



Research Center

NRC-TR-2006-005

Providing HTTP Access to Web Servers Running on Mobile Phones

Johan Wikman and Ferenc Dosa

Nokia Research Center Helsinki

<http://research.nokia.com>

May 24, 2006

Abstract:

Mobile phones are powerful enough to host web servers, but HTTP access to them is currently impracticable in most operator IP networks. We present a demo system providing operator independent HTTP access to web servers running on mobile phones for anyone browsing the Internet. A general overview of problems to solve, our implementation, and possible future directions are also discussed in this paper.

Index Terms:

mobile web server

mobile website

mobsite

HTTP

1 INTRODUCTION

While it has been possible for quite some time to access webpages on the Internet from mobile phones, reaching web servers on mobile phones has been neither possible in the general case nor in demand. First of all, almost all people perceive the mobile phone solely in the role of a service consumer, and not a service provider, for instance [1] [15] [12], while applications where the mobile phone is a content provider are almost non-existing [2] [10]. Consequently, operators, intent on protecting and optimizing their networks, neither allow autonomous downlink traffic (from Internet to phone) nor optimize for significant uplink traffic (from phone to Internet) today.

With the ever spreading adoption of Service Oriented Architecture [7] [17] [16], solutions for desktop applications involving locally run web services, and even special purpose electronic appliances incorporating web servers [11], we expect that HTTP access to mobile phones will soon be essential service to mobile users. Furthermore, these devices are becoming ever more powerful in terms of processing and storage capacity, and have already bypassed those of the early web servers when the web was young.

A necessary first step towards such a goal is putting a working HTTP server implementation on the mobile phone. This readily makes webpages and web services available to web browsers accessing mobile content locally or via some proximity radio technology such as Bluetooth, WLAN, etc. However, for ubiquitous and unified access, a web server should be reachable via cellular IP from the public Internet, a service generally not available in most operator networks.

This paper discusses obstacles in today's operator networks one has to overcome when providing connectivity to mobile web servers, describes a solution we have developed at Nokia Research Center, and finally suggests possible future directions.

2 OPERATOR NETWORKS

In most operator networks, an HTTP request originating from the public Internet cannot reach a web server that is running on a phone inside the operator network for several reasons.

1. Operator firewalls are usually configured to prevent all traffic that is not initiated from inside the operator network. This is a simple measure to make operator IP networks resistant to most forms of attacks, with the assumption that traffic initiated only from inside is legal, and from outside only responses are allowed. This means that an HTTP request cannot reach a mobile web server inside most operator networks.
2. While TCP connectivity is supported in operator networks, it usually has some special traits that makes it subtly different to conventional — wired — TCP connections. In addition to being initiated only from inside the operator network, they are prone to become *stale*¹. Problems induced by such behaviour do not manifest themselves in current use of cellular IP networks, for instance, during web browsing, as HTTP requests sent from inside are quickly responded to by Internet web servers. Therefore a solution is needed to provide true always-on connectivity in an environment that is currently not used and therefore not tuned for traffic patterns involving prompt responses to asynchronous requests from outside.
3. Mobile phones usually have dynamically allocated IP addresses, since current applications using cellular IP involve

¹A cellular TCP connection which is idle for a certain, operator specific, period of time will become blocked for the external party and only a message sent from the internal party can revive the TCP connection.

mobile phones as clients, where the IP address of the client hardly matters. There are operators providing services with fixed dynamic addresses, but such support is far from general, and certainly not scalable. Without a solution to this problem, there is no user friendly way to determine the actual IP address of a mobile web server, hence it cannot be easily reached.

4. Network Address Translation (NAT) is commonly used for security reasons as well as to alleviate IPv4 address depletion problem, preventing an HTTP request from outside the operator network to find the mobile web server.

Among these, there are two closely related but different types of obstacles: accessibility and addressability. The first two are related to accessibility, as in those cases the traffic itself is either impossible or breaks down. While addressability assumes accessibility solved, it is a subtly different problem that makes a system impracticable; even if traffic itself is possible, a user must have an easy way to address the mobile web server. For instance, forcing people to figure out and manually type in dynamic IP addresses is not viable option.

