



PeerFuse

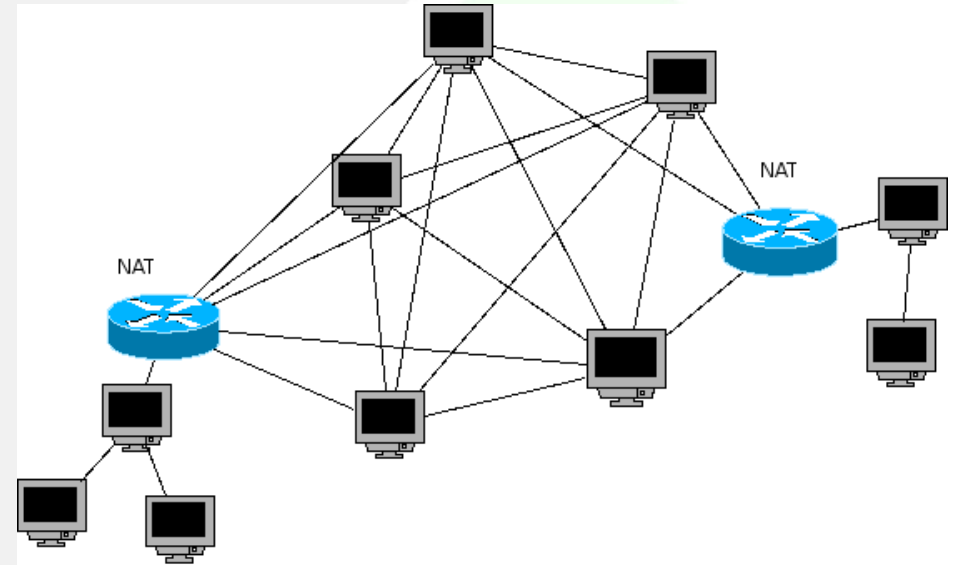
Le système de fichier pair-à-pair distribué

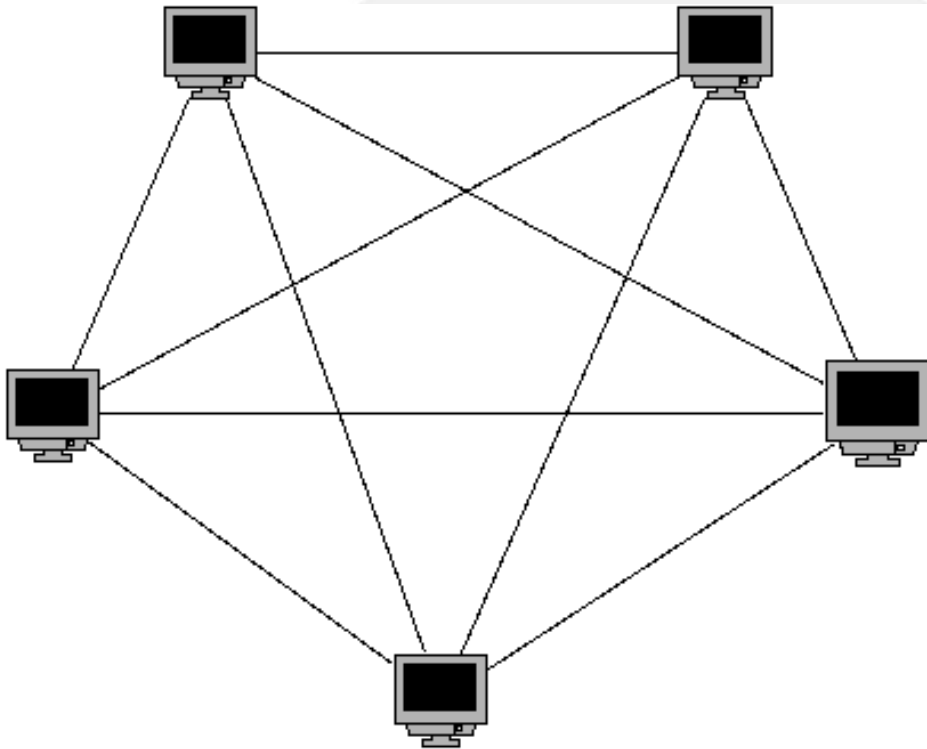
- Présentation de Peerfuse
 - Le but recherché
 - Les fonctionnalités
- Utilisation de la bibliothèque FUSE
 - L'API
- Solutions aux problématiques de sécurité réseau
 - DHT
 - Protocole
 - Chiffrement des messages et des données avec OpenSSL
 - Protections contre les attaques

