RPL Border Router Redundancy in the Internet of Things

Quang-Duy Nguyen, Julien Montavont, Nicolas Montavont, Thomas Noël

RPL Border Router Redundancy in the Internet of Things

Quang-Duy Nguyen[1], Julien Montavont[1], Nicolas Montavont[2] and Thomas Noël[1]

[1] ICube laboratory (CNRS), University of Strasbourg, France
{qdnguyen,montavont,noel}@unistra.fr
[2] Institut Mines-Telecom / Telecom Bretagne, Rennes France
nicolas.montavont@telecom-bretagne.eu

Abstract The Internet of Things (IoT) refers to a broad variety of objects with communication capabilities that are integrated into Internet. The interconnection between those objects and the Internet is enabled thanks to border routers. In this article, we investigate the aftermath of the failure of border routers on ongoing communications. Next, we propose to overcome the exposed problems by providing objects with multiple border routers. The corresponding subnet is therefore multihomed, i.e. all objects in this subnet are reachable via multiple paths, one per active border router. Whenever a border router fails, we dynamically re-route traffic to an active border router. Such flows redirection remains transparent to remote peers. Our solution, referred to as Syn-RPL, is based on the well-known IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL). Syn-RPL is evaluated through experimentations on a real testbed.

Keywords: Internet of Things, 6LoWPAN, RPL, Multihoming, Failover.

1 Introduction

In the recent years, the rapid development of low-power wireless technologies together with the miniaturization of electronic components gave birth to what we commonly call the Internet of Things (IoT). The IoT refers to a set of physical objects (ranging from sensors to common household electrical goods) with communication capabilities that are able to collect, exchange and receive information throughout the Internet. The IoT enables a large variety of new applications, ranging from scientific observations [16] to personal home automation [14].

In the IoT, objects in a given neighborhood use their wireless communication capabilities to form a multihop wireless network known as Low-power and Lossy wireless Network (LLN). Such networks are characterized by a variety of lossy links (low speed, low energy consumption and unstable connectivity) and constrained devices (limited computational power, memory and energy). Interconnecting LLNs with the Internet is made possible by the IPv6 over Low power

Wireless Personal Area Network (6LoWPAN) IETF standard [10]. 6LoWPAN introduces IPv6 header compression and provides a fragmentation and reassembly adaptation layer below IP, enabling the transport of IPv6 packets over LLNs. IPv6 packets originated from or destined to a LLN are processed by the 6LoWPAN Border Router (BR) [17]. This entity is located at the junction between the LLN and the IPv6 Internet and is responsible to compress/decompress or fragment/defragment IPv6 packets regarding the 6LoWPAN standard before forwarding them towards the destination. Inside a LLN, packets are routed with the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [15]. RPL builds a Destination Oriented Directed Acyclic Graph (DODAG) rooted at the BR. As a result, all the traffic between the LLN and the IPv6 Internet goes through the BR.

Similarly to wired IPv6 networks with access routers, the IPv6 connectivity of each smart object is therefore directly dependent of the BR status. Whenever the BR becomes unreachable (as a result of system failure, congestion due to funneling effect [6], lack of connectivity due to power outage on neighbor nodes, etc.) the whole LLN is disconnected from the Internet, terminating all ongoing communications with no possibilities to start new ones. In this article, we address such issue by providing LLNs with multiple BRs. In addition to increasing the overall network bandwidth and coverage (the overall throughput increases linearly with the number of egress points), the cooperation of multiple BRs enable a failover mechanism to prevent network disconnection. Our proposal, referred to as Syn-RPL, extends RPL and introduces a virtual BR that federates each graph rooted at a single BR into a unique DODAG. In addition, Syn-RPL only extends the BR and does not require additional software on leaf or intermediate nodes located inside the LLN. Syn-RPL was implemented in Contiki OS and evaluated throughout an extensive experimentation campaign. The obtained results show that Syn-RPL allows smart objects to remain connected to the Internet even after the failure of BR. We show that the traffic redirection from one BR to another is almost transparent for the remote hosts.

The rest of the paper is organized as follows. Next, we present the motivations and advantages of a LLN served by multiple BRs. Section 3 presents solutions currently available in the literature that consider multiple BRs. Then, our proposal referred to as Syn-RPL is introduced in Section 4, followed by an overview of the experimentation campaign and performance analysis. Finally, conclusions and future work are presented in Section 6.

