Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: July 16, 2014

O. Nakamura
Keio Univ./WIDE Project
H. Hazeyama
NAIST / WIDE Project
Y. Ueno
Keio Univ./WIDE Project
A. Kato
Keio Univ. / WIDE Project

January 12, 2014

IPv4 Address Literal in URL
draft-osamu-v6ops-ipv4-literal-in-url-01

Abstract

In an IPv6-only environment with DNS64/NAT64 based translation service, there is no way to get access a URL whose domain name part includes an IPv4 address literal. This memo discusses a few methods to rewrite the URL on an IPv6-only host so that the URL is accessible from the IPv6-only host.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 16, 2014.

1. Introduction and Overview

When a host in an IPv6 only environment (an IPv6-only host) has to access an IPv4-only destination, a translator-based approach is a powerful tool. The translator-based approach is usually composed of a DNS64 server [RFC6147] and a stateful NAT64 translator [RFC6146]. The DNS64 server responds with an AAAA record of an IPv4 embedded IPv6 address with significant IPv6 prefix assigned to the NAT64 translator. The IPv6-only host sends an IPv6 packet, which is translated by the NAT64 box to an IPv4 packet. The translation of responded IPv4 packet back into an IPv6 packet is also performed in the NAT64 translator.

The NAT64 with DNS64 approach works well for most destinations. It does not work well when the DNS response packet included NXDOMAIN or SERVFAIL to the AAAA query, partly described in [RFC4074]. Resolution of this case is out of scope of this memo.

It is legitimate to embed an IPv4 address literal in an URL such as follows:

    http://192.0.2.10/index.html

In the environment described above, the destination is not accessible from an IPv6-only host. This problem has already been reported in [RFC6586] and other experiences.

The reason why we cannot access the destination specified by above notation is that no DNS lookup is performed in most cases, and no DNS64 service is able to tell an IPv4 embedded IPv6 address to the host. To perform DNS64/NAT64 translation against such an IPv4

address literal notation, some mechanism will be required.

This memo proposes a special-use TLD. We denote the special-use TLD as ``.TLD'' which will be replaced with actual TLD based on discussion in Section 3.5. The concept of ``.TLD'' is simple; all IPv4 address literal notations are rewritten to ``<ipv4-address-literal>.TLD'' on the IPv6-only host, letting DNS servers and translators translate the IPv4 address literal to appropriate IP address on each leaf network. For example, <ipv4-address-literal>.TLD in DNS64/NAT64 environment would be translated to a NAT64 prefix mapped address.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Scope of this memo

Before discussing solutions, we define the scope of this memo. We focus only on smooth migration to an IPv6-only environment with the DNS64/NAT64 solution. Therefore, we focus on only ``IPv4 address literal'' problem mentioned in [RFC6586].

The ``IPv6 address literal'' is out of scope of this memo, because an URL including IPv6 address literal can be accessible in IPv6-only networks and in dual stack networks. The solutions to keep IPv4-only hosts or IPv4-only applications in IPv6 only environment are out of scope on this memo.

