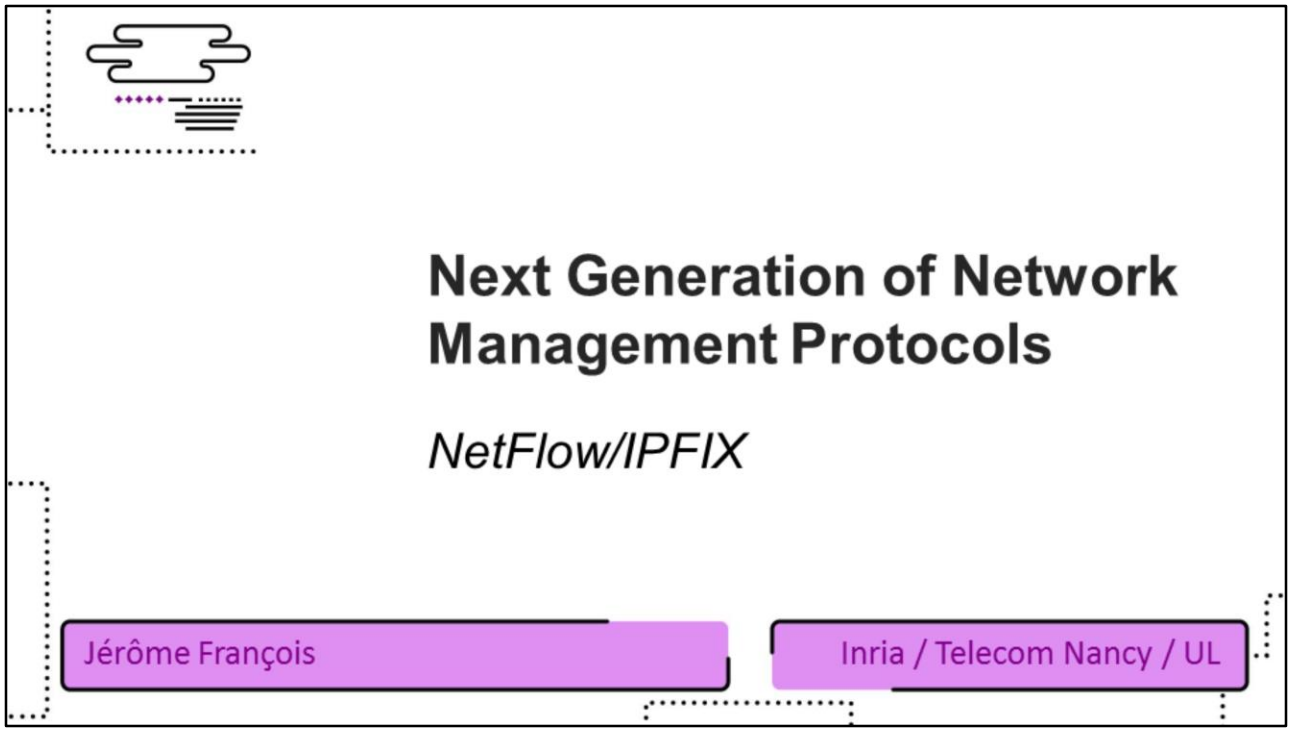
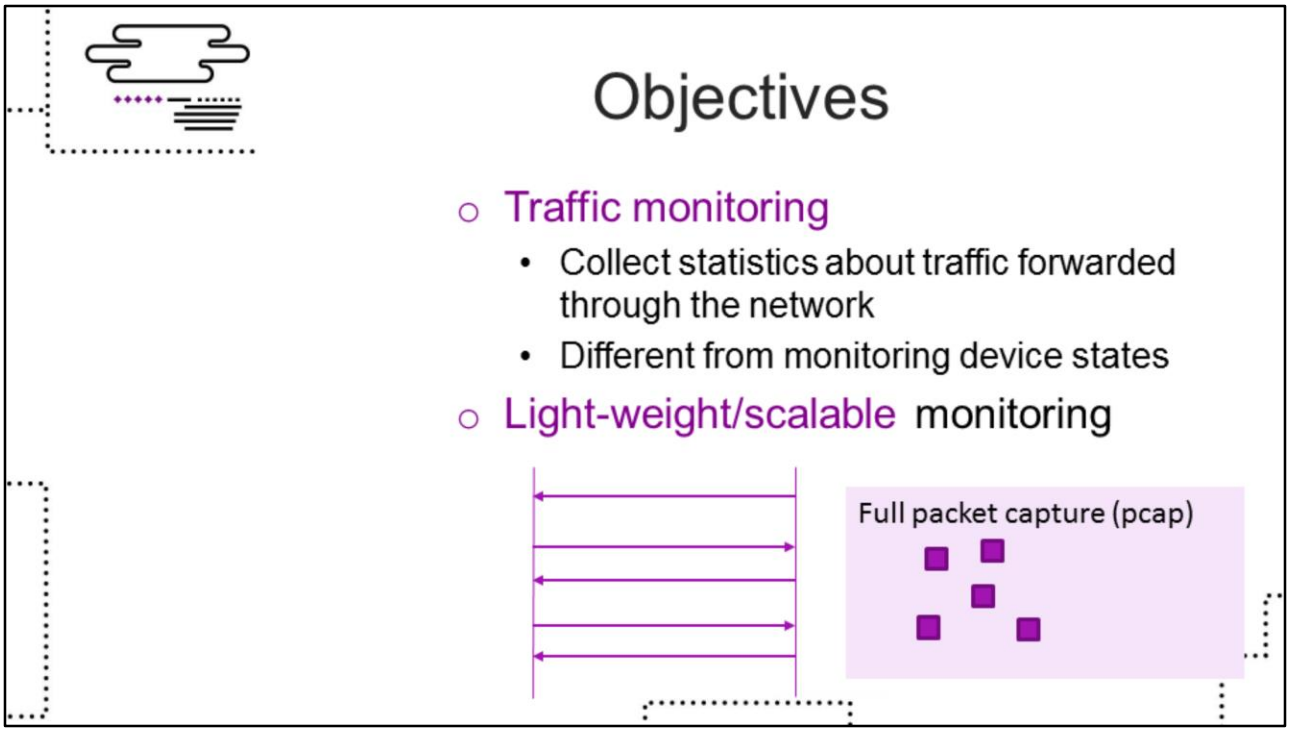


