Journée thématique du GDR RSD : Pratiques expérimentales de la communauté systèmes & réseaux

Mathieu Bacou, David Beserra, Eugen Dedu, Loïc Desgeorges, Didier Donsez, Alexandre Guitton, Baptiste Jonglez, Arnaud Legrand, Georgios Papadopoulos, Olivier Richard, et al.

Journée thématique du GDR RSD : Pratiques expérimentales de la communauté systèmes & réseaux

Mathieu Bacou David Beserra Eugen Dedu Loic Desgeorges Didier Donsez
Alexandre Guitton Baptiste Jonglez Arnaud Legrand Georgios Z. Papadopoulos
Olivier Richard Samir Si-Mohammed Nina Tamdrari Fabrice Théoleyre

31 janvier 2025

Contributeurices lors des journées

Table des matières

Liste des Recommandation

1 Introduction

1.1 Objectifs de ce document

Ce groupe de travail s'est réuni le 14 septembre 2024 à Paris pour échanger sur les pratiques expérimentales dans nos communautés. L'objectif de ce document est multiple :

- Réfléchir sur nos pratiques pour faire de la recherche expérimentale en RSD ;
- Partager avec la communauté les bonnes pratiques que nous avons identifiées ;
- Référencer les ressources disponibles (plateformes mises à part, faisant l'objet d'un atelier disjoint) ;
- Proposer des recommandations pour faire diffuser ces bonnes pratiques, et créer une *culture expérimentale*.

Attention, nous n'avons pas de volonté d'exhaustivité ni de représentativité puisque nous représentons un échantillon de la communauté. Nous avons tenté de maintenir une discussion générale, et ne ciblant pas de sous-domaine précis.

1.2 Expériences en RSD : définition du spectre

Nous utilisons dans ce document le terme dans son acceptation la plus large : il s'agit de code exécuté dans un environnement, produisant des données à des fins d'évaluation de performances.

En particulier, ce terme regroupe les usages suivants :

- Un ensemble de code et outils exécutés dans un environnement de type HPC pour en tester le passage à l'échelle ;
- Des protocoles exécutés dans une plateforme cellulaire de type 5G pour tester des débits plus élevés ;
- Du code pour découvrir la topologie d'Internet ;
- Des simulations type Monte-Carlo pour valider un modèle analytique ;
- Un simulateur de HPC et/ou de réseau pour tester in vitro un comportement particulier.

Les problématiques peuvent être dans ces cas d'une complexité très diverse. Cependant, une partie des problématiques et bonnes pratiques est commune.

1.3 Participant·es

Ont participé à l'atelier du 14 septembre et à la rédaction de ce document (ordonnés alphabétiquement) :

- Mathieu Bacou (MCF, Telecom Sud Paris) : systèmes virtualisés Faas, expériences sur Grid5000, SLICES
- David Beserra (MCF, EPITA) : évaluation de la performance des infrastructures virtualisées et des systèmes embarqués avec des benchmarks standardisés (HPL, NetPIPE, etc.) et recherche sur la gestion des ressources.
- Eugen Dedu (MCF, FEMTO-ST, Belfort-Montbéliard) : réseaux denses (nano-réseaux, 10K-20K nœuds), protocoles de routage/transport dans ces réseaux, simulation.
- Loic Desgeorges (MCF, LIP) : travaille avec Isabelle Guérin Lassous, réseau, un peu d'expérimental, mais pas tant que ça.
- Didier Donsez (PR, LIG, Grenoble) : au départ systèmes/BD, puis sys dis, et maintenant embarqué, réseaux longue distance (LoRA, satellite, …)
- Olivier Fourmaux (PR LIP6, SU) : mesure d'Internet, plateforme EdgeNet, responsable du site SU pour SLICES-FR
- Alexandre Guitton (PR, LIMOS, Clermont-Ferrand) : réseaux IoT et longue distance (couches MAC et réseau). Beaucoup de simulation et un peu d'expérimental
- Baptiste Jonglez (IR Inria, Nantes) : infrastructure Grid5000, SLICES, réseau, co-mainteneur d'EnOSlib
- Rahim Kacimi (MCF, IRIT, Univ. Toulouse 3) : Plateforme autOCampus (véhicule autonome, mobilité intelligente), utilisation de FIT-IoT Lab avant le COVID.
- Fabrice Kordon (PR Univ. Sorbonne) : réseaux de Petri pour les programmes concurrents, fiabilité du logiciel et méthodes formelles. Fait partie dans sa communauté de ceux qui promeuvent les théorèmes qui s'exécutent. Concours/benchmarks.Challenge
- Arnaud Legrand (DR LIG) : HPC, évaluation et optimisation de performance, utilisateur de Grid5000, auteur des MOOCs sur la rechercher reproductible.
- Pierre Neyron (IR LIG, Grenoble) : administrateur et développeur de l'infrastructure Grid5000, SLICES, plutôt système/HPC
- Lucas Nussbaum (responsable du Programme plateformes d'Inria) : animateur du GT plateformes, "contributeur" Grid5000, coordination nationale des SED Inria en charge des
- Olivier Richard (Mcf LIG, Grenoble) : mise en place de Grid5000, conduite d'expériences, approche fonctionnelle des environnements et des expériences (NiX)
- Georgios Z. Papadopoulos (PR, IMT Atlantique, Rennes) : réseaux maillés sans fil pour l'IoT et longue distance (couches MAC et réseau). Simulations et expérimentations.
- Benoit Parrein (MCF, LS2N, Nantes) : réseaux sans-fil pour l'IoT, focalisé sur l'IoT sous-marin, animateur du GT Plateforme
- Hugo Rimlinger (Doctorant, LIP6) : mesure internet et reproduction

— Franck Rousseau (MCF, LIG, Grenoble) : réseau sans-fil (sécurité, réseaux de capteurs), avec une composante expérimentale forte, projet WalT, maintenant maintenu
— Samir Si Mohammed (Postdoc, ICUBE, Strasbourg) : Réseau, expérimentation à la fin de la thèse, FIT-IOT lab
— Etienne Dublé (IR CNRS, LIG) : développeur principal de WalT, expérimentation sur des noeuds en environnement contrôlé ou déployés sur le terrain, réseaux à longue distance
— Nina Tamdrari (Postdoc, INRIA AGORA, Lyon) : réseau, FIT-IOT et plates-formes locales
— Fabrice Théoleyre (DR CNRS, ICUBE) : réseaux sans fil IoT, industriels sans fil avec FIT-IoT Lab
— Kevin Vermeulen (CR CNRS, LAAS) : mesures d'Internet en n'utilisant que des expérimentations, utilisation de jeux de données publics, contributeur de M-Lab (Measurement Lab) pour Reverse Traceroute
— Cassandre Vey (IR, IRIT, Toulouse) : IR sur locura4IoT et iot-lab, ultra-wide band et localisation à l'intérieur

2 Méthode expérimentale

Nous regroupons ici les discussions et réflexions quant à nos pratiques expérimentales. Cette section se focalise sur la méthodologie à mettre en œuvre pour rendre les expériences reproductibles. En effet, il existe une demande forte pour une science ouverte : transparence, pérennité des résultats, vérifiabilité.

