TBXCAST

Un protocole de routage multicast explicite

Rapport de Planification

Cyril BOULEAU
Hamze FARROUKH
Loïc LE HENAFF
Mickaël LECUYER
Josef LEGENY
Benoît LUCET
Emmanuel THIERRY

Encadrants : Miklós MOLNÁR, Bernard COUSIN

Sommaire

1 Introduction

La planification de notre projet nous a permis de réfléchir sur les différentes étapes que nous allons traverser. Nous avons ainsi déterminé les objectifs que nous devrons atteindre d'ici la fin de l'année. Nous avons également évalué les ressources à notre disposition afin de connaître les capacités de l'équipe et d'estimer le temps que nous devrons donner à chaque phase.

Dans ce rapport de planification, nous vous proposons tout d'abord un bref rappel de notre projet, TBXcast, qui a pour objectif de développer un protocole de communication. Nous expliquerons ensuite notre modèle de développement et exposerons les différentes versions de TBXcast. Nous poursuivrons ensuite sur l'évolution de la salle d'expérimentation. Enfin, nous présenterons Microsoft Project que nous avons utilisé pour la planification et nous joindrons les résultats de notre travail : le diagramme de Gantt et la liste des tâches.

2 Présentation de TBXcast

Notre projet a pour objectif de créer un protocole de communication entre les machines. Ce dernier devra gérer la transmission de données depuis une source vers un ou plusieurs destinataires. En effet, lorsque plusieurs destinataires désirent recevoir les mêmes données, ils s'abonnent à un groupe, dit multicast. Si le nombre de groupes est trop important, la taille des tables de routage augmente et le temps d'acheminement devient alors plus long. Une solution à ce problème est l'utilisation du routage multicast explicite qui tire parti des avantages de l'unicast et du multicast afin d'éviter la redondance des données envoyées.

TBXcast est quant à lui un protocole multicast explicite arborescent. C'est à dire que l'entête d'un paquet contient à la fois les destinataires et l'arbre de routage. Pour l'implémenter, nous nous appuierons sur un protocole multicast explicite déjà existant: Xcast. Ce dernier est fonctionnel et libre. Nous nous servirons ainsi du code à notre disposition et le modifierons pour intégrer TBXcast.

Le projet TBXcast a débuté l'année dernière. Nous avons ainsi récupéré le travail déjà accompli et étudié les difficultés rencontrées. Nous avons ainsi pu contourner les problèmes de l'année dernière, notamment en étudiant Xcast dès la pré-étude.

Notre projet étant à la fois logiciel et matériel, nous disposons d'une salle d'expérimentation dans laquelle nous allons pouvoir tester notre protocole de routage sur différentes topologies de réseau. L'organisation de la salle a nécessité un gros investissement que nous détaillerons un peu plus loin dans ce rapport.

3 Les phases de développement

Notre projet étant conséquent, plusieurs phases ont été définies et découpées en plusieurs sous-tâches. La fin de chaque phase est caractérisée par la rédaction d'un rapport présentant en détail le travail accompli.

La première phase a concerné l'étude de domaine de notre projet. Nous avons dû nous former au fonctionnement des réseaux et comprendre les rouages de notre projet. Ce fut également l'occasion pour nous de découvrir quelles ont été les erreurs commises par le groupe de l'année dernière et quelles ont été les solutions envisagées. Ainsi, nous avons pu dès le départ commencer à étudier le fonctionnement de Xcast. Cette première phase fut l'occasion de nous organiser entre nous et de se répartir le travail. Elle a duré de septembre à fin octobre.

La deuxième phase concerna les spécifications fonctionnelles. Dans cette phase, nous avons terminé d'étudier les différents modules de Xcast dans les grandes lignes. Nous avions ainsi toutes les cartes en main pour spécifier TBXcast. Nous connaissions les fonctionnalités de Xcast, nous savions ce que devait apporter TBXcast, et surtout, nous avons identifié les parties du code de Xcast à modifier pour implémenter notre projet. Nous avons également pu mettre en place la salle d'expérimentation. Cette phase a duré de fin octobre à début décembre.

