

6LoWPAN
Internet-Draft
Intended status: Standards Track
Expires: December 8, 2010

P. Thubert
Cisco
June 6, 2010

6LoWPAN Backbone Router
draft-thubert-6lowpan-backbone-router-02

Abstract

Some LLN subnets are expected to scale up to the thousands of nodes and hundreds of routers. This paper proposes an IPv6 version of the Backbone Router concept that enables such a degree of scalability using a high speed network as a backbone to the subnet.

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 December 8, 2010.

Copyright Notice

Copyright (c) 2010 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 (<http://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 the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Overview	6
4. New types and formats	8
4.1. Binding Tracking Option	8
5. Backbone Router Operations	10
5.1. Backbone Link and Router	10
5.2. ND Proxy Operations	10
5.3. Claiming and defending	12
5.4. Conflict Resolution	12
5.5. Assessing an entry	13
6. Security Considerations	15
7. IANA Considerations	16
8. Acknowledgments	17
9. References	18
9.1. Normative References	18
9.2. Informative References	18
Author's Address	20

1. Introduction

The ISA100.11a standard has introduced the concept of a Backbone Router that would interconnect small LLNs over a high speed transit network and scale a single ISA100.11a network up to the thousands of nodes. In that model the LLNs and the backbone form a single subnet in which nodes can move freely without the need of renumbering, and the Backbone Router is a special kind of Border Router designed to manage the interaction between the LLNs and the backbone at layer 3. Similar scalability requirements exist in the metering and monitoring industries. In a network that large, it is impossible for a node to register to all Border Routers as suggested for smaller topologies in Neighbor Discovery Optimization for Low-power and Lossy Networks [I-D.ietf-6lowpan-nd].

This paper specifies IP layer functionalities that are required to implement the concept of a Backbone Router with IPv6, in particular the application of the "IP Version 6 Addressing Architecture" [RFC4291], " the Neighbor Discovery Protocol" [RFC4861] and "IPv6 Stateless Address Autoconfiguration" [RFC4862].

The use of EUI-64 based link local addresses, Neighbor Discovery Proxying [RFC4389], Neighbor Discovery Optimization for Low-power and Lossy Networks [I-D.ietf-6lowpan-nd], the IPv6 Routing Protocol for Low power and Lossy Networks [I-D.ietf-roll-rpl] and Optimistic Duplicate Address Detection [RFC4429] are discussed. Also, the concept of Transit Link is introduced to implement the backbone network that was envisioned by ISA100.11a.

This operation of the Backbone Router requires that some protocol operates over the LLNs from which node registrations can be obtained, and that can disseminate the location of the backbone Router over the LLN. Further expectations will be detailed.

The way the PAN IDs and 16-bit short addresses are allocated and distributed in the case of an 802.15.4 network is not covered by this specification. Similarly, the aspects of joining and securing the network are out of scope. The way the nodes in the LLN learn about their Backbone Router depends on the protocol used in the LLN. In the case of RPL, a Border Router is the root of the DODAG that it serves and represents all nodes attached to that DODAG.

2. Terminology

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].

Readers are expected to be familiar with all the terms and concepts that are discussed in "Neighbor Discovery for IP version 6" [RFC4861], "IPv6 Stateless Address Autoconfiguration" [RFC4862], "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919], "Neighbor Discovery Optimization for Low-power and Lossy Networks [I-D.ietf-6lowpan-nd] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

Readers would benefit from reading "Mobility Support in IPv6" [RFC3775], "Neighbor Discovery Proxies (ND Proxy)" [RFC4389] and "Optimistic Duplicate Address Detection" [RFC4429] prior to this specification for a clear understanding of the art in ND-proxying and binding.

Additionally, this document uses terminology from [I-D.ietf-roll-terminology], and introduces the following terminology:

Backbone

This is an IPv6 transit link that interconnects 2 or more Backbone Routers. It is expected to be deployed as a high speed backbone in order to federate a potentially large set of LLNs. Also referred to as a LLN backbone or transit network.

Backbone Router

An IPv6 router that federates the LLN using a transit link as a backbone.

Extended LLN

This is the aggregation of multiple LLNs as defined in [RFC4919] interconnected by a Transit Link via Backbone Routers and forming a single IPv6 link.

Binding

The association of the LLN node IPv6 address and Interface ID with associated proxying states including the remaining lifetime of that association.

Registration

The process during which a LLN node injects its address in a protocol through which the Border Router can learn the address and proxy ND for it.

Primary BR

The BR that will defend a registered address for the purpose of DAD over the backbone

Secondary BR

A BR to which the address is registered. A Secondary Router MAY advertise the address over the backbone and proxy for it.

3. Overview

A Transit Link that we'll refer to as the LLN Backbone federates multiple LLNs as a single IP subnet. Each LLN in the subnet is anchored at a Backbone Router. The Backbone Routers interconnect the LLNs over that Backbone Link. A node can move freely from a LLN anchored at a Backbone Router to a LLN anchored at another Backbone Router on the same backbone and conserve its link local and any other IPv6 address it has formed.

