。実際には「社内むけ WWW」がメイン?

- データの保護、帯域の保証、必要なソフトの導入 → 「仕事に使える WWW」

3 インターネットのコンテンツ

□ コンテンツとは? → 「中身」のこと。封筒の中身でも、鞄の中身でもいい。しかしここでは、インターネット上の情報流通技術 (典型的には WWW) の上でやりとりされる「情報」のこと

□ 静的コンテンツとは? → 一度送られると、もはや変化しないもの。典型的には、Web ページそのもの

□ QUIZ: 次のもののうち、静的「でない」コンテンツはどれか

- A. 写真などの画像
- B. ムービー (動画) ファイル、サウンド (音声) ファイル
- C. アニメーション GIF(ページの中で絵が動く)
- D. リアルオーディオ、インターネット電話
- E. アクセスカウンタ

□ マルチメディアとは? → 計算機の内部では、いちばん多く使われる情報表現 (メディア) は「文書」。だからそれ以外はすべてマルチメディア。たとえば:

- 画像、動画、音声
- その他… におい、動き、などなど???

□ このため、WWW は「ハイパーテキスト」でなく「ハイパーメディア」だという説も

□ 「マノレチメディア」という用語もある

□ コンテンツの種別とは? → 要するに上で述べたような分類のどれか。だけでなく、ファイル形式なども問題となる。「このプログラムでは GIF は読めるけど JPEG は読めない」など

□ MIME とは? → MIME は「Multipurpose Internet Mail Extension」。要するに電子メールで普通の文書以外の何でも送れるようにするための規格。WWW ではいろいろなデータを送るのに MIME の規格が援用されてる

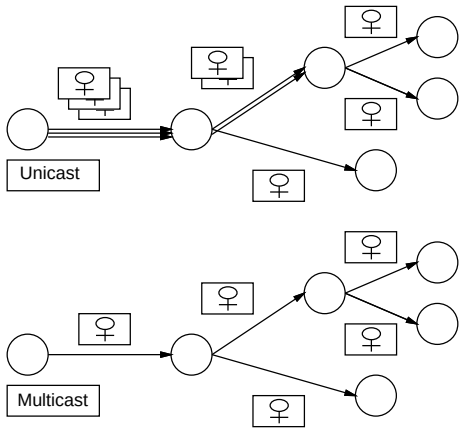
□ MIME タイプとは? → MIME で規定されている、コンテンツ種別の表記法。たとえば:

- <TT>text/html</TT> --- HTML テキスト
- <TT>text/plain</TT> --- ただのテキスト
- <TT>image/gif</TT> --- GIF イメージ
- <TT>application/postscript</TT> --- PostScript コード
- <TT>text/x-server-parsed-html</TT> --- サーバ解析 HTML(SSI)

□ このように「<TT>種別/副種別</TT>」という形をしている。「<TT>x-</TT>」で始まるのは「まだ正式規格じゃないよ」

□ 連続メディア → インターネット電話、インターネット TV、インターネットラジオのように、「ずーっと続いているような」メディア → そんなデータ、どうやって配送するんだと思います?

□ マルチキャスト → 1 対多通信。テレビのようにチャネル番号があって、それぞれの節目のところで「このチャネルはこっちの方へも流してくれ」などと頼める (cf. 通常の通信==unicast)



□ この機能がないところを通るときは「トンネル」を掘って通過する→ MBone (Multicast BackBone)

□ なぜマルチキャストが重要か? → 本当に大量の連続メディアを多数のユーザが受け取るには、これしか方法がない。通常の 1 対 1 通信 (ユニキャスト) では、送り切れない

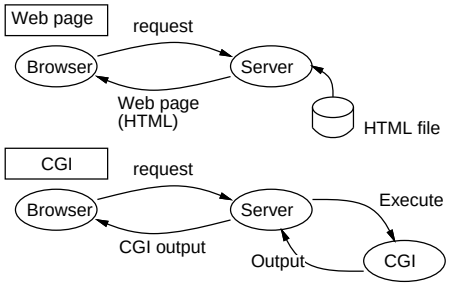
□ それで結局、動的コンテンツとは? → 広い意味では、「送り始めるときに内容が決まっていない」もの (だからラジオなども含む) → ここではもっと狭く、「プログラムできるようなコンテンツ」