- Initié fin janvier 2008
 - Développé par Laurent Defert et Romain Bignon
 - Écrit en C++ avec la bibliothèque FUSE
 - Toujours en cours de développement et non utilisable en prod
- Alternatives
 - Pastis/Pastry
 - Freenet

- Espace de partage commun
- Entièrement décentralisé
- Mode déconnecté
- Deux visions différentes :
 - Peerfuse-lan : déploiement au sein d'un réseau local
 - Peerfuse-net : utilisation via Internet

- UID unique par utilisateur
- Possibilités de créer des groupes d'utilisateurs
- Chiffrement des données
- Routage entre les pairs

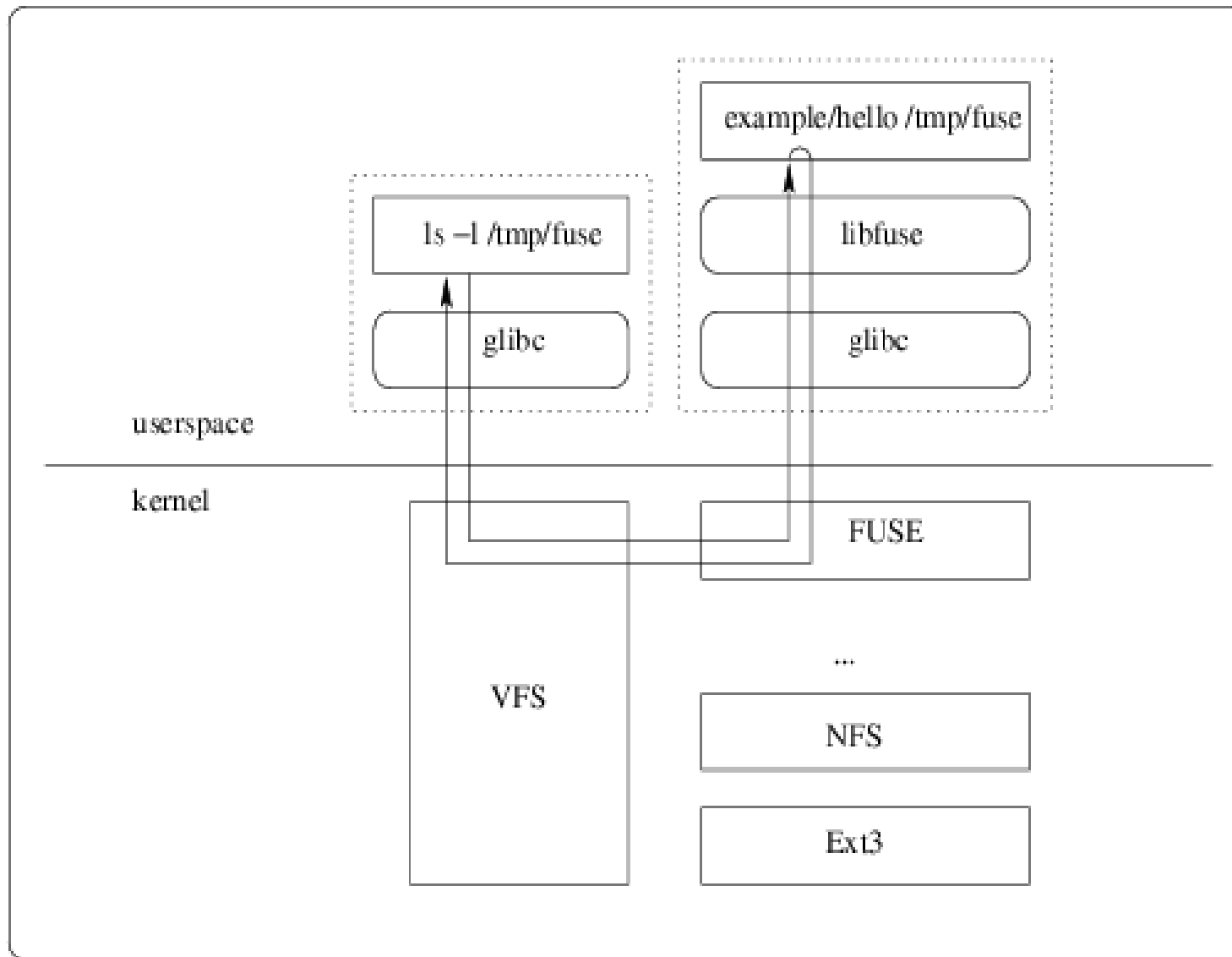




- Utilisation des permissions UNIX des systèmes hôtes.
- Permettre de monter un /usr commun à plusieurs machines.

- Filesystem in UserSpace
- Modules noyau pour Linux, FreeBSD, Mac OS X, Windows, OpenSolaris, Hurd, ...
- Systèmes de fichiers virtuels :
 - SSHFS
 - WikipediaFS
 - GmailFS
 - NTFS-3G

FUSE



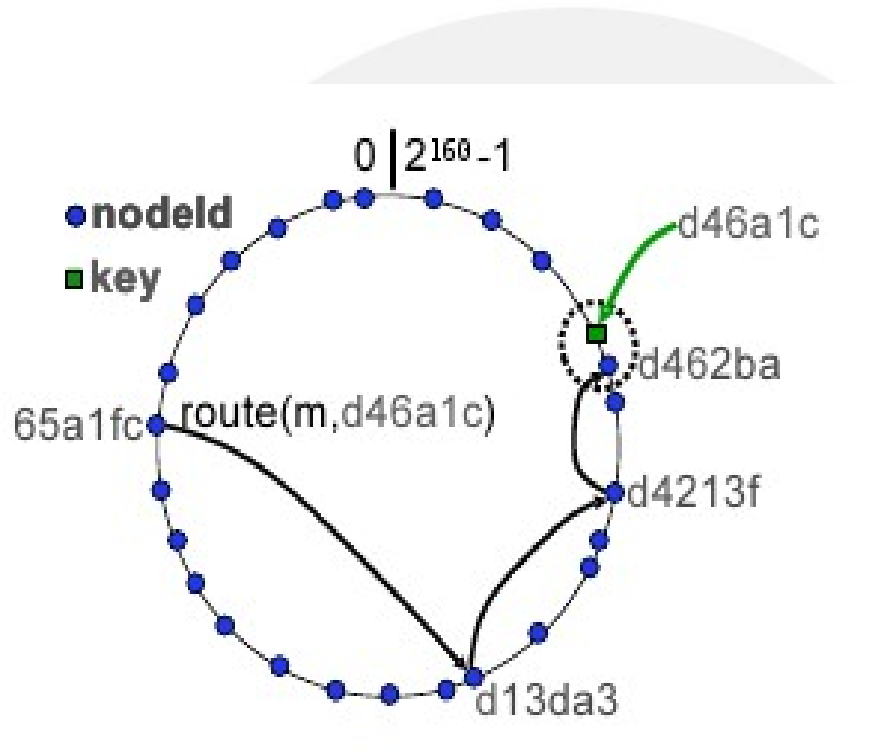
FUSE - API



```
int (*getattr) (const char *, struct stat *);
int (*readlink) (const char *, char *, size_t);
int (*mknod) (const char *, mode_t, dev_t);
int (*mkdir) (const char *, mode_t);
int (*unlink) (const char *);
int (*rmdir) (const char *);
int (*symlink) (const char *, const char *);
int (*rename) (const char *, const char *);
int (*link) (const char *, const char *);
int (*chmod) (const char *, mode_t);
int (*chown) (const char *, uid_t, gid_t);
int (*truncate) (const char *, off_t);
int (*utime) (const char *, struct utimbuf *);