Cette semaine, nous allons aborder un autre protocole de supervision réseau normalisé au cours des années 2000 et très largement adopté. Il s'agit du protocole IPFIX anciennement NetFlow. Ce protocole entre dans la catégorie des protocoles de gestion ou supervision réseau, même s'il se cantonne, comme on va le voir, uniquement au monitoring de trafic IP.

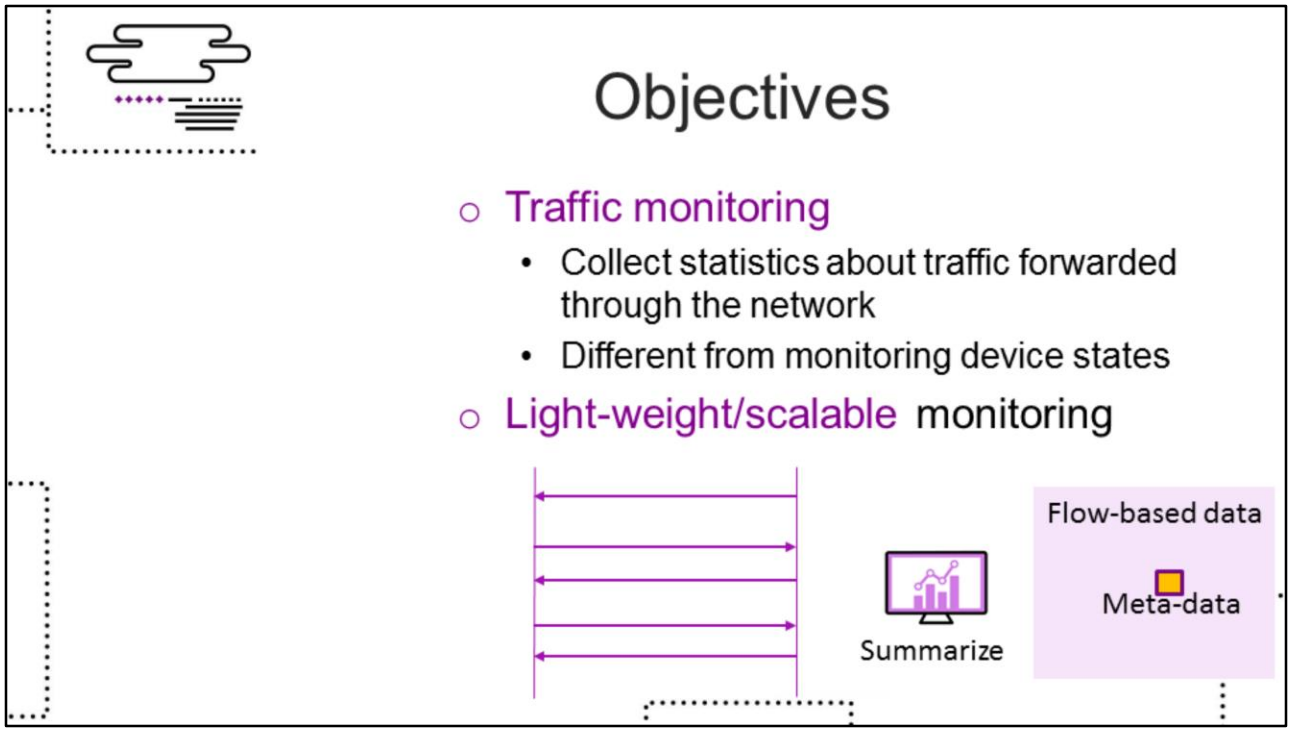


Quel est donc l'objectif de ce protocole ? Le monitoring du trafic réseau est essentiel dans de nombreux domaines, que ce soit pour vérifier la bonne marche d'un réseau, en détecter les problèmes, voire étudier sa dynamique pour mieux le provisionner dans le futur.

