Peer-to-Peer Architectures and the Magi Open-Source Infrastructure

Richard N. Taylor
Institute for Software Research
University of California, Irvine
http://www.ics.uci.edu/~taylor

Peer-to-Peer

- Autonomous hosts interacting as equals
- Individuals have their own (unique) abilities
- Individuals benefit from services available from their peers
- "Network effect" increases value

Enablers

- Network connectivity
- Bandwidth
- Processor/memory capacity

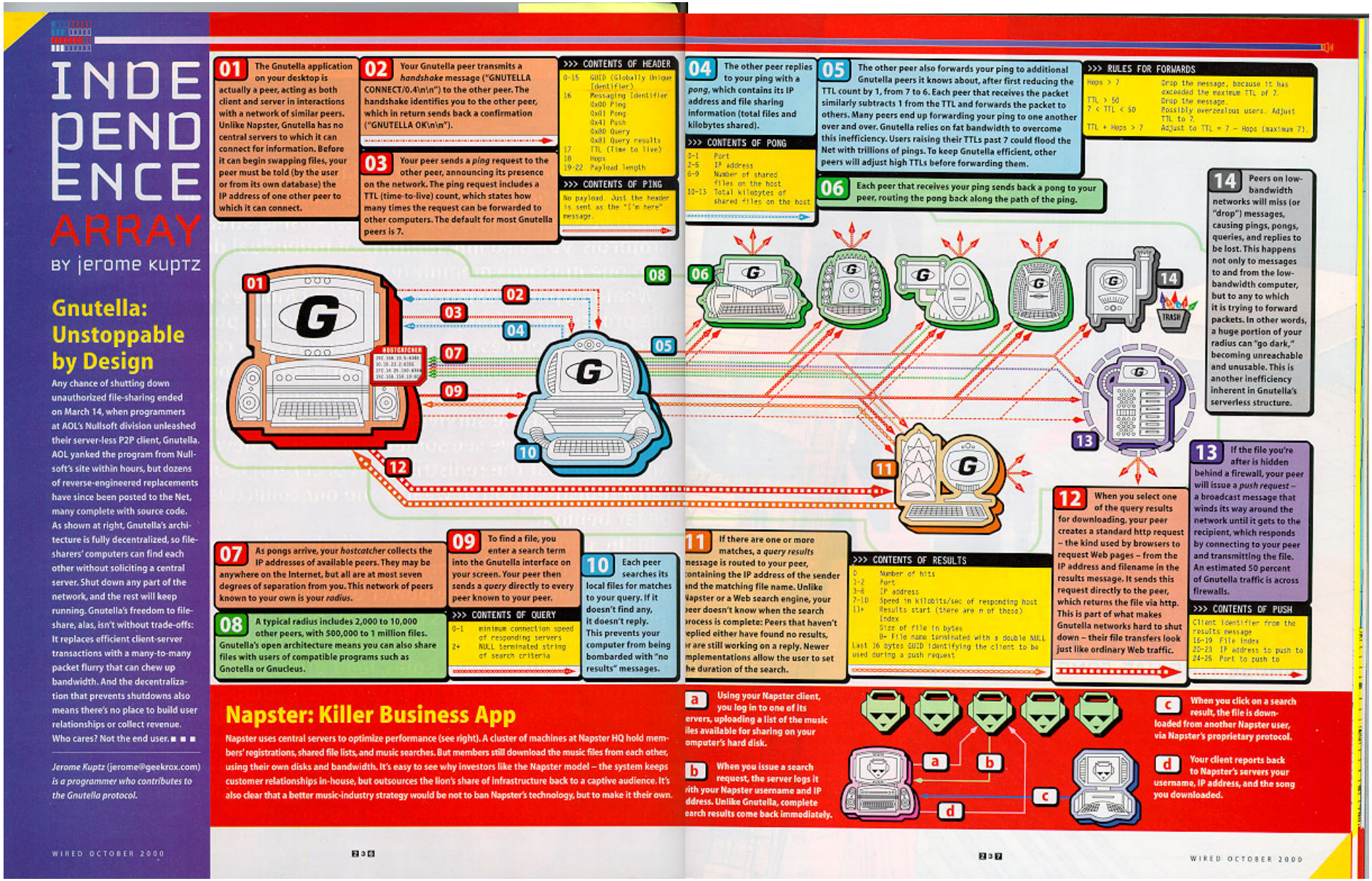
Usages

- File sharing
- Field service repair dispatch
- Cooperative work
- "Home security"
- Virtual communities
- Event-notification applications

