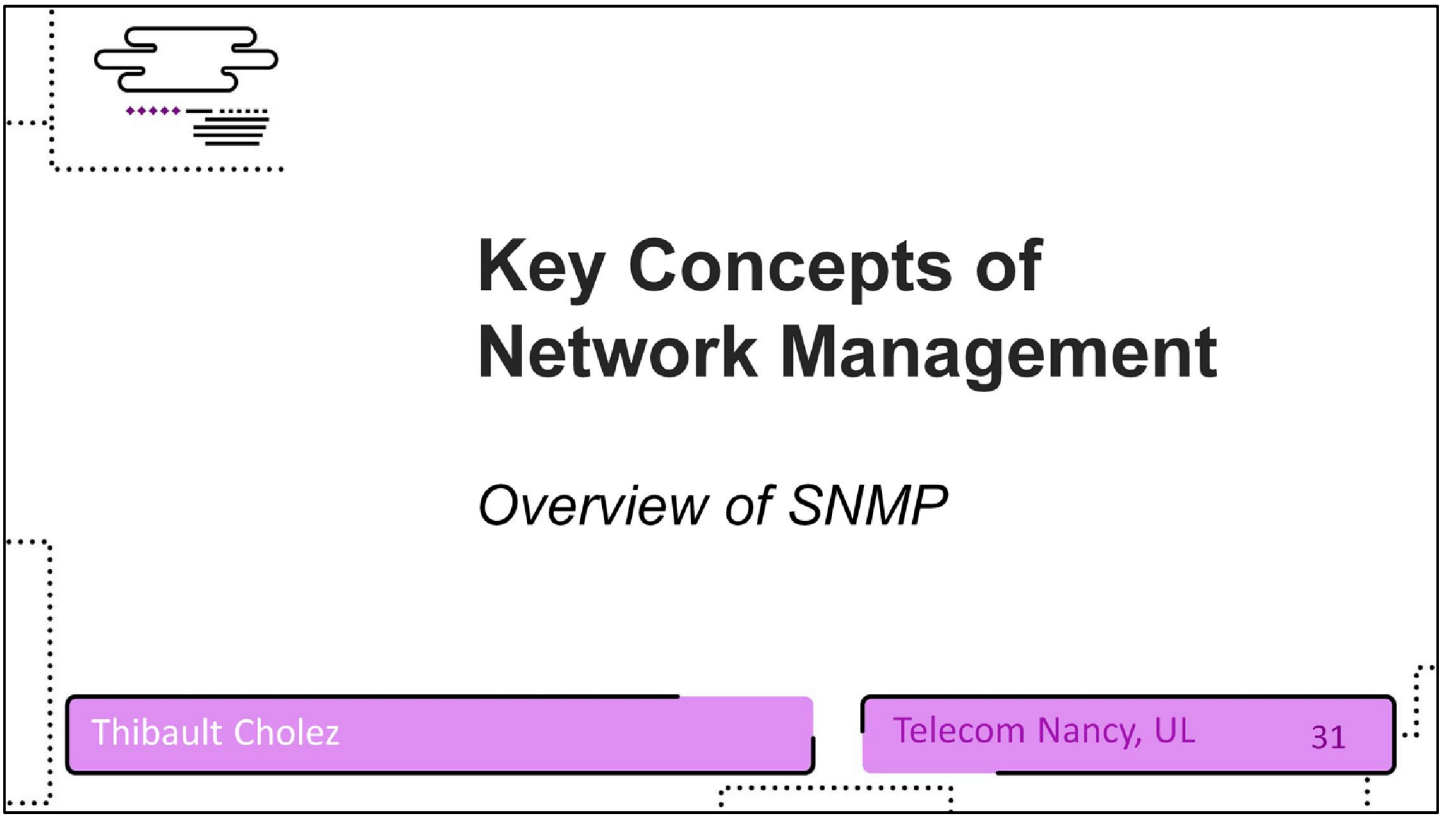
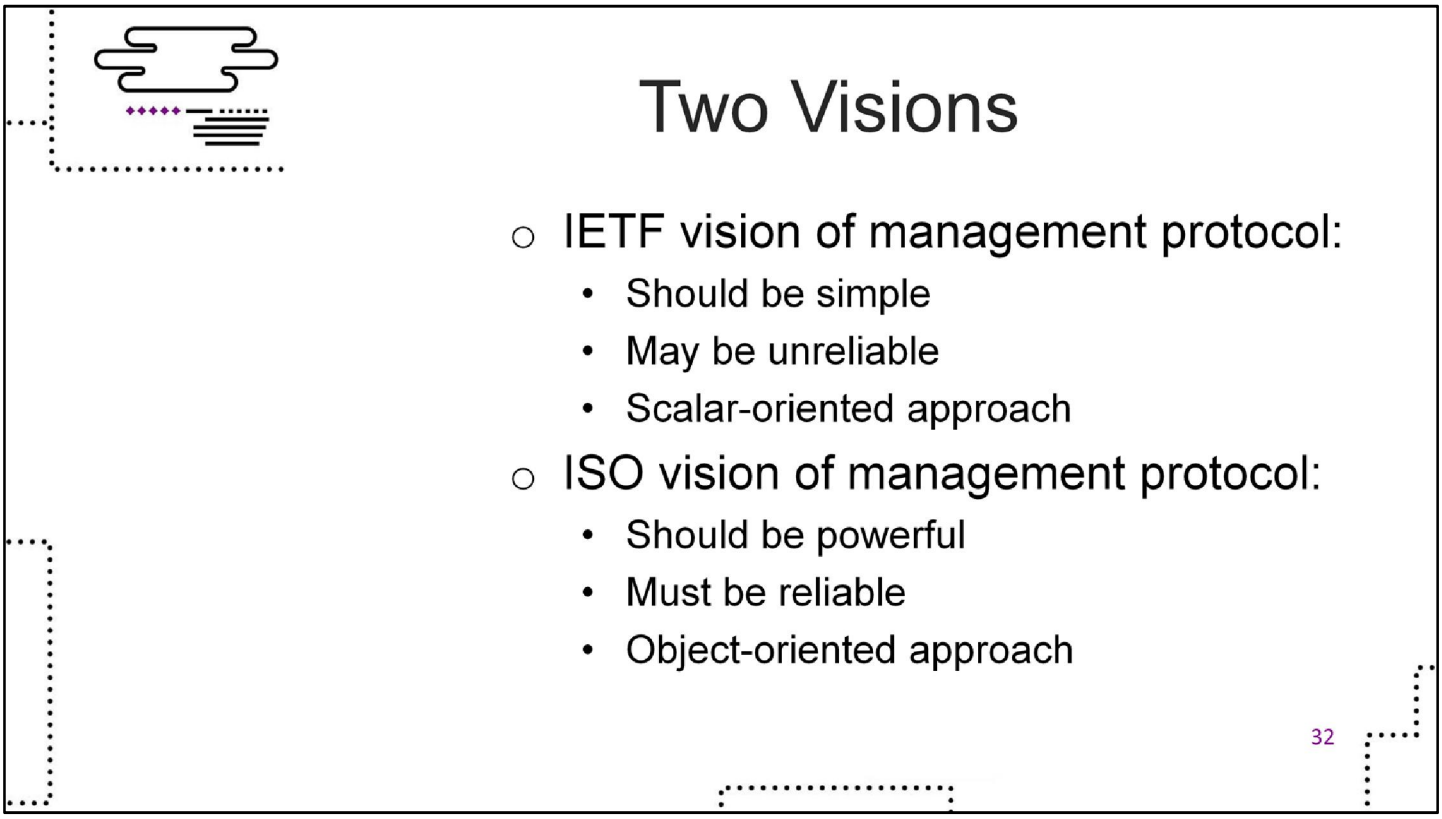