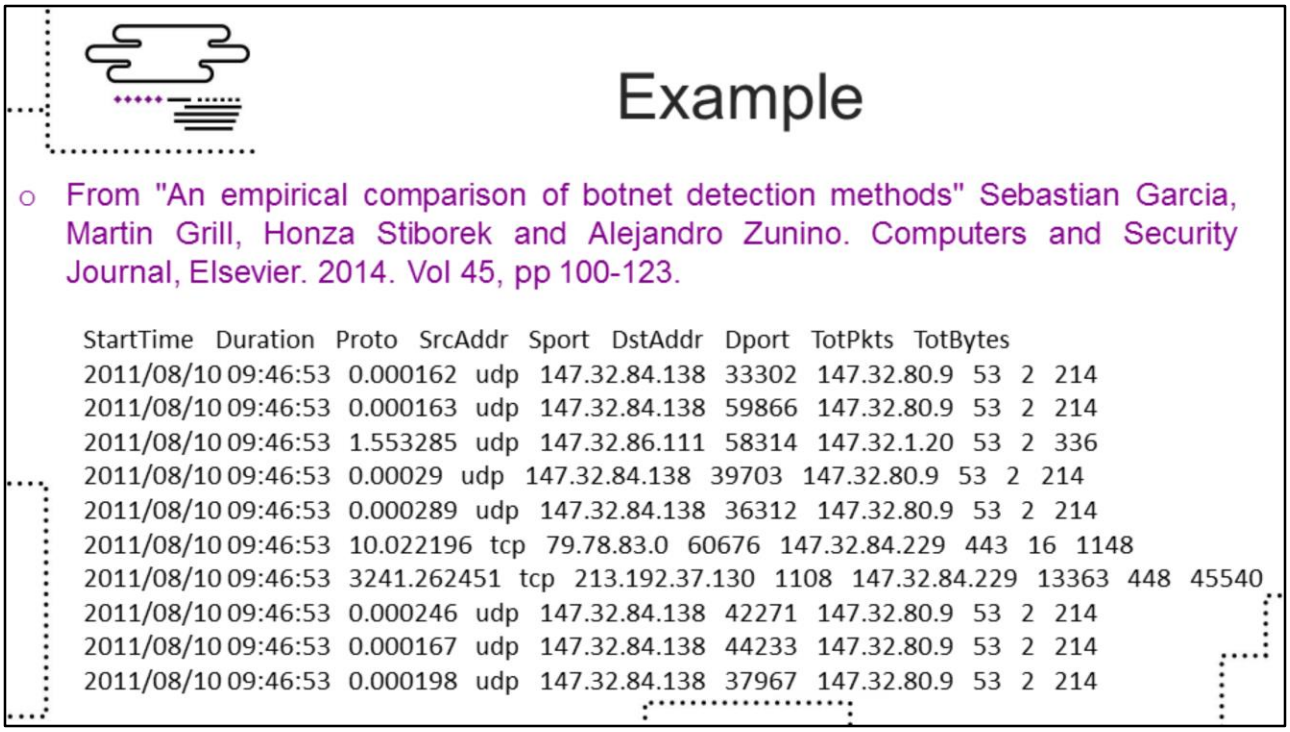
D'autres moyens peuvent être utilisés, à l'instar de NETCONF ou SNMP. Ces derniers sont très bien pour récupérer des informations relatives aux équipements mais sont très peu adaptés au traffic en lui-même. En effet, ils sont centrés sur l'équipement. On aura alors facilement accès à des informations sur les éléments sous-jacents de ce dernier. Par exemple, on pourra récupérer le nombre de paquets échangés sur un routeur, voire connaître ce nombre par interface ou par routes dans la table de routage. A l'inverse on ne connaîtra pas exactement quels ont été les paquets échanges, entre qui et qui, avec quel protocole, à quel moment, etc.

L'idée d'IPFIX est donc de récupérer des informations à propos du trafic qui est transmis, typiquement au niveau d'un routeur. Vous avez probablement expérimenter la capture de trafic complète, dite full-capture, avec des outils comme wireshark. Il s'agit alors d'écouter une ou plusieurs interfaces réseaux et de faire une copie des paquets échangés dans un fichier pcap. On retrouve ainsi l'intégralité des paquets, même s'il est cependant possible de limiter à une partie seulement. Notamment, le payload, c'est à dire la partie données applicatives, est souvent exclu.

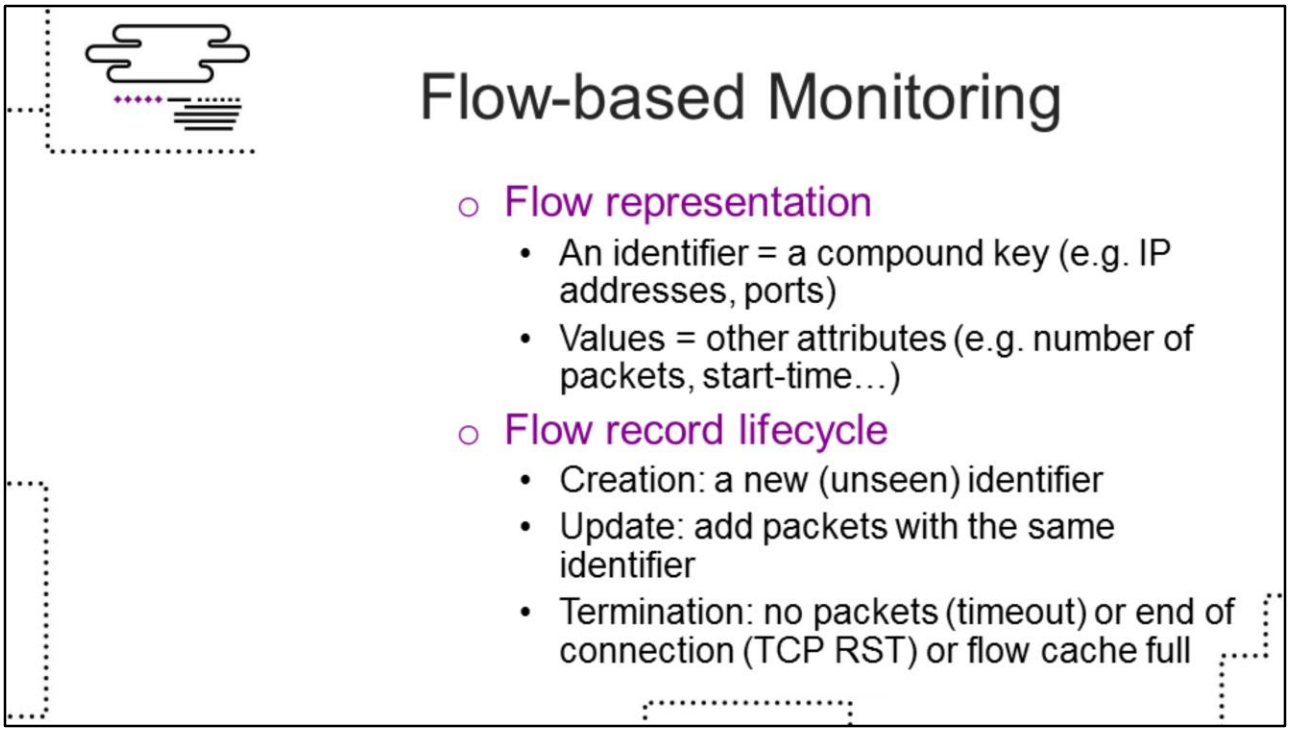
Ce type d'approche se révèle pratique lorsqu'il faut ensuite analyser précisément le contenu des paquets mais dans la plupart des exemples cités précédemment, seules des statistiques plus globales sont largement suffisantes. De plus sauvegarder et analyser des captures complètes posent naturellement un problème de passage à l'échelle ; imaginez un opérateur national devant garder une copie intégrale de son traffic, sachant qu'il faut en principe garder un historique sur plusieurs jours, mois voire années pour pouvoir faire des analyses intéressantes.