2.1 Pratiques

La recherche expérimentale en informatique est encore relativement jeune en comparaison d'autres disciplines telles que la biologie, la chimie, etc. À la différence d'autres communautés, le développement de code est une pratique courante et nous suivons, autant que possible, les pratiques de l'état de l'art (documentation, suivi de version, test, intégration continue). Par ailleurs, notre communauté a suivi une évolution positive depuis quelques années : le code est fourni maintenant de façon fréquente, les données brutes d'expérimentations le sont quelquefois. Voici un début de référencement de nos pratiques.

2.1.1 Cahiers de laboratoire

Le cahier de laboratoire possède un objectif double : i) tracer les expériences, ii) protéger la propriété intellectuelle (PI). Le CNRS promeut notamment le *cahier de laboratoire électronique*, permettant d'offrir ces fonctionnalités sur une plateforme numérique (https://qualite-en-recherche.cnrs.fr/gt/cahier-de-laboratoire/). La solution logicielle repose sur le projet opensource `eLabFTW` (https://www.elabftw.net/) conçu à l'origine pour répondre à des besoins en biologie cellulaire.

L'usage d'un cahier de laboratoire est très largement répandu dans de nombreuses communautés en sciences expérimentales (biologie, chimie, etc.). Il est étonnant que très peu de chercheur·euse·s de nos communautés se soient emparés de cet outil car il présente de nombreux atouts :

— Tracer la genèse des idées (qui et quoi) et protéger intellectuellement les contributions ;
— Tracer les différentes versions d'expériences, leurs paramètres, etc.
— Garder une trace à long-terme lorsqu'un personnel temporaire quitte la structure.

Un cahier de laboratoire peut éventuellement être remplacé pour la traçabilité des expériences par un *notebook* (outil informatique permettant de rédiger du texte à l'aide d'un langage de balisage léger, d'exécuter simplement des codes simples, et d'intégrer les résultats produits) géré à l'aide d'un système de *versioning* (cf. section 5.2). Par contre, les aspects de PI ne pourront pas être protégés dans ce cas.

2.1.2 Automatisation et orchestration

Tant que faire se peut, une expérience "interactive" au cours de laquelle le·la chercheur·euse modifie à la volée les paramètres ou les versions des packages installés devrait être évitée. Ce procédé est bien souvent non reproductible.

Ainsi, une expérience devrait idéalement suivre une suite automatisable d'actions à réaliser :

1. Réservation des ressources sur une plateforme d'expérimentation
2. Configuration des équipements réservés (firmware, paramètres, etc.)
3. Lancement des expériences
4. Collecte des données

Des outils d'*orchestration* peuvent aider l'expérimentateur à gérer les tâches répétitives liées à ce processus. Cela peut être des outils standard (Ansible, outils de *workflow*) ou des outils plus spécialisés pour un domaine de recherche scientifique (Execo [4], EnOSlib [20], E2Clab [16]).

Recommandation 1 (Mise en commun d'outils d'orchestration d'expériences) : Au sein d'une communauté scientifique, il est souhaitable de mutualiser des outils d'orchestration adaptés au domaine visé afin de faciliter l'adoption

des bonnes pratiques expérimentales. Ces outils devraient être développés en étroite coordination avec des plateformes expérimentales majeures comme SLICES-FR.

2.1.3 Versionnage de code

La communauté d'informaticiens utilise maintenant de façon intensive les solutions de versioning qui permettent une traçabilité du code, et une automatisation des tests. Il n'est donc pas surprenant que cet usage soit également très largement répandu lorsqu'il est question d'expérimentation.

Recommandation 2 (Versionnage de code avec des expériences embarquées) : Il est vital de versionner le code utilisé pour produire l'expérience, ce qui comprend :

1. Le code source qui sera exécuté par les nœuds embarqués ;
2. Le code source qui sera exécuté pour collecter les données ;
3. Les scripts ayant permis de produire le firmware à l'aide du code source. Certains paramètres sont passés à la compilation du firmware : il est donc très important de pouvoir reproduire le code binaire, avec les bonnes valeurs des paramètres.

De façon générale, les scripts utilisés devraient également sauvegarder le numéro de version des outils utilisés pour produire le code (ex : compilateur) ainsi que de l'ensemble des dépendances logicielles (bibliothèques). En effet, un compilateur va produire un code de taille et de comportement potentiellement différents selon sa version et les options utilisées.

La version binaire d'un firmware pourrait ne pas être stockée, le code pouvant être régénéré, mais il est essentiel de s'assurer au préalable que cette re-génération est bien maîtrisée et reproductible. Attention, si le firmware est protégé intellectuellement, le versionnage de code pourrait être gardé privé, et les binaires seraient alors fournis sur demande pour reproduire les expériences. Cependant, ce cas d'usage ne devrait pas être la règle car il rend la reproductibilité particulièrement ardue.

La vérification des résultats est la première étape vers la reproductibilité. Cependant, de nombreux auteur·ice·s oublient de fournir les outils leur ayant permis d'analyser les résultats expérimentaux. Ils sont pourtant la clé pour comprendre la démarche des auteur·ice·s, et d'éventuellement la questionner.

Recommandation 3 (Analyse des données) : Les auteur·ice·s devraient fournir les outils ayant permis d'analyser les données, générer les graphiques, manipulant en entrée directement les données brutes. Ces outils devraient également être versionnés, au même titre que le code d'expérimentation. L'utilisation d'un notebook pour documenter le processus d'analyse est également recommandé.

2.1.4 Que sauvegarder dans les résultats d'expérience ?

Recommandation 4 (Données produites) : Il faudrait **tout** sauvegarder dans la production d'une expérience pour autoriser toute évaluation post-mortem. En particulier, l'état des connaissances peut évoluer dans le futur (ex : biais identifié, phénomène physique), et demander de ré-analyser des résultats expérimentaux. Ainsi, les données comprennent :

- les données brutes générées ;
- les paramètres utilisés pour une expérience (de la version du compilateur aux paramètres de l'expérience)
- l'environnement matériel et logiciel de l'expérience (modèle CPU, version du système d'exploitation, voire jusqu'à la température mesurée par le matériel…)
- le code ayant permis de traiter les données brutes et de générer les graphiques (cf. recommandation 3)

Un code exécuté pour une expérience pourrait de façon systématique dumper ses paramètres à des fins de sauvegarde. Ce dump serait alors stocké avec les données produites par l'expérience. On pourrait alors analyser les résultats a posteriori ainsi que reproduire l'expérience.

2.1.5 Intégration continue

Les tests de non-régression sont souvent utiles en sciences expérimentales, afin de vérifier qu'une modification du prototype expérimental continue à se comporter de la façon attendue. L'intégration continue est utilisée de façon intensive par de nombreux projets opensource afin de vérifier que les modifications apportées par la communauté se

comportent comme prévu (compilation correcte, simulation ou émulation de code, absence de régression en fonctionnalités ou en performance, etc)

Par contre, des tests d'intégration continue sur plateformes réelles sont moins fréquents. Il faudrait que chaque projet puisse réserver des ressources matérielles et exécuter automatiquement des expériences avec le nouveau code. De façon plus ambitieuse, il faut pouvoir interpréter si les performance sont celles attendues (ex : latence d'une communication, délai de traitement d'un micro-services). Sachant que les métriques de performances sont très sensibles à l'environnement (particulièrement dans les plateformes mutualisées), vérifier la non-régression peut poser problème.

