Minage et Consensus

2022-03-12

# Minage et Consensus

## Présentation

Le mot "exploitation minière" ou "minage" est quelque peu trompeur. En évoquant l'extraction de métaux précieux, il focalise notre attention sur la récompense du minage, le nouveau bitcoin créé dans chaque bloc. Bien que l'exploitation minière soit incitée par cette récompense, l'objectif principal de l'exploitation minière n'est pas la récompense ou la génération de nouvelles pièces. Si vous considérez l'exploitation minière uniquement comme le processus par lequel les pièces sont créées, vous confondez les moyens (incitations) avec le but du processus. Le minage est le mécanisme qui sous-tend la chambre de compensation décentralisée, par laquelle les transactions sont validées et compensées. Le minage est l'invention qui rend le bitcoin spécial, un mécanisme de sécurité décentralisé qui est à la base de l'argent numérique P2P.

L'exploitation minière *sécurise le système Bitcoin* et permet l'émergence d'un *consensus à l'échelle du réseau sans autorité centrale*. La récompense des pièces nouvellement frappées et des frais de transaction est un système d'incitation qui aligne les actions des mineurs sur la sécurité du réseau, tout en mettant en œuvre simultanément l'offre monétaire.

Le but du minage n'est pas la création de nouveaux bitcoins. C'est le système d'incitation. Le minage est le mécanisme par lequel la *sécurité* du bitcoin est *décentralisée*.

Les mineurs valident les nouvelles transactions et les enregistrent dans le grand livre mondial. Un nouveau bloc, contenant les transactions qui se sont produites depuis le dernier bloc, est "miné" toutes les 10 minutes en moyenne, ajoutant ainsi ces transactions à la chaîne de blocs. Les transactions qui font partie d'un bloc et ajoutées à la chaîne de blocs sont considérées comme "confirmées", ce qui permet aux nouveaux propriétaires de bitcoin de dépenser le bitcoin qu'ils ont reçu dans ces transactions.

Les mineurs reçoivent deux types de récompenses en échange de la sécurité fournie par le minage : de nouvelles pièces créées avec chaque nouveau bloc, également appelées *récompense de bloc* ou *récompense de coinbase*, et les frais de transaction de toutes les transactions incluses dans le bloc. Pour gagner cette récompense, les mineurs s'affrontent pour résoudre un problème mathématique difficile basé sur un algorithme de hachage cryptographique. La solution au problème, appelée Proof-of-Work (ou preuve de travail), est incluse dans le nouveau bloc et agit comme preuve que le mineur a déployé des efforts informatiques importants. La compétition pour résoudre l'algorithme de preuve de travail pour gagner la récompense et le droit d'enregistrer les transactions sur la chaîne de blocs est la base du modèle de sécurité de Bitcoin.

Le processus est appelé minage parce que la récompense (nouvelle génération de pièces) est conçue pour simuler des rendements décroissants, tout comme le minage de métaux précieux. La masse monétaire de Bitcoin est créée par l'exploitation minière, de la même manière qu'une banque centrale émet de la nouvelle monnaie en imprimant des billets de banque. La quantité maximale de bitcoins nouvellement créés qu'un mineur peut ajouter à un bloc diminue environ tous les quatre ans (ou précisément tous les 210 000 blocs). Il a commencé à 50 bitcoins par bloc en janvier 2009 et a été réduit de moitié à 25 bitcoins par bloc en novembre 2012. Il a été réduit de moitié à 12,5 bitcoins en juillet 2016 et de nouveau à 6,25 bitcoins en mai 2020. Sur la base de cette formule, les récompenses de minage de bitcoins diminuent de façon exponentielle. jusqu'à environ l'an 2140, lorsque tous les bitcoins (20,99999998 millions) auront été émis. Après 2140, aucun nouveau bitcoin ne sera émis.

