iPRP
INTERNET-DRAFT
Intended Status: Standard Track
Expires: August 4, 2016

Miroslav Popovic
Maaz Mohiuddin
Jean-Yves Le Boudec
EPFL-LCA2
Dan-Cristian Tomozei
Cisco Systems
Febraury 1, 2016

iPRP: Parallel Redundancy Protocol for IP Networks
draft-popovic-iprp-01

Abstract

Reliable packet delivery within stringent delay-constraints is of paramount importance to mission-critical computer applications with hard real-time constraints. Because retransmission and coding techniques counteract the delay requirements, reliability may be achieved through replication over multiple fail-independent paths. Existing solutions, such as the parallel redundancy protocol (PRP), replicate all packets at the MAC layer over parallel paths. PRP works best in local area networks. However, it is not viable for IP networks that are a key element of emerging mission-critical systems. This limitation, coupled with diagnostic inability and lack of security, renders PRP unsuitable for reliable data-delivery in these IP networks. To address this issue, we present a transport-layer solution: the IP parallel redundancy protocol (iPRP). Designing iPRP poses non-trivial challenges in the form of selective packet-replication, soft-state and multicast support. iPRP replicates only time-critical unicast or multicast UDP traffic. iPRP requires no modifications to the existing monitoring application, end-device operating system or to the intermediate network devices. It only requires a simple software installation on the end-devices. iPRP has a set of diagnostic tools for network debugging. With our implementation of iPRP in Linux, we show that iPRP supports multiple flows with minimal processing-and-delay overhead. It is being installed in our campus smart-grid network and is publicly available.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as

Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1. Introduction

Specific mission-critical computer applications have hard delay-constraints. Failure to satisfy these constraints can result in economic losses or, even worse, human lives can be endangered in cases when these failures affect safety mechanisms. Notable examples of such applications (often built on top of cyber-physical systems) are process-control and emergency-management applications in the oil and gas industry, real-time detection of pollutants in the water/wastewater industry, vehicle-collision avoidance in car industry, automatic train control, process control in chemical industry, state estimation in smart grid, high-frequency trading and distributed online gaming.

Reliable and timely packet delivery, even in the order of 10 ms, is of utmost importance in satisfying the hard-delay constraints. The classic approaches to reliable communication through coding and retransmission are not compatible with the hard delay-constraints. An alternative is to achieve reliability through replication over multiple fail-independent paths, which is the focus of this paper. More precisely, we present a solution for packet-replication over multiple paths in IP networks. Indeed, as we discuss next, existing solutions apply only to MAC-layer networks and cannot be transposed to IP networks that are a requirement for many of the above-mentioned applications.

1.1. From MAC- to IP-Layer Parallel Redundancy Protocol

The parallel redundancy protocol (PRP) IEC standard [1] was proposed as a solution to reliable data delivery problem for deployments inside a local area network (LAN). Communicating devices need to be connected to two cloned (disjoint) bridged networks. The sender tags MAC frames with a sequence number and replicates it over its two interfaces. The receiver discards redundant frames based on sequence numbers.

In addition to extending PRP functionality to IP networks, the new design should also avoid the drawbacks of PRP. The most limiting feature of PRP is that the two cloned networks need to be composed of devices with identical MAC addresses. This contributes to making network management difficult. Furthermore, PRP duplicates all the traffic unselectively, which is acceptable for use in a local environment, but which cannot be done in general IP networks, because some links can be expensive and unnecessary traffic should be avoided. Moreover, PRP has no security mechanisms.

Note that a parallel redundancy protocol for IP networks needs to support IP multicast, as this is used in many of the aforementioned

applications.