2 Problem statement

In constrained environments, routing is usually provided with the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [15]. RPL builds a Destination Oriented Directed Acyclic Graph (DODAG) rooted at the border router (BR) of the network. The DODAG is shaped according to link metric(s) and an objective function which define how to compute the paths. Each node periodically broadcasts a DODAG Information Object (DIO) message to announce a

potential attachment to the DODAG. When a node receives a DIO, it updates a list of potential next hops to the BR, also known as the parent set, and select a preferred parent from this set based on the objective function and the link metric. A rank is also computed, giving the relative position of node in the DODAG. The preferred parent will serve as the next hop in the default IPv6 route toward the BR. Nodes can also solicit the transmission of DIO by sending DODAG Information Solicitation (DIS) messages. RPL supports point-to-point and point-to-multipoint communications using DODAG Destination Advertisement Object (DAO) messages. After computing its rank, a node can send DAO to its preferred parent in order to advertise a new downward destination. In non-storing mode, where source routing is used, DAO are simply propagated towards the BR. In storing mode, nodes store routing table entries for destinations learned from DAOs. DAO are therefore forwarded upward until reaching a node for which the advertised destination is already known.

Due to the broadcasting nature of the wireless communications and because the node density can be important, there is generally a multitude of paths towards the BR. RPL can take advantage of such situation by using alternatively or simultaneously multiple paths [3]. Whenever a node fails, RPL could be able to detect this failure using unreachability detection mechanisms [5] and compute alternative routes that bypass this node. However, there is currently no solutions to recover from a situation where the BR itself becomes unreachable. Such situation could be the result of a system failure of the BR itself, a serious congestion occurring at the BR due to funneling effect [6] (all upward traffic and point-to-point traffic in non-storing mode are routed towards the BR) or a lack of connectivity (all BR's neighbors experience a power outage). For readability reasons, we will refer to one of these causes by the terms *BR failure* in the rest of the article. In RPL, and more generally in the IoT, the BR represents a single point of failure for the LLN located behind. When the BR fails, all nodes in the LLN are affected as all ongoing communications with remote pairs are instantly broken and no new communication could be initiated. A failure of the BR also results in breaking local communications, especially in non-storing mode of RPL. One solution to resolve those problems is to provide a LLN with multiple BRs. Deploying multiple BRs not only allow to have alternative paths to the Internet when one of the BR fails, but it also provides load sharing through multiple egress interfaces towards the Internet. However, how RPL can be extended to efficiently support multiple BRs? In the next section, we investigate the solutions currently available in the literature before introducing our own contribution called Syn-RPL.

3 Related work

In IP networks, routers are in charge of forwarding data packets along networks towards their final destinations. Furthermore, access routers enable the interconnection of local networks to the Internet. For this reason, the failure of an access router leads to the disconnection of the hosts located behind this router. In static

networks, one of the standard solutions for solving this problem is the Virtual Router Redundancy Protocol (VRRP) [9]. VRRP allows the deployment of one master router and several backup routers. Whenever the master router fails, one of the backup routers dynamically takes in charge the forwarding responsibility. However, VRRP makes a heavy use of multicast communications which are not desirable in LLNs due to energy conservation.

Another common solution is to use multiple routers simultaneously [13]. The corresponding subnet is said to be multihomed, i.e. all hosts in this subnet are reachable via multiple paths, one per active router. Such situation allows redundancy (if a router fails a host can use one of the other active routers) and load sharing (traffic can be distributed among all active routers). Again, such solutions are not adapted to the characteristics of LLNs. The subnet concept does not apply in LLNs that are composed of a large number of overlapping radio ranges, forming a complex Non-Broadcast Multiple Access (NBMA).

In LLN, there are many proposals that take advantage from multiple BRs (also referred to as sinks in the literature). Increasing the number of sinks allows increasing the lifetime of the network together with the reduction of the number of hops towards a sink and load sharing [2,8,11]. However, LLNs with multiple sinks require complex signaling protocols to operate [4]. By contrast, RPL [15] is designed to operate either as a single DODAG with a single root, as multiple uncoordinated DODAGs with independent roots or as a single DODAG with a virtual root that coordinates multiple BRs. However, the coordination between multiple BRs is not yet defined by the IETF. Nevertheless, the authors of [7] study the usage of a RPL virtual root together with multiple BRs. They show that using such an architecture allows reducing the energy consumption (by a factor of 30%) and reducing packet loss (by a factor of 39%). However, this solution does not address the failure of one BR and how incoming and outgoing packets can be re-routed to another active BR.