Les mineurs de Bitcoin gagnent également des frais sur les transactions. Chaque transaction comprend généralement des frais de transaction, sous la forme d'un surplus de bitcoin entre les entrées et les sorties de la transaction. Le mineur de bitcoin gagnant peut "garder la monnaie" sur les transactions incluses dans le bloc gagnant. Aujourd'hui, les frais représentent 0,5% ou moins des revenus d'un mineur de bitcoin, la grande majorité provenant du bitcoin nouvellement créé. Cependant, à mesure que la récompense de bloc diminue avec le temps et que le nombre de transactions par bloc augmente, une plus grande proportion des revenus de minage de bitcoins proviendra des frais. Progressivement, la récompense minière sera dominée par les frais de transaction, qui constitueront la principale incitation pour les mineurs. Après 2140, la quantité de nouveaux bitcoins dans chaque bloc tombe à zéro et l'extraction de bitcoins ne sera encouragée que par les frais de transaction.

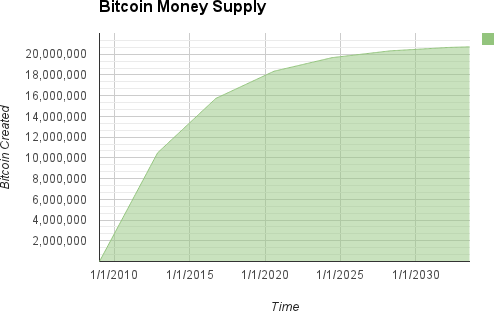
Dans ce chapitre, nous examinerons d'abord l'exploitation minière en tant que mécanisme d'approvisionnement monétaire, puis nous examinerons la fonction la plus importante de l'exploitation minière : le mécanisme de consensus décentralisé qui sous-tend la sécurité du bitcoin.

Pour comprendre l'exploitation minière et le consensus, nous suivrons la transaction d'Alice telle qu'elle est reçue et ajoutée à un bloc par l'équipement minier de Jing. Ensuite, nous suivrons le bloc tel qu'il est extrait, ajouté à la chaîne de blocs et accepté par le réseau Bitcoin à travers le processus de consensus émergent.

### Économie Bitcoin et création de devises

Les Bitcoins sont "frappés" lors de la création de chaque bloc à un taux fixe et décroissant. Chaque bloc, généré en moyenne toutes les 10 minutes, contient des bitcoins entièrement nouveaux, créés à partir de rien. Tous les 210 000 blocs, soit environ tous les quatre ans, le taux d'émission des devises est diminué de 50 %. Pendant les quatre premières années de fonctionnement du réseau, chaque bloc contenait 50 nouveaux bitcoins.

En novembre 2012, le nouveau taux d'émission de bitcoins a été réduit à 25 bitcoins par bloc. En juillet 2016, il a été réduit à 12,5 bitcoins par bloc, et en mai 2020, il a de nouveau été réduit à 6,25 bitcoins par bloc. Le taux de nouvelles pièces diminue ainsi de manière exponentielle sur 32 "halvings" (ou "réductions de moitié") jusqu'au bloc 6 720 000 (miné environ en 2137), lorsqu'il atteint l'unité monétaire minimale de 1 satoshi. Enfin, après 6,93 millions de blocs, en 2140 environ, près de 2 099 999 997 690 000 satoshis, soit près de 21 millions de bitcoins, seront émis. Par la suite, les blocs ne contiendront plus de nouveaux bitcoins et les mineurs seront récompensés uniquement par les frais de transaction. [figure\_title](#bitcoin_money_supply) montre le bitcoin total en circulation au fil du temps, à mesure que l'émission de devises diminue.



Offre de monnaie bitcoin au fil du temps basée sur un taux d'émission géométriquement décroissant

Le nombre maximum de pièces minées est la *limite supérieure* des récompenses minières possibles pour le bitcoin. En pratique, un mineur peut intentionnellement exploiter un bloc en prenant moins que la récompense complète. Ces blocs ont déjà été extraits et d'autres pourraient l'être à l'avenir, ce qui entraînerait une émission totale inférieure de la monnaie.

Dans l'exemple de code dans [example\_title](#max_money), nous calculons le montant total de bitcoin qui sera émis.

# Original block reward for miners was 50 BTC  
start\_block\_reward = 50  
# 210000 is around every 4 years with a 10 minute block interval  
reward\_interval = 210000  
  
def max\_money():  
 # 50 BTC = 50 0000 0000 Satoshis  
 current\_reward = 50 \* 10\*\*8  
 total = 0  
 while current\_reward > 0:  
 total += reward\_interval \* current\_reward  
 current\_reward /= 2  
 return total  
  
print "Total BTC to ever be created:", max\_money(), "Satoshis"

[example\_title](#max_money_run) affiche la sortie produite en exécutant ce script.