4 動的コンテンツの基本: CGI と SSI

□ CGI とは? → Common Gateway Interface の略。そのココロは、Web サーバから下請けプログラムを呼び

出す際の決まりごと。この呼び出されるプログラムのことを「CGI プログラム」と呼ぶ

- Web サーバとブラウザは HTTP(HyperText Transfer Protocol) で通信
- HTTP では、ブラウザとサーバがさまざまな情報を交換
- CGI では、プログラムを呼び出す際にブラウザからの情報を利用可能にする。逆にプログラムからデータに加えて各種の情報が返せる



□ CGI の基本 → 単に URL で指している先が CGI プログラムだと、そのプログラムが起動されてそれが返した結果がブラウザに返される。たとえば次のようにすれば好きなコマンドの出力を送り返せる

```
echo 'Content-type: text/plain'  ← MIME タイプを通知
echo ''                          ←ヘッダの終り
date
uptime
```

これを呼び出すには単にリンクでこれを格納したプログラムを指定すればよい:

```
<A HREF="s4-test0.cgi">テスト起動</A>
```

□ CGI へのパラメタ渡し → URL の一部が CGI に渡される。具体的には <TT>QUERY_STRING</TT>(「<TT>?</TT>」より先の部分) と <TT>PATH_INFO</TT>(CGI プログラム名と「<TT>?</TT>」にはさまれた部分がそう

```
echo 'Content-type: text/plain'  ← MIME タイプを通知
echo ''                          ←ヘッダの終り
echo $QUERY_STRING     ←以下テキストデータ
echo $QUERY_STRING
echo $PATH_INFO
echo $PATH_INFO
```

というようなシェルスクリプトを <TT>s4-test1.cgi</TT> というファイルに入れておいて次のようにすると、これらのパラメタつきでプログラムが呼び出せる:

```
<A HREF="s4-test1.cgi/hihihi?booboo">テスト起動</A>
```

□ CGI とフォーム → CGI を普通のリンクから呼び出す代わりに <TT>FORM</TT> タグから呼び出すことができる → フォームのデータが URL 符合化と呼ばれる形に変換されて CGI に渡される → CGI ではフォームのデータを受け取って処理 → そこから先は他の CGI と同じ。たとえば 2 つの数を足す、というのをやってみる。

```
<FORM METHOD=POST ACTION="s4-test2.cgi">
<INPUT TYPE=TEXT NAME=A>た
す<INPUT TYPE=TEXT NAME=B>は?
<BR>
<INPUT TYPE=SUBMIT VALUE="計算!"></FORM>
```

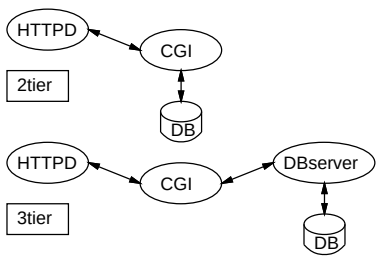
これを処理する CGI は次のとおり。

```
PATH=/usr/local/bin:$PATH      ← cgiparse を使うための設定
eval `cgiparse -form`          ← cgiparse で引数を受け取る
echo 'Content-type: text/plain'
echo ''
R=`expr $FORM_A + $FORM_B`     ←足し算する
echo "$FORM_A + $FORM_B = $R" ←結果の表示
```

□ イメージマップ → CGI をイメージ (画像) のリンクから呼び出し → マウスでクリックした位置が <TT>QUERY_STRING</TT> で渡される → どの位置でクリックしたらどの URL へ行く、というデータ (マップファイル) を参照 → その URL へジャンプせよ、という HTTP 応答を返す

