

TorUsNet X FR

Architecture directionnelle à Tore Privé auto-mutant

L'Architech (Gabriel Breton-Breault)

Introduction

TorUsNet X représente une nouvelle étape dans le développement de l'architecture directionnelle. Cette évolution conserve les fondations établies dans TorUsNet 9.0, notamment le Tore Privé volumétrique de $100 \times 100 \times 100$, les ID internes, les états normal/fantôme, les déplacements directionnels et les mécanismes de mutation.

La différence fondamentale de TorUsNet X réside cependant dans la nature même de la mutation.

Dans les versions précédentes, le parcours directionnel pouvait modifier différents états internes du tore. TorUsNet X introduit un mécanisme supplémentaire dans lequel **les cases elles-mêmes peuvent permuter au cours de l'évolution du Tore Privé**.

Le Tore Privé ne constitue donc plus uniquement une structure dans laquelle les données internes changent. Sa propre organisation directionnelle peut être réorganisée entre deux parcours.

Le principe général devient :

$$T_n \xrightarrow{\text{parcours}} T_{n+1}$$

avec :

$$T_n \neq T_{n+1}$$

Le tore possède ainsi plusieurs formes successives. Chaque nouvel état résulte du parcours précédent et des transformations déterminées par les cases rencontrées.

Les ID comme mécanisme de sélection

TorUsNet X utilise les **ID des cases traversées** pour déterminer quelles cases doivent participer au processus de permutation.

Dans l'architecture précédente, les ID sont représentés sur 16 bits et constituent des marqueurs directionnels pouvant être associés aux cases du Tore Privé.

Dans TorUsNet X, leur lecture acquiert une nouvelle fonction.

Les ID rencontrés pendant le parcours sont lus alternativement :

$$\begin{aligned} ID_1 &\rightarrow \\ ID_2 &\leftarrow \\ ID_3 &\rightarrow \\ ID_4 &\leftarrow \\ &\dots \end{aligned}$$

Le premier ID est lu dans son ordre normal.

Le deuxième est lu dans l'ordre inverse.

Le troisième revient à une lecture normale.

Le quatrième est de nouveau inversé.

Cette alternance se poursuit pendant l'ensemble du parcours.

Lorsque la valeur obtenue est **impaire**, la case correspondante devient une candidate à la permutation :

$$\boxed{R(ID_i) \bmod 2 = 1 \Rightarrow C_i \in S}$$

où S représente l'ensemble des cases sélectionnées.

La sélection dépend donc directement **du contenu directionnel rencontré pendant le parcours**.

La permutation de la case complète

La permutation ne concerne pas uniquement le bit normal et le bit fantôme.

Dans TorUsNet X, une case doit être considérée comme une structure directionnelle complète.

Lorsqu'une case permute avec une autre, les propriétés qui lui appartiennent se déplacent avec elle, notamment son ID et ses propriétés directionnelles associées.

Le mécanisme peut être représenté simplement par :

$$C_A \leftrightarrow C_B$$

Après la permutation, la structure qui occupait la position A se retrouve en B , tandis que celle qui occupait B prend la position A .

Le tore change donc réellement de configuration.

Mutation des états + Permutation des cases = Mutation structurelle du tore

Un Tore Privé en évolution permanente

Cette mécanique introduit une caractéristique centrale de TorUsNet X :

Le parcours transforme le tore qu'il vient lui-même de parcourir.

Le Tore Privé utilisé pour un parcours donné n'est donc plus nécessairement identique à celui qui sera utilisé pour le parcours suivant.

La séquence générale devient :

Lecture → ID → Sélection → Inversion → Permutation → Nouvel état du tore

Chaque parcours participe ainsi à la construction de la prochaine configuration.

Deux états temporaires pour la synchronisation

TorUsNet X introduit également une nouvelle logique de synchronisation entre le Client et le Host.

Le Client applique d'abord le parcours à son Tore Privé et génère :

$$T_{n+1}$$

Pendant cette opération, le Host conserve temporairement l'état précédent :

$$T_n$$

tout en préparant lui aussi le nouvel état :

$$T_{n+1}$$

Le Host maintient donc temporairement deux états :

$$T_n + T_{n+1}$$

Lorsque le Client confirme que le parcours a été correctement appliqué, le Host peut valider la transition et effectuer le basculement :

$$T_n \rightarrow T_{n+1}$$

Le nouvel état devient alors le Tore Privé actif utilisé pour poursuivre la session.

Cette évolution prolonge le principe de synchronisation déjà présent dans TorUsNet 9.0 où Host et Client exécutent des étapes coordonnées afin de maintenir la cohérence de leurs Tores Privés.

Une architecture qui se réécrit elle-même

TorUsNet X peut ainsi être représenté par la chaîne suivante :

$$T_n \rightarrow \text{Parcours} \rightarrow \text{Lecture alternée des ID} \rightarrow \text{Sélection des cases} \rightarrow \text{Inversion} \rightarrow \text{Permutation} \rightarrow T_{n+1}$$

puis :

$$\text{Confirmation Client} \rightarrow \text{Validation Host} \rightarrow T_{n+1} \text{ actif}$$

TorUsNet X introduit ainsi le concept d'un **Tore Privé auto-mutant**, dont l'organisation directionnelle évolue avec son utilisation.

Le parcours ne se contente plus de traverser le tore.

Il participe à sa transformation.

Et chaque transformation prépare la structure directionnelle dans laquelle le prochain parcours devra évoluer.