3. A special-use TLD for IPv4 Address Literal

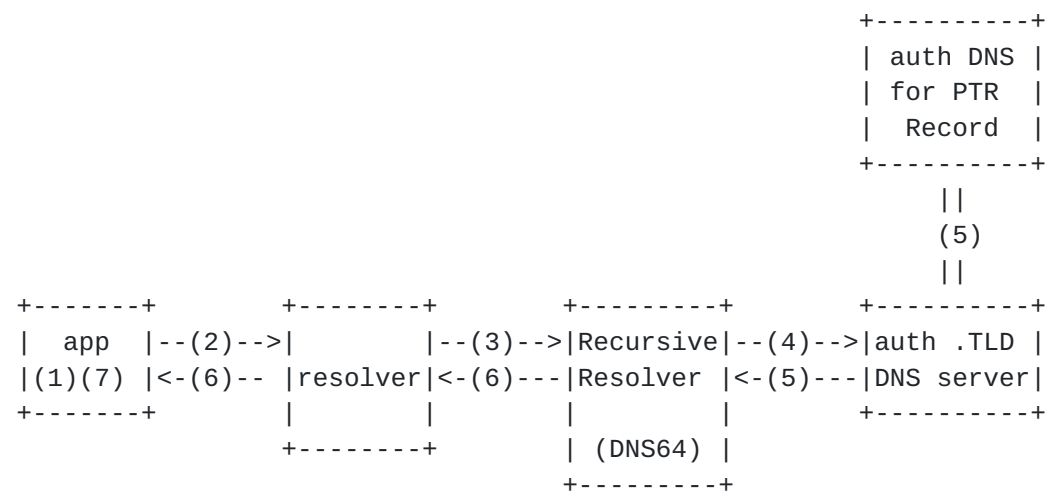
When the part of IPv4 address literal is written to form a pseudo FQDN, the DNS64 server can return an AAAA record with the specified IPv4 address that is mapped to a NAT64 prefix.

Once an AAAA record is obtained, the IPv6-only host can send an IPv6 packet to the destination. The IPv6 packet will be translated via NAT64 translator in the same way as a regular IPv4-only destination. Figure 1 figures out an example of the DNS query flow in the proposal of this memo.

3.1. The procedure in detail

The procedure of the special-use TLD is as follows;

1. An host internally attaches ``.TLD'' to an IPv4 address literal. For example, <ipv4-address-literal> is rewritten to <ipv4-address-literal>.TLD.
2. The host gets access to <ipv4-address-literal>.TLD instead of <ipv4-address-literal>.
3. As <ipv4-address-literal>.TLD looks like a regular FQDN, the IPv6-only host will query the FQDN to a recursive resolver through the local resolver.
4. The DNS server recognizes that the FQDN ``<ipv4-address-literal>.TLD'' is a special-use TLD, then, the recursive resolver forwards the query to a special authoritative DNS server for ``.TLD'' (authoritative .TLD DNS server).
5. When the authoritative .TLD DNS server receives a DNS query, then,
    1. If the query asks ``<ipv4-address-literal>.TLD '', the authoritative .TLD DNS sever simply removes ``.TLD'' part from the FQDN and returns ``<ipv4-address-literal>'' as the A record.
    2. To avoid misuses, the authoritative .TLD DNS server MAY check whether the PTR record corresponding to <ipv4-address-literal> exists or not. If the PTR record of <ipv4-address-literal> exists, the authoritative .TLD DNS authoritative server MAY return a DNS response with a TXT record as well to notify the existence of the PTR record corresponding to the issued IPv4 address literal.
    3. Otherwise, the authoritative .TLD DNS server returns NXDOMAIN.
6. When the recursive resolver, that issued ``<ipv4-address-literal>.TLD '', receives the DNS response from the authoritative .TLD DNS server, then,
    1. If the network is an IPv6-only network, and if the IPv6-only network has a DNS64 server (or function) and NAT64 translator(s), then, the DNS64 server WOULD convert <ipv4-address-literal> into a NAT64 prefix mapped address and returns the NAT64 prefixmapped address as an AAAA record.
    2. If the network is a dual stack network, and if the dual stack network does not have any DNS64/NAT64 function, then, DNS4 server returns <ipv4-address-literal> as an A record.
    3. If the network is a dual stack network, and if the dual stack network has any DNS64/NAT64 function, then, <ipv4-address-literal> MAY be returned as an A record and NAT64 prefix mapped address MAY be returned as an AAAA record.
    4. If the network is an IPv4 only network, then, DNS4 server returns <ipv4-address-literal> as an A record.
7. After the name resolution procedure is completed, the host will access the <ipv4-address-literal> through an appropriate socket.

