Computer Networks 104 (2016) 122–136

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

JITeR: Just-in-time application-layer routing

Alysson Bessani[a], Nuno F. Neves[a], Paulo Veríssimo[b], Wagner Dantas[c], Alexandre Fonseca[d], Rui Silva[d], Pedro Luz[d], Miguel Correia[d,*]

[a] LaSIGE/FCUL, Portugal
[b] Uni.Lu, Luxembourg
[c] UFSC, Brazil
[d] INESC-ID/IST, Portugal

ARTICLE INFO

Article history:
Received 13 October 2015
Revised 4 March 2016
Accepted 12 May 2016
Available online 13 May 2016

Keywords:
Overlay networks
Multihoming
Timely communication
Critical
Information infrastructures

ABSTRACT

The paper addresses the problem of providing message latency and reliability assurances for control traffic in wide-area IP networks. This is an important problem for cloud services and other geo-distributed information infrastructures that entail inter-datacenter real-time communication. We present the design and validation of JITER (*Just-In-Time Routing*), an algorithm that timely routes messages at application-layer using overlay networking and multihoming, leveraging the natural redundancy of wide-area IP networks. We implemented a prototype of JITER that we evaluated experimentally by placing nodes in several regions of Amazon EC2. We also present a scenario-based (geo-distributed utility network) evaluation comparing JITER with alternative overlay/multihoming routing algorithms that shows that it provides better timeliness and reliability guarantees.

1. Introduction

An increasing number of applications over wide-area IP networks exhibit timeliness requirements. Many of them can be served by protocols that provide those guarantees most of the time, for example by guaranteeing a given average throughput and accepting occasional violations of deadlines (e.g., video streaming). However, other applications exhibit more stringent requirements, i.e., the need that some of their messages meet individual deadlines in the presence of faults like congestion and omissions. Whilst solutions exist to the problem within over-provisioned datacenter networks, we know of no solution for generic wide-area IP networks.

This paper addresses the problem of providing latency and reliability assurances for control traffic – not for all traffic – in wide-area IP networks. Two examples show the relevance of the work for what we designate by *geo-distributed information infrastructures* (GDII).

First, *cloud services* with soft real-time requirements often span multiple datacenters in different geographical locations, implying deadline propagation amongst them, hampered by the wide-area IP network interconnects [58]. An example are cloud services that need to exchange control traffic such as Google's Megastore coordination service access [8] and Amazon's Dynamo failure detection protocol [17].

The second example is the context of *critical information infrastructures*, such as power generation, transport and distribution. Such *cyber-physical systems* are spread over large geographical areas and controlled remotely from command and control centers using SCADA/PCS[1] systems, over communication networks usually based on wide-area IP networks [10,20,31,37]. Despite the use of IP, the timeliness of critical remote commands is essential to maintain the integrity of the infrastructure (e.g., to avoid power outages). Several examples of such applications and commands were described in project CRUTIAL [23].

We present the design and validation of a novel algorithm for *Just-In-Time Routing*, JITER (pronounced "jitter"), which routes deadline constrained messages –control messages– at application layer, using an overlay network created on top of a multihomed communication infrastructure, leveraging the natural communication redundancy that exists in geo-distributed information infrastructures [37]. JITER uses a set of nodes located in different sites of the GDII to route messages among them, instead of following the routes imposed by the network-level routing. For instance, if the network-level routing makes a message sent by node r_a to node r_b pass through the autonomous system AS1, r_a may send the

* Corresponding author.
E-mail address: miguel.p.correia@tecnico.ulisboa.pt (M. Correia).

[1] Supervisory Control and Data Acquisition (SCADA) and Process Control Systems (PCS).

http://dx.doi.org/10.1016/j.comnet.2016.05.010
1389-1286/

message to r_c and this node send it to r_b, letting the message pass alternatively through AS2 and AS3.

Our scheme is based on two important assumptions. The first is that amongst a collection of alternative overlay routes between two sites, there will be a subset which will be fast enough to perform reliable and timely communication in the presence of faults. The second is that it is used to send only control traffic and that that traffic consumes negligible resources (e.g., bandwidth) in comparison to the rest of the traffic.

The key objective of our work is to devise a *practical and non-intrusive* solution to achieve timely and reliable communication with *high probability* in current GDIIs, taking three requirements into consideration: (1) *Compatibility with current GDIIs:* JITER should allow seamless integration with current GDIIs, without requiring major changes to the operation and organization of existing networks; (2) *No wide-area IP network changes:* the solution should not require any special support from the underlying network (e.g., resource reservation). Timeliness should be obtained on top of best-effort communication channels such as those provided by IP-based networks, and therefore, JITER cannot ensure strict hard real-time properties (e.g., like in small-scale real-time networks); and (3) *Cost consciousness:* JITER should use redundancy parsimoniously, avoiding expensive solutions like traffic flooding.

Unlike previous works on overlay networks, which aim to improve reliability by detecting and deviating communication from faulty and congested paths, JITER aims to provide soft real-time communication by securing individual message delivery deadlines with high probability. To achieve this goal, JITER uses temporal and spatial redundancy judiciously. In a nutshell, each message is sent through several overlay channels, possibly from different service providers: one base channel plus a few backup channels. In the presence of delays or omissions, the message may be retransmitted. A novel channel scheduling policy in the JITER algorithm, which we call *just-in-time*, selects the base and backup channels not to be the fastest, but the ones which match each message's deadline needs (some faster, some slower). The reader may wonder that it is impossible to guarantee real-time behavior on best-effort IP networks. As a matter of fact, not even hard real-time systems have 100% coverage: they have to achieve *sufficient* coverage [53].

We implemented JITER and evaluated it experimentally by placing nodes in several regions of the Amazon AWS cloud offering (in the EC2 service). Moreover, we evaluated the strategy by simulation of scenarios over a realistic model of a wide-area utility network (the Italian electric power infrastructure) with both accidental and malicious faults (DDoS attacks). Its effectiveness and costs were compared with several other overlay/multihoming routing algorithms in the literature. The evaluation showed that JITER provides better timeliness and reliability guarantees than the other schemes, very close to those of a flooding strategy but sending much fewer messages.

The paper provides the following contributions:

1. JITER, the first (to the best of our knowledge) wide-area IP overlay network algorithm and architecture with the objective of providing message latency and reliability guarantees;
2. A comparative analysis of several overlay networks proposed in the literature, showing that JITER provides better timeliness;
3. The description and modelling of a representative critical information infrastructure, the Italian power system utility network, which we believe to be of use as a benchmark for future studies.
4. An evaluation of JITER and other techniques in providing timely communication between Amazon EC2 regions. This analysis also shows the communication latency and path diversity between the EC2 availability zones.

2. Related work

There is a vast bibliography on the topic of the paper, so this section is necessarily a summary. It presents work along the following axes: the properties we want to provide (QoS, timeliness); existing challenges (network failures), the techniques we use (overlay networks, multihoming); the scenarios we consider (clouds, critical information infrastructures).

QoS in multimedia networks. We assume that the network provides only a best-effort service, with no latency and bandwidth guarantees. It is possible to have these guarantees, e.g., by using DiffServ [38] or ATM [16]. However, these services are not provided by the generality of Internet service providers (ISPs), especially in a geo-distributed context, which would constrain the applicability of our solution. It is also possible to use leased lines, which are rather expensive. On the other hand, many ISPs provide phone and TV over IP, which have timeliness requirements. These providers usually employ fast convergence mechanisms for sub-second recovery from link and router failures: bidirectional forwarding detection [33], stateful switchover [12], and fast hello packets [11]. Nevertheless, deadlines are frequently missed, causing image freezes of several seconds. We propose a solution that does not require such guarantees from the network, only plain Internet-like IP network service.

Peer-to-peer networks have been proposed as a solution to stream multimedia over best-effort networks such as the Internet. CoopNet is one of the first of this line of research [39]. It focus on live streaming and leverages the notion of multiple description coding, i.e., of encoding audio and video into separate streams. Zigzag provides scalable single-source media streaming using an application-layer multicast tree [51]. Promise is a peer-to-peer media streaming system that supports multiple senders and allows one recipient to receive media from several senders [27]. These systems aim to provide QoS guarantees, but not specifically the delivery of messages before a certain, possibly short, deadline. Moreover these systems tolerate some level of packet loss, whereas we are interested in delivering all messages. On the other hand, these systems handle much more traffic, as they are targeted at continuous media (audio, video).

Timeliness in networks. When the web started to be adopted for commercial purposes, it became clear that there are limits on the time users are willing to accept for replies to arrive, i.e., that there are timeliness (maximum latency) requirements for web communication. Content distribution networks (CDNs) like Akamai [19] appeared as a solution to this problem [7,40]. The approach consists essentially in placing content geographically near its consumers, reducing the latency. This solution is unfeasible for the applications we envision as they do not have content to distribute, but control messages that have to be sent over a distance that cannot be reduced. There are common issues though, e.g., ensuring path diversity [7].

Recently there has been some work on the problem of ensuring timeliness in datacenter networks (e.g., [52,57]). The fundamental difference of these works in relation to ours is that they require modifications of the network devices. Consequently, these solutions can not be easily adapted to public/legacy networks, like the Internet and other WANs. However, they increase the significance of our work, since they motivate the need for solving the deadline propagation problem in the geo-distributed inter-datacenter communication, for applications that span multiple datacenters, without modifying the network.

Network failures in IP backbones. Network backbone failures may adversely affect IP routing and delay or disrupt information in-

Table 1
List of the strategies evaluated. OV and MH means that an overlay or multihoming are used.

