Plateforme de diffusion audio Net-jukeboxe

Annexes de la plateforme

Beytullah Gunduz, Iliès Selmi, Johan Blaix, Maurice Zarka, et Tony Georges

21 juin 2006

A propos de ce document

Ce document présente les annexes pour notre plateforme de diffusion audio sur internet Net-jukeboxe. Ces documents ont leur importance et montrent bien souvent le travail de fond que nous avons réalisé pour ce projet. Nous avons ici uniquement une séléction de document que nous pensons judicieux de présenter

Vous trouverez notamment des documents sur les différentes recherches que nous avons pu éffectué durant ce projet. Des présentations de technologie accompagné de petits tutoriaux pour une prise en main rapide.

Nous avons aussi décidé d'y glisser des manuels techniques concernant notre plateforme tel que le lancement du diffuseur ou encore la maintenance de la base de données.

Table des matières

Annexe A

Gestion de projet

A.1 Plan qualité

A.2 Environnement technique du projet

La première étape du projet a été définir l'environnement du projet c'est à dire définir les différentes activités reliées au projet.

Le tableau suivant énumère ces différentes activités :

Revue	Phase	Livrable
Revue de lancement	Spécification des besoins	CR de réunion, Fiche Besoin
Revue de Spécification	Spécifications fonctionnelles	documentation technique
Revue de conception	Conception Technique	documentation technique
Revue de test	Réalisation et test unitaire	Fiche Reporting
Revue de clôture	Intégration	CR de réunion, Manuels d'utilisation

A.3 Documents de qualité produits :

Compte rendus de réunion : énumère les participants de la réunion, rappelle les points abordés, puis synthèse de ceux-ci est rédigée sous une forme de liste d'actions et enfin une date de prochaine réunion est convenue.

Fiche besoin : Nous pouvons assimiler cela comme une interprétation du cahier des charges, après segmentation du projet en plusieurs parties, un responsable est désigné pour chacunes d'entre elles, des livrables lui sont attribués, la technologie de conception est fixée, une date de début et de fin de réalisation de sa partie sont convenues

Fiche de reporting : Elle est produite lors de l'achèvement d'une « partie », elle rapelle les objectifs fixées, énumère les points blocants ou sensibles éventuellement rencontrés ainsi que les plans d'actions correctifs mis en place.

Ceci est résumé par ce plan de production qualité :

Activité	Outil/Machine	Version
Management : Guide Qualité, formulaires, travail collaboratif...	Portail WEB : Wikini	
Production de la documentation	Docbook, pages Wiki	
Gestion de version	CVS	
Outil de communication Interne	Messagerie, téléconférence	
Plannification de projet	Umbrello, dbdesigner	
Plate-forme matérielle	Serveur Ubuntu Server Tomcat Openldap Mysql	Dell 1440

>

A.4 Comptes rendues de réunion

A.4.1 Samedi 7 Janvier

- Objet :
  - Mise en place de la gestion de projet pour le JukeBoxe
- Date :
  - Samedi 07 janvier 2006
- Lieu :
  - domicile de Beytullah
- Auteur :
  - Iliès Selmi
- Mise en forme :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
- Points abordés :

  - Plan d'étude pour débuter le Projet
- Compte rendu :
  - Cette réunion nous a permit de cibler les points difficiles du projet.Cela nous a permit de 8220 ;mesurer8221 ; le travail à effectuer durant les six prochain mois. Lors de cette réunion nous avons commencé à monter un plan d'action pour le projet c'est-à-dire que nous avons déclarés les premiers points à étudier et à implémenter pour le projet :
    - Streaming
    - Temps réel
    - Protocole de transport
- Liste des actions :
  - Aujourd'hui :
    - Mise en place de la gestion de projet avec MRPROJECT et autre
    - Définition des tâches
    - Estimation des facteurs critiques
    - Définition des ressources
  - Semaine prochaine :
    - Temps réel
    - Protocole de transport
- Prochaine réunion :
  - 14 janvier chez Beytullah

A.4.2 Samedi 14 Janvier

- Objet :
  - Première étude des différents points critiques du projet
- Date :
  - Samedi 14 Janvier 2006
- Lieu :
  - domicile de Beytullah
- Auteur :
  - Iliès Selmi
- Mise en forme :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
- Points abordés :
  - Plan d'étude pour débuter le Projet
- Compte rendu :
  - Cette réunion nous a permit de cibler les points difficiles du projet.Cela nous a permit de 8220 ;mesurer8221 ; le travail à effectuer durant les six prochain mois. Lors de cette réunion nous avons commencé à monter un plan d'action pour le projet c'est-à-dire que nous avons déclarés les premiers points à étudier et à implémenter pour le projet :
    - Streaming
    - Temps réel
    - Protocole de transport
- Liste des actions :
  - Aujourd'hui :
    - Mise en place de la gestion de projet avec MRPROJECT et autre
    - Définition des tâches
    - Estimation des facteurs critiques
    - Définition des ressources
  - Semaine prochaine :
    - Temps réel
    - Protocole de transport
- Prochaine réunion :
  - 14 Janvier chez Beytullah

A.4.3 Samedi 28 et Dimanche 29 Janvier

- Objet :
  - Technologie utilisé pour le projet, étude détaillé des existants, mise en place d'un premier protocole(transport+connection)
- Date :
  - >Samedi 28 et Dimanche 29 Janvier 2006
- Lieu :
  - domicile de Beytullah
- Auteur :
  - Iliès Selmi
- Mise en forme :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
- Points abordés :
  - Etude des existants(IceCeast,NetJuke)
  - Première mise en place du protocole sur papier(connection + transport)
- Compte rendu :
  - Compte rendu :
    - Cette réunion nous a permit de décrire d'une façcon assez précise du fonctionnement des existants pour le streaming audio et des différentes technologies permettant de mettre en place une radio Web.
      Elle nous a aussi permit de définir et de mettre en place sur papier le protocole entre un client et un serveur.Notre protocole c'est traduit par la connection entre un serveur et un client ;nous avons définit quelles sont les informations nécessaires qu'a besoin les 2 interlocuteurs pour communiquer comme les ports,le login/password,les droits...
- Liste des actions :
  - Aujourd'hui :
    - Présentation des existants
    - Définition mise en place du protocole
    - Technologie à utiliser pour la partie traitement des fichiers audio comme le découpage d'un fichier audio en petit paquet pour la transmission,exploitation des méta-données du fichier audio pour définir l'encodage,la durée... et tout cela avec la techno JAVA (package javax.sound)
  - Semaine prochaine :
    - Implémentation du protocole(Connection et transport)
    - Implémentation du traitement d'un fichier audio pour le streaming avec javax.sound
- Prochaine réunion :
  - 11 Février chez Beytullah

A.4.4 Samedi 11 et Dimanche 12 Février

>

- Objet :
  - Premières maquettes pour le protocole et le traitement fichier audio
- Date :
  - Samedi 11 et Dimanche 12 Février 2006
- Lieu :
  - domicile de Beytullah
- Auteur :
  - Iliès Selmi
- Mise en forme :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
- Points abordés :
  - Protocole de connection et de transport

  - Traitement des fichiers audio avec le package JAVA javax.sound
- Compte rendu :
  - Cette réunion nous a permit de d'implémenter des petites maquettes sur certains domaines du projet comme le protocole de transport et de connection sans exploiter le temps réel ainsi que d'implémenter la partie traitement d'un fichier audio afin qu'il soit envoyé au client.
- Liste des actions :
  - Aujourd'hui :
    - Implémentation des 2 maquettes.
      La première maquette pour le protocole composé de la connection entre un client et un serveur. Ici l'utilisation des sockets et des protocoles TCP et UDP ont été utilisés.On a un échange d'information comme la négociation de ports,l'envoi de login/password puis transfert d'un fichier audio sans la prise en compte du temps réels.
      La seconde maquette implémente la gestion d'un fichier audio,c'est à dire qu'elle s'appuyait sur le package javax.sound pour donner des informations sur le fichier audio comme le titre,auteur...mais aussi l'encodage,la taille du fichier, la durée... Plusieurs fonctions ont été implémentées en JAVA comme lire un fichier audio et le stocker dans un byteArray,écrire les données audio d'un byteArray dans un fichier audio,de déterminer la durée d'un fichier Ogg Vorbis,de transférer des données audio à travers un réseau.
  - Semaine prochaine :
    - Finaliser les 2 maquettes
    - Choix définitif des technologies à utiliser(format unique des fichiers audio à transférer,choix de la base de données,pour la partie vue...)
- Prochaine réunion
  - 25 Février chez Beytullah

A.4.5 Samedi 25 et Dimanche 26 Février

- Objet :
  - Présentation des maquettes et choix définitif des technologies à utiliser pour le projet
- Date :
  - Samedi 25 et Dimanche 26 Février 2006
- Lieu :
  - Domicile de Beytullah
- Auteur :
  - Iliès Selmi
- Mise en forme :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
- Points abordés :
  - Présentation des 2 maquettes et tests
  - Choix des technologies pour la partie streaming serveur/client,pour la partie vue,pour le SGBD...
- Compte rendu :
  - Cette réunion nous a permit de définir toutes les technologies à utiliser dans les différents domaines du projet.De plus on a avec nos maquettes finalisées maîtrisé des parties importantes et critiques du projet comme le protocole,le streaming audio et en partie le temps réel.
- Liste des actions :
  - Aujourd'hui :
    - Technologie à utiliser :
    - pour le protocole de connection serveur/client on utilise JAVA
    - pour le format audio unique à envoyé c'est le OGG Vorbis
    - pour le traitement du fichier audio,le streaming,le temps réels et le protocole de transport on utilise les librairies LibOgg et LibVorbis
  - Semaine prochaine :
    - Présentation du prototype pour le DEST
- Prochaine réunion :
  - 05 mars chez Beytullah

A.4.6 Dimanche 5 Mars

- Objet :
  - Répartition des tâches en vue de la présentation de la maquette
- Date :
  - Dimanche 05 Mars 2006
- Lieu :
  - domicile de Beytullah
- Auteur :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
- Excusés :
  - Maurice Zarka : problème de voiture
  - Johan Blaix n'a donc pas pu passer
- Points abordés :
  - Documentation qualité
  - Cahier des charges
  - Problematique technique
  - Gestion des données
  - Partie 'vue' de la plateforme
- Compte rendu :
  - Réecriture du cahier des charges : nous avions fixé nos objectifs globals (definition d'un canal, systeme d'abonnement...) lors d'une réunion chez moi. Maurice avait dit qu'il mettrait au propre. J'ai relu ce qu'il y a sur le wiki et il semble subsister des zones d'ombres. Maurice passe chez moi samedi, j'en profiterais pour qu'on fige ce document
  - Documentation qualité : Johan m'avait dis qu'il connaissait bien sans que nous y prettions plus d'attention, nous devons focaliser plus d'energie sur ce point pour lundi.
  - comprehension du format OggVorbis et méchanisme de transport du flux avec pourquoi pas un petit bou de code pour l'illustrer : beytu,ilies
    - nous considerons la comprehension du format OggVorbis acquis. Il faut à présent mettre ca sur papier de façon 'human readable'.
    - je dois voir le prof pour lui détaillé notre methode pour transporter le flux histoire d'avoir une validation de ce point.
    - j'ai fait une maquette fonctionnelle permettant d'illustrer ces deux points.
  - base de donnée : schéma de base de donnée et definition d'une interface à fournir aux autre services (notamment la partie vue/visuel (tomcat) et le serveur de diffusion : Maurice
  - ldap : schema et definition de l'interface pour que d'autre services puisse utiliser LDAP de façon transparente : Johan
  - Tony devait s'interesser à Tomcat (installation,mise en oeuvre, exploitation) et comment faire un transcodage du son
- Problème identifié :
  - Le problème majeur identifié ce soir la, est que beaucoup dérivent de leurs objectifs pour aller fleurter avec d'autres aspects du projet (moi y compris). Il faut donc cadrer les objectifs de chacun et s'y tenir.
- Liste des actions
  - état d'avancement du projet,
  - présentation des objectifs atteints
  - mise au clair de ceux à atteindre pour lundi dans l'immediat
    - Mise au propre de mes brouillons (Beytullah) et rediger de façon clair (non confu comme j'ai l'habitude de faire) les spec OggVorbis ainsi que AlgorithmeUtilise ? qui devrait plutôt s'appeller 'principe de fonctionnement global' en y ajoutant la notion d'identification de l'utilisateur. Je vais ecrire aussi quelque lignes sur la petite étude de code que j'ai faite concernant IceCast ?. Je vais mettre au propore le schema global que j'avais fais pour présenter l'articulation des différentes téchnologies utilisé.
    - Ilies va rediger une présentation de RTP,RTSP et JMF
  - spécification de l'interface de base de donnée, Maurice devrais poster le 2006/03/09 de quoi satisfaire ce problème, si mes souvenir son bons le schema de base de donnée doit être en phase

final.
  - spécification de l'interface LDAP manquante. Johan a travaillé sur l'installation LDAP et de son interfacage avec JAVA. Mais aucun schéma de stockage n'a été présenté.
  - Tony travail sur la partie Tomcat. Mise en oeuvre et exploitation. Il devait aussi permettre le transcodage du son : anything2OggVorbis en gros

A.4.7 Jeudi 9 Mars

- Objet :
  - Répartition des tâches en vue de la présentation de la maquette
- Date :
  - Jeudi 09 Mars 2006
- Lieu :
  - Domicile de Beytullah
- Auteur :
  - Johan Blaix,
  - Beytullah Gunduz
- Mise en forme :
  - Beytullah Gunduz
- Participants :
  - Maurice Zarka
  - Beytullah Gunduz
  - Iliès Selmi
  - Johan Blaix
- Points abordés :
  - Bilan des tâches déjà éffectués
- Compte rendu :
  - La réunion a permi d'attribuer des missions aux collaborateurs du projet
- Liste des actions :
  - Johan Blaix : Interface LDAP, plan qualité
  - Maurice Zarka : schéma de la base, interface pour la base
  - Beytullah Gunduz : technologie de streaming, format OGG/Vorbis et utilisation avec JAVA
  - Ilies Selmi : technologie de streaming
- Prochaine réunion :
  - 11/03/2006 au domicile de Tony Georges

A.4.8 Mardi 21 Mars

- Objet
  - État d'avancement et prévision des tâches
- Date
  - Mardi 21 Mars 2006
- Lieu
  - CUEFA
- Auteur
  - Maurice Zarka
- Mise en forme
  - Beytullah Gunduz
- Participants :
  - Maurice Zarka
  - Beytullah Gunduz
  - Iliès Selmi
  - Johan Blaix
  - Tony Georges
- Points abordés :
  - Réécriture du cahier des charges.
  - Maquette IHM.
  - Manuel d'utilisation.
  - Cas d'utilisation.

  - diagramme de classe.
  - Architecture évolutive.
  - encodage OGG VORBIS
  - Load balancing
- Compte rendu :
  - Cette réunion a eu lieu après une intervention de Monsieur Aouane. Il nous à demandé explicitement un retour sur différents travaux, chose que nous aurions du mettre en place déjà pour la présentation de la semaine dernière face à nos clients :
    - Cas d'utilisation.
    - Diagramme de classe.
    - Diagramme de séquence.
    - Diagramme de contrainte

    Au cour de cette réunion, nous avons donc fait le point sur l'état d'avancement de ses tâches et planifié comment lui rendre un travail complet pour le lundi 12 avril.
- Liste des actions :
  - Aujourd'hui :
    - Finir la réécriture du cahier des charges
    - Faire une Maquette IHM (qqc de très simple sous word par exemple)
    - Faire un manuelle d'utilisation en relation avec l'IHM
    - Rajouter le cas d'utilisation Visiteur.
    - Validation des cas d'utilisations pour passer à la suite.
    - documentation des cas d'utilisations.
    - Décrire, spécifier et implémenter les différents diagramme de classe à faire.
  - Semaine prochaine :
    - Architecture du « load balancing ».
    - Architecture de récupération des pages perdu.
    - Réunion pour mettre a plat la suite des tâche à effectuer.
- Prochaine réunion :
  - Mardi 28 Mars au CUEFA

A.4.9 Samedi 15 Avril

- Objet :
  - base de donnée
- Date :
  - Samedi 15 Avril 2006
- Lieu :
  - Domicile de Beytullah
- Auteur :
  - Beytullah Gunduz
- Participants :
  - Maurice Zarka
  - Beytullah Gunduz
- Points abordés :
  - Architecture de la base de donnée et interface.
- Compte rendus :
  - Cette réunion a permit d'avancé l'architecture de la base de donnée et mieux comprendre les services qui vont être rendu au différentes unités du projet (partie vue avec Tomcat et le serveur de diffusion.
- Prochaine réunion :
  - le 18 Avril au domicile de Beytullah

A.4.10 Mardi 18 Avril

- Objet :
  - base de donnée
- Date :
  - Mardi 18 Avril 2006
- Lieu :

  - domicile de Beytullah
- Auteur :
  - Beytullah Gunduz
- Participants :
  - Maurice Zarka
  - Beytullah Gunduz
- Points abordés :
  - Architecture de la base de donnée et interface.
- Compte rendu :
  - Dans le même esprit que la reunion précedente, nous avons avancé l'architecture de la base ainsi que discuté sur la problèmatique de l'interface
- Prochaine réunion :
  - Le 20 au domicile de Beytullah

A.4.11 Jeudi 20 Avril

- Objet :
  - état d'avancement et prévision des tâches
- Date :
  - Jeudi 20 Avril 2006
- Lieu :
  - domicile de Beytullah
- Auteur :
  - Beytullah Gunduz
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
  - Johan Blaix
- Points abordés :
  - Base de donnée
  - Interface LDAP
  - Diagramme de séquences
- Compte rendu :
  - Nous avons longuement discuter de l'etat des lieux du projet surtout concernant les interfaces MySQL et LDAP qui deviennent bloquantes. Tony ne peut pas avancé du coté de Tomcat sans ces interfaces cruciales. Nous avons essayé de comprendre les raison du retard et prendre des mesures si nécessaire
- Liste des actions :
  - Johan s'engage à réaliser la spécification d'LDAP.
  - Beytullah finalisera les diagrammes de séquence cette même semaine.
  - Ilies va soutenir Maurice dans l'avancement de l'interface MySQL.

A.4.12 Jeudi 25 mai

- Objet :
  - Discution autour de la BD
- Date :
  - Jeudi 25 mai 2006
- Lieu :
  - Domicile de Beytullah
- Auteur :
  - Maurice Zarka
- Mise en forme :
  - Maurice Zarka
- Participants :
  - Beytullah Gunduz
  - Iliès Selmi
  - Johan Blaix
  - Maurice Zarka

  - Tony Georges
- Points abordés :
  - Résoudre des problèmes de requêtages
- Compte rendu :
  - Nous avons décidé de rajouter certaines tables dans la BD pour pallier à des problèmes de requêtages.
- Liste des actions :
  - Aujourd'hui :
    - Définition des tables supplèmentaires
  - Semaine prochaine :
    - Création des tables
- Prochaine réunion :
  - 1 Juin 2006

A.5 Calendrier du projet

>

A.5.1 Beytullah Gunduz

- Janvier 2005 :
  - Prise de connaissance du sujet,
  - première étude pour dégager les différents points
  - Premiers planning prévisionnels
  - Choix général de la plateforme : Java
- Décembre 2006
  - Etude des différente solution audio connue telle que le Vorbis, MP3 et les solution de transport Matrsoka et OGG
  - Exploitation du son avec Java et Tony nous montre jsressources.org, un site très interressant sur l'exploitation du son avec Java
- Février 2006
  - Etude de l'éxistant : IceCast, Savonet, Netjuke...
  - Ecriture des algorithmes principaux et prototypage rapide d'un client/serveur pour la faisabilité
  - Découverte de la librairie jOGG et jVorbis, étude de ces librairies et prise en main
  - Arrivé de deux nouveaux membres dans le projet
- Mars 2006
  - Participation à la réécriture du cahier des charge
  - Travail sur les cas d'utilisation
  - Prototype d'un système de diffusion simple pour tester différentes architecture
  - Prototype de diffusion audio opérationnel avec le OGG
  - Prototype de connexion,négociation entre client et serveur opérationnel/
  - Fin de la phase de recherche, début de spécifications avec réalisation des diagrammes de classes pour les soumettres au groupe
  - Mise en place de notre serveur Subversion fin Mars
- Avril 2006
  - Fin de la phase de spécification,
  - Définition des algorithmes utilisé,
  - Système d'ordonnanceur/diffusion opérationnel
  - Travail sur la base de donnée pour débloquer la situation
- Mai 2006
  - Apparition de la notion d'allocateur et de centre de sécurité.
  - Phase de développement du client/serveur
  - Découverte de Log4j grâce à Tony et implémentation dans le client et le serveur
  - Réalisation d'un connecteur simulant la base de donnée en utilisant un système de fichier.
  - Travail avec Iliès pour savoir si il manquait des fonction dans l'interface MySQL vis à vis des besoins du serveur
  - Proposition d'utiliser mp32ogg ou soundconvertor avec Java pour convertir les fichier MP3 en OGG.

- Juin 2006
  - Finalisation du client/serveur
  - Développement d'une servlet pour controler le serveur et pour montrer le fonctionnement de l'applet.
  - Travail avec Tony sur l'intégration du l'applet avec Javascript en utilisant de l'AJAH
  - Prise en charge de la documentation du projet au format Docbook. Mon travail a été de fédérer toute les documentations qui étaient dans des formats et encodages différents.

A.5.2 Iliès Selmi

- Décembre 2006 :
  - Mise en place de la gestion de projet pour le JukeBoxe
  - Cerner les points faciles et difficiles du projet
  - Premières prévisions et fixation des premières échéances
- Janvier 2006 à :
  - Etude du temps réel et des protocoles de transports(RTP/RSTP)
- février 2006
  - Etude détaillée des existants(IceCast...)
  - mise en place simple d'un protocole simple de connection et de transport (définition des échanges de messages entre 1 serveur et 1 client)
  - Connection TCP/UDP entre client/Serveur
  - Etude du package 8220 ;javax.sound8221 ; pour le découpage d'un fichier audio
- Mars 2006
  - Choix du type d'encodage pour le transport(Ogg Vorbis) du flux audio
  - Spécification Ogg et spécification Vorbis
  - Début Spécification du JukeBoxe(diagramme classe,diagramme séquence...)
- Avril 2006
  - Fin Spécification du JukeBoxe(diagramme classe,diagramme séquence...)
  - Prototype de l'ordonnanceur(découpage paquets-opérationnelle mi-mai)
  - Prototype IHM en HTML
- Mai 2006
  - Prise en main de linterface Base de données
  - Interface base de données finalisé(des fonctions arrivent maintenant au compte goute)
- Juin 2006
  - Développement IHM en JSP pour toute la partie historique du JukeBoxe. Le module est près pour une intégration dans l'application final
  - Finalisation Interface Base de données
  - Evolution JukeBoxe Log4J
  - Evolution JukeBoxe Mp32Ogg

A.5.3 Johan Blaix

- Mai 2006
  - Bibliographie sur JNDI et LDAP et authentification LDAP
  - Déploiement de LDAP
  - Début du dévelopement de l'interface LDAP
- Juin 2006
  - Dévelopement de l'interface LDAP
  - Rédaction de la documentation client

A.5.4 Maurice Zarka

- Mars 2006
  - Recherche personnelle sur la base de donnée
  - Réécriture du cahier des charges
  - Ecriture de cas d'utilisation
- Avril 2006
  - Finalisation de la réecriture du cahier des charges et cas d'utilisation
  - Développement sur la base de donnée

  - Prototype IHM en HTML
- Mai 2006
  - Recherche personnelle pour la réalisation d'une interface en Java avec la base de donnée
  - Recherche personnelle sur Hibernate + ajustement base de donnée
  - Début d'implementation d'Hibernate
- Juin 2006
  - Implémentation Hibernate et test unitaire
  - Rédaction documentation client

A.5.5 Tony George

- Décembre 2005
  - réflection sur l'architecture globale du serveur : propositions d'utiliser Java, Tomcat
- Janvier 2006
  - écriture d'un client Java lisant un flux audio sur le système audio de la machine divers tests de conversion audio
- Fevrier 2006
  - recherches sur OGG pour faire de l'encodage via une api Java
- Mars 2006
  - Travail sur les cas d'utilisation
  - Travail sur les diagrammes de classe du client et du serveur
- Avril 2006
  - Travail sur les diagrammes de classe du client et du serveur
- Mai 2006
  - recherches perso sur HTML et CSS2, choix frame / tableaux / interraction Applet
  - IHM HTML
  - Début de l'IHM JSP + intégration BD et LDAP
- Juin 2006
  - Fin de l'IHM JSP + intégration BD et LDAP
  - IHM en classe Java + intégration LDAP + intérraction AJAH
  - intégration Applet + doc architecture IHM + manuel utilisateur
  - Rédaction du manuel utilisateur.

>

Annexe B

Exploitation de la plateforme Net-jukeboxe

B.1 Implémentation du serveur OpenLDAP

B.1.1 Introduction

OpenLDAP est un projet libre de serveur d'annuaire conforme à la norme LDAP 3, il dérive de l'implémentation mis au point par l'université de Michigan.

Il est composé des éléments suivants

- Le serveur LDAP : slapd
- La passerelle LDAP vers X500 : lapd
- Le serveur de réplication : slurp
- Des outils d'administration

B.1.2 Installation

B.1.2.1 Paquets à installer

Le serveur mis en place est une distribution Ubuntu Server avec un noyau 2.6.xxx, par conséquent openLdap est packagé.