Recommandation 5 (Intégration continue) : Les nouvelles plateformes à large-échelle comme SLICES-FR pourraient prévoir dans le futur des facilités pour l'intégration continue basée sur des expériences.

2.2 Pistes d'actions envisagées

Nos usages sont très divers : un ballon stratosphérique pour la communication, un code HPC, un réseau cellulaire, de petits objets communicants, des infrastructures de stockage distribuées, etc. Cependant, la méthodologie est souvent très similaire, et il est nécessaire de créer une culture de l'expérimental dans nos disciplines. De trop nombreux articles, même publiés par des laboratoires français, ne permettent pas de reproduire un résultat par manque d'information, dont le code utilisé. Pire même, des expériences peuvent ne pas être reproductibles (ou même réanalysables) au sein d'une même équipe après le départ d'un·e doctorant·e, qui part avec son expertise. Cette problématique était pourtant au cœur des préoccupations des pionniers de la recherche reproductible, en Géo-Sciences il y a plus de 30 ans [1].

2.2.1 Ecole du GDR

L'école d'hiver du GDR RSD (https://sites.google.com/site/rsdwinterschool) cible déjà les jeunes chercheur·euse·s. Le programme englobe déjà une partie des problématiques associées à la recherche expérimentale.

Recommandation 6 (Ecole du GDR) : Des modules pourraient être régulièrement prévus à destination des doctorant·es et postdoctorant·es de la communauté pour développer une culture expérimentale. L'école peut également être l'occasion de donner les pointeurs vers les ressources en ligne déjà disponibles afin que les jeunes chercheur.euse·s puissent compléter leur formation après l'école.

2.2.2 Ressources pédagogique pour les jeunes chercheur·euse·s

Recommandation 7 (Formation en ligne des étudiant·es) : Les MOOCs sur la recherche reproductible [15, 23] offrent chacun 24 h de formation reconnus par les écoles doctorales. Nous ne pouvons qu'encourager tous les chercheur·euse·s souhaitant acquérir les techniques de base à suivre cette formation.

2.2.3 Conférences nationales soutenues par le GDR

La reproductibilité des résultats scientifiques est un enjeu fondamental pour la crédibilité et la pérennité de la recherche. Promouvoir et mettre en lumière cette culture au sein des conférences nationales comme COMPAS (Conférence francophone en Parallélisme, Architecture et Système) et Algotel (Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications) ou CoRes (Rencontres Francophones sur la Conception de Protocoles, l'Evaluation de Performance et l'Expérimentation des Réseaux de Communication) semble donc pertinent. Ces conférences offrent une opportunité idéale pour encourager les bonnes pratiques de reproductibilité au sein de la communauté scientifique nationale, permettant ainsi de renforcer la transparence des recherches.

Cependant, il est important de reconnaître que COMPAS et Algotel/CoRes servent avant tout de plateforme d'échanges scientifiques. Certains articles sont des résumés courts de publications internationales (aval), d'autres y sont plutôt présentés en tant que résultats préliminaires (amont). L'objectif n'est donc pas de fournir un cadre rigide qui serait inadapté pour certains articles.

Auteur·ice·s : Les conférences nationales jouent par ailleurs un lieu de sensibilisation quant à la méthodologie pour les jeunes chercheur·euse·s (doctorant·es et postdoctorant·es). Ainsi, systématiser une évocation de la partie reproductibilité des résultats paraît opportun pour renforcer nos pratiques.

Recommandation 8 (Conférences nationales (conseils aux auteur·ice·s)) : Les auteur·ice·s devraient être encouragé·es dans les instructions de soumission à décrire les aspects de reproductibilité de leurs articles soumis. Cette description pourrait être placée en annexe, et donc en dehors de la limite habituelle du nombre de pages. Un canevas pourrait être fourni à titre d'exemple afin de guider les auteur·ice·s :

- Disponibilité des données : les données brutes d'entrée et de sortie sont publiés dans une base de données publique ;
- Accessibilité du code source : Le code source, les scripts d'automatisation et les instructions sont fournis sur un dépôt public
- Formats des résultats : les données sont-elles fournies de façon structurée, en format standard (ex : JSON), etc. ?
- Notebooks interactifs : un notebook est fourni permettant de rejouer éventuellement les scénarios, de filtrer les données, d'analyser les résultats, etc.

Ce canevas devrait être sans doute adapté au domaine de la conférence. Les auteur·ice·s peuvent s'aider de diverses ressources, comme le livret "Vers une recherche reproductible" [13], le guide FAIR [6], ou les consignes de description d'artefacts [a].

a. Voir par exemple https://github.com/ctuning/artifact-evaluation/blob/master/docs/checklist.md

Relecteur·ices : Il est essentiel que les évaluateur·ices adoptent une approche bienveillante et constructive dans leurs retours, en gardant à l'esprit que l'objectif est de former les jeunes chercheur·euse·s à des pratiques rigoureuses sans pour autant imposer des contraintes trop strictes. En encourageant des formats de résultats standardisés, des analyses de sensibilité et l'utilisation de notebooks interactifs, ils contribuent à établir une culture de reproductibilité, créant un cercle vertueux.

Recommandation 9 (Conférences nationales (conseils aux relecteur·ices)) : Les évaluateur·ices sont encouragé·es à s'assurer que les jeux de données et le code source nécessaires à la reproduction des expériences sont accessibles sur des plateformes publiques reconnues. Une attention particulière devrait être portée à la clarté de la description des environnements expérimentaux, ainsi qu'à l'automatisation des expériences pour leur reproductibilité.

Organisateur·ices / Chairs : Les chairs devraient créer une dynamique positive dans la conférence qu'ielles dirigent, en encourageant par exemple le TPC à considérer les aspects de reproductibilité, de méthodologie, et de science ouverte. Ielles sont bien souvent prescripteur·ices, et les bonnes pratiques mises en place perdurent souvent lors des éditions suivantes. Un manuel des bonnes pratiques à destination des futurs chairs est notamment bien souvent pertinent.

Recommandation 10 (Conférences nationales (comité d'évaluation de reproductibilité)) : Un comité ad-hoc pourrait être monté, en s'appuyant notamment sur la communauté des jeunes chercheur·euse·s, pour évaluer la reproductibilité des expériences. Le comité pourrait suivre les badges de l'ACM (https://www.acm.org/publications/policies/artifact-review-and-badging-current) évaluant plusieurs niveaux de reproductibilité :

- Artefacts évalués - Fonctionnel
- Artefacts évalués - Réutilisable
- Artefacts disponibles
- Résultats validés
- Résultats reproduits
- Résultats répliqués

La participation à ce comité pourrait faire partie de la formation des doctorant·es du domaine et être valorisé lors des recrutements.

3 *Benchmarking* et compétitions

Tout d'abord il faut distinguer les benchmarks des compétitions : les benchmarks donnent un score pour une méthode, indépendamment des autres méthodes, alors que les compétitions servent à désigner un *vainqueur*.

3.1 Est-ce utile d'avoir des benchmarks ?

La loi de Goodhart (https://fr.wikipedia.org/wiki/Loi_de_Goodhart) stipule que "lorsqu'une mesure devient un objectif, elle cesse d'être une bonne mesure". Cependant, des benchmarks sont aussi bénéfiques.