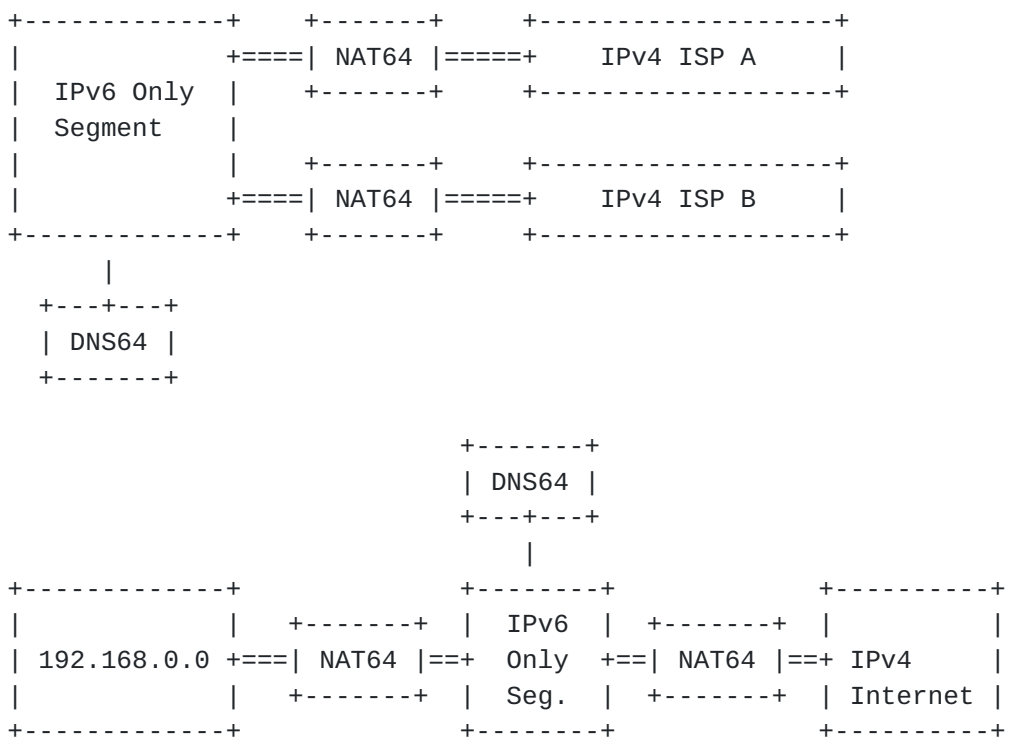


DNS Query Flow

Figure 1

This solution would not require the modification of common shared libraries on Operating Systems. The DNS implementations SHOULD support the special-use TLD and the procedure mentioned above, or a special-use DNS server, that treats only ``.TLD'' domain, SHOULD be placed. The modification of NAT64 or DHCP are not required.

This solution has other advantages. As drawn in Figure 2, a network operator may set DNS64 and several NAT64 nodes to handle or separate for the IPv4 only private segment and/or for multiple global IPv4 access from IPv6 only segments in the local network. In this solution, users do not need to take care of the selection of NAT64 prefixes and NAT64 nodes.



Example of Possible Network Diagrams with DNS64/NAT64

Figure 2

3.2. Use case 1: manual use

For example, consider living on an IPv6-only network with DNS64/ NAT64, and receiving a message like ``please download a file foo.doc from a ftp server 192.0.2.10''. Usually, you may estimate the NAT64 prefix and calculate NAT64 prefix mapped address through [RFC7050] or [RFC7051]. Under the proposed mechanism, you can just type as follow;

```
% ftp 192.0.2.10.TLD
```

The packet would be transferred along with [RFC6384].

3.3. Use case 2: browser plug-in

An IPv4 address literal is often used in URL for the lazy DNS operation, a temporary HTTP server or a hidden (private) server. Taking into account user convenience, a browser plug-in can be developed that it converts the <ipv4-address-literal> on the hostname part of an URL to <ipv4-address-literal>.TLD. It may suggested to turn this on when the host is on IPv6-only network, however, it may not be easy to detect it. The sample of Google Chrome plug-in is attached in Appendix B

3.4. Other Possible Solutions

Several possible solutions can be considered described below, however, most of them would not work between considerable fraction of IPv6 only networks and dual stack networks.