Strategies	Technique	Basic idea	Reference
Best-Path	OV	Send message through the best overlay channel. In case of failure, retransmit message at most 3 times using a timeout of 3 seconds using the best overlay channel.	RON [4]
JITER^0	OV/MH	JITER without backup channels.	This work
JITER^1	OV/MH	JITER with 1 backup channel.	This work
Flooding	OV/MH	Send message a single time using all available overlay channels.	N/A
Multi-Path	OV	Send message through 2 overlay channels: the direct channel and a randomly chosen overlay channel (direct or not).	Mesh-routing [49] with variation in [6]
Hybrid	OV	Send message through one direct channel. If failure, send message via 4 randomly chosen overlay channels (direct or not).	SOSR [25]
Round-Robin	MH	Send message in a circular fashion alternating among all existent access links. Retransmit 3 times at most using a timeout of 3 seconds.	N/A
Primary-Backup	MH	Send message always through one specific access link until it fails. In case of failure and if there are redundant links, pick another one. Retransmission scheme as RR.	Used in the Italian backbone

frastructure communication. Studies on the impact of failures in IP backbones have shown that they occur daily, being often the result of problems either at or under the IP level [35]. Some of these problems can lead to network instability periods and disrupt applications [30]. Other failures come from interference of misconfigured or obsolete routing protocols running in customer networks connected to the ISP backbone [55]. Even stable core routing infrastructures, BGP-based, are prone to failures [46,48]. Malicious faults can also happen as a result of acts of hacktivism, cyber-crime or cyber-terrorism [56], such as distributed denial-of-service (DDoS) attacks, in which the attacker(s) use a large number of computers (bots or zombies) to generate traffic and cause congestion [29,36]. Depending on the capabilities of the attacker, the rate of messages delayed and lost can be alarmingly high. Research on this matter is vast and many ways of countering those attacks have appeared [44], but a final solution is still to be found.

Overlay routing. Nodes of an overlay network relay messages through the virtual paths among them, according to application-level criteria. This fits quite well with applications with specific requirements, hard to satisfy by normal wide-area networks. Therefore, over the years application-aware overlay routing solutions have been proposed, with various virtual channel selection schemes [2,4,6,49,50]. However, to the best of our knowledge none of these works aims to provide *latency guarantees*, which is the objective of our work. The most common objective of overlay routing algorithms is to deviate traffic from channels that are faulty or congested. For instance, RON monitors the network to decide to route messages directly or through an overlay channel [4]. The works nearest to ours have the objective of improving end-to-end communication latency, not of attaining individual message delivery deadlines [2,49]. Spines, uses a dense overlay network with several overlay nodes per-channel, recovering missing packets in a per-hop basis [2]. This recovery is attempted only once, since Spines targets video transmission, in which missing packets are undesirable, but acceptable to a certain level. Mesh-routing uses XML routers and the Diversity Control Protocol for multicasting data [49]. Although timeliness appears to be a requisite, it is not clear whether or how it would be achieved in stringent scenarios with, for example, persistent packet losses caused by long-term congestions [6]. OverQoS explores the controlled loss virtual link abstraction to provide statistical loss and bandwidth guarantees [50]. Han et al. introduced the idea of topology-aware overlay routing to improve the diversity of paths when detouring traffic to escape congestion or failures [26]. However, the topology may change due to changes at network (IP) layer routing, so we do not create an overlay network based on the topology, but instead select overlay channels dynamically taking diversity into account.

Table 1shows a comparison of RON, Mesh-routing and other overlay routing strategies with JITER.

Multihoming. Multihoming allows hosts to access a WAN through two or more redundant links, to resist network failures like those exemplified above and improve properties such as availability and performance [9]. Information infrastructure stakeholders frequently deploy IP connections contracted with more than one ISP [1]. The work closest to ours in the sense of exploiting overlays and multihoming is MONET [5], but its objective is to mask faults and improve the availability perceived by web clients. On the contrary to JITER, MONET does not take latency explicitly into consideration, as it does not aim to meet message deadlines.

The initial ideas of JITER appeared in a workshop position paper some years ago [15]. That paper also explored overlay networks and multihoming. The present paper thoroughly improves on that earlier version in several ways: the algorithm was improved, is now formalized and its properties discussed; we compare it analytically with other representative routing strategies; we implemented it and have experimental results; the simulations is way more complete and realistic (e.g., underlying network routing effects are accounted for, more strategies are compared) and provide more results.

Cloud communication and SDNs. The popularity of software defined networks (SDNs) promises to increase the possibility to control the network fabric by applications. In particular, recently Google showed how it manages its dedicated inter-datacenter backbone (a WAN) using SDN technology to achieve impressive levels of bandwidth utilization without sacrificing application SLAs (including latency) [32]. Their design is based on a centralized traffic engineering algorithm that controls the network. Although not explicitly designed for timeliness, this solution can solve or at least alleviate the need for an application level solution like JITER. However, we do not envision solutions like that being used for systems spanning multiple administrative domains (without centralized control) or for critical information infrastructures (GDIIs) that usually do not own the communication backbone (e.g., power grid operators).

Critical information infrastructure communication. Ratatoskr [54] exploits several channels and retransmissions to provide communication timeliness in critical infrastructures, but on the contrary of JITER it does not try to enforce deadlines explicitly and it is based on a publish-subscribe middleware, GridStat [24]. Esposito et al. present a broadcast protocol for publish-subscribe systems [22]. This protocol aims to support reliable communication among the nodes of an overlay network based on network coding and gossiping [21]. Although the paper mentions timeliness as a desirable property, the protocol neither considers the

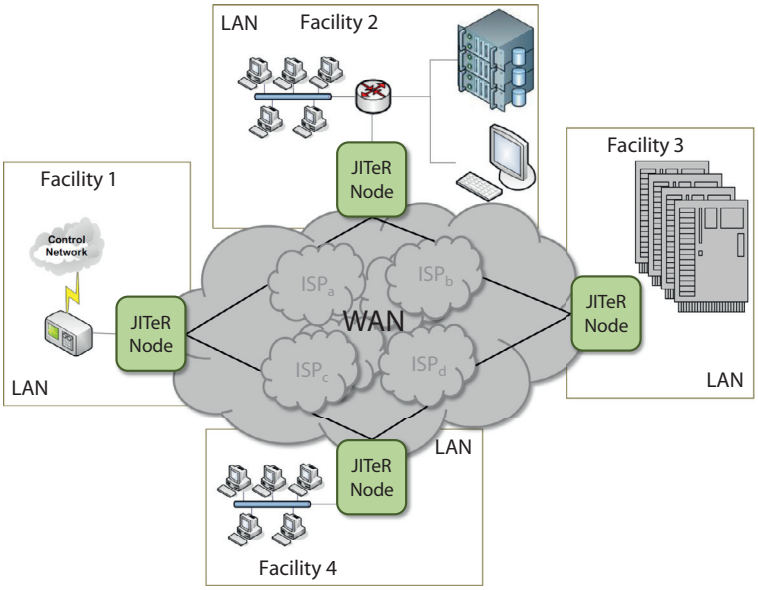


Fig. 1. WAN-of-LANs model of an information infrastructure using JITER.

existence of deadlines nor takes into account communication delays. The protocol is similar to flooding as it aims to deliver the messages to all nodes. Todai is a peer-to-peer data dissemination scheme for large-scale complex critical infrastructures [10]. Similarly to JITER, it leverages an overlay network in order to improve communication in GDIIs. However, its main goals are to provide reliability, scalability, and resilience (using semi-active replication for this purpose), whereas JITER's main goals are timeliness and reliability, as it is focused on delivering control messages only.

3. The JITER algorithm

This section presents JITER, a channel selection scheme that leverages the available connection redundancy and diversity to provide timely and reliable delivery of critical messages with high probability.

3.1. Design rationale

WAN-of-LANs structure. The design rationale of JITER is driven by the architecture of modern GDIIs. GDIIs are geo-distributed over several facilities, following a WAN-of-LANs model (see Fig. 1). In this model, depending on the kind of infrastructure (e.g., utilities, cloud providers), facilities can be any of: cloud datacenters, corporate offices, substations, SCADA command and control centers, etc. Facilities generally have high connectivity links, the LAN-type part, interconnected by a point-to-point wide-area network, the WAN-type part. Each LAN (or set thereof) of a facility is logically connected to the WAN through a JITER node, which executes the overlay and multihoming channel selection algorithm. If the company has too few facilities, helper JITER nodes can be placed somewhere else, e.g., in cloud services. JITER nodes, however, are neither access routers nor used to send all the facilities' traffic, only *time-critical control messages*. That traffic is assumed to have negligible impact in terms of bandwidth. The JITER architecture makes no modification to the existing WAN network, and preserves legacy features of internal subnetworks and systems, only requiring the introduction of the JITER nodes.

Stable overlay network. Contrary to classical uses of overlay networks, e.g., in the context of peer-to-peer applications, whose granularity is often at the level of individual hosts, and whose dynamics is quite high, overlay networking in GDIIs is best performed at the inter-facility level, i.e., amongst JITER nodes. These are pretty stable in time as well (it is unlikely to have GDII facilities join and leave the system frequently) and this comes for free as a design principle which presents several advantages. In consequence, we assume each node knows all the other nodes of the overlay network. This allows aggressive re-routing and monitoring policies, fundamental for providing the strong timeliness properties desired. Finally, a JITER node can be replicated for fault tolerance and scalability reasons.

One-hop source overlay routing. JITER nodes define an overlay network atop a general IP network, and run the JITER algorithm to select overlay channels that are expected to provide timely communication. The JITER algorithm is a *one-hop source routing scheme*. The overlay route of each message is defined at the sender (source routing), based on the local knowledge of the state of the links, and is composed of at most one intermediate relaying JITER node (one-hop). The option of having a single hop, i.e., a single intermediate node, is due to its simplicity and the conclusion of Gummadi et al. [25] that *there is no considerable benefit in using more hops*.

Proactive monitoring. JITER does monitoring proactively in order to to deal with network changes. JITER nodes may be connected to the WAN via multihoming, i.e., by two or more access links provided by distinct ISPs, allowing messages to be transmitted through several overlay channels. Multihoming can only guarantee fault tolerance effectively if the ISPs' networks share a minimum amount of resources. GDIIs normally try to ensure this link independence in a best-effort and ad-hoc manner when selecting the ISPs. The JITER algorithm uses route inspection mechanisms to assess the degree of independence among the overlay channels [13], and keeps a metric of the status and quality of the links between pairs of JITER nodes, measured by the latency or *transmission time*[2] (*TXT*). Both kinds of information are updated periodically and used to guide routing decisions. In fact there is a tradeoff involved: if

[2] This time includes not only the physical-level transmission delay at the sender, but also other delays such as transmission delays, propagation delays, queuing and processing delays at routers

the periodicity is high the algorithm adapts slowly to the network conditions; if low more messages are sent.

Deadline-aware multichannel transmission. Since IP offers only best-effort communication guarantees, JITER has to use temporal and spatial redundancy to obtain the desired message latency and reliability assurances. Each message is transmitted using one or more tries, until either an ACK is received or the message reception deadline is reached.