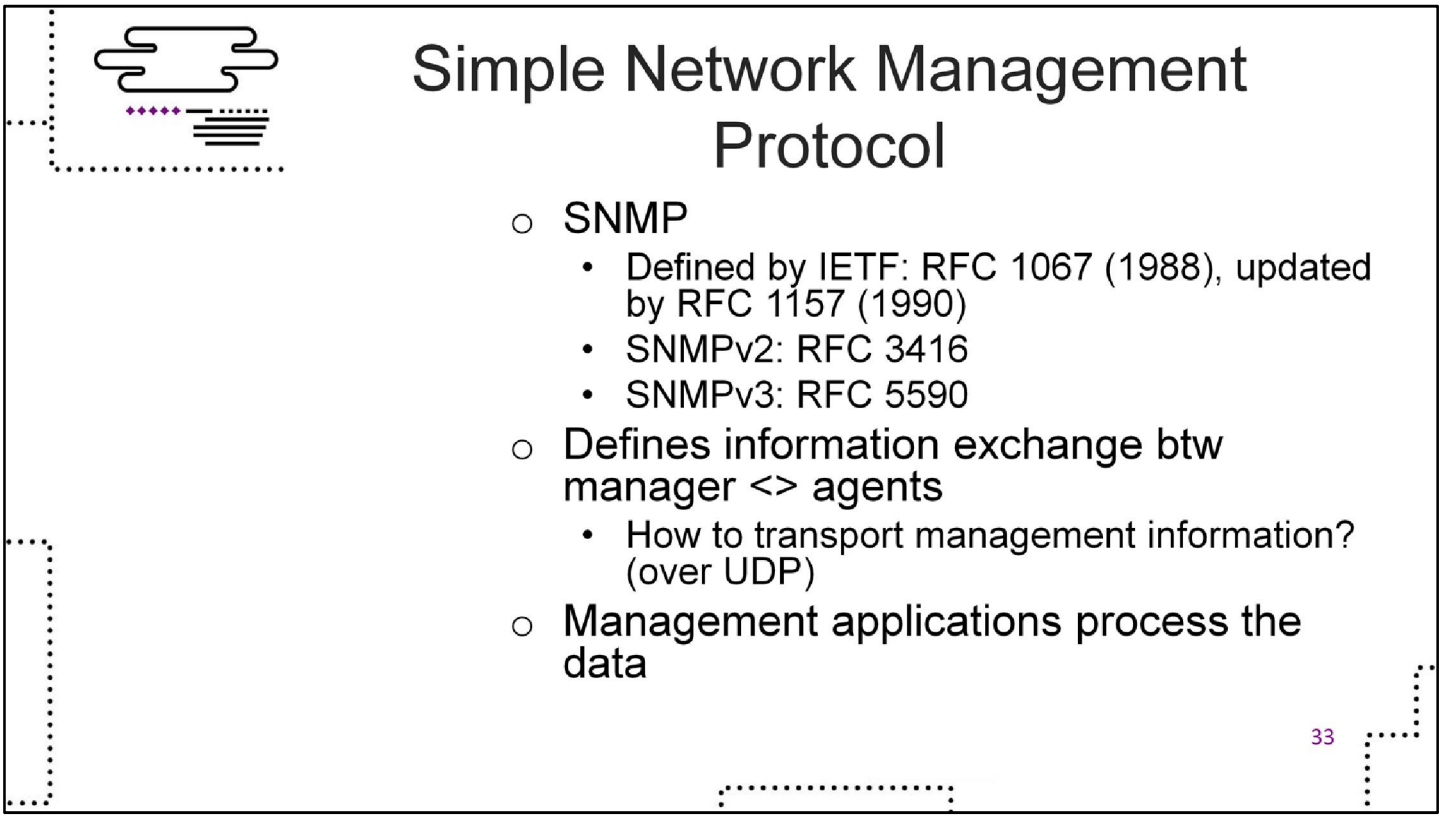


