iPRP: Parallel Redundancy Protocol for IP Networks

Miroslav Popovic*, Maaz Mohiuddin*, Dan-Cristian Tomozei* and Jean-Yves Le Boudec*
*I&C-LCA2, EPFL, Switzerland

***Abstract*—Reliable packet delivery within stringent delay constraints is of primal importance to industrial processes with hard real-time constraints, such as electrical grid monitoring. Because retransmission and coding techniques counteract the delay requirements, reliability is achieved through replication over multiple fail-independent paths. Existing solutions such as parallel redundancy protocol (PRP) replicate all packets at the MAC layer over parallel paths. PRP works best in local area networks, e.g., sub-station networks. However, it is not viable for IP layer wide area networks which are a part of emerging smart grids. Such a limitation on scalability, coupled with lack of security, and diagnostic inability, renders it unsuitable for reliable data delivery in smart grids. To address this issue, we present a transport-layer design: IP parallel redundancy protocol (iPRP). Designing iPRP poses non-trivial challenges in the form of selective packet replication, soft-state and multicast support. Besides unicast, iPRP supports multicast, which is widely using in smart grid networks. It duplicates only time-critical UDP traffic. iPRP only requires a simple software installation on the end-devices. No other modification to the existing monitoring application, end-device operating system or intermediate network devices is needed. 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.**

I. Introduction

Specific time-critical applications (found for example in electrical networks) have such strict communication-delay constraints that retransmissions following packet loss can be both detrimental and superfluous. In smart grids, critical control applications require reliable information about the network state in quasi-real time, within hard delay constraints of the order of approximately 10 ms. Measurements are streamed periodically (every 20 ms for 50 Hz systems) by phasor measurement units (PMUs) to phasor data concentrators (PDCs). In such settings, retransmissions can introduce delays for successive, more recent data that in any case supersede older ones. Also, IP multicast is typically used for delivering the measurements to several PDCs. Hence, UDP is preferred over TCP, despite its best-effort delivery approach. Increasing the reliability of such unidirectional (multicast) UDP flows is a major challenge.

The parallel redundancy protocol (PRP, IEC standard [1]) was proposed as a solution for deployments inside a local area network (LAN) where there are no routers. 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.

PRP works well in controlled environments, like a substation LAN, where network setup is entirely up to the substation operator, who ensures that the requirements of PRP are met (e.g., all network devices are duplicated). At a larger scale (for example, a typical smart grid communication network that spans an entire distribution network) routers are needed and PRP can no longer be used. Thus, a new solution is needed for IP wide area networks (WANs).

In addition to extending PRP functionality to WANs, 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 LAN, but which cannot be done in a WAN, because links are expensive and unnecessary traffic should be avoided. Moreover, PRP has no security mechanisms, and multicasting to a specific group of receivers is not natively supported.

Concretely, Fig. 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”, while some have a third interface connected to a 4G cellular wireless service (labeled “Swisscom LTE backbone” in the figure). 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.

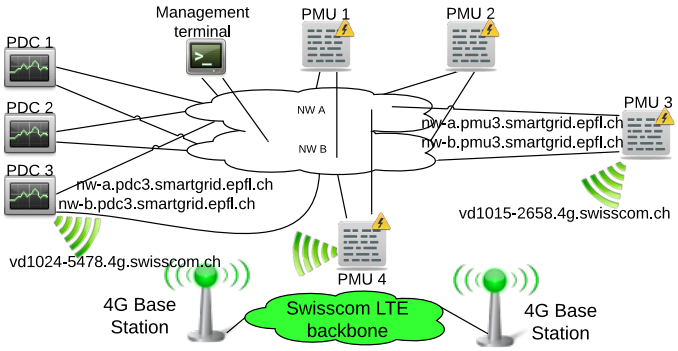

Fig. 1: A typical iPRP use-case in the context of smart grids. Devices (PDCs, PMUs) are connected to two overlapping network subclouds (labeled A and B). Some devices use an additional LTE connection providing a low latency cellular service. Every PMU streams data to all PDCs, using UDP and IP multicast.

