INSA de Rennes
département informatique

TBXcast

Soutenance finale – Juin 2009

Cyril Bouleau, Hamze Farroukh, Loïc Le Henaff, Mickaël Lecuyer, Jozef Legény, Benoît Lucet, Emmanuel Thierry
Encadreurs : Bernard Cousin, Miklós Molnár

Introduction

- Le projet TBXcast a pour objectif la création d'un nouveau protocole de routage
  - Projet bas niveau
    - Modification du noyau du système NetBSD
  - Projet réseau
    - Nécessité d'une plateforme de test, sous IPv6
  - Projet basé sur l'existant
    - Etude approfondie du protocole Xcast
  - Projet en sa 2ème année de vie

Plan

- Cadre du projet
- Fonctionnalités du protocole
- Architecture de TBXcast
- Implémentation de TBXcast
- Présentation de la plateforme
- Bilan du projet
- Compléments de planification

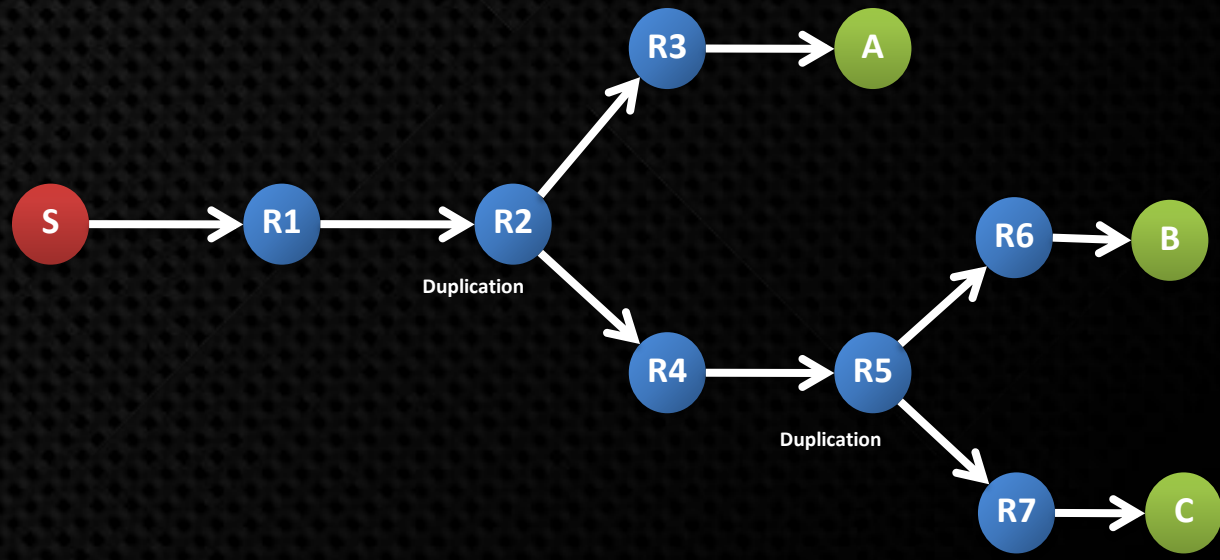
Cadre du projet

Contexte

- Les applications de communication en groupe se multiplient
  - Vidéoconférences
  - Jeux vidéo
- Les trafic généré est important

Routage multicast

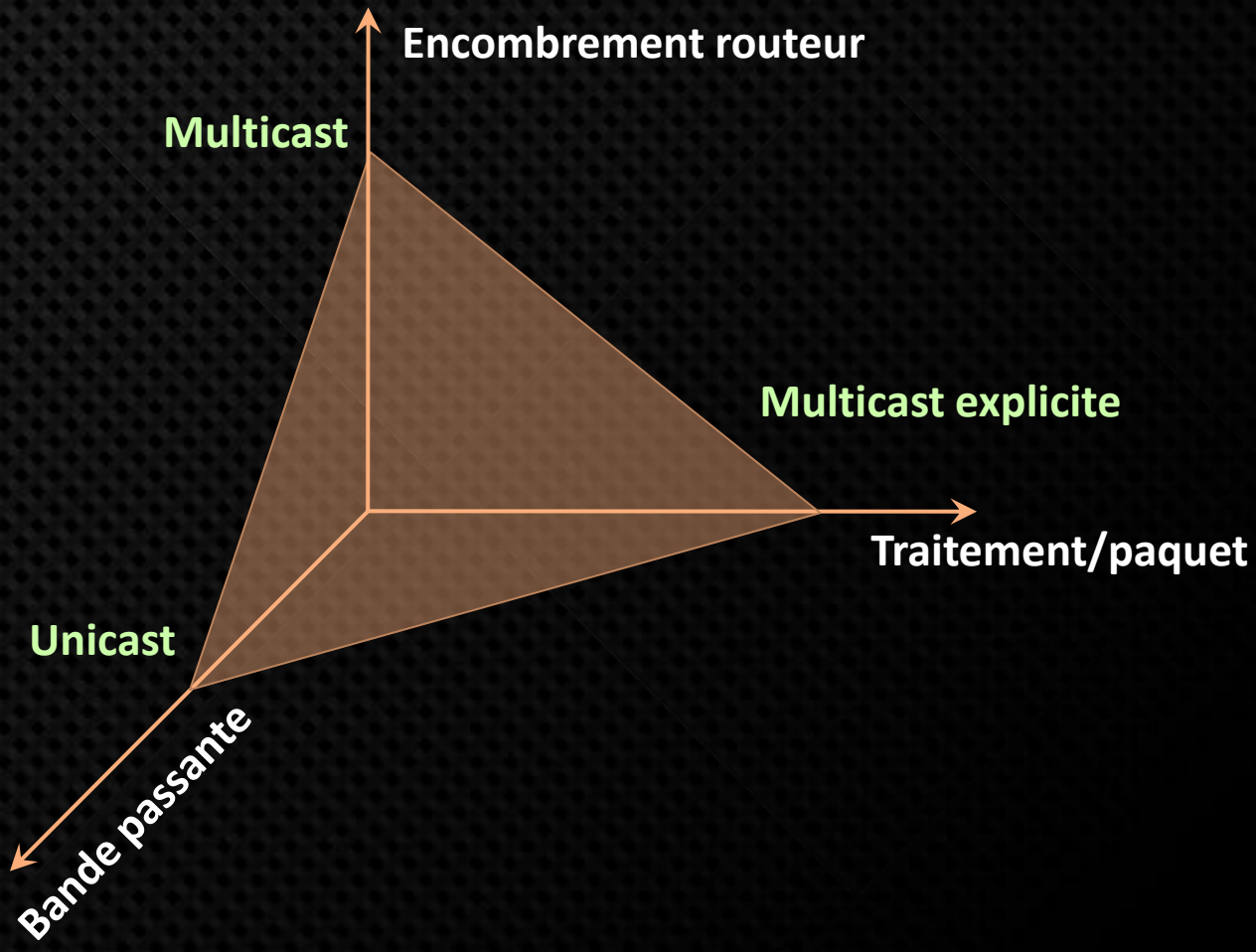
- Envoi d'un paquet à un groupe de destinataires
  - Economie de bande passante
  - Routeurs rapidement surchargés : une ligne présente par groupe dans la table de routage
    - Inenvisageable pour un très grand nombre de groupes



Routage multicast explicite

- Le paquet est lui-même porteur de la liste des destinataires
  - Les tables de routage sont soulagées
  - Mais cela nécessite davantage de traitement par paquet
- Multicast explicite plat : Xcast
  - Les destinataires sont représentés par une liste à plat
  - On ne connaît pas les chemins entre la source et les destinataires