Chapitre 1 : La structure du Tore Privé

1.1 Structure volumétrique fondamentale

Dans **TorUsNet X**, le Tore Privé conserve l'architecture volumétrique introduite dans TorUsNet 9.0 :

$$100 \times 100 \times 100$$

Il contient donc :

$$1\,000\,000 \text{ cases}$$

organisées sous la forme de :

$$100 \text{ canevas de } 100 \times 100$$

Chaque canevas contient :

$$100 \times 100 = 10\,000 \text{ cases}$$

et constitue une **couche directionnelle** du Tore Privé. Cette organisation en 100 canevas de 100×100 était déjà au cœur de la structure volumétrique de TorUsNet 9.0.

Les couches sont disposées successivement en profondeur :

$$C_1, C_2, C_3, \dots, C_{100}$$

Cette représentation permet au Tore Privé d'être lu selon plusieurs axes différents en fonction du type d'opération demandé.

Le même tore peut ainsi être lu :

- horizontalement;
- verticalement;
- circulairement;
- en profondeur.

La géométrie de lecture dépend du déplacement exécuté.

1.2 Les 100 canevas comme couches successives

Les 100 canevas forment les 100 niveaux successifs du Tore Privé.

Chaque couche contient :

10 000 cases

et l'ensemble constitue :

$100 \times 10\,000 = 1\,000\,000$ cases

La structure peut être représentée conceptuellement comme :

$$C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow \dots \rightarrow C_{100}$$

Cette succession ne crée cependant pas une limite physique définitive.

Le Tore Privé est une structure fermée.

Lorsqu'une lecture dépasse une limite, elle reboucle selon la géométrie correspondant au déplacement exécuté.

1.3 Lecture circulaire lors d'un saut exponentiel

Lorsqu'un **saut exponentiel** est suivi d'une opération mathématique :

– + / *

le canevas est interprété comme **un cercle**.

La lecture commence à partir de la case déterminée comme point de départ.

Elle progresse ensuite dans la direction imposée par le parcours.

Si la lecture atteint la dernière case du canevas et doit continuer, elle ne s'arrête pas.

Elle revient automatiquement au début :

$$1 \rightarrow 2 \rightarrow 3 \rightarrow \dots \rightarrow 10\,000 \rightarrow 1$$

Le canevas devient donc une structure de lecture circulaire.

Il n'existe plus, du point de vue du parcours, de début et de fin absolus.

Il existe uniquement :

un point de départ et une direction de lecture

Cette logique circulaire lors d'un saut exponentiel accompagné d'un opérateur était déjà présente dans la mécanique de lecture de TorUsNet 9.0.

1.4 Déplacement cardinal et comportement torique

Lorsqu'un déplacement cardinal est exécuté, le canevas ne se comporte plus simplement comme une suite de cases.

Il se comporte comme **un tore**.

Les limites opposées du canevas deviennent connectées.

Ainsi :

haut → bas

et :

bas → haut

De la même manière :

gauche → droite

et :

droite → gauche

Un déplacement ne rencontre donc jamais une frontière définitive.

Cette continuité torique des déplacements dans les différents axes faisait déjà partie de la structure volumétrique de TorUsNet 9.0.

1.5 Lecture verticale en colonnes

Lorsqu'un déplacement demande une progression vers le haut ou vers le bas, le mode de lecture change.

Au lieu de lire le canevas horizontalement, le Tore Privé lit les cases **en colonne dans la direction du déplacement**.

Pour un déplacement vers le bas :

haut ↓↓↓ bas

La lecture descend dans la colonne.

Lorsqu'elle dépasse la dernière position :

bas → haut

et la lecture continue.

Pour un déplacement vers le haut, le principe est inversé :

bas ↑↑↑ haut

puis :

haut → bas

La colonne constitue donc elle aussi **une boucle circulaire**.

La direction du déplacement détermine le sens de lecture.

1.6 Lecture horizontale

Le même principe s'applique aux déplacements vers la gauche et vers la droite.

Vers la droite :

1 → 2 → 3 → ... → 100 → 1

Vers la gauche :

$$100 \rightarrow 99 \rightarrow 98 \rightarrow \dots \rightarrow 1 \rightarrow 100$$

Ainsi, chaque ligne peut être considérée comme une boucle directionnelle.

Chaque colonne peut également être considérée comme une boucle directionnelle.

Le canevas possède donc simultanément une continuité horizontale et verticale.

1.7 L'intensité et la lecture en profondeur

TorUsNet X conserve également le principe d'**intensité directionnelle**.

Lorsqu'une intensité est rencontrée, la lecture ne se limite plus au canevas courant.

Elle progresse **en profondeur dans les différentes couches du Tore Privé**.

Les 100 canevas deviennent alors une structure verticale :

$$C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow \dots \rightarrow C_{100}$$

L'intensité détermine la profondeur atteinte par la lecture.

1.8 Exemple d'intensité positive

Considérons :

$$I = 150$$

Le Tore Privé possède seulement 100 couches.

La lecture traverse donc les 100 premières couches, reboucle, puis continue jusqu'à la couche 50.

La couche finale est obtenue par :

$$150 \ 100 = 50$$

Ainsi :

$$I = 150 \Rightarrow C_{50}$$

Le principe est donc circulaire en profondeur :

$$C_{100} \rightarrow C_1$$

exactement comme les lignes et les colonnes rebouclent à l'intérieur d'un canevas.

1.9 Inversion des bits pendant la traversée

Lorsqu'une intensité provoque une traversée en profondeur, les cases traversées subissent également une transformation.

