COMMENT GÉNÉRER DES ÉCHANTILLONS UNIFORMES SUR DE GRANDES BASES DE DONNÉES

Yann Busnel[1]

[1] Crest (Ensai) / Inria Rennes, Campus de Ker Lann, 35170 Bruz, France : yann.busnel@ensai.fr

Résumé. Au sein d'un réseau ou d'un système distribué informatique, un service de sondage aléatoire idéal se doit de retourner un pointeur vers un nœud (qui peut-être une machine, un processus, un service, etc.), correspondant à un échantillon indépendant sans biais du groupe considéré. Ce service dit d'échantillonnage uniforme offre une primitive simple servant de brique de base à de nombreuses applications dans les systèmes à très grande échelle, telles que la dissémination de l'information, la métrologie (via des opérations de comptage et des métriques statistiques), la synchronisation d'horloge logique, etc. Malheureusement, la présence inévitable d'agents malveillants dans ces systèmes ouverts entrave la construction de ces services d'échantillonnage.

Nous proposons ici une solution au problème d'échantillonnage uniforme dans les systèmes informatiques à grande échelle en présence de comportements byzantins. Ces derniers reflètent la non-conformité des résultats d'un système qui ne respecte pas ses spécifications. Les pannes byzantines les plus difficiles à appréhender proviennent principalement d'attaques volontaires visant à faire échouer le système (sabotage, virus, déni de service, etc.). Nous proposons un premier algorithme permettant d'uniformiser à la volée un flux de données (ou d'items) de taille non bornée, sous l'hypothèse que les probabilités exactes d'occurrence des items sont connues. Nous modélisons le comportement de notre algorithme par une chaîne de Markov et fournissons les résultats de l'étude du régime stationnaire et transitoire. Notre second algorithme relâche l'hypothèse forte de connaissance de la probabilité d'occurrence des items dans le flux initial. Ces probabilités sont alors estimées à la volée en utilisant une structure de données de type agrégat, avec un espace mémoire logarithmique en la taille du flux. Nous évaluons ensuite la résilience de cet algorithme face à des attaques ciblées et par inondation. De plus, nous quantifions l'effort que doit fournir l'adversaire (i.e., le nombre d'items à injecter dans le flux initial) pour violer la propriété d'uniformité