□ SSI: CGI の兄弟分 → HTML ファイルの一部に特別なコマンドを埋め込み → 具体的なコマンドとして、任意のプログラムを起動したり、CGI を呼び出せる → その結果をその位置に埋め込む

□ DB アクセス → お仕事で CGI を使う場合の典型的な用途 → DB 機能は CGI に一緒に組んでもよいし、DB サーバを別に動かして CGI からこのサーバに接続してもよい



□ なぜ今さら CGI か? → CGI はサーバ上で好きなプログラムを起動することができる → 既にあるプログラムを WWW に組み込める

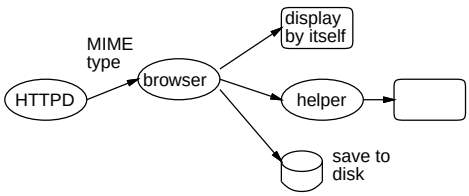
□ CGI の弱点 → 世界中のどこからでも呼び出され得る → セキュリティ対策をしっかりしないといけない

□ その他の CGI の可能性 → サーバ上のデータを複数人で共有するようなものなら何でも → たとえば我々はビジネスゲームのシステムを CGI で開発し授業に使っている

5 クライアント側の拡張: ヘルパーとプラグイン

□ 「ブラウザは、すべての種類のコンテンツを扱えるわけではない。」当たり前。

□ ではどうする? → ヘルパー (補助) アプリケーション → MIME タイプごとに、もしそのようなファイルがやってきたらどのプログラムを起動して表示/演奏するかを指定しておく (もちろん、HTML などいくつかの基本的なデータはブラウザが自ら処理する)



□ 処理できるヘルパーアプリケーションがない時は? → ファイルとしてダウンロードして、ディスクに保存

□ ヘルパーアプリケーションの例: 各種のビューア (PostScript、dvi、PDF、QuickTime Video、AVI、各種音声…

□ ヘルパーアプリケーションの問題点 → あくまで別のプログラム → ブラウザの機能と融合というわけには行きにくい (ある程度はできる。たとえばヘルパー上で何かを選択するとブラウザが対応するページを表示するなど)

□ プラグイン → 「より密接な」ヘルパーアプリケーション。ブラウザの窓のなかに「埋め込まれた」形で実行 → ブラウザとの連帯もよりスムーズ。代表的なプラグイン: ShockWave for Director

□ 問題点 → ありとあらゆるファイルを見るために、ありとあらゆるヘルパーやプラグインを持って来てインストールしなければならない。

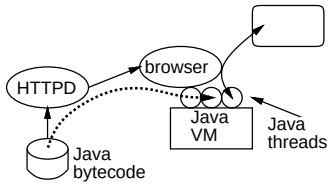
- いくらディスクがあっても足りない。肥大ソフト文明
- 持ってない人には見えないコンテンツだらけ
- インストールしようにも、特定プラットフォーム用しかない → WWW のクロスプラットフォーム性の喪失

6 クライアントプログラミング: Java と Active-X

□ ヘルパーもプラグインも、基本的にはできあいのデータを持って来て演奏/表示するだけ (Shockwave for Director などはある程度動作も記述できるが…)

□ CGI では任意のプログラムが動かせるが、サーバ内で動くため、やりとりに時間が掛かる → アニメーションや対話的操作などには向いていない

□ そこへ出現したのが Sun の Java とアプレット。その基本的な考えかたは、ブラウザにインタプリタを組み込み、ダウンロードしてきたコードをブラウザの窓の「中で」実行させるというもの。



□ Java の歴史:

- Sun Microsystems 社はワークステーションメーカーの老舗
- 研究開発の一環として、家電組み込み情報システム用言語/システムというのをやっていた → Java 言語とインタプリタ
- タイムワーナーの VOD 実験に応募して敗れる → プロジェクトは撤収 → チームの一人が HotJava ブラウザを開発 → 評判を呼び、Java ブームに
- HotJava はすべて Java で書かれていて、Java インタプリタを内蔵しているので、画面の中で Java プログラムを動かすのは造作もなかった (しかしアイデアはなかなかだった)