The new solution should, like PRP, take advantage of network redundancy to increase the reliability of UDP flows. Furthermore, it should also work both if the network subclouds are physically separated, and if they have interconnections. Supporting such flexibility is essential for real deployments,

This research has received funding from Nano-Tera SmartGrid project. The authors would also like to thank Prof. Edouard Bugnion of EPFL for his invaluable suggestions.

but it poses significant challenges in protocol design. In order for our solution to be easily deployed, we furthermore 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 this paper we present the design and implementation of iPRP (IP parallel redundancy protocol), a transport layer solution for transparent replication of unidirectional unicast or multicast UDP flows on multihomed devices. Our iPRP implementation (http://goo.gl/N5wFNt) is for IPv6, as it is being installed in our IPv6 based smart-grid communication network (smartgrid.epfl.ch). Adaptation to IPv4 is straightforward.

To reach (multicast) destinations, a transport-layer solution for packet replication of UDP flows such as iPRP should have at its disposal two or more parallel paths; ideally, the paths should be disjoint. However, given the current technology, a device cannot easily control the path taken by an IP packet, beyond the choice of an 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 routers. Also, the type-of-service field is used by applications and should not be tampered with by iPRP. Hence, we assume a setting where sources and destinations of time-critical UDP flows have multiple communication interfaces. The job of iPRP is to transparently replicate packets over the different interfaces for the UDP flows that need it, match corresponding interfaces, and remove duplicates at the receiver. The paths between matching interfaces are calculated by the routing layer: by routing protocols or by software-defined networking.

iPRP does not require cloned networks. It exploits any available network redundancy, without imposing special requirements on the network. End-to-end disjoint paths are not required, though in most cases this is beneficial. Also, 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 over the matched interfaces. Finally, iPRP needs to be secure and also ensure that replication does not compromise secure UDP streams.

II. Related Work

As mentioned in §I, 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 (absence of diagnostic tools, lack of multicast support, need of special hardware). In contrast, our transport layer approach does not have these drawbacks.

Solutions like MPTCP [4], LACP [5], ECMP [6] require non-negligible amount of time for giving up on a path that has failed. Similarly, network coding and source coding (e.g. Fountain codes) introduce coding and decoding delays that are not suitable for UDP flows with hard-delay constraints.

Multi-topology routing extends existing routing protocols 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.

III. Operation of iPRP

A. 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 devices (such as PMUs) do not require any configuration. Streaming applications running on such devices benefit from the increased reliability of iPRP without being aware of its existence.

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).

B. 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 (Fig. 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 (§IV-C). 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 §V).

C. 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 our 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.

D. 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. PRP achieves this by requiring two physically separated 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 our implementation, we compute an interface's IND from its fully qualified domain name . In Figure 1, the rule in the iPRP configuration maps the regular expression `nw-a*` to the IND value `0xa`, `nw-b*` to IND `0xb`, and `*swisscom.ch` to IND `0xf`.

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. Only those interfaces with matching INDs are allowed to communicate in iPRP mode. In our example, IND matching prevents iPRP to send data from a PMU A interface to a PDC B interface. Moreover, each iPRP data packet (Fig. 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 us to detect misconfiguration errors that cause a packet expected on an A interface to arrive on a B interface.

IV. A Glimpse of iPRP Design

(A comprehensive description can be found in [7].)

A. Control Plane

The control plane establishes 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 messages 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 (§IV-D) to avoid flooding. The `iPRP_ACK` message also serves as terminating message for impending `iPRP_CAP`s thereby preventing a flood. To complete the iPRP session establishment, the sender performs IND matching (§III-D) and creates a record that contains all information needed for replication of data packets.

B. Data Plane: Replication and Duplicate Discard

The replication phase occurs at the sender to send out data plane messages once the iPRP session is established. All outgoing packets destined to UDP port p of the receiver are intercepted. These packets are subsequently replicated and iPRP headers (Fig. 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.