Il peut être tentant de se comparer à la méthode réputée la plus inefficiente, en étant ainsi certain de pouvoir présenter un gain en performances. Inversement, certaines solutions de la littérature peuvent être très complexes à reproduire,

d'autant plus quand le code n'est pas disponible (cf. section 2.1.3). De plus, il est fréquent de considérer des scénarios qui avantagent la solution à présenter, quelquefois pour des raisons valides. Cependant, l'absence de cadre global rend au final les comparaisons ardues. Des benchmarks pourraient fournir ce cadre.

Avoir des benchmarks serait utile aussi comme indicateur de l'avancement (du progrès) d'un domaine de recherche, un peu comme https://fr.wikipedia.org/wiki/Thorme_des_quatre_couleurs#Histoire. Cela structurerait la communauté, la rendant focalisée sur des progrès clairs. Par exemple, « la RoboCup reste principalement un moyen de faire avancer les recherches dans le domaine de la robotique » (https://fr.wikipedia.org/wiki/RoboCup).

Il faut une volonté de la communauté pour aller vers des cas d'usage structurants, et reconnus comme suffisamment génériques. Derrière un challenge, nous pouvons avoir de multiples objectifs : mise au même niveau de toute une communauté, créer une base de comparaison, animer une communauté recherche, trouver les points difficiles, former les étudiant·es. Il faut que cet objectif soit clair dès la création.

3.2 Quelles sont les initiatives dans d'autres communautés ?

Certaines communautés utilisent des benchmarks très visibles :
- pour les CPUs (https://www.notebookcheck.net/Mobile-Processors-Benchmark-List.2436.0.html) ;
- pour les supercalculateurs (https://top500.org) ;
- pour le machine learning (de très nombreux benchmarks son listés sur https://paperswithcode.com/about [1]) ;

Nous avons aussi identifié des compétitions :
- RoboCup (équipe de football robotique, https://fr.wikipedia.org/wiki/RoboCup).
- Le Model Checking Contest (https://mcc.lip6.fr/2024/) a été créé en 2011 et en est donc à sa 14ième édition. Son utilisation est courante dans la communauté. Le framework a été automatisé depuis 2014, et a peu évolué depuis. Le focus y est donné plus sur les résultats obtenus que sur les performances (qui ne sont donc qu'indicatives). Chacun des développeurs reçoit une machine virtuelle dans laquelle il peut installer les outils qu'il souhaite. Le confinement est assuré par des scripts de pilotage, et le log des utilisations mémoire et autre (sysstat) sont retournés à l'utilisateur. En particulier, le système garde un log de tout (stdout/stderr) pour analyse post-mortem (les traces peuvent être très importantes). Le Benchkit correspond à environ 1500 lignes de bash.

  Quel bilan tirer de cette compétition ? La fiabilité a augmenté avec le temps (de 80% au début à 99,6% maintenant) et a été structurant pour la communauté. Le défi accueille 10–12 compétiteurs chaque année, c'est à peu près la même chose que pour les SAT-solveurs. Cependant, le benchmark fournit aussi des cas d'étude génériques, utilisables en dehors de la compétition proprement dite.

3.3 Est-ce possible dans les nôtres ? Qu'est-ce qui nous manque ?

Derrière tous ces défis, les communautés (recherche opérationnelle, machine learning, SAT, …) ont un formalisme théorique qui permet de bien définir le cadre d'étude. Cependant, est-ce bien le cas pour la communauté RSD ?

Notre communauté n'utilise pas (habituellement ?) des benchmarks. Nous avons tenté de référencer les freins à un benchmark selon l'objet d'étude :

HPC : il peut être pertinent de définir des applications et caractéristiques clé ;

Cloud : il serait nécessaire de définir du matériel particulier, une architecture, ce qui restreindrait ainsi les cas d'étude ;

Réseaux de cœur : il n'existe pas de plateformes de test, et les modèles connus sont très imparfaits (ce qui justifie les études expérimentales). Est-il donc possible dans ces conditions de définir des hypothèses partagées ?

Protocoles Internet : nous connaissons une certaine ossification de l'Internet. Déployer à la large-échelle de nouveaux protocoles reste un défi souvent inatteignable : il faudrait trouver des volontaires (particuliers ou acteurs industriels majeurs). Un benchmarking dans ce cadre semble complexe ;

Internet des Objets : nous sommes tributaires des technologies utilisées (ex : les *wake up* radio ne fonctionnent pas comme le Wi-Fi ou même comme la 5G).

Nous avons notamment les contraintes communes suivantes :

Longue durée : les expériences longues (ex : une semaine) peuvent être pertinentes. Cependant, un défi qui prendrait ces échelles de temps se heurterait à des problèmes pour isoler des expériences concurrentes sur la même plateforme.

Dépendance technologique : beaucoup de nos domaines d'application sont dépendants de la technologie, et de l'architecture matérielle. Dans un tel cadre, redévelopper du code spécifique à un nouveau matériel est coûteux.

L'évaluation de performances considère un nombre croisant de critères (consommation d'énergie, efficacité, débit, latence, passage à l'échelle, etc.), faisant qu'une solution est quelquefois performante pour certains des critères seulement. Cependant, ce multi-critères se prêterait bien à un benchmark, mesurant des métriques différentes.

1. "the mission of Papers with Code is to create a free and open resource with Machine Learning papers, code, datasets, methods and evaluation tables"

3.4 L'existant

Nous avons pu identfier un petit nombre de d'initiatives dans les thématiques du GDR :

IETF : nous avons des ateliers d'interopérabilité de protocole (cf. IETF) ou des hackatons (https://www.ietf.org/meeting/hackathons). D'autres domaines scientifiques ont déjà bien adopté ces approches : la bioinformatique réalise par exemple des ReproHackatons [21].

CPS IoT Bench : la Dependability Competition a été organisée de nombreuses fois avec la conférence EWSN. Elle a ensuite évolué en D-CUBE, puis en IoTBench. Cependant, le scénario est restreint aux déploiements de réseaux simple ou multisaut avec de fortes interférences.

Grid Plugtest : le défi GRID PLUGTEST (2005) considérait les problèmes d'interopérabilité dans la grille. Sur 3 jours, il a regroupé jusqu'à 80 participants de 10 pays différents. La communauté grille s'est réorientée sur le cloud.

Repro : des chercheur·euse·s ont discuté de l'intérêt d'un évènement de Benchmarking en réseau [7, 9] éventuellement coorganisé par ACM SIGCOMM. Bien que ces initiatives datent de 2017, force est de constater que la dynamique n'a pas mené à un évènement visible et pérenne. Il y a aussi "Benchmarking computers and computer networks" 2010 (http://www.crew-project.eu/sites/default/files/Benchmarking%20computers%20and%20computer%20networks.pdf).

3.4.1 La simulation comme outil de benchmark ?

Les simulateurs que nous utilisons dans la communauté se prêteraient mieux à la création de benchmarks. Le code de simulation peut par ailleurs être adapté au cas d'usage relativement aisément (plus que quand le code est dépendant du matériel). Lorsque la communauté s'est concentrée sur un simulateur, il est donc plus facile d'organiser un benchmark. Cependant, un simulateur reste un modèle (imprécis) du monde réel [2, 3].

Dans de nombreux cas, un simulateur phare a émergé, mais il persiste des applications ayant leurs propres simulateurs développés de façon ad hoc. Si de nombreux simulateurs coexistent, la communauté ne sera vraisemblablement pas prête à aller vers une mutualisation et des hypothèses communes. En effet, les résultats sont souvent très dépendants du simulateur (modèles différents), rendant toute comparaison inéquitable.