Après avoir vu comment sont structurées les informations de gestion, nous allons maintenant présenter dans cette leçon le principal protocole standardisé pour la transmission des informations de gestion, à savoir le « Simple Network Management Protocol », connu sous l'acronyme SNMP.



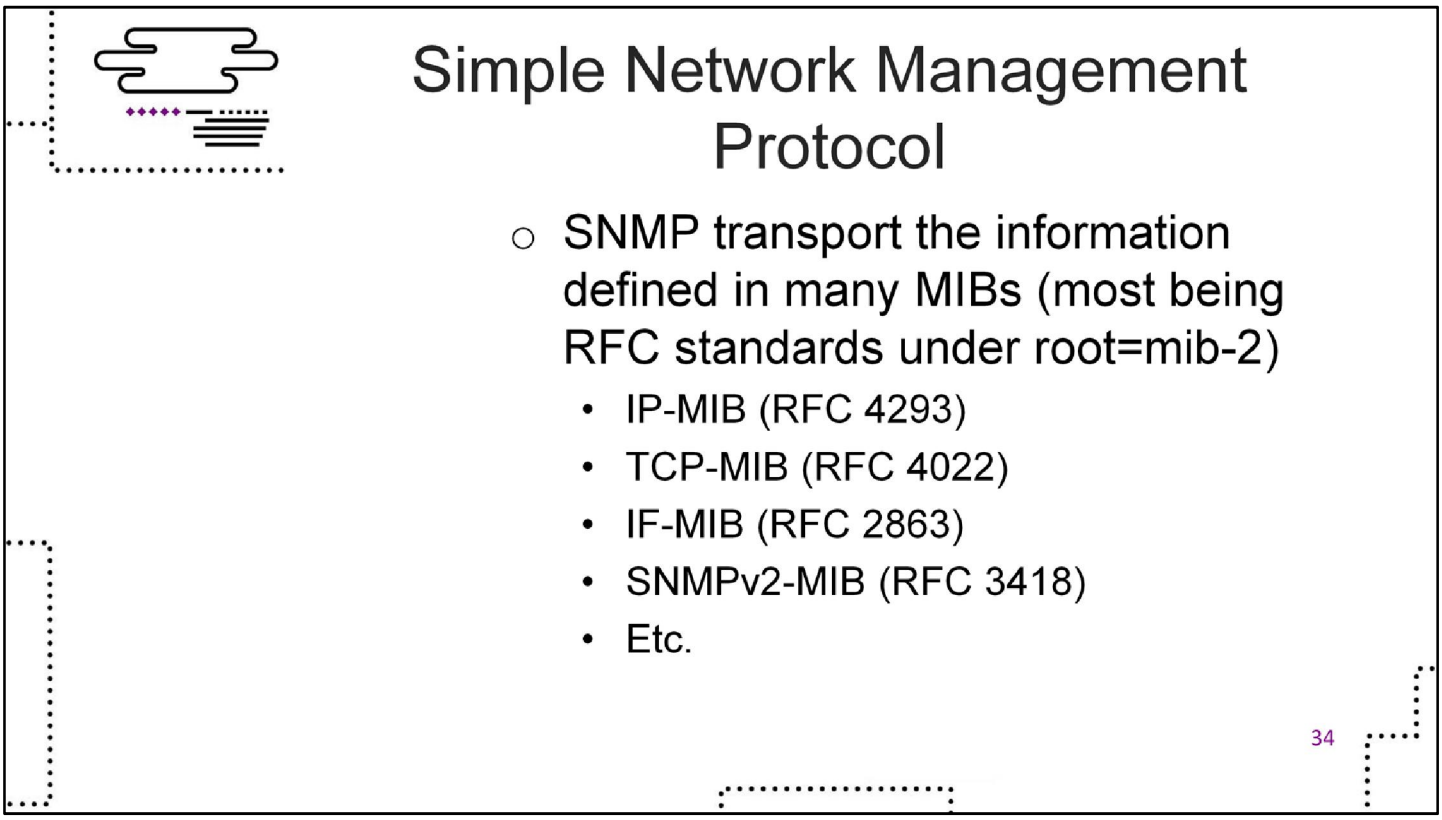
Le besoin d'un protocole de gestion de réseaux a émergé à la fin des années 80, alors que la croissance d'Internet et de la complexité des réseaux ne permettait plus une gestion efficace avec les quelques outils disponibles. Le besoin étant acté, deux visions de ce que doit être un protocole de gestion se sont affrontées. Pour l'IETF, un tel protocole doit rester simple, sans fiabilité garantie et focalisé sur les valeurs à transmettre.
A l'opposé, la vision de l'ISO est plus exigeante et considère qu'un protocole de gestion doit être puissant, fiable, et tirer pleinement partie du modèle objet.