Bilan des modes de routage



Routage multicast explicite arborescent

- L'ensemble des chemins depuis la source vers les destinataires est représenté par un arbre
  - Facilite et accélère le traitement dans les routeurs
- C'est la forme que nous avons choisi pour TBXcast, qui signifie « Tree Based eXplicit multicast »

TBXcast concrètement

- Extension de routage IPv6
  - Un entête situé entre l'entête IP et l'entête UDP/TCP du paquet

- Le code de TBXcast est déployé dans les routeurs du réseau

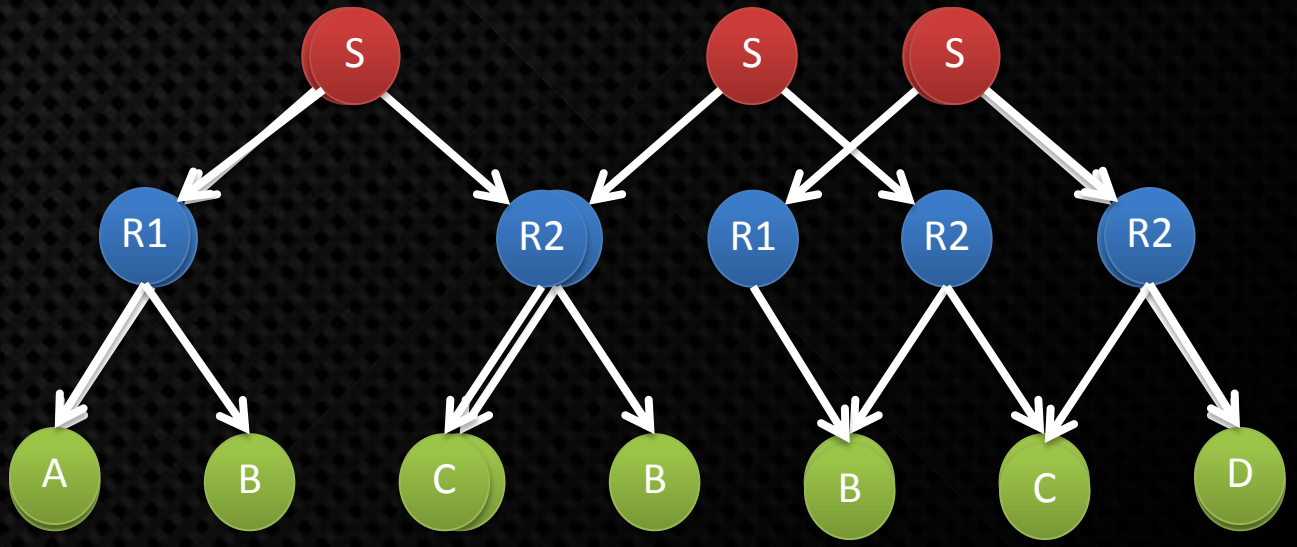
Fonctionnalités du protocole

De quoi doit être capable TBXcast ?

- Récupération de la topologie
- Gestion des groupes
  - Ajout, retrait de membres dans des groupes
- Création de l'arbre en fonction de la topologie et du groupe multicast concerné

Segmentation

- On a recours à la segmentation lorsque l'arbre est trop volumineux
- La difficulté est de bien diviser le paquet pour équilibrer les nouveaux entêtes



Déploiement sur le réseau

- Routage des paquets : mise à jour de l'arbre et renvoi au routeur suivant
- Fonctionnement sur un réseau hétérogène
  - Certains routeurs ne « comprennent » pas TBXcast

Qualité de Service

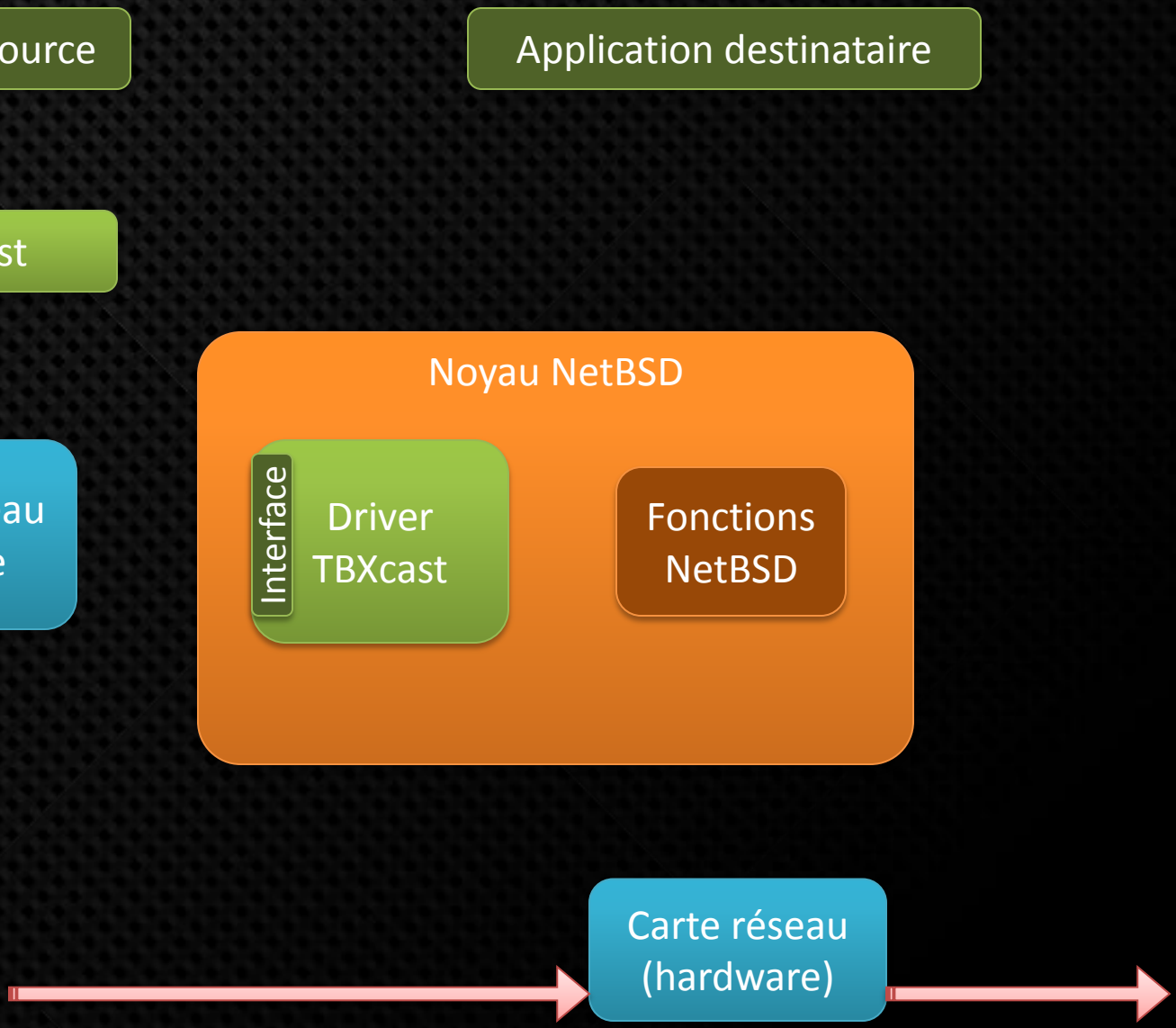
- La «QoS » permet un ajustement de la communication en fonction de paramètres
  - Délai
  - Variation de délai
  - Perte d'information
- Concrètement, cela est réalisé par un arbre enrichi dont la création intègre des contraintes