int (*open) (const char *, struct fuse_file_info *);
int (*read) (const char *, char *, size_t, off_t,
            struct fuse_file_info *);
int (*write) (const char *, const char *, size_t, off_t,
            struct fuse_file_info *);
int (*statfs) (const char *, struct statvfs *);
int (*flush) (const char *, struct fuse_file_info *);
int (*release) (const char *, struct fuse_file_info *);
int (*fsync) (const char *, int, struct fuse_file_info *);
int (*setxattr) (const char *, const char *, const char *, size_t, int);
int (*getxattr) (const char *, const char *, char *, size_t);
int (*listxattr) (const char *, char *, size_t);
int (*removexattr) (const char *, const char *);
int (*opendir) (const char *, struct fuse_file_info *);
int (*readdir) (const char *, void *, fuse_fill_dir_t, off_t,
            struct fuse_file_info *);
int (*releasedir) (const char *, struct fuse_file_info *);
int (*fsyncdir) (const char *, int, struct fuse_file_info *);
void *(*init) (struct fuse_conn_info *conn);
void (*destroy) (void *);
int (*access) (const char *, int);
int (*create) (const char *, mode_t, struct fuse_file_info *);
int (*ftruncate) (const char *, off_t, struct fuse_file_info *);
int (*fgetattr) (const char *, struct stat *, struct fuse_file_info *);
int (*lock) (const char *, struct fuse_file_info *, int cmd,
            struct flock *);
int (*utimens) (const char *, const struct timespec tv[2]);
int (*bmap) (const char *, size_t blocksize, uint64_t *idx);
```