In each try, the message is sent through one *base channel* plus *B backup channels*. These channels are selected in a way that improves the number of possible retransmissions of the message before the deadline. For each message, JITER starts by using a base channel that does not offer the best *TXT* but is fast enough to still permit messages to arrive in time and be retransmitted through other channels. This approach leaves the best *TXT* channels to be used: (1) for retransmissions, as the available time for delivering the message becomes shorter; (2) for transmitting other messages with a shorter deadline. This solution achieves some load balancing among messages with different deadlines scheduled for transmission within a given short interval. Messages with stringent deadlines are transmitted through the fastest channels, while messages allowing larger delivery times go through slower channels.

The backup channels are selected in such a way that they have as little correlation as possible with the base channel and between themselves (based on monitoring information about the used links and routers), while still being able to deliver the message within the time constraints. As a result, the messages are not sent *as fast as possible*, but *fast enough*, or just in time.

Immediate and incremental deployment. From the deployment point of view, using baseline IP protocols ensures immediate deployment, without the need for global changes at the network level. In consequence, GDII stakeholders do not need to add the JITER nodes immediately to all facilities: deployment may be incremental.

3.2. System model

The system is composed by a set of JITER nodes $\mathcal{R} = \{r_0, ..., r_{n-1}\}$. A JITER logical *overlay channel* interconnects two JITER nodes, and is implemented either by a *direct channel* or by a one-hop *indirect channel*. In this sense, each pair of nodes r_i, r_j (with $i \neq j$) is connected by at least one *direct channel* c_{ij}, provided by routing across the network IP level. When two nodes are connected by an *indirect channel*, another JITER node works as a relay. In this sense, an indirect channel connecting r_i to r_j passing through r_k, is the composition of two direct channels c_{ik} and c_{kj}, and denoted by c_{ikj}.

The notion of channel is, however, deeper than this. A channel is an abstraction defined at each JITER node. The algorithm takes advantage of multihoming, so each channel begins with the connection to one of the ISPs the node is connected to, from the ISP set $\mathcal{I} = \{p_0, ..., p_{n'-1}\}$. Whenever needed, we use a superscript to indicate the ISP. For example, if the direct channel is c_{ij}^{p}, then when r_i sends a message through this channel the message is first passed to ISP p. The recipient r_j typically receives the message from a different ISP q (the message can pass through several ISPs / autonomous systems, see Section 5.3); if it replies to the message, it sends the reply through q. For indirect channels, we employ a superscript with the ISPs to which the sender and the relay pass the messages. For instance, if the channel is c_{ikj}^{pq}, then the sender r_i passes the messages to ISP p and the relay r_k passes the messages to ISP q. If $p = q$ we abuse of the notation and write only one letter.

The algorithm is used to exchange control messages $m = \langle data, d \rangle \in \mathcal{M}$, where *data* is the content of the message and d the deadline by which it has to be delivered, relative to the instant when it was sent (it is a time interval, not an instant). These messages are assumed to be sporadic (do not create congestion) and small (fit in one IP datagram). In this work we disregard the delays that occur within the LANs of the facilities (typically smaller than 0.2 ms, much less than the values in a WAN), and we also ignore the processing delays in the JITER nodes, since these delays are usually much smaller than the WAN transmission times. In the cases where this is not true, one can assume a bound on these two delays T_{extra}, and use it to update the deadline d accordingly (one would use a deadline $d' = d - 2 \times T_{\text{extra}}$). A message is said to be *sent* when it is placed in the external transmission queue at the sender JITER node, and is said to be *delivered* when it is put in the internal transmission queue of the destination JITER node, to be forwarded to its destination by the receiver node.

We assume that channels are *lossy* and *asynchronous*, i.e., they can lose messages and there are no bounds on the communication delays within the WAN links. We also assume that corrupted messages are detected and dropped (e.g., using cyclic redundancy checks or message authentication codes), allowing only correct messages to be delivered.

Channel correlation. Pairs of channels are assigned correlation numbers in the range [0, 1]. 0 means that there is no correlation at all, i.e., that there are no transmission media, equipment or administration common to both channels; 1 means that these items are common to both channels.

There are several possibilities for the data employed to compute the correlation. A practical solution is to use the set of routers common to both channels, which can be obtained by running tools like *traceroute*. We consider that each channel c is characterized in terms of a set of *routers rout* such that, given the set of routers of another channel c', the correlation of the two channels can be computed from *rout* and *rout'* using the Sørensen-Dice coefficient: $s = 2|rout \bigcap rout'|/(|rout| + |rout'|)$. Data about routes has to be updated periodically to account for route changes.

An alternative solution is to consider diversity in terms of autonomous systems. The concept of AS has been evolving [45], but it suggests a set of routers under the same technical administration, so it can be used as a unit of diversity. This makes sense especially in world-wide networks, as our experiments with Amazon EC2 show that pairs of regions are connected by a considerable number of ASs. The correlation of ASs can be calculated similarly to what was given for routers.

A third solution would be to compute channel correlation based on path availability history, as proposed by Zhang and Perrig [59].

Channel latency. Each node keeps information about the transmission time (*TXT*) of every overlay channel. The actual measurement method of *TXT* is independent from the algorithm, but currently we approximate it as half of the round-trip-time ($TXT = RTT/2$). Measuring one-way delays is known to be difficult as it requires strict time synchronization between hosts [41]. For the sake of example, in the experiments for this paper we have used a method similar to the TCP protocol [42], estimating *TXT* using an exponential weighted moving average: $TXT = (1 - \alpha) \times TXT + \alpha \times TXT_{\text{measured}}$. If no normal traffic is being exchanged between the nodes, then messages are exchanged periodically to support these measurements. The protocol used to send these messages is UDP instead of ICMP, because JITER sends messages over UDP. Moreover, it was recently shown that the use of ICMP for this purpose is problematic due at least in part to strange processing that some routers do to packets [43]. If round-trip-time measurement messages for a direct channel take more than a certain threshold

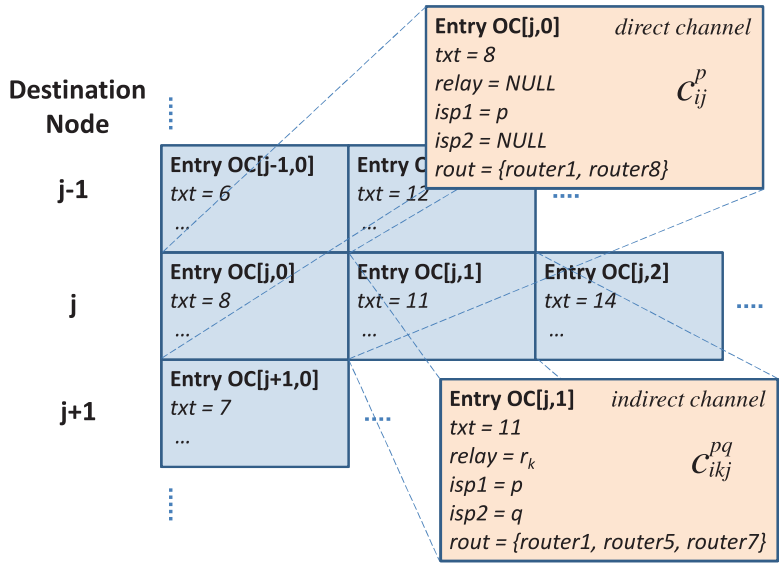


Fig. 2. The Overlay Channel (OC) matrix of a node r_i. A row j of the matrix contains the channels for sending messages from r_i to r_j ordered by *TXT*. Each matrix entry can represent either a direct channel (e.g., *OC*[*j*, 0]) or an indirect channel (e.g., *OC*[*j*, 1]).

$TXT_{\max}$ to be answered more than $Od_{\max}$ times, we assume this channel to have $TXT = +\infty$, and the algorithm stops using it. Notice that since these measurements are done periodically, if a channel is reestablished, it will again be considered for transmission.

If a message takes more than the current estimate *TXT* to be delivered (or is not delivered) then there is a *communication timing fault*. More formally, given any two nodes r_i and r_j and the estimate TXT_{ij} of the transmission time between them maintained at r_i, there is a communication timing fault if r_i sends a message to r_j at instant t_{send} and the message is not received at r_j by $t_{\text{send}} + TXT_{ij}$.

3.3. Supporting data structures

Every node r_i keeps a copy of a system-wide matrix *DC* that stores data about all direct channels c^p_{jk}, for $r_j, r_k \in \mathcal{R}$ and $p \in \mathcal{I}$. Each entry *DC*[*j, k, p*] has two fields: *txt* is an estimate of the *TXT* of the channel c^p_{jk}; and *rout* is the vector with the routers that messages traverse on this channel (as explained above).[3] Node r_i periodically updates the entries *DC*[*i*, ∗, ∗] with the new values estimated locally, and then disseminates this submatrix to the other nodes. Whenever a node r_k receives such a submatrix from r_i, it updates the entries corresponding to *DC*[*i*, ∗, ∗] in its own matrix. It is important to notice that different nodes do not need to have exactly the same *DC* matrix, but only converge to the same, just like in distance vector routing protocols.

The data in the *DC* matrix is used to populate the *overlay channel matrix OC* (see a representation in Fig. 2). A node r_i stores in the *OC* matrix data about all overlay channels, direct and indirect, available to connect itself and all other nodes r_j. Each entry of the matrix represents an overlay channel and contains a structure with the following fields: *txt* is the *TXT* of the channel (the sum of the *TXT*s of the two sub-channels if it is an indirect channel); *relay* is the node that relays the message (or NULL for direct channels); *isp1* is the ISP to which r_i is connected; *isp2* is the ISP to which the relay node is connected (or NULL for direct channels); and *rout* is the vector with the routers that have to be traversed. The entries *OC*[*j*, ∗] in the array correspond to the overlay channels towards the destination node r_j, and they are ordered from the lowest *TXT* to the highest, which places slow (i.e., high-latency) channels in the last columns.

[3] Although there is at most one relay per channel, there can be an arbitrary of (network-layer) routers per channel.

3.4. The algorithm

The algorithm works in a loop. Each cycle consists in sending the message though one or more channels, then waiting for an acknowledgment. If the acknowledgment is not received until a certain timeout, a new iteration of the loop is executed.

