HAL
open science

Comment générer des traces applicatives avec FIT IoT-LAB pour la science ouverte

Nina Santi, Brandon Foubert, Nathalie Mitton

To cite this version:

Nina Santi, Brandon Foubert, Nathalie Mitton. Comment générer des traces applicatives avec FIT IoT-LAB pour la science ouverte. CORES 2021 – 6ème Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication, Sep 2021, La Rochelle, France. hal-03216768

HAL Id: hal-03216768

https://hal.science/hal-03216768v1

Submitted on 4 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Comment générer des traces applicatives avec FIT IoT-LAB pour la science ouverte

Nina Santi[1] et Brandon Foubert[1] et Nathalie Mitton[1]

[1]*Inria, 40 Avenue Halley, 59650 Villeneuve-d'Ascq, France*

L'essor récent de l'apprentissage automatique et l'intérêt grandissant porté à l'intelligence artificielle a généré une demande de données de plus en plus importante. Le mouvement de la science ouverte, et plus particulièrement les données ouvertes, apportent une réponse à cette demande en offrant à tous un accès et un usage libre à des jeux de données. Dans cet article, nous proposons une méthodologie pour générer des traces correspondant à différentes applications IoT. Pour cela, nous caractérisons les différents types de trafic, comme leur fréquence de communication et la taille des paquets échangés. Puis, nous simulons ces applications sur la plateforme FIT IoT-LAB pour générer les traces. Les paramètres des simulations, tels que le nombre de nœuds employés, sont choisis selon les caractéristiques de l'application simulée. Les traces ainsi générées sont enrichies de plusieurs données qui permettent de déduire des métriques utiles, telles que le taux de livraison des paquets et le délai de bout en bout. Nous partageons en accès ouvert les traces obtenues ainsi qu'une base de code pour générer, manipuler et analyser les données obtenues de FIT IoT-LAB.

Mots-clefs : données en accès libre, science ouverte, internet des objets, banc d'essai, réseau de capteurs sans fil, IoT

1 Introduction

Les données sont aujourd'hui le cœur de multiples champs de recherches : elles permettent de tester des algorithmes, d'entraîner des modèles et plus généralement d'aider à la compréhension des règles inhérentes au monde qui nous entoure. Récemment, nous constatons une demande grandissante de jeux de données, notamment depuis l'essor récent de l'apprentissage automatique, directement dépendant de la qualité et l'exhaustivité des données utilisées. C'est dans ce contexte que le mouvement de la science ouverte apporte une réponse à cette demande et permet directement à la recherche d'avancer grâce à la grande disponibilité des données ouvertes. En plus de proposer des ressources libres, les données ouvertes sont un atout majeur concernant la reproductibilité des résultats, qui est d'une importance capitale dans de nombreux domaines scientifiques. De plus en plus de conférences et journaux proposent de soumettre au côté d'un article une courte extension pour présenter des données variées qui ont été utilisées pour les expériences et/ou simulation, appelées "artifacts", preuve de cet intérêt. C'est dans cet esprit que nous proposons dans cet article une méthode pour que chacun puisse générer des ensembles de données adaptés aux besoins spécifiques de ses recherches grâce à la plate-forme FIT IoT-Lab [ABF+15]. Nous mettons à disposition de la communauté les premières données que nous avons générées avec cette méthode ainsi que la base de code correspondante.

Nos contributions peuvent se résumer ainsi :

(i) Nous partageons la méthodologie employée pour produire les données avec la plateforme FIT IoT-LAB. Pour cela, nous avons utilisés différents outils fournis par la plateforme pour lancer les expérimentations et un code python pour analyser les résultats. *(ii)* Nous fournissons une caractérisation d'applications IoT pour les bâtiments intelligents et l'utilisons pour définir les paramètres des expérimentations. *(iii)* Finalement, nous décrivons les résultats bruts des simulations ainsi que les différentes métriques à disposition à partir de ces résultats et celles que l'on pourraient obtenir. Toutes ces données sont laissées en accès libre [†] avec une base de code permettant l'extraction de métriques supplémentaires.