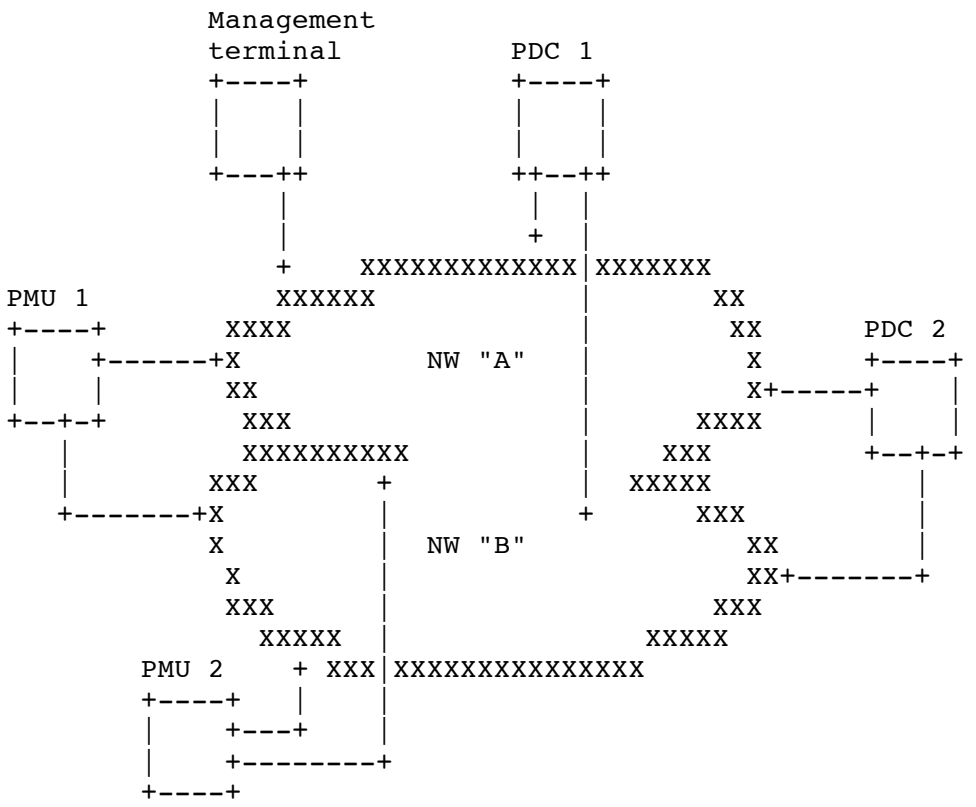

Figure 1: A typical iPRP use-case in the context of smart grids. Devices (Phasor Data Concentrators (PDCs), Phasor Measurement Units (PMUs)) are connected to two overlapping network subclouds (labeled "A" and "B"). Every PMU streams data to all PDCs, using UDP and IP multicast.

Concretely, Figure 1 depicts a smart grid WAN where PRP cannot be directly deployed. Devices are multi-homed and each interface is assigned a different IP address. Most devices have two interfaces connected to a main network cloud made of two fail-independent network subclouds labeled "A" and "B". It is assumed that paths between interfaces connected to the "A" network subcloud stay within it (and similarly with "B"). The "A" and "B" network subclouds could be physically separated, however in practice they are most likely interconnected for network management reasons.

As an example, we use the smart-grid communication network depicted in Figure 1. Here, measurements are streamed periodically (every 20 ms for 50 Hz systems) by phasor measurement units (PMUs) to phasor data concentrators (PDCs), where the information about the electrical network state is expected in quasi-real time. PRP cannot be directly deployed here: devices are multi-homed and each interface is assigned a different IP address. Most devices have two interfaces connected to a main network cloud made of two fail-independent network subclouds labeled "A" and "B". It is assumed that paths between interfaces connected to the "A" network subcloud stay within it (and similarly with "B"). The "A" and "B" network subclouds could be physically separated, however in practice they are most likely interconnected for network management reasons.

A simple way to realize the arrangement described before is to divide the network into two logical subclouds, "A" and "B". Then, by adjusting the routing weights of the links interconnecting the "A" and "B" subclouds, it can be ensured that "A"->"A" and "B"->"B" traffic stays within "A" and "B" subclouds respectively, thereby giving rise to fail-independent paths. In such a setting, the interconnections will be used only for "A"<->"B" traffic.

The solution should, similarly to PRP, takes advantage of network redundancy for increasing the reliability of UDP flows, and that works in scenarios such as the one in Figure 1. The existence of fail-independent paths is fundamental for the optimal operation of such a solution. However, in the event of a network-component failure, the paths can become partially overlapping. Then, the solution should reap the maximum possible benefit by operating in a degraded-redundancy mode. In other words, even if complete end-to-end redundancy is no longer possible the solution should continue to work.