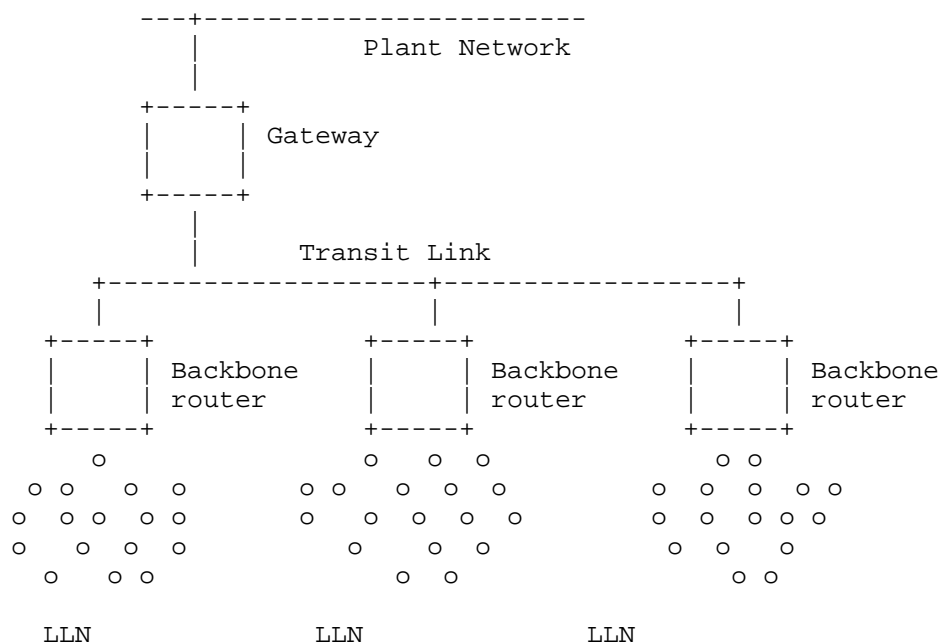


Figure 1: Backbone Link and Backbone Routers

The Backbone Link is used as reference for Neighbor Discovery operations, by extending the concept of a Home Link as defined in [RFC3775] for Mobile IPv6. In particular, Backbone Routers perform ND proxying for the LLN nodes in the LLNs they own through a node registration.

The Backbone Router operation is compatible with that of a Home Agent. This enables mobility support for LLN devices that would move outside of the network delimited by the transit link. This also enables collocation of Home Agent functionality within Backbone Router functionality on the same interface of a router.

A LLN node registers and claims ownership of its addresse(s) using proactive acknowledged registration exchanges with a neighboring router. In case of a complex LLN topology, the router might be an intermediate LLN Router that relays the registration to the LBR as described for instance in [I-D.ietf-6lowpan-nd] and [I-D.ietf-roll-rpl]. In turn, the Backbone Routers operate as a distributed database of all the LLN nodes and use the Neighbor Discovery Protocol to share that information across the transit link in a reactive fashion.

For the purpose of Neighbor Discovery proxying, this specification documents the LLN Master Neighbor Registry, a conceptual data structure that is similar to the MIP6 binding cache. The Master Neighbor Registry is fed by redistributing addresses learnt from the registration protocol used over the LLN.

Another function of the Backbone Router is to perform 6LoWPAN compression and expansion between the LLN and the Transit Link and ensure MTU compatibility. Packets flow uncompressed over the Transit Link and are routed normally towards a Gateway or an Application sitting on the transit link or on a different link that is reachable over the IP network.

4. New types and formats

The specification expects that the protocol running on the LLN can provide a sequence number called Transaction ID (TID) that is associated to the registration. When a node registers to multiple BRs, it is expected that the same TID is used, to enable the BR to correlate the registrations as being a single one, and differentiate that situation from a movement. Otherwise, the resolution makes it so that only the most recent registration was perceived from the highest TID is kept.

The specification expects that the protocol running on the LLN can provide a unique ID for the owner of the address that is being registered. The Owner Unique ID enables to differentiate a duplicate registration from a double registration. In case of a duplicate, the last registration loses. The Owner Unique ID can be as simple as a EUI-64 burnin address, if the device manufacturer is convinced that there can not be a manuf error that would cause duplicate EUI64 addresses. Alternatively, the unique ID can be a hash of supposedly unique information from multiple orthogonal sources, for instance:

- o Burn in address.
- o configured address, id, security keys...
- o (pseudo) Random number, radio link metrics ...

In any fashion, it is recommended that the device stores the unique Id in persistent memory. Otherwise, it will be prevented to reregister after a reboot that would cause a loss of memory until the Backbone Router times out the registration.

The unique ID and the sequence number are placed in a new ND option that is used by the Backbone Routers over the transit link to detect duplicates and movements. The option format is as follows:

4.1. Binding Tracking Option

This option is designed to be used with standard NS and NA messages between backbone Routers over a backbone link and may be used between LRs and LBRs over the LLN. By using this option, the binding in question can be uniquely identified and matched with the Master Neighbor Registry entries of each Backbone Router.

