Abstract

This document proposes a new, backwards-compatible, approach to packet forwarding, where the service required rather than the IP address required acts as the vector for routing packets at the edge of the network.

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 https://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 December 23, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
1. Introduction

An important aspect of the Internet today is that it is no longer a uniform space with uniform requirements. For both technical and economic reasons, we see an emerging trend of usage scenarios that are confined to some form of limited domain, and which inevitably lead to applications and protocols that are only suitable within a given scope [I-D.carpenter-limited-domains]. This trend collides immediately with two factors: the original design concept of an Internet with end to end IP transparency (such that any locally defined protocol running over IP is almost certain to escape the local network), and with the increasing presence of interfering middleboxes. In this emerging context, where end to end IP service is no longer a safe assumption, and where there is increasing demand for specific services, this document proposes a new, backwards-compatible, approach to packet forwarding, where the service required rather than the IP address required acts as the vector for routing packets at the edge of the network. This is known as Service Based Packet Forwarding (SBPF) or Service Oriented IP (SOIP).

We propose an addition to the existing core function of the network, which is reachability over IPv6 or IPv4. Today, IP is focussed on reachability, using best effort forwarding both to find a route and to automatically share transmission resources in a simple and low-cost way. As a result, transport protocols such as TCP and UDP learn little or nothing about the network, beyond congestion or loss signals. Several ISPs may lie on the path between a user and a server, but they are largely ignorant about the services a user requires. Typical services could be streamed content, regular
content, user posting, storage access, or calculation, but this list is not exclusive.

Both service providers and users will benefit if a packet stream can be identified intrinsically as requiring a certain kind of service. This is particularly applicable for edge networks, such as those supported by 5G technology, where there is an emphasis on upper layer service provision. Whatever the business model - for example the ISP operates all types of service, or the ISP operates no user services at all and has contracts with specific service providers, or the ISP is agnostic about user services - SOIP will allow for optimised packet delivery. The ISP will have the choice to provide some or all services. The user will have the choice to use ISP services or bypass them. Traffic that leaves the domain where SOIP is in use will be perfectly normal IPv6 or IPv4 traffic, sent by an exit node acting as a proxy (not an IP-layer translator) for the user. Additionally, IPv6 and IPv4 will be modelled as services available to the user, thereby giving continuity of access to everything the user has today. This is a logical extension of a principle already adopted to model IPv4 as a service available via IPv6 [RFC8585].

2. Proposed Solution

NOTE: This is a preliminary draft expected to stimulate discussion, so many details are not yet defined.

The approach is to make the service be the central component of the network. Conceptually, the user’s packets will be directed at a service, not at an IP host.

The service actions that the network can provide are abstracted into a number of classes called Service Action Types (SATs). While there needs to be flexibility and extensibility, the number of service action types will be limited. They will not be numerous like IP protocol numbers or well-known TCP or UDP port numbers. Along with the SAT, a source IPv6 address is used to identify the client system. As will be seen below, the destination IPv6 address becomes optional. A consequence is that the IP header and some aspects of the protocol stack have to be redesigned. We will show below how this can be done without disturbing most of the running network. Another consequence is that the first step in processing a packet is to process the SAT, not the destination address (if there is one).

Traditional reachability, when required, is provided by classical IPv6, or by IPv4 as-a-service.
When an SOIP packet enters a router, it is classified at line speed according to the SAT. A preliminary list of SATs is shown in Figure 1, with brief descriptions:

<table>
<thead>
<tr>
<th>SATs</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4 reachability</td>
<td>Request</td>
<td>IPv4 destination host</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>IPv4 source host</td>
</tr>
<tr>
<td>IPv6 reachability</td>
<td>Request</td>
<td>IPv6 destination host</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>IPv6 source host</td>
</tr>
<tr>
<td>Discovery Service</td>
<td>Request</td>
<td>Discover network services, e.g.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DNS, CDN. May map to IP Anycast</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Content ID, service ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or &quot;service not available&quot; error</td>
</tr>
<tr>
<td>Multicast Service</td>
<td>Request</td>
<td>Join multicast service for some</td>
</tr>
<tr>
<td></td>
<td></td>
<td>content, e.g. video stream</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Multicast directory answers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>request, provides m/c source.</td>
</tr>
<tr>
<td>Computation Service</td>
<td>Request</td>
<td>Submit task to network, with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>computation type, task</td>
</tr>
<tr>
<td></td>
<td></td>
<td>descriptor and requester ID</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Computation resource ID, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>direct result of task.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or &quot;service not available&quot; error</td>
</tr>
<tr>
<td>Storage Service</td>
<td>Request</td>
<td>Submit/retrieve data to/from</td>
</tr>
<tr>
<td></td>
<td></td>
<td>network storage, with data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>description and/or data ID</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Storage locator, data ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or &quot;service not available&quot; error</td>
</tr>
<tr>
<td>App Server Service</td>
<td>Request</td>
<td>To submit source code or deploy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>package to application platform,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>with necessary configurations.</td>
</tr>
<tr>
<td></td>
<td>Response</td>
<td>Answer with service ID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Or &quot;service not available&quot; error</td>
</tr>
</tbody>
</table>

Figure 1
For each request there will be a corresponding response. The details remain to be worked out - probably a generic response message including the SAT. To allow multiple overlapping sessions, each request/response sequence should have a unique ID.

For IPv6-only networks, it is expected that IPv4 reachability will be provided by a solution such as 464XLAT. Also, no separate SAT is needed for IPv6 to IPv4 translation. For example, if a host requests IPv4 reachability but supplies an IPv6 address as its own locator, NAT64 is implied.

For the moment codes for the SATs are undefined, but they are assumed to be small integers. There are two possible approaches to the packet format. One is to use a traditional Type-Length-Value (TLV) layout. Another is to use a more flexible encoding at the lowest level, taking advantage of some form of network processor in the routers. An obvious choice would be Concise Binary Object Representation (CBOR) [RFC7049], which combines flexibility with efficiency. In either case we could require that the first four bits of the wire format are a new IP version number other than 4 or 6. An alternative, at least for early experimentation, is to run SOIP over UDP and IPv6.

Examples of both encoding choices are described below. In either case, the essential content of a packet header is as follows:

- The SAT code (small integer)
- Flag bits
- Traffic class (as for IPv6)
- Session Identifier (so that sessions can be tracked regardless of IP address)
- Hop limit (small integer)
- User locator (IP address or identifier)
- Service data length (not needed in CBOR version)
- Service data (length depends on SAT)
- Payload length (not needed in CBOR version)
- Payload
Experience with IPv4 options, and IPv6 extension headers and options, has shown that new ones are very hard to deploy on an operational network, and that the ones defined during the initial design are not always useful. Therefore we propose that all options and extensions are defined as part of the service data and are not visible as part of the basic packet header, giving good flexibility.

We propose to include the exact equivalent of the IPv6 Traffic Class, which can work exactly as for IPv6 and IPv4. In contrast, one defect in the IPv6 flow label is that it is different in the two directions of a flow. Instead we propose a session ID that is the same in both directions, which has various advantages by allowing immediate session identification.

The flag bits provide useful indications to the routing system, if set:

- Mobile - set if the user system is mobile
- Flow size (3 bits)
  - 000 means a single packet, no flow/congestion state needed
  - other values TBD indicate type of flow/congestion state
- Authenticated - set if packet authenticated (details TBD)
- Encrypted - set if encryption applied (details TBD)

Note that fragmentation is not supported. Fragmentation, and the related mechanisms of MTU discovery, are a significant operational problem in the current Internet [I-D.ietf-intarea-frag-fragile]. We simply abolish this problem area in SOIP. It is designed for use in managed networks where a single size of maximum transmission unit (MTU) is available everywhere. An SOIP network will have a globally defined MTU. Of course, IPv4 and IPv6 reachability services via the open Internet will have to support PMTU and fragmentation as best they can, but this concerns the embedded IP packets, not the SOIP packets, and will be invisible locally.

Appendix A outlines possible TLV and CBOR encodings of the SOIP protocol.

3. Coexistence Issues

SOIP is expected to coexist with IPv6; in a sense it is a low-level method of orchestrating IPv6 connections. We assume that each SOIP client host has at least one IPv6 address, and that SOIP routers will
announce themselves using a suitable IPv6 Router Advertisement extension [I-D.troan-6man-universal-ra-option]. Normally, the first-hop SOIP router will be the same as the IPv6 first-hop router.