Recommandation 11 (Création de défis de benchmarking) : Le GDR RSD devrait poursuivre sa réflexion quant à des évènements normatifs de benchmarking, et sur la pertinence (ou son absence) qu'il pourrait avoir pour structurer la communauté.

4 Jeux de données publics et publication de jeux de données

4.1 Pratiques actuelles

Pour que les résultats de nos recherches soient reproductibles, il faut publier le code mais aussi les données. Ces données pourraient ensuite être utilisées plus simplement pour établir des benchmarks. Les relecteur·ices pourraient être intéressé·es par l'accès aux données, au moment de la relecture du papier. Le concept d'*open data*, de plus en plus répandu, ne semble pas complètement diffusé dans notre communauté.

Actuellement, la publication de jeux de données ne semble pas très répandue dans la communauté. Cela peut venir du fait que les expérimentations sont considérées comme très spécifiques, et potentiellement peu utilisables dans d'autres contextes que le contexte initial. Mais il existe néanmoins des bonnes pratiques en la matière :

- des concours/challenges qui se concentrent sur des jeux de données ;
- des organisateur·ices de conférences ou des éditeur·ices de journaux qui mettent en place des plateformes pour déposer les jeux de données (ex : le système d'artefact de l'ACM, Mendeley Data chez Elsevier) ;
- des chercheur·euse·s qui diffusent certains jeux de données.

4.2 Que doivent contenir les données ?

Des données sans métadonnées sont très difficilement exploitables [19]. Malheureusement, il n'existe pas de format de données standard et chacun s'efforce d'avoir la démarche la plus générique possible. Les activités de type benchmarks/contests (section 3) pourraient permettre de faire émerger des standards sur des cas d'usage précis. Il est possible qu'à mesure que les chercheur·euse·s publient des jeux de données, de plus en plus d'autres chercheur·euse·s les utilisent, ce qui alimenterait ensuite un cercle vertueux et la production de jeux de données pouvant être utilisés comme benchmarks.

Les jeux de données publiés doivent être complets. Idéalement, il nous semble qu'ils devraient contenir

Scénario considéré : application, cas d'usage réel ou modèles, etc.

Inventaire des ressources matérielles utilisées : serveur/client, CPU, mémoire, fabricant, etc. L'inventaire devrait pouvoir être organisé hiérarchiquement (ex : un réseau connectant des serveurs possédant des GPU/CPUs, etc.)

Paramètres de toutes les briques logicielles utilisées ;

Données non filtrées : traces d'exécutions, paquets reçus (même en doublon) avec des caractéristiques des mesures (force du signal, fréquence).

Post-traitement : les données capturées ont-elles subi un post-traitement (ex : k-anonymisation) ;

Déroulement de l'expérience : il peut être pertinent de taguer des évènements d'intérêt pour expliquer les données (ex : crash d'un serveur, vol d'un noeud IoT déployé dans la nature, maintenance d'un serveur ou changement de version, etc.)

Métadonnées : il est important de référencer comment les mesures ont été réalisées (ex : version d'un logiciel de mesure, granularité, fréquence de mesure, unités, etc.)

Il peut être intéressant de fournir des scripts de manipulation des données, en plus des données elles-mêmes.

Recommandation 12 (Métadonnées des jeux de données) : Les métadonnées doivent être publiées sous format structuré (ex : json) avec les données et doivent décrire l'expérience, le matériel utilisé, la configuration, etc. Idéalement, des expériences similaires devraient utiliser les mêmes métadonnées afin de pouvoir automatiser leur interprétation. Plus les métadonnées sont riches, plus le jeu de données pourra être exploité pour un cas d'usage non identifié initialement.

Définir un cadre standard fait partie des perspectives du Groupe de Travail.

4.3 Droits

Les données peuvent créer des problèmes spécifiques en termes de droit. Sans être exhaustifs :
- Mise à disposition par des partenaires industriels, ne souhaitant pas divulguer les données ;
- Problèmes de vie privée pour les données collectées relevant de l'humain ;
- Contraintes pour les données de santé ;

Lorsque les données elles-mêmes ne peuvent pas être publiées, il est au moins nécessaire de publier leur distribution statistique, leurs caractéristiques, leurs limites, etc. Idéalement, un modèle pourrait être créé à partir des données, et le modèle pourrait être publié afin de rejouer les données pour une autre expérience et servir de base de comparaison.

Recommandation 13 (Publication des jeux de données) : Dans le cadre de l'Open Science, les jeux de données devraient être de façon générale publiés. Quand c'est impossible, les auteur·ice·s doivent pouvoir fournir suffisamment d'informations sur les données pour que la communauté puisse interpréter et idéalement rejouer les scénarios (distributions statistiques, modèles synthétiques, etc.)

La licence CeCILL (http://www.cecill.info/) est promue par Inria, le CNRS, et le CEA. Elle correspond à une licence libre, adaptée au logiciel, transposée au droit français, . Nous suggérons de lire le guide du CNRS [11] s'appliquant au code généré, mais également aux données, de se renseigner sur les différents types de licences libres [5], et d'utiliser des outils comme https://choosealicense.com/ ou https://creativecommons.org/choose/. En effet, dans de nombreux pays, des données sans licence (même mises à disposition librement sur le web) sont considérées comme par défaut interdites d'utilisation. Ne pas prendre le temps de renseigner de licence revient donc à s'assurer que personne ne pourra rien tirer des données mises à disposition.

4.4 Valorisation

Il existe une question centrale : comment donner une visibilité aux données générées ? Le concept d'open data est central normalement, afin de permettre à la communauté de se réapproprier les données, et de les réutiliser (pour un cas d'usage éventuellement non identifié par les auteur·ice·s).

L'investissement initial pour la diffusion de données semble coûteux : il peut y avoir des problèmes de vie privée nécessitant de l'anonymisation (parfois complexe) et la construction de datasets de benchmark nécessite de bien identifier les besoins d'une communauté ayant une masse critique et travaillant sur les mêmes données. Cependant, la génération de jeux de données est maintenant relativement bien valorisé (cf. section 7).

ACM a mis en place un système de badges pour les artefacts [24]. Participer au processus permet de valider un certain degré d'exigence et de le valoriser dans sa liste de publications.

Pour les jeux de données privées, il s'agit souvent d'une collaboration avec un industriel (traces des opérateurs mobiles, mobilité d'utilisateurs, utilisation d'infrastructures Cloud, etc.) Dans un tel cas, le problème de valorisation n'est pas la clé puisque le jeu de données est souvent généré par les industriels, et les académiques ne font que les *utiliser*.

Table 1 – Choix de la plateforme à utiliser pour stocker et publier des données expérimentales

Plateforme	Pérennité	Fiabilité	Taille	Open Science	Versions	Institutionnel
Zenodo	✓	✓	✓	✓	✓	✓
Figshare	✓	✓	✓	✗	✗	✗
Page/Site personnel	✗	✗	✓	✗	Manuel	✗
Ressources laboratoires	✗	✓	✓	✓	Dépend	✓
BigQuery	✓	✓	✓	✗	✗	✗
recherche.data.gouv.fr	✓	✓	✓	✓	✓	✓
Journaux	✓	✓	✗	✗	✗	✗
HAL	✓	✓	✓	✓	✗	✓

Recommandation 14 (Faire reconnaître l'activité de diffusion des données) : Utiliser les conférences du GDR (Algotel, CoRES, COMPAS) pour identifier des tendances, des pratiques, et idéalement faire émerger des datasets.