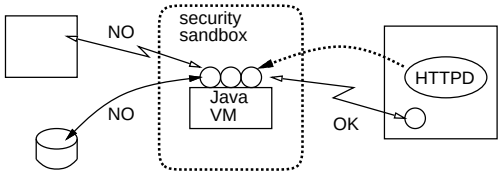
□ Java の技術的に興味ふかい点:

- プラットフォーム独立 → Java ソースコードは Java 仮想マシンコードへのコンパイルを行う → 仮想マシンがあればどのような計算機でも動く
- 実行性能の工夫 → その場コンパイラなどの高速化技術

- オブジェクト指向 → Java 言語はオブジェクト指向言語。C++を綺麗にした、という感じ
- 豊富な API --- Java ブームになったおかげで、さまざまな用途のためのライブラリ API が用意されるようになった。

□ インターネットがらみではセキュリティはとても重要

- セキュリティサンドボックス → アプレットは基本的に危険 (どこの何とも分からないコードがダウンロードされてきて動く) → アプレットにできる操作を制限した。ファイルの読み書き、プログラムの起動は一切禁止。ネットワーク接続はダウンロード元のサーバとのみ可能



- ディジタル署名 → 署名を通じて信頼できることが証明されたアプレットに対してはこの制限を緩められる (JDK 1.1 から)

□ Java でできること、できないこと…

- ○ 普通のプログラミング言語だから、普通にプログラムでできることは何でもできる (別にアプレットに限るわけではない)
- ○ アプレットは WWW のページに埋め込まれた知的なユーザインタフェースとして使うことができる
- ○ 豊富な API によってさまざまな仕事がこなせる
- ○ プラットフォーム独立。Write once, run everywhere
- △ 実行速度は JIT がないと遅い
- △ アプレットはセキュリティモデルによる制約がある

□ Microsoft と Java の関係は…

- Microsoft はパーソナルコンピュータ界の「帝国」… OS も、アプリケーションも、言語処理系も圧倒的なシェアで支配
- ところが Java は Microsoft ではない → 最初は静観、敵対 → 非常なブームになったので取り込みを計る (ビル・ゲイツの才覚?)
- たとえば Visual J++などの処理系、Microsoft Internet Explorer での Java サポートなどなど
- ただし、Microsoft の Java は Microsoft 文明でこそうまく使えるように巧みに誘導? これに対し、Sun は 100% Pure Java で対抗している

□ Active-X → Microsoft が Java Applet に対抗するもの。複数の部品 (Active X Control) をブラウザに持ってきて動かすことができる

- Java がバイトコードをダウンロードしてきて動かすのに対し、Active-X ではマシンコードをいきなりダウンロードしてきて動かす
- そのため、実行速度は速い
- し か し 、WIntel(Windows + Intel Processors) のみを前提
- マシンコードなので、何でもやり放題 → ディジタル署名によって信頼できる人が作っていることを保証といっているが… → 本当にそれで大丈夫なの?

7 クライアントスクリプティング: JavaScript と VBScript

□ Java は簡単、とか言われてもやはりあくまでもプログラミング言語であり、プログラミングの素養のない人には書けない

□ そこで、もっと簡単に誰にでも書けるような、簡単な言語を→ スクリプト言語

□ Web サーバの中で動くスクリプト言語… Netscape Navigator の JavaScript (もとは LiveScript だったが、Java ブームのときに Sun から名前を買った)

- 構文的には Java に似ているが、クラスを作ったりしない。
- 実行は普通のインタプリタ方式
- できること…その場でページの内容を生成する、簡単なユーザインタフェースを作る、どこかへジャンプする、など

□ Microsoft は対抗して…

- Microsoft Internet Explorer で も JavaScript をサポート (名前は JScript)
- おなじみ VisualBasic と同じ構文、というウリで VBScript というのもサポート

□ 要するにクライアントスクリプティングに向いたことは? → 本格的なプログラミング言語でなくてもできる程度のことで、ちょっとページの使い勝手を味付けするようなものを作成