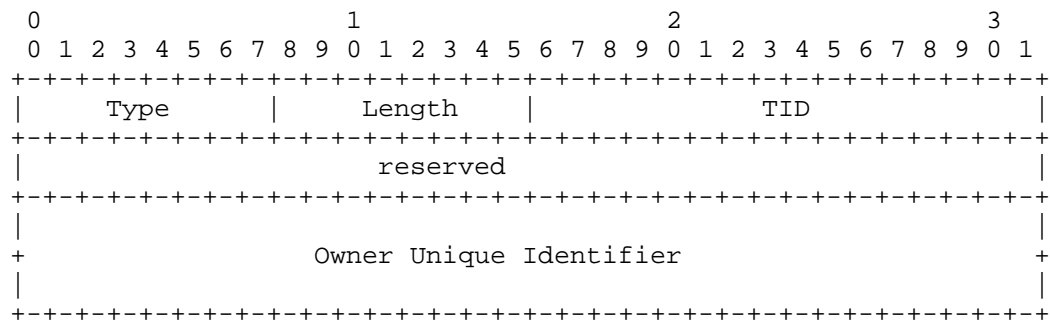


Figure 2: Binding Tracking Option

Option Fields

Type:

Length: 2

TID: A unique Transaction ID assigned by the host in the associated NR and used to match NC replies. The TID is set to zero when the node boots and then follows a lollipop lifetime, wrapping directly from 0xFFFF to 0x10.

Reserved: This field is unused. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Owner Unique Identifier: A globally unique identifier for the host's interface associated with the binding for the NS/NA message in question. This can be the EUI-64 derived IID of an interface, which can be hashed with other supposedly unique information from multiple orthogonal sources.

5. Backbone Router Operations

5.1. Backbone Link and Router

The Backbone Router is a specific kind of Border Router that performs proxy Neighbor Discovery on its backbone interface on behalf of the nodes that it has discovered on its Low Power Lossy Network interfaces. On the LLN side, the Backbone Router acquires its states about the nodes by terminating protocols such as RPL [I-D.ietf-roll-rpl] or 6LoWPAN ND [I-D.ietf-6lowpan-nd] as a LLN Border Router. It is expected that the backbone is the medium used to connect the subnet to the rest of the infrastructure, and that all the LBRs are connected to that backbone and support the Backbone Router feature as specified in this document.

The backbone is expected to be a high speed, reliable transit link, with affordable multicast capabilities, such as an Ethernet Network or a fully meshed NBMA network with multicast emulation, which allows a full support of classical ND as specified in [RFC4861] and subsequent RFCs. In other words, the backbone is not a LLN. Still, some restrictions of the attached LLNs will apply to the backbone. In particular, it is expected that the MTU is set to the same value on the backbone and all attached LLNs.

5.2. ND Proxy Operations

This specification enables a Backbone Router to proxy Neighbor Discovery operations over the backbone on behalf of the nodes that are registered to it, allowing any device on the backbone to reach a LLN node as if it was on-link.

In the context of this specification, proxy ND means:

- o defending a registered address over the backbone using NA messages with the Override bit set
- o advertising a registered address over the backbone using NA messages, asynchronously or as a response to a Neighbor Solicitation messages.
- o Looking up a destination over the backbone in order to deliver packets arriving from the LLN using Neighbor Solicitation messages.
- o Forwarding packets from the LLN over the backbone, and hte other way around.

- o Eventually triggering a look up for a destination over the LLN that would not be registered at a given point of time, or as a verification of a registration.

The draft introduces the concept of primary and secondary BRs. The concept is defined with the granularity of an address, that is a given BR can be primary for a given address and secondary or another one, regardless on whether the addresses belong to the same node or not. The primary Backbone Router is in charge of protecting the address for DAD over the Backbone. Any of the Primary and Secondary BR may claim the address over the backbone, since they are all capable to route from the backbone to the LLN device.

When the protocol used to register the nodes over the LLN is RPL [I-D.ietf-roll-rpl], it is expected that one BR acts as virtual root coordinating LLN BRs (with the same DODAGID) over the non-LLN backbone. In that case, the virtual root may act as primary BR for all addresses that it cares to support, whereas the physical roots to which the node is attached are secondary BRs. It is also possible in a given deployment that the DODAGs are not coordinated. In that case, there is no virtual root and no secondary BR; the DODAG root is primary all the nodes registered to it over the backbone.

When the protocol used to register the nodes over the LLN is 6LoWPAN ND [I-D.ietf-6lowpan-nd], the Backbone Routers act as a distributed DAD table, using classical ND over the backbone to detect duplication. This specification requires that:

1. Registrations for all addresses that can be required to reach the device over the backbone, including registrations for IPv6 addresses based on burn-in EUI64 addresses are passed to the DAD table.
2. Nodes include the Binding Tracking Option in their NS used for registering those addresses and the LRs propagate that option to the LBRs.

A false positive duplicate detection may arise over the backbone, for instance if the node registers to more than one LBR, or if the node has moved. Both situations are handled gracefully unbeknownst to the node. In the former case, one LBR becomes primary to defend the address over the backbone while the others become secondary and may still forward packets back and forth. In the latter case the LBR that receives the newest registration wins and becomes primary.

5.3. Claiming and defending

Upon a new or an updated registration, the BR performs a DAD operation. If either a TID or a OUI is available, the BR places them in a Binding Tracking Option in all its ND messages over the backbone. If content is not available for a given field, it is set to 0.

If a primary already exists over the backbone, it will answer the DAD with an RA.

- o If a RA is received with the O bit set, the primary rejects the DAD and the DAD fails. the BR needs to inform the LLN protocol that the address is a duplicate.
- o If a RA is received with the O bit reset, the primary accepts the BR as secondary and DAD succeeds. The BR may install or maintain its proxy states for that address. This router MAY advertise the address using a NA. during a registration flow, it MAY set the O bit.
- o If no RA is received, this router assumes the role of primary and DAD succeeds. The BR may install or maintain its proxy states for that address. This router advertises the address using a NA with the O bit reset.