The present article focuses on a failover solution to prevent nodes disconnection whenever a BR fails in the context of LLNs. We will see that our solution also proposes load balancing between the BRs. By contrast to [7], our solution considers multiple BRs that can be connected to the Internet independently via the same or different access networks.

4 Contribution

This section presents our contribution, referred to as Syn-RPL, which consists in extending RPL with multiple BR support. In stock RPL, deploying multiple BRs in the same area will result in multiple DODAGs that are independent from each others. Nodes may attach to each DODAG but switching from one BR to another (in case of a BR failure) will not be seamless as BRs do not necessarily share the same IPv6 prefix (BRs can interconnect the LLN via different access networks). Once a node changes its default BR, it is also required to change its IPv6 address to the one operated by the new BR (in order to avoid ingress

filtering). In addition, all incoming packets destined to its previous IPv6 address are still routed to the old BR, resulting in packet loss.

To overcome those problems, Syn-RPL uses the virtual node (VR) introduced by RPL [15]. The VR acts as the unique root of all branches anchored at each BR. In the following, we will call these branches sub-DODAG. All cooperating BRs will therefore construct a single DODAG rooted at the VR instead of creating their own DODAG. Each node will select the best BR using legacy RPL operations, but whatever its choice, each node belongs to the same (and unique) DODAG. As a result, the traffic load is automatically shared as each packet will be forwarded towards the closest (regarding the objective function and metrics used by RPL) BR. Communications between nodes belonging to different sub-DODAGs are considered as off-link traffic and nodes use their default BR to forward such traffic over the Internet. With this set up, a BR failure is managed almost just as a node failure in a more standard RPL DODAG. Orphan nodes simply re-attach to another sub-DODAG and their traffic is redirected via the new selected BR.

Obviously, the cooperating BRs should share some information to build a unique DODAG. Syn-RPL introduces a new entity known as the Anchor Agent (AA). The AA has all parameters required to build the DODAG and sends them to each cooperating BR whenever necessary. To maintain the IPv6 connectivity when one or more BRs become unavailable, the AA also acts as a relay station to forward the traffic destined to or originated from the LLN. By default, all IPv6 prefixes used by the BRs inside the LLN are routed towards the AA. As a result, connectivity is maintained between the AA and each BR through a bi-directional IPv6-in-IPv6 tunnel created during the bootstrap. Upon the failure of a specific BR, the AA will update the endpoint address of the corresponding tunnel with one of the BR still operating this DODAG. The AA is located in a more standard IPv6 link and is therefore not as prone to failure as BR (no energy constraints on neighbors, no contention thanks to wire links, etc.). Nevertheless, the AA can be a single point of failure but we can adapt well-known failover mechanisms (such as VRRP [9]) to allow a backup AA to take over and continue providing service to LLN. This is a part of our future work. All Syn-RPL operations are carried out using two new messages referred to as *register* and *register acknowledgment*. Fig. 1 illustrates the Syn-RPL framework.

4.1 Bootstrap operations

At bootstrap, the AA is pre-configured with all the necessary parameters to shape DODAGs. These parameters include for each DODAG, the RPL Instance ID (a unique identifier), the DODAG ID (the identifier of the DODAG root, i.e. the VR in Syn-RPL), the DODAG Version Number (the current version of the DODAG), the objective function (i.e. how RPL nodes select and optimize routes within a RPL Instance), the routing metric, the lifetime (the maximal duration during which a BR is considered as reachable), a list of BR identifiers (e.g. the EUI-64 of each BR that belongs to this DODAG), one or more IPv6 prefix(es) to delegate and a shared secret (to authenticate an authorized BR to