3.4.1. IPv4-Compatible IPv6 Address

The literally embedded IPv4 address is converted to an IPv4-Compatible IPv6 address, and sent to the NAT64 converter as usual. This solution requires application or common shared library modification.

Also, the procedure for learning about the NAT64 prefix to be used for a particular IPv4 destination is issued. Looking up to some known IPv4-only anchor host is a possible method. [RFC7050] says this anchor host solution. However, current ISC BIND DNS64 implementation can set different NAT64 prefixes to different IPv4 address prefixes. Therefore, this solution would not work well.

3.4.2. Various Heuristics and Analysis

[RFC7051] analyzes pros. and cons. of various solutions to learn NAT64 prefix on a host :

1. DNS query for a well-known name,
2. EDNS0 Option Indicating AAAA Record Synthesis and Format,
3. EDNS0 Flags Indicating AAAA Record Synthesis and Format,
4. DNS Resource Record for IPv4-Embedded IPv6 Address,
5. Learning the IPv6 Prefix of a Network's NAT64 Using DNS,
6. Learning the IPv6 Prefix of a Network's NAT64 Using DHCPv6,
7. Learning the IPv6 Prefix of a Network's NAT64 Using Router Advertisements,
8. Using Application-Layer Protocols such as STUN
9. Learning the IPv6 Prefix of a Network's NAT64 Using Access-Technology-Specific Methods.

The proposal of this memo likes to the 5th solution of RFC7051, however, this proposal is a much more simple DNS solution.

3.4.3. Using BIH or 464XLAT CLAT on a host

Bump in the Host (BIH) [RFC6535] and 464XLAT [RFC6877] are possible solutions, both of which are aimed at retaining the usage of IPv4-only applications in IPv6-only environment. BIH has a local translator function between the socket API and the TCP/IP stack. On the other hand, 464XLAT provides PLAT (Provider side XLAT) [RFC6146] and CLAT (Customer side XLAT) [RFC6145]. Some mobile carriers have started CLAT CPE on smart phones. Therefore, some CLAT plug-in on a browser or CLAT CPE on a host may work.

The BIH case requires several extensions on Operating Systems, so we have to wait many independent Operating System Open Source Projects and/or commercial Operating System vendors support BIH in their Operating Systems.

In case of the 464XLAT, 464XLAT assumes that a user always uses a specific CLAT and PLAT pair. If a user want to change or move various public wi-fi networks, then, a global reachable PLAT might be placed and a CLAT on a host has to equip a function that switches the CLAT function depending on the condition of the current network.

3.5. special-use TLD alternatives

A few candidates of the special-use TLD is discussed here.

3.5.1. Use of .host special TLD

A special string ``.host'' is attached to an IPv4 address literal notation like ``192.0.2.10'' to form a pseudo FQDN such as ``192.0.2.10.host''. Also, IPv4 address and port number literal is often used, for instance, ``192.0.2.10:8080'' becomes ``192.0.2.10.host:8080''.

3.5.2. Use of .ip4host.arpa special TLD

This idea is almost the same as ``.host'', but uses ``ip4host.arpa'' top level labels rather than ``.host''. While the pseudo FQDN is expected to appear only in the DNS look up by an IPv6-only host, it may leak to the global Internet. In order to avoid the query to the Root DNS servers, use of the sub-domain of ".arpa" may be recommended. The generated pseudo FQDN looks like ``192.0.2.10.ip4host.arpa''.

3.5.3. Use of .dns64 special TLD

The special-use TLD is mainly aimed to force IPv4 address literal notation to be translated by DNS64/NAT64 in IPv6-only environments. Therefore, ``.dns64'' reflects the request for DNS64/NAT64 translation against an IPv4 address.

3.5.4. Our recommendation

We, authors of this memo, discussed which special-use TLD is suitable. Our recommendation is to use ``.host'' from the view of usability. Of course, we will accept any feedbacks from the community such as IETF v6ops and dnsops working groups and users community.