As a result of this, all standard management mechanisms such as NETCONF may be used without further specification. Also, when a data connection of any kind is established after a SOIP request/response exchange, all standard transport mechanisms are available over IPv6. As noted above, they are subject to the locally defined MTU as long as they remain within the SOIP domain.

We do not define how SOIP would operate in an IPv4-only network.

4. Some Usage Examples

- Storage request (upload content):
  * Service data identifies storage requirement (temporary/permanent, private/public, encryption, etc.)
  * Payload identifies data (path/name.format, etc.)

- Storage request (download content):
  * Service data identifies transmission requirement (streamed, block, etc. and the specific transport protocol - UDP, TCP, QUIC, etc. - if needed)
  * Payload identifies data (path/name.format, etc.)

- Computation request
  * Service data identifies computing requirement
  * Payload identifies computing application

- Reachability request
  * Service data gives destination IP address (or DNS name)
  * Indication of transport protocol required (details TBD)
  * Indication of options or extension headers required (details TBD)
5. Continuity with the Existing Internet

Continuity is provided in two ways:

1. A user node can simply use IP completely in parallel with SOIP. The network stack in the user node will simply encode the IP packets as SOIP packets with the SAT for IP reachability, and a gateway function will send and receive IP packets at the SOIP domain boundary.

2. If a service in the SOIP domain needs service from elsewhere in the IP Internet to respond to a user request, it will use a gateway function to do so. This could also be described as a proxy mechanism. (Of course, services in interconnected SOIP domains may talk to each other directly.)

6. Security Considerations

It is intended that both authentication and encryption should be available for all SOIP packets. However, this requires further work, especially to determine whether existing mechanisms for key management can be used.

Since clients are identified by an IPv6 address, existing layer 3 privacy considerations for IPv6 addresses will apply to SOIP. (References needed.) Upper layer privacy considerations will depend on the service concerned.

7. IANA Considerations

This document makes no request of the IANA.

8. References

[I-D.carpenter-limited-domains]

[I-D.ietf-intarea-frag-fragile]
Appendix A. Possible TLV and CBOR Encodings

A.1. TLV Mapping
Figure 2

Version - IP version number TBD

R - Reserved, must be zero

SAT - Service Action Type code

M - Mobile flag

FFF - Flow type flags (FFF = 000 for single-packet flows; other values for longer flows)

A - Authentication flag

E - Privacy (encryption) flag

Traffic class, exactly as for IPv6

Session identifier, 32-bit pseudo random number

Client Locator / Identifier - globally unique IPv6 address
A.2. CBOR Mapping

The packet consists of a CBOR byte string preceded by a single byte whose format is:

```
+-----------------------------+
| Version| R R R R |
+-----------------------------+
```

For example, for version 7, this byte would be 0x70. This byte is not decoded as CBOR. The CBOR bytes then obey the following CDDL [RFC8610] specification:

```
sat-packet = [sat, flags, traffic-class, session-id, hop-limit,
              source, service-data, ?payload]
```

sat = 0..255
flags = bytes .size 1
traffic-class = 0..255
session-id = 0..4294967295 ;up to 32 bits
hop-limit = 0..255
client = ipv6-address
service-data = any
payload = any
ipv6-address = bytes .size 16

The syntax of the various service-data formats can be defined in separate documents for each SAT value.

We assume that routers capable of handling a CBOR-based layer 3 protocol will exist, and will use some form of programmable network processor rather than traditional ASIC or FPGA designs. This allows great flexibility and software-friendly extensibility, especially of the service data formats. Further investigation is needed whether this is realistic.

Appendix B. Change log [RFC Editor: Please remove]

draft-jiang-service-oriented-ip-00, 2019-05-07:

Initial version

draft-jiang-service-oriented-ip-01, 2019-06-21:
Editorial corrections

Authors’ Addresses

Brian Carpenter
The University of Auckland
School of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand
Email: brian.e.carpenter@gmail.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China
Email: jiangsheng@huawei.com

Guangpeng Li
Huawei Technologies
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China
Email: liguangpeng@huawei.com