When node r_i wants to send a message to r_j, the algorithm selects from row *j* of the *OC* matrix a number of channels accordingly to the following rules:

Base channel: The base channel is chosen to maximize the possible number of retransmissions of the message through different channels before the deadline expires, and to allow some level of load balancing among messages with different deadlines, leaving the best channels for the messages with shortest deadline. Therefore, the base channel is not the one that provides the best *TXT*, only a *TXT* that is enough for the message to be delivered in time.

Backup channels: A total of *B* backup channels are selected in a way that: (i) minimizes the correlation with the base channel and between themselves; (ii) preserves their ability to deliver the message before the deadline.

Upon a transmission, a timeout of $2 \times TXT$ sets the waiting time for an acknowledgment of the message delivery. If no acknowledgment is received, then the selection process is executed again to allow for the retransmission of the message, possibly through other channels.

Algorithm 1 contains the pseudo-code executed by node r_i when a message $m = \langle data, d \rangle$ is to be sent to node r_j. The algorithm uses primitive *send*(TYPE, m, c) to transmit message m of type TYPE through an overlay channel c (TYPE is one of DATA or ACK). The destination node is implicit in c, and can be obtained with function *destination*(c). Function *channel*(*OC*[*j, i*]) returns the overlay channel of the corresponding entry of *OC*, while function *correlation*() gives the correlation between the routes of two *OC* entries. The number of channels in an *OC* matrix row *j* is denoted by #*OC*[*j*, ∗]. The deadline of message m relative to the instant in which the message was sent is in the field *m.d*.

The main part of the algorithm is in procedure *jiter_send* (Lines 3-21). This procedure is executed in two cases: *(i)* when r_i receives

Algorithm 1 JITER algorithm executed by node r_i.

```
1: upon there is a message m to deliver to r_j:
2: jiter_send(m, r_j)

3: procedure jiter_send(m, r_j)
4: if (OC[j, 0].txt > m.d) then
5:    signal NOT_ENOUGH_TIME(m)                {no time to send, exit}
6: end if
7: bc ← 0                                      {the base channel}
8: elapsed ← OC[j, 0].txt                      {overall used time}
9: while bc + 1 < #OC[j, *] do
10:   if (elapsed + 2 × OC[j, bc + 1].txt ≤ m.d) then
11:      elapsed ← elapsed + 2 × OC[j, bc + 1].txt
12:      bc ← bc + 1
13:   else
14:      exit loop                             {no time to use more channels}
15:   end if
16: end while
17: send(DATA, m, channel(OC[j, bc]))          {send to base channel}
18: P ← set of B backup channels where
      (i) OC[j, *].txt ≤ m.d and
      (ii) correlation(OC[j, bc], OC[j, *]) and between themselves is
           the lowest
19: ∀p ∈ P : send(DATA, m, p)                  {send to backup channels}
20: m.d ← m.d − 2 × OC[j, bc].txt
21: start_timer(m, r_j, 2 × OC[j, bc].txt)

22: upon expires timer for m to reach r_j:
23: jiter_send(m, r_j)                         {retry to send m to r_j}

24: upon (DATA, m, c) is received:
25: if (destination(c) = r_i) then
26:   send(ACK, m, c′)                         {c′: is the backward channel of c}
27:   deliver m to its final destination
28: else
29:   send(DATA, m, destination(c))            {relay m to its destination}
30: end if

31: upon (ACK, m, c) is received:
32: if (destination(c) = r_i) then
33:   stop_timer(m)
34:   signal OK_DELIVERED(m)
35: else
36:   send(ACK, m, destination(c))             {relay m to its destination}
37: end if
```

a message m to be transmitted through the WAN (Lines 1-2); *(ii)* when the timer that is started when the message is sent (Line 21) expires (Lines 22-23). Node r_i first checks if the fastest channel has a *TXT* low enough to allow the message to be received before the deadline, i.e., if *TXT* is at least equal to the time available to send the message (Line 4). If the *TXT* is not low enough, an error signal is raised (Line 5).

Otherwise, the algorithm will find the base channel by testing the $OC[j, *]$ entries in ascending *TXT* order (Lines 7-16). The algorithm predicts the maximum possible number of future retransmissions of m through different base channels based on the reference deadline value $m.d$. It progressively takes advantage of the distinct overlay channels available in the entries of OC, each of which with latency cost $OC[j, *].txt$. The algorithm chooses as base channel the one with the highest *TXT*, from those that in principle allow the transmission of the message, and sends the message through that channel (Lines 17).

Next, a set P of B backup channels is created (Line 18). Backup channels satisfy two conditions: (i) they expectedly allow the delivery of m before its deadline; and (ii) they have the lowest correlation with the base channel and between themselves. The message is sent through the backup channels (Line 19), and then the algorithm updates the message deadline and starts a timer (Lines 20-21). If the timer expires, the message is resent (Lines 22-23). Notice that the message can be resent without need due to a late ACK, but this is not problematic.[4]

When the node receives a DATA message m, it either delivers it to its final destination in the LAN, sending an ACK to the sender node (Lines 26 and 27), or relays m to its destination JITER node (Line 29), depending on its destination. If r_i receives an ACK message from node r_j, the timer is stopped and the algorithm terminates with success (Lines 33-34). Notice that the error signaled in Line 5 can be pessimistic, as the message may be delivered but the acknowledgment lost or received later, after the signal is raised.

3.5. Properties

The main property that the algorithm guarantees is timely message delivery with high probability, despite communication timing faults. Consider that bc is the index of the base channel (starting with 0) when a message m is first transmitted (Algorithm 1, Line 17) and that B is the number of additional backup channels used on each try (Line 18). The algorithm transmits m through $1 + B$ diverse channels in each try (execution of *jiter_send*), and at most $bc + 1$ tries are executed to send m. It means that the system tolerates at most $(1 + B) \times (bc + 1) - 1$ communication timing faults on overlay channels. This expression reflects the main idea of JITER: to explore space and time redundancy to increase the probability of timely message transmission. Naturally, the effectiveness of the redundancy employed is dependent on the characteristics of the network. If the network is very unstable (the channels' TXT estimates do not reflect the actual transmission times) and/or if the links exhibit a high correlation, neither JITER nor any other routing strategy can effectively ensure timely communication. However, given ISP diversity and considering the natural redundancy of well-engineered networks, our algorithm will explore both quite effectively, even under harsh scenarios of disasters or DDoS attacks, as we show in Section 5.2.

3.6. An example

This section presents an example to show: (1) the way a JITER node selects channels to transmit a message; (2) that bad channels are not used; and (3) that load balancing is achieved and the best channels are left for the messages with shortest deadline. We consider an overlay with four nodes $\{r_i, r_j, r_k, r_l\}$ and $B = 1$. Node r_i is connected to two ISPs, p and q, and sends two messages m_1 and m_2 to node r_k in a row, with deadlines respectively of 30 ms and 70 ms. Fig. 3 represents the line of the OC matrix with channels connecting r_i to r_j (line j of Fig. 2). Each column corresponds to a channel. The channel id was added to help our explanation.

Message m_1 has a deadline of 30 ms. Lines 7-16 of the algorithm conclude that to send m_1 there is time to use channels 0 and 1 ($2 \times 11 + 8 \leq 30$), but not channel 2 ($2 \times 14 + 2 \times 11 + 8 > 30$). Therefore, the first base channel used for m_1 is channel 1. For backup channel it chooses the less correlated channel that allows achieving the deadline, channel 5, which has correlation $s = 0.33$ with channel 1. Channel 9 is excluded because its *TXT* is too high. If an acknowledgment is not received until $2 \times TXT_2 = 22$ ms, then r_i picks the second base channel (channel 0) and another backup channel and retransmits the message.

[4] Instead of setting the timeout to $2 \times TXT$, line 21 might add to that value four times an estimative of the deviation, similarly to what is done for TCP's retransmission timer [42]. However, this would waste time that could be used to resend the message through a different channel and improve the chances of reception on time.

channel id	0	1	2	3	4	5	6	7	8	9
txt	*8*	*11*	*14*	*16*	*17*	*18*	*20*	*23*	*25*	*33*
relay	*null*	r_k	r_k	*null*	r_l	r_l	r_k	r_k	r_l	r_l
isp1	*p*	*p*	*p*	*q*	*p*	*p*	*q*	*q*	*q*	*q*
isp2	*null*	*p*	*q*	*null*	*q*	*p*	*p*	*q*	*p*	*q*
rout	*{R1,R8}*	*{R1,R5, R7}*	*{R1,R3, R5,R8, R9}*	*{R5,R7, R9}*	*{R1,R3, R5,R7, R8,R9}*	*{R1,R3, R8}*	*{R1,R3, R5,R7, R8,R9}*	*{R5,R7, R9}*	*{R1,R3, R7,R8}*	*{R5,R7, R9}*

Fig. 3. Line of the OC matrix at JITER node r_i with the channels connecting it to r_j.

For m_2 something similar happens. The deadline is 70 ms, so it has time to use channels 0 to 2 ($2 \times 14 + 2 \times 11 + 8 \leq 70$), but not channel 3, so the first base channel is 2. For backup channel it chooses among channels 1, 3, 7 and 9, all with correlation $s = 0.5$ with channel 2.

This example shows how nodes select channels. It also demonstrates that bad channels are never selected (channel 9 is never selected to send m_1). Finally, it shows load balancing in action: the two messages are sent through different base channels the first time they are sent. Its is important to recall that the OC matrix is quite dynamic: our aggressive monitoring strategy ensures that the values of this table (and the order of the channels) are constantly being updated, ensuring the system adapts to changes in the network conditions.

4. JITER implementation

Current implementation. We implemented the JITER nodes in Java (around 6000 lines of code). The prototype uses the JGroups toolkit[5] to manage membership, i.e., to keep and update views of the JITER nodes that are active. The prototype uses the functions of that toolkit to allow nodes to enter the group of JITER nodes, to leave that group, and to be removed automatically in case they become inaccessible, similarly to membership services in the literature [3]. Otherwise the prototype does not use the communication primitives provided by JGroups, but sends messages on top of UDP. The prototype also implements the Flooding and Primary-Backup strategies (see Table 1).

In normal operation the nodes constantly monitor the *TXT* between themselves, either explicitly by sending heartbeat messages, or implicitly by measuring the time taken to get replies to the data messages they send. The diversity among channels is periodically assessed using information about routers obtained using *traceroute*.