Architecture de TBXcast

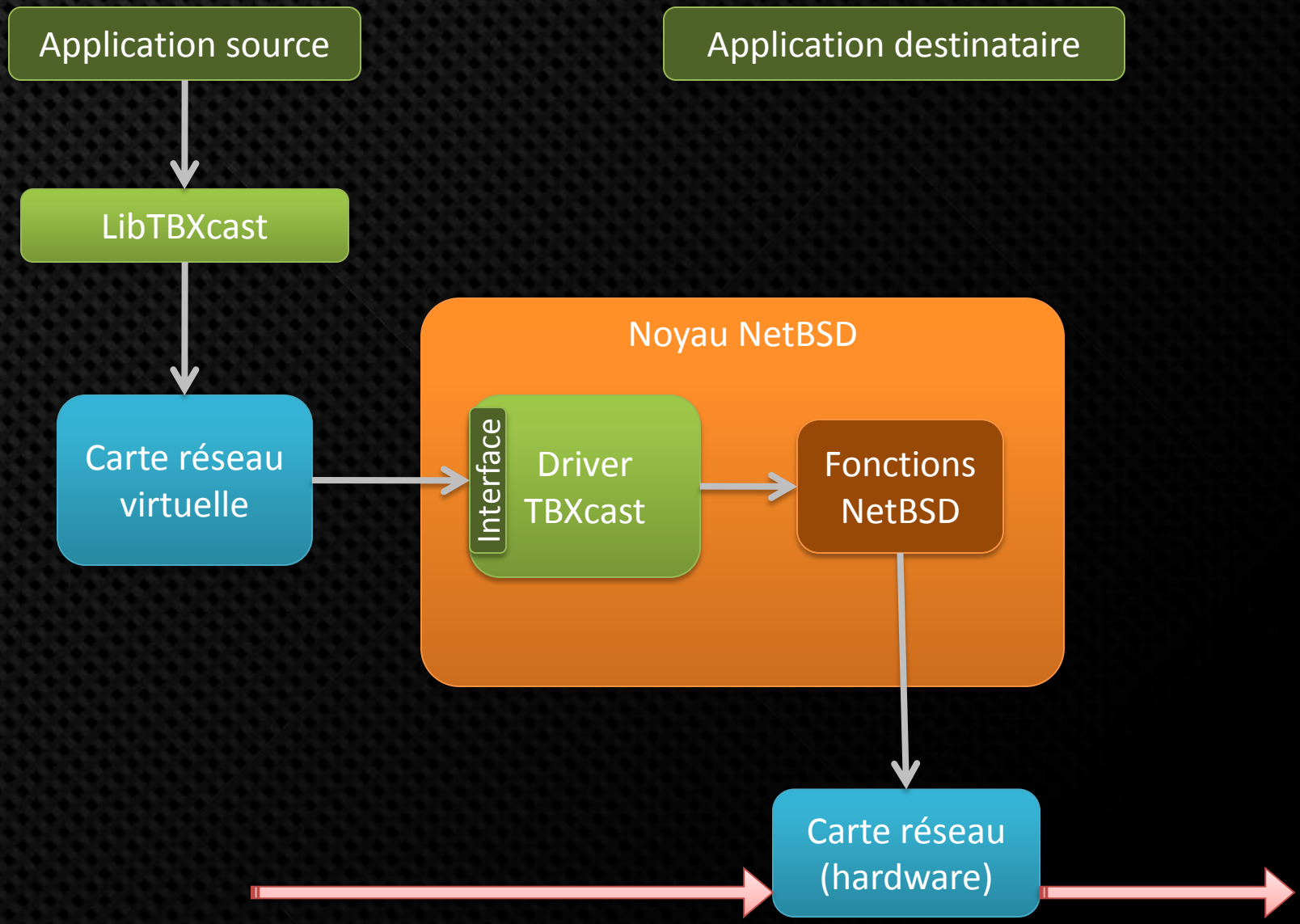
Aspect modulaire de l'architecture

- Décomposition en trois principaux modules
  - La librairie LibTBXcast, implémentée à la source
    - Récupère la topologie, construit l'arbre, gère les groupes
  - Driver TBXcast
    - Code noyau présent sur les routeurs
  - Application de test : TBXtest
    - Réalise le simple envoi d'un paquet à un groupe

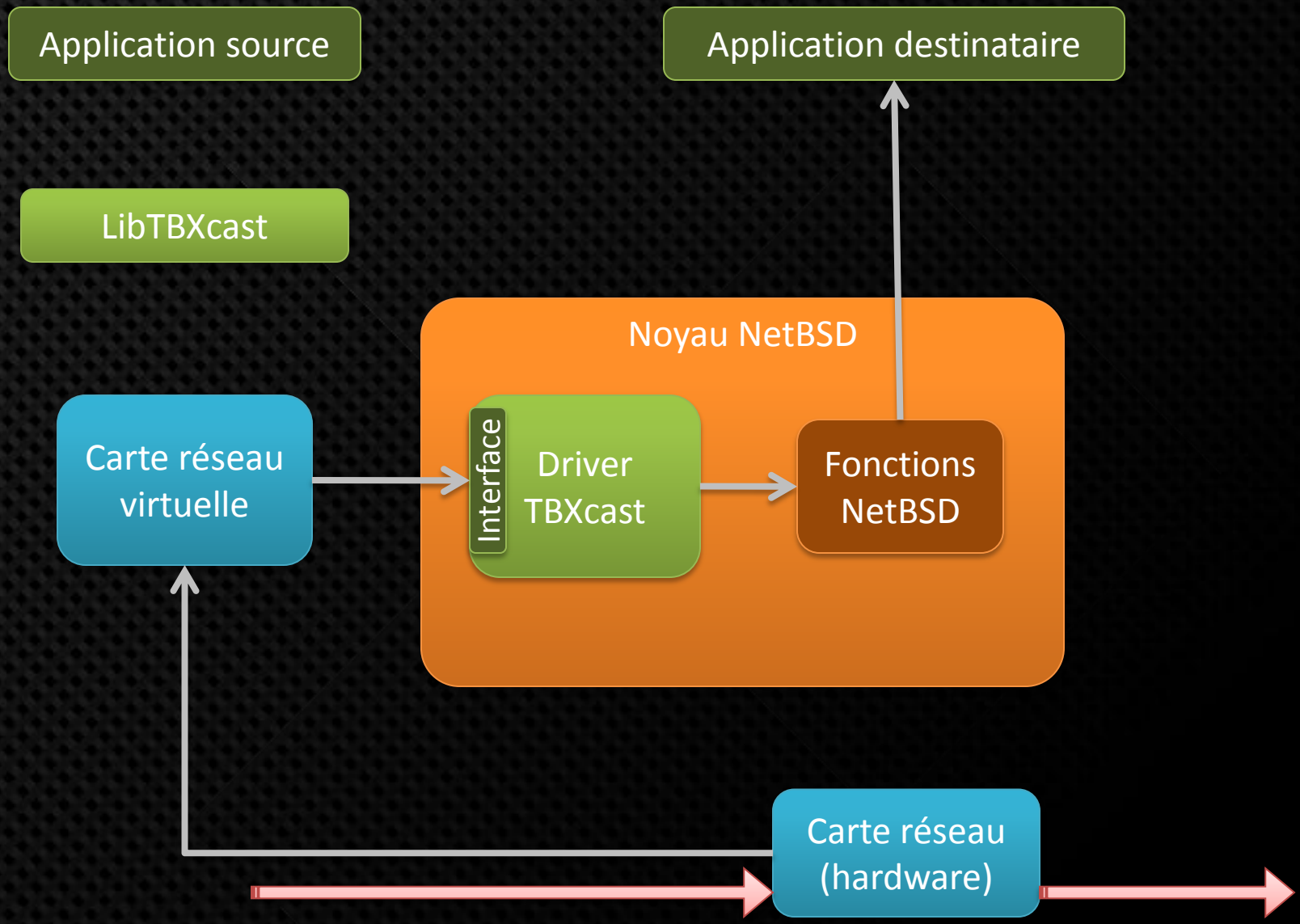
Composants


Application source
Application destinataire
LibTBXcast
Noyau NetBSD
Carte réseau virtuelle
Interface
Driver TBXcast
Fonctions NetBSD
Carte réseau (hardware)