Nous nous trouvons actuellement dans la troisième phase de notre projet : la planification. C'est l'occasion pour nous de faire le bilan à propos de notre répartition des tâches, et d'estimer un calendrier pour les phases suivantes. Nous avons réfléchi sur la répartition du travail pour la suite par rapport au temps nécessaire pour chaque tâche. Cette phase se terminera fin décembre.

Début janvier, nous rentrerons dans la phase de conception logicielle. Ce sera pour nous l'occasion de définir précisément la modélisation de notre projet. Nous pourrons alors nous focaliser sur des questions internes au développement de TBXcast comme les choix d'algorithme et de codage. Nous devrons également travailler sur la modélisation de ping6tbx (l'application test pour TBXcast) et sur LibTBXcast (l'API pour manipuler des paquets TBXcast). Nous y réfléchirons jusqu'à mi-février environ.

Suite à la conception logicielle, nous commencerons à modifier le code de Xcast pour implémenter TBXcast. Nous avons décidé de travailler en plusieurs cycles développement-test-validation. Nous proposons un rappel des versions un peu plus loin dans le rapport. L'objectif est ici d'obtenir plusieurs versions de notre protocole, chacune étant entièrement fonctionnelle et plus complète que la précédente. Pour la phase de construction, notre organisation repose donc sur un développement en spirale, ou plusieurs petits cycles s'enchaînent. Nous y travaillerons jusqu'à fin mai.

La fin du mois de mai sera consacrée à la dernière phase, la phase de livraison. TBXcast devra alors être fonctionnel et documenté. Nous présenterons l'ensemble de notre travail durant la soutenance finale.

Notre choix de développement s'inscrit ainsi à mi-chemin entre un cycle en V et un cycle en spirale. En effet, les premières phases de notre projet s'apparentent d'avantage à un cycle en V. Quant à la phase de codage, elle se rapproche d'un développement en spirale, sous forme de version incrémentale.

4 Les différentes versions de TBXcast

Nous avons établi huit versions de TBXcast. Chacune amène son lot de nouveautés par rapport à la précédente et chacune doit être fonctionnelle. Toutefois, notre objectif pour cette année est d'arriver à la version 3 dans le meilleur des cas. Les fonctionnalités de TBXcast y seront intégrées dans une grande majorité. Nous proposons un descriptif de chacune de ces versions ci-dessous.

Version 0
Renommer toutes les fonctions, constantes et macros Xcast.
Effectuer également les changements pertinents dans l'en-tête du paquet Xcast (route type, version etc.).

On pourra également ajouter des fonctions d'affichage pour suivre le comportement du programme avec plus de précision.

Version 1
Réaliser l'envoi de paquets TBXcast simplifiés à travers les routeurs : on donne un **arbre simplifié** formé d'une racine et de N nœuds correspondant aux N destinataires.

C'est fondamentalement la technologie Xcast qui agit, mais cette version permet d'introduire les « vrais » paquets TBXcast (envoi et réception de N paquets unicast).

Version 2
Les **routeurs intermédiaires** sont **capables de traiter les paquets TBXcast** et les destinataires sont capables de les recevoir.

L'arbre est donné à la source.

Dans cette version, les routeurs intermédiaires (de branchement) savent réellement traiter l'arbre non-simplifié donné en entrée.

Version3
La source sait construire l'arbre à partir d'une topologie fournie en entrée.

Version 4
Gestion de **la segmentation de l'arbre** et fragmentation des paquets de manière cohérente.

Version 5
Gestion des groupes avec xcgroup.

Version 6
Récupération de la topologie grâce au protocole OSPF (entre autres).

On a ici un protocole autonome qui sait récupérer la topologie, construire et segmenter l'arbre, et transmettre les informations aux routeurs afin que les paquets soient traités et reçus correctement.

Version 7
Gestion de la QoS. Prise en compte simultanée de différents indicateurs : délai, gigue, perte d'information etc.

Extension de l'algorithme de création de l'arbre pour s'adapter suivant ces différents paramètres et ainsi fournir un arbre adapté à des contraintes données.