Avec une approche de type flux comme IPFIX, un processus intermédiaire récupère un ensemble de paquets, les agrège et finalement les sauvegarde sous un format condensé. C'est donc une approche plus légère. Une première question est donc « comment sont agrégés les paquets ? Selon quels critères? »



Un petit aperçu. Vous observez ici une capture de type NetFlow. Vous remarquez en fait que chaque ligne, ou chaque flux, représente une communication, soit un échange de paquets, entre deux adresses IP et deux ports précis, sur TCP ou UDP. C'est ce qui va constituer la clé d'agregation. Tous les paquets IP ayant en commun ces différents éléments vont au final être regroupés ensemble. On retrouve alors le nombre total de paquets échangés, TotPkts, le nombre d'octets échangés ainsi que l'horodatage et la durée du flux. Notez que cet exemple est tiré de l'article cité ici et montre comment une représentation de ce type, qui semble très simple, peut servir dans le domaine de la cybersécurité en détectant des botnets.

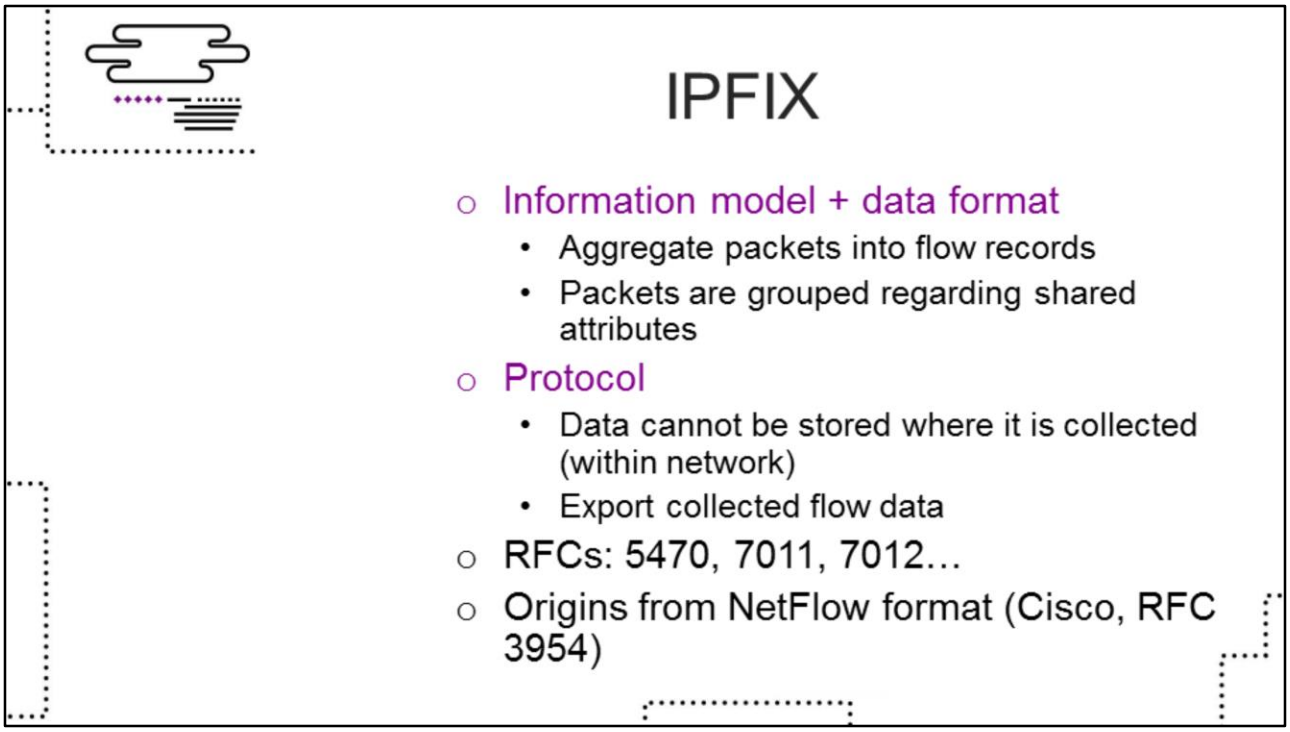


Avec IPFIX, plusieurs types de champs ou attributs vont être récupérés dans chaque paquet, ceux qui composent la clé comme les adresses IP (Cela représente l'identifiant du flux) auqxuel sont associées des valeurs comme le nombre de paquets ou d'octets.