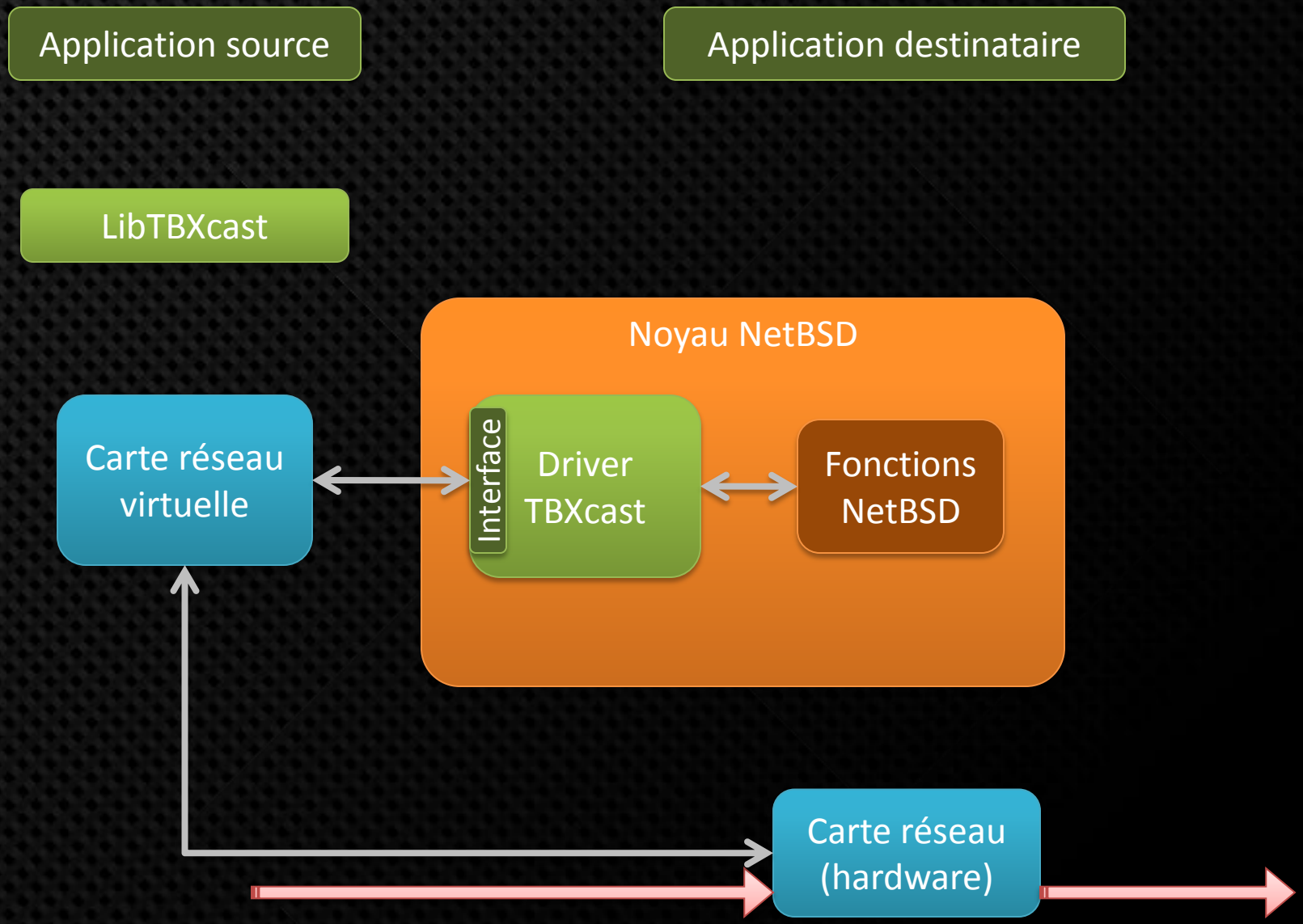
Envoi d'un paquet



Réception du paquet

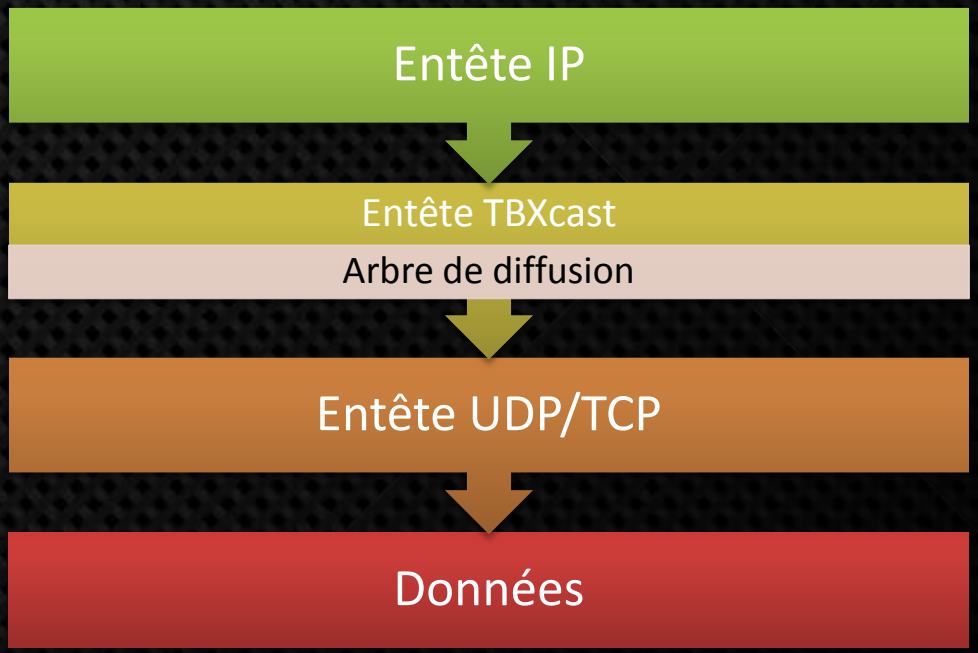


Routage d'un paquet



Entête et structure des paquets

- L'enchaînement des entêtes structure le paquet qui circule sur le réseau



Entête spécifique à notre protocole.
Le paquet est reconnu comme « TBXcast » par le routeur et envoyé depuis la carte réseau vers la carte virtuelle.