5 La salle d’expérimentation

Sur notre projet, l'intérêt d'une salle d'expérimentation est stratégique. Nous travaillons sur un sujet délicat, de développement système et réseau, avec des technologies que nous ne maîtrisons pas. Il est probable que nous aurons beaucoup plus de bugs et d'erreurs que d'autres projets et cela nous demandera des tests conséquents et fréquents. Par ailleurs, des tests unitaires, ou bien des tests préliminaires sur nos propres machines ne sont pas envisageables : l'implémentation d'un protocole se fait sur un système précis, et quasiment aucun de ses composants n'a de sens sans un réseau significatif.

Nous avons hérité de la salle dans l'état dans lequel l'équipe précédente l'avait laissée. Nous avions plusieurs postes assez anciens installés sous NetBSD 3.1 et Xcast. La première chose à faire a été de réinstaller le système et Xcast pour prendre en main la plateforme. Nous avons ensuite migré vers des machines homogènes et plus récentes. Puis, après quelques études, nous avons décidé de tester Xcast sous NetBSD 4.0 pour éventuellement changer de base de travail. Les tests terminés, nous avons testé à grande échelle, en utilisant le switch qui nous a été fourni.

Cette première étape a posé les bases de la plateforme, c'est-à-dire que tout le travail suivant vise essentiellement à faciliter et accélérer les tests. On peut considérer son développement comme un projet préliminaire au codage de TBXcast. Son cahier des charges est simple, automatiser la compilation, la configuration, et les tests. La méthode de développement est tout aussi basique, une méthode agile. L'évolution de la plateforme laisse constamment entrevoir de nouvelles possibilités et ferme d'autres portes, on ne peut s'enfermer dans une solution définitive.

La suite du développement a été de mettre en place un netboot pour faciliter l'installation du système et s'assurer que la plateforme soit cohérente. Ainsi, cela prépare le travail pour un script d'installation de NetBSD.

La suite logique des choses demandera l’implémentation d’une suite de scripts pour contrôler les postes (redémarrage, arrêt, informations du système), pour installer le système, pour configurer le réseau (configuration du switch, des IPs, et des routes), et pour tester (lancer ping6tbx et tester la réception avec tcpdump).

Le reste du travail est une prise en main de nouveaux outils (whireshark, débuggeur noyau, …) ou l'installation de services divers (serveur de stockage, serveur ssh, …).

Pour l'évaluation du temps nécessaire, il est très approximatif, l'objectif n'est pas fixe, il est d'aller le plus loin possible et, comme dit précédemment, il peut être amené à changer.