$ python max\_money.py  
Total BTC to ever be created: 2100000000000000.0 Satoshis

L'émission finie et décroissante crée une masse monétaire fixe qui résiste à l'inflation. Contrairement à une monnaie fiduciaire, qui peut être imprimée en nombre infini par une banque centrale, le bitcoin ne peut jamais être gonflé par l'impression.

La conséquence la plus importante et la plus débattue d'une émission monétaire fixe et décroissante est que la monnaie a tendance à être intrinsèquement déflationniste. La déflation est le phénomène d'appréciation de la valeur due à une inadéquation entre l'offre et la demande qui fait monter la valeur (et le taux de change) d'une monnaie. Le contraire de l'inflation, la déflation des prix, signifie que l'argent a plus de pouvoir d'achat au fil du temps.

De nombreux économistes soutiennent qu'une économie déflationniste est une catastrophe qu'il faut éviter à tout prix. En effet, en période de déflation rapide, les gens ont tendance à accumuler de l'argent au lieu de le dépenser, en espérant que les prix chuteront. Un tel phénomène s'est déroulé au cours de la « décennie perdue » du Japon, lorsqu'un effondrement complet de la demande a poussé la monnaie dans une spirale déflationniste.

Les experts du Bitcoin affirment que la déflation n'est pas mauvaise en soi. La déflation est plutôt associée à un effondrement de la demande car c'est le seul exemple de déflation que nous ayons à étudier. Dans une monnaie fiduciaire avec possibilité d'impression illimitée, il est très difficile d'entrer dans une spirale déflationniste à moins qu'il n'y ait un effondrement complet de la demande et une réticence à imprimer de l'argent. La déflation du bitcoin n'est pas causée par un effondrement de la demande, mais par une offre limitée de manière prévisible.

L'aspect positif de la déflation, bien sûr, c'est qu'elle est le contraire de l'inflation. L'inflation provoque une dépréciation lente mais inévitable de la monnaie, se traduisant par une forme de taxation cachée qui punit les épargnants afin de renflouer les débiteurs (y compris les plus gros débiteurs, les gouvernements eux-mêmes). Les monnaies sous contrôle gouvernemental souffrent de l'aléa moral d'une émission facile de dette qui peut ensuite être effacée par la dépréciation aux dépens des épargnants.

Il reste à voir si l'aspect déflationniste de la monnaie est un problème lorsqu'il n'est pas entraîné par une rétraction économique rapide, ou un avantage parce que la protection contre l'inflation et l'avilissement l'emporte largement sur les risques de déflation.

## Consensus décentralisé

Dans le chapitre précédent, nous avons examiné la chaîne de blocs, le registre public mondial (liste) de tous transactions, que tout le monde dans le réseau Bitcoin accepte comme enregistrement de propriété faisant autorité.

Mais comment tout le monde dans le réseau peut-il s'entendre sur une seule « vérité » universelle sur qui possède quoi, sans avoir à faire confiance à personne ? Tous les systèmes de paiement traditionnels dépendent d'un modèle de confiance qui a une autorité centrale fournissant un service de chambre de compensation, vérifiant et compensant essentiellement toutes les transactions. Bitcoin n'a pas d'autorité centrale, mais d'une manière ou d'une autre, chaque nœud complet possède une copie complète d'un grand livre public auquel il peut faire confiance en tant qu'enregistrement faisant autorité. La chaîne de blocs n'est pas créée par une autorité centrale, mais est assemblée indépendamment par chaque nœud du réseau. D'une manière ou d'une autre, chaque nœud du réseau, agissant sur les informations transmises via des connexions réseau non sécurisées, peut arriver à la même conclusion et assembler une copie du même registre public que tout le monde. Ce chapitre examine le processus par lequel le réseau Bitcoin parvient à un consensus mondial sans autorité centrale.

La principale invention de Satoshi Nakamoto est le mécanisme décentralisé de *consensus émergent*. Emergent, parce que le consensus n'est pas atteint explicitement - il n'y a pas d'élection ou de moment fixe où le consensus se produit. Au lieu de cela, le consensus est un artefact émergent de l'interaction asynchrone de milliers de nœuds indépendants, qui suivent tous des règles simples. Toutes les propriétés du bitcoin, y compris la monnaie, les transactions, les paiements et le modèle de sécurité qui ne dépend pas de l'autorité centrale ou de la confiance, découlent de cette invention.