On a donc la clé et les valeurs sachant que la clé peut, et est, souvent composée. Un nouveau flux est initié au moment où l'on observe son premier paquet, c'est à dire qu'aucun flux actif n'existe avec la même clé. Chaque nouveau paquet avec cette clé est donc associée au flux et lorsque le flux devient inactif, celui-ci est considéré comme terminé.

Une des questions principales est donc le critère de terminaison d'un flux. Le plus compréhensible est la présence du flag reset, RST, en TCP. Cette méthode n'est cependant qu'applicable avec TCP et il est également possible que certaines connexions TCP ne soient pas fermées correctement. Pour palier ce problème, un flux est également considéré comme expiré après un certains temps d'inactivité, c'est-à-dire lorsqu'aucun paquet associé n'a été reçu depuis un certain delai. Cela correspond à ce qui s'appelle l'inactive timeout, généralement de l'ordre de quelques dizaines de secondes. De manière générale, notez bien qu'un flux est uniquement identifié par sa clé mais que plusieurs d'entre eux peuvent se partager la même, si ces derniers occurent à des instant différentes. En effet, une fois le flux expiré, la clé est libérée.

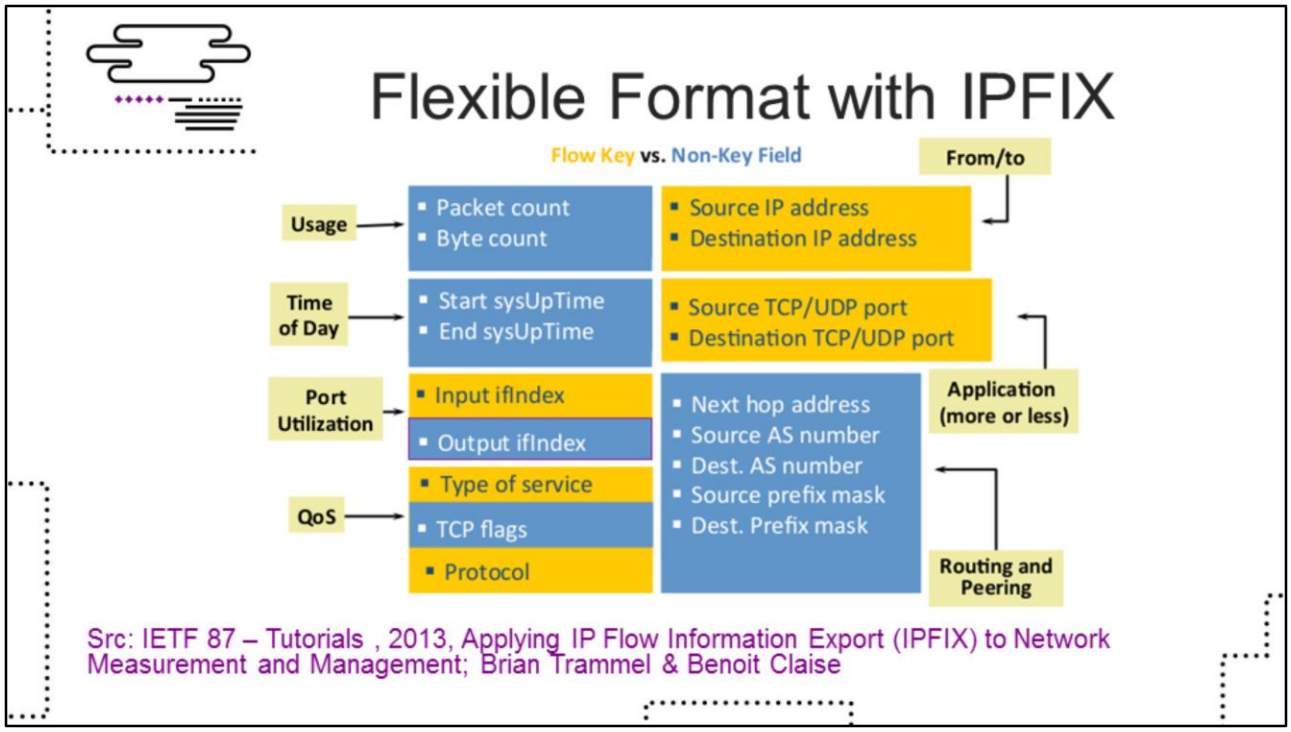
Egalement, l’active timeout qui est un mécanisme un peu arbitraire. Il permet de terminer un flux même si des paquets continuent d’arriver. Ce type de timeout est généralement de l’ordre de quelques dizaines de minutes. Enfin la dernière condition pour qu’un flux se termine est lorsque que le cache de flux actifs est plein et qu’il est indispensable de libérer de la place.



IPFIX a été normalisée à l'IETF et est en fait décomposé en plusieurs parties.

Tout d'abord, les données échangées sont définies par un format spécifique et modèle d'information. La norme définit donc comment les paquets doivent être groupés et correspond aux explications sur les diapositives précédentes. Ensuite le protocole. Ce protocole s'inscrit dans la supervision de réseaux à large échelle. Il ne s'agit donc pas de collecteurs individuels sur le réseau qu'il faudra interroger manuellement mais d'une collecte décentralisée et organisée pour rapatrier l'ensemble des données de monitoring, d'un ou plusieurs points d'observation. Cela évite à la fois d'avoir des capacités de stockage sur les équipements qui capturent le trafic, comme les routeurs, et permet d'avoir une vue globale du trafic IP.