In order for our solution to be easily deployed, we also require it to be transparent to both the application and network layers: it should only require installation at end-devices and no modifications to running application software or to intermediary network devices (routers or bridges). In addition, real-time applications typically use UDP rather than TCP because (1) TCP does not work with IP multicast or (2) TCP retransmissions, following packet losses, require several round-trip times and can be both detrimental and superfluous. Hence, we target a solution that supports UDP applications only.

This document is a summary of a conference paper [12] that describes iPRP (IP parallel redundancy protocol), a transport layer solution for transparent replication of unidirectional unicast or multicast UDP flows on multihomed devices. A technical report [7] that contains

all the relevant details of the solution has also been made available. The prototype iPRP implementation (http://goo.gl/N5wFNt) is for IPv6, as it is being installed in an actual IPv6-based smart-grid communication network (smartgrid.epfl.ch) [11]. Adaptation to IPv4 is straightforward.

1.2. iPRP

An iPRP host has to send different copies of the same packet over different paths. With the current technology, a device cannot control the path taken by an IP packet, beyond the choice of a destination address, exit interface and a type-of-service value. Other fields, such as the IPv6 flow label or source routing header extensions, are either ignored or rejected by most routers. Also, the type-of-service field is used by applications and should not be tampered with by iPRP. Hence, it is assumed that a choice of the path is done at the sources by choosing communication interface and the destination address. The job of iPRP is then to transparently replicate packets over the different interfaces for the UDP flows that need it, match corresponding interfaces, remove duplicates at the receiver, and do this in a way that is resilient to crashes.

Not all traffic requires replication, only certain devices and certain UDP flows do (time-critical data). Hence, replication needs to be selective: a failure-proof mechanism, transparent to applications, is required for detecting and managing packet replication. It needs to match well the interfaces, so that independent paths are used whenever they exist. However, the solution should continue to work if some paths are not disjoint.

The iPRP protocol design is such that it does not interfere with the existing security mechanisms and does not introduce any new security weaknesses (see Section 5).

iPRP assumes that the network is traffic-engineered; the critical UDP data streams receive enough resources and are not subject to congestion. iPRP instantly repairs packet losses due to failures or transient problems such as transmission losses. It does not solve congestion problems due to under-dimensioned network links. TCP flows are not affected.

1.3. 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 RFC 2119 [RFC2119].

2. Related work

As mentioned in the Introduction, iPRP overcomes the limitations of PRP [2]. The authors of [3] are aware of these limitations and suggest a direction for developing PRP in an IP environment. Their suggestion is neither fully designed nor implemented. Also, it requires that the intermediate routers preserve the PRP trailers at the MAC layer, which in turn requires changes in all of the routers in the networks. It does not address all the shortcomings of PRP (diagnostic tools, lack of multicast support, need of special hardware). In contrast, the transport layer approach of iPRP does not have these drawbacks.

MPTCP[4] is used in multi-homed hosts. It allows TCP flows to exploit the host's multiple interfaces, thus increasing the available bandwidth for the application. Like MPTCP, iPRP is a transport layer solution and is transparent to network and application. Unlike MPTCP, iPRP duplicates the UDP packets on the parallel paths, while MPTCP sends one TCP segment on only one of them. In a case of loss, MPTCP resends the segment on the same path until enough evidence is gathered that this path is broken. So, a lost packet is repaired after several RTTs (not good for time-critical flows). Similarly, LACP [5] and ECMP [6] require seconds for failover.

Network coding exploits network redundancy for increasing throughput [13], and requires intermediary nodes to recode packets (specialized network equipment needed). Also, it is not suitable for time-critical applications as typically packets are coded across "generations" which introduces decoding delays. Source coding (e.g. Fountain codes [14]) can be useful for the bursty transmissions of several packets. However, it adds delay, as encoding and decoding are performed across several packets (not suitable for UDP flows with hard-delay constraints).

MPLS-TP 1+1 protection feature [15] performs packet duplication and feeds identical copies of the packets in working and protection path. On the receiver side, there exists a selector between the two; it performs a switchover based on some predetermined criteria. However, some time is needed for fault detection and signaling to take place, after which the switchover occurs. Hence, a 0-ms repair cannot be achieved.

Multi-topology routing extends existing routing protocols (e.g. [16]) and can be used to create disjoint paths in a single network. It does not solve the problem of transparent packet replication, but serves as a complement to iPRP.