Il suffit donc de taper les lignes suivantes :

```
# apt-get install ldap-server ldap-clientpara

Ce qui execute en réalité :para

# apt-get install slapd ldap-utilspara

Enter your DNS domain : cuefa.inpg.frpara
Enter the name of organisation : cuefa.inpg.compara
Admin password : xxxxpara
```

B.1.2.2 Répertoires installés

L'installation génére un certain nombre de scripts de configuration et va créer les répertoires suivants :

- /etc/ldap : répertoire des fichiers de configuration
- /var/lib/ldap : répertoire par défaut où va être stocké l'annuaire, à l'aide de la base de données de LDAP, la bdb (Berkeley DataBase)
- /usr/share/ldap : répertoire contenant les documentations et les outils pour migrer par exemple un système NIS (yellow page) existant dans l'annuaire LDAP.

Les traditionnelles pages de manuel et les commandes LDAP sont respectivement installées dans /usr/man et /usr/bin.

B.1.3 Configuration

La configuration d'un serveur LDAP est la phase préliminaire à toute mise en oeuvre de celui-ci. Pour OpenLDAP, cela consiste a éditer un fichier : /etc/ldap/slapd.conf.

Afin de pallier a toute erreur de manipulation et pouvoir revenir à la configuration de base il est vivement recommandé de faire une copie de ce fichier :

```
# cp /etc/ldap/slapd.conf /etc/ldap/slapd.conf.old
```

Le fichier slapd.conf est constitué de trois types d'informations de configuration :
– global
– spécifique au backend
– spécifique à la base de données.

L'information globale est spécifiée en premier, suivie par l'information associée à un type de backend particulier, qui est elle-même suivie par l'information associée avec une instance de base de données particulière.

Les lignes blanches et les commentaires commençant par le caractère « # »sont ignorés. Si une ligne commence avec un espace, elle est considérée comme la continuation de la ligne précédente.

B.1.3.1 Gestion des schémas

Un objet LDAP est décrit par des attributs et des classes d'objets. Un schéma regroupe des attributs et des classes d'objet que pourront posséder les objets de l'annuaire. Il précise pour chaque attribut et chaque classe les contraintes, les héritages, les syntaxes, les règles de comparaison,...

Les schémas sont des fichiers textes se situant dans /etc/ldap/schema/. Les schémas nécessaires au bon fonctionnement du serveur sont inclus dans le fichier slapd.conf.

La ligne suivante :
Schema and objectClass definitions
include /etc/ldap/schema/core.schema
va permettre de spécifier quel shéma l'annuaire doit mettre en ouevre. Ici ce fichier décrit la base de tous les annuaires LDAP. Il contient les définitions des attributs et classes d'objets standards. Ce schéma est obligatoir, c'est le minimum attendu par un serveur LDAP.

Voici un extrait du fichier core.schema avec la class 'person'
objectclass (2.5.6.6 NAME 'person'
DESC 'RFC2256 : a person'
SUP top STRUCTURAL
MUST (sn $ cn)
MAY (userPassword $ telephoneNumber $ seeAlso $ description))
Explication :
MUST correspond aux attributs obligatoires et MAY à ceux facultatifs
ObjectClass est le nom de la classe qui descend elle même de la classe top
sn correspond dans notre cas au prénom (généralement sn : surname correspond au nom)
cn correspond dans cas au login (généralement cn : common name correspond au nom + prénom)

Pour créer notre annuaire avec les attributs nécessaire il a fallu rajouter des schémas supplémentaires tels que :
Schema and objectClass definitions
include /etc/ldap/schema/core.schema
include /etc/ldap/schema/cosine.schema
include /etc/ldap/schema/inetorgperson.schema
Ces shémas permettent d'avoir une description beaucoup plus étoffée pour les objets de l'annuaire.

B.1.3.2 Gestion du serveur

Le serveur, lors de son démarrage, essaye d'écrire dans deux fichiers en particuliers, s'il échoue dans l'écriture, ceci n'empêche pas son fonctionnement mais il vaut mieux que ces fichiers soient présents en cas de problèmes ultérieurs.Les deux fichiers sont les suivants :

– /var/run/slapd/slapd.pid, il contient le numéro du premier processus UNIX sous lequel le serveur tourne
– /var/run/slapd.args, il contient la liste des arguments avec lesquels a été lancé le serveur.

B.1.3.3 Gestion de la base de données

La gestion de la base de données va permettre de préciser plusieurs choses :
– Le nom (suffixe) de la base de données
– L'identité (DN) du gestionnaire de base
– L'endroit où seront stockés les différents fichiers représentant les données de l'annuaire

B.1.3.4 Le suffixe de la base de données

C'est en quelque sorte l'identifiant général de la base de données. Toutes les entrées de la base contiendront ce suffixe.

Il est défini ainsi : « dc=cuefa,dc=inpg,dc=fr »

B.1.3.5 Le gestionnaire de la base

C'est une entrée spéciale de la base. Elle peut être virtuelle. Elle est gérée par la ligne rootdn. La solution la plus simple consiste à utiliser une forme en fonction du choix du suffixe.

rootdn "cn=admin,dc=cuefa,dc=inpg,dc=fr"

Le gestionnaire de la base doit se connecter à l'aide d'un mot de passe, celui-ci est décrit par la ligne suivante :

rootpw motdepasse

Le répertoire de stockage des données de l'annuaire est indiqué par la ligne suivante :

directory « /var/lib/ldap »

B.1.3.6 Gestion des contrôles d'accès

L'accès aux entrées et attributs slapd est contrôlé par la directive de configuration d'accès. La forme générale d'une ligne d'accès est la suivante :

<access directive : := access to <what>
[by <who> <access><control>]
<what> : :=* | [dn.<target style>]=<regex>]
[filter=<ldapfilter>] [attrs=<attrlist>]
<who> : := [* | anonymous | users | self | [dn.<subject style>]
<what> : définit les entrées et/ou les attributs sur lesquelles les règles s 'appliquent
<who> : définit quelles identités ont accès
<access> : définit le type d'accès.

Dans notre cas :

access to attrs=userPassword
by dn="cn=admin,dc=cuefa,dc=inpg,dc=fr" write
by anonymous auth
by self write
by * none

Cette acl permet à l'administrateur de modifier n'importe quel mot de passe, à l'anonyme de s'authentifier, à l'utilisateur authentifié de changer son mot de passe

access to *
by dn="cn=admin,dc=cuefa,dc=inpg,dc=fr" write
by dn="cn=admin,dc=cuefa,dc=inpg,dc=fr" read
by users read
by * none

Accès en ecriture et en lecture pour tout l'annuaire pour l'administrateur

Accès en lecture pour les utilisateurs authentifiés

B.1.4 Lancement

Pour démarrer le serveur il suffit de taper la commande suivante :
/etc/init.d/slapd start

B.1.5 Ajout du schéma

Pour ajouter des données au serveur LDAP il faut fournir un fichier au format ldif, le format est un format texte facilement lisible au contraire du format interne de l'annuaire.

Voici notre fichier de base :

```
dn: dc=cuefa,dc=inpg,dc=fr
objectClass: top
dn: o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: organization
objectClass: top
o: netjuke

dn: ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: organizationalUnit
objectClass: top
ou: people

dn: uid=administrateur,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: administrateur
employeeNumber: 1
sn: admin
uid: administrateur
userPassword:: xxxxxxxx

dn: uid=user1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: user1
employeeNumber: 2
givenName: userName
l: grenoble
mail: user@user.com
postalAddress: 1 rue des fruits
postalCode: 38900
sn: userFisrtName
st: france
telephoneNumber: 12345678
uid: user1
userPassword:: xxxxxx

dn: uid=userOff1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: userOff1
employeeNumber: 2
```

```
givenName: userName
l: grenoble
mail: userOff@userOff.com
postalAddress: 1 rue des fruits
postalCode: 38900
sn: userOffFisrtName
st: france
telephoneNumber: 12345678
uid: userOff1
userPassword:: xxxxxx

dn: uid=tv1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: tv1
employeeNumber: 2
givenName: tvName
l: grenoble
mail: tv@tv.com
postalAddress: 1 rue des fruits
postalCode: 38900
sn: tvFisrtName
st: france
telephoneNumber: 12345678
uid: user1
userPassword:: xxxxx

dn: uid=tvTemp1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: tv1
employeeNumber: 2
givenName: tvTempName
l: grenoble
mail: tvTemp@tvTemp.com
postalAddress: 1 rue des fruits
postalCode: 38900
sn: tvTempFisrtName
st: france
telephoneNumber: 12345678
uid: user1
userPassword:: xxxxx

dn: uid=tvOff1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: person
objectClass: top
cn: tvOff1
employeeNumber: 2
givenName: tvTempName
l: grenoble
mail: tvTempOff@tvTemp.com
postalAddress: 1 rue des fruits
postalCode: 38900
```

```
sn: tvTempOffFisrtName
st: france
telephoneNumber: 12345678
uid: user1
userPassword:: xxxxx

dn: ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: organizationalUnit
objectClass: top
ou: groups

dn: cn=NetJukeAdmin,ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: groupOfUniqueNames
objectClass: top
cn: NetJukeAdmin
uniqueMember: uid=administrateur,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr

dn: cn=NetJukeTv,ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: groupOfUniqueNames
objectClass: top
cn: NetJukeTv
uniqueMember: uid=tv1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr

dn: cn=NetJukeUser,ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: groupOfUniqueNames
objectClass: top
cn: NetJukeUser
uniqueMember: uid=user1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr

dn: cn=NetJukeTvTempo,ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: groupOfUniqueNames
objectClass: top
cn: NetJukeTvTempo
uniqueMember: uid=tvTemp1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr

dn: cn=NetJukeUserOff,ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: groupOfUniqueNames
objectClass: top
cn: NetJukeUserOff
uniqueMember: uid=userOff1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr

dn: cn=NetJukeTvOff,ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr
objectClass: groupOfUniqueNames
objectClass: top
cn: NetJukeTvOff
uniqueMember: uid=tvOff1,ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr
```

Cela nous permet de créer l'organisation netjuke, les deux unités organisationnelles groups et people, les six rôles et états NetJukeAdmin, NetJukeUser, NetJukeUserOff, NetJukeTv, NetJukeTvTemp et NetJukeTempOff, ainsi qu'une personne pour chacun de ses rôles.

B.2 Maintenance de la base de donnée

B.2.1 Introduction

Les instructions suivantes vous permettront de restaurer votre base de données en cas de problème. Vous trouverez pour cela une première partie contenant les scripts de créations de tables, une seconde,

présentant différents moyens de sauvegardes de vos données et une troisième, la restauration de ces données.

B.2.2 Script de création de table

Création de la base :
CREATE DATABASE ‘jukeboxe‘ ;
Création de la table « canal » :
DROP TABLE IF EXISTS ‘Canal‘ ;
CREATE TABLE IF NOT EXISTS ‘Canal‘ (
‘id_canal‘ int(3) unsigned NOT NULL auto_increment,
‘id_descripteurliste‘ int(10) unsigned NOT NULL default ’0’,
‘date_debut‘ datetime NOT NULL default ’0000-00-00 00 :00 :00’,
‘port_ecoute‘ int(10) unsigned default NULL,
‘port_envoi‘ int(10) unsigned default NULL,
‘ip_server‘ varchar(15) default NULL,
PRIMARY KEY (‘id_canal‘)
) ENGINE=InnoDB ;
Création de la table « DescripteurListe » :
DROP TABLE IF EXISTS ‘DescripteurListe‘ ;
CREATE TABLE IF NOT EXISTS ‘DescripteurListe‘ (
‘id_descripteurliste‘ int(10) unsigned NOT NULL auto_increment,
‘id_detailliste‘ int(11) unsigned NOT NULL default ’0’,
‘nom_liste‘ varchar(50) NOT NULL default ’’,
PRIMARY KEY (‘id_descripteurliste‘)
) ENGINE=MyISAM ;
Création de la table « DetailListe » :
DROP TABLE IF EXISTS ‘DetailListe‘ ;
CREATE TABLE IF NOT EXISTS ‘DetailListe‘ (
‘id_detailliste‘ int(11) unsigned NOT NULL default ’0’,
‘id_detailliste_2‘ int(11) unsigned default NULL,
‘id_doc_audio‘ int(11) unsigned default NULL,
‘ordre_passage‘ tinyint(4) unsigned default NULL,
‘heure_passage‘ time default ’00 :00 :00’
) ENGINE=InnoDB ;
Création de la table «DocumentAudio » :
DROP TABLE IF EXISTS ‘DocumentAudio‘ ;
CREATE TABLE IF NOT EXISTS ‘DocumentAudio‘ (
‘id_doc_audio‘ int(11) unsigned NOT NULL auto_increment,
‘id_ldap_proprietaire‘ int(10) unsigned default NULL,
‘titre‘ varchar(255) NOT NULL default ’’,
‘artiste‘ varchar(50) NOT NULL default ’’,
‘genre‘ varchar(50) NOT NULL default ’’,
‘album‘ varchar(50) NOT NULL default ’’,
‘id_remuneration‘ int(3) unsigned NOT NULL default ’0’,
‘remuneration‘ float NOT NULL default ’0’,
‘donnees‘ longblob,
‘duree‘ time NOT NULL default ’00 :00 :00’,
‘diffusable‘ tinyint(1) NOT NULL default ’0’,
PRIMARY KEY (‘id_doc_audio‘)
) ENGINE=InnoDB ;
Création de la table «Historique_diffusion » :
DROP TABLE IF EXISTS ‘Historique_diffusion‘ ;
CREATE TABLE IF NOT EXISTS ‘Historique_diffusion‘ (
‘id_historique_diffusion‘ int(11) NOT NULL default ’0’,
‘date‘ datetime NOT NULL default ’0000-00-00 00 :00 :00’,
‘id_doc_audio‘ int(11) unsigned default NULL,
‘id_canal‘ int(3) unsigned default NULL,
PRIMARY KEY (‘id_historique_diffusion‘)

) ENGINE=InnoDB ;
Création de la table « Historique_ecouteur » :
DROP TABLE IF EXISTS 'Historique_ecouteur' ;
CREATE TABLE IF NOT EXISTS 'Historique_ecouteur' (
'id_Historique_ecouteur' int(11) NOT NULL auto_increment,
'id_utilisateur' int(11) default NULL,
'id_historique_diffusion' int(11) default NULL,
PRIMARY KEY ('id_Historique_ecouteur')
) ENGINE=InnoDB ;
Création de la table « Remuneration » :
DROP TABLE IF EXISTS 'Remuneration' ;
CREATE TABLE IF NOT EXISTS 'Remuneration' (
'id_remuneration' int(3) NOT NULL auto_increment,
'description' varchar(64) default NULL,
PRIMARY KEY ('id_remuneration')
) ENGINE=InnoDB ;

B.2.3 Sauvegarde manuelle de la base « JukeBoxe »

B.2.3.1 Par le WEB :

Pour effectuer une sauvegarde de la base de donnée, il faut procéder comme suit :

1. Ouvrir une page web
2. Taper l'adresse suivante : <https://195.220.26.5/phpmyadmin>
3. Une authentification au serveur va être demandée.
4. Une fois celle-ci saisie, il faut s'authentifier auprès de la base en tant qu'administrateur.
5. Une fois l'authentification faite, Cliquer sur le lien « EXPORTER » de la page principale.
6. Sélectionner la table « JUKEBOXE », le langage SQL, cocher « structure », « Inclure des énoncés "DROP TABLE" », « Inclure la valeur courante de l'AUTO_INCREMENT », « Données » et « Insertions complètes ».
7. Cliquer sur le bouton «EXECUTER »

Le résultat permet de reconstruire les tables et de réinsérer les données qu'elles contenaient au moment de cette sauvegarde. Il est judicieux de la sauvegarder dans un fichier texte.

B.2.3.2 Par accès au serveur de Base de donnée

Sauvegarde complète :

Comme les tables MySQL sont stockées sous forme de fichiers, il est facile d'en faire une sauvegarde. Par un accès au shell, il faut écrire la commande suivante :

shell> mysqldump –all-databases –host=localhost –user=administrateur –password=motdepasseadmin > backup_day_houre_AMorPM.sql

Le fichier .sql résultant, produit par mysqldump contient les commandes SQL INSERT qui peuvent être utilisées pour recharger les tables ultérieurement.

Sauvegarde incrémentale :

Pour réaliser des sauvegardes incrémentales, vous devez sauver les modifications incrémentales. Le serveur MySQL doit être lancé avec l'option –log-bin pour qu'il puisse stocker ces modifications au fur et à mesure des modifications des données. Cette option active le log binaire.

Voici la commande à taper :

shell> mysqldump –single-transaction –flush-logs –master-data=2
–all-databases > backup_day_date_AMorPM.sql

Nous recommandons vivement une automatisation des sauvegardes avec une sauvegarde complète, une fois par semaine, à une heure de faible utilisation de votre base, et une sauvegarde incrémentale une fois par jour.

Il vous appartient d'utiliser l'utilitaire de programmation de tâche qui vous convient, mais nous recommandons l'utilitaire « CRON » pour l'environnement linux.

B.2.4 Restauration manuelle de la base de donnée

Pour effectuer une restauration de la base de donnée, il faut procéder comme suit :

1. Ouvrir une page web
2. Taper l'adresse suivante : <https://195.220.26.5/phpmyadmin>
3. Une authentification au serveur va être demandée.
4. Une fois celle-ci saisie, il faut s'authentifier auprès de la base en tant qu'administrateur.
5. Une fois l'authentification faite, cliquer sur le lien « SQL » en haut de page, et y coller le texte exporté lors de la dernière procédure de sauvegarde.
6. Cliquer sur le bouton « Executer »

B.2.4.1 Par accès au serveur de Base de donnée

il faut commencer par utiliser la première sauvegarde complète de la manière suivante :
shell> mysql < backup_sunday_1_PM.sql
Après cela, il faut utiliser les sauvegardes incrémentales, c'est à dire les fichiers de log binaire :
shell> mysqlbinlog fichier-bin.00000x fichier-bin.00000x | mysql

B.3 Mise en production du serveur de diffusion

Le lancement du serveur de diffusion se fait en plusieurs étape. Il vous faut au préalable un annuaire RMI disponible sur un serveur.

B.3.1 Configuration

La configuration se fait à l'aide d'un fichier de configuration présent dans le répertoire de l'applicatif et est nommé `config_serveur.ini`. Visualisons son contenue :

```
#serveur ayant le service RMI disponible
rmi=localhost

#serveur ou est présent l'allocateur
allocateur=localhost

#serveur qui gére la sécurité
securite=localhost

#le type de chiffrage désiré ... pour le moment DES et Blowfish
chiffrage=DES

#Ou sont les donnée musicale ?
interfaceDonnee=systemeDeFichier

#Uniquement si l'interface et le systeme de fichier
systemeDeFichier=/home/beytu/Documents/Musiques/ogg

#votre adresse public
adresse=127.0.0.1
```

- `rmi` : ce paramètre permet d'indiqué où est localisé votre annuaire RMI
- `allocateur` : ce paramètre permet d'indiqué où est localisé votre allocateur
- `securite` : ce paramètre permet d'indiqué où est localisé votre centre de securite
- `chiffrage` : Paramètre indiquant le chiffrage utilisé pour le système de signature client/serveur. Vous avez le choix entre Blowfish et DES pour le moment

- `interfaceDonnee` : le type d'interface de donnée désiré. Pour le moment les interface suporté sont systemeDeFichier et MySQL
- `systemeDeFichier` : si vous utilisez l'interface `systemeDeFichier`, il faut indiquer ici le répertoire où se trouve vos musiques. Les sous-répertoire de ce répertoire seront considéré comme des canaux à diffuser
- `adresse` : adresse public du serveur de diffusion

B.3.2 Lancement

Il faut dans un premier temps vous placer dans le répertoire de l'application et lancer les services d'allocation de ressource et de sécurité comme suit :

- L'allocateur :

```
beytu@proteus:~/workspace$ ./allocateur
2006-06-20 13:27:13 INFO  [Allocateur] Démarrage de l'allocateur...
2006-06-20 13:27:13 INFO  [Allocateur] L'allocateur est prêt
```

- Le centre de sécurité :

```
beytu@proteus:~/workspace$ ./securite
2006-06-20 13:27:58 INFO  [CentreSecurite] Le centre de securité se lance...
2006-06-20 13:27:59 INFO  [CentreSecurite] Le centre de securité est prêt
```

Enfin, le lancement du serveur peut se faire :

```
beytu@proteus:~/workspace$ ./launcher
2006-06-20 13:30:57 INFO  [OrdonnanceurServeur] Démarrage du superviseur...
2006-06-20 13:30:57 INFO  [OrdonnanceurServeur] Le superviseur est prêt
2006-06-20 13:30:57 INFO  [ImplementationSuperviseur] Balayage des canaux
2006-06-20 13:30:57 INFO  [ImplementationSuperviseur] Le canal n'existe pas ... je l'ajout
2006-06-20 13:30:57 INFO  [CoucheReseau] Mon port d'ecoute : 4000
2006-06-20 13:30:57 INFO  [CoucheReseau] Mon port d'envoie : 5000
2006-06-20 13:30:57 INFO  [OrdonnanceurServeur] Création de l'ordonnanceur pour P.O.BOX - 
2006-06-20 13:30:57 INFO  [OrdonnanceurServeur] Lancement de l'ordonnanceur pour P.O.BOX -
2006-06-20 13:30:57 DEBUG [ImplementationSystemeFichier] Canal interface : P.O.BOX - Rock 
2006-06-20 13:30:57 DEBUG [ImplementationSystemeFichier] Fichier : /home/beytu/Documents/M
```

Vous pouvez à présent profiter du système en vous connectant au portail Web.

Annexe C

Quelques recherches personnelles

C.1 Du son au Ogg/Vorbis

C.1.1 Le son

C.1.1.1 Introduction

La numérisation sonore a très vite été une préoccupation des informaticiens. Aujourd'hui son utilisation et son transport et devenu banal mais avant d'en arriver là, le traitement sonore est passé par bien des étapes. Afin de répondre aux difficultés rencontrées ainsi qu'aux désires des utilisateurs. Essayons de voir au fil de ce document cette évolution.

C.1.1.2 Présentation du son

Le son est une onde véhiculé grâce à l'air ambiant. Par exemple, lorsque nous jouons d'un instrument ou que nous parlons tous simplement, du son est produit à une certaine fréquence sous forme d'onde puis transporté grâce à l'air. Bien entendu les molécules composant l'air ne se déplace pas durant ce processus. Nous avons affaire à un phénomène de variation de pression qui est transmis de proche en proche à travers les molécules d'air. C'est ainsi qu'on peut expliquer l'absence de vent devant un haut parleur par exemple. Pour comprendre ce phénomène, nous pouvons faire une analogie avec une pierre jeté dans l'eau : les vagues se déplace mais pas les molécules d'eau, nous pouvons le vérifier en posant au préalable un objet flottant sur l'eau.

Cette onde sonore est caractérisé par bon nombres de spécificité avec entre autre sa fréquence ou son amplitude. La fréquence d'un son est exprimé en Hertz et il nous indique le nombre de variation d'amplitude en une seconde. Typiquement, une faible fréquence correspond à un son grave et une fréquence élevé correspond à un son aigu. Il est communément admis que l'oreille humaine perçois les son se trouvant entre 30 Hertz et 15 000 Hertz. Bien entendu, cette plage varie en fonction de l'âge, la culture et d'autre facteurs environementaux. L'amplitude quant à elle détermine le son perçu. Elle correspond à la moitié de la distance existante entre le point le plus haut de l'onde et le point le plus bas.

Ci dessous, un schéma pour expliquer ces différents point clé du son :

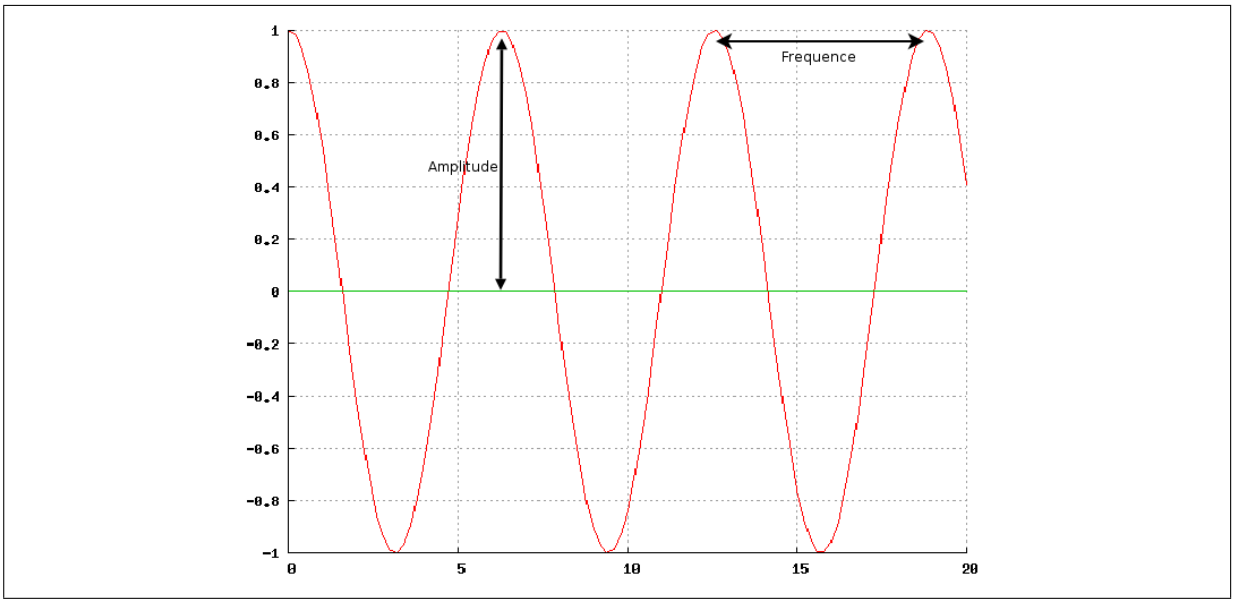


FIG. C.1: *Une onde sonore*

C.1.1.3 La numérisation sonore

Comme beaucoup de technologie informatique, la numérisation et l'exploitation informatique du son à trouvé ses débuts pendant la seconde guerre mondial. A cette époque, des recherches étaient effectué pour convertir cette onde sonore en une suite de valeur numérique facilement transportable, exploitable...