4. Discussions

4.1. Usages of IPv6 address literal

The special-use TLD may be applied to IPv6 address cases in same ways, however, such notation is not required in dual stack / IPv6-only environment, generally.

4.2. Attached the special-use TLD to a regular FQDN

Conceptually, the special-use TLD would be attached to only IPv4 address literals, however, the special-use TLD may be attached to a regular FQDN notation like ``foo.bar.com.TLD''. We propose that such misuses should be discarded on the DNS or applications on the host side.

4.3. An embedded IP address literal in the content part of URL

In some case, <ipv4-address-literal> may be embedded into the content part of a URL, however, it may be difficult for users or browser plug-ins to recognize unambiguously that a string like <ipv4-address-literal> surely means some IPv4 address. From the point of view of IPv6 migration, embedded IP address literal in the content part of an URL MUST be avoided.

4.4. Prevention the leak of the special-use TLD

When the special-use TLD ``.TLD'' is actually employed in the operation, ``.TLD'' will leak to the public DNS infrastructure including root DNS servers as seen in ``.local''. Therefore, once consensus is obtained, the relevant TLD SHOULD be delegated to a set of DNS servers.

Two possible DNS operation methods can be considered. One is to delegate the TLD to AS112 servers [as112-servers]. When one of the AS112 servers received a query with the special-use TLD, it returns with NXDOMAIN.

The other possible DNS operation is to deploy a set of special purpose DNS servers which accept queries with ``.TLD'' and synthesize an A record corresponding to the IPv4 address in the QNAME when it is a legitimate IPv4 address. Otherwise, NXDOMAIN is returned.

4.5. Possibitily to break connections with Apache VirtualHost concept

Changing the URL (swapping the DNS name or adding in a Pref64) frequently breaks the connections since the application is aware of the name it expects, and connecting correctly to the correct IP

address is not sufficient, the name must also be the same in many cases.

For example, many websites use the Apache VirtualHost concept. When a web site that changes contents along with accessed IP address family like http://www.kame.net/ or http://dual.tlund.se/ , and If some client accesses such web site by <ipv4-address-literal>.TLD instead of FQDN, the VirtualHost may not work as intended.

Therefore, such web site, that uses the Apache VirtualHost concept, SHOULD NOT use <ipv4-address-literal> in URL and SHOULD use appropriate FQDN.

4.6. Inaffinity with HTTP/HTTPS Cookie

This solution may not work with HTTP/HTTPS cookie. We should also consider the HTTP security considerations for the cases where someone puts one of the names into a URL. For example, consider http://192.0.2.10.TLD/ to an origin that sets a cookie on the domain "*.10.TLD".

There are likely already plenty of ways to do the same thing out there, so this may not be a major issue.

5. Implementation Strategy

It is suggested to implement the .TLD rewriting as in the following order:

1. Define .TLD
   Once the community agrees to accept the rewriting scheme described in this memo, it must fix the .TLD to be used. The .TLD WOULD require the update of [RFC6761].
2. .TLD delegation
   DNS queries with .TLD can leak to the DNS of the global Internet, it is highly suggested to delegate .TLD to a set of authoritative DNS servers as discussed in Section 4.4.
3. DNS64 modification
   DNS64 implementation is suggested to modify to respond corresponding AAAA record to a query with .TLD. This process can be done in parallel to the step 2 above.
4. Start using .TLD rewriting
   After, at least the step 2 is completed, the TLD rewriting may be used in manually described in Section 3.2 or automatically by browser plugins described in Section 3.3.
   While further discussions and observation is required, we may discourage to use an URL in IPv4 literal embedded. Instead, we may encourage to use .TLD notation as a legitimate URL even

in the server side.

6. Security Considerations

The recommendation contains security considerations related to DNS. The special DNS of this memo only treats the IPv4 address literal with ``.TLD''. Therefore, the special DNS MAY use self-signed / authorized key for DNS responses.

When a client is to access an URL with IPv4 literal address embedded, it triggers a DNS query, and the query may be sent over the Internet to the nearest authoritative .TLD DNS server. It may break the confidentiality against the DNS service.