Scalability and node replication. The JITER node of a facility can be replicated to cope with more messages (scalability) and to tolerate faults. The basic idea is to replicate each JITER node in a set of hosts in the same geo-location. Each replica handles a fraction of the messages and availability is ensured by the other replicas if some of them crash.

The implementation is basically the following. Each replica has soft state reflecting cached data structures that are maintained in a coordination services such as Zookeeper [28]. The membership (list of replicas) of each node is also kept in Zookeeper, that can trivially detect replica failure and support replica addition and removal.[6] All replicas will monitor different channels of the network in order to update the data structures on Zookeeper, and will read this matrix to memory periodically (e.g., every few seconds). A client that wants to send a message using such replicated node will choose one of the replicas randomly and use it as its JITER node. The same thing happens when other JITER nodes want to use this node as a relay. This solution allows scaling up as long as Zookeeper is not the bottleneck, which is unlikely as it scales well with the number of read operations [28].

Access and admission control. JITER improves the timeliness of control traffic that we assume to be negligible in comparison to the overall network traffic. However, in practice access control and admission control have to be implemented to limit, respectively, who can send messages using JITER and how much traffic can be sent. There are several options to implement both mechanisms. For instance, access control can be based on SOCKS5 and admission control similar to ATM's [16].

5. Evaluation

This section evaluates JITER analytically, using simulations, and experimentally. These three evaluations complement each other and shed light on the fundamental characteristic of JITER and related strategies.

5.1. Analytical comparison of strategies

This section compares analytically JITER with other potential candidates to improve the timeliness of control traffic in wide-area IP networks, whether with overlays, multihoming or both. Table 1 describes the strategies, where we consider two possible configurations for JITER: using $B = 0$ or $B = 1$ backup overlay channels, dubbed JITER^0 and JITER^1, respectively. Results are not presented for higher numbers of backup channels because our experiments have not shown benefits in relation to JITER^1.

The various strategies can be roughly characterized in terms of three metrics: the number of times they can resend messages; the number of channels they use (overlaying); and the number of access ISPs they exploit (multihoming). Notice that these numbers are not constant, as they depend on the actual physical configuration of the network (e.g., the number of ISPs) and the runtime conditions (e.g., the message deadlines and the channels' *TXT* actually define how many retransmissions are possible in JITER). In any case, it is still possible to calculate a level of *fault tolerance* of each strategy: the number of faulty channels (channels that are interrupted or with very high delay) and access ISPs that are tolerated. These metrics mean that if that number of channels or ISPs fail altogether it is still possible for a message to be received in time, because there is still redundancy in the system (assuming that failures are independent). They do not mean that the message will actually be received because they disregard temporary issues, like short duration congestion that can affect the other channel/ISP and delay the message. It is also possible to obtain values of *cost* in terms of the number of extra messages transmitted.

[5] http://www.jgroups.org/.
[6] http://zookeeper.apache.org/doc/trunk/recipes.html.

Table 2
Analytical comparison of the algorithms. Refer to Table 1 for the evaluated algorithms descriptions (#ch is the number of overlay channels available).

Strategies	Technique	Features			Fault tolerance		Cost (extra messages sent)		
		#Resends	#Channels used	#ISPs used	#faulty channels tolerated	#faulty ISPs tolerated	no faults	one omission	two omissions
Best-Path	OV	3	4	1	3	0	1	2	3
JITeR^0	OV/MH	bc	$bc+1$	#ISPs	bc	#ISPs−1	1	2	3
JITeR^1	OV/MH	bc	$2(bc+1)$	#ISPs	$2(bc+1)-1$	#ISPs−1	2	2	4
Flooding	OV/MH	0	#ch	#ISPs	all−1	#ISPs−1	#ch	#ch	#ch
Multi-Path	OV	0	2	1	1	0	2	2	2
Hybrid	OV	1	5	1	4	0	1	5	5
Round-Robin	MH	3	#ISPs	#ISPs	#ISPs−1	#ISPs−1	1	2	3
Prim.-Backup	MH	3	#ISPs	#ISPs	#ISPs−1	#ISPs−1	1	2	3

These metrics and numbers are shown in Table 2. The symbol # means "number of" and bc is the index of the base channel when the message is first transmitted (Algorithm 1, Line 17), which gives the number of times the message can be sent through base channels minus one. Notice that #Resends accounts for tries to resend the message; for instance JITeR resends a message up to bc times. Briefly, the table was obtained the following way. The 3rd to 5th columns come from Table 1. The number of faulty channels tolerated (6th column) is equal to the number of channels used (4th column) minus 1. The number of faulty access ISPs tolerated (7th column) is the number of ISPs used by the scheme (5th column) minus 1. The cost with no faults (8th column) is the number of packets sent without faults (e.g., JITeR^1 always sends two packets per application-level message and flooding sends as many as the number of channels). With omissions this number increases (9th/10th column).

The table clearly shows that JITeR^1 and JITeR^0 explore all the dimensions of diversity, considering time, channel and access ISP redundancy. This suggests that they are able to achieve better timeliness than alternative schemes that do not explore all those options. Flooding exploits channel and ISP redundancy to the extreme, with the associated high cost of transmitting messages through all available channels (#channels). The Best-Path, Multi-Path and Hybrid strategies explore time and/or channel redundancy within a single ISP. Round-Robing and Primary-Backup explore time and ISP redundancy, but not channel redundancy through other facilities.

5.2. Scenario-based simulation: critical information infrastructure

This section presents a simulation-based evaluation of the strategies described in Table 1 using a model of a real-world critical information infrastructure. We developed a detailed model of the Italian power grid GDII using information publicly available [18,47]. We opted for simulation because it would be virtually impossible to have access to any GDII production environment.

The evaluation aims to answer two important questions: (1) Given a set of messages with different deadlines, to what extent are these messages received in time when employing the various strategies? (2) What are the transmission costs incurred by these strategies?

Simulated network environment. Simulations were carried out on the J-Sim simulator.[7] The simulated network is based on a ISP backbone topology that is used by the largest Italian power grid company for control communication [47]. This topology is composed of 31 routers and 51 direct channels, as depicted in Fig. 4(a). Each router is capable of pushing data at 1 Gbps, and network channels provide a propagation delay of 50 ms. To represent multihoming, we replicate the network topology to create two fully decoupled ISP backbones.

On the underlying network topology, we considered 17 candidate gateways (the polygons in Fig. 4(b)), each corresponding to an Italian region. Gateways are located at a control center (CC) or a substation (SS), and they can send data at 100Mbps. Four special gateways (circles in the figure) are located in the main Italian cities in different regions. They connect regional remote controller stations, thus being responsible for passing along all traffic related to them (e.g., receive monitoring data from various SSs and transmit commands to reconfigure a SS). The remainder gateways (squares in the figure) are in charge of traffic relative to a SS (e.g., signaling messages sent by the SS to the respective CC). The gateways correspond to the JITeR nodes , but they can run any of the strategies of Table 1. To limit the number of transmitted messages, we used only 7 of the 17 gateways in the simulations: the 4 CCs and 3 other well positioned nodes (dark squares in Fig. 4(b)). Every node had access to the WAN through two redundant links, each one provided by a distinct ISP.

Table 3
Power control messages of the simulation.

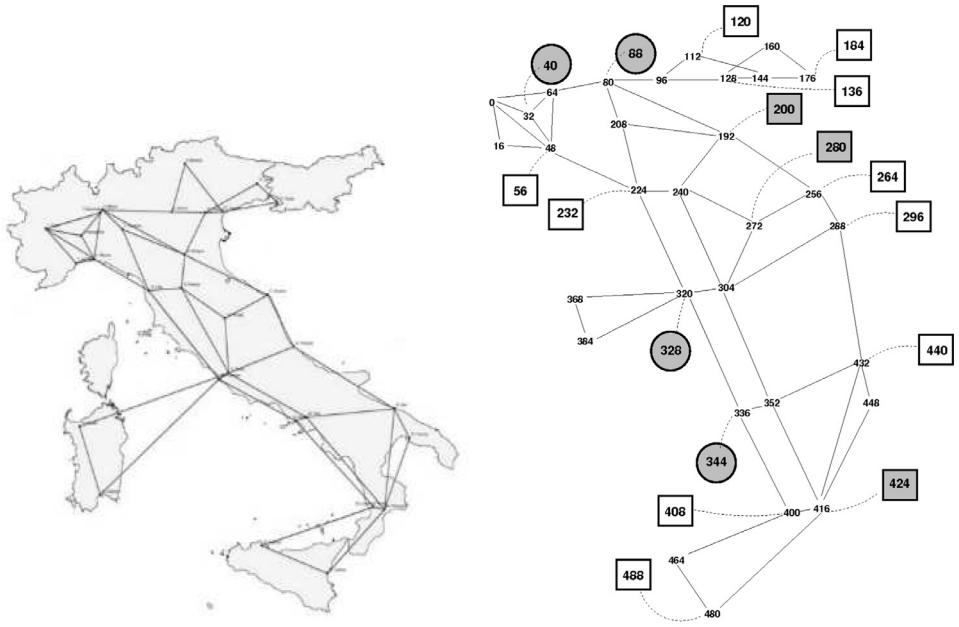
Message	Deadline	Period	Description
CC-STATE	4s	4s	CCs state info exchange
SS-STATE	1s	2s	SS state info to its CC
SS-ALARM	1s	sporadic	SS alarm to its CC
CC-CMD	2s	sporadic	CC command to a SS
CC-HPCMD	1s	sporadic	CC high-priority command to a SS

Workload. The traffic of each CC and SS nodes involved in the setup was generated separately. We created 17 different traffic sources with the same duration of a simulation, totalizing 90,134 messages sent. The same traffic sources were used in all strategies.

The workload was generated based on information about typical power control traffic [18]. Traffic can be periodic or sporadic and have different deadlines, according to the type of operation. Five distinct traffic patterns were identified, as shown in Table 3. All these patterns are applied in the simulation, with periodic messages starting to be sent after a random initial delay.

Faultloads. The faultloads were generated for the simulation interval of 5 hours. A faultload has a total number of faults $f = 148$ to be injected across all network components. Faults are injected in backbone routers, links and gateway interfaces (to emulate the effect of a failure on the ISP access routers). A fault number f_c was defined for each component class, following the distribution of unplanned ISP IP backbone failures shown in [35].

