marímba

Marimba's Castanet

Automatic Software Deployment over the Internet

Douglas Donaldson & Mike Bursell

APM Ltd

14 October 1997

Setting the Scene

- There are general unsolved problems with deploying internet applications
  - How to manage the installation?
  - How to manage the upgrades?
    - IT Support Staff are expensive
  - How to start up the distributed services and clients?
  - How to keep information secure and services available
  - How to cope when the network is unavailable?
  - How to monitor progress?
- Push Technology could solve some of these

Limitations of Pull Technology

- Web browsers ‘Pull’ information and applications off the network
- The user has to search for up-to-date information or software upgrades
- The providers have to rely on users returning to their sites
- The administrator has a scalability problem in managing software upgrades

The Potential of Push Technology

- Up-to-date **information** and **applications** are ‘Pushed’ to the user automatically
- The network can be disconnected afterwards
- The user subscribes to information s/he’s interested in - no more searching
- The providers don’t have to shrinkwrap software, or rely on users returning to their sites
- The administrator doesn’t have to manage software upgrades

Marimba's Push Technology - Castanet

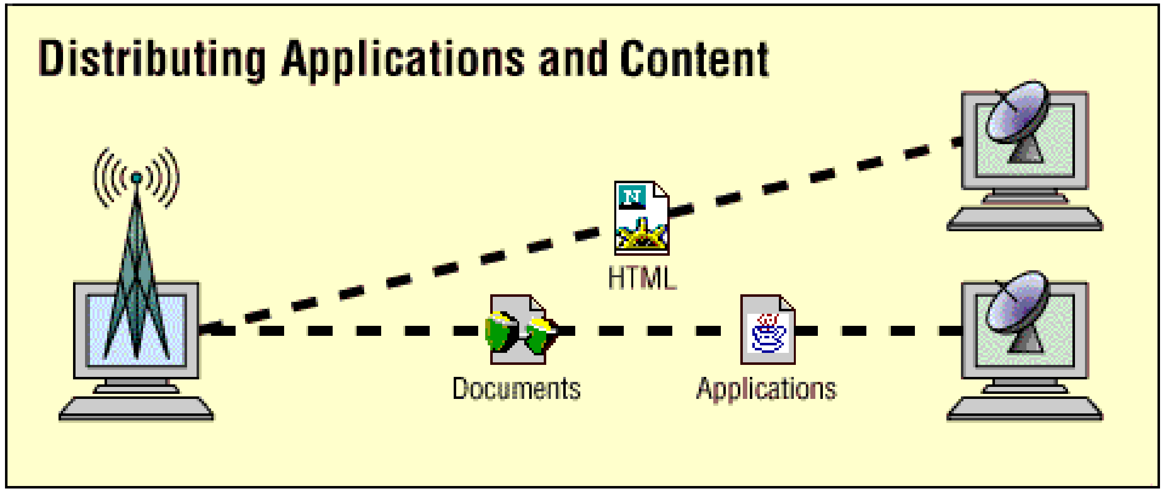
marimba Castanet

- "A system for distributing, installing and updating software and content over intranets and the Internet"
  - Castanet documentation
- "We were envisioning automatic seamless deployment of control applications, word processors and financial portfolio management applications that a company such as Schwab or Lehman Brothers might ship. We weren't thinking about news headlines and sports scores."
  - Kim Polese, co-founder, Marimba. VARBusiness Interview, July 15 1997. http://techweb.cmp.com/vb/july/172pqa.htm

Marimba's Claims for Castanet

- State of the Art application management system
- Reduces the cost of corporate ownership of PCs
- Supports disconnected portables and dialup PCs
- Uses bandwidth economically
- Highly scalable
- Eases automation of business processes
- Allows new customers to be reached, and existing customer relationships to be nurtured