Chaque bit rencontré est inversé :

$$1 \rightarrow 0$$

et :

$$0 \rightarrow 1$$

Ainsi, la lecture n'est pas passive.

Le passage du parcours modifie l'état des cases qu'il traverse.

Dans TorUsNet 9.0, l'inversion des bits traversés faisait déjà partie de la logique de lecture des couches.

Dans TorUsNet X, ce mécanisme devient l'une des composantes d'un système de mutation plus large, puisque les chapitres suivants pourront également introduire **la permutation structurelle des cases déterminée par leurs ID**.

La transformation générale devient donc :

$$\text{lecture} \rightarrow \text{traversee} \rightarrow \text{inversion des bits}$$

avant l'application des mécanismes supplémentaires de TorUsNet X.

1.10 Intensité négative

TorUsNet X introduit également une distinction selon le signe de l'intensité.

Une intensité positive produit une lecture orientée depuis la surface vers la profondeur :

$$C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_{100}$$

Une intensité négative produit la lecture opposée.

La lecture commence alors depuis les couches profondes et remonte vers la surface :

$$C_{100} \rightarrow C_{99} \rightarrow C_{98} \rightarrow \dots \rightarrow C_1$$

Le signe de l'intensité devient donc un **opérateur de direction en profondeur**.

On peut représenter cette règle par :

$$I > 0 \Rightarrow \text{surface} \rightarrow \text{profondeur}$$

et :

$$I < 0 \Rightarrow \text{profondeur} \rightarrow \text{surface}$$

La valeur absolue de l'intensité détermine la distance parcourue :

$$|I|$$

tandis que son signe :

$$\text{sgn}(I)$$

détermine le sens de lecture.

1.11 Quatre modes fondamentaux de lecture

La structure du Tore Privé de TorUsNet X peut ainsi être comprise à travers quatre mécanismes.

Lecture circulaire

Déclenchée par un saut exponentiel suivi d'une opération mathématique :

saut exponentiel + $(-, +, /, *)$ → lecture circulaire

Lecture torique

Déclenchée par un déplacement cardinal :

déplacement cardinal → continuité torique

Lecture en colonne

Déclenchée lorsque le déplacement se fait verticalement :

$\uparrow/\downarrow \rightarrow$ lecture circulaire de la colonne

Lecture en profondeur

Déclenchée par une intensité :

$I \rightarrow$ lecture des couches

avec :

$I > 0 \Rightarrow$ surface $\rightarrow$ profondeur

et :

$I < 0 \Rightarrow$ profondeur $\rightarrow$ surface

1.12 Principe général du Tore Privé

Le Tore Privé de TorUsNet X constitue donc une structure volumétrique fermée dans laquelle **chaque axe possède sa propre continuité**.

Il n'existe pas de bord définitif.

Horizontalement, la lecture reboucle.

Verticalement, la lecture reboucle.

En profondeur, la lecture reboucle.

Un parcours peut donc évoluer dans :

X

Y

et :

Z

sans quitter la structure volumétrique.

Le principe général peut finalement être résumé par :

$100 \times 100 \times 100 \rightarrow$ lecture circulaire + lecture torique + lecture verticale + lecture en profondeur

L'intensité détermine la profondeur et son sens.

Les déplacements cardinaux déterminent l'orientation locale.

Les sauts exponentiels déterminent les parcours circulaires.

Les cases traversées voient leurs bits s'inverser.

TorUsNet X conserve ainsi le Tore Privé volumétrique de la génération précédente tout en préparant sa transformation fondamentale :

Le Tore Privé n'est plus seulement une structure parcourue. Chaque parcours peut modifier l'état dans lequel le parcours suivant trouvera cette structure.

Chapitre 2 : La logique des cases

2.1 Structure fondamentale d'une case

Dans **TorUsNet X**, chaque case du Tore Privé constitue une unité directionnelle complète.

Chaque case contient :

- un **ID de 16 bits**;
- un **bit normal**;
- un **bit fantôme**;
- **quatre cases de déplacement possibles**;
- un **saut exponentiel** associé à une opération mathématique;
- une **direction cardinale**;
- une **intensité privée**, positive ou négative.

Une case peut donc être représentée conceptuellement par :

$$C_i = (I, D_i, B_i, \bar{B}_i, D_{i1}, D_{i2}, D_{i3}, D_{i4}, S_i, O_i, D_i, I_i)$$

La case n'est donc pas seulement une position dans une grille. Elle contient les informations permettant au parcours de déterminer **quoi lire, comment se déplacer et comment transformer progressivement le Tore Privé**.

2.2 Le bit normal et le bit fantôme

Chaque case contient un bit normal :

$$B_i \in \{0, 1\}$$

et un bit fantôme qui représente toujours son contraire :

$$\bar{B}_i = NOT(B_i)$$

Ainsi :

$$B_i = 0 \Rightarrow \bar{B}_i = 1$$

et :

$$B_i = 1 \Rightarrow \bar{B}_i = 0$$

Les deux états forment donc une paire complémentaire.

Lorsqu'une inversion est déclenchée :

$$0 \leftrightarrow 1$$

le bit fantôme reste automatiquement l'opposé du nouvel état normal.

2.3 Les quatre possibilités de déplacement

Chaque case possède quatre possibilités de déplacement.

Ces déplacements permettent au parcours de déterminer la prochaine case pouvant être atteinte depuis la case courante.

Le choix du déplacement dépend de la logique directionnelle appliquée au parcours.