Cette conversion c'est faite en utilisant la notion d'échantillonnage : on relève l'amplitude de l'onde sonore à intervalle régulier plusieurs fois par seconde. Cependant, il faut que le nombre d'échantillon par seconde soit au minimum deux fois plus élevé que la fréquence de son la plus élevé. afin d'éviter une 'illusion acoustique' (en référence à l'illusion d'optique perçu lorsqu'on film une roue qui tourne plus ou moins vite). Pour de plus ample explications, nous pouvons faire référence au théorème de Nyquist. Cette spécificité explique entre autre le choix de 44 100 échantillon par seconde adopté pour les CD audio.

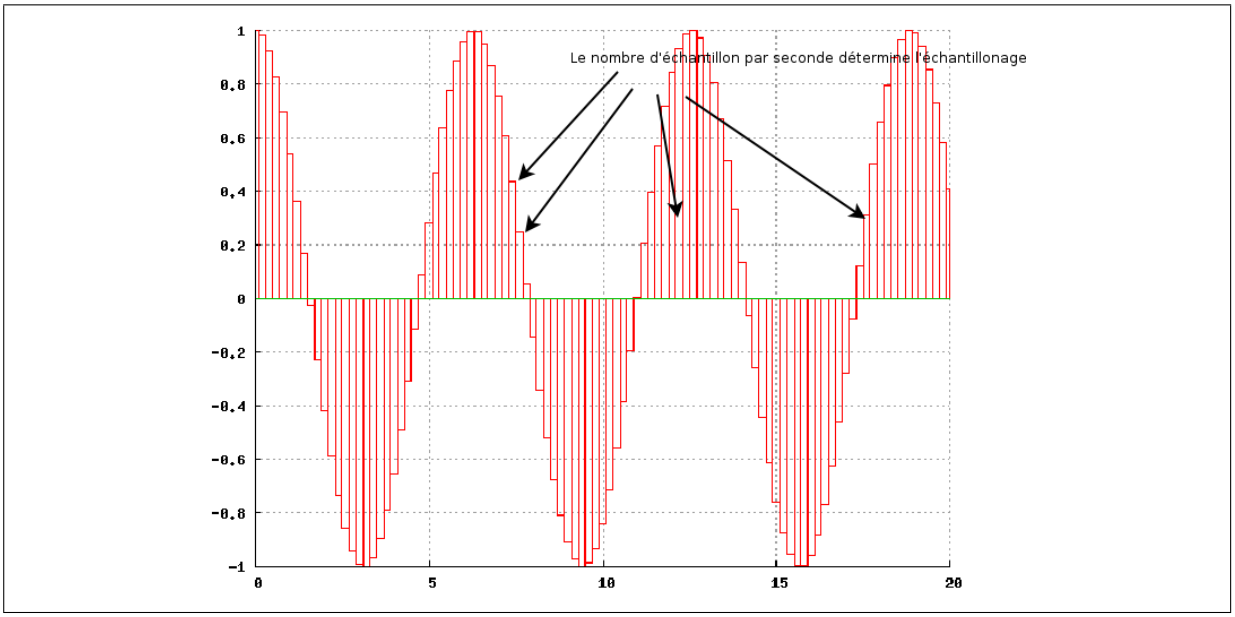


FIG. C.2: *Une onde sonore numérisé*

C.1.1.4 Le problème du transport

Chaque échantillon est représenté par un nombre codé en mémoire sur 16 bits ce qui nous donne une plage de nombre compris entre -32 768 et 32 767. Un son qui passe de -32 768 à 32 767 sera perçu

comme étant très fort tandis que des 0 nous signalerons du silence.

Cette méthode d'acquisition et de restitution sonore nous rend un son fidèle au son original. Seulement un problème fait vite voir le boue de son nez : le poids. En effet, l'espace disque nécessaire pour stocker une musique sauvegardé de telle manière et non négligeable. Si nous prenons une musique de trois minute à raison de 44 100 échantillons par seconde et avec un son mono, nous arrivons à :

```
16 bits * 44 100 échantillons * (60 * 3) secondes = 127 mégabits
```

Et 10 morceaux de trois minutes nécessitent :

```
127 mégabits * 10 = 1,27 gigabits
```

Il faudrait encore multiplier par deux ces résultat pour avoir une idée de l'espace disque nécessaire pour les morceaux stocké en stéréo. Il ne faut pas oublié aussi que certain artiste diffuse des DVD audio dont les pistes contiennes jusqu'à six canaux différents. Il faudrait donc multiplier ces résultats non pas par deux mais par six pour avoir une idée de l'espace disque nécessaire.

Si aujourd'hui nous avons des disques durs de grandes capacités et des connections haut débits, transférer de la musique resterais très longs et cher à stocker. De plus, les appareils nomades permettant d'écouter de la musique attire surtout pour le nombre de morceaux qu'ils peuvent transporter.

Nous pouvons donc dire qu'il est nécessaire de compresser le son afin d'avoir des morceaux plus facilement transportable, diffusable, stockable, ...

C.1.2 Méthodes de compression audio et présentation du Vorbis

C.1.2.1 Compression sans perte

Je ne vais pas beaucoup m'attarder sur ce type de compression. Mais ceux-ci reste très intéressant et attire surtout les amoureux du son qui ne supporte pas les distorsions ou la dégradation du son inérant aux algorithme de compression audio avec perte.

Avec ce type de compression, il est possible de retrouver le son original non compresser à partir du fichier compresser. Tout comme le ZIP, le RAR ou encore le ARJ vous permettent de compresser un fichier puis le retrouver ensuite intact pour le réutiliser, ce type de compression agit de façon analogue. Ceci dit, tandis que le ZIP vous permettrait de gagné environ 5% d'espace disque, des algorithmes de compression sans perte telle que le FLAC permettent de gagner jusqu'à 50% d'espace disque, ce qui reste non négligeable mais insuffisant pour la portabilité.

C.1.2.2 Compression avec perte

Très vite, pour des soucis de portabilité et de transfert, des formats de compression plus agressifs ont fait leur apparition dont notamment le très célèbre MP3 ou encore le Vorbis. Contrairement au format sans perte, il n'est pas possible d'obtenir la fichier source à partir du fichier compressé. En effet, ce type de compresseur supprimer les informations non utile pour notre compréhension du son afin de gagner de l'espace disque.

Comme je l'ai mentionné plus haut, il est communément admit que l'oreille ne peut entendre toute les fréquences codées dans le son original, on va donc les supprimer. De plus, aujourd'hui, quasiment toutes les musiques utilisent du stéréo. Ces formats de compression vont garder qu'une seul fois les informations identiques sur les deux canaux. C'est ainsi qu'en décompressant, nous n'obtenons pas une information identique à la source.

C.1.2.3 Le MP3

Introduction Le `MP3` est l'abbréviation `MPEG Audio Layer 3`. C'est un algorithme de compression audio qui s'attaque aux sons inaudible à l'oreille humaine. Bien que la perte soit significative dût à l'algorithme de compression, l'oreille d'un humain n'est pas capable d'entendre la différence avec le son original. Bien sûr, ce n'est pas le seul principe qui est utilisé dans ces algorithmes mais c'est celui qui à marqué le plus les professionnels du son.

La réduction drastique de la taille du fichier après compression a créé un engouement pour ce format et est rapidement devenu l'icône du piratage musical sur Internet. En effet, le succès est dû à un contexte particulier. Dans les débuts d'Internet, la bande passante disponible pour l'utilisateur était de cinq kilo-octets par seconde. Une musique au format WAV prennait un temps de transfert fou. Le `MP3` réduisit ce temps à environs trente minute. Cette révolution à notamment permis le succès et surtout l'exemple judiciaire Napster.

Taux de compression Le taux de compression est un concept difficile à cerner ici puisque bon nombre de paramètre entre jeux. De plus, l'introduction du principe de variation dynamique de débit àjouter de la complication.

De manière général, l'utilisateur définira un débit cible. L'utilisateur moyen ne trouve aucune différence entre un fichier MP3 encodé en utilisant débit cible de 192 kilo-bits par seconde. Pendant longtemps, ce débit était de 128 kilo-octes par secondes. Ce debit permettait d'avoir une musique avec une qualité raisonnable avec un temps de transfert d'environ trente minutes. Les utilisateurs arrivait à approximé une taille de fichier en partant du principe qu'une minute de musique prendrait 1 méga-octets d'éspace disque.

Aujourd'hui, avec l'Internet à haut débit, le 'standart' tend à être 320 kilo-bits par seconde pour ce format. Bien qu'il existe des formats de nouvelle génération permettant d'avoir une meilleur qualité sonore pour un débit moindre, ce format est encore très utilisé. Il reste dans l'ésprit de l'internaute moyen le format de fichier audio de réference.

Enfin, il existe trois familles d'utilisation :

– 384 kbit/s, compression 4 :1 appellé `Layer I`
– 160 à 256 kbit/s, compression 6 :1 à 10 :1 appellé `Layer II`
– 112 à 128 kbit/s, compression 12 :1 à 14 :1`Layer III`

Structure d'un fichier Un fichier `MP3` est fait d'une succéssion d'unité sonore appellé `Frame`. Nous appellerons `séquence` cette unité sonore dans la suite du document.

Chaque séquence peut être décompressé et écouté independamment du reste du flux. Ceci est à première vu une bonne nouvelle puisqu'il serait possible de se connecter à un flôt audio MP3 à tout moment et être capable de décompresser ce flôt. Ceci dit, sur une longue diffusion, la perte de bande passante est notable. En effet, nous transmettons avec chaque séquence les informations de décodage. Hors bien souvent, ces informations sont identique. Nous pensons donc que ce format très bien adapté pour une diffusion via Internet. L'idéal serait de transmettre cette donnée une seul fois.

Licence Beaucoup pense que ce format est libre d'utilisation et que les l'utilisation des algorithmes présents dans la documentation déposé auprès de l'ISO l'est aussi. Ceci dit, bon nombre de développeurs ont reçu en Septembre 1998 une lettre leur demandant des droits d'utilisation des algorithmes par l'institut Fraunhofer qui detient bon nombre de brevet concernant ce format.

C.1.2.4 Vorbis

Introduction Comme beaucoup de format de compression audio utilisé aujourd'hui, le **vorbis** est un format de compression à perte. Concretement, un fichier son compressé puis décompressé ne sera pas identique bit à bit au fichier original. A l'origine, ce format de compression a été fait pour concurrencér et apporter une alternative libre au MP3 ou encore au WMV.

Technologie utilisé Le Vorbis est un encodage très complexe. Je vais tenté de survoler ici ce format sans rentré trop dans les détails.

Pour commencer, nous pouvons dire que ce format ne se contente pas de supprimer les sons inaudibles. Il met en jeu un algorithme appellé MCDT : Modified Discrete Cosine Transforme ou Transformé de Consinus discret modifié. Cette algorithme transpose les données audio de l'échelle du temps sur l'échelle des fréquences. Le résultat est ensuite divisé en `Noise block` et residue. Les données sont quantisé en utilisant ce qu'ils appellent un `Codebook`. Le décodage est fait en utilisant ces opérations dans l'autre sens.

Ce qu'il faut retenir ici, c'est la séparation du codage est du décodage en couche d'abstraction. La configuration de ces couches se trouve dans l'entête du flux que nous décrivons ci-dessous.

Structure du flôt Le flôt Vorbis est constitué en paquets. Ces paquets peuvent être des entêtes de configuration ou des données audios. Chaque paquet est marqué pour connaître son contenue : entête ou donnée audio. Les entêtes de configuration permet de connaître les configurations adoptées pour l'encodage et permettre ainsi le décodage. Ici, le risque est que si l'entête est corrompu, toutes les données audio deviennent illisible.

Entête de configuration

- `Identification` : permet d'identifié le flôt de donnée comme étant du Vorbis.
- `Commentaires` : ce champs contient tous ce qui est considéré comme étant un commentaire. Le nom de l'artiste, le nom de l'album, le genre ... son stocké ici de la forme `[NOM_DU_CHAMPS]=[VALEUR]`.
- `Configuration` : contient les éléments permettant de configurer tous les couche d'abstraction du décodeur telles que l'interpreteur de `codebook`, l'interpreteur de `Noise block`...

Ces paquets sont marqués comment étant de type <0>. Une fois ces paquets réceptionnés, nous ne devons plus recevoir ce type de paquet. L'arrivé de ce type de paquet signie le comencement d'un nouveau flôt.

Pour décoder le son, nous devons absolument avoir l'entête d'identification afin de confirmer la présence de données Vorbis ainsi que la configuration utilisé lors de l'encodage. Les entêtes contenant les commentaires peuvent être corrompu. Nous n'aurons tous simplement pas d'informations sur l'artiste, l'album, le genre...

C.1.3 Un format d'encapsultation : le OGG

C.1.3.1 section

Ce format est avant tous utilisé pour le transport. Il peut être utilisé aussi bien pour du son que pour de la vidéo, des sous-titres... Il implémente des méchanismes de vérification du flôt ainsi que de synchronisation. Sa structure simple à comprendre en fait un choix idéal pour le transport de media. Généralement, un flôt Vorbis est encapsulé à l'aide du OGG pour le transport. Nous avons opter pour ce couple dans ce projet.

Nous abordons ce format plus en détail dans les annexes suivantes.

C.2 Spécification du format OGG

C.2.1 Definition

Pour décrire le processus d'encapsulation du Ogg,je vais donner un certain nombre de définitions. Le resultat d'une encapsulation Ogg est apellé "Physical Bitstream".Il encapsule un ou plusieurs flux encodés,qui sont apellés "Logical Bitstream". Un Logical Bitstream ,fournit au processus d'encapsulation Ogg,a une structure qui consiste a le découpé en séquence apellés «Packets».Les packets crées par l'encodeur du logical bitstream représentent une entité a part entière pour cet encodeur.

Ogg : Segments et pages

L'Ogg possède deux structures basiques : le segment et la page. Quand les paquets sont envoyés pour former un ogg, ils se présentent souvent sous la forme de petits paquets appelés segments. Les règles régissant les segments sont les suivantes :

- Chaque segment a une longueur de 255 octets, excepté le segment de fin.
- Le segment de fin DOIT avoir une longueur inférieure à 255 octets
- Pour les paquets ayant exactement 255 octets ou un multiple de 255, le segment de fin est zero, la raison de cela sera expliquée par la suite.
- Les paquets ayant moins de 255 octets ne sont pas fragmentés. Ils sont simplement déclarés comme étant le segment de fin.

La raison d'être de ces règles concernant le segment de fin est la suivante : quand vous avez un segment de longueur inférieure à 255, cela signifie que vous êtes arrivé à la fin du paquet.

Les pages sont l'ossature de l'ogg et fournissent une structure qui permet le support de la recherche aléatoire et de la détection d'erreurs dans l'ogg. Une page peut contenir jusqu'à 255 segments de 255 octets chacuns, ce qui donne une taille maximale de 65025 octets (sans compter l'entête). Mais il n'y a pas du tout obligation de faire rentrer totalement un segment dans une page (donc un segment peut chevaucher plusieurs pages). C'est le rôle de l'implémentation de définir si une page doit se terminer ou non avec une paquet (et Vorbis ne fait cela que dans une seule situation, par exemple).

C.2.2 FORMAT ENCAPSULATION DU OGG

Je vais vous présenter le format d'encaspulation du OGG. Le OGG est capable d'encapsuler tout les genres de format video ou audio ou informations encodés un simple flux.

Le résultat de l'encapsulation OGG est appellé le "Physical BitStream". Il encapsule un ou plusieus flux encodés qui sont appellée "Logical Bitstream". Un logical Bitstream fournit au processus d'encapsulation Ogg a une strutcture qui est décomposé/découpé en séquence appellée "Packets".

Les packets sont crée par l'encodeur du Logical BitStream et représente les entités significatives pour celui-ci.Ils ne contiennent pas d'informations de frontières,ficelés ensemble ils semblent être des flux de bits aléatoires sans bornes de limite.

L'idée de conception derrière l'Ogg était de fournir un format générique et linéaire de transport de médias pour permettre de stocker ensemble des données(informations et audio par exemple) dans un ou plusieurs flux intercalés indépendant de tout format de codage. Un tel format de codage doit fournit :

- Encadrement pour des Logical Bitstream
- Intercallage de différent Logical BitStream
- Détection des erreurs
- Possibilité de reprendre après une erreur d'analyse
- Capable de transmettre un flux
- Petit entête (qui n'utilise pas beaucoup de bande passante utilisé par le flux-1 à 2 % de la bande passante utilisé pour le flux)
- Possibilité d'analyser le flux rapidement
- Mécanisme de concaténation de plusieurs Physical BitStream

Tous ces considérations ont été prise pour l'Ogg.

C.2.3 FORMAT DU FLUX OGG

Un flux physique OGG est constitué de multiple Logical Bitstream intercallés appellés "Pages".Un logical BitStream est identifié par un numéro de série unique dans l'entête de chaque page d'un Physical Bitstream.

Ce numéro unique est crée aléatoirement et n'a aucun lien avec le contenu et l'encodage du logical bitstream qu'il représente.

Des pages de tous les logicals bitstream sont intercallées à la suite des autres ,mais n'ont pas un ordre régulier,ils sont seulement éxigés pour être consécutifs dans le bitstream logique. Le démultiplexeur Ogg reconstruit le logical bitstream original du physical bitstream en prenant les pages dans l'ordre et les redirige vers l'entité de décodage logique. Chaque page contient un type de données.Des pages sont de tailles variables et contiennent un entête de page contenant les informations d'encapsulation et de reouvrement d'erreur. Chaque logical bitstream dans un flux ogg commence avec une page de début(bos=beginning of stream) et finit avec une page spécial(eos=end of stream).

La page bos contient des informations unique qui identifie le type de codec et doit contenir des informations sur le processus de décodage. La page bos doit aussi aussi contenir des infos sur le type de média encodé(par exemple pour l'audio il doit y avoir le sample rate et le nombre de canaux par exemple).Par convention le premier octet de la page bos contient des données qui identifie le codec requis.

En résumé un physical bitstream commence avec les pages bos de tous les logical bitstream contenant un paquet d'entête initial par page,suivit par un autre paquet d'entête subsidiaire de tous les flux,suivit par des pages contenant les paquets de données.

La spécification d'encapsulation pour un ou plusieurs flux logiques est appellé le "media mapping". Un exemple pour un media mapping est le "Ogg Vorbis",qui utilise le framework OGG pour encapsuler les données audio Vorbis d'un fichier par exemple et le transporte (comme un flux TCP).

Le Ogg Vorbis fournit dans la page bos le nom et le type de codec Vorbis,le rate et la qualité de celui-ci.Il fournit aussi deux autres en-tête par flux logique.La page bos commence avec l'octet 0x01,suivit par "vorbis" (un total de 7 octet d'identification).

L'Ogg connaît 2 type de multiplexage : le multiplexage concurrant (appellé "Grouping") et le multiplexage séquentiel(appellé "chaining").

Le "grouping" définit comment intercallées plusieurs flux logiques (logical bistream)dans le même flux physique(physical bitstream). Par exemple il permet de grouper un flux audio avec plusieurs pistes audio utilisant des codecs différents.

Le "chaining" d'un autre côté est définit pour fournir un simple mecanisme de concaténation de flux physique Ogg,comme dans le plus souvent pour les applications de streaming.

Dans le grouping toutes les page ogg de tous les flux logiques doivent apparaître ensemble au début du flux OGG.Le media mapping spécifie l'ordre des pages initiales.Par exemple le grouping d'un flux Ogg video et audio doit spécifier que le flux physique doit commencer avec la page bos du flux logique video suivit par la page bos du flux audio.A la différence des pages bos,les pages eos des flux logiques

n'ont pas besoin de se suivre.Les pages eos peuvent être vides mais peuvent contenir une simple page d'entête contenant la position et un flag eos. Chaque flux logique groupés doivent avoir un numéro de série unique .

Dans le chaining les flux logique sont concaténé et ne doivent pas se chevaucher par contre les pages eos d'un flux logique donné est suivit immediatement par une page bos.Chaque flux logique chainé doit avoir un numéro de série unique.

C'est possible de chainé des groupes de flux multiplexés.Les groupes quand ils ne sont chaînés doivent attendre d'être un flux multiplexé valide. Le diagramme suivant montre un exemple d'un flux physique qui obéis au règles de groupage et de chaînage de flux multiplexés.

```
Flux physique avec des pages de différents flux logiques groupés et chainés.
      ------------------------------------------------------------
      |*A*|*B*|*C*|A|A|C|B|A|B|#A#|C|...|B|C|#B#|#C#|*D*|D|...|#D#|
      ------------------------------------------------------------
       bos bos bos             eos           eos eos bos       eos
```

Dans cet exemple,il y a deux flux physiques chainés,le premier est un flux groupés de 3 flux logiques 1,B et C.Le second est chainé après la fin du flux groupé,qui finit après la dernière page eos de tous les flux logique groupé.Comme on la vue des flux groupés commence ensemble-toutes les pages bos se trouvent avant les pages de données.

Ogg ne connaît pas toutes les spécificités du codec excepté le faîte qu'il peut savoir que chaque flux logique peut provenir d'un codec différents,les données du codec viennent dans l'ordre et a un marqueur de position(appellé granule position).

Ogg n'a pas de concept de temps,il connaît la notion d'incrémentation séquentielle,avec le marqueur de position. Une application peut aussi avoir une information temporelle à travers les couches hautes qui ont accès au codec de l'API et qui convertissent le granular position en temps.

Une définition spécifique du media mapping utilisant Ogg peut apporter d'autres contraintes dans son utilisation du format du flux Ogg.

Par exemple un media mapping specifique peut avoir besoin que toutes les pages eos de tous les flux groupés doivent apparaître dans une séquence directe. On a vu le media mapping «ogg Vorbis» mais il y a aussi le "Ogg Theora" qui encapsule des données vidéos et audio dans un flux ogg.

Comme le Ogg ne spécifie pas la relation de temps entre des flux multiplexés concurrents ,la synchronisation entre le flux audio et video est spécifié dans ce media mapping. Pour permettre de faire du streaming,les pages de flux logiques variés vont être intercallés dans un ordre chronolique.

C.2.4 LE PROCESSUS D'ENCAPSULATION

Le processus de multiplexage des différents flux logiques interviennent au niveau des pages décrites ci dessus. Les flux fournit par l'encodeur sont cependant manipulés par le Ogg et sont appellées paquets. Le processus d'encapsulations des paquets en pages va être décrit.

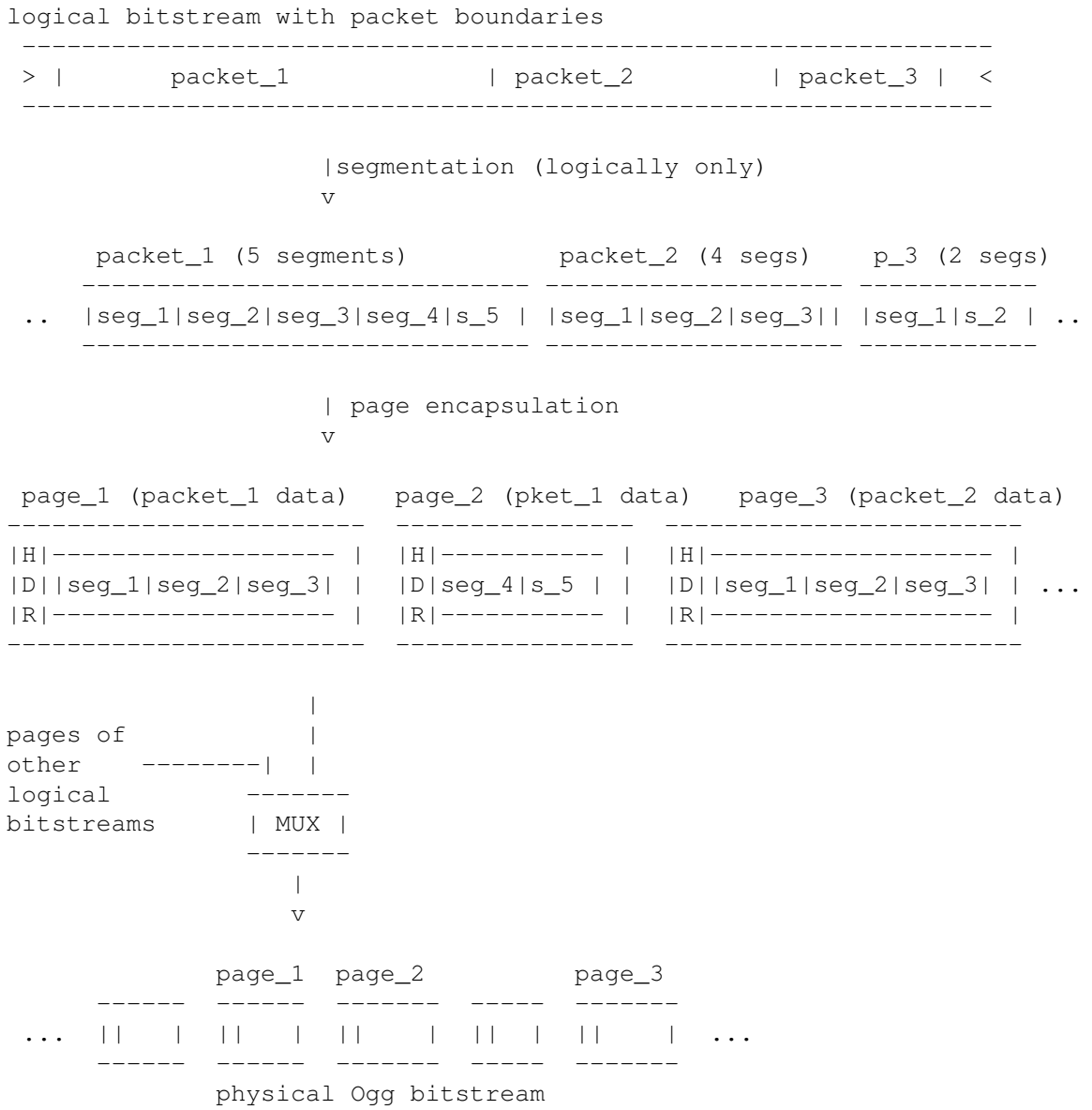
Pour l'Ogg les paquets peuvent être de n'importe quelle taille.Le media mapping définit le faîte de regrouper ou de découper des paquets en fonction de l'encodage.