3. Operation of iPRP

3.1. How to Use iPRP

iPRP is installed on end-devices with multiple interfaces: on streaming devices (the ones that generate UDP flows with hard delay constraints) and on receiving devices (the destinations for such flows). Streaming applications (such as PMU streaming applications) do not require any changes. These applications benefit from the increased reliability of iPRP without being aware of its existence. iPRP operates as a modification to the UDP layer.

On receiving devices the only thing that needs to be configured is the set of UDP ports on which duplication is required. For example, say that an application running on a PDC is listening on some UDP port for measurement data coming from PMUs. After iPRP is installed, this port needs to be added to the list of iPRP monitored ports in order to inform iPRP that any incoming flows targeting this port require replication. The application does not need to be stopped and is not aware of iPRP. Nothing else needs to be done for iPRP to work. In particular, no special configuration is required for intermediary network equipment (routers, bridges).

3.2. General Operation: Requirements for Devices and Network

iPRP provides 1+n redundancy. It increases, by packet replication, the reliability of UDP flows. It does not impact TCP flows. iPRP-enabled receiving devices configure a set of UDP ports as "monitored". When a UDP packet is received on any of the monitored ports, a one-way soft-state iPRP session is triggered between the sender and the receiver (or group of receivers, if multicast is used). Soft-state means that: (i) the state of the communication participants is refreshed periodically, (ii) the entire iPRP design is such that a state-refresh message received after a cold-start is sufficient to ensure proper operation. Consequently, the state is automatically restored after a crash, and devices can join or leave an iPRP session without impacting the other participants.

Within an iPRP session, each replicated packet is tagged with an iPRP header (Figure 2). It contains the same sequence number in all the copies of the same original packet. At the receiver, duplicate packets with the same sequence number are discarded (Section 4.3). The original packet is reconstructed from the first received copy and forwarded to the application.

In multicast, the entire receiver group needs to run iPRP. If by mishap, only part of the receivers support iPRP, these trigger the start of an iPRP session with the sender and benefit from iPRP; however, the others stop receiving data correctly. To ensure disjoint trees the use of source-specific multicast (SSM) is recommended, see [7]. All iPRP-related information is encrypted and authenticated. Existing mechanisms for cryptographic key exchange are

applied (security reflections in Section 5).

3.3. UDP Ports Affected by iPRP

iPRP requires two system UDP ports (transport layer) for its use: the iPRP control port and the iPRP data port (in the prototype implementation 1000 and 1001, respectively). The iPRP control port is used for exchanging messages that are part of the soft-state maintenance. The iPRP data port receives data messages of the established iPRP sessions. iPRP-capable devices always listen for iPRP control and data messages.

The set of monitored UDP ports, over which iPRP replication is desired are not reserved by iPRP and can be any UDP ports. UDP ports can be added to/removed from this set at any time during the iPRP operation. Reception of a UDP packet on a monitored port triggers the receiver to initiate an iPRP session. If the sender is iPRP-capable, an iPRP session is started (replicated packets are sent to the iPRP Data Port), else regular communication continues.

3.4 Matching the Interconnected Interfaces of Different Devices

One of the design challenges of iPRP is determining an appropriate matching between the interfaces of senders and receivers, so that replication can occur over fail-independent paths. To understand the problem, consider Figure 1 where the PMUs and PDCs have at least two interfaces. The "A" and "B" network subclouds are interconnected. However, the routing is designed such that, a flow originating at an interface connected to subcloud "A" with a destination in "A", will stay in subcloud "A". A potential problem can arise if a sender's interface, say "SA", intended to be connected to the "A" subcloud, is mistakenly connected to the "B" subcloud, and vice-versa. Then one path from source to destination will go from "SA" (on subcloud "B") to the destination interface "DB" (on subcloud "B"), and conversely on the other path. Following the routing rules, these flows will use interconnecting links between "A" and "B" subclouds. This is not desirable as it is no longer guaranteed that such paths are fail-independent. Furthermore, these links can be of insufficient capacity because they are not intended to carry such traffic. PRP avoids this problem by requiring two physically separated and cloned networks. iPRP does not impose these restrictions. Hence, iPRP needs a mechanism to match interfaces connected to the same network subcloud.