Le consensus décentralisé de Bitcoin émerge de l'interaction de quatre processus qui se produisent indépendamment sur les nœuds du réseau :

* Vérification indépendante de chaque transaction, par chaque nœud complet, sur la base d'une liste complète de critères
* Agrégation indépendante de ces transactions dans de nouveaux blocs par des nœuds de minage, associée à un calcul démontré via un algorithme de preuve de travail
* Vérification indépendante des nouveaux blocs par chaque nœud et assemblage dans une chaîne
* Sélection indépendante, par chaque nœud, de la chaîne avec le calcul le plus cumulatif démontré par la preuve de travail

Dans les prochaines sections, nous examinerons ces processus et la manière dont ils interagissent pour créer la propriété émergente du consensus à l'échelle du réseau qui permet à n'importe quel nœud Bitcoin d'assembler sa propre copie du grand livre public mondial faisant autorité.

## Vérification indépendante des transactions

Dans [???](#transactions), nous avons vu comment un logiciel de portefeuille crée des transactions en collectant des UTXO, en fournissant les scripts de déverrouillage appropriés, puis en construisant de nouvelles sorties attribuées à un nouveau propriétaire. La transaction résultante est ensuite envoyée aux nœuds voisins du réseau Bitcoin afin qu'elle puisse être propagée sur l'ensemble du réseau Bitcoin.

Cependant, avant de transmettre les transactions à ses voisins, chaque nœud Bitcoin qui reçoit une transaction vérifiera d'abord la transaction. Cela garantit que seules les transactions valides sont propagées sur le réseau, tandis que les transactions non valides sont rejetées au premier nœud qui les rencontre.

Chaque nœud vérifie chaque transaction par rapport à une longue liste de critères :

* La syntaxe et la structure des données de la transaction doivent être correctes.
* Ni les listes d'entrées ni de sorties ne sont vides.
* La taille de la transaction est inférieure à la taille maximale autorisée pour un bloc excluant les données témoins, comme indiqué dans [*tx\_check.cpp*](https://bit.ly/3t623VL).
* Chaque valeur de sortie, ainsi que le total, doit être dans la plage de valeurs autorisée (moins de 21 millions de pièces, plus que le seuil de *poussière*).
* Aucune des entrées n'a hash=0, N=–1 (les transactions coinbase ne doivent pas être relayées).
* nLocktime est égal à INT\_MAX, ou les valeurs nLocktime et nSequence sont satisfaites selon MedianTimePast.
* La taille de la transaction en octets est supérieure ou égale à 82.
* Le nombre d'opérations de signature (SIGOPS) contenues dans la transaction est inférieur à la limite d'opérations de signature.
* Le script de déverrouillage (scriptSig) ne peut pousser que des nombres sur la pile, et le script de verrouillage (scriptPubkey) doit correspondre aux formes IsStandard (cela rejette les transactions "non standard").
* Une transaction correspondante dans le bassin ou dans un bloc de la branche principale doit exister.
* Pour chaque entrée, si la sortie référencée existe dans une autre transaction du bassin, la transaction doit être rejetée.
* Pour chaque entrée, regardez dans la branche principale et le bassin de transactions pour trouver sa transaction parente. Si la transaction parent est manquante pour une entrée, il s'agira d'une transaction orpheline. Ajouter au bassin de transactions orphelines, si une transaction correspondante n'est pas déjà dans le bassin.
* Pour chaque entrée, si sa transaction parente est une transaction coinbase, elle doit avoir au moins COINBASE\_MATURITY (100) confirmations.
* Pour chaque entrée, la sortie référencée doit exister et ne peut pas déjà être dépensée.
* En utilisant les transactions parentes pour obtenir les valeurs d'entrée, vérifiez que chaque valeur d'entrée, ainsi que la somme, se situent dans la plage de valeurs autorisée (moins de 21 millions de pièces, plus de 0).
* Rejeter si la somme des valeurs d'entrée est inférieure à la somme des valeurs de sortie.
* Rejeter si les frais de transaction seraient trop bas (minRelayTxFee) pour entrer dans un bloc vide.
* Les scripts de déverrouillage pour chaque entrée doivent être validés par rapport aux scripts de verrouillage de sortie correspondants.