†. `https://gitlab.irisa.fr/0000H82G/traces`

TABLE 1: Paramètres des simulations

Application	Nombre de noeuds	Fréquence d'envoi de message	Taille de charge utile (octets)	Durée (minutes)
Systèmes CVC	100	1 paquet par 4 minutes	60	60
Éclairages intelligents	100	1 paquet par 8 minutes	30	90
Systèmes d'urgence	40	1 paquet par 30 secondes	127	10
Surveillance	30	99 paquets par secondes	127	10
Réalité virtuelle ou augmentée	10	197 paquets par secondes	127	10
VoIP	10	16 paquets par secondes	127	10

2 Méthodologie

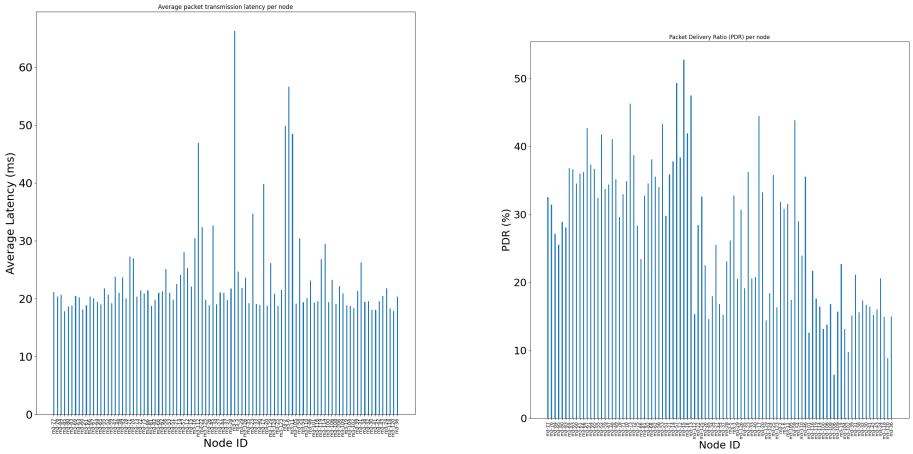
Nous avons utilisé la plateforme FIT IoT-LAB. La plateforme propose des noeuds ouverts, c'est-à-dire des noeuds complètement programmables. Nous avons programmé ces noeuds pour qu'ils échangent des paquets en broadcast avec la couche MAC IEEE 802.15.4 et le protocole de routage RPL.Cela nous a permis de simuler les différents trafics des applications.

Les applications simulées concernent plusieurs cas d'utilisations, notamment pour les bâtiments intelligents. Nous simulons un système CVC (chauffage, ventilation, climatisation) et des éclairages intelligents, qui régulent leurs activités selon l'environnement. Nous supposons le bâtiment également équipé *i)* d'un système d'urgence qui surveille les point sensibles d'un bâtiment et alerte si des évènement dangereux interviennent, comme une fuite de gaz, *ii)* de caméras de surveillance, *iii)* de salles de réalité virtuelle ou augmentée, *ivv)* de la Voix sur Ip (VoIP) qui sert pour les services d'assistance automatique et la reconnaissance de voix.

Avant de lancer les expérimentations, nous caractérisons ces applications pour décider des différents paramètres. Ces applications sont caractérisées par la fréquence de messages échangés, le nombre de nœuds utilisés, leur densité et le type de message. Les systèmes CVC et les éclairages intelligents ont une fréquence d'échange faible, mais avec un nombre important de nœuds. Ces deux applications échangent des fichiers simples. Les systèmes d'urgence emploient un nombre modéré de nœuds avec une fréquence d'envoi de message elle aussi modérée. Les messages échangés sont des images et des sms. Les systèmes de surveillance avec caméras ont un nombre modéré de nœuds avec une haute fréquence de messages, les messages sont les vidéos prises par les caméras. Les applications de réalité augmentée et virtuelle ont un petit nombre de nœuds, une fréquence de messages élevée et échangent de la vidéo. La Voix sur IP (VoIP) dans l'Internet des objets tourne sur un faible nombre de nœuds avec une fréquence de message modérée. Les messages embarquent de la voix. On a simulé l'échange d'images par des paquets de plus grande taille, comparés aux applications qui échangent des fichiers simples. Une fois la caractérisation faite, nous décidons des paramètres des simulations pour chaque type d'application. Ces paramètres sont résumés dans la Table 1. De même, la durée des expérimentations est ajustée selon la fréquence d'échange de paquets.