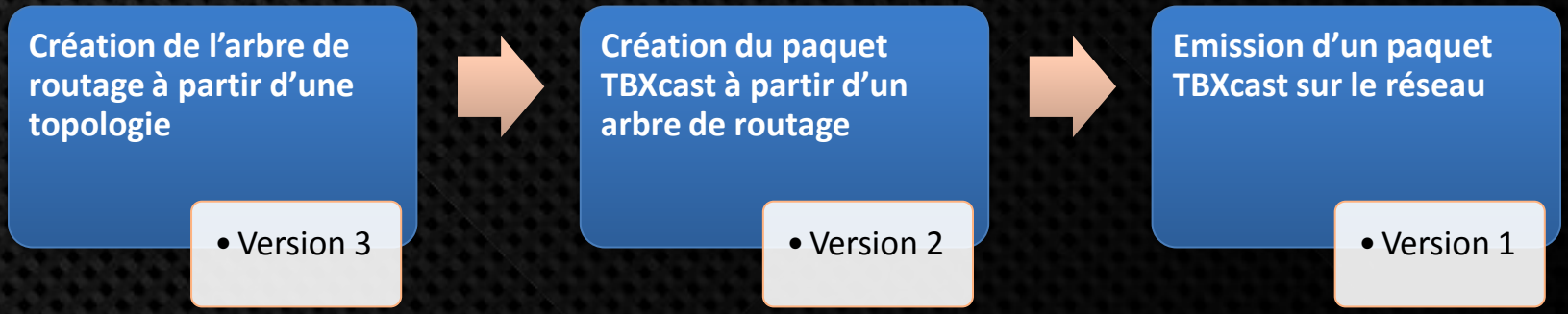
Implémentation de TBXcast

Organisation du développement

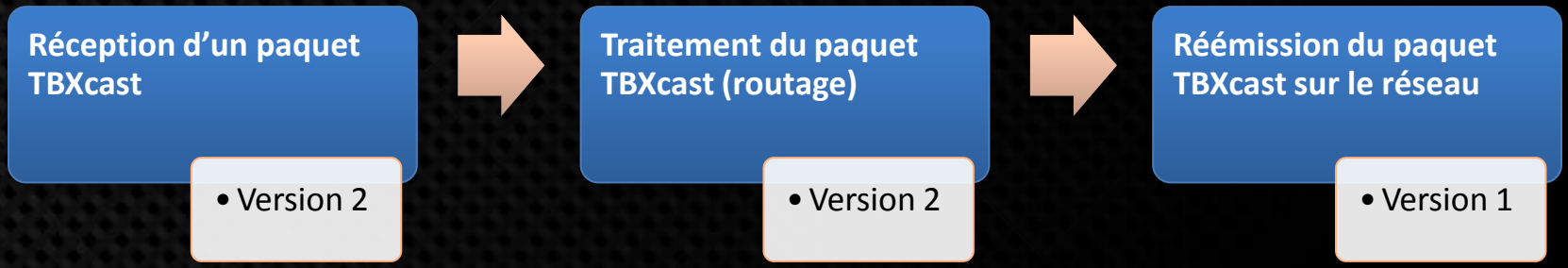
- Cycles « développement → test → validation » pour chaque version
- Huit versions incrémentales
  - Implémentation jusqu'à la version 3 pour cette année
  - Développement en parallèle des versions

Visualisation des versions

A la source



Au niveau d'un routeur



Version 0 : renommage

- Objectif : renommer des éléments du code de Xcast
  - Obention d'un protocole identique
  - Fonctionnement en parallèle possible
- Conservation des conventions de nommage
- Testé avec succès avec l'application TBXtest

Version 1 : implémentation du tunneling

- Rappels
  - Environnement hétérogène : tous les routeurs ne sont pas forcément compatibles avec le protocole TBXcast
  - Encapsulation du paquet TBXcast dans un paquet IPv6
- Objectifs
  - Utiliser systématiquement le tunneling classique pour le routage des paquets TBXcast
  - Créer un tunnel entre chaque routeur TBXcast explicitement codé dans l'arbre de routage

Principe du tunneling

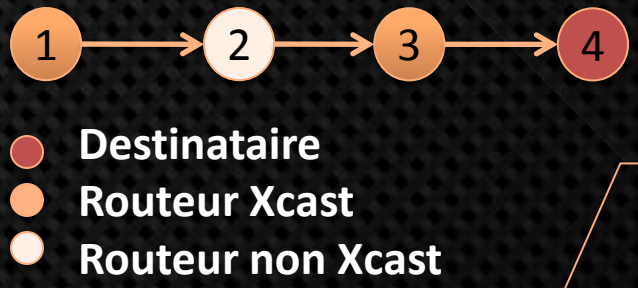
- **Routeur compatible avec TBXcast et codé dans l'arbre de routage**
- **Routeur incompatible avec TBXcast**

L'entête extérieur permet de mettre en œuvre un tunnel entre le routeur 1 et 3. Le routeur 2 traite le paquet comme un paquet IPv6 classique.

Entête IP extérieur	Entête IP intérieur	Entête TBXcast	Entête de transport
Source : 1 **Destination : 3**	Source : 1 Destination : TBXcast	Codage de l'arbre de routage	TCP / UDP

Tunneling sous Xcast

- Le tunneling semi-perméable
  - Xcast implémente un tunneling non adapté à notre problématique d'arbre



Sur une route, chaque routeur lit l'entête Hop-by-Hop afin de savoir s'il est capable d'interpréter la suite.

Entête IP extérieur	Entête Hop-by-Hop	Entête IP intérieur	Entête Xcast
Source : 1 Destination : 4	**Attention : la suite est un paquet Xcast**	Source : 1 Destination : Xcast	Liste des destinataires

Difficultés liées au tunneling

- Le tunneling semi-perméable
  - Inadapté à TBXcast
    - Inutile de tester si un routeur peut interpréter la suite
    - En effet, les routeurs compatibles TBXcast sont connus et codés dans l'arbre de routage
    - Seuls les routeurs codés dans l'arbre de routage doivent traiter le paquet TBXcast !
  - Mais
    - Implémentation du tunneling semi-perméable très minutieuse et étroitement liée aux fichiers systèmes de NetBSD
    - Modifications très techniques et manque de temps

Notre implémentation du tunneling

- Changement d'objectif
  - Conservation de l'implémentation du tunneling semi-perméable de Xcast
  - Hypothèse forte pour la suite
    - Sur une route donnée, tous les routeurs TBXcast sont codés dans l'arbre de routage

Version 2 : Routage arborescent

- Objectifs
  - On implémente la structure de notre arbre
  - Les routeurs doivent interpréter et router les paquets TBXcast

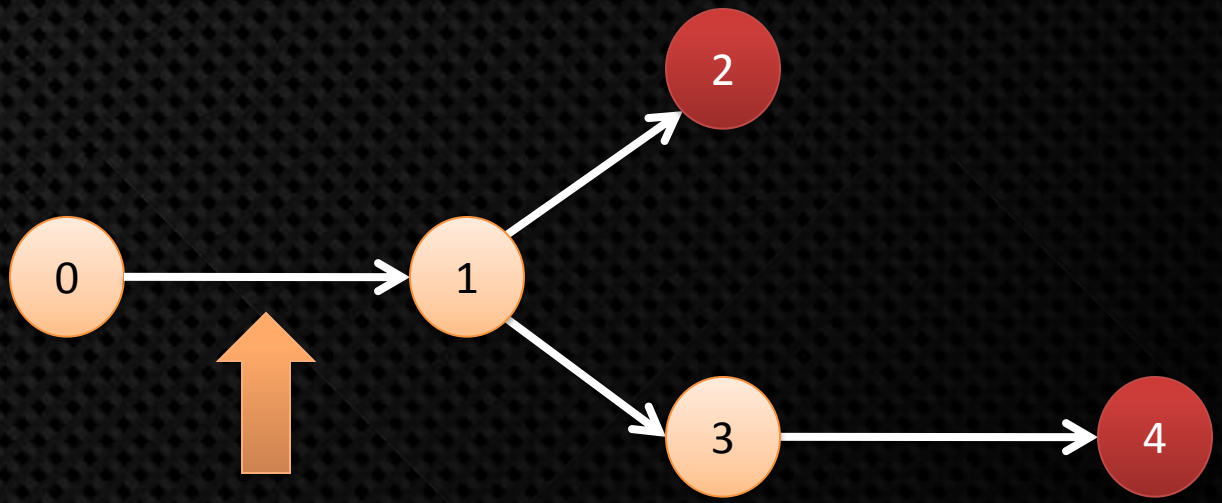
Structure de l'arbre

- Un nœud de l'arbre est une structure qui contient :
  - La longueur du sous-arbre qui lui est associé
  - L'adresse IP du nœud
  - Un booléen indiquant si ce nœud est destinataire ou non