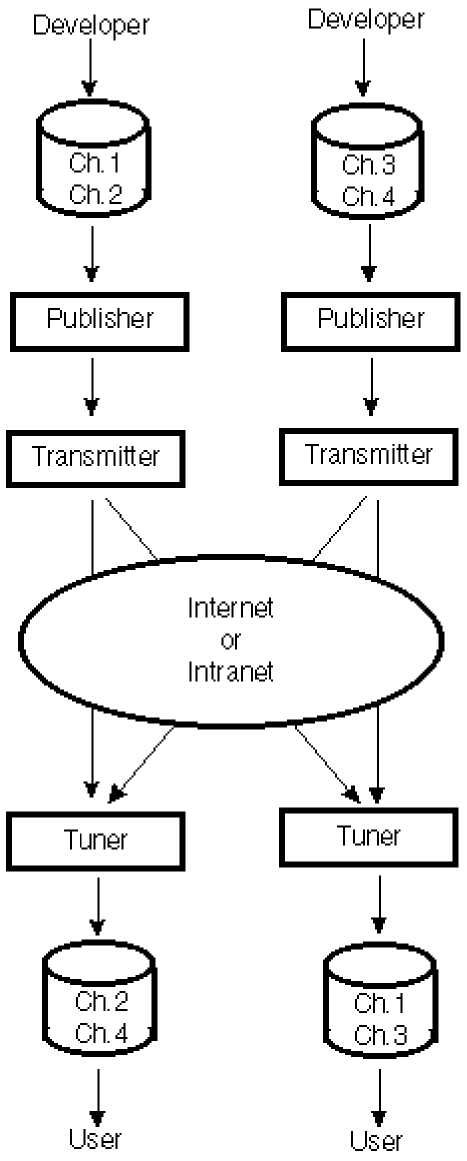
Castanet's Main Components



- *Channel* - An application or data to be distributed
- *Transmitter* - A network service which maintains and broadcasts Channels on request
- *Tuner* - A User's application which installs, receives and monitors Channels

Application LifeCycle

- A Developer uses programming or authoring tools to write a Channel (a program or HTML)
- A Publisher program *publishes* (copies) the files to a Transmitter
- A User *subscribes* to a Channel (downloads and installs it) using a Tuner
- The Tuner keeps the Channel current using its *schedule*



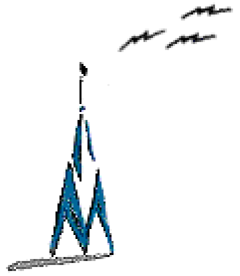
Channels

- An *HTML Channel* is a Website
  - The Tuner launches subscribed HTML channels by passing them to Web Browsers
  - A *Log* of usage can be sent back to the Transmitter for analysis
- An *Executable Channel* is a 100% Pure Java program
  - The Java applet security model is used…
    - …except each Channel can use one local directory for file IO (scratch space), allowing for persistent data

Publisher

- A Program to copy a Channel’s files to a Transmitter
  - The developer supplies a description of the Channel (HTML, Executable, etc.)
  - The developer supplies a desired update schedule (how the Tuner should install updates)
    - A User of the Tuner can override this schedule
  - The Transmitter can be remote from the site where the developer runs the Publisher
  - *Atomic* and *Differential Update* is used to revise a Channel