Protocole réseau



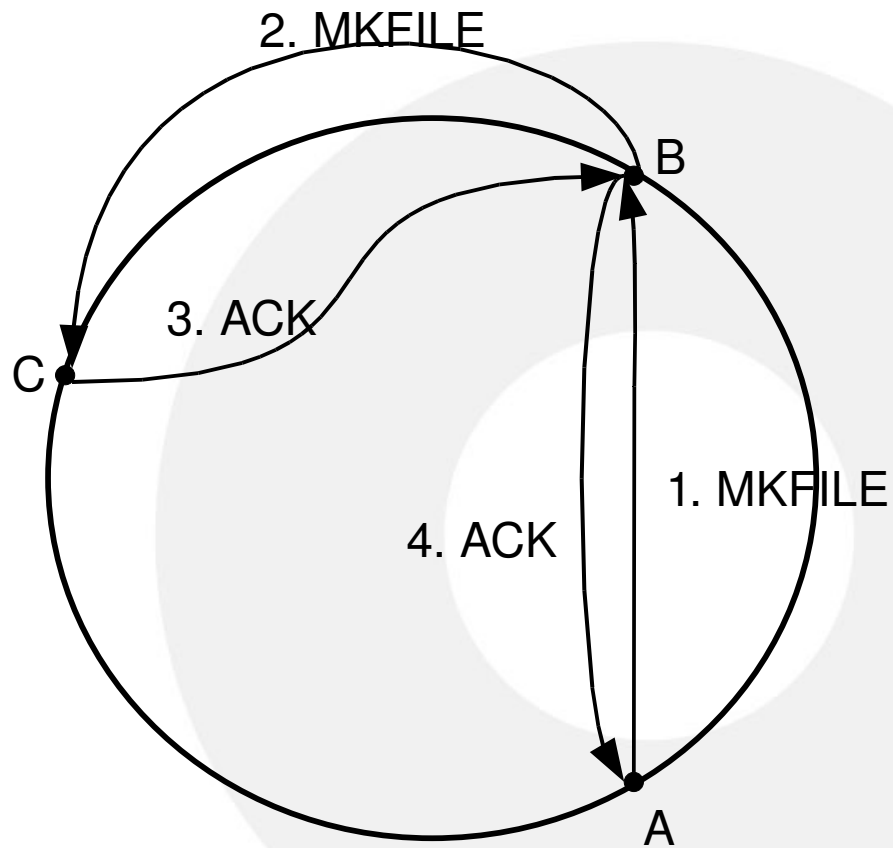


- Identifiant unique sur 160 bits (SHA-1 de la clef publique)
- Représentation sur un cercle
- Routage des messages