L'algorithme de routage

- On se place sur la racine de l'arbre
  - Si ce nœud est un destinataire, les données sont remontées à l'application
- Pour chacun de ses fils directs
  - On construit un nouveau paquet TBXcast en élaguant l'arbre et on l'envoie au fils considéré

Déroulement d'un exemple (1/2)



Routeur		1	2	3	4
Structure	Adr.	@	@	@	@
	Long.	4	1	2	1
	Dest.	0	1	0	1

i routeurs destinataires

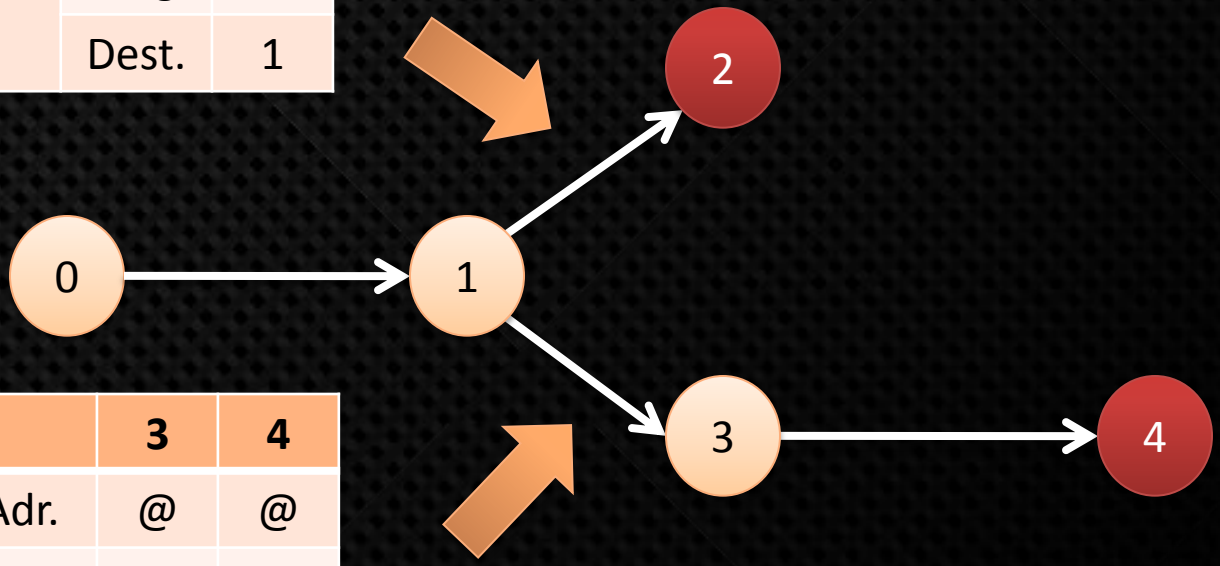
i routeurs intermédiaires

Entête TBXcast d'un paquet circulant entre le routeur 0 et le routeur 1

Déroulement d'un exemple (2/2)

Routeur		2
Structure	Adr.	@
	Long.	1
	Dest.	1

Entête TBXcast d'un paquet circulant entre le routeur 1 et le routeur 2



Routeur		3	4
Structure	Adr.	@	@
	Long.	2	1
	Dest.	0	1

Entête TBXcast d'un paquet circulant entre le routeur 1 et le routeur 3

i routeurs destinataires

i routeurs intermédiaires

Test de la version 2

- Driver TBXcast non testé
  - Problème lors du passage entre la librairie LibTBXcast et le driver TBXcast
  - Solution : coder le paquet « en dur »

Version 3 : construction de l'arbre

- Objectifs
  - Construction de l'arbre de routage à la source à partir d'une topologie fournie manuellement dans la librairie LibTBXcast
  - Ajout de l'arbre construit dans l'entête du paquet TBXcast

Représentation de la topologie

- Ensemble de liens entre interfaces
- Structure
  - Source
  - Destinataire
  - Adresse IP de la destination

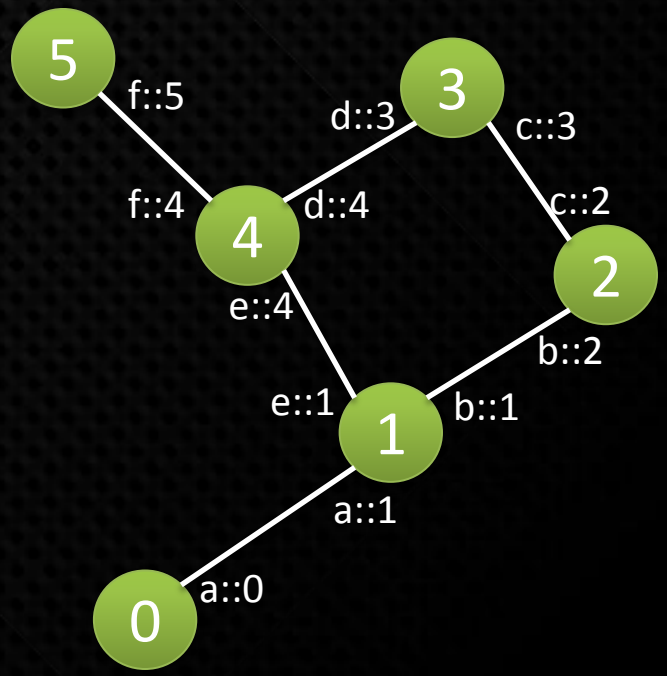
{ source = 0, dest = 1, addr = a::1 }
{ source = 1, dest = 0, addr = a::0 }

Algorithme de construction de l'arbre

- Arbre couvrant minimal à partir du graphe qui décrit la topologie
  - On utilise l'algorithme de Moore-Dijkstra
  - L'algorithme travaille sur une matrice des liens
- A chaque étape
  - On sélectionne le nœud non encore pris le plus proche de la source
  - On met à jour les distances et les prédécesseurs
- Arrêt quand tous les destinataires sont couverts