Napster

- File sharing: mp3's
- Peers hold the files
- Napster Inc's servers hold catalog and broker relationships
  - You upload your IP address, music you have, and requests
  - You receive locations where requests can be satisfied
- File transfer is p2p, using proprietary protocol

Gnutella, as seen by "Wired"



INDEPENDENCE ARRAY

BY jerome kuptz

Gnutella: Unstoppable by Design

Any chance of shutting down unauthorized file-sharing ended on March 14, when programmers at AOL's Nullsoft division unleashed their server-less P2P client, Gnutella. AOL yanked the program from Nullsoft's site within hours, but dozens of reverse-engineered replacements have since been posted to the Net, many complete with source code. As shown at right, Gnutella's architecture is fully decentralized, so file-sharers' computers can find each other without soliciting a central server. Shut down any part of the network, and the rest will keep running. Gnutella's freedom to file-share, alas, isn't without trade-offs: It replaces efficient client-server transactions with a many-to-many packet flurry that can chew up bandwidth. And the decentralization that prevents shutdowns also means there's no place to build user relationships or collect revenue. Who cares? Not the end user. ■ ■ ■

Jerome Kuptz (jerome@geekrox.com) *is a programmer who contributes to the Gnutella protocol.*

01 The Gnutella application on your desktop is actually a peer, acting as both client and server in interactions with a network of similar peers. Unlike Napster, Gnutella has no central servers to which it can connect for information. Before it can begin swapping files, your peer must be told (by the user or from its own database) the IP address of one other peer to which it can connect.

02 Your Gnutella peer transmits a *handshake* message ("GNUTELLA CONNECT/0.4\n\n") to the other peer. The handshake identifies you to the other peer, which in return sends back a confirmation ("GNUTELLA OK\n\n").

03 Your peer sends a *ping* request to the other peer, announcing its presence on the network. The ping request includes a TTL (time-to-live) count, which states how many times the request can be forwarded to other computers. The default for most Gnutella peers is 7.

>>> CONTENTS OF HEADER

0-15	GUID (Globally Unique Identifier)
16	Messaging Identifier 0x00 Ping 0x01 Pong 0x41 Push 0x80 Query 0x81 Query results
17	TTL (Time to live)
18	Hops
19-22	Payload length

>>> CONTENTS OF PING

No payload. Just the header is sent as the "I'm here" message.

04 The other peer replies to your ping with a *pong*, which contains its IP address and file sharing information (total files and kilobytes shared).

>>> CONTENTS OF PONG

0-1	Port
2-5	IP address
6-9	Number of shared files on the host
10-13	Total kilobytes of shared files on the host

05 The other peer also forwards your ping to additional Gnutella peers it knows about, after first reducing the TTL count by 1, from 7 to 6. Each peer that receives the packet similarly subtracts 1 from the TTL and forwards the packet to others. Many peers end up forwarding your ping to one another over and over. Gnutella relies on fat bandwidth to overcome this inefficiency. Users raising their TTLs past 7 could flood the Net with trillions of pings. To keep Gnutella efficient, other peers will adjust high TTLs before forwarding them.

>>> RULES FOR FORWARDS

Hops > 7	Drop the message, because it has exceeded the maximum TTL of 7.
TTL > 50	Drop the message.
7 < TTL < 50	Possibly overzealous users. Adjust TTL to 7.
TTL + Hops > 7	Adjust to TTL = 7 − Hops (maximum 7).

06 Each peer that receives your ping sends back a pong to your peer, routing the pong back along the path of the ping.

07 As pongs arrive, your *hostcatcher* collects the IP addresses of available peers. They may be anywhere on the Internet, but all are at most seven degrees of separation from you. This network of peers known to your own is your *radius*.

08 A typical radius includes 2,000 to 10,000 other peers, with 500,000 to 1 million files. Gnutella's open architecture means you can also share files with users of compatible programs such as Gnotella or Gnucleus.

09 To find a file, you enter a search term into the Gnutella interface on your screen. Your peer then sends a *query* directly to every peer known to your peer.

>>> CONTENTS OF QUERY

0-1	minimum connection speed of responding servers
2+	NULL terminated string of search criteria

10 Each peer searches its local files for matches to your query. If it doesn't find any, it doesn't reply. This prevents your computer from being bombarded with "no results" messages.

11 If there are one or more matches, a *query results* message is routed to your peer, containing the IP address of the sender and the matching file name. Unlike Napster or a Web search engine, your peer doesn't know when the search process is complete: Peers that haven't replied either have found no results, or are still working on a reply. Newer implementations allow the user to set the duration of the search.