La définition d'un protocole répondant aux critères de l'ISO étant plus long et compliqué, l'IETF a développé SNMP afin d'avoir un premier protocole fonctionnel bien qu'imparfait. Les deux organismes se sont néanmoins entendus sur la structure SMI et des MIBs pour faciliter une transition éventuelle.

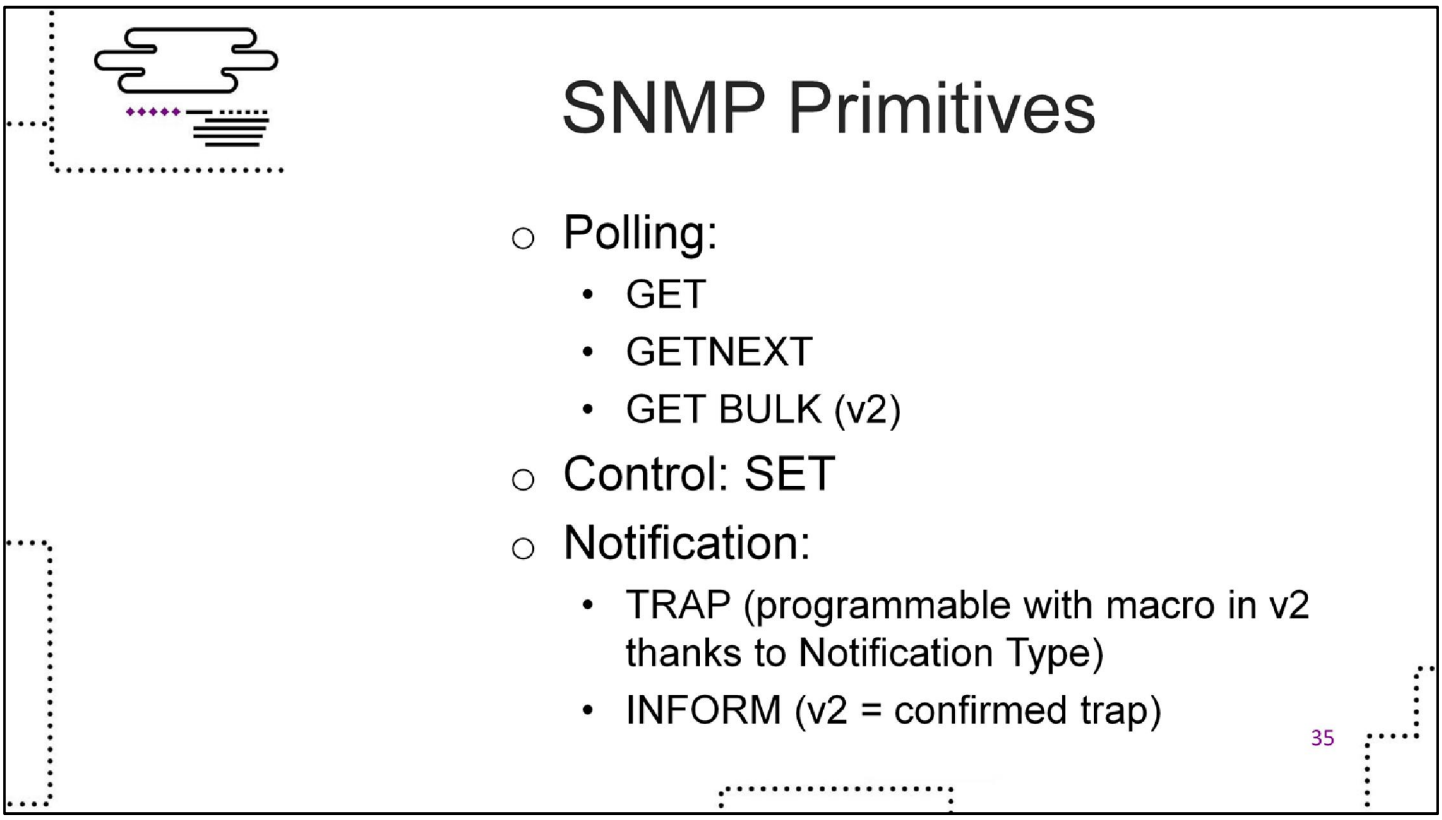


Ainsi, la première version de SNMP a été définie par l'IETF en 1988, puis mise à jour en 1990. Deux autre versions majeures de SNMP se sont ensuite succédées, améliorant et sécurisant le protocole. Je vous encourage à consulter les RFC de référence listés dans la slide.