Ces conditions peuvent être vues en détail dans les fonctions AcceptToMemoryPool, CheckTransaction et CheckInputs dans Bitcoin Core. Notez que les conditions changent avec le temps, pour faire face à de nouveaux types d'attaques par déni de service ou parfois pour assouplir les règles afin d'inclure davantage de types de transactions.

En vérifiant indépendamment chaque transaction lors de sa réception et avant de la propager, chaque nœud crée un bassin de transactions valides (mais non confirmées) connu sous le nom de *bassin de transaction*, *bassin de mémoire* ou *mempool*.

## Nœuds miniers

Certains des nœuds du réseau Bitcoin sont des nœuds spécialisés appelés *mineurs*. Dans [???](#ch01_intro_what_is_bitcoin) nous avons présenté Jing, un étudiant en génie informatique à Shanghai, en Chine, est un mineur de bitcoin. Jing gagne du bitcoin en exécutant une "plate-forme minière", qui est un système informatique spécialisé conçu pour exploiter le bitcoin. Le matériel minier spécialisé de Jing est connecté à un serveur exécutant un nœud Bitcoin complet. Contrairement à Jing, certains mineurs minent sans nœud complet, comme nous le verrons dans [Groupes miniers](#mining_pools). Comme tous les autres nœuds complets, le nœud de Jing reçoit et propage les transactions non confirmées sur le réseau Bitcoin. Le nœud de Jing, cependant, agrège également ces transactions dans de nouveaux blocs.

Le nœud de Jing est à l'écoute de nouveaux blocs, propagés sur le réseau Bitcoin, comme tous les nœuds. Cependant, l'arrivée d'un nouveau bloc a une signification particulière pour un nœud minier. La compétition entre mineurs se termine effectivement par la propagation d'un nouveau bloc qui fait office d'annonce d'un gagnant. Pour les mineurs, recevoir un nouveau bloc valide signifie que quelqu'un d'autre a gagné la compétition et qu'il (on) a perdu. Cependant, la fin d'un tour d'une compétition est aussi le début du tour suivant. Le nouveau bloc n'est pas seulement un drapeau à damier, marquant la fin de la course; c'est aussi le pistolet de départ dans la course pour le bloc suivant.

## Agréger les transactions en blocs

Après avoir validé les transactions, un nœud Bitcoin les ajoutera au *bassin de mémoire*, ou *bassin de transaction*, où les transactions attendent jusqu'à ce qu'elles puissent être incluses (exploitées) dans un bloc. Le nœud de Jing collecte, valide et relaie les nouvelles transactions comme n'importe quel autre nœud. Contrairement aux autres nœuds, cependant, le nœud de Jing regroupera ensuite ces transactions dans un *bloc candidat*.

Suivons les blocs qui ont été créés à l'époque où Alice achetait une tasse de café au Bob's Cafe (voir [???](#cup_of_coffee)). La transaction d'Alice a été incluse dans le bloc 277 316. Dans le but de démontrer les concepts de ce chapitre, supposons que ce bloc a été miné par le système de minage de Jing et suivons la transaction d'Alice au fur et à mesure qu'elle fait partie de ce nouveau bloc.

Le nœud minier de Jing maintient une copie locale de la blockchain. Au moment où Alice achète la tasse de café, le nœud de Jing a assemblé une chaîne jusqu'au bloc 277 314. Le nœud de Jing écoute les transactions, essaie d'extraire un nouveau bloc et écoute également les blocs découverts par d'autres nœuds. Comme le nœud de Jing est en train d'exploiter, il reçoit le bloc 277 315 via le réseau Bitcoin. L'arrivée de ce bloc signifie la fin de la compétition pour le bloc 277,315 et le début de la compétition pour créer le bloc 277,316.

Au cours des 10 minutes précédentes, alors que le nœud de Jing cherchait une solution au bloc 277 315, il collectait également des transactions en vue du bloc suivant. À ce jour, il a collecté quelques centaines de transactions dans le bassin de mémoire. Lors de la réception du bloc 277,315 et de sa validation, le nœud de Jing le comparera également à toutes les transactions du pool de mémoire et supprimera celles qui étaient incluses dans le bloc 277,315. Quelles que soient les transactions restantes dans le bassin de mémoire, elles ne sont pas confirmées et attendent d'être enregistrées dans un nouveau bloc.