Comme des pages Ogg ont une taille max de 64 kBytes,quelques fois le paquet peut être répartie sur plusieurs pages.Pour simplifier le processus, le Ogg divise chaque paquet en segment de 255 byte plus un segment plus court.

Ces segments sont appellées «segments Ogg».Il ont seulement une construction logique et n'ont pas d'entête pour eux-memes.

Un groupe contigue de segments sont encapsuler dans une page de longueur variable précédé par un entête. Une table de segment dans l'entête de page spécifie la taille(Lacing values) des segments inclus dans la page.Un flag dans l'entête de page dit si une page contient la continuité du paquet d'une page précédente. Notez qu'un Lacing Value de 255 implique qu'un second Lacing Value suit dans le paquet et que la valeur est inférieur à 255 pour marquer la fin du paquet. Un paquet de 255 bytes(ou un multiple de 255 bytes) est terminé par un lacing value de 0.Notez aussi qu'un paquet de longueur 0 n'est pas une erreur, cela amène un lacing value de 0 dans l'entête. L'encodage est optimisé pour la vitesse et dans le cas ou la majorité des paquets font entre 50 et 200 bytes de larges.Cet encodage évite d'imposé une taille max de paquets aussi bien qu'une taille min .

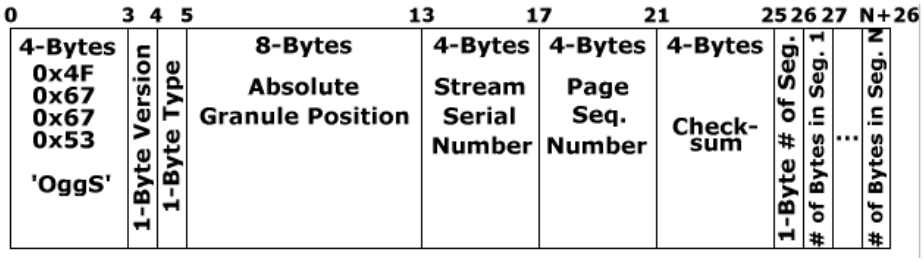
Le diagramme suivant montre un exemple de media mapping utilisant le Ogg et groupant des flux logiques :



Dans cet exemple nous avons prit une partie du processus d'encapsulation d'un flux logique. Nous pouvons voire une partie de la division du flux en paquets par le codec. Le processus d'encapsulation Ogg coupe les paquets en segments.Les paquets dans cet exemple sont plutôt large comme le paquet 1 découpé en 5 segments(4 segments de 255 bytes et un dernier plus petit).Le paquet 2 est découpé en 4 segments(3 segments de 255 bytes et un dernier très petit.Enfin le paquet 3 est découpé en 2 segments. Le processus d'encapsulation crée ensuite les pages. La page1 et la page 2 contiennent les 5 segments du paquet 1 et la page 3 contientles 3 segments du paquet 2. Enfin le flux logique est multiplexé en flux physique Ogg avec des pages d'autres flux logiques.

C.2.5 ENTETE DE PAGE OGG

Le schéma suivant est une description concise de ce qu'est un entête de page dans le format ogg :



OggS

Les quatre premiers octets de l'entête de page sont TOUJOURS OggS codé en UTF-8. En hexadécimal, cela donne 4F 67 67 53. Cela permet de rechercher les entête de pages. Note : il y a une chance sur 2^32 (~ 4 milliard) que la suite 0x4F676753 apparaissent aléatoirement dans le flux de données, donc il faut s'assurer que ce qu'on a trouvé EST bien un entête de page.

- 0x4f 'O'
- 0x67 'g'
- 0x67 'g'
- 0x53 'S'

Ces 4 octets signifient le début de page.Il aide le décodeur à trouver la frontière entre les pages et permet de le resynchroniser si le flux est corrompu.

Version : Stream Structure Version

Cet octet signifie le numéro de version du format du fichier Ogg utilisé dans ce flux. Pour le moment, cet octet est mis à 0. Il semblerait que ce nombre fasse référence à la structure ogg.

Type : Header Type

- Bit 0x01 : La page contient des données d'un paquet qui continuait depuis la page précédente.
- Bit 0x02 : C'est la première page du flux logique(logical Bitstream)->bos
- Bit 0x04 : C'est la dernière page du flux logique->eos

Il y a trois flags contenus dans cet octet (un flag est une variable qui peut prendre seulement 1 ou 2 valeurs) :

- Premier bit : 0 si vous commencez avec un nouveau paquet dans cette page. 1 si vous continuez un paquet de la page précédente.
- Deuxième bit : 1 si c'est le début d'un flux binaire, 0 sinon
- Troisième bit : 1 si c'est la fin d'un flux binaire, 0 sinon.

Comme il n'y a que 3 bits, je vais donner les différentes combinaisons possibles :

- 0x00 - On est au milieu d'un flux binaire, et on commence un nouveau paquet avec cette page
- 0x01 - On est au milieu d'un flux binaire, et on continue un paquet de la page précédente.
- 0x02 - On commence un flux binaire, et on démarre avec un nouveau paquet.
- 0x03 - On commence un flux binaire, en continuant un paquet de la page précédente (NOTE : cela pourrait être impossible par définition)
- 0x04 - On termine un flux binaire, mais on commence cette page avec un nouveau paquet.
- 0x05 - On termine un flux binaire en continuant un paquet de la page précédente.
- 0x06 - On commence ET termine un flux binaire : cela se produit quand le flux rentre entièrement dans une seule page.
- 0x07 - Même chose, mais en prolongeant un paquet de la page précédente (NOTE : cela pourrait être impossible par définition)

Position granulaire absolue (Absolute Granular Position)

NOTE

Tous les nombres à partir d'ici sont stockés dans le header en Little Endian, ce qui signifie que 44100, qui devrait normalement être écrit 0xAC44, sera écrit 44 AC.

Ce nombre est utilisé par une implémentation spéciale. Il réprésente une référence à un temps. En audio, cela serait par exemple le numéro de l'échantillon. En vidéo, ce serait un numéro de frame. Il fait référence à la position granulaire absolue à la fin d'un paquet de la page. Si aucun paquet ne se termine dans la page, cette valeur est mise à -1 (remplie avec des FF, puisque -1 en est le complément à 2)

NdT : granulaire est une traduction brutale de granular. L'idée qui est derrière, c'est qu'on compte la position absolue dans une unité qui est indivisible, par exemple la frame en vidéo, ou l'échantillon en audio.

Ce champ de 8 octets contient l'infomation de position.Par exemple pour un flux audio il doit contenir le nombre total de samples encodés PCM après avoir inclus toutes les frames finis sur cette page. Pour un flux video il doit contenir le nombre total de frames video encodés après cette page. Si la valeur est 8211 ;1 cela signifie qu'il n'y a pas de paquets terminés dans cette page.

Numéro de série du flux (Stream Serial Number)

Ce nombre est l'une des caractéristiques les plus importantes de l'ogg. A chaque flux binaire contenu dans un ogg, on associe un identifiant unique : le numéro de série de flux. Pour l'Ogg Vorbis, il n'y a qu'un seul flux binaire. Pour l'Ogg Media, il y en a au moins deux, vorbis et DivX / XviD par exemple, et parfois plus (si on rajoute des sous-titres).

Comme vous pouvez le constater, l'Ogg supporte jusqu'à 2^32 flux différents. Et chaque flux peut être mixer comme on veut. La seule limitation est qu'un flux doit être ordonner dans le fichier, et donc que la page n°33 d'un flux spéficique ne doit pas se retrouver avant la page n°32 de ce même flux. Mais elle peut se retrouver avant la page n°32 d'un autre flux sans aucun problème. Pour plus de détail, allez regarder la description du flux binaire de Xiph.

Ce champ contient un numéro de série unique par lequel le flux logique est identifié.

Numéro de page dans la séquence (Page Sequence Number)

C'est le compteur de pages. Il permet d'identifier la page dans un flux binaire. Dans un Ogg, à l'intérieur d'un flux binaire, les numéro de pages DOIVENT être ordonnés. Il permet au décodeur de détecter si une page est perdu.

Somme de contrôle : valeur CRC sur 32-bits (Checksum)

Cette valeur est calculée pour la page entière, en incluant l'entête, et peut être utilisée pour vérifer qu'aucun bit n'a été modifié.

Nombre de segments (Number of segments)

Cela dit combien de segments sont contenus dans la page. Souvenez vous qu'un paquet est long de 255 octets, il occupe deux segments.

Nombre d'octets dans le segment 1..n

C'est la table appelée "table des segments de l'entête" (Header segment table) dans les spécifications. Xiph nomme ces valeurs "valeurs de laçage" (Lacing value), mais il me semble que cela représente le nombre d'octets dans chaque segments. Souvenez vous que la fin d'un paquet est identifiée quand un segment contient moins de 255 octets. Souvenez vous aussi que les paquets ayant une taille multiple de 255 ont un segment additionnel vide.

Le taille totale en octet de l'entête est donnée par :

```
Taille_entête= nombre de segments + 27
```

La taille totale de la page est :

```
Taille_page=Taille_entête + somme(lacing_values:1..nombre de segments).
```

C.2.6 SECURITE DE L'OGG

Le format de l'encapsulation Ogg est un conteneur et encapsule un flux(comme le Vorbis). Il ne permet pas de faire de l'encryptage de lui-même ou de son contenu. Cependant il encapsule tous les genres de flux autant qu'il y a un codec pour cela,et donc il est capable de contenir un contenu de données encrypté.Il est aussi possible d'ajouter un mécanisme de sécurité supplémentaire qui encrypte et signe un flux physique Ogg et fournit donc un contenu confidentiel et authentique(authenticité).

Comme l'Ogg encapsule des données binaire,il est possible d'inclure un exécutable dans le flux Ogg.cela est une solution avec les applications qu'ils l'implémentent en utilisant le format Ogg, spécialement quand quand l'Ogg est utilisé pour le streamng ou le transfert de fichier dans un réseau.

C.2.7 GLOSSAIRE

`Bos page` : C'est la page initiale(débutant le flux) d'un flux logique qui contient des informations pour identifier le type de codec et des informations sur le décodage.

`Chaining(ou multiplexage séquentiel)` :C'est la concaténation de 2 ou plus de flux Ogg physique.

`Eos page` :C'est la dernière page (fin du flux) d'un flux logique.

`Granule position` :C8216 ;est un numéro de position (qui augmente) pour un flux logique spécifique qui est stocké dans la page d'entête.Ce champ est dépendant du codec pour ce flux logique et spécifié dans un spécifique media mapping.

`Grouping(multiplexage concurrent)` :Intercalage de pages de plusieurs flux logiques dans un flux physique Ogg complet avec la restriction que toutes les pages bos de tous les flux logiques doivent apparaître avant les pages de données.

`Lacing value` :une entrée dans la table des segment d'une page d'entête représentant la taille du segment relaté.

`Logical bitstream` :Une séquence de bits étant le résultat d'un encodage d'un flux.

`Media mapping` :une utilisation spécifique d'un ensemble de format d'encapsulation Ogg avec un codec spécifique.

`(ogg)packet` :une sous-partie du flux logique qui est crée par l'encodeur et représente une entité a part entière pour celui-ci,mais seulement une séquence de bits de l'encapsulation Ogg.

`(ogg) page` :Un flux physique composé d'une séquence de pages ogg contenant des données d'un seule flux logique.Il contient habituellement un groupe de segment contigue d'un seul paquet,mais quelquefois des paquets sont trop large et on besoin d'être découpé en plusieurs pages.

`Physical bitstream(ogg)` :Une séquence de bits résultant d'une encapsulation d'un ou de plusieurs flux logique. il est composé de d'une sequence de pages de flux logiques avec la restriction que les pages d'un flux logique doivent être dans l'ordre.

`(Ogg) segment` :Le processus d'encapsulation Ogg découpe chaque paquet en morceau de 255 octets plus un dernier morceau de moins de 255 octets.Ces morceaux sont apellés des segments.

C.3 La librairie C LIBOGG

C.3.1 Les types de donnée en jeu

Libogg utilise plusieurs structures de données pour les données et les informations d'états.

C.3.1.1 OGG_PAGE

Présentation

Cette structure encapsule des données dans une page de flux ogg. La structure ogg_page encapsule des données pour une page ogg. Des pages ogg sont l'unité fondammentale d'encadrement et d'entrelacement dans un flux ogg.Elles sont constituées chacune de segment de 255 octets.Chaque page peut contenir jusqu'à 255 segments ce qui ramène à une taille max de 64kB. Mais il n'y a pas du tout obligation de faire rentrer totalement un segment dans une page (donc un segment peut chevaucher plusieurs pages).

Structure

```
typedef struct {
  unsigned char *header;
  long                header_len;
  unsigned char *body;
  long                body_len;
} ogg_page;
```

Définition des champs

Header :
Pointe sur l'entête de la page.Le contenu exact de cet entête est contenu dans la specification de l'OGG partie encadrement(Framing).

Header_len :
Longueur de l'entête en octet.

Body :
Pointe sur les données de la page

Body_len :
Longueur des données en octet .

C.3.1.2 OGG_STREAM_STATE

Présentation
La structure ogg_stream_state donne(piste) l'état courant de l'encodage/décodage du flux logique.

Structure

```
typedef struct {
  unsigned char        *body_data;        /*  bytes from packet bodies */
  long       body_storage;               /*  storage elements allocated */
  long       body_fill;                  /*  elements stored; fill mark */
  long       body_returned;              /*  elements of fill returned */
  int        *lacing_vals; /* The values that will go to the segment table */
  ogg_int64_t *granule_vals; /* granulepos values for headers. Not compact
                                this way, but it is simple coupled to the
                                lacing fifo */
  long       lacing_storage;
  long       lacing_fill;
  long       lacing_packet;
  long       lacing_returned;
  unsigned char          header[282];          /* working space for header encode */
  int                    header_fill;
  int        e_o_s;            /* set when we have buffered the last packet in the
                                  logical bitstream */
  int        b_o_s;               /* set after we've written the initial page
                                  of a logical bitstream */
  long        serialno;
  int         pageno;
  ogg_int64_t packetno;               /* sequence number for decode; the framing
                                knows where there's a hole in the data,
                                but we need coupling so that the codec
                                (which is in a seperate abstraction
                                layer) also knows about the gap */
  ogg_int64_t        granulepos;
} ogg_stream_state;
```

Définition des champs

Body_data :
Indique les données du corps du paquet.

Body_storage :
espace alloué pour le corps en octets.

Body_fill :
Quantité d'espace utilisée par le corps du paquet stocké.

Body_returned :
nombre d'éléments retouné du stockage(de l'espace)

Lacing_vals :
chaine de valeurs de laçage des segments dans la page courante .Chaque valeurs est un octet,indiquant la longuer du segment.

Granule_vals :
indique les valeurs de laçage des segments de paquets dans une page courante.

Lacing_storage :
Quantité total de stockage(en octet) alloués pour stocker les valeurs de lacage.

Lacing_fill :
marqueur de remplissage de l'allocation de stockage courant contre l'allocation de stockage totale de la valeur de lacage pour la page.

Lacing_packet :
valeur de lacage du segment courant.

Lacing_returned :
nombre de valeur de lacage retourné du lacing_storage

Header :
zone de stockage temporaire pour un entête de page durant le processus d'encodage,pendant que l'entête est en train d'être crée.

Header_fill :
marqueur de remplissage de l'espace alloué pour l'entête.Utilisé durant le processus de creation de l'entête.

E_O_S :
marqueur placé quand le dernier paquet du flux logique a été bufferisé.

B_O_S :
marqueur placé après que nous avons écrit la première page dans le flux logique.

Serialno :
numéro de série du flux logique.

Pageno :
numéro de la page courante dans le flux

Packetno :
numéro du packet courant.

Granulepos :
position exact du processus d'encodage/décodage.

C.3.1.3 OGG_PACKET

Présentation

La structure ogg_packet encapsule les données pour un simple packet OGG VORBIS.

Structure

```
typedef struct {
   unsigned char *packet;
   long bytes;
   long b_o_s;
   long e_o_s;
   ogg_int64_t      granulepos;
   ogg_int64_t      packetno;
} ogg_packet;
```

Définition des champs

Packet

Indique les paquets de données.Cela est traité comme une couche opaque par la couche ogg

Bytes :

Indique la taille du paquet de donnée en octet.Des packets peuvent être de taille arbitraire.

B_O_S :

Drapeau indiquant si ce paquet commence un flux logique. 1 indique que c'est le premier paquet,0 indique une autre position dans le flux.

E_O_S :

Drapeau indiquant si le paquet termine un flux. 1 indique la fin du packet,0 indique une autre position dans le flux.

Granulepos :

Un numéro indiquant la position de ce paquet dans les données décodés.C'est le dernier échantillon,frame ou tout auttre unité d'information(8216 ;granule') qui peut être complètement décodée de ce paquet.

Packetno :

Numéro de séquence de ce paquet dans le flux ogg.

C.3.1.4 OGG_SYNC_STATE

Présentation

Cette structure contient les informations de synchronisation du flux.

Structure

```
typedef struct {
  unsigned char *data;
  int storage;
  int fill;
  int returned;
  int unsynced;
  int headerbytes;
  int bodybytes;
} ogg_sync_state;
```

Définition des champs

Data :
Pointe les données des corps du paquet

Storage :
pointe les données des corps du paquet

C.3.2 Les fonctions essentielles

Présentation

Libogg contient plusieurs fonctions qui sont généralement utilisé quand vous utilisez le flux ogg,même encodé ou décodé. Ces fonctions peuvent être utilisées pour manipuler des éléments basiques de l'OGG-flux et pages. Des flux et pages sont importants durant les processus d'encodage et de décodage.

`OGG_STREAM_INIT` : Initialise le flux OGG

`OGG_STREAM_CLEAR` : Dégage le stockage du flux ogg,mais ne le libère pas.

`OGG_STREAM_RESET` : Remise à zéro de l'état du flux dans sa position initiale

`OGG_STREAM_DESTROY` : Libère complétement le flux ogg.

`OGG_STREAM_EOS` : Indique que nous sommes à la fin du flux

`OGG_PAGE_VERSION` : Retourne la version de la page ogg que ce flux ou cette page utilise.

`OGG_PAGE_CONTINUED` : Indique que si la page courante contient un paquet de données qui est la suite du paquet qui a commencé dans la page précédente.

`OGG_PAGE_PACKETS` : Indique le nombre de paquet contenu dans une page.

`OGG_PAGE_BOS` : Indique si la page courante est le début du flux

`OGG_PAGE_EOS` : Indique si la page courante est la fin du flux

`OGG_PAGE_GRANULEPOS` : Retourne la position granulaire exacte du paquet de données contenu à la fin de cette page.

`OGG_PAGE_SERIALNO` : Retourne un numéro de série unique pour le flux logique de cette page.Chaque page contient le numéro de série du flux logique à qui elle appartient.

`OGG_PAGE_PAGENO` : Retourne le numéro de page séquentiel de cette page

`OGG_PACKET_CLEAR` : Nettoie la structure OGG_PACKET

`OGG_PAGE_CHECKSUL_SET` : Fait un checksum de la page ogg(ogg_page).

C.3.3 Fonctions d'encodages

Présentation

Libogg contient un ensemble de fonctions utilisé dans le processus d'encodage. Pendant l'encodage,l'encodeur produira des paquets qui seront placés dans le flux ogg. Les paquets sont insérés dans le flux,et une page ogg(ogg_page) est produite quand assez de paquets ont été écrit pour créer une page complète.Les pages produitent sont des indicateurs pour bufferiser des paquets de segment,et peuvent ensuite être écrit et sauvegardé comme flux ogg. Il y a plusieurs étapes de bases pour l'encodage :

- Utilisation d'un encodeur pour produire un paquet de données.
- Appel de la méthode ogg_stream_packetin pour fournir le paquet au flux.
- Utilisation de la fonction ogg_stream_pageout pour produire une page,s'il y a assez de donnée à soumettre.Sinon on continue de soumettre des données.

`OGG_STREAM_PACKETIN` : Soumet un paquet à la couche de flux,de sorte qu'il puisse être formé en une page.

`OGG_STREAM_PAGEOUT` : Produit une page complète si le flux contient assez de paquets pour former une page pleine.

`OGG_STREAM_FLUSH` : Force tous les paquets restants dans le flux à être retourné comme une page de n'importe quelle taille.

C.3.4 Fonction de décodage

Présentation

Libogg contient un ensemble de fonctions utilisé dans le processus de décodage. Le décodage est basé autour de la couche de synchronisation ogg. La structure ogg_sync_state coordonne les données entrantes et le décodeur. Nous lisons les données dans la couche de synchronisation,soumission des

données au flux et production d'un paquet pour le décodeur. Le décodage à travers la couche ogg suit une séquence logique. Les étapes logiques du décodage :

- Expose un buffer utilisant ogg_sync_buffer().
- Lecture des données du buffer,utilisant fread() ou une fonction similaire
- Appel la fonction ogg_sync_wrote() pour dire à la couche synchronisation combien d'octets vous devez écrire dans le buffer
- Ecrit des données en utilisant ogg_sync_pageout.
- Soumet une page complèté à la couche de flux avec ogg_stream_pagein
- Produit un paquet de données au décodeur utilisant ogg_stream_packetout.

En résumer,les flux sont assez complexe et OGG aussi manipule les entêtes,les pages inachevées et autres erreurs en entrée.

`OGG_SYNC_INIT` : Initialise le flux ogg

`OGG_SYNC_CLEAR` : Nettoie les informations d'états de la structure de synchronisation

`OGG_SYNC_RESET` : Remet à zéro l'état de synchronisation aux valeurs initiales.

`OGG_SYNC_DESTROY` : Libère la structure de synchronisation

`OGG_SYNC_BUFFER` : Fournit une taille propre au buffer pour l'écriture.

`OGG_SYNC_WROTE` : Demande à la couche de synchronisation combien d'octets ont été écrit dans le buffer.

`OGG_SYNC_PAGESEEK` : Trouve les bords de pages et resynchronise le flux.

`OGG_SYNC_PAGEOUT` : Produit une page de la couche de synchronisation.

`OGG_STREAM_PAGEIN` : Ajoute une page complète au flux.

`OGG_STREAM_PACKETOUT` : Produit un paquet au décodeur.

`OGG_STREAM_PACKETPEEK` : Assemble des paquets de données et les retourne sans décodage.

C.4 MP32OGG étend notre JukeBoxe aux musiques mp3

C.4.1 Notre choix

Nous avons fait le choix de concevoir notre JukeBoxe avec des technologies qui soutiennent ou qui poussent le logiciel libre.Même si une grande partie de l'application est faîte en JAVA qui n'est pas officiellement un logiciel libre mais à tout de même une licence assez ouverte qui a permit l'éclosion de nombreux logiciel libre.

Des exemples de technologies utilisées pour le développement de notre application prouve que l'on privilégie les logiciels et technologies libres comme :

- Eclipse(Environnement de développement)
- Ant (Outil de construction d'application)
- Jakarta Tomcat (Serveur web et conteneur de servlets)
- Log4j(Bibliothèque de journalisation de l'éxécution d'une application)

Cette politique s'exprime même jusqu'à la diffusion des documents audios à travers le réseau et pour cela nous utilisons le format Ogg/Vorbis et uniquement ce format pour la diffusion.

Une présentation du Ogg/Vorbis est faîte ailleurs dans la documentation donc nous allons pas nous atarder dessus ici.Nous allons rappeler seulement que le Ogg/Vorbis est un format de compression audio avec ses codecs qui fait partie du projet Ogg de la fondation Xiph.org dont le but est de proposer à la communauté des formats et codecs multimédias ouverts,libres et dégagés de tout brevet.

Nous allons rappeler succintement les avantages du Ogg/vorbis sur le MP3 en incluant le point que nous avons cité ci-dessus.

Les avantages du Ogg/Vorbis sur le MP3 sont :

- Format libre qui permet le développement actif et continu,une gratuité totale aussi bien pour l'artiste que pour l'auditeur. Rappelons que le MP3 n'est ni libre ni gratuit et il appartient à l'Institut Fraunhofer et à Thomson Multimédia qui en 1998 ont décidé de facturer l'utilisation de leur format pour chaque encodeur, et en 2002 de faire payer pour chaque décodeur. Les tarifs officiel se trouve sur le sîte http ://www.mp3licensing.com/royalty/index.html.
- Le Ogg/Vorbis permet d'encoder jusqu'à 255 canaux alors que le MP3 quand à lui permet d'encoder que 2 canaux c'est-à-dire qu'on peut encoder et décoder de la musique stéréo, mais rien de plus. Un exemple concret : si vous voulez encoder de la musique provenant d'un DVD avec du son au format 5.1 (cinq canaux plus un canal pour les basses fréquences), Ogg Vorbis vous permet de le faire sans problème, tandis que le MP3 réduira votre superbe son "surround" à un morceau stéréo banal.

– Pour la même taille de fichier, la qualité d'un morceau en Ogg Vorbis est supérieure à un fichier MP3 et pour la même qualité, un fichier MP3 est plus grand. Nous nous reposons sur le sîte http ://txtman.free.fr.free.fr/article/002_mp3_vs_ogg.htm qui comparent les ratios taille/qualité d'Ogg Vorbis et MP3 (et quelques autres formats).
– Au niveau compatibilté logiciel et plateforme le MP3 a un niveau plus élevé que le Ogg/vorbis c'est -à-dire que le MP3 est lus sur plus de logiciels multimédia.Même si actuellement le Ogg/Vorbis est compatible dans les plateformes Windows,Linux et Mac il n'est pas comprit par Windows Media Player qui est le player de Microsoft.En effet Microsoft a choisi la voie de la lecture contrôlée de fichiers audio/vidéo.