When the BR installs or maintains its proxy states for an address, it sends an NA with a SLLA option for that address. The Primary BR MAY set the O bit if it wished to attract the traffic for that address.

5.4. Conflict Resolution

A conflict arise when multiple BRs get a registration from a same address. This situation might arise when a node moves from a BR to another, when a node registers to multiple BRs, or in the RPL case when the BRs belong to a single coordinated DODAG.

The primary looks up the Binding Tracking Option in ND messages with a SLLA option.

- o If there is no option, normal ND operation takes place and the primary defends the address with a NA with the O bit set, adding the Binding Tracking Option with its own information.
- o If there is a Binding Tracking Option and the OUIs are different, then the conflict apparently happens between different nodes, and the the primary defends the address with a NA with the O bit set, adding the Binding Tracking Option with its own information. If

the TID in the Binding Tracking Option is in the straight part of the lollipop, it is possible that the request comes from the same node that has rebooted and formed a new OUI. The primary BR may assess its registered entry prior to answering.

- o If there is a Binding Tracking Option and the OUIs are the same:
 - * If the TID in the ND message is newer than the most recent one known by the primary router, this is interpreted as a node moving. The primary cleans up its states and stops defending the address.
 - * If the TID in the ND message is the same as the most recent one known by the primary router, this is interpreted as a double registration. In case of a DAD, the primary responds with a NA with the O bit reset, to confirm its position as primary, including the Binding Tracking Option.
 - * If the TID in the ND message is older than the most recent one known by the primary router, this is interpreted as a stale information. The primary defends the address with a NA with the O bit set, adding the Binding Tracking Option with its own information.
 - * If the TIDs are very different (more than 16 apart, discounting the straight part of the lollipop), it is impossible to resolve for sure. The primary BR should assess its registered entry prior to answering.

5.5. Assessing an entry

In a number of cases, it might happen that the information at the primary BR is stale and obsolete. In particular, a node with no permanent storage might reboot and form a different OUI, in which case the information at the BR is inaccurate and should be removed. In another case, the BR and the node have been out of reach for too long and the TID that the BR maintains is so far out that it is impossible to compare it with that stored at the BR.

In such situation, the primary Backbone Router has the possibility to assess the registration. This is performed by the protocol in use to register the node over the LLN.

When the protocol used to register the nodes over the LLN is RPL [I-D.ietf-roll-rpl], the BR sends a targetted DIS to the registered address over the registered path. A DAO back indicates that the current registration is still valid and provides the adequate data to resolve the conflict.

When the protocol used to register the nodes over the LLN is 6LoWPAN ND [I-D.ietf-6lowpan-nd], TBD.

6. Security Considerations

This specification expects that the link layer is sufficiently protected, either by means of physical or IP security for the Transit Link or MAC sublayer cryptography. In particular, it is expected that the LLN MAC provides secure unicast to/from the Backbone Router and secure broadcast from the Backbone Router in a way that prevents tempering with or replaying the RA messages.

The use of EUI-64 for forming the Interface ID in the link local address prevents the usage of Secure ND ([RFC3971] and [RFC3972]) and address privacy techniques. Considering the envisioned deployments and the MAC layer security applied, this is not considered an issue at this time.

7. IANA Considerations

A new type is requested for an ND option.

8. Acknowledgments

The author wishes to thank Zach Shelby for their help and in-depth review.

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.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

9.2. Informative References

- [I-D.ietf-6lowpan-nd]
Shelby, Z., Chakrabarti, S., and E. Nordmark, "Neighbor Discovery Optimization for Low-power and Lossy Networks", draft-ietf-6lowpan-nd-09 (work in progress), April 2010.
- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., and R. Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", draft-ietf-roll-rpl-08 (work in progress), May 2010.
- [I-D.ietf-roll-terminology]
Vasseur, J., "Terminology in Low power And Lossy

Networks", draft-ietf-roll-terminology-03 (work in progress), March 2010.

[I-D.van-beijnum-multi-mtu]

Beijnum, I., "Extensions for Multi-MTU Subnets", draft-van-beijnum-multi-mtu-02 (work in progress), February 2008.

[RFC3963] Devarapalli, V., Wakikawa, R., Petrescu, A., and P. Thubert, "Network Mobility (NEMO) Basic Support Protocol", RFC 3963, January 2005.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, April 2006.

[RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, August 2007.

Author's Address

Pascal Thubert
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

Phone: +33 4 97 23 26 34
Email: pthubert@cisco.com

6LoWPAN
Internet-Draft
Intended status: Standards Track
Expires: December 6, 2010

P. Thubert, Ed.
Cisco
J. Hui
Arch Rock Corporation
June 4, 2010

LoWPAN fragment Forwarding and Recovery
draft-thubert-6lowpan-simple-fragment-recovery-07

Abstract

Considering that the IPv6 minimum MTU is 1280 bytes and that an 802.15.4 frame can have a payload limited to 74 bytes in the worst case, a packet might end up fragmented into as many as 18 fragments at the 6LoWPAN shim layer. If a single one of those fragments is lost in transmission, all fragments must be resent, further contributing to the congestion that might have caused the initial packet loss. This draft introduces a simple protocol to forward and recover individual fragments that might be lost over multiple hops between 6LoWPAN endpoints.

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 December 6, 2010.