An additional question is that of latency and speed of operator IP networks, and the fact that these are nowhere near to what one expects when browsing to powerful servers over the wired Internet. However, this seeming showstopper argument stems from a wrong — although natural — assumption, namely, that mobile web servers are nothing but web servers, and that the very same content types would be served from a mobile phone as served from stationary web servers. In fact, we envision exactly the contrary; mobile phones are not meant to host huge digital libraries, vast databases to support e-business, popular portals, or news sites. Mobile web servers will be best performing as sources of dynamic, context-dependent, personal data streams (webapplications and web services). While some of these will undoubtedly be of multimedia types (audio, video, image), textual and XML-based data will play a significant role, where speed does not matter that much².

Also, while a system could easily be built where response latency can be reduced significantly by uploading/synchronizing mobile data with content servers, such solution would simply trade low latency for scalability inherent to the distributed datasource system represented by a mesh of individual mobile web servers, freshness of mobile data, and potential applications involving on-demand content generation.

Furthermore, anticipated increase in cellular IP throughput, web caches, and separation between static/nonpersonal³ and dynamic/personal⁴ data are all expected to help closing the gap between wireless and wired web experience. Finally, operators will optimize their networks for significant uplink traffic once it is demanded widely, just like they do it for current traffic patterns today.

3 SOLUTION ASPECTS

When setting out to create HTTP connectivity to mobile web servers, we have determined a number of important goals an ideal solution should meet.

- The solution must not invoke any changes in browsers. Meeting this goal ensures that all existing browsers can be used to access pages, webapplications originated from a handset, and existing web services client frameworks can be readily put into use in the mobile context as well.

²For a somewhat more in dept presentation of mobile websites, please refer to [3].

³For example style sheets, graphics, and other artefacts that can be stored anywhere.

⁴Dynamic or personal content are best served from their ultimate source, the mobile phone.

- The HTTP connectivity must be transparent to the HTTP server, that is, it should not know whether an HTTP request came from a local web browser, a nearby device via some proximity radio technology, or from a remote client via the cellular.
- The solution ideally should necessitate as little operator involvement as possible, and should be independent of any particular operator network.

We have ported Apache httpd [4] to the S60 platform in order to obtain a full-fledged web server implementation for the mobile phone. The internet gateway with the task of providing HTTP accessibility and addressability is injected into the Tomcat server [5] in our demo implementation.

4 OVERVIEW OF CONNECTIVITY

HTTP is designed to allow intermediaries — proxies, gateways, and tunnels — between the client and the origin server [6]. The targeted URL and HTTP header ‘Host’ are the two most important data in an HTTP request that allow an intermediary to determine where to send an HTTP request forward.

The internet gateway, as shown in Figure 1, providing the HTTP connectivity to the mobile phone is simply a *gateway*⁵ in HTTP/1.1, or *reverse proxy* in Apache-httpd parlance.

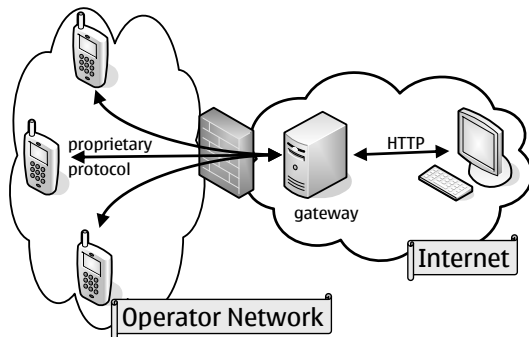


Figure 1: Overview of HTTP connectivity.

The communication between a browser/webclient and the gateway is unmodified HTTP. The problems of addressability boils down to what type of URL the webclient needs to use and how the internet gateway dispatches an incoming HTTP request to the intended mobile phone. The proprietary protocol between the gateway and the mobile phone has to overcome the accessibility problems listed in section 2.

5 ADDRESSABILITY

Let us assume for a moment that the accessibility is solved, and that the internet gateway can forward HTTP traffic to targeted phones any time, receive the response and relay that back to the client where the original request came from. If the owner of the mobile phone running the web server is called John Doe and the domain

⁵Gateway is defined in the HTTP/1.1 standard as “a server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway”.

name of the internet gateway is `www.at.openlaboratory.net`, the conventional URL to represent John’s mobile web server is

`http://www.at.openlaboratory.net/~john.doe/`.