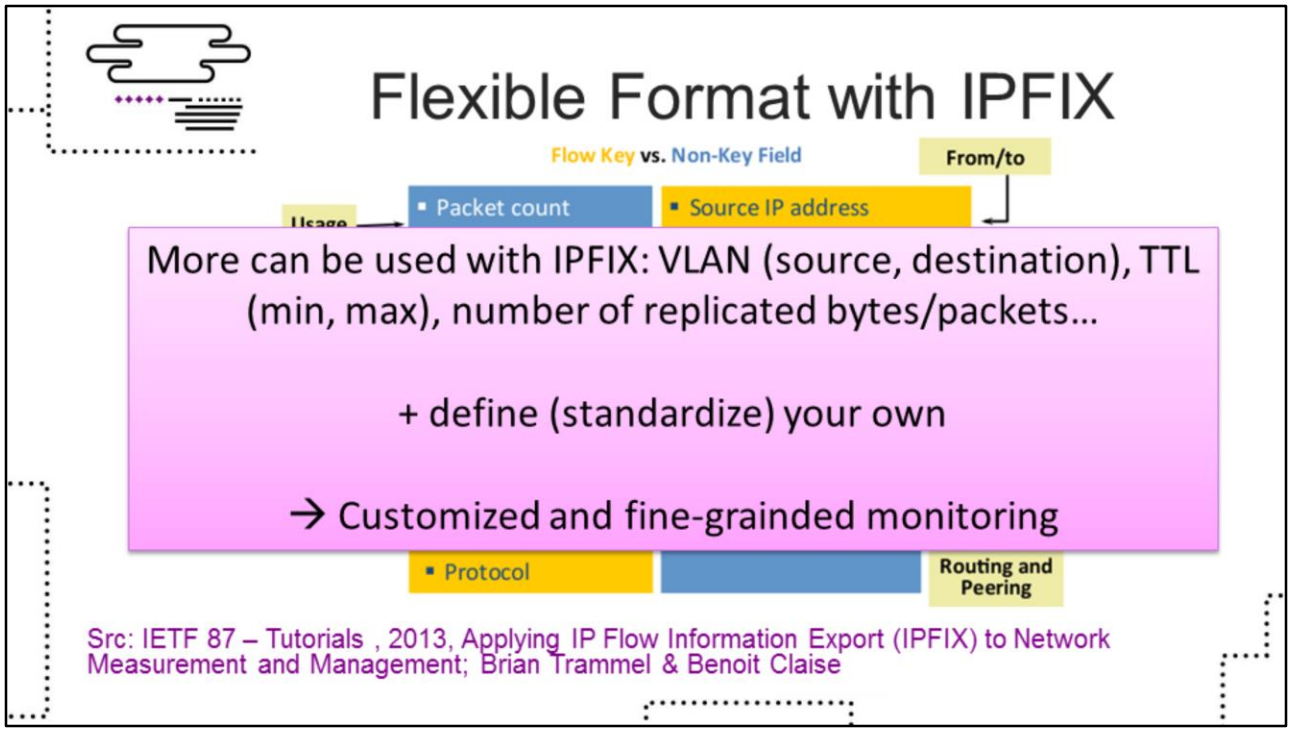
Vous entendrez souvent parler de Netflow qui est à l'origine d'IPFIX. Une des versions encore couramment utilisée de NetFlow est la version 5 qui a ensuite été amenée à évoluer pour finalement servir, dans sa version 9, à la base de la norme d'IPFIX.



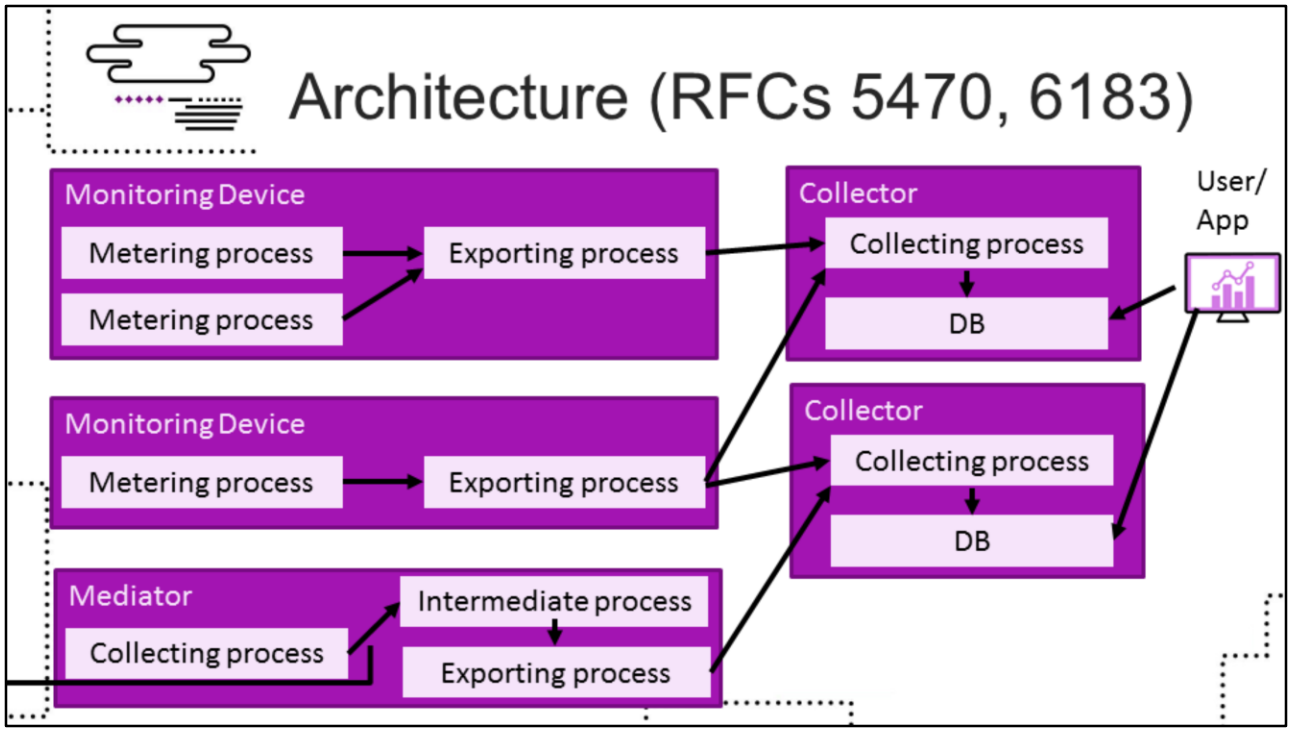
Lorsque l'on souhaite utiliser IPFIX, il est essentiel en premier lieu de définir la clé, en fait une clé composée, et les valeurs d'un flux.