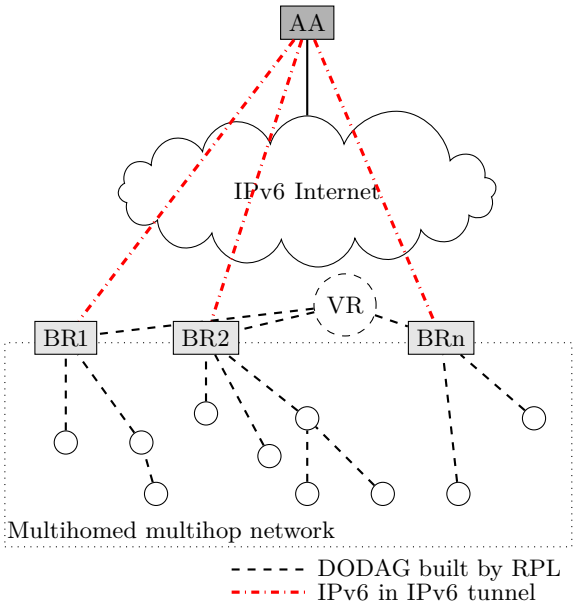

Figure 1: Framework of Syn-RPL

be part of this DODAG). Note that the (RPL Instance ID, DODAG ID, DODAG Version Number) tuple uniquely identifies a DODAG version [15]. This is why all cooperating BRs must share those parameters. An AA that manages multiple LLNs should be pre-configured with several 9-tuples, each corresponding to a specific LLN. The AA also maintains a registration cache in which it records, for each managed LLN, the BRs currently operating the network.

On the BR side, each BR is pre-configured with its identifier (e.g. the EUI-64), the IPv6 address of the AA and the shared secret corresponding to the DODAG the BR belongs to. At bootstrap, a BR sends a *register* message to the AA in order to retrieve the necessary information to start building the DODAG. Register message includes the identifier of the BR, the current IPv6 address of the BR and the shared secret. Upon reception, the AA looks up in its database to retrieve the DODAG information corresponding to this BR. If a corresponding entry is found and the provided shared secret is valid, the AA proceeds to the registration of this BR. First, it sends back a *register acknowledgment* that includes the registration status (successful, rejected). In the case of a successful registration, the register acknowledgment also includes all the parameters to build/expand the DODAG (the RPL Instance ID, the DODAG ID, the DODAG Version Number and the lifetime) together with the IPv6 prefix to use for IPv6 auto-configuration inside the LLN. The AA also adds a new entry in the registration cache containing the identifier and IPv6 address of the BR together with the delegated IPv6 prefix. Each registration cache entry is only valid for a period of time. As a result, BRs periodically send new register messages before

the expiration of the lifetime period, otherwise the entry is removed. Finally the BR and the AA set up a bi-directional IPv6-in-IPv6 tunnel. In the same time, the BR starts building/expanding the DODAG with the information provided by the AA. Once the bootstrap is complete, packets sent to the delegated prefix are routed to the AA which forwards them to the BR via the tunnel. When packets reach the BR, they are routed normally to their final destination with RPL. Packets originated from the LLN are also tunneled via the AA before being forwarded to their final destination. The bootstrap phase of Syn-RPL is illustrated on Fig. 2.

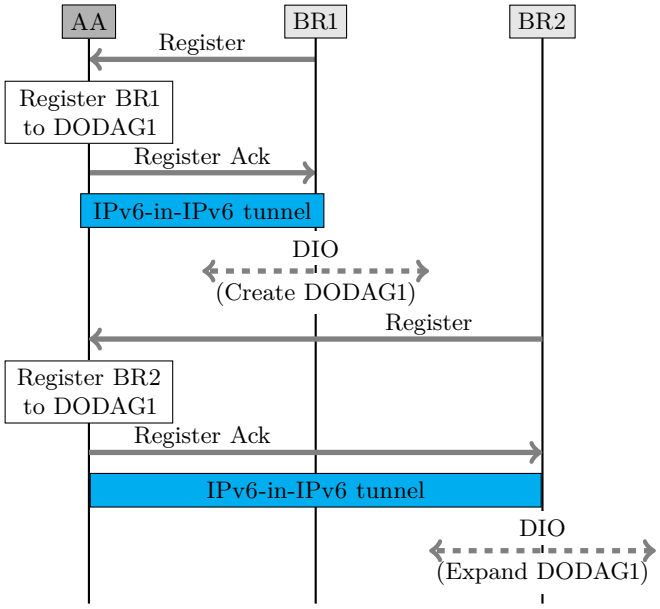

Figure 2: Syn-RPL operations at bootstrap

4.2 Failover operations

Once a BR fails, the nodes located in its sub-DODAG should first detect that their default BR is no longer reachable. The nodes directly connected to this BR will likely be the first to detect its failure with the use of external unreachability detection mechanisms [5]. Upon unreachability confirmation, these nodes will send new DIO messages advertising an infinite rank in order to poison routes towards the failed BR. All nodes in the failed sub-DODAG can now accept new DIO from nodes that belongs to another sub-DODAG. Upon reception of such DIO, a node re-attach to the DODAG and sends a DAO to advertise a new node destination information to the BR in charge of the related sub-DODAG. It is