This enables the gateway to decide which mobile web server to dispatch the request to. However, there is a more expressive way to convey that a personal mobile website is being targeted by making the personal mobile URL look like

`http://john.doe.at.openlaboratory.net/`.

It turns out that achieving this is not difficult at all. The owner of the subdomain `.at.openlaboratory.net` simply has to configure her DNS server to resolve all URLs in the general form of

`*.at.openlaboratory.net`

to point to the IP address of the internet gateway, so that any browser trying to locate web contents with URLs like `www.at.openlaboratory.net`, `john.doe.at.openlaboratory.net`, etc. will, in all cases, end up with the IP address of the internet gateway. The gateway, having received an HTTP request, looks at the targeted domain in the ‘Host’ HTTP header and in the request method line of HTTP request, and performs dispatch based on that.

Both solutions are compatible the way Internet is used and how people want to remember each other’s *mobsites*⁶; they have to remember only the domain suffix of the internet gateway in addition to remembering the name of their friends.

6 ACCESSIBILITY

The HTTP server running on the mobile phone creates a server socket and starts listening to it, just like HTTP servers generally do. In Figure 2, the arrows show the direction of connection creation. If the HTTP server were on the Internet – with no firewalls or other obstacles in between — a simple HTTP connection from the browser would suffice, and that is conceptually what we want to achieve in a transparent manner. This is accomplished by adding a level of indirection: a TCP connection between *mobile connector* and *gateway connector*.

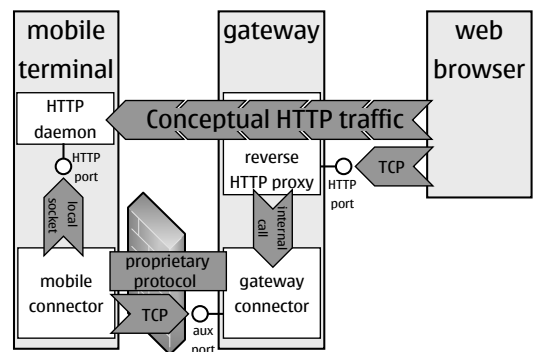


Figure 2: Protocol stack view of accessibility.

Mobile connector is a process running on the mobile phone that establishes a TCP connection to the gateway — or more accurately, to its peer component, the gateway connector, after which the mobile phone is considered *online*. This way, the connection between the mobile and gateway connector is established from inside the operator network, a legal direction. Once the link between the two connectors is alive, the gateway connector can dispatch HTTP requests any time to the mobile phone.

⁶mobile website

Whenever the mobile connector receives an HTTP request from the gateway connector, it creates locally a socket connection to the port the HTTP server is listening to. This means that from the HTTP server point of view, the mobile connector is just another HTTP client.

The protocol between the two connectors is *not* HTTP. In HTTP, the one creating the connection and sending the request is the same endpoint: the client. The proprietary protocol, on the other hand, is initiated by the mobile connector, but HTTP requests are sent by the gateway connector. Furthermore, this protocol supports more than mere HTTP traffic relaying; it can authenticate the mobile connector and keep the connection alive by a regular stream of mostly empty messages sent by the gateway connector, which is described in section 6.2.

6.1 Connection establishment

In order to be online, the owner of the mobile web server has to register on the gateway, so that it can associate between the credentials supplied by the mobile connector upon connection establishment, and the URL prefix of the user, for example `john.doe`.

Conceptually there are two types of communication channels between mobile connector and gateway connector: *control channel* and *data channel*. The TCP connection the mobile connector creates when going online is in fact the physical manifestation of the control channel - this is the essential part that is kept alive, and is used to initiate HTTP relay sessions. Whenever there is a need, the gateway connector asks the mobile connector to create data channels, which are currently implemented as separate TCP connections to the same port. These data channels can be reused for the next HTTP request.

The problem of dynamic IP address and NATs in between the two connector peers do no matter to the gateway connector, it simply associates the control channel (TCP socket connection) to the URL prefix of the owner — `john.doe`— of the mobile phone.

6.2 Keeping the connection alive

As mentioned above, the gateway connector has the responsibility to generate regular keepalive traffic by which the control channel is kept usable. If the period between two keepalive messages — or HTTP request — is less than the period after which a TCP connection would become stale, the connection will be always permeable from the Internet end.