To facilitate appropriate matching, each interface is associated with a 4-bit identifier called iPRP network subcloud discriminator (IND), which qualifies the network subcloud it is connected to. The iPRP software in end-devices learns the interfaces' INDs automatically via simple preconfigured rules. Network routers have no notion of IND. A

rule can use the interface's IP address or its DNS name. In the prototype implementation, an interface's IND is computed from its fully qualified domain name. In Figure 1, the names of the interfaces are assigned in the following way. PDC1: nw-a.pdc1.smartgrid.epfl.ch for the interface connected to NW "A" and nw-b.pdc1.smartgrid.epfl.ch for the interface connected to NW "B". Similarly for PMU2, for example: nw-a.pmu2.smartgrid.epfl.ch for the interface connected to NW "A" and nw-b.pmu2.smartgrid.epfl.ch for the interface connected to NW "B". Then, the rule in the iPRP configuration maps the regular expression nw-a* to the IND value 0xa, nw-b* to IND 0xb.

The receiver periodically advertises the IP addresses of its interfaces, along with their INDs to the sender (via iPRP_CAP messages). The sender compares the received INDs with its own interface INDs. Replication is done only on the interfaces which were successfully matched. In the example from Figure 1, IND matching prevents iPRP to send data from a PMU "A" interface to a PDC "B" interface. Moreover, each iPRP data packet (Figure 2) contains the IND of the network subcloud where the packet is supposed to transit. This eases the monitoring and debugging of the whole network. It allows detection of misconfiguration errors that cause a packet expected on an A interface to arrive on a "B" interface.

4. A Glimpse of iPRP Design

A comprehensive description can be found in [7].

4.1. Control Plane

The control plane establishes and maintains an iPRP session. When triggered by the reception of a UDP packet on one of the ports configured as monitored, the receiver starts sending iPRP_CAP messages to the control port of the sender every T_CAP seconds. This informs the sender that the receiver is iPRP enabled and provides information required for selective replication over alternative paths (e.g. IP addresses of all receiver interfaces). This is also used as a keep-alive message for iPRP session as an iPRP session is terminated if no iPRP_CAP message is received for a period of 3T_CAP.

On receiving the iPRP_CAP, the sender acknowledges it with an iPRP_ACK. The iPRP_ACK contains the list of sender IP addresses which are used by the receiver to subscribe to alternate network subclouds. In multicast, the receivers send iPRP_CAP after a back-off period (Section 4.4) to avoid flooding. The iPRP_ACK message also serves as the terminating message for impending iPRP_CAPs thereby preventing a flood. To complete the establishment of an iPRP session, the sender performs IND matching (Section 3.4) and creates a record that contains all information needed for replication of data packets.

iPRP_CAP also contains symmetric key that is used for encryption and authentication of the replicated iPRP packets.

4.2. Data Plane: Replication and Duplicate Discard

The replication occurs at the sender to send out data plane messages once the iPRP session is established. All outgoing packets, destined to the UDP port of the receiver that were configured as monitored, are intercepted. These packets are subsequently replicated and iPRP headers (Figure 2) are prepended to each copy of the payload. iPRP headers are populated with the iPRP version, a sequence-number-space ID (used to identify an iPRP session), a sequence number, an original UDP destination port (for the reconstruction of the original UDP header), and IND. The 32-bit sequence number is the same for all the copies of the same packet. The destination port number is set to iPRP data port for all the copies. An authentication hash is appended and the whole block is encrypted. The iPRP header is placed after the inner-most UDP header. So, iPRP works well, even when tunneling is used (e.g., 6to4). Finally, the copies are transmitted as iPRP data messages over the different matched interfaces.