Après cette petite présentation des avantages du Ogg/Vorbis sur le MP3 nous poursuivons par la problématique que nous pose la diffusion exclusive du Ogg/Vorbis.

C.4.2 Problématique

La première version de notre Jukeboxe est limité à la diffusion du seul format Ogg/Vorbis.Ce qui signifie qu'actuellement seul les fichiers Ogg/Vorbis ne peuvent être postés dans notre application ce qui limite considérablement ses possibilitées. Dans cette version aucun système n'est mit en place pour répondre à ce manque évident d'accepter d'autres formats de compression audio.

C.4.3 Solution

Dans la prochaine version du Jukeboxe une nouvelle fonctionnalité va faire son apparition et qui va répondre à cette contrainte. En effet il sera possible aux personnes le désirant de poster des musiques au format mp3. Notre Jukeboxe sera capable d'accepter des fichiers mp3 pour les encoder au format Ogg/Vorbis et ainsi permettre la diffusion de ces fichiers à travers le réseau. De plus il étend l'utilisation de notre système à d'autres propriétaires de musiques (artiste,maison de disque)qui souhaitent pour certaines raisons qui peuvent être financières de diffuser des documents audios au format Ogg/Vorbis.

Nous revenons rapidement sur l'avantage que peut proposer la diffusion du Ogg/Vorbis au point de vue financier aux auteurs de chansons et distributeurs par exemple :

– Nous savons que le format mp3 est payant. Bien sur, les entreprises qui détiennent le brevet mp3 ne demandent pas de subsides à l'utilisateur,par contre, elles imposent aux sociétés qui souhaitent utiliser le format une rémunération . Or le mp3 est très utilisé aujourd'hui, notamment pour les jeux, vous payez donc indirectement la redevance démandée par les possesseurs du brevet. Rien n'interdit à Thomson Multimedia(c), détentrice du brevet, de rendre du jour au lendemain cette technologie payante.

Au final notre JukeBoxe va permettre de convertir les fichiers MP3 des auteurs de chansons et distributeurs au format Ogg/Vorbis pour ceux qui n'ont pas la possibilité de le faire.Notre JukeBoxe va permettre de passer d'un format propriétaire vers un format libre

Même s'il y a à priori des risques théoriques de perdre en qualité par rapport à un fichier source au format .WAV, la conversion des MP3 en OGG peut être très intéressante ! La perte de qualité à l'oreille humaine est peu flagrante voire inexistante,la perte est bien inaudible.

C.4.4 Tout avec MP32OGG

Mp32Ogg est un script Perl en 8220 ;license artistic8221 ;(http ://www.opensource.org/licenses/artistic-license.html) qui permet de convertir des fichiers MP3 au format Ogg en reprenant les ID3 Tag du fichier MP3 .Il reconstruit correctement le fichier MP3 en fichier Ogg. Il permet aussi de renommer le fichier avec l'extension .ogg et de supprimer à la volée l'original(fichier mp3) et bien d'autres options.

L'auteur de ce script est Nathan Walp qui ne le maintient plus,faute de temps. Nous allons montrer une simple démonstration de Mp32Ogg sans l'avoir intégrer dans notre JukeBoxe

Installation sous Debian où autre système d'exploitation s'en inspirant(Ubuntu...) :

Recherche du nom du paquet :

```
root@zeus:/home/ilies# apt-cache search mp32ogg
mp32ogg - Converts MP3 file to Ogg Vorbis
```

Installation de Mp32Ogg

```
root@zeus:/home/ilies# apt-get install mp32ogg
```

Prérequis pour l'installation de Mp32Ogg,il faut aussi installer les paquets suivants :

```
-libmp3-info-perl
-liboggflac3
-libstring-shellquote-perl
-mpg321 vorbis-tools
```

C.4.5 Fonctionnement

Par exemple se placer sous un répertoire contenant des fichiers MP3 et affichons les options de Mp32Ogg :

```
root@zeus:/home/ilies# mp32ogg --help
   mp32ogg v0.11
   (c) 2000-2002 Nathan Walp
   Released without warranty under the terms of the Artistic License

   Usage: /usr/bin/mp32ogg [options] dir1 dir2 file1 file2 ...

Options:
   --quality=[-1..10]       Set Ogg/Vorbis quality level
                               Defaults to bitrate of original .mp3
   --delete                    Delete files after converting
   --rename=format         Instead of simply replacing the .mp3 with
                                  .ogg for the output file, produce output
                                  filenames in this format, replacing %a, %t
                               and %l with artist, title, and album name
                               for the track
   --lowercase              Force lowercase filenames when using --rename
   --verbose                Verbose output
   --preserve-timestamp     Preserve file timestamp
   --help                   Display this help message
```

Conversion d'un fichier Mp3 en Ogg

```
root@zeus:/home/ilies/Document/Music/gravure# mp32ogg --quality=10 --rename=%t          SCA

mp32ogg v0.11
(c) 2000-2002 Nathan Walp
Released without warranty under the terms of the Artistic License

Converting SCARFACE FUNK - Track 12 (1).mp3 to OGG...
./Track_07.ogg done!
```

Nous obtenons après cette manipulation un fichier Ogg écoutable qui au niveau qualité audio est équivalent au fichier MP3 original.

Cette conversion au format Ogg amène à penser directement que notre décodeur prend en entrée que le format Ogg pour avoir en sortit le même format de fichier. Afin de permettre aux utilisateurs de poster des documents MP3 dans notre JukeBoxe la conversion au format Ogg se fera au moment où le document sera posté et avant qu'il soit stocké dans la base de données.

Le script fait appel à des outils externes qu'on a installé en même temps que Mp32Ogg :

– pour le codage, oggenc (.ogg) qui se trouve dans le package mpg321 vorbis-tools,
– pour la lecture des tags des fichiers sources on fait appel à la bibliothèque PERL Audio : :File (libmp3-info-perl).

C.4.6 Intégration de Mp32Ogg dans le JukeBoxe

Nous allons intégrer la commande Mp32Ogg à notre Jukeboxe grâce à JAVA qui nous permet de lancer une commande Linux(ou autre) à travers un programme JAVA.

Exemple simple de ce qui vient d'être dit :

```
import java.applet.* ;
import java.awt.* ;
import java.io.*;

public class Mp32Ogg {

      public static void main(String[] args)
     {
          String cmd;
      //conversion du fichier MP3 music.mp3 en fichier Ogg
            cmd = "mp32ogg --quality=10 --rename=%t music.mp3";
        try
          {
            Runtime r = Runtime.getRuntime();
           Process p = r.exec(cmd);
            p.waitFor();
          }
          catch(Exception e)
          {
            System.out.println("erreur d'execution " + cmd + e.toString());
          }
      }
}
```

L'intégration de Mp32Ogg dans notre JukeBoxe se fera de cette façon simple.

C.4.7 Conclusion

Nous prévoyons donc pour la prochaine version du JukeBoxe l'intégration d'un plugin permettant d'accepter de la part des auteurs de chansons et des distributeurs des fichiers MP3 qui seront convertient en fichier Ogg afin de profiter eux aussi d'un format de compression libre et ouvert.

C.5 RTP/RTSP

C.5.1 RTP

C.5.1.1 Introduction

RTP veut dire Real-time Transport Protocol.

Le réseau de test de l'ARPA (DARTNET) qui regroupait plusieurs sites fut à l'origine des expériences autour des outils MBONE et RTP au début des années 90. Depuis 1993, le MBONE permit d'expérimenter des applications de transmission audio et vidéo temps réel.

Van Jacobson au Network Research Group du LBNL (Lawrence Berkeley National Laboratory) développa l'outil VAT (Visual Audio Tool). Ce sont les protocoles utilisés au sein de cet outil qui furent à l'origine de RTP.

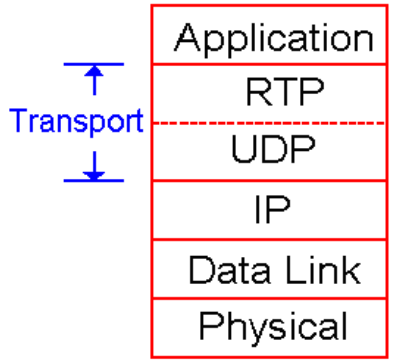
VAT ne traitant que la partie audio, VIC (adapté de l'IVS : 'INRIA Videoconferencing System) fut développé pour la vidéo ainsi que d'autres outils tels que le WB (White Board) pour le partage des données et le répertoire de session SD.

RTP devient un standard en 1996 par le groupe de travail AVT-WG (Audio Video Transport) de l'IETF (Internet Engineering Steering Group). Plusieurs constructeurs et éditeurs se rallient (Intel, Microsoft, Netscape...). L'ITU l'intègre dans H.323.

C.5.1.2 Les bases

RTP se situe au dessus de la couche UDP.

Les données audio ou vidéo sont encapsulées dans un packet RTP et chaque paquet RTP est a son tour encapsulé dans un paquet UDP. Comme RTP fournit des services/informations(numéro de séquence,le nombre de paquets8230 ;) à l'application multimedia,il peut être vu comme une sous couche de la couche transport comme dans la figure ci-dessous.



Donnons un exemple de l'utilisation de RTP dans le transport de la voie. Supposons que l'application serveur envoie des flux audio à partir d'une source(fichier,micro ou autre) à une application cliente. Selon le codage utilisé(par exemple PCM)l'application serveur découpe le flux source en petit morceau puis ajoute pour chaque morceau un entête RTP qui inclut le type du codage audio,le numéro de séquence et un compteur.Le morceau audio plus l'entête RTP forme le paquet RTP. Ensuite le paquet RTP est envoyé à l'interface socket ou il sera encapsulé dans un paquet UDP puis envoyé au client. Du côté client l'interface socket réception recevra le paquet UDP puis extrairera le paquet RTP pour l'envoyer à l'application cliente. Quand l'application cliente recevra ce paquet RTP,il extraira à son tour le morceau audio du paquet RTP tout en tenant compte de l'entête RTP pour pouvoir décoder proprement le morceau audio et l'utiliser/jouer convenablement.

Format du paquet envoyé au client :

En conclusion si une application utilise RTP, à la place d'autre solution propiétaire , elle peut facilement coopérer avec d'autres applications réseaux différentes.Par exemple si deux entreprises développent un logiciel internet téléphonique et si tous les deux incorpores le protocole RTP dans leurs produits,il est possible pour un client de communiquer avec un autre client même s'il utilise des réseaux téléphoniques et des sytèmes différents.

RTP permet pour chaque source (camera,microphone8230 ;) d'assigner son propre flux de paquet avec le protocole RTP.Par exmple pour une videoconference entre 2 participants, quatre flux RTP peuvent être ouverts :

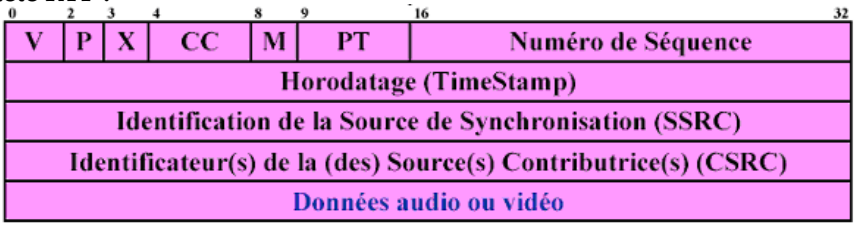
- 2 flux pour la transmission audio(1 pour chaque direction)
- 2 pour la transmission video(1 pour chaque direction)

Cependant des technologies comme le MPEG1 et MPEG2 permettent d'empaqueter le flux audio et video dans un même flux durant le processus d'encodage. Une fois que cet encodage est fait on peut envoyer qu'un seul flux RTP dans chaque direction.

Les paquets RTP ne sont pas limités par les applications unicast.Ils peuvent être envoyé à travers 1 à plusieurs et de plusieurs à plusieurs arbres. Pour plusieurs à plusieurs session multicast,tous les

émétteur et sources dans la session envoient leurs flux RTP vers le même arbre multicast avec la même adresse multicast.Les flux RTP multicast comme les flux audios et videos venant de plusieurs émétteur dans une videoconférence par exemple appartiennet tous à la même session RTP.

Entête RTP :



`version V:`

Ce champ identifie la version de RTP, la version actuelle est la version 2.

`padding P:`

Si le bit de padding est mis à 1, le paquet contient un ou plusieurs octets de rembourrage à la fin qui ne font pas partie des données.

`extension X:`

Si le bit d'extension est mis à 1, l'entête est suivie par une seule entête d'extension.

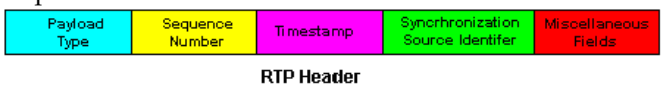
`CSRC count CC:`

Le compteur des CSRC contient le nombre des identificateurs CSRC qui suivent l'entête de taille fixe, c'est à dire qui suivent les trois premiers mots de 32 bits de l'entête RTP.

`marker M:`

L'interprétation du bit M est définie par l'application à travers un profile spécifique à cette application. Il existe un profile pour l'audio et la vidéo,

L'entête RTP se compose de 4 champs essentiels tels que le payload type,le sequence number,le timestamp et le source identifier :



RTP Header

`Payload type :`

Il est composé de 7 bits.Il y a donc 128 type de payload différent supporté par RTP. Pour un flux audio le payload type est utilisé pour indiquer le type d'encodage (PCM,MPEG8230 ;). Ce champ identifie le type du payload (audio, vidéo, image, texte, html, etc.) Si un émetteur décide de changer l'encodage au milieu de la session,l'émetteur informe le récepteur du changement d'encodage par le champ payload type.L'émetteur peut changer cette valeur pour augmenter ou diminuer la qualité du bit rate (débit binaire)du flux.

Voici les différents type de payload pour le flux audio :

Payload Type Number	Audio Format	Sampling Rate	Throughput
0	PCM mu-law	8 KHz	64 Kbps
1	1016	8 KHz	4.8 Kbps
3	GSM	8 KHz	13 Kbps
7	LPC	8 KHz	2.4 Kbps
9	G.722	8 KHz	48-64 Kbps
14	MPEG Audio	90 KHz	-
15	G.728	8 KHz	16 Kbps

Voici les différents type de payload pour le flux vidéo :

Payload Type Number	Video Format
26	Motion JPEG
31	H.261
32	MPEG1 video
33	MPEG2 video

Séquence number :

Ce champ est composé de 16 bits.Le séquence number est incrémenter de 1 à chaque envoi d'un paquet RTP par l'émetteur et peut être utilisé par le récepteur pour détecter si un paquet est perdu ou pour redemander un paquet RTP.Par exemple si du côté récepteur on reçoit un flux audio avec un espace entre le séquence number 86 et 89 ,cela signifie qu'on a perdu le paquet 87 et 88.A ce moment là ,le récepteur peut essayer de cacher les données perdu.

Timestamp field

Ce champ est composé de 32 bits. Il reflète l'instant où le premier octet du paquet RTP à été échantillonné. Cet instant doit être dérivé d'une horloge qui augmente de façon monotone et linéaire dans le temps pour permettre la synchronisation et le calcul de la gigue (les variations de délai)à la destination.

Synchronisation Source Identifier(SSRC) :

Il est composé de 32 bits.Il identifie la source du flux RTP.Chaque flux dans une session RTP possède un SSRC distinct. Le SSRC n'est pas l'adresse IP de l'émetteur mais à la place un nombre que la source assigne aléatoirement quand le nouveau flux a commencé. La probabilité que 2 flux possèdent le même SSRC est très faible.

C.5.1.3 Conclusion

RTP fournit des fonctions de transport de bout en bout pour les applications temps réel sur des services réseaux multicast ou unicast.

- conférence audio, vidéo interactive,
- diffusion vidéo, audio,
- simulation.

Mais il n'interfère pas dans la transmission. Il permet de :

- reconstituer la base de temps des différents flux multimédia (audio, vidéo...),
- détecter les pertes de paquets,
- identifier le contenu des paquets pour leur transmission sécurisée.

Il ne permet pas de :

- réserver des ressources dans le réseau,
- apporter une fiabilité dans le réseau,
- garantir le délai de livraison.

RTP peut s'appuyer sur différents protocoles. Dans l'architecture TCP/IP, il s'appuie sur UDP. Il fait partie intégrante de l'application contrairement à d'autres protocoles de transport comme TCP.

Dans une session mutimédia, chaque média est transporté dans des sessions RTP distinctes. Cela permet de s'adapter à la bande passante des destinataires : certains ne peuvent recevoir que l'audio. C'est grâce à l'identificateur de la source et à l'horodatage des échantillons que la synchronisation peut être assurée.

Enfin, RTP peut être vehiculé par des paquets multicast afin d'acheminer des conversations vers des destinataires multiples.

C.5.2 RTP Control Protocol (RTCP)

La partie contrôle du protocole RTP est spécifiée dans RTCP (Real-time Transport Control Protocol). Il assure un trafic de contrôle : c'est un "feedback" pour l'émetteur sur la qualité de transmission et d'autres informations . L'objectif de RTCP est de fournir différents types d'informations et un retour quant à la qualité de réception.

Le protocole RTCP est basé sur des transmissions périodiques de paquets de contrôle par tous les participants dans la session.

Quatre fonctions :

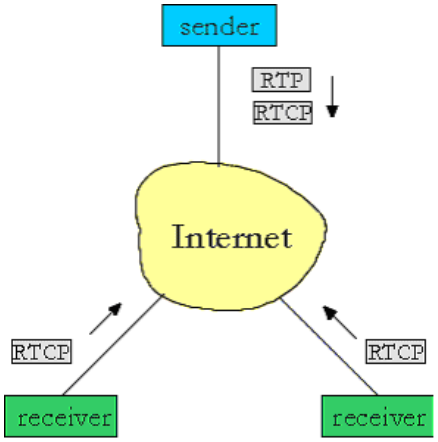
1. Fournir des informations sur la qualité de la session :

   (a) information en retour pour une source (feedback) permet à une source de changer de politique met en évidence des défauts de distribution individuels, collectifs.

2. Garder une trace de tous les participants à une session

(a) CNAME (Canonical Name) : identifiant unique et permanent pour un participant SSRC (Synchronisation Source identifier)

3. Contrôler le débit auquel les participants à une session RTP transmettent leurs paquets RTCP
   (a) Plus il y a de participants, moins la fréquence d'envoi de paquets RTCP par participant est grande. Il faut garder le trafic RTCP en dessous de 5% du trafic de la session
4. Transmettre des informations de contrôle sur la session (optionnel)
   (a) exemple : identifier un participant sur les écrans des participants



Les paquets RTCP n'encapsulent pas les morceau audio ou video.A la place les paquets RTCP sont envoyés périodiquement et contiennent les rapports de l'émétteur ou du récepteur qui décrivent les statistiques qui peuvent être utiles à l'application.

Ces statistiques incluent le nombre de paquets transmit,le nombre de paquets perdu...

Les spécifications RTP (RFC 1889) ne dicte pas ce que doit faire l'application avec ces informations.C'est la personne développant l'application qui décide de ce qu'il veut faire à la réception de ces informations.Par exemple l'émetteur peut utiliser ces informations pour modifier le débit de transmission.De plus ces informations dîtes « informations de feedback » peuvent permettre d'émettre des diagnostiques,par exemple le récepteur peut déterminer si les problèmes sont local,régional ou global.

C.5.2.1 Les types de paquets RTCP

Les paquets reçus par le récepteur

Pour chaque flux RTP que reçoit le récepteur en tant qu'élément de session, le récepteur génère un rapport de réception.Le récepteur empacte ce rapport dans un simple paquet RTP.Le paquet est envoyé en multicast pour atteindre touts les participants connectés ensemble dans la session.Ce rapport inclu plusieurs champs,les plus importants sont listés ci 8211 ;dessous :

- Le SSRC du flux RTP pour lequel le rapport de réception a été généré.
- La quantité de paquets perdu dans le flux.Cette quantité est donnée sous forme de fraction.Chaque récepteur calcule le nombre de paquets RTP perdu divisés par le nombre de paquets envoyés faisant partie du flux.Si un émetteur reçoit ce rapport indiquant que le récepteur a recu seulement une petite fraction des paquets transimt par l'émetteur ;alors l'émetteur à ce moment là peut changer le niveau d'encodage pour diminuer la congestion dans le réseau et ainsi augmenter le taux de réception.
- Le dernier nombre de séquence reçu dans le flux des paquets RTP.
- L'interval calculé entre les paquets reçu dans le flux RTP.

Le rapport de l'émetteur

Pour chaque flux RTP que l'émetteur envoie ,il crée un rapport et l'envoie.Ce rapport inclus des informations sur le flux RTP :

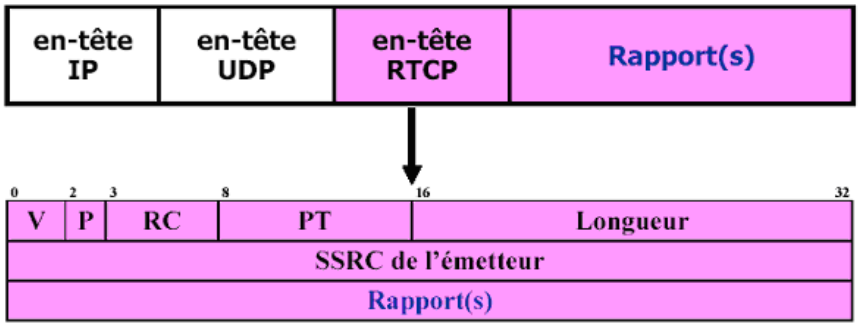
- Le SSRC du flux RTP
- L'horodateur et le « wall-clock » du paquet le plus récent émis dans le flux RTP.
- Le nombre de paquets envoyés dans le flux
- Le nombre de bits envoyés dans le flux

Le rapport de l'émetteur peut être utilisé pour synchroniser différent flux dans la session RTP.

RTCP définit les paquets suivants

- `sender report (SR)` : Le rapport envoyé par la (les) source(s).
- `receiver report (RR)` : Le rapport envoyé par la (les) destination(s).
- `Source description (SDES)` : La description d'une source.
- `Goodbye (BYE)` : Le paquet envoyé quand une source quitte la session.

L'en-tête d'un paquet RTCP



L'en-tête RTCP comportera les informations suivantes :

- Le champ version (2 bits)
- Le champ padding (1 bits) indique qu'il y a du bourrage dont la taille est indiquée dans le dernier octet
- Le champ reception report count (5 bits) : nombre de compte-rendus dans le paquet
- Le champ packet type (8 bits) 200 pour SR
- Le champ length (16 bits) longueur du paquet en mots de 32 bits
- Le champ SSRC (32 bits) : identification de la source spécifique à l'émetteur
- Le champ NTP timestamp (64 bits)
- Le champ RTP timestamp (32 bits)
- Le champ sender's packet count (32 bits)
- Le champ sender's octet count (32 bits) statistiques
- Le champ SSRC-n (32 bits) numéro de la source dont le flux est analysé
- Le champ fraction lost (8 bits)
- Le champ cumulative number of packets lost (24 bits)
- Le champ extended highest sequence number received (32 bits)
- Le champ interarrival jitter (32 bits). C'est une estimation de l'intervalle de temps d'un packet de donnés RTP qui est mesuré avec le timestamp et qui est sous forme d'un entier. C'est en fait le temps relatif de transit entre deux paquets de donnés.
  La formule pour le calculer est : J=J+(|D(i-1,i)|-J)/16 L'interarrival jitter est calculé à chaque packet de donnée reçu par la source SSRC_n

```
i --> Premier paquet
i-1 --> paquet précédent
D --> différence
J --> Second paquet
```

- Le champ last SR timestamp (32 bits)
- Le champ delay since last SR (32 bits)

C.5.2.2 Conclusion

RTCP est un protocole de contrôle associé à RTP, il mesure les performances, par contre il n'offre pas de garantie. Pour cela il faut, employer un protocole de réservation du type RSVP ou bien s'assurer que les liens de communications utilisés sont correctement dimensionnés par rapport à l'utilisation qui en est faite.

C.5.3 RTSP(Realtime Transport Streaming Protocole)

RTSP (Real Time Streaming Protocol) permet de contrôler la distribution de flux multimédias (streaming) sur un réseau IP. C'est un protocole de niveau applicatif prévu pour fonctionner sur des protocoles tels que RTP/RTCP et RSVP.

Les flux peuvent provenir soit de clips stockés soit d'une source temps réel (caméra, micro).

RTSP a été développé par Real Networks, Netscape et l'Université de Columbia. Il est implanté sur les produits de ces sociétés.

C.5.3.1 Quelles sont les fonctions de RTSP ?

RTSP offre des fonctions de type magnétoscope à distance (lecture, pause, avance rapide, rembobinage rapide, arrêt...). Il peut être utilisé pour rechercher un média sur un serveur de médias, inviter un serveur de médias à rejoindre une conférence (dans le e-learning par ex), ou ajouter un média à une présentation existante.

RTSP peut être utilisé aussi bien dans des applications unicast que multicast.

RTSP peut contrôler et synchroniser plusieurs flux audio ou vidéo. Il ne fournit pas lui-même le flux qui est à la charge d'autres protocoles comme RTP.

RTSP est similaire, au niveau de la syntaxe et des fonctionnalités, à HTTP. Il utilise les URLs et les mécanismes complémentaires de HTTP peuvent être utilisés. Mais il y a deux différences essentielles : RTSP est un protocole à états nécessaire pour pouvoir corréler les demandes RTSP avec un flux et dans RTSP le client et le serveur peuvent exprimer des requêtes.

Il n'y a pas de notion de connexion dans RTSP, bien que le serveur maintient une session ayant un identificateur. Une session RTSP ne correspond pas à une connexion de transport comme TCP. Durant une session, RTSP peut ouvrir et fermer plusieurs connexions de transport à chaque requête. Il est aussi possible d'utiliser un protocole de transport tel qu'UDP.