Pour coder les noeuds, nous nous sommes basés sur un base de code contiki [‡] que nous avons modifiée pour nos besoins. Nous ajustons la fréquence d'envoi des messages des noeuds et la taille de la charge utile des paquets envoyés selon les paramètres choisis, ainsi que l'ajout d'un identifiant de paquet. De plus, nous demandons aux noeuds d'écrire en sortie les paquets qu'ils envoient ou réceptionnent. Pour le coté matériel, nous avons choisi d'utiliser pour toutes les simulations les noeuds ouverts M3 [§] basés sur les micro-contrôleurs STM32 avec un processeur ARM Cortex M3 car ce sont ceux qui répondent aux besoins du plus grand nombre d'applications. Nous avons mené les simulations avec un script et des outils fournis par la plateforme. Nous avons utilisé l'interface client Python pour programmer les noeuds avec notre code et lancer les simulations. Puis avons récupéré les résultats des simulations, c'est-à-dire la sortie des noeuds, avec l'agrégateur de liens en série. Quand un noeud écrit sa sortie, la plateforme ajoute un horodatage et le nom du noeud, utiles pour retrouver les informations. Le script utilisé est disponible avec les données.

‡. Disponible ici : https ://github.com/iot-lab/contiki/tree/master/examples/ipv6/simple-udp-rpl
§. Plus d'information : https ://github.com/iot-lab/iot-lab/wiki/Hardware-M3-node



(a) Délai moyen de transmission par nœud

(b) Taux de réception par nœud

FIGURE 1: Exemples de métriques recueillies pour les systèmes CVC.

3 Résultats

Nous obtenons comme résultat le trafic généré, c'est-à-dire les échanges de paquets entre les nœuds, conscrits dans un fichier dont chaque ligne est une sortie de nœud et donc représente un paquet reçu ou envoyé. Chaque ligne contient l'horodatage de la transmission ou la réception du paquet, le nom du nœud récepteur ou transmetteur, le contenu du paquet qui indique si c'est un message reçu ou transmis et enfin un identifiant de message. Nous avons séparé ce fichier de données brutes en deux autres fichiers pour une meilleure analyse. Un des fichiers contient les messages reçus et l'autre les messages transmis. Les deux fichiers contiennent les informations des données brutes, mais on retrouve en plus dans les messages transmis le nombre de nœuds qu'a réussi à atteindre le message ; dans le fichier des messages reçus, on retrouve le délai de réception en millisecondes. On peut utiliser ces données pour entraîner des modèles d'apprentissage automatiques et connaître la relation entre les différentes métriques de l'application. Dans notre cas, les données doivent servir à prédire le délai selon l'état du réseau et de l'application. C'est pourquoi nous avons extrait différentes métriques comme le ratio de livraison de paquet [KFQA13] et le délai moyen de transmission de paquet. La figure 1 illustre le délai moyen de transmission et le ratio de livraison de paquet par nœud pour les éclairages intelligents. D'autres métriques peuvent être extraites et mises en relation. La séparation des données, les métriques et les graphes de la figure 1 sont produits avec un code python, disponible en libre accès sur le Gitlab. Ce code est prévu pour être réutilisable afin d'extraire d'autres métriques, comme la fenêtre de réception des messages, ou sur d'autres données avec le même format.