There would have been a good reason to give the mobile connector the task of generating regular keepalive messages: according to our experience, a message sent over a stale TCP connection from inside the operator network revives the connection, and this way the protocol could have been made self-correcting naturally. On the other hand, this would have resulted a significantly more complex state machine of both endpoints of the protocol.

In the current demo implementation, once the connection is established, it's only the gateway connector that can send a message autonomously over the control channel: either a request for opening a data channel, or a dummy keepalive message if there is nothing else to send. The mobile connector only waits for messages after the initial handshake. If the gateway cannot send a keepalive message, it considers the mobile phone *offline*, and generates a "Mobsite offline" page to any web client requesting content from the mobile phone.

There is, of course, the question of how regular the keepalive messages have to be. To this end, we have defined and implemented a protocol that can employ one of the following three strategies.

- The mobile connector can tell the gateway connector not to generate a keepalive stream. This feature is not only useful when testing the system, but also allows for the keepalive

mechanism to be turned off when it is known that it is not needed, as is the case when the connection goes, for instance, over USB.

- By default, the gateway connector tries to discover a near-optimal value for the keepalive period, which is why keepalive messages of our protocol are more than empty messages: they specify when the next keepalive message will be due. If the mobile connector does not receive a keepalive message (or a data channel request) by the time the next message is due, it will consider the connectivity broken, and will try to fix it by establishing a new control channel.

Meanwhile, the gateway connector is trying to discover a sufficiently good keepalive period by attempting to use different values for it — a non-viable keepalive period is signaled by a new control channel from the mobile connector that did not receive its due keepalive message. In such case, the gateway selects the last known good value for the keepalive period. Our current implementation of the discovery algorithm tries safe but non-optimal values first (short keepalive period), and then tries incrementally pushing them towards less safe but more optimal values (longer keepalive periods). The discovery algorithm the gateway connector employs is heuristical and a likely subject to improvements, but the conceptual algorithm, its implementation, or any of its running parameters can be centrally changed without modifications to any existing mobile connectors, as they only care about is how much time to wait before trying to re-connect.

- It is also possible to have keepalive messages of fixed keepalive period without dynamic discovery by explicitly setting the keepalive parameters in the settings of the mobile connector software.

6.3 Fetching Mobile Content

An exemplary interaction of a whole browsing session is shown in Figure 3, assuming that a page is requested from from `john.doe.at.openlaboratory.net`, and John Doe's mobsite is online.

The DNS setup described in section 5 makes web clients send their HTTP requests to the gateway by returning its IP address for any domain name lookup matching the pattern `*.at.openlaboratory.net`, see messages 1 and 2.

Once a web client has sent its request (message 3), the gateway will determine the targeted mobile website by analyzing the request. Assuming it is online, the mobile connector is identified by its control channel (open TCP socket connection), and is requested to create a data channel for relaying HTTP traffic (message 4.1). Once the requested data channel is established (message 4.2), the original HTTP request is sent over it by the gateway connector to the mobile connector (message 5).

The mobile connector obtains the HTTP response locally (message 7) from the web server by forwarding the request to it (message 6). Finally, the response will travel back to the web client via the gateway (messages 8, 9, 10).

Once a HTTP response was fully read by the gateway connector, the data channel that conveyed it becomes free for reuse. The majority of webpages require multiple HTTP requests to render the whole page, as they tend to contain references to further web artefacts: style sheets, javascripts, images, animations, Java applets, etc. Therefore further roundtrips are somewhat speedier: new data channels are created only when needed.

Data channels are not kept alive, they are disposed the moment the keepalive period elapses. Since this time is usually around 5

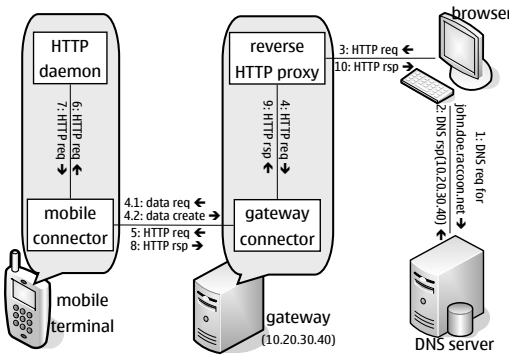


Figure 3: Collaborations of fetching mobile content.

minutes⁷, most browsing sessions do not necessarily need new data channel creation, as old idle ones may be available. Obviously, whenever a data channel is used, it becomes fully revived and the count down timer for disposal can be reset.