Exemple de matrice des liens

	0	1	2	3	4	5
0		1				
1	1		1		1	
2		1		1		
3			1		1	
4		1		1		1
5					1	



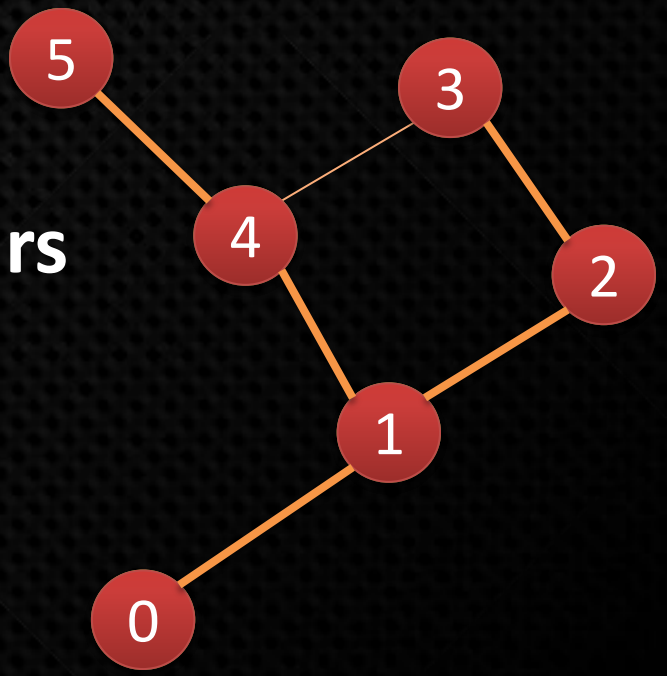
Exemple du déroulement de l'algorithme

- Envoi depuis 0 jusqu'à 2 et 5
- Tableau des longueurs

0	1	2	3	2	3

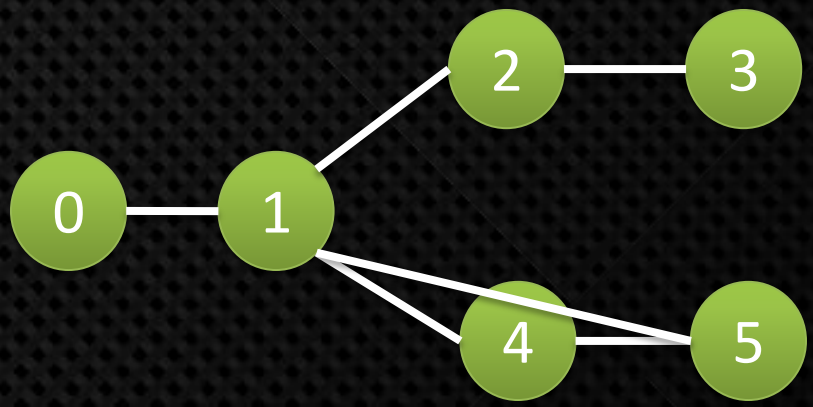
- Tableau des prédécesseurs

-1	0	1	2	1	4



Traitements supplémentaires

- Elagage de l'arbre
  - Suppression des branches sans destinataire



- Suppression des nœuds inutiles au routage

Test de la version 3

- Protocole de test
  - Construction de la topologie
  - Création d'un groupe
    - Ajout de membres
  - Construction de l'arbre
    - Grâce à l'algorithme précédemment présenté
- Tests unitaires effectués
  - Algorithme de création de l'arbre
  - Gestion de la topologie
  - Ajout et retrait de membres

Présentation de la plateforme

Nécessité d'une plateforme

- Code dans le noyau de NetBSD
  - Besoin de postes sous NetBSD
  - Besoin de systèmes dédiés
- Développement d'un protocole de routage
  - Besoin d'un réseau de test conséquent
  - Besoin de simuler de nombreuses topologies

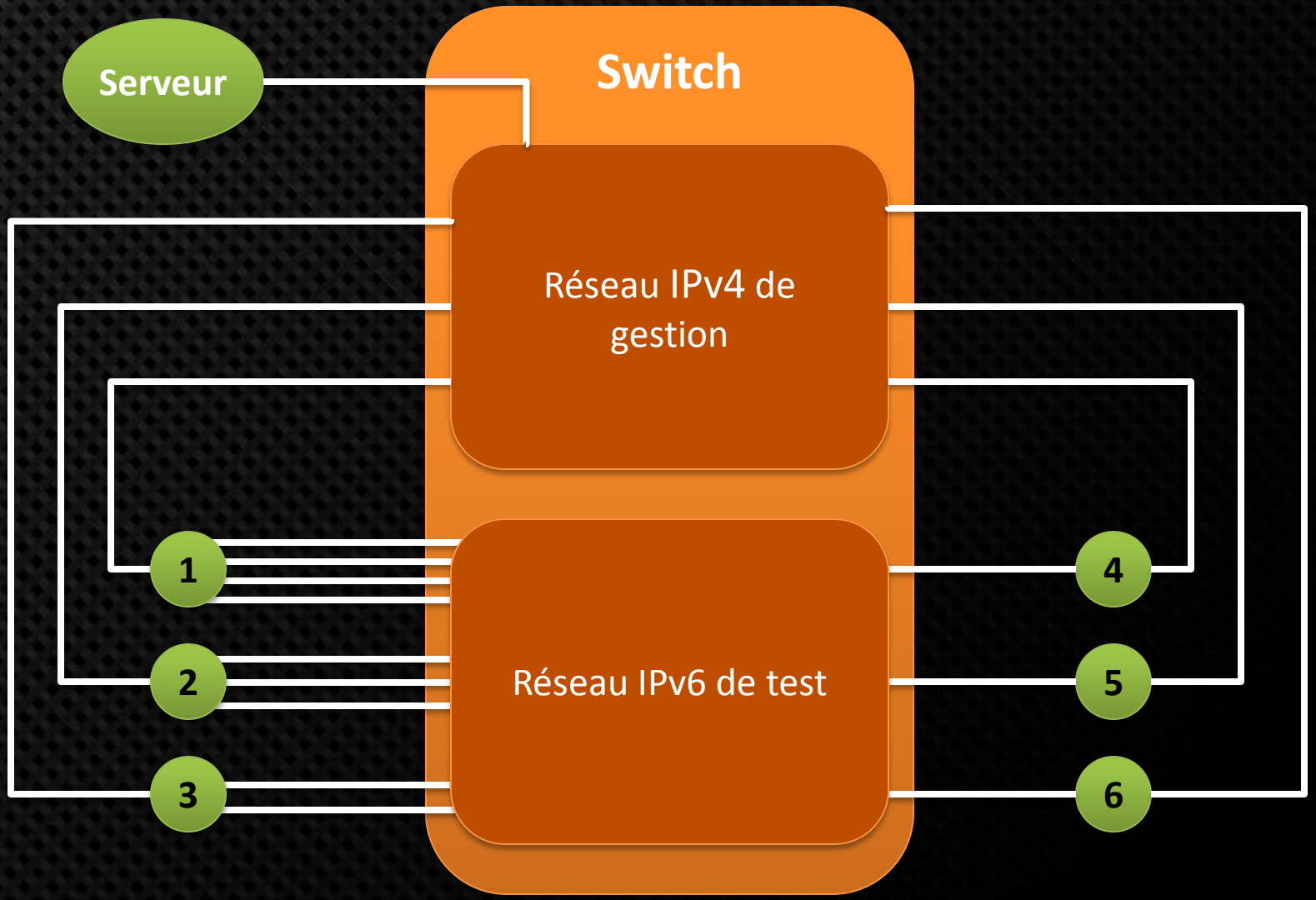
Plateforme initiale

- Configurations hétérogènes
  - Administration complexe
- Aucune centralisation
  - Données difficiles à récupérer
  - Systèmes indépendants
- Aucun système de gestion
  - Compilation longue
  - Gestion du réseau fastidieuse
  - Tests lourds

Solutions mises en place

- Renouvellement du matériel
- Serveur Netboot
  - Démarrage par le réseau
  - Stockage des systèmes sur le serveur
  - Centralisation des données
  - Facilité de la maintenance
- Programmes de gestion
  - Gestion des machines
  - Gestion du réseau

Architecture



Programmes de gestion de la plateforme

- Makefile
  - Mise en cache des versions
  - Compilation unique
  - Installation globale
- Gestion des machines : TBXpower
  - Arrêt et redémarrage des machines
  - Démarrage avec la technologie Wake On Lan
- Gestion du réseau : TBXnet
  - Etablissement de liens entre deux machines
  - Chargement d'une topologie complète
  - Administration transparente du switch et des machines