C.5.3.2 Comment fonctionne RTSP ?

Chaque présentation et chaque flux média est identifié par une URL RTSP. Une présentation peut contenir plus d'un flux média. Le fichier de description de la présentation contient les codages, langage et d'autres paramètres qui permettent au client de choisir la combinaison la plus adéquate de médias.

```
<session>
   <group>
      < track src="rtsp://audio.example.com/twister/audio.en">
      < track src="rtsp://video.example.com/video">
   </group>
</session>
```

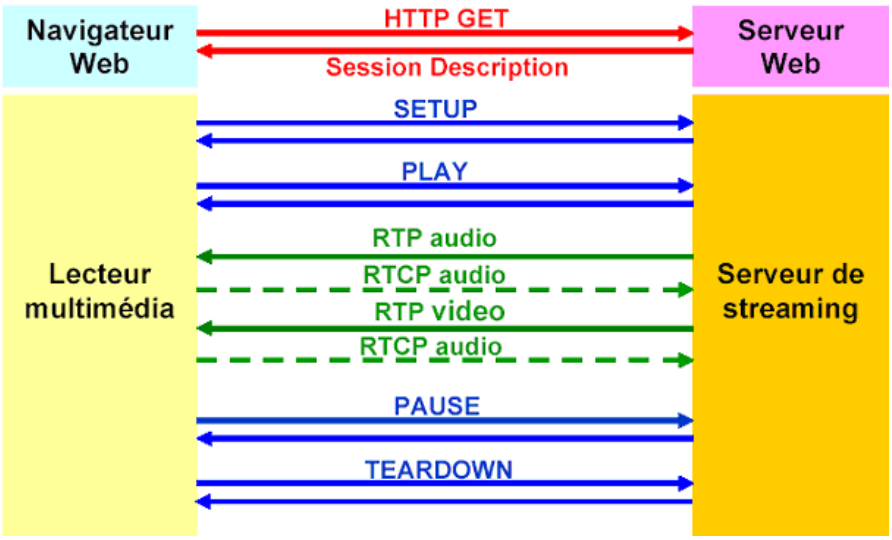
C.5.3.3 Exemple d'un échange Client-Serveur

Parmi les méthodes ci-dessus, seulement 5 contribuent à définir l'état de la session :

– SETUP : Le client demande au serveur l'allocation des ressources pour un flux et commence une session RTSP,

– PLAY : Le client demande la transmission des données d'un flux alloué selon SETUP,
– RECORD : Le client initie l'enrgistrement selon les paramètres de la description de la présentation,
– PAUSE :Le client arrête temporairement le flux sans libérer les ressources du serveur,
– TEARDOWN : Le client demande de libérer les ressources associées au flux. La session RTSP cess d'exister sur le serveur.

C.5.3.4 Fonctionnement



C.6 LOG4J pour la journalisation

C.6.1 Introduction

Dans la prochaine version de notre JukeBoxe nous allons généraliser l'implémentation d'un système de journalisation pour deux raisons essentielles :

– Journaliser l'application c'est-à-dire donner des informations sur le déroulement et l'état de notre application en temps réels.
– Aide pour debugger notre application

Dans cette version de notre JukeBoxe nous avons mit en place un système d'Exception JAVA permettant de répondre qu'à une partie de nos exigences,avec le système d'exception on ne pouvait avoir que des retours lors d'un problème dans notre application.

De plus pour le debuggage il est plus difficile et plus long avec le système d'exception qu'avec un outil de journalisation car les messages envoyés par les exceptions Java sont souvent difficiles à comprendre et ne donne pas directement l'endroit exacte où a eu lieu le problème.

Nous avons choisit l'API (Application Programming Interface) de journalisation Log4J pour plusieurs raisons :

– Outil Open Source et gratuit, ce qui respecte bien la politique adopté pour notre JukeBoxe qui n'utilise que des technologies OpenSource.
– Génération de logs à partir de notre code
– Configuration par niveau (par degré) la sortie et le formattage des logs à l'aide de fichiers de configuration externes à l'application.

Log4J est un système de journalisation pour java de Jakarta.

C.6.2 Concepts fondamentaux

Cette API utilise 3 classes JAVA :

– le `Logger` qui envoie les logs à un `Appender` qui les affiche selon un `Layout`.

La classe LOGGER

Elle est responsable de la majorité des opérations de logs. On associe à un LOGGER un niveau minimum de LOG,et il n'enverra à l'Appender que les logs qui correspondent à ce niveau ou à un niveau

supérieur. Il y a au moins un logger par application (le RootLogger de niveau Debug par défaut) mais il peut aussi y en avoir plusieurs selon un principe père-fils. Log4j gère les Logger de façon hiérarchique. C'est à dire qu'un Logger peut avoir des enfants et des parents. La hiérarchie est gérée par le nom des Logger. Les niveaux sont définis par des points, un peu comme la structure des packages de classes en JAVA.

La classe APPENDER

Elle est responsable de diriger les logs vers un flux de sortie. Il existe donc des appenders spécialisés pour les fichiers (FileAppender), la console (ConsoleAppender), les bases de données (JDBCAppender)... Elle a le rôle de filtrer les messages et de déterminer le mode de sortie des messages selon l'implémentation.

La classe LAYOUT

Cette classe est chargée de mettre en forme l'affichage des logs.Les Layout définissent le format du message. Je peux par exemple avoir la date, l'heure, le nom de la classe qui envoie le message et le niveau de logging au début de chaque message.C'est le rôle du Layout de faire ça et non celui du développeur qui envoie le message.

Les différents niveaux des messages envoyés par un LOGGER

La notion de niveau de journalisation ou de priorité d'un message est souvent un fonctionnement commun à plusieurs outil de journalisation .L'importance du message à journaliser est représentée par la classe org.apache.log4j.Level. Un message n'est journalisé que si sa priorité est supérieure ou égale à la priorité du Logger effectuant la journalisation.

L'API Log4j définit 5 niveaux de logging présentés ici par gravité décroissante :

- FATAL : utilisé pour journaliser une erreur grave pouvant mener à l'arrêt prématuré de l'application,
- ERROR : utilisé pour journaliser une erreur qui n'empêche cependant pas l'application de fonctionner,
- WARN : utilisé pour journaliser un avertissement, il peut s'agir par exemple d'une incohérence dans la configuration, l'application peut continuer à fonctionner mais pas forcément de la façon attendue,
- INFO : utilisé pour journaliser des messages à caractère informatif (nom des fichiers, etc.),
- DEBUG : utilisé pour générer des messages pouvant être utiles au débogage.

C.6.3 Installation

Il faut tout d'abord aller sur le site de la fondation Apache pour télécharger log4j http ://logging.apache.org/log4j/docs/download.html

Télécharger la dernière version stable. Ensuite extraire l'archive dans n'importe quel répertoire de votre disque dur.

L'archive devrait avoir la structure de répertoires suivante :

- logging-log4j-x.x.x
- build
- contribs
- dist
- docs
- exemples
- src

Le répertoire dist devrait contenir un sous-répertoire lib. Ce dernier devrait contenir le fichier log4j-x.x.x.jar (remplacez x.x.x par le numéro de version)

Pour installer log4j, il suffit d'inclure le fichier log4j-x.x.x.jar dans le classpath. Dans le cas d'une application web avec Tomcat, c'est dans le répertoire /WEB-INF/lib

C.6.4 Configuration

Nous présentons la configuration de Log4J dans notre prochaine version du JukeBoxe.

La configuration peut se faire de différentes manières mais pour notre JukeBoxe nous allons utiliser un fichier externe de configuration qui nous permettra d'être modifié pour configurer les traitements des logging sans avoir à toucher le code source de notre JukeBoxe.

On créer un fichier "Log4J.properties" qui est sauvegarder dans le classpath de notre application.Comme nous utilisons le conteneur de servlet Tomcat pour notre application nous sauvegardons le fichier sous $TOMCAT_HOME/WEB-INF/classes/

Voici un extrait de notre fichier de configuration

```
#définition du niveau et des Appender du rootLogger
log4j.rootLogger=DEBUG, monAppender

#configuration de "monAppender"
#nous allons envoyer les messages dans la console de Tomcat
log4j.appender.monAppender=org.apache.log4j.ConsoleAppender

#définition du Layout pour "monAppender"
log4j.appender.monAppender.layout=org.apache.log4j.PatternLayout

#définition du pattern d'affichage pour "monAppender"
#voici un exemple de sortie que l'on va obtenir : 2005-06-18 14:53:37 DEBUG   [Main] Hello
log4j.appender.monAppender.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss}   %-5p [%c{1}]
```

Explication du fichier de configuration

Il faut savoir qu'au niveau du fichier de configuration, chaque Appender doit avoir un nom afin de pouvoir y faire référence lors de la configuration des Loggers. Les paramètres de configuration concernant les Appenders sont préfixés par log4j.appender.

La déclaration d'un Appender d'un nom donné se fait de la façon suivante : `log4j.appender.monAppender=org.apache.log4j.ConsoleAppender`

Ici l'Appender nécessite des paramètres comme un Layout, il faut les lui passer en utilisant la syntaxe suivante : `log4j.appender.monAppender.layout=org.apache.log4j.PatternLayout`

C.6.5 Utilisation

Utilisation de Log4J dans la classe exemple Mp32Ogg qui convertie un fichier MP3 en fichier Ogg

```
import java.applet.* ;
import java.awt.* ;
import java.io.*;

//package nécessaire pour apeller le Logger
import org.apache.log4j.Logger;

public class Mp32Ogg {

   /**
             *déclaration et création du Logger
             *
   *obtention d'une instance de Logger se fait en appelant la méthode statique
            */
   private final static Logger  logger = Logger.getLogger(Mp32Ogg.class);

    public static void main(String[] args)
     {
          String cmd;
      //conversion du fichier MP3 music.mp3 en fichier Ogg
```

```
            cmd = "mp32ogg --quality=10 --rename=%t music.mp3";
        try
          {

            logger.debug("La conversion a commencé");

            Runtime r = Runtime.getRuntime();
           Process p = r.exec(cmd);
            p.waitFor();
          }
          catch(Exception e)
          {
            System.out.println("erreur d'execution " + cmd + e.toString());
          }
      }
  }
```

C.6.6 Conclusion

Avec Log4J notre JukeBoxe aura la technologie nécessaire pour contrôler le fonctionnement correcte de celui-ci mais il permettra aussi de tenir des rapports formatés de logs.

C.7 Présentation de LDAP

C.7.1 Les annuaires

C.7.1.1 Les annuaires électroniques

Un annuaire électronique est un catalogue de données dont le but premier est de proposer, grâce à des fonctions de recherche, un accès, rapide à ses ressources aux différents clients qui les consulte. (Définition de Laboratoire supinfo Sun)

Les annuaires électoniques stockent donc des données sur lesquelles il est possible d'effectuer des opérations de :

- comparaison
- création
- modification
- suppression

L'objectif d'un annuaire est de faciliter la localisation de tous types d'objets tels des :

- personnes
- organisations
- ressources informatiques
- applications

Ils ont trois caractéristiques principales, ils sont :

- Dynamique :

Tout changement sur les annuaires électroniques s'effectuent en temps réels. La responsabilité de la mise à jour de l'annuaire est délégué à des administrateurs et, si le droit de modification leur est donné, aux propriétaires des informations. Les coûts de mise à jour sont par conséquent très faibles.

- Flexibles :

Dans un annuaire électronique, sa structure n'est jamais figée, elle peut être modifiée facilement, à la volée, sans nécessiter de reconstruire tout l'annuaire.Il est toujours possible d'ajouter de nouveux champs ou de nouvelles familles d'objets.

- Sécurisé :

Le contrôle des informations affichées se font en fonction de l'identité de l'utililisateur.

C.7.1.2 Organisation d'un annuaire

Contrairement aux bases de données conventionnelles, un annuaire électronique est une base de données spécialisée, elle diffère par le fait que :

– elle est conçue pour recevoir beaucoup plus de requête en lecture qu'en écriture.
– Les données sont stockées de manière hiérarchique et non relationnelles, on pourrait faire l'analogie de la structure d'un annuaire à la structure d'un système de fichiers sous linux.

Voici un exemple de représentation qui s'adapte à notre projet :

Par conséquent la recherche d'informations dans un annuaire Ldap ne comporte pas de requêtes compliquées à l'inverse des bases de données Sql (jointures), de plus les annuaires peuvent communiquer entre eux.

C.7.1.3 Utililisation des annuaires

Les annuaires peuvent être utilisé pour plusieurs fonctionnalités :
– Recherche de personnes
– Recherche de ressources
– constitution de carnets d'adresses
– définition de droits d'accès à des utilisateurs
– recensement des informations sur un parc matériel
– description des applications
– stockage et diffusion des certificats dans une infrastructure de clé publique (PKI)

C.7.2 Le protocole Ldap

C.7.2.1 Introduction

LDAP (Lightweight Directory Access Protocol) est un protocole standard permettant de gérer des annuaire. Il a été développé en 1993 par l'université du Michigan, son objectif était de remplacer le protocole DAP en l'intégrant à la suite TCP/IP. Le protocole DAP servait à accéder au service d'annuaire X.500 de l'OSI.

Actuellement le protocole Ldap en est à la version 3 (LDAPv3), il a été normalisé par l'IETF. LDAPv3 est déféfini par neuf RFC : de 2251 à 2256 et 2829,2830 ainsi que 3377.

X500 Le standard X.500 a été établi pour normaliser les annuaires électroniques, quel que soit leur domaine d'apllication. L'objectif étant de mettre à disposition de l'industrie des télécommunication un standard capable de faire fonctionner ensemble une multitude d'annuaires à l'échelle mondiale, afoin de constituer un annuaire Pages Blanches et Pages Jaunes virtuel unique. Elle doit permettre à chaque pays de mettre à jour son propre annuaire et d'interroger les autres de la même manière.

Les caractéristiques exigées et implémentées sont les suivantes :
– l'ouverture : assurant une interconnexion des annuaires
– l'extensibilité : permettant de modifier simplement la structure des données tout en conservant la compatibilité avec la structure d'origine et d'être ainsi adaptable à tous les besoins.
– La distribution : rendant capable de répartir ou de répliquer les données sur plusieurs serveurs
Composition de l'annuaire X.500

Lexique :

DUA :Directory User Agent = client qui interroge l'annuaire

DAP : Protocol DAP, Directory Access Protocol = protocole de communication entre client et serveur X.500. Ce protocole s'appuie sur deux standards OSI qui sont ROSE (Remote Operations Service) et ACSE (Association Control Service).

DSA : Directory System Agent = serveur d'annuaire qui comprend la base de données DIB (Directory Information Base). Ce composant peut soit dialoguer avec des clients, soit avec d'autre DSA.

DSP : Directory Systme Protocol = protocole de communication entre deux serveur X.500, Similaire à DAP.

DISP : Directory Information Shadowing Protocol = protocole permettant la réplication d'un serveur DSA maître vers un autre serveur miroir.

De X500 à LDAP DAP est un protocole heavyweight (lourd) car il nécessite que le client et le serveur communique en utilisant le modèle OSI. Ce modèle de sept couches est beaucoup plus lourd que le modèle TCP/IP qui n'en comporte que quatre.

En 1993, l'université du Michigan a adapté le protocole DAP de la norme X.500 au protocole TCP/IP et mis au point LDAP.

Il est à l'origine une passerelle d'accès à des bases d'annuaires X.500. La première implémentation de LDAP contient le démoin de LDAP (ldapd) qui est une passerelle entre LDAP et DAP.

A partir de 1995, LDAP est devenu un annuaire natif (standalone LDAP), afin de ne plus servir uniquement de passerelle d'accès à des annuaires X500.

Standalone LDAP va gérer son propre mécanisme de sauvegarde de données qui va être incorporé au démon de LDAP, il s'agit de slapd (standalone ldap daemon).

LDAPv3 L'étape suivante dans le développement de LDAP était LDAPv3, cette version, entièrement compatible avec LDAPv2, vise à combler les limitatins de LDAPv2. Ces principaux ajouts sont :

- La prise en compte des caractères internationaux via le standard UTF-8
- La standardisation du mécanisme de chaînage des requêtes par renvoie de référence (refferals), lorsque l'infiormation est répartie sur plusieurs serveurs LDAP.
- La gestion de la sécurité poir l'authentification SASL (Authentification and Security Layer) et le transport des données confidentielles TLS (Transport Layer Security).
- La norme inclut maintenant des mécanismes d'extension. Il est possible de réaliser des opérationssupplémentair à celles décrites dans la norme, tout en s'appuyant sur le protocole existant. Il est possible, par le biais de contrôle, de modifier le comprtement des opérations de base.
- Un annuaire peut être interrogé pour accéder à son schéma, et pour connaître les extensions qu'il implémente.
- Intégration dans la norme LDAP du schéma X500. Certaines classes d'objets et attributs définis dans la norme X500 doivent être reconnus par les serveurs LDAP.

C.7.2.2 Les modèles

LDAP est défini par 5 modèles qui permettent de décrire différents aspects de l'annuaire : nommage, structure de stockage et structure hiérarchique, sécurisation et échange de données. Certains de ces modèles sont définis dans la norme comme le modèle d'échande de données ou de nommage. D'autres doivent être définis dans chaque annuaire comme le modèle d'authentification et le modèle d'information.

Le modèle d'information Le modèle d'information du protocole LDAP définit le type de données pouvant être stocké dans l'annuaire LDAP.

On appelle entrée (entry) l'élément de base de l'annuaire. Chaque entrée de l'annuaire LDAP correspond à un objet abstrait ou réel (par exemple une personne, un objet matériel, des paramètres, ...). Une entrée est constituée de plusieurs objets.

L'ensemble des paramètres qui définissent le type des données ainsi que leurs syntaxes forment ce qu'on appelle le schéma de l'annuaire.

Le schéma de l 'annuaire Ainsi, on appelle schéma (Directory Schema) l'ensemble des définitions d'objets et d'attributs qu'un serveur LDAP peut gérer ainsi que leur syntaxe. On peut donc définir des contraintes sur les entrées pour s'assurer de la validité des données insérées.

De cette façon, un annuaire peut uniquement comporter des entrées correspondant à une classe d'objet définie dans le schéma. Le schéma est en effet lui-même stocké dans l'annuaire à un emplacement spécifique (il s'agit pour être exact d'une instance de la classe subschema).

Grâce au schéma, l'annuaire peut garantir de façon autonome la validité des enregistrements et de leur syntaxe. Lorsqu'une entrée est créée dans l'annuaire, celui-ci vérifie sa conformité à la classe d'objet, on parle alors de schema checking.

Les attributs des entrées Chaque entrée est constituée d'un ensemble d'attributs (paires clé/valeur) permettant de caractériser l'objet que l'entrée définit. On distingue habituellement deux types d'attributs :

- Les attributs utilisateurs (user attributes) sont les attributs caractérisant l'objet manipulé par les utilisateurs de l'annuaire (nom, prénom, ...)
- Les attributs opérationnels (system attributes) sont des attributs auxquels seul le serveur peut accéder afin de manipuler les données de l'annuaire (dates de modification, ...)

LDAP permet de définir des types d'attributs, c'est-à-dire des caractéristiques permettant de le définir de façon précise. Chaque attribut possède de cette façon une syntaxe qui lui est propre (la façon selon laquelle l'attribut doit être renseigné, c'est-à-dire le format des données, mais aussi la

manière selon laquelle la comparaison doit s'effectuer lors d'une recherche de l'annuaire (par exemple définir si la recherche sera sensible à la casse, c'est-à-dire si la recherche devra différencier minuscules et majuscules.

Voici les principales syntaxes d'attributs définies dans le protocole LDAPv3 :

Voici les principales règles de comparaison d'attributs définies par le standard LDAPv3 :

Les attributs prédéfinis LDAP définit un ensemble de classes et d'attributs par défaut convenant pour la grande majorité des applications. Ces attributs doivent impérativement être implémentés par les serveurs d'annuaires LDAPv3. Cela permet de garantir une certaine homogénéité entre les différents annuaires.

Voici une petite liste non exhaustive des principaux attributs utilisateurs définis par le standard LDAPv3 :

Voici une petite liste non exhaustive des principaux attributs opérationnels définis par le standard LDAPv3

Les classes d'objets Par analogie avec la terminologie objet on parle de classe d'objet pour désigner la structure d'un objet, c'est-à-dire l'ensemble des attributs qu'il doit comporter. De cette façon on dira qu'un objet est une "instanciation" de la classe d'objet, c'est-à-dire un ensemble d'attributs avec des valeurs particulières.

Une classe d'objet est ainsi composée d'un ensemble d'attributs obligatoires (devant obligatoirement être renseignés dans les objets qui en découlent) et éventuellement des attributs facultatifs.

On distingue plusieurs types de classes d'objets :

- Les classes abstraites sont des classes non instanciables. Il s'agit de classes pouvant être dérivées, c'est-à-dire dont d'autres classes peuvent hériter. La classe d'objet de plus haut niveau étant la classe top dont toute classe d'objet dérive.
- Les classes structurelles sont des classes instanciables. Il est donc possible d'avoir des objets.
- Les classes auxiliaires sont des classes permettant d'ajouter des attributs facultatifs à des classes structurelles.

Une des caractéristiques intéressantes des classes d'objets LDAP est la possibilité d'utiliser l'héritage.

Ainsi la classe de plus haut niveau est la classe top dont toutes les classes d'objets dérivent. Avec LDAP seul l'héritage simple est autorisé (donc pas d'héritage multiple), c'est-à-dire qu'une classe ne peut dériver que d'une seule classe, mais qu'une classe peut avoir plusieurs filles.

Les attributs sont caractérisés par :

- leur nom unique
- un Object Identifier (OID) qui permet de les identifier de façon unique
- une syntaxe et des règles de comparaison
- un indicateur d'usage
- un format ou une limite de taille

Il s'agit d'utiliser une série de paires clé/valeur permettant de repérer une entrée de manière unique.

Voici une série de clés généralement utilisées :

- uid (userid) : identifiant unique obligatoire
- cn (common name) : nom de la personne
- givenname : prénom de la personne
- sn (surname) : nom usuel de la personne
- o (organization) : entreprise de la personne
- ou (organization unit) : service de l'entreprise dans laquelle la personne travaille
- mail : adresse de courrier électronique de la personne

Le modèle de nommage Le modèle de nommage (aussi appelé modèle de désignation) a pour but de définir la façon selon laquelle les objets de l'annuaire sont nommés et classés. Ainsi les objets LDAP sont classés hiérarchiquement et possèdent un espace de nom homogène. Cela signifie que d'un annuaire à un autre, un objet de la même classe possèdera le même nom afin de garantir une compatibilité (on parle d'interopérabilité) entre les annuaires.

L'arborescence d'information (DIT) LDAP présente les informations sous forme d'une arborescence d'informations hiérarchiques appelée DIT (Directory Information Tree), dans laquelle les informations, appelées entrées (ou encore DSE, Directory Service Entry), sont représentées sous forme de branches.

Une branche située à la racine d'une ramification est appelée racine ou suffixe (root entry).

Chaque entrée de l'annuaire LDAP contient à un objet abstrait ou réel (par exemple une personne, un objet matériel, des paramètres, ...). Ceci signifie que chaque noeud de l'arbre correspond à un objet pouvant appartenir à n'importe quelle classe d'objets. Les classes d'objets peuvent donc être utilisées comme à n'importe quel niveau de la hiérarchie et même à la racine de l'arbre.

Il existe une entrée particulière de l'annuaire appelée rootDSE (root Directory Specific Entry) contenant la description de l'arbre.

La désignation des entrées La norme LDAPv3 permet de désigner un objet de deux façons :
- grâce à son nom relatif (RDN - Relative Distinguished Name)
- grâce à son nom absolu (DN - Distinguished Name)

Le nom relatif (RDN) est composé d'une (ou plusieurs) paire(s) clé/valeur (attribut). Ainsi, un RDN sera de la forme cn=inpg ou bien c=fr.

Un RDN doit respecter certains critères :
- Un objet ne doit posséder qu'un et un seul RDN
- Le RDN doit être un nom unique dans la branche de l'objet (à un même niveau)
- Un RDN peut être composé d'un ensemble d'attributs. Le RDN est alors dit multivalué (par exemple cn=inpg+sn=uilisateursJukeBox)

Ainsi il est conseillé de faire en sorte que l'attribut servant de RDN soit obligatoire. Le DN (Distinguished Name) d'un objet est un moyen d'identifier de façon unique un objet dans la hiérarchie. Un DN se construit en prenant le nom relatif de l'élément (RDN - Relative Distinguished Name), et en lui ajoutant l'ensemble des noms relatifs des entrées parentes. Le DN d'un élément est donc la concaténation de l'ensemble des RDN de ses ascendants hiérarchiques.

Ainsi une entrée indexée par un nom absolu (DN, distinguished name) peut être identifiée de manière unique dans l'arborescence. Le nom absolu (DN) d'un objet ne comportant aucune information relative à la localisation de l'annuaire lui-même, la norme LDAPv3 recommande de le compléter par l'adresse DNS de l'annuaire.

Le modèle fonctionnel LDAP fournit, à travers le modèle fonctionnel, un ensemble de neuf fonctions (appelées parfois procédures ou opérations) de base pour effectuer des requêtes sur les données afin de rechercher, modifier, effacer des entrées dans les répertoires.

Les opérations sont généralement classées en trois catégories :
- les fonctions d'interrogation : il s'agit des opérations permettant de rechercher ou comparer des entrées de l'annuaire (recherche, comparaison).
- les fonctions de mise à jour : il s'agit des opérations permettant de modifier des entrées de l'annuaire (ajout, suppression, modification, renommage)
- les fonctions de session : il s'agit des opérations permettant d'ouvrir une session (s'identifier), de la fermer ainsi que d'annuler une requête.