Copyright Notice

Copyright (c) 2010 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 (<http://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 the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	4
3. Rationale	4
4. Requirements	6
5. Overview	7
6. New Dispatch types and headers	7
6.1. Recoverable Fragment Dispatch type and Header	8
6.2. Fragment Acknowledgement Dispatch type and Header	8
7. Fragments Recovery	10
8. Forwarding Fragments	12
8.1. Upon the first fragment	12
8.2. Upon the next fragments	13
8.3. Upon the fragment acknowledgements	14
9. Security Considerations	14
10. IANA Considerations	14
11. Acknowledgments	14
12. References	15
12.1. Normative References	15
12.2. Informative References	15
Authors' Addresses	15

1. Introduction

In many 6LoWPAN applications, the majority of traffic is spent sending small chunks of data (order few bytes to few tens of bytes) per packet. Given that an 802.15.4 frame can carry 74 bytes or more in all cases, fragmentation is often not needed for most application traffic. However, many applications also require occasional bulk data transfer capabilities to support firmware upgrades of 6LoWPAN devices or extraction of logs from 6LoWPAN devices. In the former case, bulk data is transferred to the 6LoWPAN device, and in the latter, bulk data flows away from the 6LoWPAN device. In both cases, the bulk data size is often on the order of 10K bytes or more and end-to-end reliable transport is required.

Mechanisms such as TCP or application-layer segmentation will be used to support end-to-end reliable transport. One option to support bulk data transfer over 6LoWPAN links is to set the Maximum Segment Size to fit within the 802.15.4 MTU. Doing so, however, can add significant header overhead to each 802.15.4 frame. This causes the end-to-end transport to be aware of the delivery properties of 6LoWPAN networks, which is a layer violation.

An alternative mechanism combines the use of 6LoWPAN fragmentation in addition to transport or application-layer segmentation. Increasing the Maximum Segment Size reduces header overhead by the end-to-end transport protocol. It also encourages the transport protocol to reduce the number of outstanding datagrams, ideally to a single datagram, thus reducing the need to support out-of-order delivery common to 6LoWPAN networks.

[RFC4944] defines a datagram fragmentation mechanism for 6LoWPAN networks. However, because [RFC4944] does not define a mechanism for recovering fragments that are lost, datagram forwarding fails if even one fragment is not delivered properly to the next IP hop. End-to-end transport mechanisms will require retransmission of all fragments, wasting resources in an already resource-constrained network.

Past experience with fragmentation has shown that missassociated or lost fragments can lead to poor network behavior and, eventually, trouble at application layer. The reader is encouraged to read [RFC4963] and follow the references for more information. That experience led to the definition of the Path MTU discovery [RFC1191] protocol that limits fragmentation over the Internet.

For one-hop communications, a number of media propose a local acknowledgement mechanism that is enough to protect the fragments. In a multihop environment, an end-to-end fragment recovery mechanism

might be a good complement to a hop-by-hop MAC level recovery. This draft introduces a simple protocol to recover individual fragments between 6LoWPAN endpoints. Specifically in the case of UDP, valuable additional information can be found in UDP Usage Guidelines for Application Designers [RFC5405].

2. Terminology

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].

Readers are expected to be familiar with all the terms and concepts that are discussed in "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals" [RFC4919] and "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

ERP

Error Recovery Procedure.

LoWPAN endpoints

The LoWPAN nodes in charge of generating or expanding a 6LoWPAN header from/to a full IPv6 packet. The LoWPAN endpoints are the points where fragmentation and reassembly take place.

3. Rationale

There are a number of uses for large packets in Wireless Sensor Networks. Such usages may not be the most typical or represent the largest amount of traffic over the LoWPAN; however, the associated functionality can be critical enough to justify extra care for ensuring effective transport of large packets across the LoWPAN.

The list of those usages includes:

Towards the LoWPAN node:

Packages of Commands: A number of commands or a full configuration can be packaged as a single message to ensure consistency and enable atomic execution or complete roll back. Until such commands are fully received and interpreted, the intended operation will not take effect.

Firmware update: For example, a new version of the LoWPAN node software is downloaded from a system manager over unicast or multicast services. Such a reflashing operation typically involves updating a large number of similar 6LoWPAN nodes over a relatively short period of time.

From the LoWPAN node:

Waveform captures: A number of consecutive samples are measured at a high rate for a short time and then transferred from a sensor to a gateway or an edge server as a single large report.

Data logs: 6LoWPAN nodes may generate large logs of sampled data for later extraction. 6LoWPAN nodes may also generate system logs to assist in diagnosing problems on the node or network.

Large data packets: Rich data types might require more than one fragment.

Uncontrolled firmware download or waveform upload can easily result in a massive increase of the traffic and saturate the network.

When a fragment is lost in transmission, all fragments are resent, further contributing to the congestion that caused the initial loss, and potentially leading to congestion collapse.

This saturation may lead to excessive radio interference, or random early discard (leaky bucket) in relaying nodes. Additional queuing and memory congestion may result while waiting for a low power next hop to emerge from its sleeping state.

To demonstrate the severity of the problem, consider a fairly reliable 802.15.4 frame delivery rate of 99.9% over a single 802.15.4 hop. The expected delivery rate of a 5-fragment datagram would be about 99.5% over a single 802.15.4 hop. However, the expected delivery rate would drop to 95.1% over 10 hops, a reasonable network diameter for 6LoWPAN applications. The expected delivery rate for a 1280-byte datagram is 98.4% over a single hop and 85.2% over 10 hops.