Le nœud de Jing construit immédiatement un nouveau bloc vide, un candidat pour le bloc 277 316. Ce bloc est appelé *bloc candidat* car ce n'est pas encore un bloc valide, car il ne contient pas de preuve de travail valide. Le bloc ne devient valide que si le mineur réussit à trouver une solution à l'algorithme de preuve de travail.

Lorsque le nœud de Jing regroupe toutes les transactions du bassin de mémoire, le nouveau bloc candidat contient 418 transactions avec des frais de transaction totaux de 0,09094928 bitcoin. Vous pouvez voir ce bloc dans la chaîne de blocs à l'aide de l'interface de ligne de commande du client Bitcoin Core, comme indiqué dans [example\_title](#block277316).

$ bitcoin-cli getblockhash 277316  
  
0000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4  
  
$ bitcoin-cli getblock 0000000000000001b6b9a13b095e96db41c4a928b97ef2d9\  
44a9b31b2cc7bdc4

{  
 "hash" : "0000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4",  
 "confirmations" : 35561,  
 "size" : 218629,  
 "height" : 277316,  
 "version" : 2,  
 "merkleroot" : "c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e",  
 "tx" : [  
 "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",  
 "b268b45c59b39d759614757718b9918caf0ba9d97c56f3b91956ff877c503fbe",  
  
 ... 417 more transactions ...  
  
 ],  
 "time" : 1388185914,  
 "nonce" : 924591752,  
 "bits" : "1903a30c",  
 "difficulty" : 1180923195.25802612,  
 "chainwork" : "000000000000000000000000000000000000000000000934695e92aaf53afa1a",  
 "previousblockhash" : "0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569"  
}

### La transaction Coinbase

La première transaction d'un bloc est une transaction spéciale, appelée *transaction coinbase*. Cette transaction est construite par le nœud de Jing et contient sa *récompense* pour l'effort de minage.

Lorsque le bloc 277 316 a été miné, la récompense était de 25 bitcoins par bloc. Depuis, deux périodes de "halving" (ou "réduction de moitié") se sont écoulées. La récompense globale est passée à 12,5 bitcoins en juillet 2016 et à 6,25 bitcoins en mai 2020.

Le nœud de Jing crée la transaction coinbase en tant que paiement sur son propre portefeuille : "Pay Jing's address 25.09094928 bitcoin." Le montant total de la récompense que Jing collecte pour l'extraction d'un bloc est la somme de la récompense coinbase (25 nouveaux bitcoins) et des frais de transaction (0,09094928) de toutes les transactions incluses dans le bloc, comme indiqué dans [example\_title](#generation_tx_example).

$ bitcoin-cli getrawtransaction d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f 1

{  
 "hex" : "01000000010000000000000000000000000000000000000000000000000000000000000000ffffffff0f03443b0403858402062f503253482fffffffff0110c08d9500000000232102aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac00000000",  
 "txid" : "d5ada064c6417ca25c4308bd158c34b77e1c0eca2a73cda16c737e7424afba2f",  
 "version" : 1,  
 "locktime" : 0,  
 "vin" : [  
 {  
 "coinbase" : "03443b0403858402062f503253482f",  
 "sequence" : 4294967295  
 }  
 ],  
 "vout" : [  
 {  
 "value" : 25.09094928,  
 "n" : 0,  
 "scriptPubKey" : {  
 "asm" : "02aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21OP\_CHECKSIG",  
 "hex" : "2102aa970c592640d19de03ff6f329d6fd2eecb023263b9ba5d1b81c29b523da8b21ac",  
 "reqSigs" : 1,  
 "type" : "pubkey",  
 "addresses" : [  
 "1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N"  
 ]  
 }  
 }  
 ]  
}

Contrairement aux transactions régulières, la transaction coinbase ne consomme pas (ne dépense pas) d’UTXO comme entrées. Au lieu de cela, il n'a qu'une seule entrée, appelée *coinbase*, qui crée du bitcoin à partir de rien. La transaction coinbase a une sortie, payable à la propre adresse Bitcoin du mineur. La sortie de la transaction coinbase envoie la valeur de 25,09094928 bitcoin à l'adresse Bitcoin du mineur ; dans ce cas c'est 1MxTkeEP2PmHSMze5tUZ1hAV3YTKu2Gh1N.