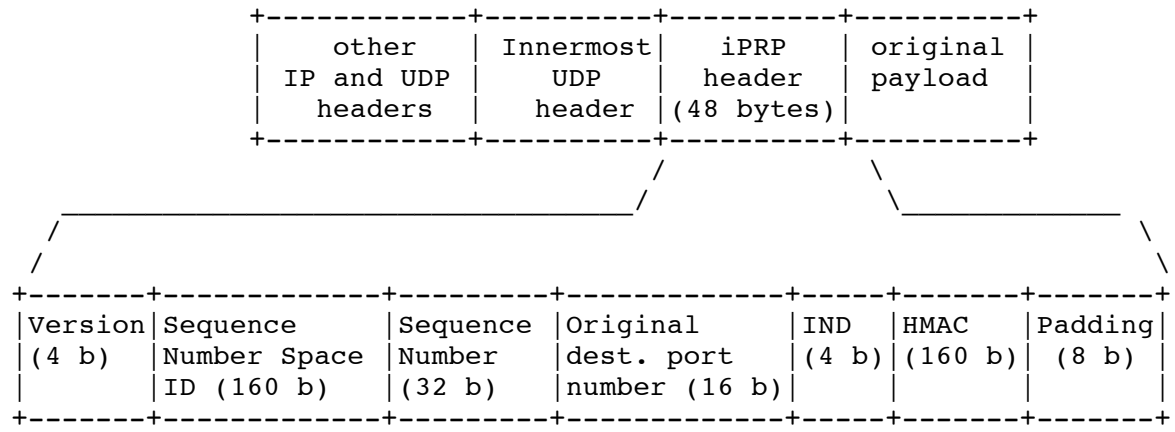

Figure 2: Location and fields of iPRP header

Upon reception of packets on the iPRP data port the iPRP header is decrypted at the beginning of the payload using the symmetric key used in iPRP_CAP message. Based on the sequence-number-space ID and the sequence number, the packet is either forwarded to the application or discarded.

4.3. The Discard Algorithm

The discard algorithm forwards the first copy of a replicated packet to the application and discards all subsequent packets. The discard

algorithm proposed for PRP[8] fails at the latter when packets are received out of order. Packet reordering cannot be excluded in IP networks. The iPRP discard algorithm [7] forwards only one copy of the packet even in cases of packet reordering. Also, it is soft-state, thereby resilient to crashes and reboots.

4.4. The Backoff Algorithm

The soft-state in a multicast iPRP session is maintained by periodic advertisements (iPRP_CAP) sent to the source by each member in the multicast group of receivers. The iPRP backoff algorithm prevents "message implosion" at the source, for groups of receivers ranging from several tens to millions of hosts. It guarantees that the source receives an iPRP_CAP within a bounded time D (by defalut D=10s) after the start of the iPRP-relevant communication (executed periodically every T_CAP=30s). To this end, the backoff time is sampled from the distribution from [9] as described in [7].

5. Security Considerations

The iPRP protocol design is such that it does not interfere with upper-layer security protocols. However, in addition to these solutions, iPRP layer itself needs to be secure, as there are attacks that can stay undetected by upper-layer security protocols. Concretely, if an attacker manages to alter the sequence-number field of iPRP packets transmitted over one (compromised) network subcloud, the discard algorithm can be tricked in a way that the packets from both (compromised and non-compromised) network subclouds are discarded. Note that similar attacks exist for PRP, where an attacker, with access to one network, can force the discard of valid frames on another network. For example, say an attacker has access to network subcloud "A". A PRP frame is represented as "A5, where "A" is the network subcloud it belongs to and 5 is the sequence number. If "A5" and "B5" were received and the attacker retransmits the frame "A5" by altering the sequence number as "A6", then the actual "A6" and "B6" frames will both be discarded. In other words, an unsecured PRP or iPRP could weaken the network instead of making it more robust. Yet another argument for protecting the iPRP layer is that by doing so, the exposure for prospective attacks in the future is minimized.

The iPRP control messages are encrypted and authenticated. This guarantees that the security of replicated UDP flows is not comprimised by iPRP and that it does not interfere with application layer encryption/authentication.

Specifically, iPRP_CAP messages and the corresponding iPRP_ACK messages are transmitted over a secure channel. The iPRP header

inserted in the data packets is authenticated and encrypted with a pre-shared key. Thus, replay attacks and forged messages insertion are avoided.

A secure channel is established for the transmission of iPRP_CAP messages depending on the type of communication, unicast or multicast. Details follow below.

Unicast: In unicast mode, a DTLS session is maintained between the sender and the receiver. It is initiated by the receiver upon the arrival of the first UDP datagram from the source. iPRP_CAP messages are transmitted within this session. So, the iPRP capabilities of the receiver are transmitted only to an authenticated source. iPRP_ACKs are not required in unicast (since message implosion can occur in multicast only).

Unicast iPRP_CAP messages contain a symmetric key used to authenticate and encrypt the iPRP header. This key is updated periodically during a unicast iPRP session. Hosts keep a small fixed number of valid past keys to prevent losing the iPRP session because of delayed receiption of a new key. The oldest key is discarded upon reception of a new one.