Considering that the IPv6 minimum MTU is 1280 bytes and that a 802.15.4 frame can limit the MAC payload to as little as 74 bytes, a packet might be fragmented into at least 18 fragments at the 6LoWPAN shim layer. Taking into account the worst-case header overhead for 6LoWPAN Fragmentation and Mesh Addressing headers will increase the number of required fragments to around 32. This level of fragmentation is much higher than that traditionally experienced over the Internet with IPv4 fragments. At the same time, the use of radios increases the probability of transmission loss and Mesh-Under

techniques compound that risk over multiple hops.

4. Requirements

This paper proposes a method to recover individual fragments between LoWPAN endpoints. The method is designed to fit the following requirements of a LoWPAN (with or without a Mesh-Under routing protocol):

Number of fragments

The recovery mechanism must support highly fragmented packets, with a maximum of 32 fragments per packet.

Minimum acknowledgement overhead

Because the radio is half duplex, and because of silent time spent in the various medium access mechanisms, an acknowledgement consumes roughly as many resources as data fragment.

The recovery mechanism should be able to acknowledge multiple fragments in a single message and not require an acknowledgement at all if fragments are already protected at a lower layer.

Controlled latency

The recovery mechanism must succeed or give up within the time boundary imposed by the recovery process of the Upper Layer Protocols.

Support for out-of-order fragment delivery

A Mesh-Under load balancing mechanism such as the ISA100 Data Link Layer can introduce out-of-sequence packets.

The recovery mechanism must account for packets that appear lost but are actually only delayed over a different path.

Optional congestion control

The aggregation of multiple concurrent flows may lead to the saturation of the radio network and congestion collapse.

The recovery mechanism should provide means for controlling the number of fragments in transit over the LoWPAN.

5. Overview

Considering that a multi-hop LoWPAN can be a very sensitive environment due to the limited queuing capabilities of a large population of its nodes, this draft recommends a simple and conservative approach to congestion control, based on TCP congestion avoidance.

From the standpoint of a source LoWPAN endpoint, an outstanding fragment is a fragment that was sent but for which no explicit acknowledgment was received yet. This means that the fragment might be on the way, received but not yet acknowledged, or the acknowledgment might be on the way back. It is also possible that either the fragment or the acknowledgment was lost on the way.

Because a meshed LoWPAN might deliver frames out of order, it is virtually impossible to differentiate these situations. In other words, from the sender standpoint, all outstanding fragments might still be in the network and contribute to its congestion. There is an assumption, though, that after a certain amount of time, a frame is either received or lost, so it is not causing congestion anymore. This amount of time can be estimated based on the round trip delay between the LoWPAN endpoints. The method detailed in [RFC2988] is recommended for that computation.

6. New Dispatch types and headers

This specification extends "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944] with 4 new dispatch types, for Recoverable Fragments (RFRAG) headers with or without Acknowledgment Request, and for the Acknowledgment back.

Pattern	Header Type
11 101000	RFRAG - Recoverable Fragment
11 101001	RFRAG-AR - RFRAG with Ack Request
11 10101x	RFRAG-ACK - RFRAG Acknowledgment

Figure 1: Additional Dispatch Value Bit Patterns

In the following sections, the semantics of "datagram_tag," "datagram_offset" and "datagram_size" and the reassembly process are changed from [RFC4944] Section 5.3. "Fragmentation Type and Header." The size and offset are expressed on the compressed packet as opposed to the uncompressed form.

6.1. Recoverable Fragment Dispatch type and Header

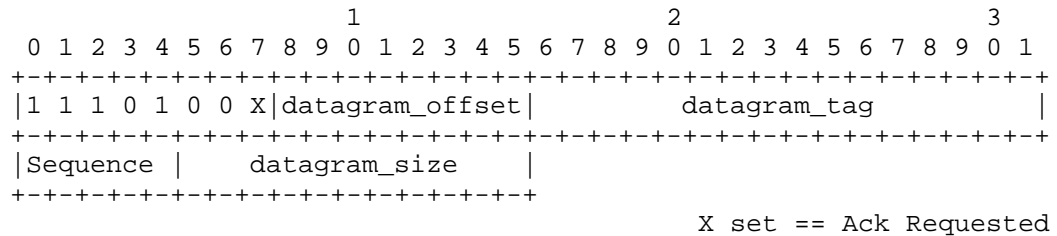


Figure 2: Recoverable Fragment Dispatch type and Header

X bit

When set, the sender requires an Acknowledgment from the receiver

Sequence

The sequence number of the fragment. Fragments are numbered [0..N] where N is in [0..31].

6.2. Fragment Acknowledgement Dispatch type and Header

The specification also defines an acknowledgement bitmap that is used to carry selective acknowledgements for the received fragments. A given offset in the bitmap maps one to one with a given sequence number.

The bitmap is compressed as a variable length field formed by control bits and acknowledgement bits. The leftmost bits of the compressed form are control bits up to the first 0. The rest is ack bits encoded right to left:

Pattern	Size	Ack
0XXXXXXXX	1 octet	1 -> 7
10XXXXXX XXXXXXXX	2 octets	1 -> 14
110XXXXX XXXXXXXX XXXXXXXX	3 octets	1 -> 21
1110XXXX XXXXXXXX XXXXXXXX XXXXXXXX	4 octets	1 -> 28

Figure 3: Compressed acknowledgement bitmap encoding

The highest sequence number to be acknowledged determines the pattern to be used. The format can be extended for more fragments in the future but this specification only requires the support of up to 4 octets encoding, which enables to acknowledge up to 28 fragments.