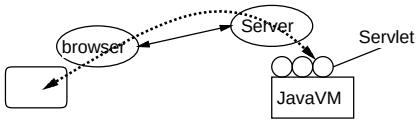
8 サーバ側プログラミング: サーブレット

□ クライアント側プログラミングが可能になった今、CGI のようなものは不要か? → そんなことはない。共有される情報はサーバに置くしかない。

□ しかし、CGI は起動すると、独立したプログラムとして動作し、そのまま終ってしまう → 小刻みなやりとりには不向き、状態を保持するのが面倒

□ このような問題に対して、Sun が提唱しているのが「サーブレット (Servlet)」…アプレットの反対 (?)

- サーバの中に、動的にロードできる。必要ならネットワーク経由でも
- サーブレットもセキュリティサンドボックスを使用
- 一度実行を始めると、サーバの中でずっと動いている
- クライアント (とくにアプレット) と簡単に通信できる
- これらの特性を活かして、複数のクライアント間で情報を共有するようなシステムが容易に作成できる



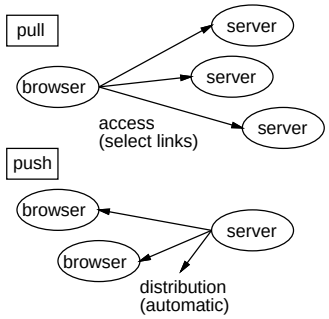
□ Sun の Java Web Server 以外にも、WWW コンソーシアムが開発し公開している Web サーバ Jigsaw もサーブレットをサポートしている。また、Apache、Netscape Server や Microsoft Server などもサポートしている (らしい)。

□ なお、Web サーバの機能をカスタマイズして拡張するメカニズムはいろいろある → ただしその多くはサーバをコンパイルして組み込んだり、管理者が設定 → サーブレットのように実行時に自由に、というのはなさそう

9 プッシュ技 術: PointCast、Marimba

□ プッシュ技術とは? → プル技術の反対

□ プル技術とは? → プッシュ技術の反対…じゃなくて。→通常の WWW では、URL によって指し示されている情報源に情報を取りに行く → だからこそ、最新の/オリジナルそのものの情報が取れるし、事実上利用可能な情報の量に限界がない → 「取りに行く」を pull と表現



□ ではプル方式の弱点は?

- どこに取りに行くか知っていないと取りに行けない → 面白い情報を探すのは結構大変
- 情報が変化したかどうかは再度取ってみないと分からない → ようやく取ってみたら同じだったりする
- 取ろう、と思ってから取り終わるまでに (ネットワークが混雑してる時はとりわけ) 時間が掛かっていらいらする

□ ではプッシュ技術とは? → 「こういう話題を取りたい」と登録しておく → 登録元は、登録した人に向かって情報を送り込む == push

- 登録したカテゴリの新しい話題は自動的に送られて来る → 取りに行く手間もないし、どこに取りにいくかも心配しなくてよい
- 新しい情報があれば自動的に送られて来る
- 読もうと思った時はそこにある → タイムラグがなくなる

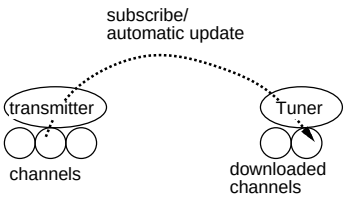
□ 技術的にはどうなってるのか? → プッシュといいながら実は…クライアント側で定期的に「何かない? あったらちょうだい」とサーバに問い合わせている。ユーザが介在しないだけ → 用もないのに読みもしない情報を転送している? → このため、ネットワークへの負荷がどんどん増しているという説も