6 Planification initiale

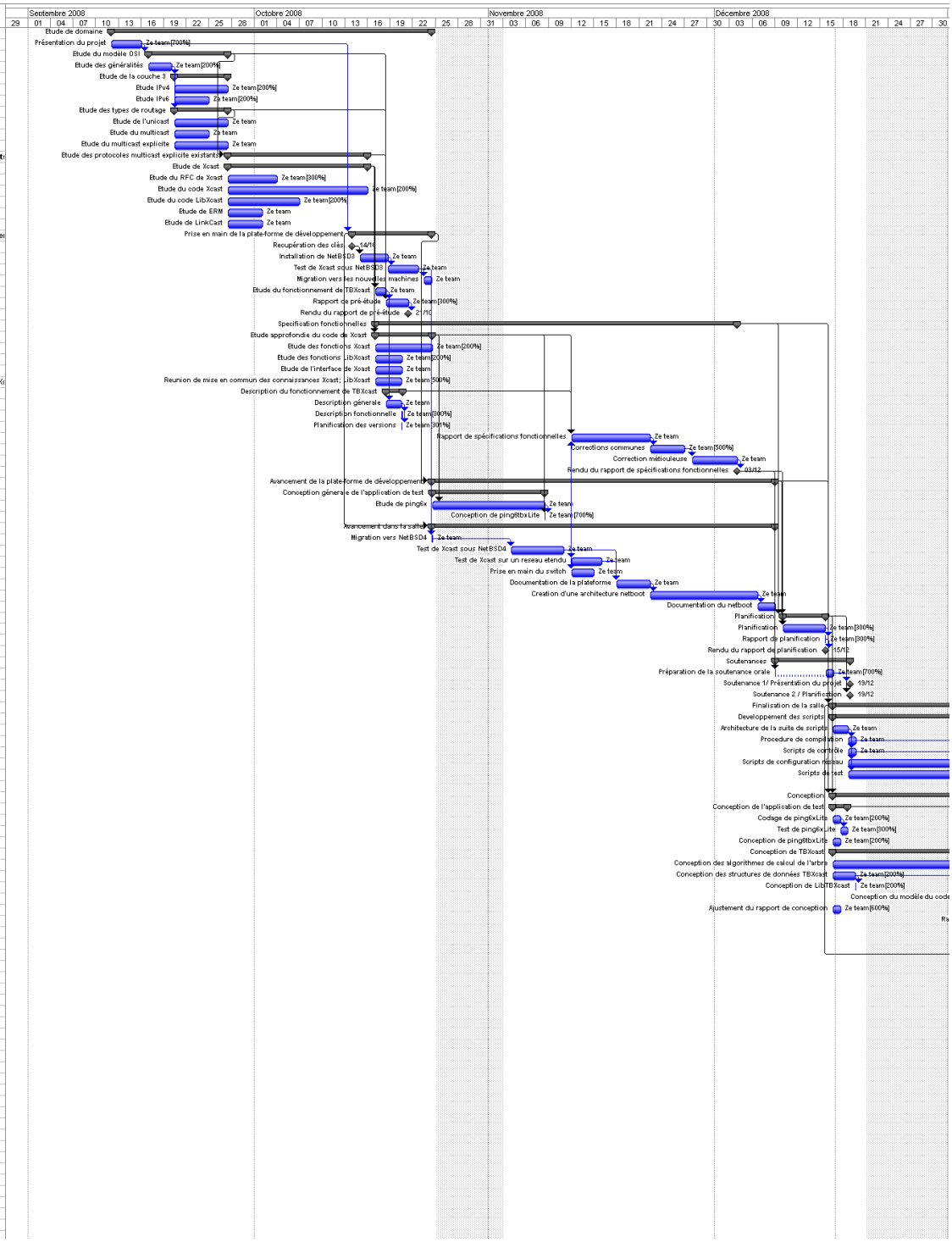
Afin de planifier le développement de notre application, nous allons utiliser un outil de gestion de projet par la méthode de Gantt : Microsoft Project. Cet outil très répandu permet de calculer la répartition de travail à partir des informations fournies. Ces informations peuvent être de sortes très variées. Au vu du contexte de notre projet, nous allons nous baser sur un nombre limité de paramètres :