important to note that a node will keep its IPv6 address when changing BR, even if its prefix does not match the one used in its new sub-DODAG. The new BR will inform the AA to delegate the corresponding IPv6 prefix, and use the existing tunnel between the AA and the BR to forward this traffic. For this, the BR sends a new register message including the new IPv6 prefix(es) to delegate. Upon reception, the AA will update all tunnel endpoints related to the failed BR to the requesting BR to reflect the new organization of the DODAG. Next, the AA sends back a register acknowledgment to the requesting BR, which in turn updates its own tunnel to the AA. From now on, all traffic destined to or originated from the IPv6 prefix used by the failed BR will be routed to the new BR in charge of this prefix. Fig.3 illustrates Syn-RPL operations when a BR fails. Note that if nodes that were using the same IPv6 prefix attach to different BR, /128 prefixes might be used to add routes for each of these nodes.

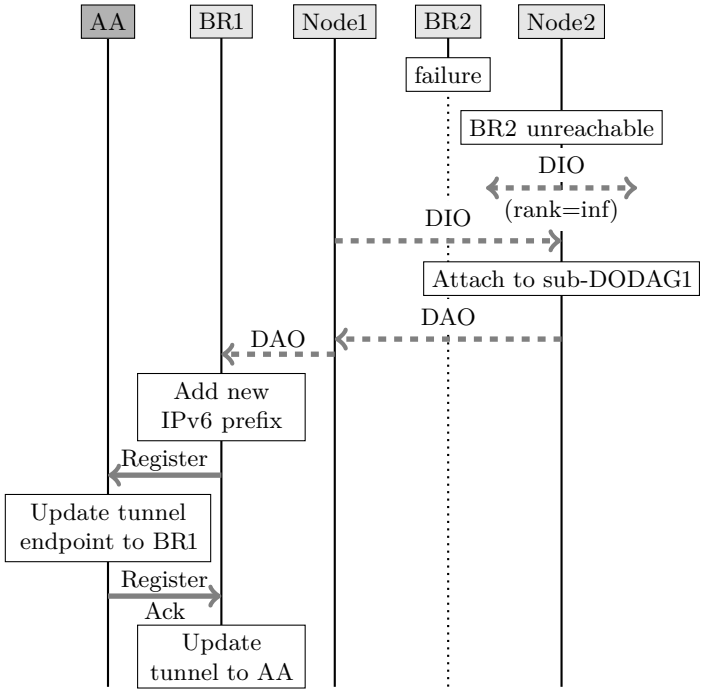

Figure 3: Syn-RPL operations upon BR failure

5 Experimentation campaign and results

5.1 Implementation and platform specifications

The Syn-RPL framework involves an Anchor Agent (AA) and 6LoWPAN Border Routers (BR) that interconnect Low-power and Lossy wireless Networks (LLN) with the Internet. On the BR side, we implemented Syn-RPL in Contiki (version 3.x). Contiki is an open source operating system designed for embedded systems and wireless sensor networks. This operating system includes an IP network stack in addition to standards dedicated to LLN such as 6LowPAN [10] and RPL [15]. Moreover, a communication serial interface between a Linux box and a Contiki device can be established by using the extra tools provided by Contiki such as tunslip6. For this reason, we can turn a Linux box with a Contiki device into a BR that interconnects a LLN with legacy IPv6 networks. On the AA side, we implemented Syn-RPL for the Linux operating system as a userland application. The application opens a UDP socket and waits for messages from BR. The input parameters (IPv6 prefixes, shared secret, etc.) are retrieved from an XML file. The messages exchanged between the BR and the AA (register and register acknowledgment) are encapsulated in UDP datagrams.