>>> CONTENTS OF RESULTS

0	Number of hits
1-2	Port
3-6	IP address
7-10	Speed in kilobits/sec of responding host
11+	Results start (there are n of these) Index Size of file in bytes 0+ File name terminated with a double NULL
Last 16 bytes	GUID identifying the client to be used during a push request

12 When you select one of the query results for downloading, your peer creates a standard http request – the kind used by browsers to request Web pages – from the IP address and filename in the results message. It sends this request directly to the peer, which returns the file via http. This is part of what makes Gnutella networks hard to shut down – their file transfers look just like ordinary Web traffic.

13 If the file you're after is hidden behind a firewall, your peer will issue a *push request* – a broadcast message that winds its way around the network until it gets to the recipient, which responds by connecting to your peer and transmitting the file. An estimated 50 percent of Gnutella traffic is across firewalls.

>>> CONTENTS OF PUSH

	Client identifier from the results message
16-19	File index
20-23	IP address to push to
24-26	Port to push to

14 Peers on low-bandwidth networks will miss (or "drop") messages, causing pings, pongs, queries, and replies to be lost. This happens not only to messages to and from the low-bandwidth computer, but to any to which it is trying to forward packets. In other words, a huge portion of your radius can "go dark," becoming unreachable and unusable. This is another inefficiency inherent in Gnutella's serverless structure.

Napster: Killer Business App

Napster uses central servers to optimize performance (see right). A cluster of machines at Napster HQ hold members' registrations, shared file lists, and music searches. But members still download the music files from each other, using their own disks and bandwidth. It's easy to see why investors like the Napster model – the system keeps customer relationships in-house, but outsources the lion's share of infrastructure back to a captive audience. It's also clear that a better music-industry strategy would be not to ban Napster's technology, but to make it their own.

a Using your Napster client, you log in to one of its servers, uploading a list of the music files available for sharing on your computer's hard disk.

b When you issue a search request, the server logs it with your Napster username and IP address. Unlike Gnutella, complete search results come back immediately.

c When you click on a search result, the file is downloaded from another Napster user, via Napster's proprietary protocol.

d Your client reports back to Napster's servers your username, IP address, and the song you downloaded.

Gnutella: just file sharing

- No servers with catalogs
- Pings the net to locate Gnutella friends
- File request broadcast to friends
- When provider located, file transferred via HTTP
  - Initial interactions were via Gnutella protocol

Groove (a.k.a. "Notester")

- www.groove.net
- Shared workspace (groupware support)
- WYSIWIS support
- A platform for development
- File replication on every peer; XML
- Closed, proprietary protocol
- Secure communication and storage
- MS platform dependent; COM usage

Data Storage and Persistence

- Persistence of a shared space is captured in an XML document database
- A copy of the shared space document DB is stored on each member's device
- Each Tool stores persistent data within unique tool document
- Tool initiated changes ("Deltas") are disseminated to each member on remote device

Groove disconnect/update

All Groove members are online, and pass changes directly to each other.

One member is disconnected, and stores changes locally. Other members send changes to the Groove relay service.

When the member reconnects, the relay service sends changes to the member and empties the queue.

Key Challenges & Characteristics (1)

- Distribution
  - Designers must deal with all the issues of distributed networked applications, including independent namespaces, synchronization, locating information, unreliability, security, latency, …
- Heterogeneity
  - Create a new layer of virtual machine?
  - Embedded devices
    - Differ by order(s?) of magnitude in processing power, communication bandwidth, memory, power reserves, persistence of connectivity

Characteristics (2)

- Mobility and intermittency
  - On-line/off-line; varying IP addresses
- Decentralized control
  - No common administrative structure
  - Trust, security, unreliability, failure, non-repeatability
- Incomplete information
  - Inconsistent information
  - Latency

Characteristics (3)

- Sharing, Coordination, and Cooperative Work
  - Distributed computation & data storage
  - Distributed content
  - Distributed relationships
    - Documents in relationship to each other, to tasks, to people
    - Time varying
  - Distributed activities

Characteristics (4)

- Emergent behavior
  - E.g. self-selection to provide services to a group; self-caching to reduce burden on peers
- Scalability
  - Across numbers of devices
  - Across device capabilities
- Security
  - Authentication
  - Authorization
  - Encryption
- Ubiquity

Magi: The Magi Design Decisions

1. Build atop the Web's infrastructure
2. Provide a platform for others
3. Exploit an asynchronous, event-based, component architecture
4. Promote "the independence of Peers"