□ PointCast --- 電光ニュースのイメージ。いろんな情報を送りつけてきて、なんとなく眺めることができる。

- いかにも情報が無駄そうだが、登録時にユーザプロフィールを尋ねてきて、興味のありそうなチャネルを設定してくれる。
- 取り始めた後でもチャネルの取捨選択が自由。CNN とかスポーツとか天気とかさまざまなのがある。
- 自動配信されるのはトップ記事レベルだけで、より詳しく見たい場合は通常のブラウザと同じようにして取り寄せる (待たされる)。
- とはいえ、PointCast のせいでネットワーク帯域があっという間に逼迫した、という説もあり。

□ Castanet (Marimba 社) --- ユーザは Castanet Tuner を操作して、チャネルを眺め、面白そうなチャネルに登録する。

- 登録したチャネルは定期的に内容の更新をチェックし、変化があると変化した分だけを転送してきて、ユーザの手もとには常に最新の内容が残る (こちらは帯域の節約になりそう!)
- チャネルの内容としては、Java アプリケーション、Java アプレット、HTML ページ、Bongo プレゼンテーションがある。
- 話だけ聞くとなんということはないが、実際に待たされずに大きなアプレット等が動くのを見るとなるほどと思う。
- 途中で切れて止まってしまっても、その続きからやってくれる
- ネットワークに繋がっていなくてもいい → モーバイルコンピューティングのニーズにぴったり!



□ 真のプッシュ: マルチキャスト → これなら多数のユーザが登録しても、1 回送るだけで全部に行き渡る (現在は N 人のユーザのためにサーバから N 回送っているのでひどく無駄)

□ 問題点: マルチキャストはすべてのサイトに通っているわけではない (MBone サイトのみ) → マルチキャストに使えるチャネル数も限られている → これらはすべて現在の TCP/IP(IPv4) の実装と設計上の限界から来ている → 新 TCP/IP(IPv6) に以降すれば、もっと自由にどこでもマルチキャストが利用?

□ なお、インターネットラジオとかインターネット中継などもすべてマルチキャストを利用している。これらはあんまり push などとは言わない (WWW とは別だてだから?)

10 2 次元から 3 次元へ: VRML97

□ Web ページはどんなにすごく飾っても、あくまで 2 次元の世界→ 3 次元が欲しい! という人は初期のころからいた

□ 一方、「バーチャルリアリティ (virtual reality、VR)」(実際には存在していない 3 次元空間の映像を計算機内部の計算で作り出し、ユーザがその中を航行できるようなもの) の技術が確立してきた

□ 蛇足ながら、「補完リアリティ (augmented reality、AR)」というのもある → VR が現実世界から途絶した仮想世界なのに対し、AR は現実世界のモノに計算機によるサポートを追加していく

□ VRML1.0: VR や CAD などで使う空間モデル記述ファイルを改良して、3 次元空間とその中にあるモノを記述できる言語を設計 → VRML (Virtual Reality Modeling Language) と名づけた → VRML ブラウザに VRML ファイルをロードすると、ファイルに記述された 3 次元空間の中をナビゲーションできるようになる → さらに、リンクとして指定されたモノに到達すると URL で指定されている別のモノ (別の空間でもよい) に飛ぶ

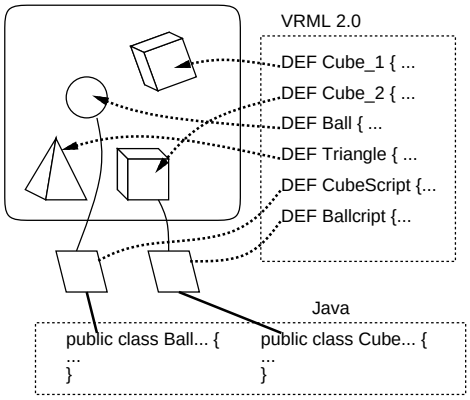
□ VRML1.0 は「すごい! これからは 3 次元だ!」というわけでもてはやされた → しかし、記述された空間は変化のない死んだ世界であり、いるのも自分だけ → その限界からわりとすぐ飽きられた