The experimental platform used to evaluate the performance of Syn-RPL includes one IPv6 router, one corresponding node (a Linux box), one AA (a Linux box), two BRs (two Linux boxes with TelosB motes connected through the USB interface) and two wireless motes (two TelosB). TelosB are developed by Crossbow and include a transceiver chipset compliant with the IEEE 802.15.4 standard at the physical and MAC layers. The corresponding node, the AA and each BR are located in different IPv6 networks. The interconnection between those networks is enabled by the IPv6 router of the platform. Routing between BRs and wireless nodes is performed by RPL [15]. At startup, one wireless mote (N_1) joins the sub-DODAG built by the first BR (BR_1) while the second wireless mote (N_2) joins the sub-DODAG built by the second BR (BR_2). All experimental parameters are given in Table 1.

Once the DODAG is set up, the correspondent node starts sending a constant bit-rate traffic to the second wireless mote (N_2). Data packets are routed towards the AA (as the IPv6 prefix used in the sub-DODAG managed by BR_2 is topologically anchored at the AA) before being forwarded via the bi-directional tunnel towards BR_2. Upon reception, BR_2 finally forwards them to N_2.

5.2 Experimentation results

The results presented in this section are an average of the overall data collected over 10 experiment trials. We also calculated the 95% confidence interval for each values to measure the reliability of our measurements. For readability reasons, we do not show the obtained confidence intervals since they were very small.

First, we evaluated the duration of the bootstrapping phase as presented in Fig. 4. In stock RPL, a BR starts sending its first DIO $5.5s$ after its startup. When N_1 receives this DIO, it can attach to the DODAG and sends back a

Parameters	Values
Platform organization	1 AA, 1 correspondent node, 2 BRs and 2 wireless motes
Application model	Constant bit-rate of 8 bytes every second
Syn-RPL	Lifetime 10 seconds
RPL	DIO sending rate fixed by [12] Objective function zero and Min-Hop Storing mode
Phy and Mac	802.15.4@2.4GHz at -25dBm, contention-based

Table 1: Experimentation parameters

DAO to BR_1, which in turn updates its routing table accordingly. Then, the first incoming data packet is forwarded to N_1 at $t = 9.16s$ after the startup of BR_1. With Syn-RPL, a BR should first retrieve the DODAG parameters from the AA by the mean of register and register acknowledgment messages. As we can see on Fig. 4, BR_1 sends the register message right after its startup (at $t = 0.68s$) and receives the register acknowledgment only $740ms$ later. Then, the first incoming data packet is forwarded to N_1 at $t = 10.65s$ after the start up of BR_1. As a result, Syn-RPL only adds $1.5s$ on average as extra bootstrap delay compared to standard RPL. This extra delay is mainly due to the exchange of register and register acknowledgment messages. In our testbed, the RTT between the BRs and the AA is lower than $10ms$ which explains the short registration delay. It is obvious that a larger delay to reach the AA will increase the duration of the Syn-RPL bootstrap phase. However, this delay remains low (within a few seconds) and only occurs at bootstrap when an extra delay does not usually affect IoT applications.