La case actuelle devient ainsi une transition entre plusieurs positions potentielles :

$$C_i \rightarrow \{C_a, C_b, C_c, C_d\}$$

La case ne représente donc pas uniquement une information à lire.

Elle représente également **un ensemble de chemins possibles**.

2.4 Le saut exponentiel et l'opération mathématique

Chaque case possède également un saut exponentiel auquel est associée une opération parmi :

$$+ \quad - \quad / \quad *$$

Lorsqu'un saut exponentiel suivi d'une de ces opérations est exécuté, la lecture utilise la logique circulaire définie au chapitre précédent.

Le parcours peut donc avancer au-delà de la limite apparente du canevas et revenir au début.

Le saut exponentiel constitue ainsi **un mécanisme de déplacement interne de la case**.

2.5 La direction cardinale

Une case possède également une direction cardinale.

Cette direction détermine comment le parcours doit interpréter le déplacement dans le canevas.

Selon la direction, le parcours peut progresser :



La géométrie torique impose alors la continuité décrite au chapitre 1.

Si le parcours dépasse une limite du canevas, il réapparaît du côté opposé.

2.6 L'intensité privée de la case

Chaque case contient également une intensité propre :

$$I_{\text{privée}}$$

Cette intensité peut être :

$$I_{\text{privée}} > 0$$

ou :

$$I_{\text{privée}} < 0$$

Elle est additionnée à l'intensité transmise par le parcours sur le réseau :

$$I_{\text{finale}} = I_{\text{transmise}} + I_{\text{privée}}$$

L'intensité finale détermine alors la lecture en profondeur du Tore Privé.

Le signe de cette intensité détermine également son orientation :

$$\begin{aligned} I_{\text{finale}} > 0 &\Rightarrow \text{surface} \rightarrow \text{profondeur} \\ I_{\text{finale}} < 0 &\Rightarrow \text{profondeur} \rightarrow \text{surface} \end{aligned}$$

L'intensité devient donc une propriété dynamique construite à partir d'une partie transmise et d'une partie appartenant au Tore Privé.

2.7 Lecture alternée des ID

TorUsNet X introduit une nouvelle fonction fondamentale : **la lecture alternée des ID des cases traversées.**

Chaque ID contient :

16 bits

Le sens de lecture dépend de la position de la case dans le parcours.

Le premier ID est lu normalement :

$$ID_A \rightarrow$$

Le deuxième est lu dans le sens inverse :

$$ID_B \leftarrow$$

Le troisième revient à une lecture normale :

$$ID_C \rightarrow$$

Le quatrième est lu en sens inverse :

$$ID_D \leftarrow$$

et ainsi de suite.

La règle générale devient :

$$R(ID_i) = \begin{cases} ID_i \text{ normal,} & i \text{ impair} \\ ID_i \text{ inverse,} & i \text{ pair} \end{cases}$$

Cette alternance se poursuit pendant tout le parcours.

2.8 La parité comme déclencheur de permutation

Après la lecture d'un ID, TorUsNet X vérifie la parité de la valeur obtenue.

Si :

$$R(ID_i) \bmod 2 = 0$$

la valeur est paire et aucune permutation n'est demandée par cette case.

Si :

$$R(ID_i) \bmod 2 = 1$$

la valeur est impaire et la case doit participer au mécanisme de permutation.

La règle devient donc :

$$ID_{lu} \text{ impair} \Rightarrow \text{permutation}$$

La parité ne détermine pas seulement une valeur logique.

Elle devient **un déclencheur structurel du Tore Privé**.

2.9 Exemple : lecture de la case A

Considérons le parcours :

$$A \rightarrow B \rightarrow C \rightarrow D$$

Le parcours commence par la case A.

L'ID de A constitue le premier ID du parcours. Il est donc lu normalement :

$$ID_A \rightarrow$$

Si la valeur obtenue est impaire :

$$R(ID_A) 2 = 1$$

une permutation est programmée entre :

$$\boxed{A \leftrightarrow B}$$

Cependant, cette permutation n'est **pas exécutée immédiatement**.

Le parcours poursuit d'abord sa progression.

Il lit :

$$A \rightarrow B \rightarrow C$$

Lorsque la case C est atteinte, la permutation précédemment programmée peut être appliquée :

$$\boxed{A \leftrightarrow B}$$

Le parcours possède donc toujours suffisamment d'information pour poursuivre son exécution avant que les deux cases précédentes ne changent de position.

2.10 Exemple : lecture de la case B

La case B constitue la deuxième case du parcours.

Son ID est donc lu dans le sens inverse :

$$ID_B \leftarrow$$

Si la valeur inversée est impaire :

$$R(ID_B) 2 = 1$$

une nouvelle permutation est enregistrée :

$$\boxed{B \leftrightarrow C}$$

Cette permutation attend elle aussi la lecture de la case suivante.

Le parcours devient :

$$A \rightarrow B \rightarrow C \rightarrow D$$

Puis, lors de la lecture de D :

$$\boxed{B \leftrightarrow C}$$

est appliqué.

Le principe général devient ainsi :

La parité d'une case détermine la permutation avec la case suivante, mais l'échange n'est exécuté qu'après la lecture de la case subséquente.

2.11 Décalage d'une case avant permutation

Ce décalage constitue une propriété importante de TorUsNet X.

Si la case courante est :

$$C_i$$

et que son ID lu est impair, TorUsNet X programme :

$$C_i \leftrightarrow C_{i+1}$$

mais attend la lecture de :

$$C_{i+2}$$

avant d'exécuter l'échange.