- Paire de clefs publique/privée
- UID 160 bits
- Connexion au premier pair
- Mise en relation
- Merge des données

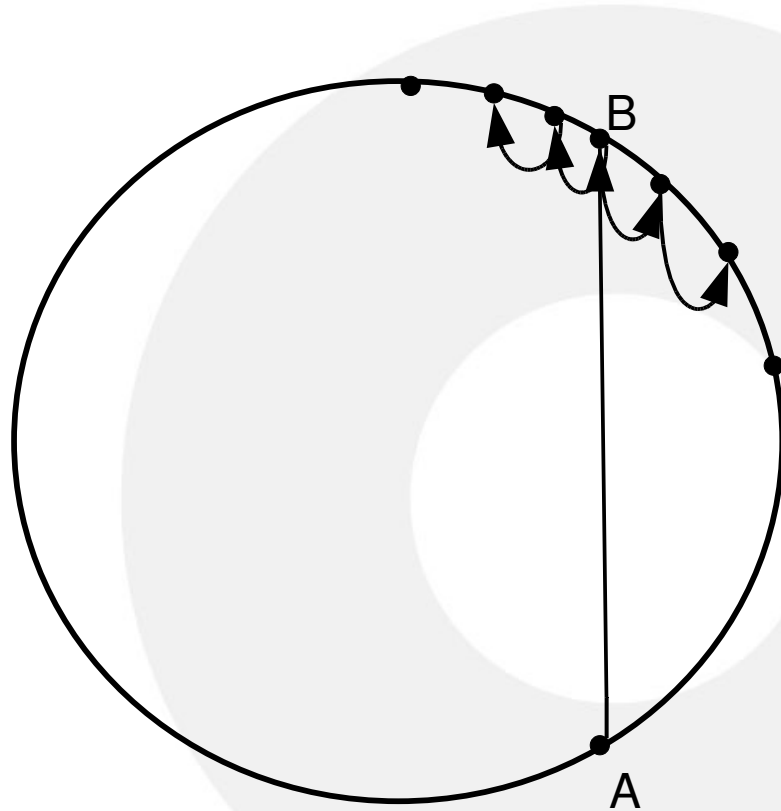
- Chiffrement et signature des messages :

Perms	Cryptage	Signature
RW	User private key	No
RW R	Group private key	User public key
RW R R	No	User public key
RW RW	Group private key	No
RW RW R	No	Group public key
RW RW RW	No	No