[7] http://sites.google.com/site/jsimofficial/.



(a) Italian ISP topology [47]. (b) Simulation topology.

Fig. 4. ISP network topology: reality and simulation.

- *Faultload 1 (fault-free).* Ideal, no failures.
- *Faultload 2 (accidental faults).* A scenario where there are accidental problems in the WAN. Accidental faults are generated using a combination of three network failure models from the literature. The model in [35] is used to determine the failure starting time and localization per network component, and the failure duration is based on the models in [14,34]. The resulting model states that *(1)* the starting time of failures is randomly picked following a Weibull distribution over the network-wide simulated time window [35]; *(2)* 30% of all failures last more than 30 seconds according to a Pareto truncated distribution [14], while the remainder ones last up to 30 seconds following an Exponential distribution [34]; *(3)* for each class of network devices, an individual element is selected according to a Power-Law based distribution [35] to inject the fault.
- *Faultload 3 (crisis).* A more stringent scenario where the WAN is subject to accidental and malicious faults. This faultload is similar to faultload 2, but it also includes faults that have a longer duration and affect more components to simulate DDoS attacks. After analyzing data about real DDoS attacks in the literature [29,36], a model was built with the following characteristics: *(1)* the initial time of a single failure is obtained by uniformly selecting a random number within the simulation interval; *(2)* 80% of all failures last more than 30 seconds according to a Pareto truncated distribution, and the remainder 20% last up to 30 seconds following an Exponential distribution; *(3)* for each class of network components, an element is uniformly chosen for fault injection.

Simulation results. Table 4 shows the results of the simulations. Two metrics are used to evaluate the strategies of Table 1 in the simulations: the *number of missed deadlines*, which is a measure of the *effectiveness* of the scheme to achieve timely communication; and the *percentage of extra messages sent*, which is a measure of *cost*. This last metric is the total number of messages sent by the scheme minus the number of messages transmitted by the application, divided by the number of messages sent.

Table 4
Simulation results in terms of missed deadlines (effectiveness) and % extra messages sent (cost) considering fault-free (FF), accidental faults (AF) and crisis (C) scenarios. for the evaluated algorithms descriptions.

Strategy	Missed deadlines			% of extra messages sent		
	FF	AF	C	FF	AF	C
B.P.	0	101	1,527	0.00	0.16	1.76
JITeR^0	0	0	97	0.04	0.96	49.29
JITeR^1	0	0	37	100.02	100.41	167.63
Flood.	0	0	5	2410.09	2410.09	2410.09
M.P.	0	43	8,041	100.00	100.00	100.00
Hybrid	0	111	8,627	0.00	0.21	26.62%
R.R.	0	109	9,600	0.00	0.16	18.67
P.B.	0	5	1,535	0.00	0.01	1.78

The table shows that there were no deadlines missed in the simulations of any of the strategies in the failure-free (FF) scenario. However, a few of the schemes incurred in additional costs in terms of extra messages. Not surprisingly, Flooding was the most expensive scheme, as it sends messages through all channels. Its cost was several orders of magnitude higher than those of JITeR^0 and JITeR^1. However, on the contrary of JITeR, Flooding does not need to keep information about the transmission time of the overlay channels, so there is a tradeoff involved. Comparing our strategies with the others, one can conclude that applying an additional backup channel implied an increase in the overhead, since JITeR^1 duplicates each transmitted message by always exploring the backup channel (like Multi-Path). JITeR^0 does not use a backup channel, so the cost is negligible.

When accidental faults (AF) are considered, we observe that the Hybrid algorithm had the highest amount of deadlines missed. Flooding missed no deadlines, but also did not JITeR^0 and JITeR^1 at much lower costs. Interestingly, the non-overlay primary-backup scheme that is used by most GDIIs outperformed some of the overlay strategies, confirming that often it can cope with accidental fault scenarios.

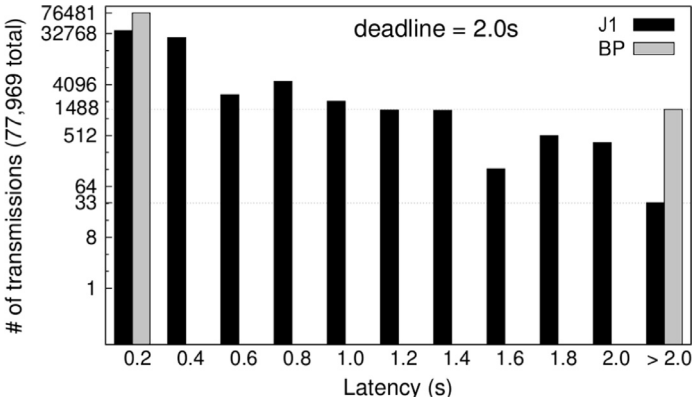


Fig. 5. Number of messages delivered by JITER[1] and Best-Path (RON) per range of latencies (faultload 3; log scale).

In the crisis (C) scenario the non-overlay Round-Robin scheme exhibited the highest number of deadlines missed: 9,600 out of 90,134, which is more than 10%. Similarly to the previous scenario, Primary-Backup had better efficiency at a lower cost than both Hybrid and Multi-Path. Flooding had a small number of deadlines missed (5), which shows that there were cases in which it was impossible to mask all faults (there were network partitions). When our solution was employed without backup channels (JITER[0]), it missed 97 deadlines, which is about 1% of the missed deadlines by the RR scheme (the worst performing strategy) and 6% of those missed by BP (the best strategy excluding our solution and flooding). These percentages decrease further with JITER[1] since it adds one backup channel to the basic scheme – here, the fraction is about 0.4% of the missed deadlines by RR and 2.4% of those missed by Best-Path (i.e., RON).

The results for missed deadlines of JITER are not as good as flooding's in this scenario, but the cost of the latter is much higher. The JITER approach allows a tradeoff by setting the number of backup channels, as shown by the improvement from JITER[0] (no backups) to JITER[1] (one backup) from 97 to 37 missed deadlines.

Just as observed with fautload 2, JITER had a higher overhead than some other strategies. However, even though Multi-Path and Hybrid explore the spatial redundancy as JITER[1], they could not exhibit the same progress in terms of reducing the number of missed deadlines. The difference comes from the overlay channel selection algorithm employed by our strategy.

Best-Path either delivers the messages in the initial 0.2s interval or too late. JITER[1] instead does not try to achieve the best latency, so only about half of the messages are delivered in the first 0.2s, and the rest is distributed over the bins, but it delivers more messages on time.

Fig. 5 compares in more detail the behavior of JITER[1] and Best-Path (BP) in the crisis scenario (Faultload 3). Recall that the BP strategy aims to minimize the communication latency by picking the channels with the lowest *TXT* to send the messages. Notice that the y-axis is logarithmic and that this figure only displays data for messages with deadlines of 2 seconds. The graph shows the number of messages that arrived with different latencies to the destination using bins with size of 0.2s. It can be observed that Best-Path either delivers the messages in the initial 0.2s interval or too late (1,527 messages miss the deadline of two seconds). JITER[1] instead does not try to achieve the best latency, so only about half of the messages are delivered in the first 0.2s, and the rest is distributed over the bins with increasingly higher latencies. However, when compared with Best-Path, JITER[1] only misses a few deadlines (37 messages arrive after two seconds), showing that being just-in-time is a better strategy than being early in a utility network application scenario.

Table 5
Results of the Amazon EC2 experiments in deadlines missed (effectiveness) and extra messages sent (cost).

Strategy	% missed deadlines (per deadline)				% extra
	250 ms	500 ms	1sec	Total	messages
JITER[0]	20.70%	0%	0%	6.92%	19%
JITER[1]	20.06%	0%	0%	6.77%	134%
Flooding	20.49%	0%	0%	6.87%	300%
Prim.-Back.	43.57%	0%	0%	14.58%	0%

Table 6
Average *TXT* between Amazon EC2 pairs of nodes during the period of the experiments (in milliseconds). The distances between nodes are rough estimates (in Km).

Node pair	Average	Standard dev.	Distance
N.Virginia-Oregon	53.76	4.00	4000
N.Virginia-S.Paulo	73.09	8.54	6000
N.Virginia-Ireland	49.73	1.97	6000
N.Virginia-Tokyo	100.27	5.89	11,000
Oregon-S.Paulo	111.18	4.90	9000
Oregon-Ireland	89.45	2.25	8000
Oregon-Tokyo	69.01	10.15	8000
S.Paulo-Ireland	111.66	7.19	7000
S.Paulo-Tokyo	147.29	11.75	16,000
Ireland-Tokyo	138.63	11.49	10,000

5.3. Amazon EC2 experiments

We run the JITER prototype in the Amazon EC2 service. We deployed 5 nodes (micro instances) in 5 different Amazon AWS regions: Ireland, Tokyo, S. Paulo, Oregon, and N. Virginia. We run experiments continuously for around 100 hours with more than 55 thousand messages sent. Each node sent messages to each of the other nodes in round-robin, using JITER[0], JITER[1], Flooding and Primary-Backup, with deadlines of 250 ms, 500 ms, and 1s. All messages had a payload of 1kB. We considered a single access ISP per node, as to the best of our knowledge it is not possible to use multihoming in Amazon EC2.

A difficulty in the experiments is the assessment if a message is received by the deadline or not. To escape this issue, in the experiments we interpreted the deadline as being the deadline for the sender to receive an acknowledgment of the reception of the message, not for the receiver delivering the message.

Experimental results. Table 5 summarizes the experimental results. The main conclusion that can be extracted from the table is aligned with the simulations: Flooding, JITER[0] and JITER[1] obtain similar results, with JITER[1] slightly better than Flooding, and this one slightly better than JITER[0]. Primary-Backup gives worse results (with 250 ms deadlines). Notice that although Flooding uses all overlay channels it does not retransmit the message, which explains why JITER[1] performs better. In terms of additional messages sent the results are almost the opposite: Primary-Backup is the cheapest, very closely followed by JITER[0] – only 19% more messages for a very low number of deadlines missed –, then JITER[1] and Flooding.

A second observation is that only messages with deadline of 250 ms miss the deadline. To understand this we need to have an idea of the *TXT* of the communication between the nodes during the period of the experiments. This information is provided in Table 6. This table shows clearly that 250 ms is short for sending and getting back an acknowledgment between the most far apart nodes (plus S. Paulo-Ireland).