Formellement :

$$\boxed{R(ID_i) \text{ impair} \Rightarrow \text{programmer}(C_i \leftrightarrow C_{i+1})}$$

puis :

$$\boxed{\text{lecture}(C_{i+2}) \Rightarrow \text{exécuter}(C_i \leftrightarrow C_{i+1})}$$

Cela permet au parcours d'avancer avant que la structure qu'il vient de traverser soit réorganisée.

2.12 Inversion des bits pendant le processus

La permutation structurelle ne remplace pas l'inversion des bits.

Les deux mécanismes coexistent.

Lorsqu'une case traversée doit subir une inversion :

$$\boxed{B_i \leftarrow NOT(B_i)}$$

Donc :

$$1 \rightarrow 0$$

et :

$$0 \rightarrow 1$$

Le bit fantôme demeure automatiquement le contraire du bit normal :

$$\boxed{\bar{B}_i = NOT(B_i)}$$

Ainsi, lorsque A et B permutent, leur état logique a également pu être transformé pendant le parcours.

Le processus comporte donc simultanément :

$$\boxed{\text{mutation logique}}$$

et :

2.13 La case entière participe à la permutation

Une permutation ne doit pas être interprétée comme l'échange de deux bits uniquement.

TorUsNet X considère la case comme une structure complète.

Ainsi :

$$A \leftrightarrow B$$

signifie que les structures associées aux positions A et B sont réorganisées selon la fonction de permutation du protocole.

Cela permet au Tore Privé de modifier non seulement ses valeurs internes, mais également **la disposition des chemins disponibles pour les parcours futurs**.

Le parcours suivant rencontre donc un Tore Privé différent de celui qui existait avant le parcours précédent.

2.14 Chaîne complète d'exécution

Pour une succession :

$$A \rightarrow B \rightarrow C \rightarrow D \rightarrow E$$

le fonctionnement peut être résumé ainsi.

Lecture de A

$$ID_A \rightarrow$$

Si impair :

$$\text{programmer } A \leftrightarrow B$$

Lecture de B

$$ID_B \leftarrow$$

Si impair :

programmer $B \leftrightarrow C$

Lecture de C

La permutation précédemment déclenchée par A peut être appliquée :

$$A \leftrightarrow B$$

Lecture de D

La permutation précédemment déclenchée par B peut être appliquée :

$$B \leftrightarrow C$$

Le mécanisme continue ensuite en suivant la même logique pour les cases suivantes.

2.15 Principe général de la logique des cases

Dans TorUsNet X, une case combine donc plusieurs fonctions :

Case = ID + bits + déplacements + saut + opération + direction + intensité

L'ID participe à la sélection des permutations.

Les bits représentent l'état logique normal/fantôme.

Les déplacements définissent les chemins possibles.

Le saut exponentiel détermine une progression circulaire.

La direction cardinale définit l'orientation locale.

L'intensité détermine la lecture en profondeur.

Et la permutation modifie l'organisation du Tore Privé après son passage.

Le cycle général devient :

lecture de la case → lecture alternée de l'ID → test de parité → déplacement → lecture suivante → inver

Ainsi :

Une case de TorUsNet X ne constitue plus seulement un emplacement contenant une information. Elle constitue une unité directionnelle active capable de participer à la transformation du Tore Privé lui-même.

Le parcours laisse donc derrière lui une structure différente de celle qu'il a rencontrée.

$$T_n \xrightarrow{\text{lecture des cases}} T_{n+1}$$

C'est cette transformation progressive qui fait du Tore Privé de **TorUsNet X** une structure réellement **auto-mutante**.

Chapitre 3 : Le parcours transmis par réseau

3.1 Principe général

Dans **TorUsNet X**, le parcours directionnel est transmis sur le réseau afin que le Host et le Client puissent appliquer une même séquence de lecture à leurs Tores Privés respectifs.

Le processus commence par une négociation permettant d'établir la case de départ et la case d'arrivée. Une fois cette négociation terminée, le parcours peut être transmis et exécuté.

La séquence générale est :

Négociation → Validation → Transmission du parcours → Reconstruction → Mutation → Synchronisation

3.2 La clé initiale

Avant la transmission du parcours, le Host génère une clé contenant les informations nécessaires à son initialisation.

Cette clé contient :

- l'ID de la case de départ;

- la **case d'arrivée**.

Elle peut être représentée conceptuellement par :

$$K = ID_{\text{départ}}' + C_{\text{arrivée}}'$$

Cette clé ne contient pas le parcours complet.

Elle permet au Host et au Client d'établir le même point de référence avant l'exécution du parcours.

3.3 Négociation entre le Host et le Client

La négociation précède obligatoirement la transmission du parcours.

Première transmission

Le Host transmet la clé au Client :

$$Host \xrightarrow{K} Client$$

Le Client reçoit et vérifie la clé.

Deuxième transmission

Le Client retourne la même clé au Host :

$$Client \xrightarrow{K} Host$$

Cette opération confirme que le Client a reçu la bonne clé.

Troisième transmission

Le Host retourne une nouvelle fois cette même clé au Client afin de confirmer qu'il reconnaît la clé retournée :

$$Host \xrightarrow{K} Client$$

Confirmation

La réception de cette dernière clé est confirmée.

À partir de ce moment :

Le parcours peut être transmis

Cette négociation établit donc un état commun entre les deux systèmes avant l'exécution du parcours.