7. IANA Considerations

According to the discussion with communities, this memo may call for changes or additions of special-use TLD to the IANA registry.

8. Acknowledgments

The authors thank all members of WIDE Project for their active discussion, implementation, and evaluation. Especially, we thank to Atsushi Onoe for the revision of this solution, Hirochika Asai for the contribution of the proto type implementation of the special authoritative DNS, and Hirotaka Nakajima for the contribution of the Google chrome plug-in.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4074] Morishita, Y. and T. Jinmei, "Common Misbehavior Against DNS Queries for IPv6 Addresses", RFC 4074, May 2005.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, April 2011.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

[RFC6384] van Beijnum, I., "An FTP Application Layer Gateway (ALG) for IPv6-to-IPv4 Translation", RFC 6384, October 2011.

[RFC6535] Huang, B., Deng, H., and T. Savolainen, "Dual-Stack Hosts Using "Bump-in-the-Host" (BIH)", RFC 6535, February 2012.

[RFC6586] Arkko, J. and A. Keranen, "Experiences from an IPv6-Only Network", RFC 6586, April 2012.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

[RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, April 2013.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, November 2013.

[RFC7051] Korhonen, J. and T. Savolainen, "Analysis of Solution Proposals for Hosts to Learn NAT64 Prefix", RFC 7051, November 2013.

9.2. Informative References

[as112-servers]
AS112 Project, "AS112 Project", October 2009, <https://www.as112.net/>.

Appendix A. A Test Server of the special TLD

We run a prototype implementation of the special-use DNS server in the WIDE backbone (AS 2500). We use ``.v4.wide.ad.jp'' as ``.TLD''.

Appendix B. Sample extension for Google Chrome

We developed a sample plug-in code for Google Chrome ``IPv4 Address Literal Appender'' that automatically converts <ipv4-address-literal> in URL to <ipv4-address-literal>.TLD. The .TLD can be customized in the option. The ``IPv4 Address Literal Appender'' is freely

available in Google Chrome Web Store.

```
var wr = chrome.webRequest;

var v4Suffix = ".TLD";
var ipAddrRegex = /^(\d|[01]?\d\d|2[0-4]\d|25[0-5])\.(\d|[01]?\d\d|2[0-
4]\d|25[0-5])\.(\d|[01]?\d\d|2[0-4]\d|25[0-5])\.(\d|[01]?\d\d|2[0-4]\d|2
5[0-5])$/;

function onBeforeRequest(details) {
  var tmpuri = new URI(details.url);
  var tmphost = tmpuri.host();
  var finalUri = '';
  tmphost.replace(ipAddrRegex,function(str, p1, p2, p3, p4, offset, s){
    finalUri = tmpuri.host(p1+"."+p2+"."+p3+"."+p4+v4Suffix).toString();
  });
 if('' != finalUri) {
  console.log(finalUri);
  return {redirectUrl: finalUri};
 }
};

wr.onBeforeRequest.addListener(onBeforeRequest, {urls: ["https://*/*",
"http://*/*", "ftp://*/*"]}, ["blocking"]);
```

Authors' Addresses

Osamu Nakamura
Keio Univ./WIDE Project
5322 Endo
Fujisawa, Kanagawa 252-0882
JP

Phone: +81 466 49 1100
Email: osamu@wide.ad.jp

Hiroaki Hazeyama
NAIST / WIDE Project
8916-5 Takayama
Ikoma, Nara 630-0192
JP

Phone: +81 743 72 5111
Email: hiroa-ha@is.naist.jp

Yukito Ueno
Keio Univ./WIDE Project
5322 Endo
Fujisawa, Kanagawa 252-0882
JP

Phone: +81 466 49 1100
Email: eden@sfc.wide.ad.jp

Akira Kato
Keio Univ. / WIDE Project
Graduate School of Media Design, 4-1-1 Hiyoshi
Kohoku, Yokohama 223-8526
JP

Phone: +81 45 564 2490
Email: kato@wide.ad.jp