Path diversity. We wanted to understand the path diversity existing between the 5 Amazon EC2 regions, as diversity is important

Table 7
ISPs and number of ASs (between parentheses) connecting Amazon's regions obtained using *lft* on August 2nd 2013.

From \ To	Ireland	N. Virginia	Oregon	S. Paulo	Tokyo
Ireland	–	Tinet (2), Amazon (4)	Tinet (3), Amazon (4)	Level3 (3), Amazon (4)	Level3 (3), BTN (1), Amazon (5)
N. Virginia	NTT (2), Telia (3), Amazon (4)	–	Amazon (5)	NTT (2), Level3 (3), Amazon (2)	Qwest (2), BTN (1), Amazon (4)
Oregon	NTT (2), Telia (3), Amazon (4)	Amazon (5)	–	NTT (1), Level3 (2), Amazon (3)	NTT (3), Amazon (3)
S. Paulo	SeaBone (1), Tinet (3), Amazon (3)	SeaBone (2), Amazon (4)	Telefonica (2), NTT (2), Amazon (3)	–	SeaBone (2), Tata (1), Amazon (3)
Tokyo	NTT (2), Telia (2), Amazon (4)	NTT (3), Amazon (5)	KDDI (4), Amazon (5)	NTT (2), SeaBone (2), Amazon (3)	–

Table 8
JITER control overhead. *#Nodes* is the number of nodes. The values were calculated considering IPv4, 2 ISPs, average of 10 routers in direct channels (*AvgRouters*), 3 requests/replies to measure *TXT* (*Rep*), and size of these messages of 50 B (*PingSize*).

Cost	Formula	5 nodes	17 nodes	50 nodes
Size of Matrix DC at a node	$(\#Nodes-1)\#ISPs(4+4(AvgRouters))$	352 B	1408 B	4312 B
Size of Matrix OC at a node	$(\#Nodes-1)(\#Nodes-2)(\#ISPs^2)\times(4+4\times2(AvgRouters)+4+1)$	7120 B	85440 B	837312 B
Bytes sent (all nodes)	$\#Nodes(\#Nodes-1)\times\ (2Rep\times PingSize+SizeOfDC)$	13.0 KB	464.6 KB	11299 KB

to tolerate faults that affect several routes. We ran the *lft* command between the 5 regions every hour for two weeks in August 2013. *lft* is essentially a version of *traceroute* that shows the ASs traversed. There were occasional changes but the ISPs and ASs crossed remained mostly constant during that period, so we show data taken at a single day, August 2nd.

Table 7 presents the ISPs connecting the nodes deployed in the regions of Amazon EC2. A first observation is that there is much diversity of ISPs used, 10 for connecting the 5 nodes. A second interesting conclusion is that the paths between two nodes are often different depending on the direction. For instance, from N. Virginia to Ireland the path traverses NTT and Telia, whereas from in the opposite direction it crosses Tinet, not NTT or Telia. The network connecting N. Virginia and Oregon (and North California, not shown) has only routers from Amazon.

The diversity of ASs is shown in the same table. The number after each ISP (and Amazon) is the number of ASs of that ISP crossed by the path. The number of ASs varies between 5 and 9. Again this suggests a considerable level of diversity, creating opportunities for the deployment of overlay solutions such as JITER.

5.4. Control costs

JITER has some control overhead in relation to the simplest alternative strategy: flooding. This session evaluates this overhead.

In terms of *memory footprint*, nodes store two matrices, DC and OC (Section 3.3). Matrix DC stores *TXT* and *rout* for every other node over all ISPs. This matrix has an average size provided by the formula in the second row of Table 8, where *#Nodes* is the number of nodes, *AvgRouters* is the average number of routers of a direct channel, 4 bytes is the size of the integers that represent *TXT*, and 4 bytes is the size of IPv4 address that represent a router (a public IP address of one of the router's interfaces). Matrix OC has data about the overlay channels between the node and all the others (see Fig. 2). This matrix has $(\#Nodes-1)(\#Nodes-2)(\#ISPs)^2$ cells that store the same data as the cells of matrix DC, the IP of the relay node, and the two ISPs. The size of the matrix is shown in the third row of the table, assuming a single byte is used to store the identifier of the ISP.

In terms of *communication overhead*, each node has to measure the *TXT* and send the DC matrix to all other nodes periodically. This cost is provided by the formula in the last row of the table, where *Rep* is the number of messages sent to measure the *TXT* to each node (ping), *PingSize* the size of that message, and *SizeOfDC* the size of the DC matrix at a node (second row of the same table). The communication overhead depends strongly on the period considered; it is higher if the period is short, and smaller if the period is long, as already pointed out.

The last 3 columns of the table provide concrete values for the memory footprint and the communication overhead. 5 nodes is the number of JITER nodes we used in the AWS experiments, so it expresses the overheads in that scenario. Next, 17 nodes is the number used in the simulations of Section 5.2. Finally, 50 nodes is a value that we consider large for this kind of scenario, which we depicted simply to show that the costs are reasonable. Traffic of 11 MB (bottom right) may seem considerable, but recall that this is not per second, but per whatever period is used (e.g., per minute or 10 minutes). Nevertheless, this value grows exponentially with the number of nodes.

6. Conclusion

We presented the design and validation of an algorithm, called JITER (*Just-In-Time Routing*), which routes deadline constrained messages at application level, using novel overlay and multihoming channel selection strategies, leveraging the natural redundancy of geo-distributed GDII's networks.

JITER solves an important problem, of providing real-time message latency and reliability assurances for traffic in wide-area networks offering non-differentiated IP services, although not with 100% coverage of the timeliness properties. Design goals met in our approach, in order to improve its applicability, included: practicality and non-intrusiveness; compatibility with current GDIIs; no wide-area IP network changes; cost consciousness.

Analytical, scenario-based and experimental evaluations with an implementation of JITER nodes have show the main benefits of JITER in relation to other approaches. We believe JITER can be a important contribution to solving timeliness problems for control traffic in inter-datacenter communication, or distributed control of critical infrastructures.

Acknowledgments

This work was partially supported by the EC through project FP7-607109 (SEGRID), Alban scholarship E07D401192BR, and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/ 50021/2013 (INESC-ID) and PEst-OE/EEI/UI0408/2014 (LaSIGE). We thank the anonymous reviewers

and Fabrizio Garrone for the discussions and detailed information on the Italian power grid in the context of EC project CRUTIAL.