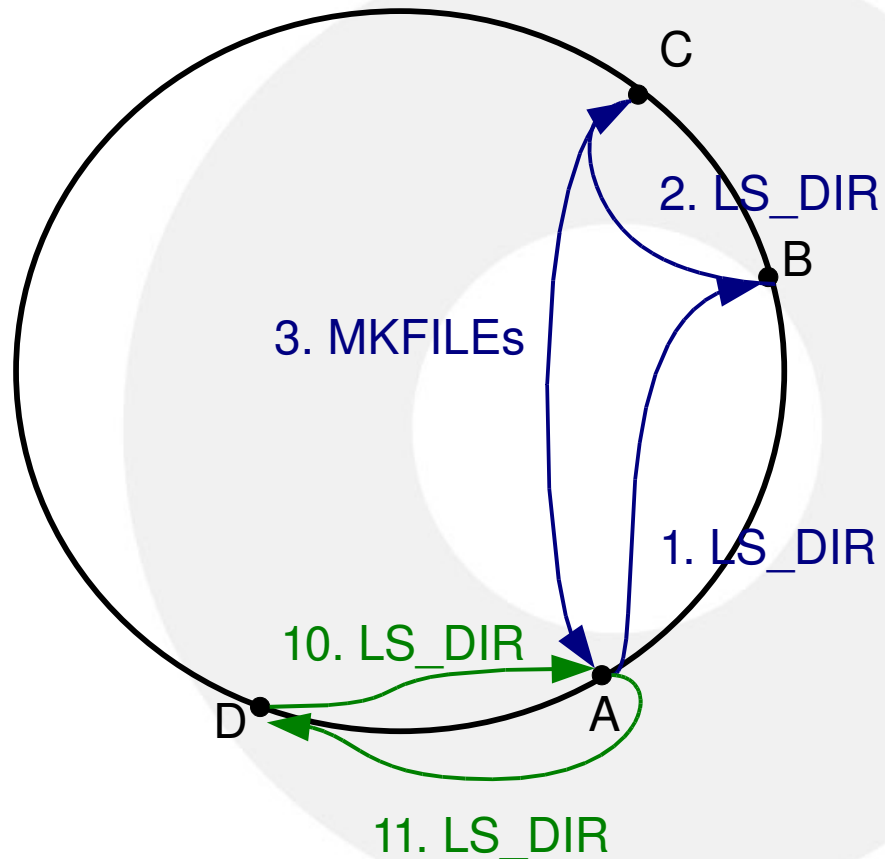


A crée le fichier
B responsable dossier
C responsable fichier

A -> B : message de création
B vérifie la faisabilité
B stocke dans les
informations dossier
B -> C : message de création
C -> B : ACK
B -> A : ACK