4 Discussion

Les données pourraient être enrichies par des informations supplémentaires, comme l'utilisation de CPU ou l'arbre RPL constitué. De plus certaines méta-données utiles à la reproduction des expérimentations ne sont pas automatiquement fournies dans les résultats, par exemple le site d'expérience. L'utilisateur doit de lui-même rechercher ces méta-données à la fin de son expérience. Finalement les données sont contenues dans des fichiers texte simple. La représentation des résultats pourrait être plus avancée, par exemple en fournissant directement une base de données. Cela permettrait de faciliter l'exploitation des résultats. Ces points n'ont pas été intégrés dans un premier temps, car nous avons fait le choix de d'abord proposer des outils et des données simples d'utilisation. Nous laissons pour un travail futur toutes ces potentielles améliorations.

5 Travaux connexes

Pflanzner et al.[PFK19] proposent un service d'indexation pour récupérer des traces ouvertes d'applications disponibles sur internet. [KdTE+19] présente un jeu de données à partir d'un banc d'essai composé de capteurs météorologiques alimentés par panneaux solaires. Les données contiennent la consommation et la production d'énergie de l'application pour aider la recherche dans les applications IoT auto-alimentées. Xiao et al. [XLW+20] introduisent une solution d'unité embarquée prête à l'emploi pour collecter des données de trajectoires de véhicules. Leur solution contient un algorithme d'apprentissage profond pour combler les manques dans la récolte de données, du par exemple à des pannes de GPS. [JDT+19] fournit un jeu de données à partir de la plateforme FlockLab. La collecte de donnée est effectuée sur une longue période de temps pour capturer les motifs périodiques à long terme qui influent sur la qualité des liaisons sans fil. Nos données complémentent les jeux actuelles et peuvent être générées à la demande.

6 Conclusion

Nous avons proposé une méthodologie pour générer des traces applicatives de l'IoT avec la plateforme FIT IoT-LAB. Les applications correspondent à différents cas d'usage et types de trafic. Nous avons d'abord caractérisé le trafic de ces applications pour définir les paramètres de simulation. Les traces ainsi générées sont enrichies d'informations, telles qu'un identifiant de message et un horodatage. Cela permet d'extraire des métriques intéressantes, comme le délai de réception. Ces traces peuvent être utilisées pour entraîner des modèle d'apprentissage automatique et connaître la relation entre les différentes métriques des application. Elles sont laissées en accès ouvert pour qu'elles puissent profiter aux différentes recherches menées. Une base de code Python est aussi mise à disposition pour manipuler ces données ainsi qu'un script pour lancer des expérimentations.

Nous remerçions Aris Leivadeas de ETS Montréal ainsi que son équipe pour sa requête ayant initié ces travaux. Ces travaux ont été partiellement financés dans le cadre du projet CHIST-ERA DRUID-NET.

Références

[ABF+15] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. Fit iot-lab : A large scale open experimental iot testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015.

[JDT+19] R. Jacob, R. Da Forno, R. Trüb, A. Biri, and L. Thiele. Dataset : Wireless link quality estimation on flocklab - and beyond. In *Proceedings of the 2nd Workshop on Data Acquisition To Analysis*, 2019.

[KdTE+19] M. Kuzman, X. d. Toro García, S. Escolar, A. Caruso, S. Chessa, and J. C. López. A testbed and an experimental public dataset for energy-harvested iot solutions. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, 2019.

[KFQA13] M. F. Khan, E. A. Felemban, S. Qaisar, and S. Ali. Performance analysis on packet delivery ratio and end-to-end delay of different network topologies in wireless sensor networks (wsns). In *2013 IEEE 9th International Conference on Mobile Ad-hoc and Sensor Networks*, 2013.

[PFK19] T. Pflanzner, Z. Feher, and A. Kertesz. A crawling approach to facilitate open iot data archiving and reuse. In *2019 Sixth International Conference on Internet of Things : Systems, Management and Security (IOTSMS)*, 2019.

[XLW+20] Z. Xiao, F. Li, R. Wu, H. Jiang, Y. Hu, J. Ren, C. Cai, and A. Iyengar. Trajdata : On vehicle trajectory collection with commodity plug-and-play obu devices. *IEEE Internet of Things Journal*, 7, 2020.