The job of maintaining a communication channel to the mobile phone is essentially done by an extra networking layer between TCP and HTTP: the connector protocol.

7 FUTURE RESEARCH

While the current demo implementation is already capable of providing connectivity service for a fair amount of mobile users — fellow researchers, early adopters —, there are numerous issues to settle before it can be of any real value for the wide public.

7.1 Scalability

Our current implementation is a concept demo, and thus scalability was not a matter of importance during development. However, the gateway is inherently a bottleneck of scalability for several reasons.

First of all, the number of simultaneous socket connections is limited; for instance, in a Unix/Linux environment, the number of file descriptors that can be opened at the same time may be huge, but bounded nevertheless⁸. Since our solution requires at least one TCP socket connection for each online mobsite, an industrial implementation would need to cope with hundreds of thousands of connections, or more.

This problem can be overcome by several orthogonal strategies. One straightforward idea is to create a farm of gateway connectors hosted by different machines each. In this case, the inprocess call between reverse HTTP proxy and gateway connector components of the gateway (cf. Figure 2) will still be internal ‘call’, but it will happen between machines over a local network as simple HTTP requests.

Another strategy to reduce socket depletion of the gateway is to make the mobile connector catch, and interpret special SMS-es with *kick-alive* semantics⁹. This means that an owner of a mobile phone

⁷In the discovery algorithm there is a cut off point at which the probing is stopped. 5 minutes is a long enough interval so as not to significantly affect the battery consumption while still allowing connection breakages to be noticed reasonably fast.

⁸On the GNU/Linux installation we use for running the gateway with default kernel configuration, the maximum number of file descriptors opened by a process is 1024, see also [14].

⁹On the Symbian platform, it is possible to register application specific SMS types. Messages matching such a type will be dispatched to the regis-

ter can be *conceptually* online and yet have no physical TCP connection at all, and whenever there is web client requesting, the gateway connector can force the mobsite to be *physically* online by signaling the need for control channel with an SMS addressed to the mobile connector. After a suitable idle period, the physical online state can be reverted to conceptually online state by closing the control channel.

A second problem to gateway scalability is the fact that fetching a mobile page takes significantly longer compared with serving it locally or from a local network, simply because of the latency and speed of cellular IP. Our current solution works inside the Tomcat server, and therefore has the paradigm of associating one thread to the whole request-response roundtrip. This implies as many times more threads to support the same throughput as many times it takes to wait for a content to come over the cellular connection. Consequently, a scalable gateway implementation should not be written with one-to-one mapping between threads and request-response roundtrips, but should rather have a single thread using one of the scalable poll idioms [8] [9].

7.2 Security

One concern about having a web server on a mobile phone is whether it will be a target for the kind of attacks stationary web servers are usually target of, such as Denial Of Service, IP spoofing attacks, HTTP messages exploiting known bugs, and theft of private content, to name a few.

While the primary function of the gateway described in this paper is to provide connectivity in current operator networks, it can be used to fight these problems as well. Indeed, corporates and news sites employ web gateways — or reverse proxies — not only for load balancing, but in order to be able to secure their private networks by controlling IP traffic. Similarly, a gateway can be put to the same task. In fact, if operators were to allow HTTP access to mobile phones, they would be likely to do so via web gateways controlling access and cost; the only practical difference between an ordinary web gateway as part of an operator firewall and a gateway is that the latter employs a proprietary protocol instead of simple HTTP proxying of the former, because we set out to implement HTTP connectivity without any operator involvement.

Finally, since web servers on the mobile phone cannot be reached unless they go online first, a control method checking the version of an HTTP server implementation on the phone can be easily implemented, and old versions with known security bugs could be automatically, or with minimal user interaction, upgraded.

7.3 Cost control

Another concern of having a web server on the mobile phone is whether it will cause huge phone bills. One of our assumptions is that operators will switch to flat-fee payment models — a trend already perceptible. With flat fee rates, the only type of cost is reaching the monthly limit of byte transfer after which the owner has to pay extra fee.