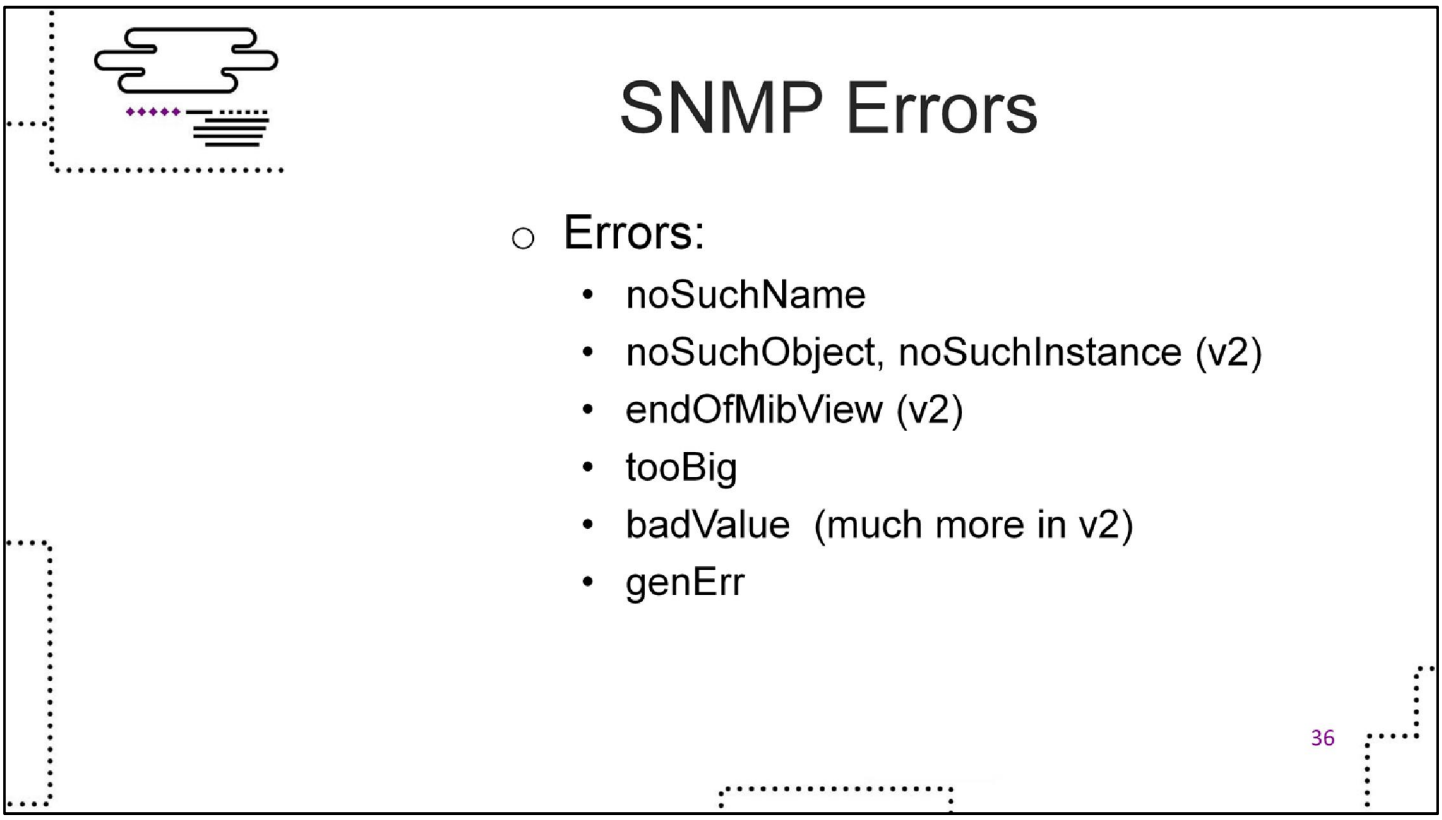
Concrètement, SNMP permet les échanges d'informations entre le gestionnaire du réseau et les agents. Il répond donc à la question: « Comment transporter les informations de gestion (au dessus d'UDP) ? ». Une fois transmises, les informations sont ensuite traitées par les diverses applications dédiées à la gestion.



SNMP transporte donc les informations contenues dans les MIBs. De nombreuses MIBs correspondant aux principaux objets réseaux ont été standardisées par des documents RFC définis au début des années 1990, comme le montrent les quelques exemple ci-contre. Bien entendu, plus un protocole est riche et complexe, plus la MIB correspondante l'est. Par exemple, la MIB gérant UDP est plus simple que celle gérant TCP.



Voyons quelles sont les opérations offertes par le protocole SNMP.
Il y a tout d'abord trois actions permettant de récupérer des valeurs :
- Get : cette opération permet de demander la valeur d'une ou plusieurs variables;
- GetNext : permet de demander le nom et la valeur de la prochaine variable de la MIB (ce qui est utile pour découvrir les attributs d'une MIB, ou itérer sur un tableau);
- GetBulk : permet de récupérer plusieurs variables consécutives en une requête. Elle permet ainsi d'améliorer les performances.
Il y a ensuite une opération de contrôle:
- Set: permet d'assigner une valeur sur un objet. Le résultat est atomique, soit l'opération aboutit, soit elle échoue.
Et enfin, deux opérations qui permettent de définir des notifications :
- Trap: demande à l'agent de signaler un événement (par exemple: linkdown, authentication failure, etc.), mais n'attend pas d'acquitement du gestionnaire, ce qui en fait une opération non fiable.
- Inform: spécifique à SNMPv2, cette opération améliore Trap en acquittant en plus la bonne réception de la notification.