Voici la liste des principales opérations que LDAP peut effectuer :

Les commandes search et compare se font sous la forme d'une requête composée de 8 paramètres :

Le scope définit la profondeur de la recherche dans le DIT. Il peut prendre différentes valeurs selon la portée de la recherche souhaitée :
- base : recherche dans le niveau courant
- one-level : recherche uniquement dans le niveau inférieur au niveau courant
- subtree : recherche dans tout le sous-arbre à partir du niveau courant

Il n'existe pas de fonction read dans LDAP. Cette fonction est simulée par la fonction search avec un search scope égal à base.

Le filtre de recherche s'exprime suivant une syntaxe spécifique dont la forme générale est : (< operator(< search operation)(< search operation)...)). Ce filtre décrit une ou plusieurs conditions exprimées sous forme d'expressions régulières sensées désigner un ou plusieurs objets de l'annuaire, sur lesquels on veut appliquer l'opération voulue.

Lors de la connexion au serveur (bind), ce dernier demande une authentification. Le client doit alors fournir un DN et le mot de passe correspondant, celui-ci transitant en clair. Pour sécuriser les transactions, LDAPv3 fournit la possibilité d'utiliser du chiffrement (SSL ou TLS) et le mécanisme Simple Authentification and Security Layer (SASL) procurant des outils d'authentification plus élaborés à base de clés.

Une fois connecté, le client peut envoyer autant de commandes qu'il souhaite jusqu'à ce qu'il ferme la session (unbind).

Chaque commande se voit attribuer un numéro de séquence, qui permet au client de reconnaître les réponses lorsque celles-ci sont multiples - ce qui peut parfois arriver lors d'une recherche simple qui peut renvoyer jusqu'à plusieurs milliers d'entrées. A chaque opération, le serveur renvoie également un acquittement pour indiquer que sa tâche est terminée ou qu'il y a une erreur.

Le format d'échange de données LDIF

LDAP fournit un format d'échange (LDIF, Lightweight Data Interchange Format) permettant d'importer et d'exporter les données d'un annuaire avec un simple fichier texte. La majorité des serveurs LDAP supportent ce format, ce qui permet une grande interopérabilité entre eux.

La syntaxe de LDIF est la suivante :

[<id>]
dn : <distinguished name>
<attribut> : <valeur>
<attribut> : <valeur>
...

Dans ce fichier id est facultatif, il s'agit d'un entier positif permettant d'identifier l'entrée dans la base de données.

- chaque nouvelle entrée doit être séparée de la définition de l'entrée précédente à l'aide d'un saut de ligne (ligne vide)
- Il est possible de définir un attribut sur plusieurs lignes en commençant les lignes suivantes par un espace ou une tabulation
- Il est possible de définir plusieurs valeurs pour un attribut en répétant la chaîne nom :valeur sur des lignes séparées
- lorsque la valeur contient un caractère spécial (non imprimable, un espace ou :), l'attribut doit être suivi de : : puis de la valeur encodée en base64.

Le modèle sécurité L'annuaire doit pouvoir être protégé contre des intrusions intempestives, et ce, de manière efficace. De plus, chaque acteur, suivant ses droits ne doit pouvoir effectuer que certaines actions. Les aspects de sécurité et confidentialité doivent être pris en compte dès la phase de conception.

Les aspects à étudier sont les suivants :

- Les accès non autorisés
- Les attaques de type denial-of-service
- Les droits d'accès aux données.

La tâche primordiale réside dans l'établissement des règles d'accès aux données. Il faut déterminer pour chaque attribut quel est son niveau de confidentialité et quel utilisateur ou quelle application pourra y accéder en lecture ou en écriture.

Les mécanismes qui peuvent être mis en oeuvre sont les suivants :

- L'authentification
- Les signatures électroniques
- La cryptographie
- Le filtrage réseau
- Les règles d'accès aux données (ACL ou access control list)
- L'audit des journaux

Les ACL permettent de décrire les habilitations de tout utilisateur référencé dans l'annuaire sur les autres objets de l'annuaire.

Authentification LDAP propose plusieurs choix d'authentification :

- Anonymous authentification : correspond à un accès au serveur sans authentification, qui permet uniquement de consulter les données accessibles en lecture pour tous.
- Root DN Authentification : Utilisateur privilégié. Il a accès en modification à toutes les données.
- Mot de passe + SSL : La session entre le serveur et le client est chiffrée et le mot de passe ne transite plus en clair
- Simple Authentification and Security Layer : permet de faire des mécanismes d'authentification plus élaborés à base de clés
- Certificats sur SSL : Echange de certificats (clefs publiques/privées)

Définition des droits d'accès Il s'agit de définir les droits d'accès des utilisateurs sur les ressources de l'annuaire (objets et attributs).

Préciser à qui appartiennent chaque attribut et chaque classe d'objet, ainsi que les droits d'accès des acteurs spécifiés. Ces droits comprennent essentiellement la lecture, la modification, la création, la suppression et la recherche.

Pour définir ces droits il faut commencer par lister les actions possibles :
- Rechercher et lister des données
- Comparer des données
- Modifier un objet
- Supprimer un objet
- Ajouter un objet
- Renommer le DN d'un objet

Protection réseau Toute la panoplie d'outils de sécurité est à la disposition de l'administrateur pour sécuriser les accès réseau au service et la confidentialité des transactions. LDAP supporte les protocoles SSL et TLS pour chiffrer les données qui transitent sur le réseau.

Le modèle de réplication La réplication est très importante pour plusieurs raisons. La première est la volonté d'assurer une disponibilité maximale des données gérées par l'annuaire. La seconde est l'optimisation des performances.

Cette optimisation d'accès à l'annuaire peut se faire de la façon suivante : plusieurs serveurs dédiés à la lecture et un seul dédié à l'écriture.

De plus, la réplication d'un serveur sur plusieurs serveurs peut pallier à :
- Une coupure du réseau
- Surcharge du service
- Une panne de l'un des serveurs

Une stratégie consiste à avoir un seul serveur maître sur le site principal qui centralise toutes les mises à jour. Cette stratégie est simple de mise en place. Le serveur maître va se charger des répliquer ses informations de manière régulière sur les serveurs dédiés à la lecture.

Toutefois si ce type de stratégie est simple à mettre en oeuvre, il y a deux inconvénients :
- Si le serveur maître tombe en panne, les mises à jour ne peuvent plus s'effectuer. Solution éventuelle : mise en place d'un doublon non accessible pour pallier à une éventuelle panne du serveur maître.
- Si le serveur maître est situé sur un site distant qui n'est pas relié en permanence avec les postes clients, la mise à jour ne pourra pas se faire à tout moment. Ce point n'aura pas de répercussion étant donné que le serveur maître est à proximité et en réseau avec les postes clients.

Les possibilités de réplication :
- L'arbre entier ou seulement un sous arbre
- Une partie des entrées et de leurs attributs qu'on aura spécifiés via un filtre.

La synchronisation des serveurs peut se faire de façon totale ou incrémentale. La réplication se fait en temps réel ou à heure fixe.

C.7.2.3 Communication LDAP Client-Serveur

Le dialogue entre un client et un serveur LDAP est basé sur un protocole de type client-serveur. Sa particularité est de reposer sur un mécanisme de questions et de réponses sous forme de messages traités par le serveur de façon synchrone ou asynchrone.

Dans le cas de communication synchrone le client attend la réponse avant de transmettre une nouvelle requête.

Dans le cas de communications asynchrones, un numéro de contexte est associé à chaque requête, permettant au serveur d'envoyer ses réponses sans contrainte d'ordre.

Dans ce cas, on observe que la réponse 2 arrive avant la réponse 1. Ceci permet donc d'optimiser le serveur si la réponse 2 est plus rapide à obtenir que la réponse 1.

C.7.3 Sources

- Labo-sun.com
- Université CFAI
- Comment ça marche

C.8 Authentification LDAP dans Tomcat

>

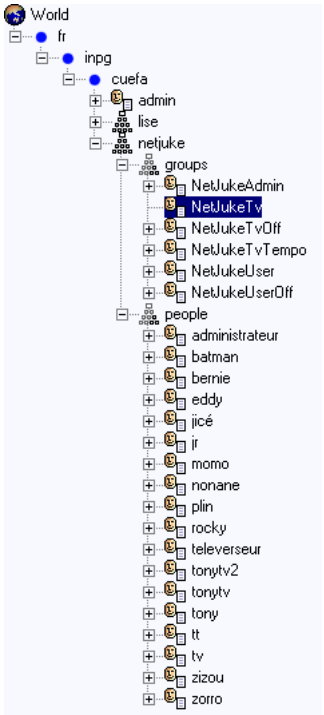
C.8.1 Relation Tomcat – LDAP

Tomcat utilise un mode d'authentification assez simple : il est composé d'un nom d'utilisateur, d'un mot de passe et le filtrage des différents types d'utilisateur est ensuite réalisé dans l'application grâce à des rôles. Ces rôles portent de simples nom que l'on teste pour voir si l'utilisateur dispose du rôle ou non. Un utilisateur (représenté par un nom) peut disposer de plusieurs rôles.

C.8.2 Création de l'annuaire LDAP

Pour répondre au schéma émi par Tomcat, nous allons créer une branche qui contiendra les utilisateurs et une branche qui contiendra les rôles. On cré donc une branche de type OrganisationalUnit pour les utilisateurs et une pour les groupes. Ensuite, chaque utilisateur sera une feuille de type inetOrgPerson et chaque rôle une feuille de type groupOfUniqueNames. Cette dernière permet d'associer à elle même des utilisateur et les référençant par plusieurs attributs "groupOfUniqueNames". Ces attributs contiendront le DN des utilisateurs concernés par ce rôle.

Voici une représentation de l'annuaire où la branche contenant les rôles est nommée "groups" et celle contenant les utilisateurs "people" :



C.8.3 Liaison Tomcat à l'annuaire LDAP

Pour lier Tomcat afin qu'il utilise un annuaire LDAP avec une Application Web, il faut créer un Royaume (REALM) dans le fichier XML du contexte de l'application. Pour Tomcat 4, le contexte se défini dans le fichier général de configuration : "server.xml" et pour Tomcat 5, le fichier se nomme context.xml et il se trouve dans le répertoire "META-INF" de l'application web. Voici un exemple du fichier context.xml :

```
<Context path="/NetJuke"
                    docBase="NetJuke"
                    crossContext="false"
                    debug="99"
                    reloadable="true"
                    trusted="false">
```

```
 <!--configuration du realm pour authentifier via LDAP-->
 <Realm className="org.apache.catalina.realm.JNDIRealm" debug="99"
  connectionURL="ldap://195.220.26.5:389"
  userPattern="uid={0},ou=people,o=netjuke,dc=cuefa,dc=inpg,dc=fr"
  roleBase="ou=groups,o=netjuke,dc=cuefa,dc=inpg,dc=fr"
  roleName="cn"
  roleSubtree="false"
  roleSearch="(uniqueMember={0})"
 />

</Context>
```

On remarque la définition d'une classe utilisée pour accéder au royaume, ici c'est la classe JNDIRealm, qui est utilisée pour se connecter à un annuaire LDAP. On aurait pu utiliser un royaume de type MemoryReal (classe : org.apache.catalina.realm.MemoryRealm) permettant d'authentifier les utilisateur via un fichier XML dont le format est imposé par Tomcat.

Ensuite on trouve l'url de connection au royaume, qui permet de trouver le serveur sur lequel est l'annuaire.

Puis le userPatern défini une chaine d'accès dans l'annuaire pour trouver les noms d'utilisateur (login). le {0} représente donc le login

Enfin arrive la définition des rôles, qui commence par dire où se trouvent les feuilles définissant les rôles. Ensuite RoleName explicite quel attribut de la feuille "groupOfUniqueNames" informe sur le nom du rôle, puis on indique si on recherche les rôle dans une sous arborescence de la branche défini plus haut. Enfin, on définit quel attribut doit contenir le login de l'utilisateur auquel est associé ce rôle.

C.8.4 Authentification HTTP d'une application Web

Après ces étapes, reste à donner à l'application Web le type d'authentification (balise login-config) et des contraintes d'identification qui permettrant d'accepter ou refuser des personnes sur certaines pages de l'application (balises security-constraint). Cela se fait dans le descripteur de déploiement : le fichier "web.xml" contenu dans le répertoire "WEB-INF" de l'application concernée. Voici un exemple de contraintes d'accès dans lequel la première balise security-constraint est commentée pour expliquer le fonctionnement :

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>NetJukeBox</display-name>
<!--
  Définit la zone réservée aux Membres, en définissant une "Security Constraint"
  sur cette Application, et en la redirigeant vers le sous-répertoire (URL) que
  nous souhaitons protéger.
 -->

<!--Contrainte d'accès pour les administrateurs-->
<security-constraint> <!-- on déclare une contrainte de sécurité pour les demandeurs d
  <web-resource-collection> <!-- on déclare une ressource à laquelle va s'appliquer la
    <web-resource-name> <!-- le nom de la ressource -->
      NetJukeAdmin
    </web-resource-name>
    <url-pattern>/connected/administrateur.jsp</url-pattern> <!--la page ou les pages
                                                              <!-- on pourrait utilise
  </web-resource-collection>
  <auth-constraint> <!--déclaration des rôles authorisés à accéder à la page -->
      <role-name>NetJukeAdmin</role-name> <!-- le nom d'un rôle -->
```

```
  </auth-constraint>
</security-constraint>

<!--Contrainte d'accès pour les utilisateurs-->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      NetJukeUser
    </web-resource-name>
    <url-pattern>/connected/utilisateur.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
      <role-name>NetJukeUser</role-name>
  </auth-constraint>
</security-constraint>

<!--Contrainte d'accès pour les televerseurs validés-->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      NetJukeTeleverseur
    </web-resource-name>
    <url-pattern>/connected/televerseur.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
      <role-name>NetJukeTv</role-name>
  </auth-constraint>
</security-constraint>

<!--Contrainte d'accès pour les televerseurs non validés-->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      NetJukeTeleverseurTemp
    </web-resource-name>
    <url-pattern>/connected/televerseurTemp.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
      <role-name>NetJukeTvTempo</role-name>
  </auth-constraint>
</security-constraint>

<!--Contrainte d'accès pour tous les membres identifiés ayant un rôle quelconque-->
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      NetJukeConnecteur
    </web-resource-name>
    <url-pattern>/connected/redirector.jsp</url-pattern>
  </web-resource-collection>
  <auth-constraint>
      <role-name>NetJukeTv</role-name>
      <role-name>NetJukeUser</role-name>
      <role-name>NetJukeAdmin</role-name>
      <role-name>NetJukeTvTempo</role-name>
  </auth-constraint>
</security-constraint>

<!-- Authentification HTTP simple -->
```

```
<login-config>
 <auth-method>BASIC</auth-method>

</login-config>

</web-app>
```

C.9 Documentation sur l'évolution vers Hibernate

C.9.1 Introduction

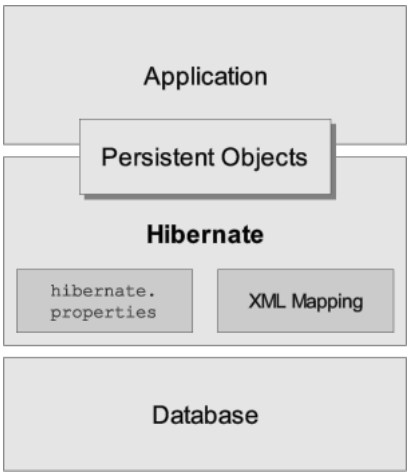
Hibernate est un outil de mapping objet/relationnel pour le monde Java qui permet de soulager les développeurs lorsqu'ils sont amenés à travailler avec l'orienté objet et la base de donnée relationnelle.

Le terme mapping objet/relationnel décrit la technique consistant à faire le lien entre la représentation objet des données et sa représentation relationnelle basée sur un schéma SQL.

Non seulement, Hibernate s'occupe du transfert des classes Java dans les tables de la base de données (et des types de données Java dans les types de données SQL), mais il permet d'exécuter des requêtes sur les données et propose des moyens de les récupérer. Il réduit ainsi de manière significative le temps de développement qui aurait été dépensé autrement dans une manipulation manuelle des données via SQL et JDBC.

Le but de Hibernate est de libérer le développeur de 95% des tâches de programmation liées à la persistance des données communes. Hibernate est très utile dans les modèles métier orientés objets dont la logique métier est implémentée dans la couche Java dite intermédiaire. Hibernate aide à supprimer et à encapsuler le code SQL spécifique d'une base de données aide sur la tâche commune qu'est la transformation des données d'une représentation tabulaire à une représentation sous forme de graphe d'objets.

Voici une vue haut niveau de l'architecture de Hibernate :

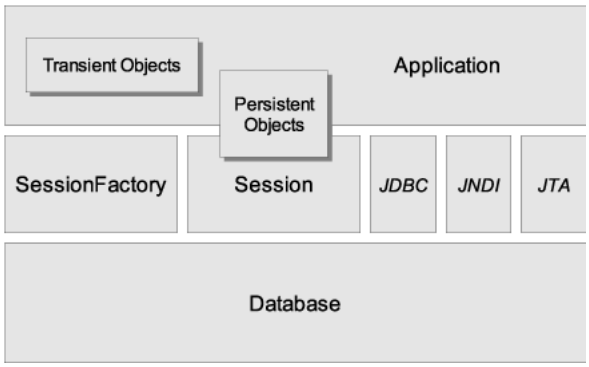


Ce diagramme montre Hibernate utilisant une base de données. Il présente aussi des données de configuration, qui fournissent un service de persistance à l'application.

La flexibilité de Hibernate et toutes les approches qu'il accepte, nous pose problème pour donner une représentation plus détaillée de son fonctionnement, c'est pour cela que nous allons présenter celle utilisée.

C.9.2 Architecture légère

L'architecture légère laisse l'application fournir ses propres connexions JDBC et gérer ses propres transactions. Cette approche utilise le minimum des APIs Hibernate. :



Parce que Hibernate est conçu pour fonctionner dans différents environnements, il existe beaucoup de paramètres de configuration. Heureusement, la plupart ont des valeurs par défaut appropriées et la distribution de Hibernate contient un exemple de fichier « hibernate.properties » ou « hibernate.hbm.xml » qui montrent les différentes options. Les options minimums à renseigner sont les paramètres de connexion et les noms des différents fichiers de mapping à charger.

Voici le contenu de notre fichier « hibernate.hbm.xml » :

<?xml version='1.0' encoding='utf-8' ?>

<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration DTD 3.0//EN" "http ://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.cglib.use_reflection_optimizer">false</property>

<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>

<property name="hibernate.connection.password">admin$23</property>

<property name="hibernate.connection.url">jdbc :mysql ://195.220.26.5/jukeboxe</property>

<property name="hibernate.connection.username">administrateur</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="show_sql">true</property>

<mapping resource="./mapping/Remuneration.hbm.xml"/>

<mapping resource="./mapping/Canal.hbm.xml"/>

<mapping resource="./mapping/DescripteurListe.hbm.xml"/>

<mapping resource="./mapping/DetailListe.hbm.xml"/>

<mapping resource="./mapping/DocumentAudio.hbm.xml"/>

</session-factory>

</hibernate-configuration>

Généralement, il n'y a qu'à placer ce fichier dans la classpath du projet et à l'adapter.

Pour ce connecter à la base, il faut d'abord obtenir une « SessionFactory ». Pour cela, il faut d'abord fournir à l'application une fabrique d'instance de « Session » qui est partagé par tous les threads de l'application.

Cependant, Hibernate permet à l'application d'instancier plus d'une « SessionFactory ». C'est utile si plusieurs bases de données sont utilisées.

Nous avons choisi que la connexions JDBC soit fournie par Hibernate et pour cela nous laissons la « SessionFactory » ouvrir les connexions. La « SessionFactory » reçoit les propriétés de connexions JDBC du fichier « hibernate.cfg.xml » qui incluent des éléments <property>

Une fois cette approche suivie, il ne reste plus qu'à ouvrir une session et à la retourner à la classe qui souhaite effectuer des accès aux données.

Voici le bout de code permettant ainsi d'obtenir une session :

```
SessionFactory sessionFactory;
sessionFactory = new Configuration().configure().buildSessionFactory();
this.session = sessionFactory.openSession();
return this.session;
```

C.9.3 Avantages et inconvénients

C.9.3.1 Avantages

1. Gain de temps si on utilise des outils pour générer automatiquement la base de données et le code. Mais ce n'est pas la seule possibilité et il devient assez rapide de le faire manuellement une fois

que l'habitude est prise.

2. Les objets métiers sont plus faciles à manipuler.
3. Peu de dépendance envers une base de données précise. Théoriquement, il n'y a que le fichier de configuration à changer si on passe d'une base de données à une autre à condition que Hibernate sache la mapper.
4. Hibernate est OpenSource
5. Efficace et fiable

C.9.3.2 Inconvénients

1. Nécessite d'apprendre à l'utiliser.

C.9.4 Bilan

Le mapping objet/relationnel permet de bénéficier des avantages de la programmation objet pour l'utilisation de données stockées en base et l'API Hibernate permet de faciliter ce type de programmation.

Toutefois, il nécessite une grande rigueur à la création de sa base de donnée relationnelle ou objet, et suppose de disposer de suffisamment de temps pour apprendre à l'utiliser et à choisir la bonne architecture parmi toutes celles proposées par Hibernate.

C.10 Mise en place d'une servlet avec Tomcat

C.10.1 Création de la Servlet Java

Une servlet Java est une Classe qui dérive de la classe mère HttpServlet
Cette classe doit surcharger deux méthodes de la classe mère dont les prototypes sont :

- protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException
- protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException

Les objets en paramètre représente les requetes et réponses HTTP. Ceci permet d'obtenir des paramètres depuis la requete et de renvoyer une réponse texte ou HTML via la réponse.

On pourra faire tout le traitement que l'on veut dans une servlet comme aller chercher des informations dans une base de donnée, ou dans un annuaire LDAP, ou envoyer des fichiers sur le serveur,...ou simplement répondre "bonjour".

Voici un exemple d'implémentation de servlet simple :

```
package org.test.servlet;

import javax.servlet.ServletException;
import java.io.IOException;
import javax.servlet.http.*;

public class BonjourServlet  extends HttpServlet{

 public BonjourServlet() {
  super();
 }
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
  response.setContentType("text/html");
  PrintWriter writer = response.getWriter();
  writer.println("Bonjour");
 }
 protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1) throws Servl
  //On fait la même chose pour les  méthodes http GET et POST
```

```
  this.doGet(arg0,arg1);
 }
}
```

Après ça, on compile et on obtient le fichier Java compilé (un .class)

C.10.2 Lier la servlet dans Tomcat

Il s'agit de créer un application Web dans Tomcat, puis de lier la servlet grâce au fichier "web.xml" situé dans le répertoire "WEB-INF" de l'application Web.

Voici un aperçu du fichier web.xml avec la liaison sur la servlet :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://

<web-app>
 <display-name>BonjourServletApplication</display-name>

 <servlet>
  <servlet-name>BonjourServlet</servlet-name>
  <servlet-class>org.test.servlet.BonjourServlet</servlet-class>
 </servlet>

 <servlet-mapping>
  <servlet-name>BonjourServlet</servlet-name>
  <url-pattern>/BonjourServlet</url-pattern>
 </servlet-mapping>
</web-app>
```

Il faudra ensuite placer la fichier java compilé (.java) dans le répertoire "WEB-INF/classes" (le créer si nécessaire). après quoi il ne reste plus qu'à lancer Tomcat et intéroger la Servlet depuis un navigateur avec l'URL : "http ://localhost/BonjourServletApplication/BonjourServlet"

C.11 Subversion

C.11.1 Introduction

Afin de mener à bien un projet, nous nous devons en tant que meneur de projet et développeur d'utiliser des outils adéquates afin de gagner en productivité et de rendre le cycle de développement bien plus agréable.

Selon la version française de Wikipedia, La gestion de version (en anglais revision control) est une activité qui consiste à maintenir l'ensemble des versions d'un logiciel. Essentiellement utilisée dans le domaine de la création de logiciels, elle est surtout concernée par le code source ; mais elle peut être utilisée pour tout type de document informatique.

Toute les commandes et procédures présentes dans ce document concerne les systèmes de type **Debian**

C.11.2 Installation

Il faut savoir que nous comptons utiliser subversion au travers d'apache pour nous faciliter l'utilisation, la prise en main et uniformiser notre politique de sécurité sur notre serveur.