□ もともと VRML を作った人たちの意図 → 仮想空間 (サイバースペース) を作った後で → モノに動きを与え、また複数のユーザが 1 つの空間を共有できるような仕掛け (サイバーワールド) が欲しい → そのための機能の提案 → 複数の提案が競った結果、Sony ほかが中心となって推した案が通る → VRML2.0 (国際規格に → VRML97)

□ VRML2.0 はイベント+スクリプティングモデル → VRML 言語としてさまざまなノードを定義し、そのノード間の情報流通経路を「ルート (route)」として定義 →

たとえば「このスイッチに触るとこのランプの灯りがつく/このモノがこう動く」といった動作が可能

□ しかし、より高度な動きや他のプログラムとの通信、ファイルの読み書きなどは VRML の中では規定しない → 代わりに、VRML の中で定義できる「スクリプト」ノードにプログラムを対応させることで、普通のプログラム言語を使って任意の処理が行えるようにした → その言語としては、Java がまず使われている → つまり、世界の記述や表現、ユーザの操作の受け取りは VRML、動作の記述は Java、という分担になっている



11 最後に: ネットワークはどこへ?

ここまでで、インターネット/イントラネット上に現在存在しているさまざまなコンテンツとその技術について見てきました。最後にまとめとして、これらが今後どのように発展し、社会にどのような影響を与えて行くかについて少し考えて見たいと思います (放言ともいう)。

まず、インターネットのこれまでにない最大の特徴は、すべての利用者が自分の関わりたいものだけを選んで見聞きでき、それでいて世界中に情報を流通させることもできる「個人メディア」だと考えます。この点が、多数の視聴者向けだが「一方向たれ流し」のマスメディアや、相互にやりとりできるが 1 対 1 のメディアである電話や信書と決定的に違っています。

では、「動的コンテンツ」の方はどうでしょう。動的コンテンツは内部にプログラムを含んでいますから、その作り方によっては自律的に動作し、利用者が自らマシンの前に座って応答しなくても、それに代わってコンテンツを取り出した人たちとやりとりして行くことができます。その意味では、動的コンテンツは「1 次個人メディア」から「2 次個人メディア」への進歩を意味している、と言ってよいかも知れません。

では次に、イントラネットについてはどうでしょうか。イントラネットはインターネットと同じ基盤技術に基づいており、したがって企業内での個人メディアのインフラとしてインターネットと同様に使うことが、まず可能です (もちろん、企業の設備ですからその上で何が許されるかはインターネットとはだいぶ違ってくるでしょうが)。たとえば電子メールによって組織内の情報流通のあり方がだいぶ変わった、というのは多くの人が知っているところだと思います。

しかし、イントラネットの場合、もう 1 つの側面として「組織メディア」があるように思います。たとえば、インターネット側からある企業のページを見た場合、そこには企業としての統一された「顔」があるわけです。一方、イントラネット側から見た場合、その統一された「顔」やその他組織の活動全般がネットワークによってサポートされ、全体として組織の円滑な運営を可能にしていく、という面があり、これはインターネットにおける「個人メディア」とはだいぶ性格を異にしていると思うわけです。

たとえば、グループウェア (ロータスノーツや Microsoft Outlook など) は既に多くの企業で取り入れられていますが、今後さらに進んだ企業では自社の情報流通や企業文化に適合した、専用の/カスタマイズされた情報システムをイントラネット上で稼働させていくようになると考えます。これらが旧来の「企業情報システム」と違うのは、単に特定の仕事 (会計管理、人事管理、情報分析など) 用のシステムとは異なり、人間のおこなう企業活動をトータルでサポートする有機的なシステムになるだろう、ということです。

そして、そのような組織メディアの実現手段としては、ここまでにお話してきたような「動的コンテンツ」が最も必要とされる技術になるのではないかと思います。そして、個人でできる動的コンテンツ開発に比べて、組織として投入できる資源はずっと多いので、より高度なシステムが作られる可能性も大きいでしょう。ただし、だからといって個人の手による動的コンテンツがつまらない、ということは全くありません。それは、ネットワークには「深さ」と「広さ」の両方が必要だと思うからです。

では、本日はどうもありがとうございました。