Multicast: In multicast, iPRP relies on any primitive that establishes a secure channel with the multicast group. For example MSEC [10] can be used for group key management and for establishing a group security association.

In this setting, both iPRP_CAP and iPRP_ACK messages, as well as the iPRP headers inserted in the replicated packets, are authenticated and encrypted with the group key. Thus, there is no need to include an additional key in the iPRP_CAP.

6. Implementation and the Diagnostic Toolkit

The prototype Linux implementation of iPRP is in user-space but care has been taken to keep the delay and processing overhead very small. It is based on the libnetfilter_queue (NF_QUEUE) framework from the Linux iptables project. NF_QUEUE is a userspace library that provides a handle to packets queued by the kernel packet filter. It requires the libnfnetlink library and a kernel that includes the nfnetlink_queue subsystem (kernel 2.6.14 or later). It supports all Linux kernel versions above 2.6.14. Concretely, the prototype implementation uses the the Linux kernel 3.11 with iptables-1.4.12 [7]. Being a user-space proof-of-concept implementation, it does not support IPsec but is compatible with higher layers security mechanisms.

As iPRP is designed to be IP friendly, it facilitates the exploitation of the diagnostic utilities associated with TCP/IP (e.g., ping and traceroute). Furthermore, an iPRP-specific diagnostic toolkit has been developed. It includes iPRPping for verification of connectivity between hosts and the evaluation of the corresponding RTTs, iPRPtracert for the discovery of routes to a host, iPRPtest to check if there is currently an iPRP session between between two hosts and establish one if absent, iPRPsenderStats and iPRPreceiverStats to obtain the loss rates and one-way network latency.

Imagine a typical scenario where an application on an iPRP enabled host experiences packet losses. To troubleshoot this problem, the user at the receiving host would use the iPRPreceiverStats to check the packet loss statistics on each network, if an iPRP session is established. If no established session is found, the user can establish a test session using iPRPtest and check the hop-by-hop UDP delivery with iPRPtracert to pin-point the problem.

7. Performance Evaluation

iPRP is being deployed on the EPFL smart-grid communication network (smartgrid.epfl.ch). Also, a lab testbed is setup to evaluate its performance.

The discard algorithm was stress-tested with heavy losses and asymmetric delays and compared the performance with theory. A sender and a receiver (Lenovo ThinkPad T400 laptops, specification given in Table 1) are interconnected with three networks (2 wired, 1 wireless). Different scenarios with bursty or independent losses, small or large link delays to simulate different possible effects observed in a real network, were generated using tc-netem. In each scenario, the observed loss rate at the application when iPRP is used is compared with the theoretical loss rate (assuming independence of networks). The findings show iPRP performs as expected in significantly reducing the effective packet losses [7].

component	version specification
CPU	Intel C2Duo, 2.53 Ghz
Ethernet Controller	Intel 82567LM Gigabit Card
Wireless Controller	Intel WiFi N d5300
Operating System	Ubuntu 12.04 LTS, 64-bit
Kernel	3.6.11-rt29 (RT-Linux Patch)

Table 1: Specifications of hosts used in the test bed

As a side benefit, iPRP improves the effective one-way network

latency by delivering the first received packet. This property was also verified by showing that the CDF of the measured network latency matches the one from theory.

Additionally, the delay and processing overhead due to iPRP was verified. GNU gprof was used to assess the average delay (over a large number of runs) incurred by an iPRP data packet in the iPRP sender and receiver daemons. It was observed that an accepted iPRP data packet encounters an average delay of 0.8 us at the sender daemon and 3.6 us at the receiver daemon. The additional percentage of CPU usage at sender (U_s) and receiver (U_r) due to iPRP was found to be:

U_s = 3.7 + 0.28*number_of_iPRP_sessions + 0.01*packets_per_second

U_r = 0.9 + 0.08*number_of_iPRP_sessions + 0.01*packets_per_second

A more fine-grained delay and CPU usage audit can be found in [7].