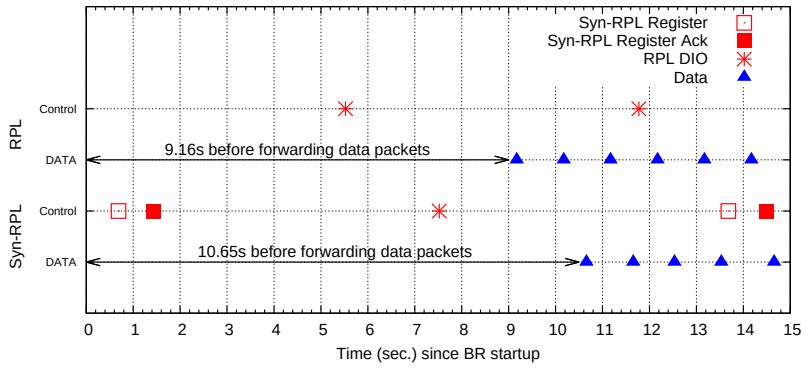

Figure 4: Bootstrap delay

Next, we evaluated the delay needed to redirect flows from a failed BR to an active one. Obtained results are shown on Fig. 5. Each dot represents the transmission or reception of a packet at the time indicated on the Y-axis. At $t = 10s$ we shutdown BR_2 in order to emulate its failure. As we can see, the flow redirection from BR_2 to BR_1 takes approximately $36.7s$ on average. During this period of time, 24 data packets are lost. At approximately $t = 35s$ N_2 detects that BR_2 is unreachable and starts sending DIO with an infinite rank in order to poison routes towards BR_2. For implementation ease, we chose to use a fixed timeout of $25s$ upon reception of DIO to detect that a BR is unreachable. This value represents the non-receipt of two consecutive DIOs in our configuration. So the redirection delay is mainly due to the detection of BR_2 unreachability by the nodes located in its sub-DODAG. Regarding the application model, such delay could be reduced (if necessary) by using one of the solutions proposed in [5]. Next, N_2 receives a fresh DIO from N_1 (attached to sub-DODAG1) at $t = 43.18s$, which allows N_2 to re-attach to the DODAG. Upon reception of the DAO transmitted by N_2, BR_1 updates its routing table and informs the AA of the new destination information by sending a new register message. Upon reception, the AA updates the tunnel end-point for this prefix/destination and sends back a registration acknowledgment. BR_1 starts receiving data packets destined to N_2 at $t = 46.74s$. By contrast, stock RPL is unable to recover from BR_2 failure as nodes located in sub-DODAG2 (created by BR_2) uses an IPv6 prefix different from the one used in sub-DODAG1. Although a node can re-attach to the DODAG once it has detected the failure of its BR, it still needs to change its IPv6 address, which will break all ongoing communications. Remote hosts should be informed of such change in order to re-start their communication towards such nodes. As we have shown, Syn-RPL enables transparent flow redirection from or towards remote hosts in the case of a BR failure.

6 Conclusions and future work

The interconnection between a network composed of objects (referred to as Low-power and Lossy Network - LLN) with the Internet is usually enabled by Border Routers (BR). On their egress interface, BRs act as legacy IP access routers. On their ingress interface, they support the communication stack designed for LLN. As a result, they act as gateways between the IP world and the LLN world. However, BRs introduce single point of failure for the IoT. Whenever the BR becomes unreachable (as a result of system failure, network congestion, lack of connectivity, etc.) all objects located in the corresponding LLN become disconnected from the Internet, breaking all ongoing communications. In this article, we presented Syn-RPL, an extension to RPL [15] that provides a LLN with multiple BRs. Syn-RPL allows transparent load sharing (each mote will automatically attach to a specific sub-graph anchored to a specific BR regarding the objective function and metrics used by RPL) and failover upon BR unreachability confirmation. The failover is provided by an Anchor Agent (AA) which is able to redirect IPv6 prefixes toward different BRs according to their avail-

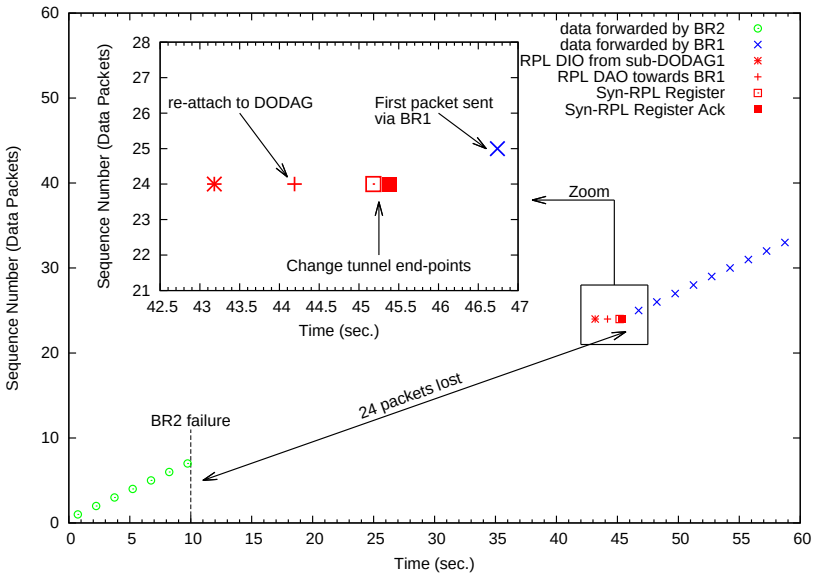

Figure 5: Flow redirection upon failure of BR

ability. Experimentation results showed that Syn-RPL adds a short extra delay at startup (approximately $1.5s$) while it enables fast flow redirection upon BR failure (the disconnection approximately lasts for $37s$). The delay introduced by flow redirection is composed of the delay required to detect the failure of the BR plus the delay required to update the AA in order to update tunnel end-points. Since it is the BR failure detection that represents most of the delay, we could use more reactive trigger if the application requires it [5]. As a result, Syn-RPL allows fast recovery from a BR failure without involving any actions from correspondent nodes.