3.4 Structure du parcours transmis

Dans TorUsNet X, chaque bit reconstruit à partir du parcours correspond à une instruction transmise sur :

11 bits

Ces 11 bits sont divisés en trois champs :

DD | IIIIIII | B

avec :

- **2 bits** pour sélectionner le déplacement;
- **8 bits** pour l'intensité transmise;
- **1 bit** pour sélectionner le bit normal ou le bit fantôme.

Ainsi :

$2 + 8 + 1 = 11 \text{ bits}$

3.5 Les deux bits de déplacement

Les deux premiers bits indiquent laquelle des quatre possibilités de déplacement contenues dans la case doit être utilisée.

Les quatre codes sont :

00

01

10

11

Chaque combinaison désigne l'une des quatre cases de déplacement disponibles.

On peut représenter cette opération par :

$$DD \rightarrow D_{i,DD}$$

Le parcours n'a donc pas besoin de transmettre directement toutes les propriétés de la prochaine case.

Il sélectionne l'un des quatre déplacements déjà présents dans la structure du Tore Privé.

3.6 Transmission de l'intensité

Après les deux bits de déplacement, le parcours transmet une intensité sur :

8 bits

Cette valeur constitue l'intensité transmise :

I_T

Elle est ensuite combinée avec l'intensité appartenant au déplacement sélectionné.

3.7 Cas particulier de la case de départ

Pour la case de départ, l'intensité transmise est utilisée directement.

Il n'y a donc pas d'addition avec une intensité de déplacement précédente.

Pour la première case :

$$I_F = I_T$$

où :

- I_T = intensité transmise;

- I_F = intensité finale utilisée.

La case de départ initialise ainsi le comportement directionnel du parcours.

3.8 Addition de l'intensité du déplacement

Après la case de départ, le déplacement sélectionné possède une intensité propre.

Cette intensité peut être :

$$I_P > 0$$

ou :

$$I_P < 0$$

Elle est additionnée à l'intensité transmise :

$$\boxed{I_F = I_T + I_P}$$

L'intensité finale I_F devient alors la valeur réellement utilisée pour la lecture en profondeur du Tore Privé.

3.9 Intensité positive ou négative

Le signe de l'intensité finale détermine le sens de lecture en profondeur.

Lorsque :

$$I_F > 0$$

la lecture évolue depuis la surface vers la profondeur :

$$\boxed{C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_{100}}$$

Lorsque :

$$I_F < 0$$

le sens est inversé :

$$C_{100} \rightarrow C_{99} \rightarrow \dots \rightarrow C_1$$

La valeur de l'intensité détermine donc la progression, tandis que son signe détermine le sens de cette progression.

3.10 Mutation de l'intensité du déplacement

L'addition :

$$I_F = I_T + I_P$$

ne sert pas seulement à déterminer la lecture courante.

Elle transforme également le Tore Privé.

Après l'opération, l'ancienne intensité du déplacement sélectionné est remplacée par la somme obtenue :

$$I_P^{nouveau} = I_F$$

Donc :

$$I_P \rightarrow I_T + I_P$$

Le parcours suivant rencontrant ce même déplacement trouvera donc une nouvelle intensité.

Le comportement futur du Tore Privé dépend ainsi des parcours qui ont déjà été exécutés.

3.11 Sélection du bit normal ou fantôme

Le dernier bit de l'instruction de 11 bits détermine le type de bit qui doit être lu.

Il sélectionne :

$$\boxed{\text{bit normal}}$$

ou :

$\hat{\text{bit fantôme}}$

Le bit fantôme demeure toujours le contraire du bit normal :

$$B_F = NOT(B_N)$$

Le dernier bit de l'instruction indique donc quelle manifestation de la case doit participer à la reconstruction.

3.12 Reconstruction d'un bit

Chaque groupe de 11 bits transmis produit donc une opération complète :

$$DD + I_T + B$$

Le Tore Privé :

1. sélectionne l'un des quatre déplacements;
2. récupère l'intensité transmise;
3. combine cette intensité avec l'intensité privée du déplacement, sauf pour la case de départ;
4. détermine la lecture en profondeur;
5. met à jour l'intensité du déplacement;
6. atteint la case correspondante;
7. sélectionne le bit normal ou fantôme;
8. reconstruit le bit correspondant.

Chaque bit reconstruit nécessite donc :

11 bits de parcours

3.13 Les 16 premiers bits du fichier reconstruit

TorUsNet X introduit une fonction supplémentaire pour les **16 premiers bits du fichier reconstruit**.

Ces bits appartiennent normalement au fichier.

Ils ne constituent pas un bloc supplémentaire.

Ils ne sont pas retirés du fichier après leur lecture.

Ils possèdent simplement **une deuxième fonction simultanée**.

Si le fichier reconstruit commence par :

$$F = b_1 b_2 b_3 \dots b_{16} b_{17} \dots b_n$$

alors :

$$b_1 b_2 b_3 \dots b_{16}$$

constituent à la fois :

1. les 16 premiers bits du fichier;
2. le nouvel ID de la case de départ.

Ainsi :

$$\boxed{ID_{\text{depart}}^{n+1} = b_1 b_2 b_3 \dots b_{16}}$$

3.14 Double fonction des 16 premiers bits

La règle fondamentale est donc :

$$\boxed{B_{1 \rightarrow 16} \rightarrow \begin{cases} \text{début du fichier reconstruit} \\ \text{nouvel ID de la case de départ} \end{cases}}$$

Le fichier demeure intégralement :

$$\boxed{F = b_1 b_2 \dots b_{16} b_{17} \dots b_n}$$

Aucun bit n'est supprimé.