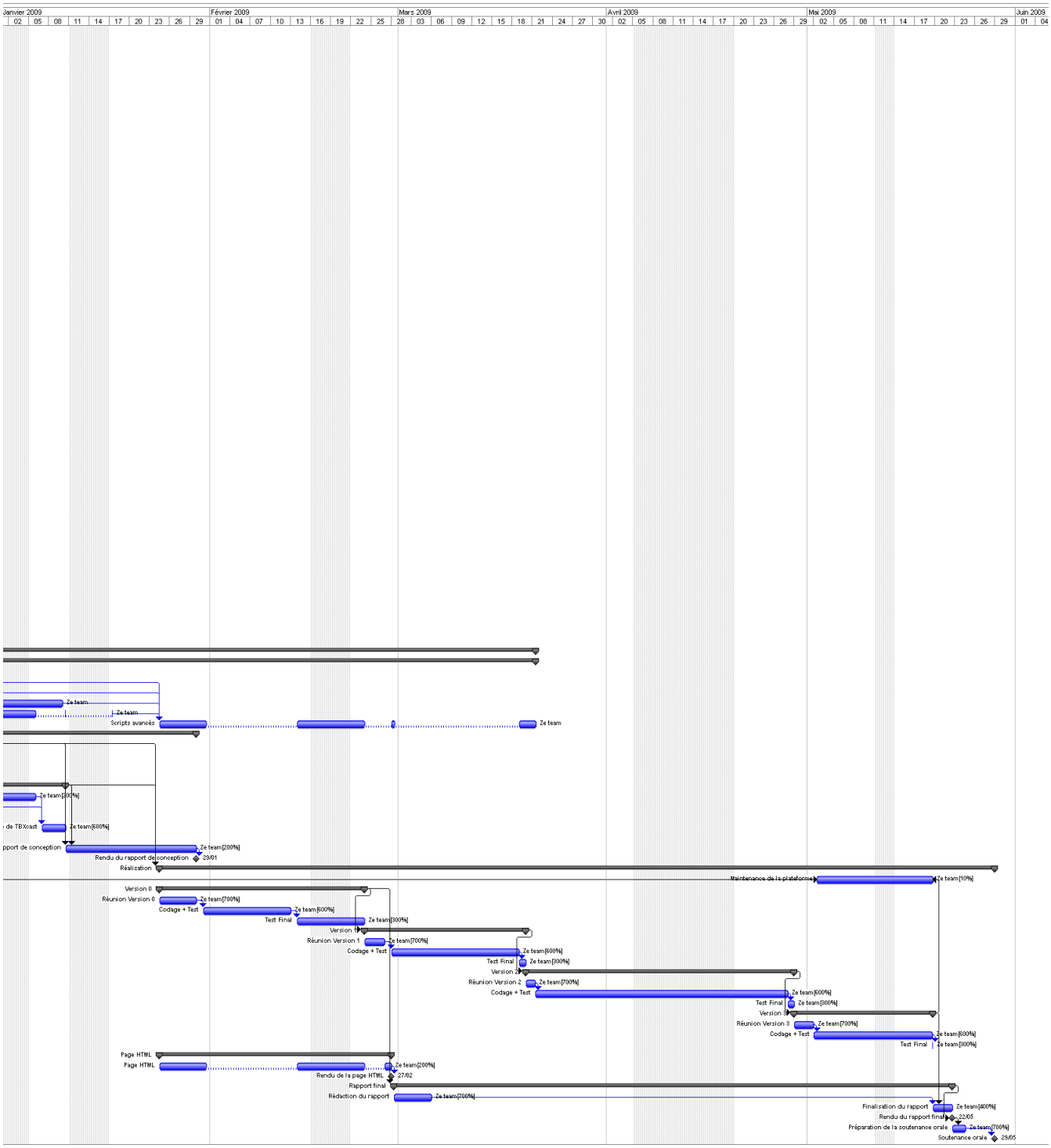
- le travail nécessaire pour une tâche
- le nombre de personnes assignées pour une tâche
- l'interdépendance des tâches
- le temps de travail fourni dans une semaine

A partir de ces informations, on saura dans un premier temps si notre projet est effectivement réalisable dans le temps imparti. S'il s'avère impossible à achever, il faudra modifier un ou plusieurs de ces quatre paramètres.

Une fois la planification achevée, l'outil nous permet de faire le suivi du projet. On pourra ensuite remettre en cause la planification si l'avancement prévu ne correspond pas à la réalité. Dans la partie « suivi », on va aussi assigner les personnes particulières aux taches, contrairement à la ressource universelle que l'on a utilisé lors de la planification.