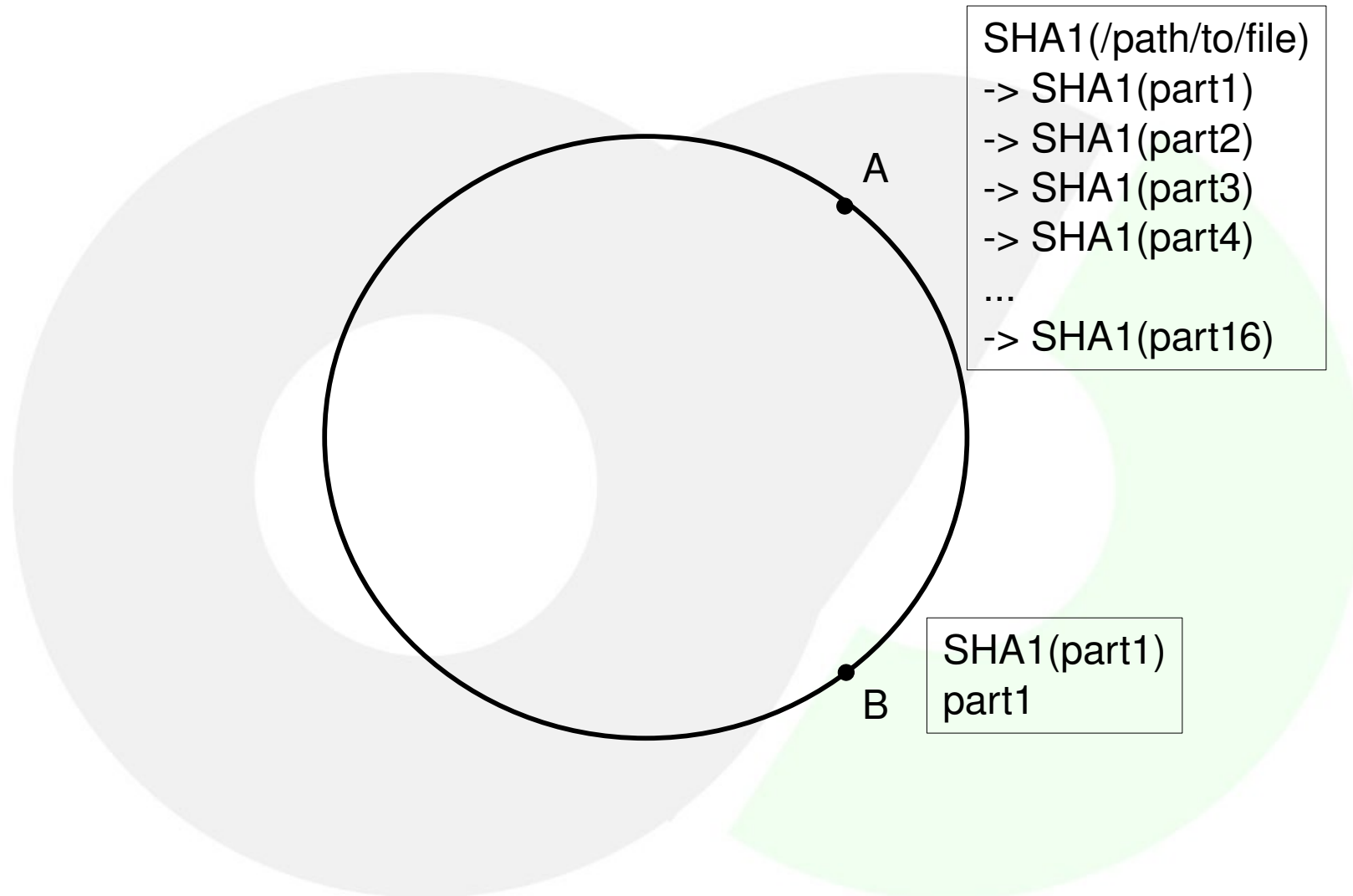


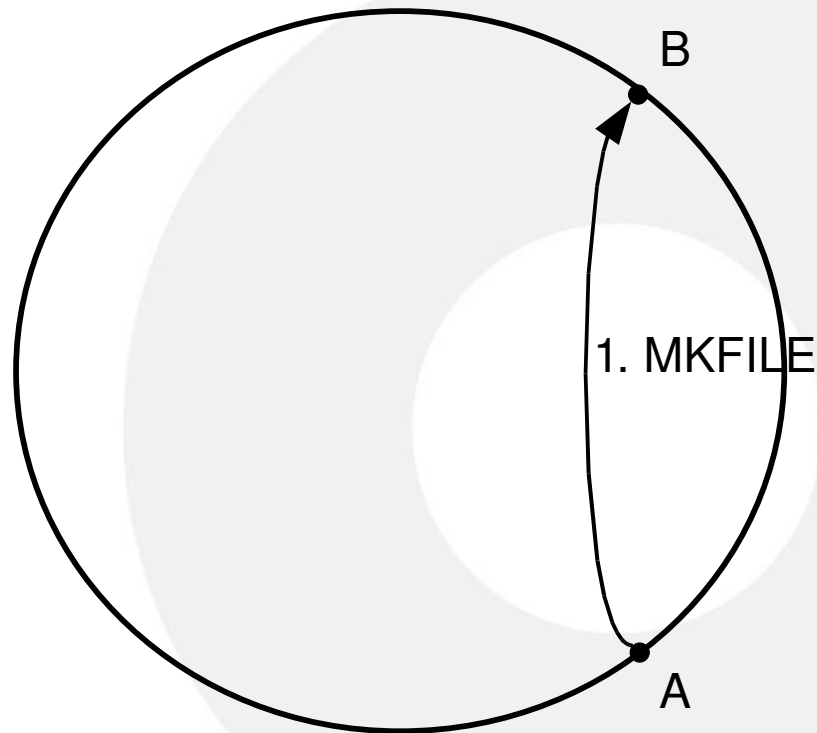
- Chaque message est répliqué sur les pairs voisins
- Si jamais B tombe, le pair le plus proche prends la relève



C responsable dossier

- A récupère le contenu du répertoire
- A le garde en cache
- D demande le contenu du répertoire
- Le message passe par A qui peut lui répondre directement