The best place to control cost is at the gateway, since an HTTP request does not yet generate cost for the owner of the mobsite at that point. It is possible to build a system where mobsite owners can ration their monthly byte limit for different uses, like certain amount for private, family, friends, colleagues, and the public Internet. The only prerequisite to this is to have an access control in place where the gateway plays an active role, that is, a scheme where the credentials of the web client and access rights to mobile content are managed by the gateway transparently. While such a mobile access control mechanism assisted by the gateway sounds like an extra requirement, it is, in fact, necessary by itself: an HTTP

requester application without any user interaction.

request challenged by the mobile web server instead of the gateway will have already generated cellular cost to the owner regardless of whether it can pass the authentication challenge or not.

Also, the fact, that operators currently do not have specific offers for ‘uplink HTTP traffic generators’, does not mean that they won’t have such in the future, when mobile webapplications and web services will be commodalties. Just as today one can select from a multitude of plans with different benefits for different phone usage profiles, there will surely be a plethora of service types for different — private, social, entrepreneur, enterprise — mobile content providers.

8 CONCLUSION

It is possible to provide HTTP access to web servers running on mobile phones on today’s operator networks without any special arrangements with operators. Our current implementation is stable enough to provide HTTP accessibility to a fair amount of mobile users, ready for use in university research projects and small businesses willing to maintain a gateway on their own. For an industrial use, scalability, security, access and cost control has to be provided as well, but none of these problems are conceptually difficult or even novel, a combination of off-the-shelf tools and techniques in webhosting and website management will likely to go a long way in developing a full-fledged solution.

All code has been open sourced and is available from SourceForge[13].

REFERENCES

- [1] Kapil Chawla, Zhimei Jiang, Xiaoxin Qiu, and Amy Reibman. Transmission of streaming video over EGPRS wireless network. In *Proceedings of First IEEE International Conference on Multimedia and Expo*, July 2000.
- [2] ComVu. Pocketcaster. www.comvu.com.
- [3] Wikman Johan Dosa Ferenc, Tarkiainen Mikko. Personal website on a mobile phone. <http://research.nokia.com/tr/NRC-TR-2006-004.pdf>.
- [4] Apache Software Foundation. Apache http server project. httpd.apache.org.
- [5] Apache Software Foundation. Apache tomcat. tomcat.apache.org.
- [6] Network Working Group. Hypertext transfer protocol – HTTP/1.1. <http://www.rfc-editor.org/rfc/rfc2616.txt>.
- [7] Hao He. What is service-oriented architecture. webservices.xml.com/pub/a/ws/2003/09/30/soa.html.
- [8] Jonathan Lemon. Kqueue: A generic and scalable event notification facility. people.freebsd.org/~jlemon/papers/kqueue.pdf.
- [9] Davide Libenzi. `epoll` scalability. lse.sourceforge.net/epoll/index.html.
- [10] Tapuz Mobile. Blogtv. <http://www.tapuzmobile.com/products.asp>.
- [11] Masahide Nakamura, Hiroshi Igaki, Haruaki Tamada, and Ken ichi Matsumoto. Implementing integrated services of networked home appliances using service oriented architecture. In *Proceedings of the 2nd international conference on Service Oriented Computing*. ACM Press, November 2004.
- [12] Timo Ojala, Jani Korhonen, Tiia Sutinen, Pekka Parhi, and Lauri Aalto. Mobile karpät – a case study in wireless personal area networking. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*. ACM Press, October 2004.
- [13] SourceForge. Mobile web server. <http://sourceforge.net/projects/raccoon>.
- [14] W. Richard Stevens. *Advanced Programming In The Unix Environment*. Addison-Wesley, 1993.
- [15] Belle L. Tseng, Ching-Yung Lin, and John R. Smith. Video summarization and personalization for pervasive mobile devices. In *Storage and Retrieval for Media Databases*, January 2002.
- [16] Jia Zhang, Jen-Yao Chung, and Carl K. Chang. Migration to web services oriented architecture - a case study. In *Proceedings of the 2004 ACM symposium on Applied computing*. ACM Press, March 2004.
- [17] Loaf Zimmermann, Sven Milinski, Michael Craes, and Frank Oellermann. Second generation web services-oriented architecture in production in the finance industry. In *Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. ACM Press, October 2004.