N°	Nom de la tâche
1	**Etude de domaine**
2	Présentation du projet
3	**Etude du modèle OSI**
4	Etude des généralités
5	**Etude de la couche 3**
6	Etude IPv4
7	Etude IPv6
8	**Etude des types de routage**
9	Etude de l'unicast
10	Etude du multicast
11	Etude du multicast explicite
12	**Etude des protocoles multicast explicite existant**
13	**Etude de Xcast**
14	Etude du RFC de Xcast
15	Etude du code Xcast
16	Etude du code LibXcast
17	Etude de ERM
18	Etude de LinkCast
19	**Prise en main de la plate-forme de développeme**
20	Recupération des clés
21	Installation de NetBSD3
22	Test de Xcast sous NetBSD3
23	Migration vers les nouvelles machines
24	Etude du fonctionnement de TBXcast
25	Rapport de pré-étude
26	Rendu du rapport de pré-étude
27	**Specification fonctionnelles**
28	**Etude approfondie du code de Xcast**
29	Etude des fonctions Xcast
30	Etude des fonctions LibXcast
31	Etude de l'interface de Xcast
32	Reunion de mise en commun des connaissances X
33	**Description du fonctionnement de TBXcast**
34	Description générale
35	Description fonctionnelle
36	Planification des versions
37	Rapport de spécifications fonctionnelles
38	Corrections communes
39	Correction méticuleuse
40	Rendu du rapport de spécifications fonctionnelles
41	**Avancement de la plate-forme de développement**
42	**Conception générale de l'application de test**
43	Etude de ping6x
44	Conception de ping6tbxLite
45	**Avancement dans la salle**
46	Migration vers NetBSD4
47	Test de Xcast sous NetBSD4
48	Test de Xcast sur un reseau etendu
49	Prise en main du switch
50	Documentation de la plateforme
51	Creation d'une architecture netboot
52	Documentation du netboot
53	**Planification**
54	Planification
55	Rapport de planification
56	Rendu du rapport de planification
57	**Soutenances**
58	Préparation de la soutenance orale
59	Soutenance 1/ Présentation du projet
60	Soutenance 2 / Planification
61	**Finalisation de la salle**
62	**Developpement des scripts**
63	Architecture de la suite de scripts
64	Procedure de compilation
65	Scripts de contrôle
66	Scripts de configuration réseau
67	Scripts de test
68	Scripts avancés
69	**Conception**
70	**Conception de l'application de test**
71	Codage de ping6xLite
72	Test de ping6xLite
73	Conception de ping6tbxLite
74	**Conception de TBXcast**
75	Conception des algorithmes de calcul de l'arbre
76	Conception des structures de données TBXcast
77	Conception de LibTBXcast
78	Conception du modèle du code de TBXcast
79	Ajustement du rapport de conception
80	Rapport de conception
81	Rendu du rapport de conception
82	**Réalisation**
83	Maintenance de la plateforme
84	**Version 0**
85	Réunion Version 0
86	Codage + Test
87	Test Final
88	**Version 1**
89	Réunion Version 1
90	Codage + Test
91	Test Final
92	**Version 2**
93	Réunion Version 2
94	Codage + Test
95	Test Final
96	**Version 3**
97	Réunion Version 3
98	Codage + Test
99	Test Final
100	**Page HTML**
101	Page HTML
102	Rendu de la page HTML
103	**Rapport final**
104	Rédaction du rapport
105	Finalisation du rapport
106	Rendu du rapport final
107	Préparation de la soutenance orale
108	Soutenance orale




Janvier 2009
Février 2009
Mars 2009
Avril 2009
Mai 2009
Juin 2009
Ze team
Scripts avancés
Ze team
Ze team[200%]
Ze team[600%]
Rendu du rapport de conception
29/01
Réalisation
Maintenance de la plateforme
Ze team[10%]
Version 0
Réunion Version 0
Ze team[700%]
Codage + Test
Ze team[600%]
Test Final
Ze team[300%]
Version 1
Réunion Version 1
Ze team[700%]
Codage + Test
Ze team[600%]
Test Final
Ze team[300%]
Version 2
Réunion Version 2
Ze team[700%]
Codage + Test
Ze team[600%]
Test Final
Ze team[300%]
Version 3
Réunion Version 3
Ze team[700%]
Codage + Test
Ze team[600%]
Test Final
Ze team[300%]
Page HTML
Page HTML
Ze team[200%]
Rendu de la page HTML
27/02
Rapport final
Rédaction du rapport
Ze team[700%]
Finalisation du rapport
Ze team[400%]
Rendu du rapport final
22/05
Préparation de la soutenance orale
Ze team[700%]
Soutenance orale
29/05