Recommandation 15 (Affichage au niveau du GDR) : Profiter de la gazette du GDR RSD pour communiquer sur la création d'un jeu de données. De même, une rubrique pourrait être créée sur le site web afin de référencer les données produites par la communauté du GDR. Il faudra dans ce cadre définir un canevas commun de description des jeux de données.

4.5 Lieux de publication

Il existe de nombreuses plateformes pour publier des jeux de données :

Zenodo est une des plateformes les plus répandues, maintenue par le CERN. Il existe un engagement de stockage sur le long-terme : "your research is stored safely for the future in CERN's Data Centre for as long as CERN exists" ;

figshare : peut partager jusqu'à 20GB de données pour des chercheur·euse·s individuels.

page/site personel : il peut-être tentant d'héberger les données sur un site web installé de façon ad hoc. Une machine virtuelle peut également mettre à disposition une visualisation des données. Cependant, il est difficile de garantir sur le long-terme l'accès ;

Ressources laboratoires : de nombreux laboratoires mettent à disposition des ressources. Attention, les forge ne sont pas un outil pérenne, comme l'ont montré plusieurs cas (forge d'Inria, d'IMAG, etc.). Par ailleurs, les fusions de laboratoire sont fréquentes sur de longues échelles de temps.

BigQuery est maintenu par Google

recherche.data.gouv.fr l'Inrae maintient l'infrastructure. La plateforme permet de donner une visibilité à des grands jeux de données ;

Certains journaux (scicompanion, elsevier, IEEE, …) fournissent directement l'infrastructure pour héberger les données (souvent de façon payante).

HAL peut également être utilisé pour déposer les données (https://hal.univ-lyon2.fr/page/deposer-des-jeux-de-donnees), mais à faible taille. L'avantage est de pouvoir lier données et publication scientifique. HAL propose également un lien direct avec Software Heritage.

Il existe des référencements plus exhaustifs de plateformes :
- France : https://cat.opidor.fr/index.php/Entrep%C3%B4t_de_donn%C3%A9es
- International : https://www.re3data.org/

Recommandation 16 (Plateforme de dépôt des données) : Il est fortement recommandé de rendre les données publiques sur des plateformes institutionnelles pérennes (par exemple Zenodo, HAL, ou recherche.data.gouv).

4.6 Relecture d'articles

En tant que relecteur·ice, nous avons une responsabilité dans la diffusion des bonnes pratiques. Bien souvent, une demande de partage de code et de données est sufisante pour inciter les auteur·ice·s à le faire. De manière générale, nous

devrions considérer en tant que relecteur·ice qu'un tel partage correspond à un pré-requis à la publication. Notre rôle en tant que relecteur·ice étant bénévole, il ne faut pas négliger notre capacité de prescription des bonnes pratiques.

Recommandation 17 (Relecteur·ices et attentes en reproductibilité) : En tant que relecteur·ices, nous devrions **toujours** demander l'accès aux données et au code lorsque nous relisons des articles, quel que soit la conférence ou le journal.

5 Outils et langages pour le traitement des données expérimentales

5.1 Langages de traitement

La communauté a à sa disposition une grande variété d'outils possibles :
- Python : Pandas, Matplotlib, Seaborn, notebooks.
- R : ggplot2, tidyverse.
- Julia : DataFrames.jl, Makie.jl (https://juliadatascience.io/)
- source de données (stockage) : MongoDB, ClickHouse, Druid/InfluxDB pour les séries temporelles.

Par définition, les informaticiens sont habitués aux langages de programmation et à la manipulation de données. Les nouveaux entrants n'ont donc bien souvent pas de frein quant à l'acquisition de ces outils pour faire de la science expérimentale dans nos disciplines.

Tandis que les outils sont standards et largement utilisés, la méthodologie dépend fortement de l'expérience conduite. Ils sont donc adaptés au cas d'étude, et il est difficile de donner ici des conseils génériques.

Recommandation 18 (Formation aux outils d'analyse de données) : Une formation à destination des étudiant·es de master, mais également des doctorant·es en début de thèse pourrait servir d'initiation aux outils d'analyse de données et à leurs bonnes pratiques, dans le cadre d'une école pour jeunes chercheur·euse·s.ses, ou sous forme de formations en ligne [a] (complétant le MOOC existant) pour les autres.

a. Voir par exemple les cours de Software Carpentry https://swcarpentry.github.io/r-novice-gapminder/ et https://swcarpentry.github.io/python-novice-gapminder/

5.2 Notebooks

Les notebooks sont maintenant assez largement répandus pour les expériences en informatique. Ils permettent de regrouper en un seul endroit le lancement d'une expérience, sa paramétrisation, les justifications, les explications du cadre méthodologique choisi, etc.

Ils permettent de structurer une grille de lecture des données autour des résultats, et représentent donc un atout pour la reproductibilité. En explicitant pas à pas les traitements, le nettoyage des données, le filtrage, l'auteur·ice permet à la communauté de pouvoir éventuellement les réanalyser. L'auteur·ice commente ses notebooks pour justifier son approche, et son interprétation. Les notebooks permettent par ailleurs de réexploiter un environnement classique de programmation (ex : Jupyter supporte plus de 40 langages de programmation différents). Les notebooks permettent une approche itérative de l'analyse : c'est en explorant les données que le chercheur identifie des problèmes, les résout, etc.

Il peut être complexe de bien structurer le notebook pour éviter la réexécution depuis le début, et donc des temps d'exécution inutilement longs. Il est recommandé d'essayer de créer des étapes de pré-traitement pour arriver à présenter des étapes bien distinctes et plus compréhensibles. Malheureusement, certains notebooks donnent lieu à des commits git peu lisibles, ce qui nuit à la compréhension des évolutions du code.

Recommandation 19 (Notebook) : Un notebook peut remplacer un cahier de laboratoire concernant la traçabilité expérimentale (sauf pour les aspects PI), en adoptant une approche de type intégration continue. Cependant, il faut qu'il soit documenté, explicité, justifié afin de pouvoir analyser (et potentiellement) reconsidérer les choix.

5.3 Gestion commune du code de l'expérience et des données

Il est nécessaire de lier les données produites à la version de l'expérience qui les a produites. Les données pourraient être directement intégrées au dépôt git, mais la taille des données produites rend cette option couteuse et peu recommandée.

La communauté utilise principalement l'une des deux options suivantes :

1. git-annex pour ne conserve dans le dépôt git que les localisations des fichiers

- les fichiers ne sont pas stockés dans le dépôt, il faut donc se reposer sur une solution tierce au choix pour le stockage ;
- intérêt : il est possible de nettoyer le dépôt et de se débarrasser des données qui n'auraient plus d'intérêt.

2. git-lfs permet de conserver les données sur un serveur distant (déployé par GitHub ou GitLab) et de ne garder dans le dépôt que des pointeurs. Il est possible de ne récupérer que les fichiers utiles pour la version considérée. La configuration est manuelle à base de règles sur les noms (extensions) de fichiers. Cependant, les données restent dans le dépôt, augmentant l'espace disque distant et local. Il est par ailleurs impossible de purger des données réellement inutiles (par exemple des expériences ratées)

Recommandation 20 (Git et données) : Les personnes intéressées pourront suivre le MOOC Recherche Reproductible 2 [23] qui aborde cette problématique en détails.