La mutation de l'ID utilise simplement une information qui vient déjà d'être reconstruite.

Le fichier participe donc directement à l'évolution du Tore Privé.

3.15 Mutation de l'ID de départ

Avant le parcours, la case de départ possède :

$$ID_{\text{depart}}^n$$

Après reconstruction des 16 premiers bits :

$$ID_{\text{depart}}^{n+1} = b_1 b_2 \dots b_{16}$$

La mutation devient :

$$\boxed{ID_{\text{depart}}^n \rightarrow ID_{\text{depart}}^{n+1}}$$

Le nouvel ID appartient au **nouvel état du Tore Privé**.

Il ne remplace donc pas nécessairement immédiatement l'ID de l'état actuellement actif.

Il est préparé pour :

$$T_{n+1}$$

3.16 Interaction avec la logique de permutation

Pendant que le fichier est reconstruit, les autres mécanismes de TorUsNet X continuent également à fonctionner.

Les ID rencontrés sont lus alternativement :

$$\rightarrow, \leftarrow, \rightarrow, \leftarrow, \dots$$

Les valeurs impaires peuvent programmer des permutations entre cases.

Les bits traversés peuvent changer d'état :

$$0 \leftrightarrow 1$$

Les intensités évoluent :

$$I_P^{nouveau} = I_T + I_P$$

et les cases programmées pour permutation changent de position selon les règles définies dans le chapitre précédent.

Le même parcours produit donc simultanément plusieurs transformations.

3.17 Deux résultats du parcours

Le parcours transmis possède ainsi deux résultats fondamentaux.

Reconstruction

$$\boxed{P \rightarrow F}$$

Le parcours produit le fichier reconstruit.

Évolution du Tore Privé

$$\boxed{T_n \xrightarrow{P} T_{n+1}}$$

Le parcours produit également la prochaine configuration du tore.

Cette nouvelle configuration comprend notamment :

- le nouvel ID de la case de départ;
- les nouvelles intensités;
- les nouveaux états des bits;
- les permutations de cases;
- la nouvelle organisation directionnelle résultante.

Ainsi :

un parcours → un fichier + un nouvel état du tore

3.18 Synchronisation du nouvel état

Après l'exécution du parcours, le Client possède le nouvel état :

$$T_C^{n+1}$$

Le Host conserve temporairement :

$$T_H^n$$

et prépare :

$$T_H^{n+1}$$

Le Host peut donc temporairement conserver :

$$T_H^n + T_H^{n+1}$$

Le Client confirme ensuite que le parcours a été appliqué.

Après validation, le Host peut effectuer le basculement :

$$T_H^n \rightarrow T_H^{n+1}$$

Le nouvel état devient alors actif.

L'ID produit par les 16 premiers bits devient actif avec lui :

$$\boxed{ID_{\text{depart}}^n \rightarrow ID_{\text{depart}}^{n+1}}$$

Host et Client poursuivent ensuite leur fonctionnement à partir du nouvel état synchronisé.

3.19 Structure complète d'une instruction réseau

Une instruction du parcours conserve donc la structure :

$$\boxed{DD \mid IIIIII \mid B}$$

soit :

$$\boxed{2 + 8 + 1 = 11 \text{ bits}}$$

Les fonctions correspondantes sont :

$$\boxed{DD \rightarrow \text{déplacement}}$$

$$\boxed{IIIIII \rightarrow I_T}$$

$$\boxed{B \rightarrow \text{normal/fantome}^\wedge}$$

Puis :

$$\boxed{I_F = I_T + I_P}$$

sauf pour la case initiale, pour laquelle :

$$\boxed{I_F = I_T}$$

Après l'opération :

$$\boxed{I_P^{\text{nouveau}} = I_F}$$

3.20 Principe général du parcours transmis

Le parcours transmis par réseau devient donc beaucoup plus qu'une simple liste de déplacements.

Il constitue simultanément :

- une instruction de déplacement;

- une instruction d'intensité;
- une instruction de lecture;
- un mécanisme de reconstruction;
- un mécanisme de mutation des intensités;
- un mécanisme de mutation de l'ID;
- un déclencheur des inversions;
- un moteur de permutation des cases;
- un mécanisme produisant le prochain état du Tore Privé.

La chaîne générale devient :

Négociation → Parcours réseau → Déplacement → Intensité → Lecture normal/fantôme → Reconstruction

Et l'une des caractéristiques particulières de TorUsNet X devient :

16 premiers bits du fichier = contenu + nouvel ID

Le parcours reconstruit ainsi le fichier tout en participant à la transformation de la structure qui vient de le reconstruire.

Dans TorUsNet X, le parcours transmis produit simultanément de l'information et un nouvel état directionnel du Tore Privé.

Conclusion générale

TorUsNet X représente une nouvelle étape dans l'évolution de l'architecture directionnelle. Cette version conserve les fondations établies dans TorUsNet 9.0 tout en introduisant une évolution majeure : le Tore Privé n'est plus seulement une structure dans laquelle des états internes changent. **La structure elle-même évolue au cours de son utilisation.**

Le Tore Privé demeure fondé sur une architecture volumétrique :

$100 \times 100 \times 100$

soit :

1 000 000 cases

organisées en cent canevas successifs de 100×100 .