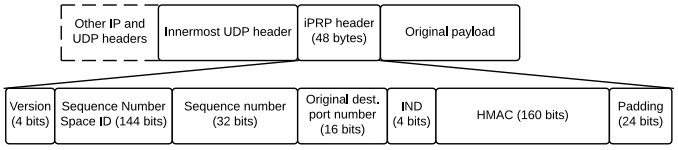

Fig. 2: Location and fields of iPRP header.

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.

C. 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 ordered. 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.

D. 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 hosts to millions. It guarantees that the source receives an `iPRP_CAP` within a bounded time D (by defalut $D = 10$s) after the start of the iPRP-relevant communication (executed periodically every $T_{CAP} = 30$s). To this end, the backoff time is sampled from the distribution from [9] as described in [7].

V. Security Considerations

iPRP control messages as well as the iPRP header in data packets are encrypted and authenticated. This guarantees that the security of replicated UDP flows is not compromised by iPRP and that it does not interfere with application layer

encryption/authentication. In unicast mode, a DTLS session is established over the control channel for key exchange; in multicast mode, we rely on existing multicast key management infrastructure [10].

VI. Implementation and the Diagnostic Toolkit

We implemented iPRP in Linux in user-space; overhead is kept very small thanks to efficient batching. We use 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. We use the the Linux kernel 3.11 with iptables-1.4.12 [7].

As iPRP is network transparent, the standard TCP/IP diagnostic tools can be used. Further, we developed an iPRP-specific diagnostic toolkit. 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.

VII. Performance Evaluation

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

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

As a side benefit, iPRP improves the effective one-way network latency by delivering the first received packet. We performed stress tests to verify that the CDF of the network latency matches the theory (Fig. 3).

We evaluated the delay and processing overhead due to iPRP. We used *GNU gprof* to assess the average delay (over a large number of runs) incurred by an iPRP data packet in the iPRP sender and receiver daemons. We observed that an accepted iPRP data packet encounters an average delay of 0.8 μs at the sender daemon and 3.6 μs at the receiver daemon. We computed the additional % of CPU usage at sender (U_s) and receiver (U_r) due to iPRP and found:

$$U_s = 3.7 + 0.28 \times (\text{\# of iPRP sessions}) + 0.01 \times (\text{packets/s})$$

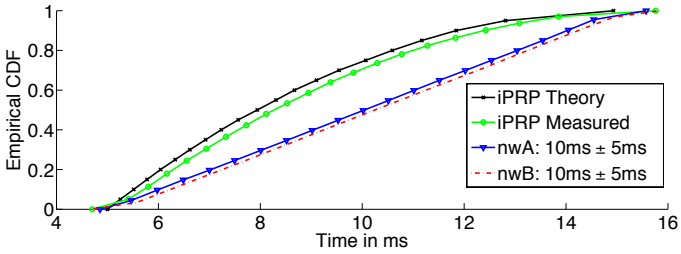

Fig. 3: CDF of delays over two networks (nwA, nwB) with latencies i.i.d and uniform on [5ms, 15ms] and CDF of latency obtained with iPRP. The theoretical value $d_{iPRP} = \min(d_{nwA}, d_{nwB})$ matches the measurements.

$$U_r = 0.9 + 0.08 \times (\text{\# of iPRP sessions}) + 0.01 \times (\text{packets/s})$$

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

VIII. Conclusion

We have designed iPRP, a transport layer solution for improving reliability of UDP flows with hard-delay constraints, such as smart grid communication. iPRP is application- and network-transparent, which makes it plug-and-play with existing applications and network infrastructure. Furthermore, our 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. We have equipped iPRP with diverse monitoring and debugging tools, which is quasi impossible with existing MAC layer solutions. With our implementation, we have shown that iPRP can support several sessions between hosts without any significant delay or processing overhead. We have made our implementation publicly available and are currently installing it in our campus smart-grid [11].

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," RFC 2992 (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," EPFL, Tech. Rep., 2014.

[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.