References

[1] A. Akella, B. Maggs, S. Seshan, A. Shaikh, R. Sitaraman, A measurement-based analysis of multihoming, in: Proceedings SIGCOMM'03, 2003.
[2] Y. Amir, C. Danilov, S. Goose, D. Hedqvist, A. Terzis, An overlay architecture for high quality VoIP streams, IEEE Trans. Multim. 8 (6) (2006).
[3] Y. Amir, D. Dolev, S. Kramer, D. Malkhi, Membership algorithms for multicast communication groups, in: Proceedings 6th WDAG, 1992, pp. 292–312.
[4] D. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Morris, Resilient overlay networks, in: Proceedings SOSP'01, 2001.
[5] D. Andersen, H. Balakrishnan, M.F. Kaashoek, R. Rao, Improving web availability for clients with MONET, in: Proceedings NSDI'05, 2005.
[6] D. Andersen, A. Snoeren, H. Balakrishnan, Best-path vs. multi-path overlay routing, in: Proceedings IMC'03, 2003.
[7] J. Apostolopoulos, T. Wong, W.-t. Tan, S. Wee, On multiple description streaming with content delivery networks, in: Proceedings INFOCOM'02, 2002, pp. 1736–1745.
[8] J. Baker, C. Bond, J. Corbett, J. Furman, A. Khorlin, J. Larson, J.-M. Leon, Y. Li, A. Lloyd, V. Yushprakh, Megastore: providing scalable, highly available storage for interactive services, in: Proceedings CIDR'11, 2011.
[9] M. Blanchet, V.P. Seite, Multiple Interfaces and provisioning domains problem statement, 2011, (IETF RFC 6418).
[10] M. Cinque, C. Di Martino, C. Esposito, On data dissemination for large-scale complex critical infrastructures, Comput. Netw. 56 (4) (2012) 1215–1235.
[11] Cisco Systems, IP routing: OSPF configuration guide, cisco IOS release 15.1MT, Cisco Syst., pp. 117–122.
[12] Cisco Systems, Cisco IOS software configuration guide, release 12.2SY, Cisco Syst., pp. 6–1–6–18.
[13] W. Cui, I. Stoica., R. Katz, Backup path allocation based on a correlated link failure probability model in overlay networks, in: Proceedings ICNP'02, 2002.
[14] M. Dahlin, B. Chandra, L. Gao, A. Nayate, End-to-end WAN service availability, IEEE/ACM Trans. Netw. 11 (2) (2003).
[15] W. Dantas, A. Bessani, M. Correia, Not quickly, just in time: Improving the timeliness and reliability of control traffic in utility networks, in: Proceedings HotDep'09, 2009.
[16] M. de Prycker, Asynchronous Transfer Mode: Solution for Broadband ISDN, 3rd, Prentice Hall, 1995.
[17] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels, Dynamo: amazon's highly available key-value store, in: Proceedings SOSP'07, 2007.
[18] G. Deconinck, H. Beitollahi, G. Dondossola, F. Garrone, T. Rigole, Testbed deployment of representative control algorithms, 2008, Deliverable D9, EC Project CRUTIAL, IST-2004-27513.
[19] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, B. Weihl, Globally distributed content delivery, IEEE Internet Comput. 6 (5) (2002) 50–58.
[20] D. Dzung, M. Naedele, T.V. Hoff, M. Crevatin, Security for industrial communication systems, Proc. of the IEEE 93 (6) (2005) 1152–1177.
[21] C. Esposito, D. Cotroneo, S. Russo, Survey on reliability in publish/subscribe services, Comput. Netw. 57 (5) (2013) 1318–1343.
[22] C. Esposito, S. Russo, R. Beraldi, M. Platania, R. Baldoni, Achieving reliable and timely event dissemination over WAN, in: Distributed Computing and Networking, 2012, pp. 265–280.
[23] F. Garrone (editor), Analysis of new control applications, 2007, Deliverable D2, EC Project CRUTIAL, IST-2004-27513.
[24] H. Gjermundrod, D.E. Bakken, C.H. Hauser, A. Bose, GridStat: a flexible QoS–managed data dissemination framework for the power grid, IEEE Trans. Power Delivery 24 (1) (2009) 136.
[25] K. Gummadi, H. Madhyastha, S. Gribble, K. Levy, D. Wetherall, Improving the reliability of Internet paths with one-hop source routing, in: Proceedings OSDI'04, 2004.
[26] J. Han, D. Watson, F. Jahanian, Topology aware overlay networks, Proc. INFOCOM'05 4 (2005) 2554–2565.
[27] M. Hefeeda, A. Habib, B. Botev, D. Xu, B. Bhargava, Promise: peer-to-peer media streaming using collectcast, in: Proceedings of ACM Multimedia, 2003, pp. 45–54.
[28] P. Hunt, M. Konar, F. Junqueira, B. Reed, Zookeeper: wait-free coordination for internet-scale services, in: Proceedings ATC'10, 2010.
[29] A. Hussain, J. Heidemann, C. Papadopoulos, A framework for classifying denial of service attacks, in: Proceedings SIGCOMM'03, 2003.
[30] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, C. Diot, Analysis of link failures in an IP backbone, in: Proceedings IMW'02, 2002.
[31] V. Igure, S. Laughter, R. Williams, Security issues in SCADA networks, Comput. Security 25 (2006).
[32] S. Jain, et al., B4: Experiences with a globally-deployed software defined WAN, in: Proceedings SIGCOMM'13, 2013.
[33] D. Katz, D. Ward, Bidirectional forwarding detection (BFD), 2010, (IETF RFC 5880).
[34] Z. Li, L. Yuan, P. Mohapatra, C.-N. Chuah, On the analysis of overlay failure detection and recovery, Comput. Netw. 51 (13) (2007).
[35] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, C. Diot, Characterization of failures in an operational IP backbone network, IEEE/ACM Trans. Netw. 16 (4) (2008).
[36] D. Moore, C. Shannon, D. Brown, G. Voelker, S. Savage, Inferring internet denial-of-service activity, ACM Trans. Comput. Syst. 24 (2) (2006).
[37] N. Neves, P. Verissimo (eds.), Architecture, services and protocols for CRUTIAL, 2009, Deliverable D18, EC Project CRUTIAL, IST-2004-27513.
[38] K. Nichols, S. Blake, F. Baker, D. Black, Definition of the differentiated services field (DS Field) in the IPv4 and IPv6 headers, 1998, (IETF RFC 2474).
[39] V.N. Padmanabhan, H.J. Wang, P.A. Chou, K. Sripanidkulchai, Distributing streaming media content using cooperative networking, in: Proceedings NOSSDAV, 2002, pp. 177–186.
[40] G. Pallis, A. Vakali, Insight and perspectives for content delivery networks, Commun. ACM 49 (1) (2006) 101–106.
[41] A. Pathak, H. Pucha, Y. Zhang, Y.C. Hu, Z.M. Mao, A measurement study of internet delay asymmetry, in: Proceedings PAM'08, 2008, pp. 182–191.
[42] V. Paxson, M. Allman, Computing TCP's retransmission timer, 2000, (IETF RFC 2988).
[43] C. Pelsser, L. Cittadini, S. Vissicchio, R. Bush, On the suitability of ping to measure latency, 2013, (RIPE 66 Meeting. https://ripe66.ripe.net/presentations/128-130513.tokyo-ping.pdf).
[44] T. Peng, C. Leckie, K. Ramamohanarao, Survey of network-based defense mechanisms countering the DoS and DDoS problems, ACM Comput. Surveys 39 (1) (2007).
[45] Y. Rekhter, T. Li, A border gateway protocol 4 (BGP-4), 1995, (IETF RFC 1771).
[46] E. Romijn, RIPE NCC and Duke University BGP experiment, 2010. Https://labs.ripe.net/Members/erik/ripe-ncc-and-duke-university-bgp-experiment.
[47] V. Rosato, et al., Final report on analysis and modelling of LCCI topology, vulnerability and decentralised recovery strategies, 2007, Deliverable D2.1.2, EC Project IRRIIS.
[48] L. Segall, Internet routing glitch kicks millions offline, 2011, (http://money.cnn.com/2011/11/07/technology/juniper_internet_outage/?hpt=hp_t3).
[49] A. Snoeren, K. Conley, D. Gifford, Mesh-based content routing using XML, in: Proceedings SOSP'01, 2001.
[50] L. Subramanian, I. Stoica, H. Balakrishnan, R.H. Katz, OverQoS: an overlay based architecture for enhancing internet QoS, in: Proceedings NSDI'04, 2004.
[51] D.A. Tran, K. Hua, T. Do, Zigzag: An efficient peer-to-peer scheme for media streaming, in: Proceedings INFOCOM'03, 2003, pp. 1283–1292.
[52] B. Vamanan, J. Hasan, T.N. Vijaykumar, Deadline-aware datacenter TCP (D^2TCP), in: Proceedings SIGCOMM'12, 2012.
[53] P. Verissimo, L. Rodrigues, Distributed Systems for System Architects, Kluwer Academic Publishers, 2001.
[54] E. Viddal, S. Abelsen, D.E. Bakken, C.H. Hauser, Ratatoskr: wide-area actuator RPC over gridStat with timeliness, redundancy, and safety, Technical Report EECS-GS-011, Washington State University, 2007.
[55] D. Watson, F. Jahaniam, C. Labovitz, Experiences with monitoring OSPF on a regional service provider network, in: Proceedings ICDCS'03, 2003.
[56] C. Wilson, Terrorist capabilities for cyber-attack, in: International CIIP Handbook 2006, II, Center for Security Studies, ETH Zurich, 2006, pp. 69–88.
[57] C. Wilson, H. Ballani, T. Karagiannis, A. Rowstron, Better never than late: meeting deadlines in datacenter networks, in: Proceedings SIGCOMM'11, 2011.
[58] H. Xu, B. Li, Joint request mapping and response routing for geo-distributed cloud services, in: Proceedings INFOCOM'13, 2013, pp. 854–862.
[59] X. Zhang, A. Perrig, Correlation-resilient path selection in multi-path routing, in: Proceedings of GLOBECOM'10, 2010.

Alysson Neves Bessani is an Associate Professor of the Department of Informatics of the University of Lisboa Faculty of Sciences, Portugal, and a member of LASIGE research unit and the Navigators research team. He received his B.S. degree in Computer Science from Maringá State University, Brazil in 2001, the MSE in Electrical Engineering from Santa Catarina Federal University (UFSC), Brazil in 2002 and the PhD in Electrical Engineering from the same university in 2006. His main interests are distributed algorithms, Byzantine fault tolerance, coordination, middleware and systems architecture.

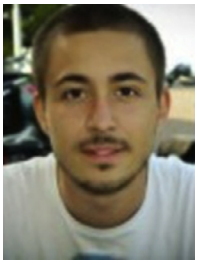
Nuno Ferreira Neves is Associate Professor with Habilitation at the Department of Computer Science, Faculty of Sciences of the University of Lisboa. Currently, he is Head of the Department. He leads the Navigators's research group and he is member of the LASIGE research unit. His main research interests are in security and dependability aspects of distributed systems. Currently, he is principal investigator of the SUPERCLOUD and SEGRID European projects. He was co-chair of the program committee of the IEEE/IFIP International Conference on Dependable Systems and Networks 2014. His work has been recognized in several occasions, for example with the IBM Scientific Prize 2004, the William C. Carter award at IEEE FTCS 1998, and the Best student paper at DISC 2009. He has more than 100 publications in book chapters, journals and conferences, and has served on the program committee of more than 70 conferences and workshops.

Paulo Esteves Veríssimo is a Professor and FNR PEARL Chair at the University of Luxembourg Faculty of Science, Technology and Communication (FSTC), since fall 2014, and head of the CritiX group (Critical and Extreme Security and Dependability) at SnT, the Interdisciplinary Centre for Security, Reliability and Trust at the same University. He is adjunct Professor of the ECE Dept., Carnegie Mellon University. Previously, he has been a Professor of the Univ. of Lisbon, member of the Board of the same university and Director of LaSIGE. Veríssimo is Fellow of the IEEE and Fellow of the ACM, and he is associate editor of the IEEE Transactions on Computers (TC - 2015—). He is currently Chair of the IFIP WG 10.4 on Dependable Computing and Fault-Tolerance and vice-Chair of the Steering Committee of the IEEE/IFIP DSN conference. He is currently interested in secure and dependable distributed architectures, middleware and algorithms for: resilience of large-scale systems and critical infrastructures, privacy and integrity of highly sensitive data, and adaptability and safety of real-time networked embedded systems. He is author of over 170 peer-refereed publications and co-author of 5 books.

Wagner Dantas is research associate at Universidade Federal de Santa Catarina in Florianópolis, Brazil. Previously he did post-graduate studies at University of Lisboa Faculty of Sciences, Portugal.

Alexandre Fonseca did his Bachelor at Instituto Superior Técnico of the Universidade de Lisboa and his MSc at Universitat Politècnica de Catalunya and KTH Royal Institute of Technology. Currently he is research associate at the Qatar Computing Research Institute.

Rui Silva did his Bachelor and MSC degrees at Instituto Superior Técnico of the Universidade de Lisboa. He is currently a PhD student a dual degree program from IST and Carnegie Mellon University, USA.



Pedro Luz did his Bachelor at Instituto Superior Técnico of the Universidade de Lisboa and a Master in Computer Science, Kerckhoffs security track, at Radboud University Nijmegen. He is currently a Security Analyst at DearBytes B.V., Holand.

Miguel Correia is an Associate Professor at Instituto Superior Técnico (IST) of the Universidade de Lisboa, in Lisboa, Portugal. He is a researcher at INESC-ID in the Distributed Systems Group. He is currently the coordinator of the Master Degree (MSc) in Information Systems and Computer Engineering. He has a PhD in Computer Science from the University of Lisboa Faculty of Sciences. He has been involved in several international and national research projects related to intrusion tolerance and security, including the SafeCloud, PCAS, TCLOUDS, ReSIST, CRUTIAL, and MAFTIA European projects. He has more than 100 publications. His main research interests are: security, intrusion tolerance, distributed systems, distributed algorithms, computer networks, cloud computing, and critical infrastructure protection.