Cette structure peut être parcourue horizontalement, verticalement, circulairement et en profondeur. Les déplacements cardinaux utilisent la continuité torique des canevas, tandis que les sauts exponentiels associés aux opérations $+$, $-$, $/$ et $*$ permettent une lecture circulaire depuis le point de départ.

L'intensité introduit une troisième direction de lecture : **la profondeur**.

Une intensité positive produit une progression depuis la surface vers les couches profondes :

$$C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_{100}$$

alors qu'une intensité négative inverse cette orientation :

$$C_{100} \rightarrow C_{99} \rightarrow \dots \rightarrow C_1$$

Une case devenue structure directionnelle active

Dans TorUsNet X, chaque case constitue une unité complète comprenant notamment :

- un ID de 16 bits;
- un bit normal;
- un bit fantôme toujours opposé au bit normal;
- quatre possibilités de déplacement;
- des sauts exponentiels associés à des opérations;
- une direction cardinale;
- une intensité positive ou négative.

La case n'est donc plus simplement un emplacement contenant une valeur.

Elle participe directement à la progression et à la transformation du Tore Privé.

Les ID sont lus alternativement selon leur position dans le parcours :

$$\rightarrow, \leftarrow, \rightarrow, \leftarrow, \dots$$

Lorsqu'un ID ainsi interprété donne une valeur impaire, la case correspondante déclenche une permutation avec la case suivante.

Cette permutation est différée afin que le parcours puisse d'abord atteindre la case subséquente.

La structure peut ainsi évoluer selon une logique du type :

$$A \rightarrow B \rightarrow C$$

puis :

$$A \leftrightarrow B$$

Le parcours modifie donc progressivement la disposition des structures qu'il vient de traverser.

Le parcours transmis

La communication entre le Host et le Client commence par la négociation de la case de départ et de la case d'arrivée.

Après validation, le parcours est transmis sous la forme d'instructions de :

11 bits

pour chaque bit reconstruit :

2 bits de déplacement + 8 bits d'intensité + 1 bit normal/fantôme

soit :

$DD \mid IIIIII \mid B$

Les deux premiers bits sélectionnent l'une des quatre possibilités de déplacement.

Les huit bits suivants transmettent l'intensité.

Le dernier bit détermine si la reconstruction doit utiliser l'état normal ou fantôme de la case.

À partir du deuxième déplacement, l'intensité transmise est combinée avec l'intensité privée du déplacement choisi :

$$I_F = I_T + I_P$$

Puis l'intensité privée du déplacement évolue elle-même :

$$I_P^{nouveau} = I_F$$

Ainsi, chaque parcours contribue à modifier les conditions dans lesquelles le prochain parcours sera exécuté.

Le fichier participe à la mutation du tore

Les 16 premiers bits du fichier reconstruit possèdent une double fonction.

Ils demeurent intégralement présents dans le fichier :

$$F = b_1 b_2 \dots b_{16} b_{17} \dots b_n$$

mais permettent également de produire le nouvel ID de la case de départ :

$$ID_{\text{depart}}^{n+1} = b_1 b_2 \dots b_{16}$$

Ainsi :

$$16 \text{ premiers bits} \rightarrow \begin{cases} \text{contenu du fichier} \\ \text{nouvel ID de départ} \end{cases}$$

Le fichier reconstruit contribue donc directement à la transformation de la structure qui vient de le reconstruire.

Un Tore Privé auto-mutant

TorUsNet X conduit finalement à une relation fondamentale :

$$T_n \xrightarrow{\text{parcours}} T_{n+1}$$

Le nouvel état peut intégrer :

- l'inversion des bits traversés;
- la modification des intensités;
- la mutation de l'ID de départ;
- les permutations des cases;
- la réorganisation des propriétés directionnelles associées.

Le parcours sert donc simultanément à :

reconstruire + transformer

Le Host peut conserver temporairement l'ancien état et le nouvel état du tore. Lorsque le Client confirme l'application correcte du parcours, le nouvel état devient l'état actif synchronisé.

La logique générale de TorUsNet X peut donc être résumée par :

Négociation → Parcours → Lecture → Reconstruction → Inversion → Mutation → Permutation → T_{n+1}

Le principe central de **TorUsNet X** peut finalement être exprimé ainsi :

Le parcours ne traverse plus simplement le Tore Privé. Il participe à la création de sa configuration suivante. Le fichier est reconstruit pendant que les bits, les intensités, les ID et l'organisation directionnelle du tore évoluent. Chaque parcours laisse donc derrière lui un nouvel état destiné au parcours suivant.

Complément indispensable : TorUsNet 9.0

TorUsNet X doit être lu en complément de TorUsNet 9.0.

Le présent document se concentre principalement sur les nouveaux mécanismes introduits par TorUsNet X et n'a pas pour objectif de répéter l'ensemble de l'architecture déjà détaillée dans TorUsNet 9.0, notamment ses fondations volumétriques et les mécanismes antérieurs du protocole.

Pour comprendre l'évolution complète de l'architecture, les deux documents doivent donc être considérés ensemble :

TorUsNet 9.0 + TorUsNet X

TorUsNet 9.0 établit les fondations.

TorUsNet X développe le principe du **Tore Privé auto-mutant**.

Site officiel

Les publications, développements et autres travaux de **L'Architech (Gabriel Breton-Breault)** sont accessibles sur :

<https://Kyzis.org>

TorUsNet X s'inscrit ainsi dans le développement continu des architectures directionnelles publiées sur Kyzis.org.

TorUsNet X : le parcours reconstruit l'information, mais transforme également le chemin qui permettra de reconstruire la suivante.