A 32 bits uncompressed bitmap is obtained by prepending zeroes to the XXX in the pattern above. For instance:

```

  0 1 2 3 4 5 6 7
+-----+
|0|1|1|0|1|1|1|1|  is expanded as:
+-----+

          1                      2                      3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|1|1|0|1|1|1|1|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 4: Expanding 1 octet encoding

and

```

          1                      2
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-----+-----+-----+-----+-----+-----+
|1|1|0|1|1|1|1|0|1|1|1|1|1|1|1|1|1|1|1|1|0|0|1|  is expanded as:
+-----+-----+-----+-----+-----+-----+

          1                      2                      3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|0|0|0|0|0|0|0|0|0|0|0|0|1|1|1|1|0|1|1|1|1|1|1|1|1|1|1|1|1|0|0|1|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Figure 5: Expanding 3 octets encoding

whereas the 4 octets encoding is expanded by simply setting the first 3 bits to 0. The 32 bits uncompressed bitmap is written and read as follows:

```

          1                      2                      3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|               Acknowledgment Bitmap               |
+-----+-----+-----+-----+-----+-----+-----+-----+
                                     ^               ^
      bitmap indicating whether:      |               |
      Fragment with sequence 10 was received --+      |
      Fragment with sequence 00 was received -----+

```

Figure 6: Expanded bitmap encoding

So in the example in Figure 5 it appears that all fragments from sequence 0 to 20 were received but for sequence 1, 2 and 16 that were either lost or are still in the network over a slower path.

The compressed form of the acknowledgement bitmap is carried in a Fragment Acknowledgement as follows:

```

                                1                2                3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-----+-----+-----+-----+-----+-----+-----+
      |1 1 1 0 1 0 1 Y|          datagram_tag          |
+-----+-----+-----+-----+-----+-----+-----+
| Compressed Acknowledgment Bitmap (8 to 32 bits)
+-----+-----+-----+

```

Figure 7: Fragment Acknowledgement Dispatch type and Header

Y bit

Reserved for Explicit Congestion Notification (ECN) signalling

Compressed Acknowledgement Bitmap

An encoded form of an acknowledgement bitmap.

7. Fragments Recovery

The Recoverable Fragments header RFRAG and RFRAG-AR deprecate the original fragment headers from [RFC4944] and replace them in the fragmented packets. The Fragment Acknowledgement RFRAG-ACK is introduced as a standalone header in message that is sent back to the fragment source endpoint as known by its MAC address. This assumes that the source MAC address in the fragment (is any) and datagram_tag are enough information to send the Fragment Acknowledgement back to the source fragmentation endpoint.

The node that fragments the packets at 6LoWPAN level (the sender) controls the Fragment Acknowledgements. It may do that at any fragment to implement its own policy or perform congestion control which is out of scope for this document. When the sender of the fragment knows that an underlying mechanism protects the Fragments already it MAY refrain from using the Acknowledgement mechanism, and never set the Ack Requested bit. The node that recomposes the packets at 6LoWPAN level (the receiver) MUST acknowledge the fragments it has received when asked to, and MAY slightly defer that acknowledgement.

The sender transfers a controlled number of fragments and MAY flag the last fragment of a series with an acknowledgment request. The receiver MUST acknowledge a fragment with the acknowledgment request bit set. If any fragment immediately preceding an acknowledgment

request is still missing, the receiver MAY intentionally delay its acknowledgment to allow in-transit fragments to arrive. delaying the acknowledgement might defeat the round trip delay computation so it should be configurable and not enabled by default.

The receiver interacts with the sender using an Acknowledgment message with a bitmap that indicates which fragments were actually received. The bitmap is a 32bit SWORD, which accommodates up to 32 fragments and is sufficient for the 6LoWPAN MTU. For all n in $[0..31]$, bit n is set to 1 in the bitmap to indicate that fragment with sequence n was received, otherwise the bit is set to 0. All zeroes is a NULL bitmap that indicates that the fragmentation process was cancelled by the receiver for that datagram.

The receiver MAY issue unsolicited acknowledgments. An unsolicited acknowledgment enables the sender endpoint to resume sending if it had reached its maximum number of outstanding fragments or indicate that the receiver has cancelled the process of an individual datagram. Note that acknowledgments might consume precious resources so the use of unsolicited acknowledgments should be configurable and not enabled by default.

The sender arms a retry timer to cover the fragment that carries the Acknowledgment request. Upon time out, the sender assumes that all the fragments on the way are received or lost. The process must have completed within an acceptable time that is within the boundaries of upper layer retries. The method detailed in [RFC2988] is recommended for the computation of the retry timer. It is expected that the upper layer retries obey the same or friendly rules in which case a single round of fragment recovery should fit within the upper layer recovery timers.

Fragments are sent in a round robin fashion: the sender sends all the fragments for a first time before it retries any lost fragment; lost fragments are retried in sequence, oldest first. This mechanism enables the receiver to acknowledge fragments that were delayed in the network before they are actually retried.

When the sender decides that a packet should be dropped and the fragmentation process canceled, it sends a pseudo fragment with the `datagram_offset`, `sequence` and `datagram_size` all set to zero, and no data. Upon reception of this message, the receiver should clean up all resources for the packet associated to the `datagram_tag`. If an acknowledgement is requested, the receiver responds with a NULL bitmap.