Build atop the Web's infrastructure

- Why?
  - Utility, scalability, extensibility, performance, adoption (ubiquity)
- What
  - HTTP/1.1 — communication protocol
  - WebDAV — collaboration and annotation
  - URI — naming and location of resources
  - MIME — resource representations

HTTP/1.1

- Open protocol: anyone's implementation OK
- Standard semantics and defined, on-the-wire syntax
  - Enables value-adding intermediaries, such as cacheing and proxying
- Wide adoption

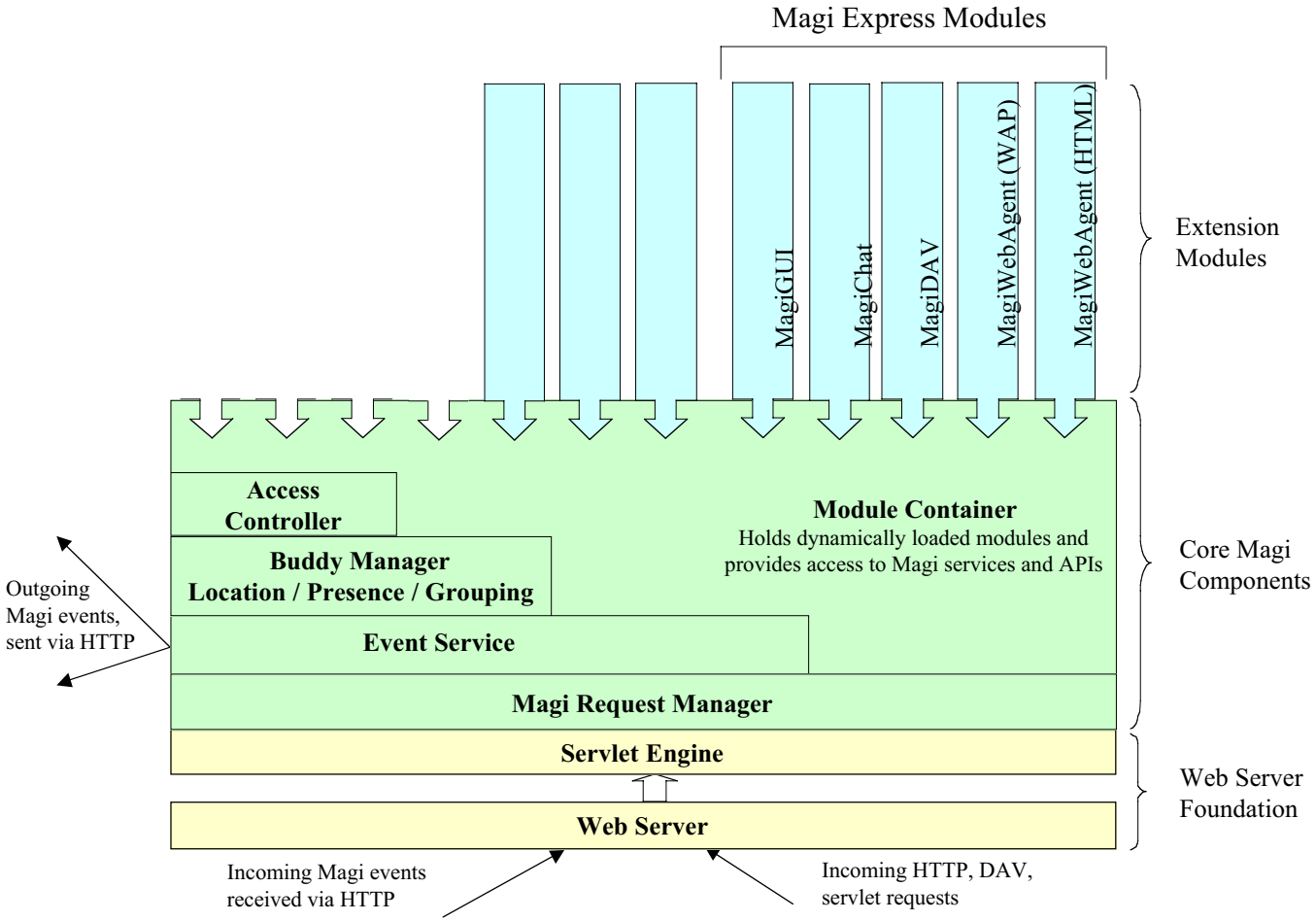
Platform for others

- No user interface constraint
- Open standards
- Open source components
- Platform-independent architecture with multiple implementations

Architectural overview of Magi

- A canonical peer
- Network architecture
- Security and authorization

A canonical peer



Magi peers

- Base infrastructure + plug-ins
- Base: Web server w/ servelet engine
  - Simple parsing of HTTP request
- Request manager invokes services based on examination of requests
- Event service: invokes services based upon their registration of interest in events, and receipt of those events from the request mgr.