C.11.2.1 Coté serveur

Dans un premier temps, il faut installer les paquets adéquates sur le serveur. Je pars du principe qu' **Apache 2** est installé et configurer correctement.

```
beytu@lise:~$
sudo apt-get install subversion libapache2-svn
```

Il faut ensuite créé et configurer notre dépôt :

```
root@lise:~$
mkdir /home/svn
root@lise:~$
svnadmin create /home/svn
root@lise:~$
chown -R apache /home/svn
root@lise:~$
chgrp -R apache /home/svn
```

Configurons à présent apache. Il faut dans un premier temps activer les modules nécessaires.

```
root@lise:/etc/apache2/mods-enabled$
ln -s ../mods-available/dav* .
```

Modifions à présent le fichier **/etc/apache2/mods-enabled/dav_svn.conf** afin d'indiquer le répertoire d'accueil de **subversion**

```
<Location /svn>
  DAV svn
  SVNPath /home/svn
</Location>
```

Relançons notre serveur apache :

```
root@lise:~#
/etc/init.d/apache2 reload
* Reloading web server config... [ ok ]
root@lise:~#
```

C.11.2.2 Coté client

Il faut installer au préalable subversion :

```
beytu@poseidon:~$
sudo apt-get install subversion
```

Ensuite, il faut réaliser notre première importation :

```
beytu@poseidon:~$

   svn import ~/Documents/jukeboxe http://195.220.26.5/svn -m "initial import"
```

C.11.3 Utilisation

C.11.3.1 Première utilisation

Avant de pouvoir profiter pleinement de subversion, il faut commancer par taper la commande qui suit afin de synchroniser un dépôt subversion distant avec un répertoire local :

```
beytu@poseidon:~/Documents$
svn checkout https://[serveur]/svn [destinationn]
```

C.11.3.2 Les commandes de base

svn update Commande essentiel permettant de synchroniser le dépôt distant avec le dépôt local. Nous rapatrions donc la dernière revision des fichier concernés

svn status Cette commande permet à tout moment de connaître l'état actuel du dépôt local. Nous pouvons ainsi connaître la liste des fichiers modifiés, ajoutés ou supprimés. La commande **svn commit** permet d'appliquer les modifications. Cette dernière commande sera vu plus loin
Ci dessous un exemple :

```
beytu@proteus:~/Documents/DEST/jukeboxe$
svn status

   ? tmp/archive ? documentation/src/annexes/compte_rendu_-_20060114.xml ?
   documentation/src/annexes/compte_rendu_-_20060107.xml M
   documentation/src/annexes/svn.xml

beytu@proteus:~/Documents/DEST/jukeboxe$
```

Dans ce cas, nous pouvons voir qu'un document a été modifié (le troisième) et qu'aucune action n'a été entrepris pour deux document (visible avec le point d'interrogation présent). Les différentes actions possibles sont décrites cidessous.

svn add **svn add** permet d'ajouter un document local au dépôt. Le point d'intérrogation présent pour le document se transformera en **A** :

```
beytu@proteus:~/Documents/DEST/jukeboxe$
svn add documentation/src/annexes/compte_rendu_-_20060114.xml

   ? tmp/archive A documentation/src/annexes/compte_rendu_-_20060114.xml ?
   documentation/src/annexes/compte_rendu_-_20060107.xml M
   documentation/src/annexes/svn.xml

beytu@proteus:~/Documents/DEST/jukeboxe$
```

svn del Cette commande nous permet de supprimer un document présent dans le dépot :

```
beytu@proteus:~/Documents/DEST/jukeboxe$
svn del documentation/src/annexes/minidoc_docbook.xml

   ? tmp/archive A documentation/src/annexes/compte_rendu_-_20060114.xml ?
   documentation/src/annexes/compte_rendu_-_20060107.xml D
   documentation/src/annexes/minidoc_docbook.xml M documentation/src/annexes/svn.xml
```

```
beytu@proteus:~/Documents/DEST/jukeboxe$
```

svn move Commande permettant de déplacer un fichier d'un répertoire à un autre au sein de notre dépôt.

svn commit Cette commande permet d'appliquer tous les changements éffectués à l'aide des commandes décrites ci-dessous. Les fainéants auront le droit d'utilisé la commande **svn ci** . Nous pouvons voir ci-dessous un déroulement normal d'une synchronisation :

```
beytu@proteus:~/Documents/DEST/jukeboxe/documentation$
svn ci

   Log message unchanged or not specified a)bort, c)ontinue, e)dit c Sending
   documentation/src/annexes/svn.xml Sending documentation/src/uml.xmi Transmitting f
   data .. Committed revision 31.
```

C.11.4 Subversion et Eclipse

C.11.4.1 Introduction

Eclipse est un EDI (environement de dévelopement integré) qui ne cesse de gagner en popularité grâce a son architecture flexible. Ce logiciel est développé par IBM afin de faire de l'ombre -d'où son nom Eclipse- au très dominant SUN dans le monde JAVA. Son système d'extensions permet de se servire d'Eclipse comme un socle sur lequel on peut greffé des fonctionnalité. Aujourd'hui, nous trouvons des extensions (ou plugins en anglais) d'accès aux base de données, d'autres pour développer en C/C++, Python, PHP...

Ces extensions donnes accès à de nouvelles perspectives. Une perspective sous Eclipse permet de donner au developpeur une vision particulière et surtout différentes en fonction de la perspective sélectionné. Plus précisement, la perspective régis l'articulation réalisé entre les différentes couche d'Eclipse :

- les données gérées dans l'éspace de travail (workspace) qu'on peut nommer ressources
- et les interfaces graphiques qui nous permettent de visualiser ces ressources

Je vais tenter de vous indiquer comment utiliser Eclipse avec Subversion par l'intermèdiaire de la perspective **SVN** .

C.11.4.2 Installation

Je ne vais pas m'attarder sur l'installation d'Eclipse qui est plutôt simple en utilisant l'outil **APT** ou en utilisant l'archive directement du site d' Eclipse : <http://www.eclipse.org/downloads/> .

Eclipse ne supportant pas Subversion par defaut, il faut installer l'extenstion à partir du site http ://subclipse.tigris.org/update. Il faut donc lancer Eclipse et choisir un répertoire de travail (Workspace). Un nouveau répertoire vide est préferable pour un premier essai. J'éspere que vous allez pouvoir comprendre pourquoi Subversion est un outil très puissant à très ces quelques explicatifs. Par exemple, j'ai l'habitude de travailler sur differents post : école, travail, maison, ordinateur, portable...

Je n'ai plus à me trimbaler avec une clé USB à me demander si c'est bien la dernière version.

Une fois Eclipse lancé il faut :

- aller dans le menu Help>Software updates>Find and Install
- selection 'Search for new features' puis 'Next'
- selection 'New remote site' pour ajouter l'adresse http ://subclipse.tigris.org/update <http://subclipse.tigris.org/update>

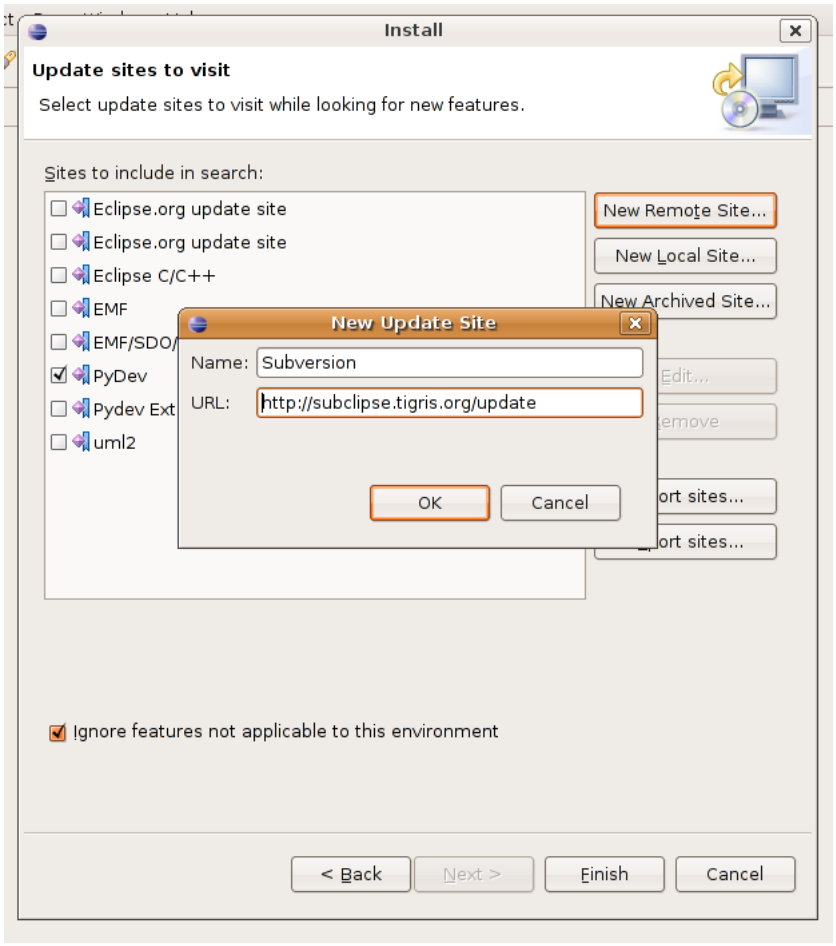


FIG. C.3: *Ajout de l'extension SVN Eclipse*. Ajout d'un serveur

– après avoir séléctionner l'extension 'Subversion', vous pouvez continuer ensuite le processus normal d'ajout de fonctionnalité.

C.11.4.3 Utilisation

Grâce à l'extension installer, vous avez droit à de nouvelles 'perspectives' sous Eclipse et notamment deux d'entre elles qui vous permet de visualiser le contenue du dépôt et de synchroniser votre depôt local avec le depôt distant.

Explorer le dépôt Il faut :

– commencer par aller dans 'Go to Window>open Perspective>new' et choisir 'SVN repository exploring'.
– cliquez avec le bouton droit dans 'SVN Repository' et cliquez sur 'New>Repository location'.
– Une fenêtre apparait vous invitant à ajouter l'adresse de votre serveur Subversion :

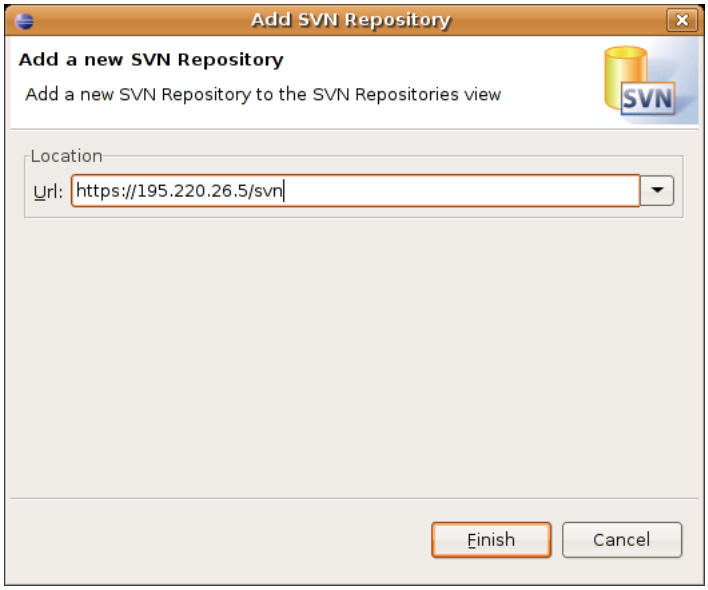


FIG. C.4: *Ajout d'un serveur Subversion*. Ajout d'un serveur SVN

– Eclipse vous invite à présent d'entré votre login et mot de passe.

Vous pouvez à présent visualiser le dépôt.

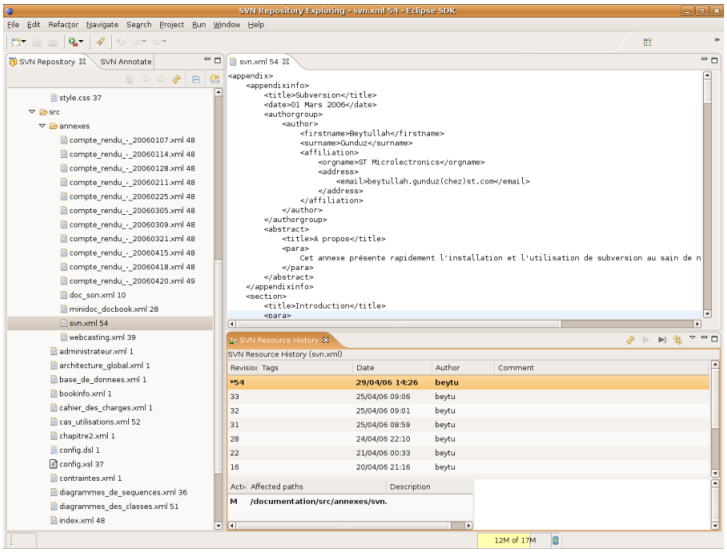


FIG. C.5: *Visualisation d'un dépot*. Exlploration d'un serveur SVN

Travailler en équipe

Introduction

L'utilité de Subversion est tout de même le travail en équipe. Je vais décrire ici comme y arriver avec Eclipse en s'appuyant sur Subversion.

Utilisation de SVN Eclipse

Si vous avez configuré votre explorateur de dépôt comme expliqué dans la partie précédente, la configuration pour le travail en équipe sera très simple.

Il faut commencer par choisir une perspective de travaille (la perspective 'Java' peut faire l'affaire). Ensuite, il faut utiliser la combinaison de touche **CTRL + N** pour la création d'un nouveau projet. Une fenêtre vous invite à selectionner un type de projet comme le montre l'image ci-dessous :

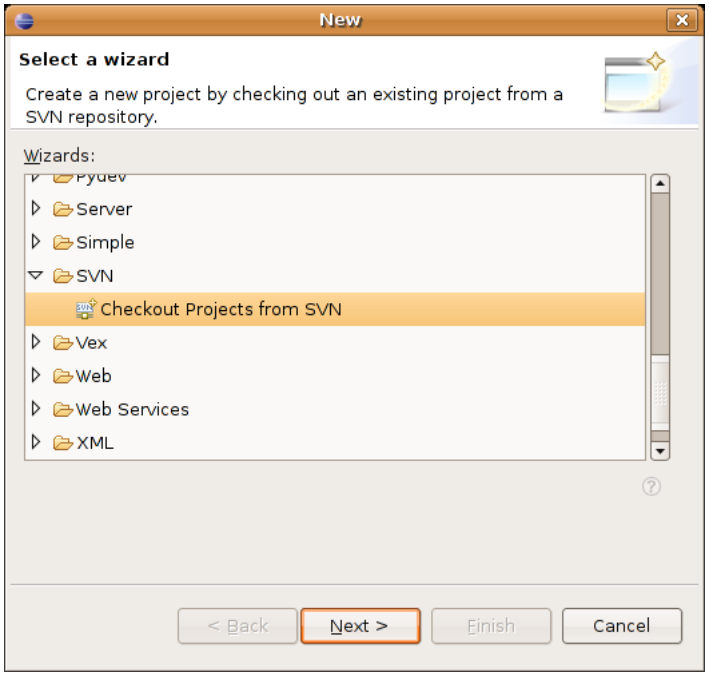


FIG. C.6: *Création d'un projet Subversion avec Eclipse*

Vous choisirez 'Checkout Projects From SVN' puis, une fenêtre vous invitera à sélectionner un dépôt distant. Selectionner celui que vous avez crée précedement avec l'explorateur de dépôt. Rien ne vous empêche bien évidemment d'en créer un nouveau ic. Enfin une dernière question vous est posée concernant les dossiers à utiliser : sélectionné les tous.

Il reste une dernière étape avant de pouvoir travailler facilement avec Subversion. Il faut sélectionner la perspective 'Team synchronisation' puis cliquer sur le petit bouton 'Synchronize mis en évidance sur l'image ci-dessous :

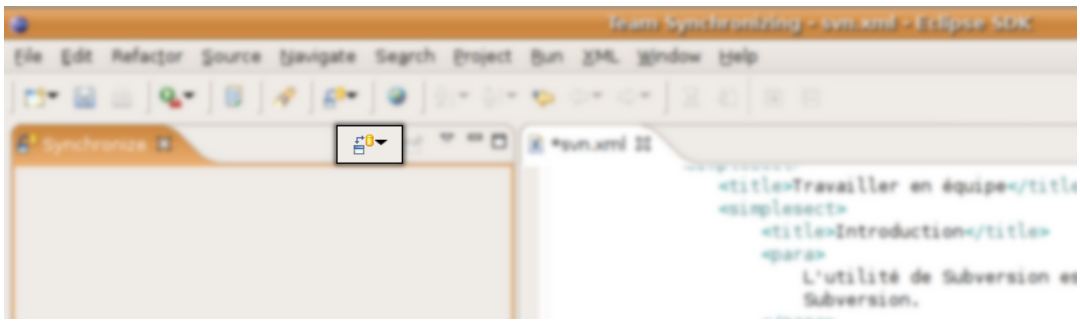


FIG. C.7: *Synchronisation d'un dépot distant avec Eclipse*

Vous êtes ensuite invité a séléctionner le type de synchronisation (CVS ou Subversion), il faut bien évidemment séléctionner Subversion (SVN) dans notre cas puis séléctionner les répertoires à synchroniser. Une fois ceux-ci sélectionner, vous pouvez cliquer sur 'Finish'.

A présent, toutes les modifications que vous apporterez à vos fichiers apparaitrons dans cette perspective. Un clique droit puis la sélection de l'action 'Commit' permettra de de mettre à jour le document concerné.

NOTE

Certains fichiers propres à votre éspace de travail ne devraient pas être mis à disposition de vos collaborateurs. L'extension de collaboration d'Eclipse inclut dans ses options la possibilité de configurer une liste de type de fichier à ignorer. Cette fonctionnalité est accessible via le menu **Window**>**Preferences...** puis sur la droite sélectionner **Team** et enfin **Ignored ressources** . L'image ci-dessous pourra vous aider :

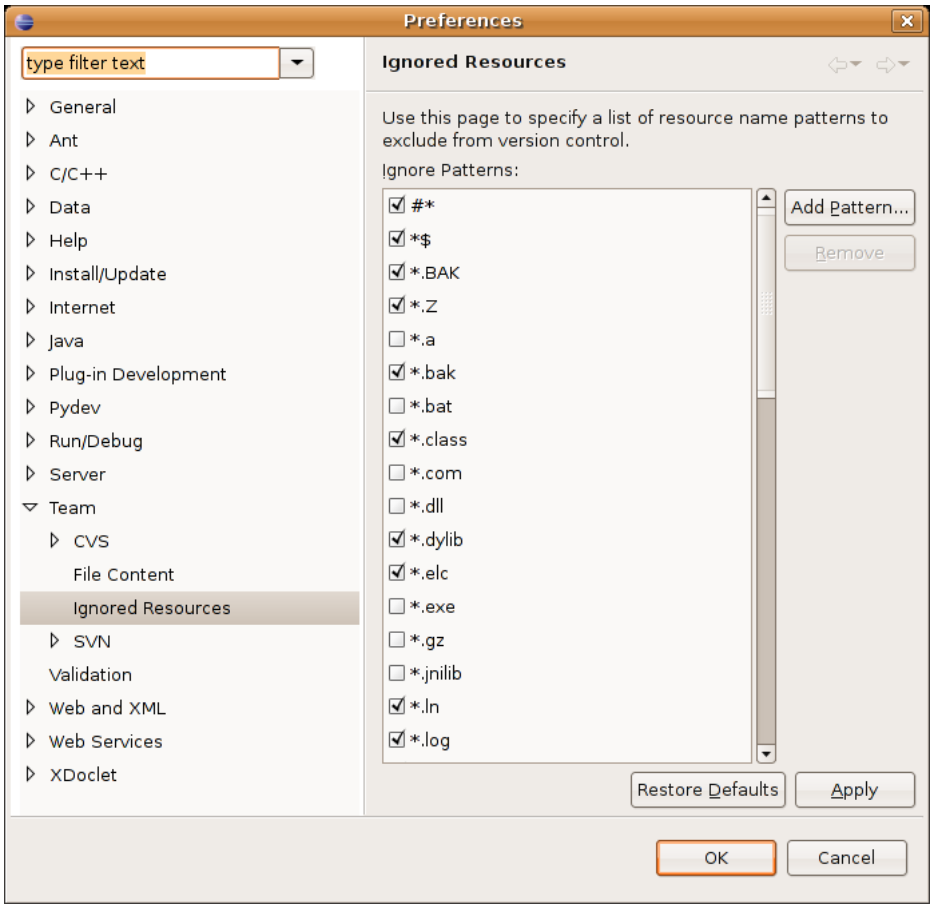


FIG. C.8: *Configuration des ressources à ignorer par Eclipse*

NOTE

Typiquement vous pourrez ajouter les fichiers nommés **.project** qui sont les fichiers employés par Eclipse pour sauvegarder des informations relatifs au dossier considéré.

Résolution de conflits

Un des aspects pratique de cet outil est sans contest une gestion des conflits avancé. Si un conflit apparait après la synchronisation, ce fichier aura un emblème rouge indiquant une flèche bidirectionnel. Il suffit de faire un clique droit sur ce fichier, puis sélectionner `Open in compare editor`.

Un editor avec double panneau apparaître. Sur la droite, on aura le fichier distant et sur la gauche le fichier local. L'éditeur nous indique toutes les différences trouvé t nous invite à résoudre le problème.

Il suffit alors de constituer à gauche le fichier final en récuperant du fichier de droite tous le nécessaire. Une fois l'operation éffectué et le travail sauvegardé, on peut fermer l'éditeur.

Il faut à présent cliquer de nouveau sur le fichier posant problème et séléctionner "Marked as merged". Enfin, il faut faire un commit du fichier pour que les modifications soient éffectivent.

C.12 Le webcasting et la législation française

C.12.1 Le Webcasting

C.12.1.1 De façon générale

Diffusion personnalisée et rafraîchie d'informations sur Internet, poussées par l'entreprise en direction de l'internaute suite à la mention par ses soins de ses centres d'intérêt (Téléobs ; compagnies aérienne qui envoie par mail la liste des places libre sur ses vols du lendemain). Synonyme : Singlecasting.

C.12.1.2 De façon plus centrée sur notre sujet

Programmation musicale semblable à la radiodiffusion mais réalisée sur l'Internet. Plusieurs sociétés offrent des sites de radiodiffusion sur le Web permettant d'écouter un vaste éventail de musique. Des concerts en direct peuvent également être diffusés, en audio et en vidéo, sur l'Internet.

C.12.2 Loi française autour du Webcasting, La SACEM

Extrait du site internet de la sacem (http ://www.sacem.fr) :

Si vous créez une WebRadio et/ou une WebTV, vous utiliserez des oeuvres musicales, littéraires, dramatiques, des sketches, de l'habillage graphique... Pour diffuser ces oeuvres protégées par le droit d'auteur, vous devez obtenir une autorisation des sociétés d'auteurs assurant la gestion de ces droits. SESAM est chargée de gérer le contrat qui permet la diffusion et la reproduction des oeuvres des sociétés de gestion de droits d'auteurs (Sacem, SDRM, SACD, SCAM, ADAGP).

Le Contrat d'autorisation flux continu (programmes destinés exclusivement à l'écoute en streaming à l'exclusion de toute faculté d'écoute oeuvre par oeuvre à la demande et de téléchargement) a vocation à autoriser :

- l'écoute d'oeuvres : WebRadio, Mix de DJ,
- la visualisation d'oeuvres : WebTV, etc.,
- la pré-écoute d'oeuvres : extraits, la pré-visualisation d'oeuvres : Bandes Annonces, etc.

Pour la diffusion d'un programme en flux continu exploitée par un Particulier ou un organisme à but non lucratif (Association, fondations, collectivités locales, administration, etc.) ne générant pas de recette via son site Internet, les conditions financières prévoient l'application d'un forfait de 72,5 euros HT par mois avec une majoration égale à 2,7 euros HT par tranche de 100 000 pages vues par mois (PAVM).

Pour la diffusion d'un programme en flux continu exploitée par une Société, ou un Particulier et un organisme à but non lucratif générant des recettes via son site Internet, les conditions financières prévoient une option laissée au choix du diffuseur :

a l'option 1 (conseillée si l'activité principale de votre site Internet consiste à la mise à disposition d'oeuvres) prévoit de percevoir une rémunération égale à :
   - 6 % sur l'ensemble des Recettes publicitaires et assimilées,
   - assortie d'un minimum garanti de 145 euros HT par mois avec une majoration égale à 5,35 euros HT par tranche de 100 000 pages vues par mois (PAVM).

b l'option 2 (conseillée si la mise à disposition d'oeuvres n'est pas l'activité principale du site) consiste à percevoir une rémunération égale à :
   - 12 % x (Recettes Publicitaires et assimilées x PAVMO/PAVM)
   - assortie d'un minimum garanti égal à 200 euros HT par mois par tranche de 500.000 PAVMO et une majoration égale à 20 euros HT par tranche de 250 000 PAVMO.

La notion de pages vues avec oeuvres (PAVMO) inclut : non seulement les pages où se trouvent les players permettant l'accès aux flux et les pages générées sur ces players, mais surtout l'ensemble des pages où sont mises à disposition des oeuvres et notamment celles qui permettent d'accéder aux pages où se trouvent lesdits players. Cet ensemble de pages est généralement regroupé sous forme de

Rubrique (ex. : " Cinéma "," Musique ", etc.) qui ont pour objet principal de mettre à disposition des contenus protégés, par opposition à d'autres rubriques (ex. : " Forum ", " Horoscope ", "Bourse ", " votre compte ", " vos e-mail ", etc.) qui n'ont pas pour objet principal de mettre à disposition des Internautes des contenus protégés.

La notion de PAVMO permet ainsi de déterminer la part des contenus protégés par rapport à l'ensemble des contenus mis à disposition. Cette seconde option peut convenir aux exploitants de site Internet dont la mise à disposition d'oeuvres des répertoires des Sociétés d'Auteurs ne constitue qu'une activité accessoire.

Les recettes entrant dans l'assiette de la rémunération sont les recettes publicitaires de sponsoring, d'échange, d'affiliation, de partenariat, etc. du site web, du player et du programme audio.

Lorsque le programme en flux Continu est exploité par une Société qui ne génère aucune recette via son site Internet (WebRadio/WebTV institutionnelles, Intranet d'entreprise), c'est l'un de ces deux minima garantis qui a vocation à s'appliquer, en fonction de l'option choisie par l'exploitant.

Fin de l'extrait.

C.12.3 Notre Choix

D'après le cahier des charges, ce site internet a un but non lucratif, dont l'activité principale est la diffusion de musique sur internet. Des recettes seront générés par la publicité. L'intégralité des abonnements à la radio seront versés aux artistes. Les recettes de la publicité pourront servir à payer un peu plus les artistes ou à payer la sacem.

Nous entrons donc dans le cas 1 : but non lucratif, activité principale et recettes générées par la publicité.

Nous rémunérerons la Sacem à hauteur de :

- 6 % des recettes publicitaires percues
- 145 euros HT par mois + 5,35 euros HT par tranche de 100000 pages vu par mois (PAVM)

Donc au minimum, ce site internet doit rapporter pour couvrir le coût de la sacem :

- 145 * 1.196 = 173,42 euros TTC

on devra ajouter :

- 5,35 * 1,196 = 6,3986 euros TTC par tranche de 100000 PAVM