5.4 Dépendance aux autres sujets

Les sujets abordés dans ce document sont fortement interdépendants. Notamment :
- Section 2 (Méthode expérimentale) : les outils d'implémentation de la méthode expérimentale peuvent influencer la forme des données, et donc les outils d'analyse disponibles ;
- Section 4 (Jeux de données publics et publication de jeux de données) : les méthodes de stockage et de publication des jeux de données peuvent influencer les outils d'analyse disponibles ;

6 Reproductibilité de l'état de l'art

La reproductibilité des résultats de l'état de l'art est souvent très fastidieuse, mais reste primordiale afin de s'assurer de l'apport des nouvelles méthodes proposées. Il a souvent été constaté que les chercheur·euse·s peinaient à reproduire les résultats dans les papiers de recherche, et ce pour diverses raisons : Absence de description détaillée des conditions d'expérimentation/simulation, absence des données utilisées, etc.

Les objectifs de la reproductibilité de l'état de l'art sont divers : i) dissiper un doute sur les résultats d'un papier, ii) montrer un *test of time* pour évaluer la robustesse d'une approche ou iii) comparer les résultats d'une approche avec une autre.

De nombreux articles se sont intéressés aux problèmes posés par la reproductibilité et donnent des conseils (càd. choses à faire ou ne pas faire) [12]. Il s'agit d'une bonne entrée en matière pour tout·e chercheur·euse s'intéressant au sujet.

Nous recommandons de suivre évidemment les bonnes pratiques en termes de partage de code et de données (sections 4). Il faut fournir le code source utilisé, les données brutes (en format structuré), la documentation, le pipeline de post-traitement, etc. Cependant, quelques aspects sont spécifiques à la reproductibilité à long-terme.

6.1 Pérennisation

La reproductibilité doit se considérer sur le temps long. Bien que reproduire les derniers articles scientifiques est souvent attractif, il peut être important de rejouer des résultats anciens afin de les réanalyser à l'aune de nouvelles connaissances.

Malheureusement, il est quelquefois difficile d'anticiper les problèmes futurs :

Plateformes de code privées : GitHub ou Gitlab sont actuellement très populaires, Sourceforge l'était dans le passé. Cependant, une plateforme privée peut disparaître, être achetée, changer le mode d'accès aux données, etc.

Plateformes de code de laboratoires : les forges sont de très bons outils de développement. Par contre, leur pérennité n'est pas assurée. Ainsi, les forges https://gforge.inria.fr ou https://forge.imag.fr/ n'existent plus. L'intégralité des liens mentionnés dans les articles passés sont donc perdus.

Plateformes expérimentales : les expériences nécessitant du matériel particulier peuvent être difficiles à reproduire si le matériel n'est plus disponible.

Dans un tel cadre, il est donc essentiel de faire des choix de long-terme : préférer les plateformes de code ou données nationales (ex : HAL) ou internationales (ex : Zenodo), spécialisées (Software Heritage [8]) qui garantissent une continuité de service à long-terme. Pour les plateformes expérimentales, il est préférable d'utiliser des plateformes financées sur le long terme (ex : SLICES) ou qui peuvent être facilement redéployées (ex : WalT [22])

Recommandation 21 (Conférences et Reproductibilité) : Encourager les conférences à autoriser les auteur·ice·s à joindre un document détaillant l'approche expérimentale suivie dans le papier, ou de permettre d'avoir une page annexe supplémentaire en fin d'article dédiée à la présentation détaillée du dispositif de simulation/expérimentation.

Recommandation 22 (Pérennisation des résultats scientifiques en tant qu'auteur) : Il est recommandé de donner un contact durable pour les articles (ex : éviter les alias emails temporaires supprimés au bout de quelques années), afin que d'autres chercheur·euse·s de la communauté puissent contacter les auteur·ice·s pour d'éventuelles questions sur l'article. De même, les prototypes devraient (ré)utiliser des outils open-source existant afin que les résultats obtenus puissent être reproduits par la communauté, et bénéficier des dernières avancées des outils. Les solutions monolithiques sont souvent difficiles à faire évoluer.

6.2 Formation

Il est dommage que les étudiant·es ne soient pas confronté·es au problème de la réplicabilité lors de leurs études (au niveau master). Une reflexion a été menée dans la communauté académique étatsunienne sur cette problématique. En particulier, Lisa Yan et Nick McKeown partagent leur retour d'expérience sur la reproductibiltié en réseaux demandant aux étudiant·es de Stanford de reproduire 40 articles de recherche [10].

Un blog regroupe les résultats de reproduction d'expériences passées pour les valoriser [2]. La simplicité est la clé pour la ré-appropriation d'un code [7] :

> "Implement experiments in a way that is easily replicated by others – ideally, another researcher can install it via a single script command, run it with a single command, and generate output graphics with a single command."

Une telle compétence s'acquiert le plus facilement en répliquant des résultats précédents, pour ne pas reproduire les problèmes rencontrés.

Malheureusement, la formation en master s'est depuis quelques années écartée des compétences recherche, créant une différence importante avec le modèle nord-américain notamment. Ainsi, même si cela a été possible dans certains master [3], introduire un cours de méthodologie ciblant explicitement la reproductibilité dans chacun des masters de nos universités semble irréaliste.

Recommandation 23 (Stages de reproductibilité) : Proposer des stages notamment pour les étudiant·es en master consistant à reproduire des résultats de méthodes de l'état de l'art, comme c'est le cas par exemple dans les TER (Travail d'Étude et de Rercherche) de certaines formations (niveau Master 1).

Recommandation 24 (Participation à des hackathon ou compétitions dans les conférences) : Encourager la participation à des hackathons dans les conférences internationales sur la reproductibilité et la réplicabilité des articles scientifiques. De nombreuses initiatives existent déjà pour de nombreuses très bonnes conférences.

7 Valorisation pour les recrutements et la carrière

Les profils expérimentaux sont de plus en plus reconnus et valorisés dans les différents processus de recrutement scientifique, notamment avec la croissance de la rigueur et de l'importance de la reproductibilité dans nos communautés. C'est tout à fait possible aujourd'hui de mieux mettre en avant ces aspects dans un dossier en détaillant l'effort méthodologique et les différents apports à la communauté.

Il existe déjà plusieurs guides de différents instituts qui fournissent des recommandations/critères sur l'évaluation de travaux ouverts et expérimentaux. Ces guides peuvent être utilisés par les candidats pour mieux comprendre ce qui est attendu et par les évaluateur·ices pour mieux prendre en compte les travaux expérimentaux pour ne pas sous-estimer cet aspect de la recherche [14, 18].

Mettre en avant son implication dans la communauté au sens large aidera un·e évaluateur·ice. Il faut également contextualiser les travaux expérimentaux et fournir des indicateurs concrets pour souligner leur impact, les situer dans un cadre plus large et mettre en avant leur potentiel. Ainsi, un prototype peut servir à i) démontrer une preuve de concept, ii) être utilisé par la communauté de chercheur·euse·s, ii) être l'objet d'un transfert technologique. Il est donc nécessaire de décrire le contexte expérimental, ainsi que les objectifs associés (verrous scientifiques levés, valorisation, etc.).