Encouraged by the results presented here, we plan to further analyze Syn-RPL via large-scale experiments including several practical scenarios. For this, we will use the FIT IoT-Lab experimental platform [1] which will allow us to scale the number of nodes up to a thousand of motes. With this amount of nodes, we will also be able to further study the load balancing property of Syn-RPL. We also plan to study AA redundancy, which is a less sensible router compared to RPL BR. AAs are located in a more standard IPv6 link, and we will develop methods to provide failover mechanisms using backup AAs. Finally, we plan to extend Syn-RPL to support mobile LLNs.

References

1. Future Internet (FIT) - Internet of Things testbed. `http://www.iot-lab.info`

2. A Bogdanov, E Maneva and S Riesenfel: Power-aware base station positioning for sensor networks. In: proc. of the Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) (March 2004)
3. B. Pavkovic and F. Teholeyre and A. Duda: Multipath Opportunistic RPL Routing over IEEE 802.15.4. In: proc. of ACM International Conference on Modeling, Analysis and Simulation of Woreless and Mobile Systems (MSWIM) (July 2011)
4. C. Buratti and A. Conti and D. Dardari and R. Verdone: An Overview on Wireless Sensor Networks Technology and Evolution. Sensors (August 2009)
5. C. Cobarzan and J. Montavont and T. Noel: Integrating Mobility in RPL. In: proc. of the 12th European Conference on Wireless Sensor Networks (EWSN) (February 2015)
6. C.-Y. Wan and S.B. Eisenman and A.T. Campbell and J. Crowcroft: Siphon: overload traffic management using multi-radio virtual sinks in sensor networks. In: proc. of the 3rd ACM International Conference on Embedded Networked Sensor Systems (SenSys'05) (November 2005)
7. D. Carels and N. Derdaele and E. D. Poorter and W. Vandenberghe and I. Moerman and P. Demeester: Support of multiple sinks via a virtual root for the RPL routing protocol. EURASIP Journal on Wireless Communications and Networking (June 2014)
8. E. I. Oyman and C. Ersoy: Multiple sink network design problem in large scale wireless sensor networks. In: proc. of the IEEE International Conference on Communications (ICC) (June 2004)
9. (Ed.), S.N.: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6, IETF Request for Comments (RFC) 5798
10. J. Hui and P. Thubert: Compression Format for IPv6 Datagrams over IEEE 802.15.4-based Networks, IETF Request for Comments (RFC) 6282 (September 2011)
11. J. Li and S. Ji and H. Jin and Q. Ren: Routing in multi-sink sensor networks based on gravitational field. In: proc. of International Conference on Embedded Software and System (ICESS) (July 2008)
12. P. Levis and T. Clausen and J. Hui and O. Gnawali and J. Ko: The Trickle Algorithm, IETF Request for Comments (RFC) 6206 (March 2011)
13. R. Kuntz and J. Montavont and T. Noel: Multihoming in IPv6 Mobile Networks: Progress, Challenge and Solutions. IEEE Communication Magazine 51(1), 128–135 (January 2013)
14. S. Hussain et al.: Applications of Wireless Sensor Networkds and RFID in a Smart Home Environment. In: proc. of the 7th Annual Conference CNSR (May 2009)
15. T. Winter and P. Thubert and A. Brandt and J. Hui and R. Kelsey and P. Levis and K. Pister and R. Struik and JP. Vasseur and R. Alexander: IPv6 Routing Protocol for Low-Power and Lossy Networks, IETF Request for Comments (RFC) 6550 (March 2012)
16. V. Dyo et al.: Evolution and Sustainability of a Wildlife Monitoring Sensor Network. In: proc. of the ACM Conference on Embedded Networked Sensor Systems (SenSys) (November 2010)
17. Z. Shelby and S. Chakrabarti and E. Nordmark and C. Bormann: Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs), IETF Request for Comments (RFC) 6775 (November 2012)