8. Conclusion

The design of iPRP, a transport layer solution for improving reliability of UDP flows with hard-delay constraints, such as smart grid communication, industrial processes, high-frequency trading and online gaming, was presented. iPRP is application- and network-transparent, which makes it plug-and-play with existing applications and network infrastructure. Furthermore, the soft-state design makes it resilient to software crashes. Besides unicast, iPRP supports IP multicast, making it a suitable solution for low-latency industrial automation applications requiring reliable data delivery. iPRP is equipped with diverse monitoring and debugging tools, which is quasi impossible with existing MAC layer solutions. With a prototype implementation, it was shown that iPRP can support several sessions between hosts without any significant delay or processing overhead. The implementation is publicly available and is currently installing it in the EPFL campus smart-grid [11].

9. IANA Considerations

iPRP requires two system UDP ports (transport layer) for its use: the iPRP control port and the iPRP data port (in the prototype implementation 1000 and 1001, respectively).

10. References

[1] H. Kirrmann, M. Hansson, and P. Muri, "Iec 62439 prp: Bumpless recovery for highly available, hard real-time industrial networks," in Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on, Sept 2007, pp. 1396-1399.

[2] IEC 62439-3 Standard, "Industrial communication networks: High availability automation networks", 2012.

[3] M. Rentschler and H.Heine, "The parallel redundancy protocol for industrial IP networks", in Industrial Technology (ICIT), 2013 IEEE International Conference on, Feb 2013, pp.1404-1409.

[4] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure,"TCP Extensions for Multipath Operation with Multiple Addresses", RFC 6824 (Experimental), Internet Engineering Task Force, Jan. 2013

[5] IEEE Standards Association, "IEEE Std 802.1AX-2008 IEEE Standard for Local and Metropolitan Area Networks - Link Aggregation.", 2008.

[6] C. Hopps. "Analysis of an Equal-Cost Multi-Path Algorithm", RFC2992 (Informational), Internet Engineering Task Force, Nov. 2000

[7] M. Popovic, M. Mohiuddin, D.-C. Tomozei, and J.-Y. Le Boudec, "iPRP: Parallel Redundancy Protocol for IP Networks: ", accepted for publication, IEEE Transactions on Industrial Informatics.

[8] H.Weibel, "Tutorial on parallel redundancy protocol", Zurich University of Applied Sciences, Tech. Rep.

[9] J. Nonnenmacher and E. W. Biersack, "Scalable feedback for large groups", IEEE/ACM Transactions on Networking (ToN), vol. 7, no. 3.

[10] M. Baugher, R. Canetti, L. Dondeti, and F. Lindholm, "Multicast Security (MSEC) Group Key Management Architecture", RFC 4046.

[11] M. Pignati and al., "Real-Time State Estimation of the EPFL-Campus Medium-Voltage Grid by Using PMUs",in Innovative Smart Grid Technologies Conference (ISGT), 2015 IEEE PES, 2015.

[12] M. Popovic, M. Mohiuddin, D.-C. Tomozei, and J.-Y. Le Boudec, "iPRP: Parallel Redundancy Protocol for IP Networks", in 11th IEEE World Conference on Factory Communication Systems, May 27-29, 2015, Palma de Mallorca, Spain

[13] C. Fragouli, and E. Soljanin, "Network Coding Fundamentals", Foundations and Trends in Networking, vol. 2, no. 1, pp. 1-133, 2007.

[14] D.J.C. MacKay, "Fountain Codes", Communications, IEEE Proceedings, vol. 152, no. 6, pp, 1062-1068, Dec. 2005.

[15] N. Sprecher, and A. Farrel, "MPLS Transport Profile (MPLS-TP) Survavibility Framework", RFC 6372 (Informational), Internet Engineering Task Force, Sep. 2011.

[16] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multy-Topology (MT) Routing in OSPF", RFC 4915 (Proposed Standard), Internet Engineering Task Force, Jun. 2007.

Authors' Addresses

Miroslav Popovic
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 6466
EMail: miroslav.popovic@epfl.ch

Maaz Mohiuddin
EPFL IC ISC LCA2
Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 1334
EMail: maaz.mohiuddin@epfl.ch

Jean-Yves Le Boudec
EPFL IC ISC LCA2

Station 14
CH-1015 Lausanne
Switzerland

Phone: +41 21 693 6631
EMail: jean-yves.leboudec@epfl.ch

Dan-Cristian Tomozei
Cisco Systems
Batiment E/Niv. 3
Quartier de l'innovation
CH-1015 Lausanne
Switzerland

Phone: +41 21 822 1714
EMail: dtomozei@cisco.com