The receiver might need to cancel the process of a fragmented packet for internal reasons, for instance if it is out of recomposition

buffers, or considers that this packet is already fully recomposed and passed to the upper layer. In that case, the receiver SHOULD indicate so to the sender with a NULL bitmap. Upon an acknowledgement with a NULL bitmap, the sender MUST drop the datagram.

8. Forwarding Fragments

This specification enables intermediate routers to forward fragments with no intermediate reconstruction of the entire packet. Upon the first fragment, the routers lay an label along the path that is followed by that fragment (that is IP routed), and all further fragments are label switched along that path. As a consequence, alternate routes not possible for individual fragments. The datagram tag is used to carry the label, that is swapped at each hop.

8.1. Upon the first fragment

In route over the L2 source changes at each hop. The label that is formed and placed in the datagram tag is associated to the source MAC and only valid (and unique) for that source MAC. Say the first fragment has:

Source IPv6 address = IP_A (maybe hops away)

Destination IPv6 address = IP_B (maybe hops away)

Source MAC = MAC_prv (prv as previous)

Datagram_tag= DT_prv

The intermediate router that forwards individual fragments does the following:

a route lookup to get Next hop IPv6 towards IP_B, which resolves as IP_nxt (nxt as next)

a ND resolution to get the MAC address associated to IP_nxt, which resolves as MAC_nxt

Since it is a first fragment of a packet from that source MAC address MAC_prv for that tag DT_prv, the router:

cleans up any leftover resource associated to the tuple (MAC_prv, DT_prv)

allocates a new label for that flow, DT_nxt, from a Least Recently Used pool or some similar procedure.

allocates a Label swap structure indexed by (MAC_prv, DT_prv) that contains (MAC_nxt, DT_nxt)

allocates a Label swap structure indexed by (MAC_nxt, DT_nxt) that contains (MAC_prv, DT_prv)

swaps the MAC info to from self to MAC_nxt

Swaps the datagram_tag to DT_nxt

At this point the router is all set and can forward the packet to nxt.

8.2. Upon the next fragments

Upon next fragments (that are not first fragment), the router expects to have already Label swap structure indexed by (MAC_prv, DT_prv). The router:

lookups up the Label swap entry for (MAC_prv, DT_prv), which resolves as (MAC_nxt, DT_nxt)

swaps the MAC info to from self to MAC_nxt;

Swaps the datagram_tag to DT_nxt

At this point the router is all set and can forward the packet to nxt.

if the Label swap entry for (MAC_src, DT_src) is not found, the router builds an RFRAG-ACK to indicate the error. The acknowledgment message has the following information:

MAC info set to from self to MAC_prv as found in the fragment

Swaps the datagram_tag set to DT_prv

Bitmap of all zeroes to indicate the error

At this point the router is all set and can send the RFRAG-ACK back to the previous router.

8.3. Upon the fragment acknowledgements

Upon fragment acknowledgements next fragments (that are not first fragment), the router expects to have already Label swap structure indexed by (MAC_nxt, DT_nxt). The router:

lookups up the Label swap entry for (MAC_nxt, DT_nxt), which resolves as (MAC_prv, DT_prv)

swaps the MAC info to from self to MAC_prv;

Swaps the datagram_tag to DT_prv

At this point the router is all set and can forward the RFRAG-ACK to prv.

if the Label swap entry for (MAC_nxt, DT_nxt) is not found, it simply drops the packet.

if the RFRAG-ACK indicates either an error or that the fragment was fully receive, the router schedules the Label swap entries for recycling. If the RFRAG-ACK is lost on the way back, the source may retry the last fragments, which will result as an error RFRAG-ACK from the first router on the way that has already cleaned up.

9. Security Considerations

The process of recovering fragments does not appear to create any opening for new threat compared to "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

10. IANA Considerations

Need extensions for formats defined in "Transmission of IPv6 Packets over IEEE 802.15.4 Networks" [RFC4944].

11. Acknowledgments

The author wishes to thank Jay Werb, Christos Polyzois, Soumitri Kolavennu and Harry Courtice for their contribution and review.

12. References

12.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2988] Paxson, V. and M. Allman, "Computing TCP's Retransmission Timer", RFC 2988, November 2000.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, September 2007.

12.2. Informative References

- [I-D.ietf-roll-rpl]
Winter, T., Thubert, P., and R. Team, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks",
draft-ietf-roll-rpl-08 (work in progress), May 2010.
- [I-D.mathis-frag-harmful]
Mathis, M., "Fragmentation Considered Very Harmful",
draft-mathis-frag-harmful-00 (work in progress),
July 2004.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
November 1990.
- [RFC4919] Kushalnagar, N., Montenegro, G., and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals",
RFC 4919, August 2007.
- [RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, July 2007.
- [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405,
November 2008.

Authors' Addresses

Pascal Thubert (editor)
Cisco Systems
Village d'Entreprises Green Side
400, Avenue de Roumanille
Batiment T3
Biot - Sophia Antipolis 06410
FRANCE

Phone: +33 4 97 23 26 34
Email: pthubert@cisco.com

Jonathan W. Hui
Arch Rock Corporation
501 2nd St. Ste. 410
San Francisco, California 94107
USA

Phone: +415 692 0828
Email: jhui@archrock.com