Toutes ces opérations peuvent rencontrer des erreurs qui sont formalisées dans le protocole.
Si l'on prend une opération Get, les erreurs peuvent être :

- nosuchname (not a leaf) si l'objet n'existe pas ou ne peut être lu,
- toobig si le message SNMP généré est trop gros,
- genErr pour les autres erreurs.

A noter que dans le cas d'une opération GetNext : l'erreur nosuchname signifie que la fin de la MIB a été atteinte. Cette erreur devient plus explicite avec un nouveau nom en SNMPv2 et devient endOfMibView. En fait, SNMPv2 introduit beaucoup de nouveaux types d'erreurs. On distingue par exemple noSuchObject et noSuchInstance, ou encore 4 nouvelles erreurs qui détaillent des cas qui étaient auparavant regroupés sous l'erreur « noSuchName »: noAccess, notWritable, noCreation, inconsistentName.
Si l'on prend l'opération Set, l'erreur classique est badvalue en SNMPv1 et signifie que le type de la donnée renseignée est incompatible avec l'objet. Là encore, en SNMPv2, on différencie maintenant différents cas : wrongValue, wrongEncoding, wrongType, wrongLength.
La liste exhaustive des erreurs peut être trouvée dans le document RFC 3416.

SNMP PDU Structure

SNMP Message

Version	Community	SNMP PDU

GetRequest, GetNextRequest PDU and SetRequest PDU

PDU type	Request id	0	0	Variablebindings

GetResponse PDU

PDU type	Request id	error-status	error-index	Variablebindings

Trap PDU

PDU type	Enterprise	Agent-addr	Generic-Trap	Specific-trap	Time-stamp	Variablebindings

Variablebindings

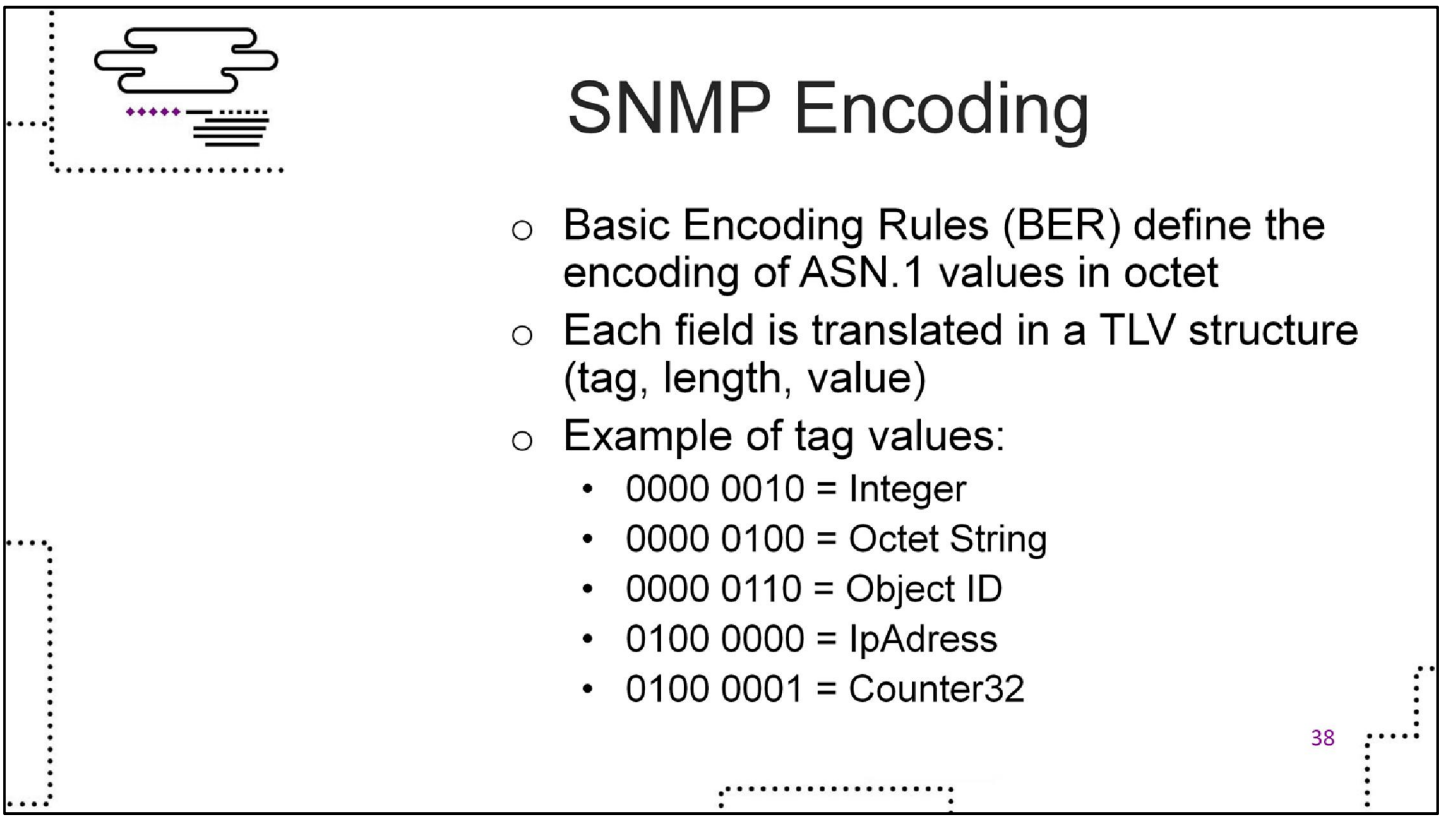
name1	value1	name2	value2	...	name n	value n