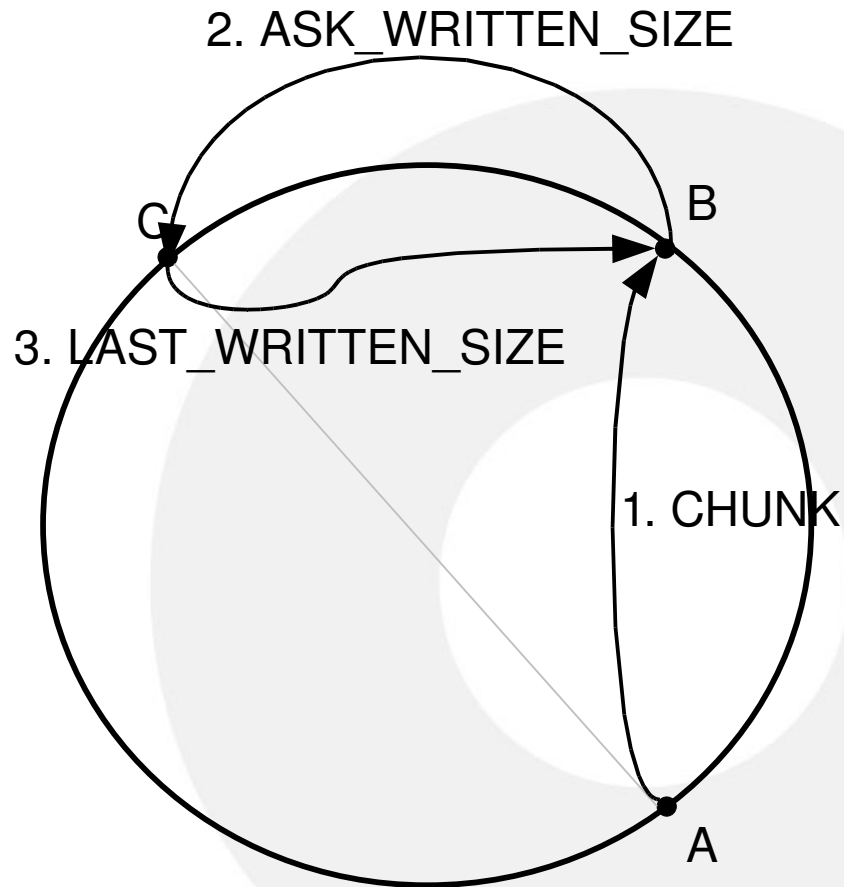


A écrit dans le fichier
B responsable du fichier

A -> B : Modification du
champ Sharers des meta
data

- Seul A est sharer
- Message bien évidemment
signé pour prouver sa
légitimité à écrire dedans

Limitation des données redondées



A veut envoyer des data

B responsable fichier

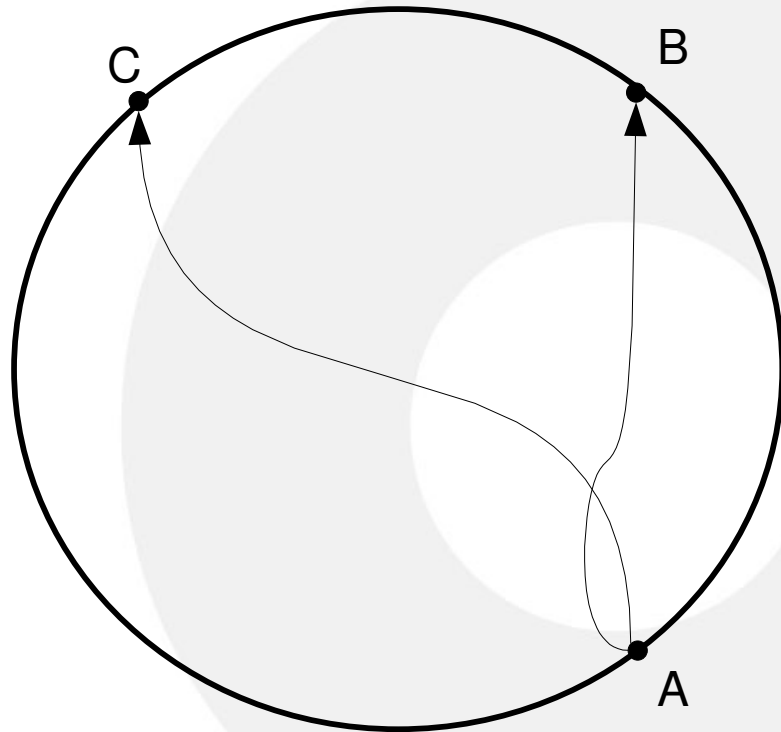
C pair dont l'ID est à l'opposé de A sur le cercle

A -> B : envoie chunk + taille data déjà écrites


B -> C : envoie taille data signé par A

C -> B : réponds l'ancienne taille, et stock la nouvelle

B vérifie qu'il n'y a pas eu dépassement, et accepte ou non



- Récupérations des données depuis la DHT
- IDs sharers stockés dans les meta-data

- Système sécurisé
 - Gros développements en cours
 - Appel à contributions
- 

Questions ?