Recommandation 25 (Contextualisation et indicateurs) : Nous encourageons les candidats à lire (et suivre) les différentes recommandations lorsqu'ils postulent à un concours. Beaucoup de comités produisent des guides à destination des candidats. Les indicateurs quantifiés sont importants qu'ils concernent par exemple la taille de la communauté,

2. https://reproducingnetworkresearch.wordpress.com/
3. Voir par exemple https://github.com/alegrand/SMPE

le nombre d'utilisations d'un logiciel, ou le nombre de laboratoires impliqués. Des clés de lecture peuvent être trouvées dans des outils d'évaluation déjà existants, comme la grille d'évaluation d'Inria [17] pour l'évaluation de la qualité et de l'impact des logiciels développés.

8 Conclusion

Nous avons dans ce document regroupé une partie des discussions menées durant l'atelier du 14 septembre 2024 à Paris organisé par le groupe de travail reproductibilité du GDR RSD. Ce document tente d'aller plus loin en collectant une liste de références et en essayant de donner un certain nombre de recommandations à destination de la communauté et du GDR sur nos pratiques expérimentales. La réflexion a vocation à être approfondie si le groupe de travail se poursuit au sein du GDR, une communauté devant s'emparer et porter ce sujet.

9 Références

1. Claerbout, J. F. & Karrenbach, M. *Electronic documents give reproducible research a new meaning* in *EG Technical Program Expanded* (Society of Exploration Geophysicists, 1992). https://doi.org/10.1190/1.1822162.
2. Iyer, A., Rosenberg, C. & Karnik, A. What is the right model for wireless channel interference ? *IEEE Transactions on Wireless Communications* **8,** 2662-2671 (2009).
3. Velho, P. & Legrand, A. *Accuracy study and improvement of network simulation in the SimGrid framework* in *International ICST Conference on Simulation Tools and Techniques* (2010).
4. Imbert, M., Pouilloux, L., Rouzaud-Cornabas, J., Lebre, A. & Hirofuchi, T. *Using the EXECO toolbox to perform automatic and reproducible cloud experiments* in *1st International Workshop on UsiNg and building ClOud Testbeds (UNICO, collocated with IEEE CloudCom 2013* (IEEE, Bristol, United Kingdom, déc. 2013). https://inria.hal.science/hal-00861886.
5. Gomez-Diaz, T. *Free software, Open source software, licenses. A short presentation including a procedure for research software and data dissemination* rapp. tech. arXiv :1409.3143 (arXiv, 2014). https://arxiv.org/pdf/1409.3143.
6. Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J. *et al.* The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* **3,** 160018. https://doi.org/10.1038/sdata.2016.18 (2016).
7. Canini, M. & Crowcroft, J. *Learning Reproducibility with a Yearly Networking Contest* in *Reproducibility Workshop* (Los Angeles, CA, USA, 2017), 9-13. https://doi.org/10.1145/3097766.3097769.
8. Cosmo, R. D. & Zacchiroli, S. *Software Heritage : Why and How to Preserve Software Source Code* in *iPRES 2017 : 14th International Conference on Digital Preservation* (Kyoto, Japan, 25 sept. 2017). https://www.softwareheritage.org/wp-content/uploads/2020/01/ipres-2017-swh.pdf%20https://hal.archives-ouvertes.fr/hal-01590958. published.
9. Scheitle, Q., Wählisch, M., Gasser, O., Schmidt, T. C. & Carle, G. *Towards an Ecosystem for Reproducible Research in Computer Networking* in *Reproducibility Workshop* (2017), 5-8. https://doi.org/10.1145/3097766.3097768.
10. Yan, L. & McKeown, N. Learning Networking by Reproducing Research Results. *SIGCOMM Comput. Commun. Rev.* **47,** 19-26. https://doi.org/10.1145/3089262.3089266 (mai 2017).
11. Ammour, L., Anne-Sophie Bonne, P. M., Schmittbiel, J.-M. & Souplet, J.-C. *Je code : quels sont mes droits ? Quelles sont mes obligations ?* rapp. tech. hal-02399517 (CNRS, 2019). %5Curl%7Bhttps://hal.science/hal-02399517/%7D.
12. Bajpai, V. *et al.* The Dagstuhl beginners guide to reproducibility for experimental networking research. *SIGCOMM Comput. Commun. Rev.* **49,** 24-30. https://doi.org/10.1145/3314212.3314217 (fév. 2019).
13. Desquilbet, L. *et al. Vers une recherche reproductible* (éd. régionale de formation à l'information scientifique et technique de Bordeaux, U.) 1-161. https://hal.science/hal-02144142 (Unité régionale de formation à l'information scientifique et technique de Bordeaux, mai 2019).
14. *Feuille de route du CNRS pour la science ouverte* (Centre national de la recherche scientifique, nov. 2019). https://www.cnrs.fr/sites/default/files/download-file/Plaquette_Science-Ouverte_18112019.pdf.
15. Pouzat, C., Legrand, A., Hinsen, K. & Farhi, L. *Recherche reproductible : principes méthodologiques pour une science transparente* https://www.fun-mooc.fr/en/courses/reproducible-research-methodological-principles-transparent-scie/. 2020.
16. Rosendo, D., Silva, P., Simonin, M., Costan, A. & Antoniu, G. *E2Clab : Exploring the Computing Continuum through Repeatable, Replicable and Reproducible Edge-to-Cloud Experiments* in *Cluster 2020 - IEEE International Conference on Cluster Computing* (Kobe, Japan, sept. 2020), 1-11. https://hal.science/hal-02916032.

17. Canteaut, A. *et al. Évaluation des Logiciels* Research Report (Inria, jan. 2021). https://inria.hal.science/hal-03110723.

18. Inria. *Conseils à destination des candidates et candidats à un poste de Chargé ou Chargée de Recherche Inria de Classe Normale (CRCN)* 2021. https://www.inria.fr/sites/default/files/2021-01/CE_guide-candidats-CRCN.pdf.

19. Leipzig, J., Nüst, D., Hoyt, C. T., Ram, K. & Greenberg, J. The role of metadata in reproducible computational research. *Patterns* **2.** https://doi.org/10.1016/j.patter.2021.100322 (2021).

20. Cherrueau, R.-A. *et al.* EnosLib : A Library for Experiment-Driven Research in Distributed Computing. *IEEE Transactions on Parallel and Distributed Systems* **33,** 1464-1477. https://inria.hal.science/hal-03324177 (juin 2022).

21. Cokelaer, T., Boulakia, S. C. & Lemoine, F. Reprohackathons : promoting reproducibility in bioinformatics through training. *Bioinformatics* **39,** i11-i20 (2023).

22. Duble, E. *et al. WALT - Software for automating network experiments* version 9.0. Oct. 2024. https://hal.science/hal-04726402.

23. Pouzat, C., Legrand, A., Hinsen, K. & Farhi, L. *Reproducible Research II : Practices and tools for managing computations and data* https://www.fun-mooc.fr/en/courses/reproducible-research-ii-practices-and-tools-for-managing-comput/. 2024.

24. *Artifact Review and Badging Version 1.1* https://www.acm.org/publications/policies/artifact-review-and-badging-current (2020).