Source: http://www.linux-france.org/article/gvallee/snmp/snmp.html

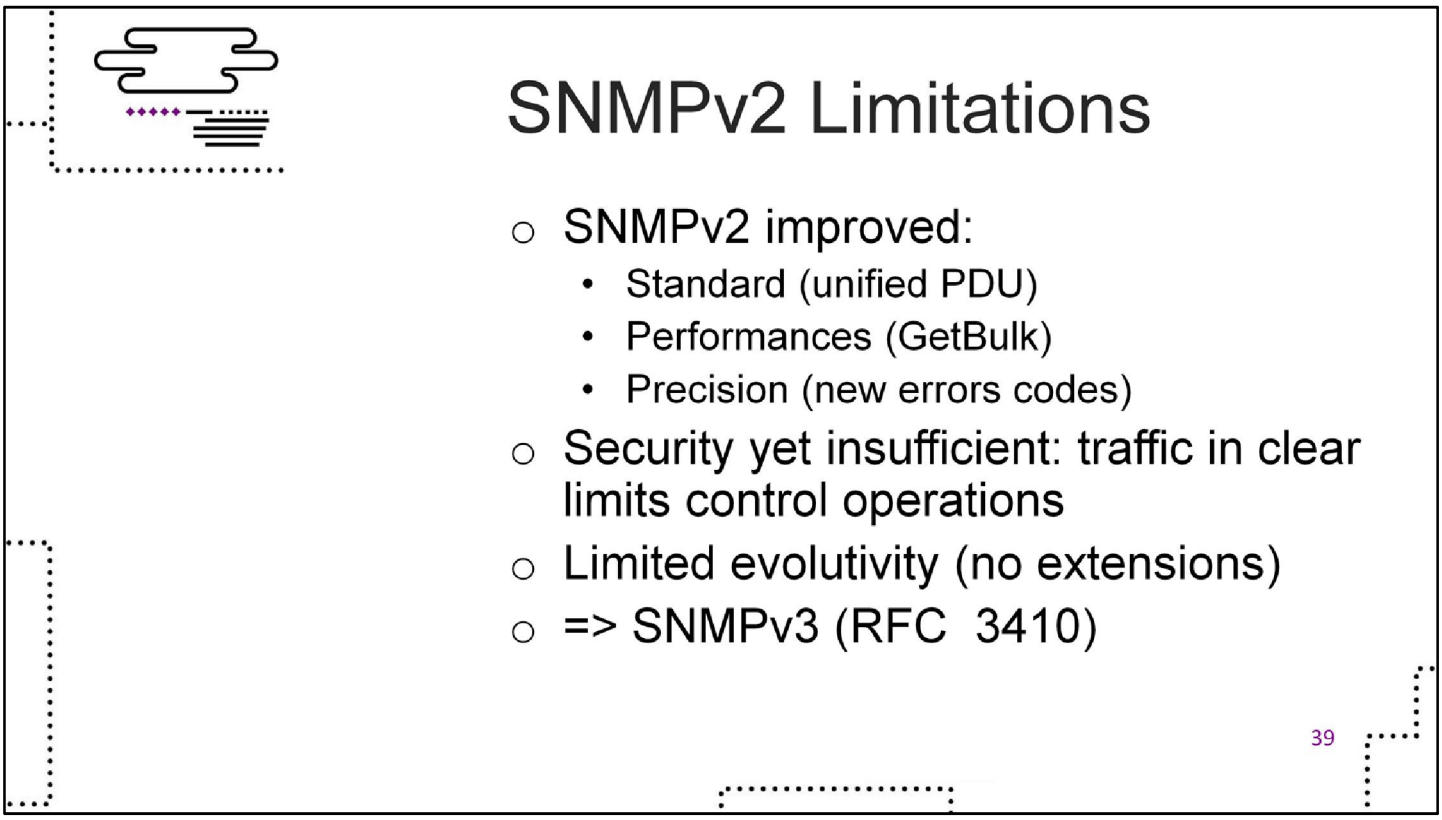
37

Voyons maintenant comment ces requêtes protocolaires s'inscrivent dans le « Protocol Data Unit » définissant la structure des messages SNMP.
Le premier champ est la « Version », qui peut prendre les trois version majeures du protocole: SNMPv1, SNMPv2 ou SNMPv3.
Le champ « Community » est une chaîne de caractère et équivaut à un mot de passe.
Le reste du PDU de SNMP peut prendre plusieurs formats selon qu'il s'agit d'une requête, d'une réponse ou d'une notification.