Ici je présente une illustration d'un tutorial donné à l'IETF en 2013 qui différencie ce qui peut être utilisé comme clé, en jaune, et ce qui peut être utilisé comme valeur, en bleu.
Les champs en jaunes représentent d'ailleurs les clés initialement définis par Cisco pour le NetFlow, puis repris ensuite comme clé de base dans IPFIX. Néanmoins, rien n'empêche d'en prendre un sous-ensemble. Par exemple, on peut uniquement sélectionner les adresses IP source et destination pour agréger les paquets. Cela permet notamment d'en déduire une matrice de trafic de type origine destination. En ajoutant les ports, on a alors plus d'informations sur les applications utilisées.

On note dans ces clés standard, l'interface sur laquelle le flux est collecté ainsi que le type de service pour différencier les flux selon la QoS. Aussi dans les valeurs, on retrouve des informations sur le routage des paquets.



De la même manière que pour les clés, ce sont les valeurs par défaut ici mais bien d'autres sont disponibles et sont normalisées dans différents RFCs. Vous pouvez également définir les votres, les utiliser et même les proposer à la standardisation. IPFIX permet donc vraiment d'affiner le monitoring pour qu'il réponde au mieux aux besoins du type de monitoring visé.


Architecture (RFCs 5470, 6183)
Monitoring Device
Metering process
Metering process
Exporting process
Monitoring Device
Metering process
Exporting process
Mediator
Collecting process
Intermediate process
Exporting process
Collector
Collecting process
DB
Collector
Collecting process
DB
User/
App

Petit point de détail, je vous ai mentionné les flux très longs qui pouvaient expiré avec l'active timeout. Ce type de flux a un statut un peu particulier en réalité, l'idée étant que pour éviter que l'utilisateur n'accède à l'information trop tard, on force leur expiration et donc leur exportation in fine. Cependant, le même enregistrement n'est pas remis à zéro au niveau du processus de mesure qui continue à être mis à jour de façon à pouvoir faire plus tard l'objet d'un nouvel export.

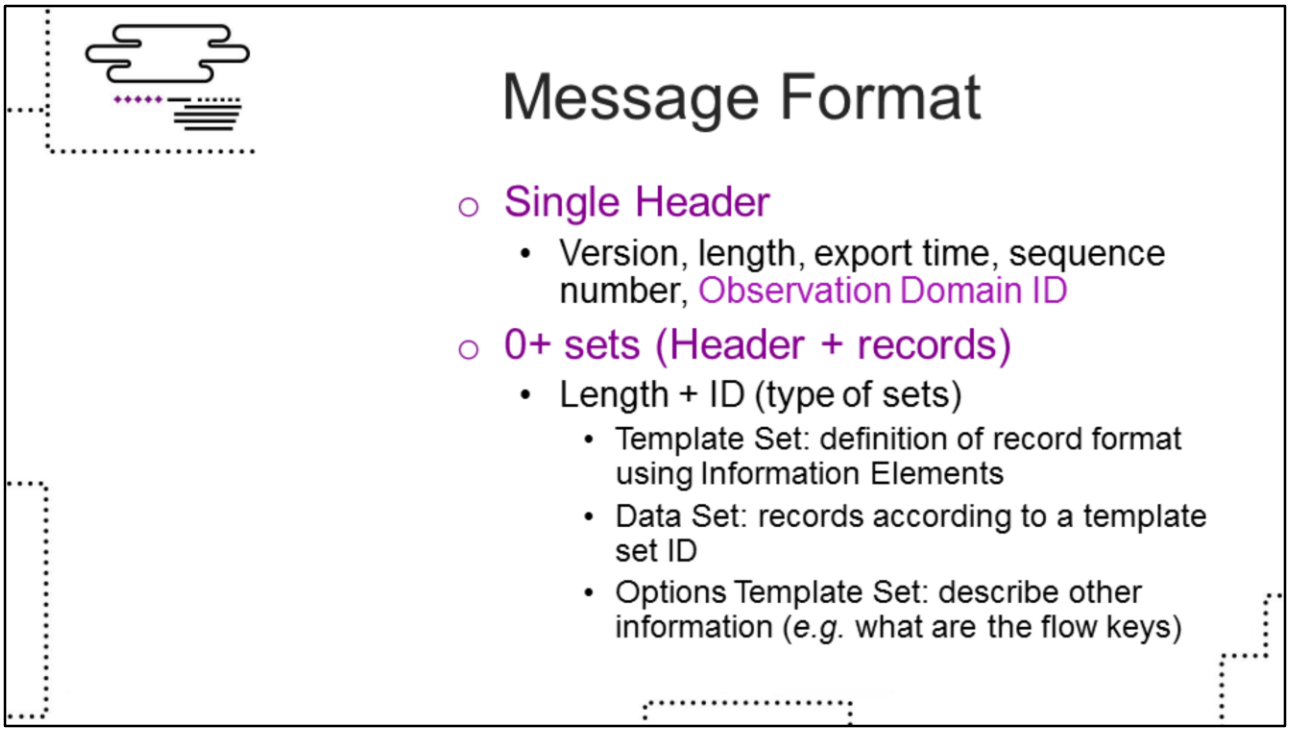
En réalité, il peut y avoir plusieurs processus de mesure sur le même équipement. Reprenons en détail ce processus. A chaque paquet capturé, la première chose qu'il fait ou devrait faire est l'horodatage afin de garantir sa précision. Ensuite ce paquet est associé à un enregistrement existant dans le cache ou crée une nouvelle entrée. Deux étapes optionnelle peuvent avoir lieu en amont.
Premièrement un échantillonnage peut être appliquer pour ne considérer qu'une fraction des paquets capturés, par exemple 1 sur 100, les métriques devant ensuite être estimées comme par exemple le nombre d'octets multipliés par 100. On perd donc en précision, mais le seul but ici est d'être plus efficace et donc de gagner en performance.