Déroulement d'un test

- Compilation et installation
  - Makefile : `make genbuild VERSION=v0`
- Redémarrage
  - TBXPower : `tbxpwr reboot 1-6`
- Configuration du réseau
  - TBXNet : `tbxnet config etoile`
- Test de la connectivité
  - ping6 : `ping6 a:12::2`
- Lancement du test
  - tbxtest : `tbxtest a:12::2`
  - tcpdump : `tcpdump -i rtk0`

Plateforme : bilan

- Compilation des éléments en 10 minutes
- Préparation de la plateforme en quelques commandes
- Automatisation des configurations complexes
- Tests devenus faisables
- Rédaction complète d'un manuel d'utilisation de la plateforme

Bilan du projet

Les versions futures de TBXcast (1/2)

- Version 4 : segmentation
  - Optimisation de la charge sur les arbres que la source envoie
- Version 5 : gestion des groupes
  - Gestion dynamique de l'ajout et du retrait des membres d'un groupe

Les versions futures de TBXcast (2/2)

- Version 6 : Récupération de la topologie
  - Utilisation du protocole OSPF
- Version 7 : Qualité de Service
  - Construction de l'arbre suivant un critère de qualité choisi

Apports du projet : aspect technique

- Apprentissage en réseaux et système
- Approfondissement et mise en pratique de l'algorithmique des graphes
- Manipulation du code de très bas niveau
  - Noyau de NetBSD
  - Driver TBXcast
- Installation, maintenance et utilisation d'un réseau sous IPv6
- Réaffirmation de l'importance du commentaire de code ☺

Apports du projet : aspect organisation

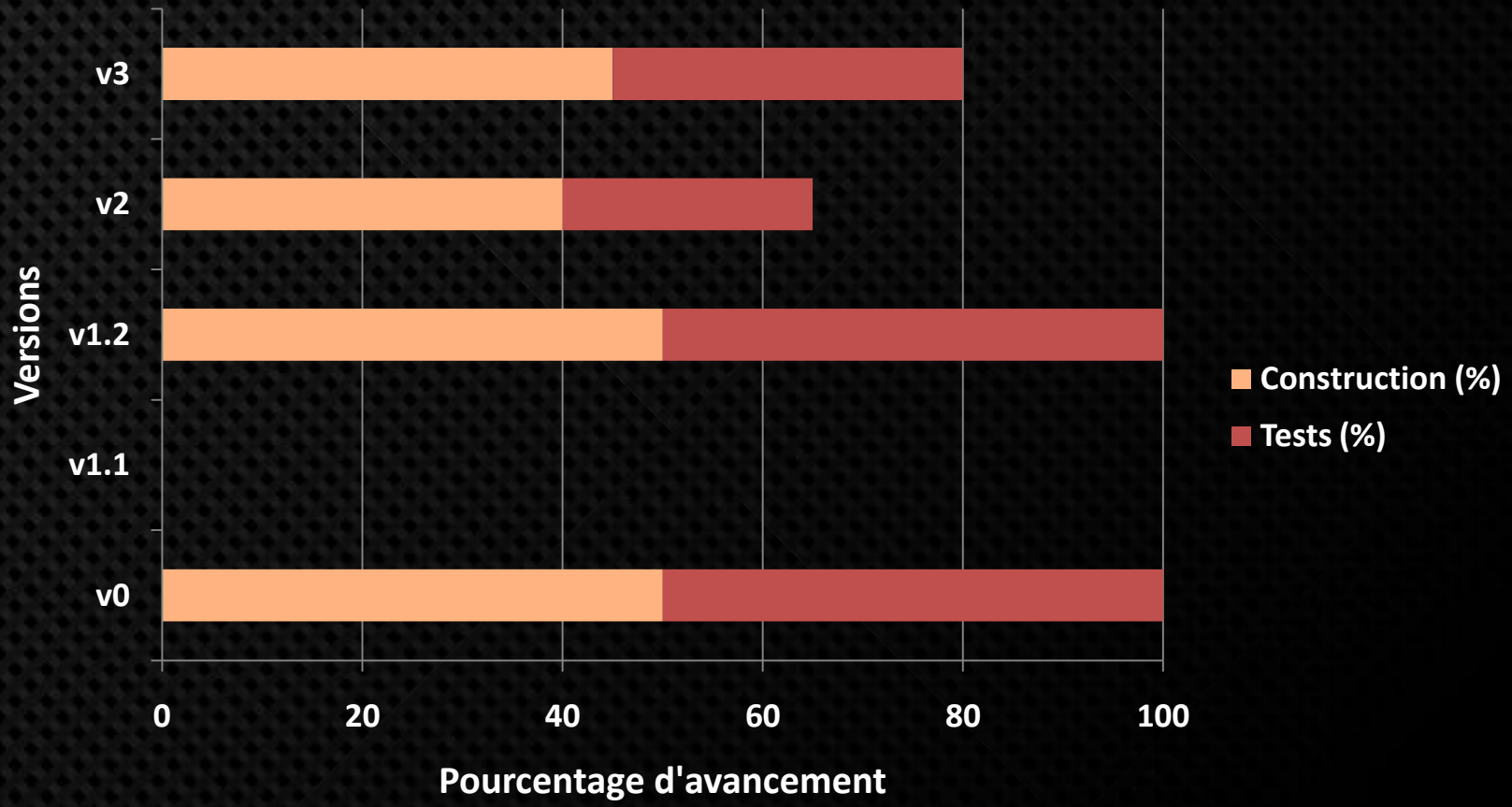
- Apprentissage de méthodologies de rédaction, de présentation
- Mise en évidence de l'importance
  - De la gestion du temps
  - Du dynamisme, de la motivation
  - De ne pas se laisser décourager par la difficulté du code

Planification

Retour sur la planification

- Evolution
  - A l'origine, versions développées en séquence
  - Puis, possibilité de développement en parallèle
- Répartition
  - 3 groupes de 2 personnes pour les versions
  - 1 personne pour la plateforme

Avancement des versions



Etat d'avancement du projet (1/3)

Tâches	2007/2008	2008/2009	Commentaires
Partie documentaire			
Documentation de Xcast			Structures et fonctions documentées.
Conception du fonctionnement de TBXcast			Fonctionnement défini. Structures et algorithmes fixés.

Non commencée

Entamée

Presque finie

Complète

Etat d'avancement du projet (2/3)

Tâches	2007/2008	2008/2009	Commentaires
Partie driver			
Traitement des entêtes des paquets			Algorithme de routage écrit. Non testé.
Gestion des routeurs non compatibles TBXcast			Implémentation du tunneling comme dans Xcast.

- **Non commencée**
- **Entamée**
- **Presque finie**
- **Complète**

Etat d'avancement du projet (3/3)

Tâches	2007/2008	2008/2009	Commentaires
Partie librairie			
Découverte de topologie réseau			Gestion d'OSPF non incluse.
Calcul de l'arbre de routage à partir de la topologie			Calcul de l'arbre implémenté et fonctionnel.
Envoi des paquets			Méthode écrite mais non opérationnelle.

- **Non commencée**
- **Entamée**
- **Presque finie**
- **Complète**

Conclusion

- Le projet TBXcast est très technique
  - Code de bas niveau
  - Connaissances réseaux apprises par nous-mêmes
- Les difficultés rencontrées sont d'ordre technique
  - Code noyau difficile à appréhender
  - Erreurs difficiles à localiser
- Le projet a fait un grand pas cette année
  - Une plateforme performante
  - Une équipe dynamique et motivée

Merci pour votre attention

tbxcast.xipp.net