Le champ « PDU type » est le type de la requête parmi les primitives que nous avons vues: get, get-next, set, etc.
« Request ID » est l'identifiant aléatoire de la requête qui permet de lier les requêtes et les réponses correspondantes.
« Variable bindings » contient des couples « nom de variable et valeur correspondante », dans le cas d'une requête get, les valeurs sont nulles.
Dans le cas des réponses, « error-status » contient les codes des types d'erreurs vus précédemment et « error-index » précise le numéro de la première variable en erreur.
Ces deux champs sont nuls dans le cas d'une requête.



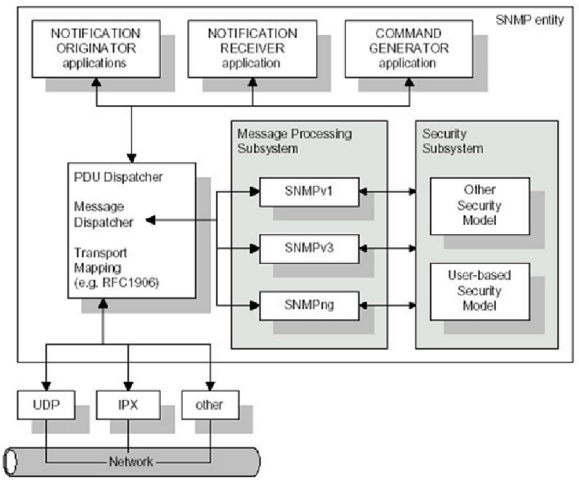
Les informations contenues dans les messages SNMP suivent des règles d'encodage simples avant d'être transmises.
Chaque champ est ainsi encodé dans une structure TLV, qui précise donc le Type (T), la longueur (L), et la Valeur (V) de l'information.
On illustre ici quelques exemple de tags correspondants à des types d'objets présentés précédemment, par exemple 0000 0010 pour un entier (Integer), ou 0000 0100 pour un octet binaire (Octet String).



Nous avons déjà mentionné quelques différences entre SNMPv1 et SNMPv2 au niveau des primitives protocolaires et des erreurs.
SNMPv2 a apporté plusieurs améliorations : tout d'abord, en unifiant le format des messages puisque les traps (permettant l'envoi de notifications) suivent désormais le même format que les requêtes et les réponses. Les performances du protocole ont été améliorées, notamment avec la primitive « GetBulk » (permettant de récupérer un ensemble de valeurs), ainsi que la précision puisque de nouveaux codes d'erreurs ont été introduits.

Néanmoins, SNMPv2 pèche encore sur deux aspects qui ont été travaillés pour la troisième version du protocole. D'une part, l'évolutivité. Alors que SNMPv2 n'autorise pas l'ajout d'extensions, SNMPv3 prévoit l'ajout de fonctionnalités plus complexes pour les grandes structures.
Et surtout, SNMPv3 améliore grandement la sécurité grâce à son architecture qui assure l'authentification, le contrôle d'accès aux tables (autrement dit vérifie qui peut faire quoi et avec quel niveau de sécurité), mais également l'intégrité des données et leur chiffrement.

SNMPv3 Architecture



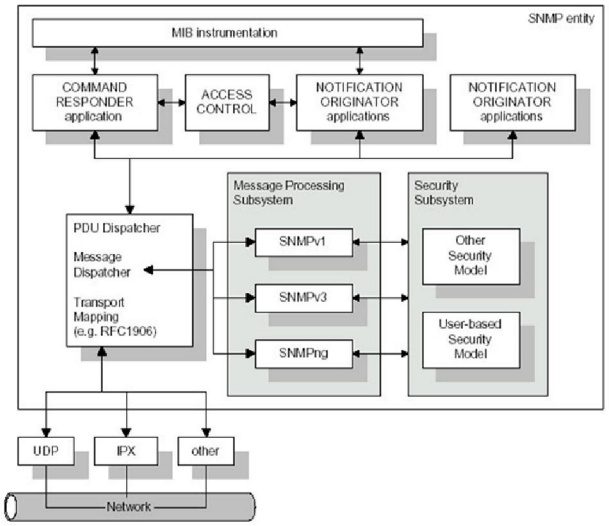
Source: RFC 2271: An Architecture for Describing SNMP Management Frameworks

40

On peut voir illustré ici l'architecture fonctionnelle d'une entité SNMPv3 pour un gestionnaire. On remarque notamment un sous-système de traitement des messages qui est rétro-compatible avec SNMPv1 et un sous-système gérant la sécurité.

Le cœur de SNMP communique avec les applications qui créent et envoient les commandes aux agents, et reçoivent en retour les notifications.

SNMPv3 Architecture



Source: RFC 2271: An Architecture for Describing SNMP Management Frameworks

41

Nous voyons ici le même model mais du point de vue d'un agent. On remarque que les applications de l'agent devant générer des réponses ou des notifications sont en relation avec les MIBs des composants instrumentés.

Pour plus d'informations, vous pouvez lire le document RFC 2271.