Deuxième étape optionnelle, on peut se cantonner à certains paquets spécifiques via des filtres, par exemple pour uniquement collecter des flux TCP ou ceux à destination de ports particuliers. Ainsi on peut avoir plusieurs processus de mesures avec différentes fonctions d'échantillonnage et de filtrage. Pour illustrer, on peut vouloir configurer un grain fin sur un port particulier, comme par exemple le port TCP 80 et spécifier un grain large sur une portion du trafic que l'on considère moins intéressante, avec donc un échantillonnage élevé.

IPFIX est par nature distribuée et permet d'avoir une vue plus générale du réseau en multipliant les équipements supervisés. Un collecteur peut donc servir à récupérer les informations de plusieurs équipements. De la même façon les flux enregistrés peuvent être envoyés à plusieurs collecteurs.

Reste un cas particulier, le médiateur ou mediator en anglais. Celui-ci s'intercale entre l'équipement supervisé et le collecteur. Il joue plus ou moins le rôle d'un proxy avec des capacités de transformation sur les flux. Une fois de plus l'objectif est d'éviter de surcharger les équipement qui servent de points d'observation. En effet, un routeur a des ressources limitée et doit en priorité jouer son rôle de routeur. Certains traitement supplémentaires comme l'anonymization des flux, le filtrage ou la transformation des flux pour y apporter plus ou moins d'information, sont alors délocalisés au médiateur ; plus particulièrement au processus intermédiaire ou intermediate process. Un médiateur dispose ainsi à la fois d'un processus de collecte et d'export.

D’un point de vue protocolaire, il peut également jouer le rôle de proxy car nous verrons un peu plus tard que plusieurs protocoles peuvent être utilisés.

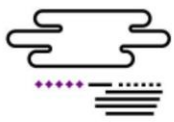


IPFIX définit donc un protocole de communication pour échanger les flux avec un format d'échange bien spécifique.
Un message IPFIX commence tout d'abord par une entête avec la version du protocole, en principe la 10 (car en dessous on parle alors de Netfow), la longueur du message, un horodatage du message, un numéro de séquence pour éliminer les doublons et détecter les pertes et aussi un identificateur du point d'observation. C'est grosso modo un identifiant du routeur ou de tout autre équipement qui envoie les données. On a ensuite un ensemble de sets. Chaque set est en composé d'une entête et d'enregistrements. On distingue particulièrement trois types de sets.

Les templates permettent de définir quel est le format des flux, la définition des clés et valeurs. Chaque élément utilisé, comme les adresse IP ou les ports, sont en fait ce qu'on appelle des information elements. Ensuite, le set de données ou data sets. Ces derniers contiennent alors des enregistrements de flux, ou disons simplement des flux, selon le format défini dans le template set. D'ailleurs les templates sont également identifiés par un identifiant et rien n'empêche d'en définir plusieurs et d'envoyer des datasets de différents formats, puisque ces font référence au template utilisé. Ils peuvent aussi faire référence à un format antérieur défini dans un message précédent, ce qui évite à chaque fois de redefinir le format avec un template pour chaque nouveau message d'export.

Enfin, les option template permettent de décrire tout autre type d'information. Par

exemple, il est possible que le processus de mesure ignore certains paquets, par exemple par manque de ressource, et il peut alors en informer le collecteur qui a une idée plus précise sur la véracité des mesures. Cela sert aussi à préciser quelle clé utiliser.



Protocol

- Unidirectional (push-based)
- Rely on a transport protocol
  - UDP: unreliable → only within a single domain
  - TCP: might be use over Internet
  - SCTP (Stream Control Transmission Protocol) = default choice:
    - Connectionless
    - Congestion-control
    - Streams (message ordering)
- Security: TLS/TCP, DTLS/SCTP