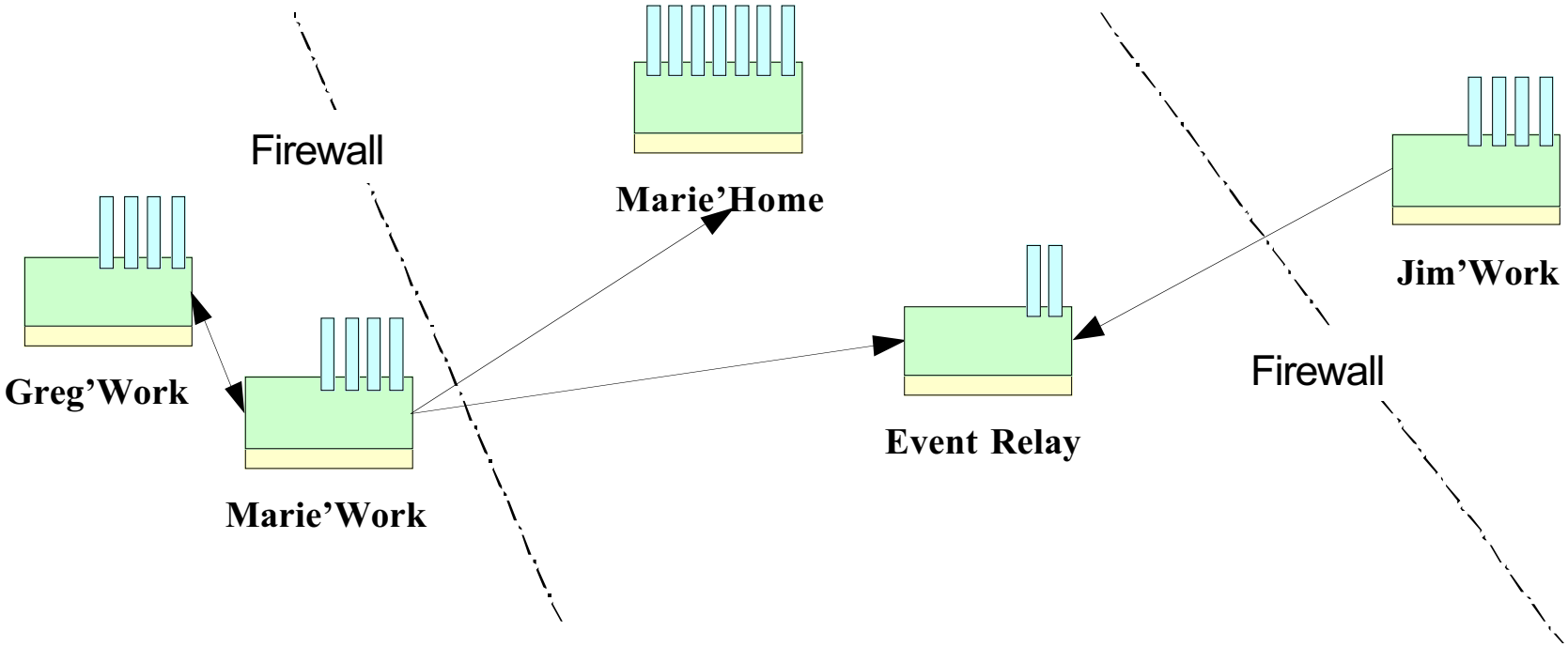
Magi peer, continued

- Buddy manager
  - Tracks location of buddies and their devices
  - On-line/off-line status tracking
- Access manager
- "Module container"

Networks

- Discovery: who's there?
  - Follow a path of ease/efficiency
    - Magi DNS, if it exists
    - Known peers, if they exist
    - Gnutella-like discovery
- Presence: opening lines of communication

Firewalls



Questions

- What distinguishes peer-to-peer from other architectures?
- When is it appropriate to use a p2p architecture?
- How do you design a p2p application?
- What tools should you have at your disposal?

Credits and Contacts

- http://www.endtech.com/html/index.html/
- http://www.magisoft.net/html/index.html
- http://conferences.oreilly.com/p2p/
- Greg Bolcer, Michael Gorlick, Arthur Hitomi, Peter Kammer, Brian Morrow, Peyman Oreizy