Transmitters

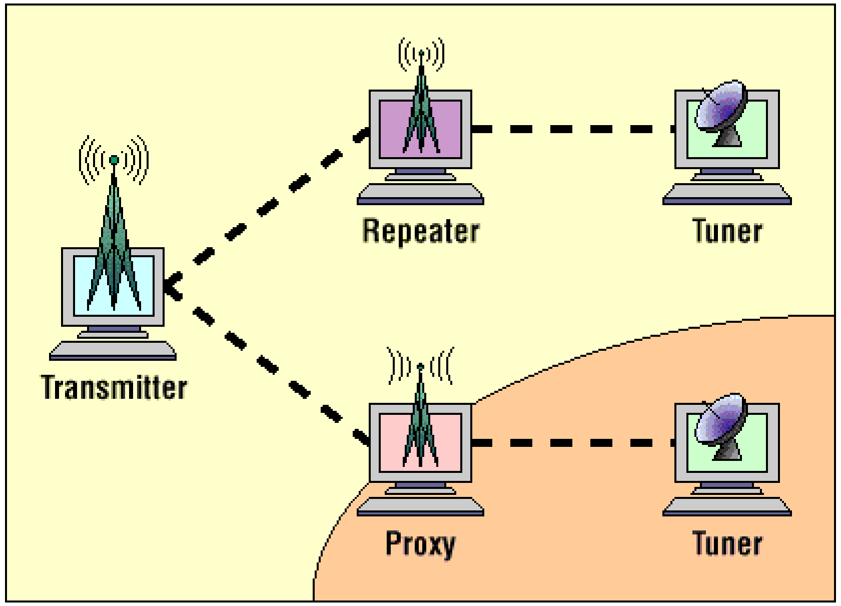


- Maintain a list of Channel files it serves
  - Similar to the way a Web server serves HTML files
    - but with more efficient network connections
- Users can use Tuners (inc. Web browsers) to obtain the list and subscribe to Channels
- Runs on a dedicated network port
  - Easy to put behind a firewall
    - A *Gateway* allows a Transmitter to share a port with a Web server

Tuners

- Subscribe to new Channels, downloading and installing them - *Intelligent Pull*
- *Atomic* and *Differential Update* of subscribed Channels
  - automatically (according to a schedule) or manually
- Launch and monitor the execution of Channels
- The Tuner is self-updating
  - a distinguished Channel
- *Netcaster* is a Tuner integrated into Netscape

Bells and Whistles



- Repeating Transmitters
  - distribute the load of Transmitting a Channel
  - the primary transmitter assigns a Tuner to itself or one of its Repeaters
- Proxies
  - cache channel files for a collection of tuners (like Web proxies)
- Plugins
  - An optional Channel component residing with the Transmitter
    - Analyse feedback data
    - Customise Channel files

Strengths

- The *Atomic* and *Differential Update* of programs and information is A Good Thing
- Ease of installation of updates, especially fixes
- Applications can be developed with an intelligence regarding network availability
  - If the net is available, use it to keep corporate data
  - When offline (e.g. mobile), save data for automatic update later
- Plugins allow customisable Channel behaviour

Problems and Limitations

- Upgrading applications can be disruptive
  - Publisher controls versioning of programs
  - The user is unable to recover an earlier version *
- No support for user authentication
  - Is the user allowed to access this Channel?
  - Has s/he paid for the upgrade?
  - Guardian may address this
- Applets have limited network and file access
- Channel content is limited *
- Channels are held in the Tuner's database *

Castanet for Application Deployment?

- This is a simple broadcast model
  - The technology could form a component of a more general application deployment strategy
  - Upgrade of servers, and of a collection of distributed components needs consideration
- How should control of updates be enforced?
  - In the Channel’s published schedules?
  - In the Tuner?
  - By an intermediate system administrator?
- If the net is highly available, this intelligent caching is not needed...

Is Push Technology new?

- Automated mirroring
  - The mechanism has strong technical similarities
  - Doesn’t automate software deployment and upgrade
- Distributed File Systems (e.g. Coda, LotusNotes)
  - Intelligently cache networked files on a user’s machine
  - Attempt to resolve conflicts when machines are rejoined to the network

Product Comparisons 1

- Most Push Technology products aim to deliver news to the desktop:
- PointCast
  - Information only (e.g. News, Sports, Weather)
- BackWeb
  - (Claims to) deliver arbitrary content
- Diffusion's IntraExpress
  - Controls delivery of corporate information via multiple communication mechanisms
    - email, webcasting, fax, pager, and hard copy

Product Comparisons 2

- Intermind’s Communicator
  - Delivers customised Web sites
- Novadigm
  - Resource management with differential update keeping target machines configured to a ‘desired state’
  - Pending patent infringement lawsuit with Marimba

What the Demo will show

- 0. Channel publication and subscription
- Dynamic update of a program’s code and data
  - 1. restart when updated code arrives
  - 2. display updated data
- 3. Different applet behaviour according to user preferences
- 4. Java applets storing persistent data on the client host