### Récompense et frais coinbase

Pour construire la transaction coinbase, le nœud de Jing calcule d'abord le montant total des frais de transaction en ajoutant toutes les entrées et sorties des 418 transactions qui ont été ajoutées au bloc. Les frais sont calculés comme suit :

Frais totaux = Somme (entrées) - Somme (sorties)

Dans le bloc 277 316, le total des frais de transaction est de 0,09094928 bitcoin.

Ensuite, le nœud de Jing calcule la récompense correcte pour le nouveau bloc. La récompense est calculée en fonction de la hauteur du bloc, à partir de 50 bitcoins par bloc et réduite de moitié tous les 210 000 blocs. Parce que ce bloc est à la hauteur 277 316, la récompense correcte est de 25 bitcoins.

Le calcul peut être vu dans la fonction GetBlockSubsidy dans le client Bitcoin Core, comme indiqué dans [example\_title](#getblocksubsidy_source).

CAmount GetBlockSubsidy(int nHeight, const Consensus::Params& consensusParams)  
{  
 int halvings = nHeight / consensusParams.nSubsidyHalvingInterval;  
 // Force la récompense du bloc à zéro lorsque le décalage vers la droite n&#39;est pas défini.  
 if (halvings >= 64)  
 return 0;  
  
 CAmount nSubsidy = 50 \* COIN;  
 // La subvention est réduite de moitié tous les 210 000 blocs, ce qui se produira environ tous les 4 ans.  
 nSubsidy >>= halvings;  
 return nSubsidy;  
}

La subvention initiale est calculée en satoshis en multipliant 50 par la constante COIN (100 000 000 satoshis). Cela fixe la récompense initiale (nSubsidy) à 5 milliards de satoshis.

Ensuite, la fonction calcule le nombre de halvings qui se sont produits en divisant la hauteur actuelle du bloc par l'intervalle de halving (SubsidyHalvingInterval). Dans le cas du bloc 277 316, avec un intervalle de moitié tous les 210 000 blocs, le résultat est 1 moitié.

Le nombre maximum de moitiés autorisé est de 64, donc le code impose une récompense nulle (ne renvoie que les frais) si les 64 moitiés sont dépassées.

Ensuite, la fonction utilise l'opérateur binaire de décalage vers la droite pour diviser la récompense (nSubsidy) par deux pour chaque cycle de réduction de moitié. Dans le cas du bloc 277 316, cela entraînerait un décalage binaire vers la droite de la récompense de 5 milliards de satoshis une fois (une réduction de moitié) et donnerait 2,5 milliards de satoshis, soit 25 bitcoins. L'opérateur de décalage binaire vers la droite est utilisé car il est plus efficace que plusieurs divisions répétées. Pour éviter un bogue potentiel, l'opération de décalage est ignorée après 63 moitiés et la subvention est définie sur 0.

Enfin, la récompense coinbase (nSubsidy) est ajoutée aux frais de transaction (nFees), et la somme est restituée.

Si le nœud minier de Jing écrit la transaction coinbase, qu'est-ce qui empêche Jing de se "récompenser" lui-même 100 ou 1000 bitcoins ? La réponse est qu'une récompense incorrecte aurait pour conséquence que le bloc serait considéré comme invalide par tout le monde, gaspillant l'électricité de Jing utilisée pour la preuve de travail. Jing ne peut dépenser la récompense que si le bloc est accepté par tout le monde.

### Structure de la transaction coinbase

Avec ces calculs, le nœud de Jing construit ensuite la transaction coinbase pour se payer 25,09094928 bitcoins.

Comme vous pouvez le voir dans [example\_title](#generation_tx_example), la transaction coinbase a un format spécial. Au lieu d'une entrée de transaction spécifiant un UTXO précédent à dépenser, il a une entrée "coinbase". Nous avons examiné les entrées de transaction dans [???](#tx_in_structure). Comparons une entrée de transaction régulière avec une entrée de transaction coinbase. [table\_title](#table_8-1) montre la structure d'une entrée de transaction normale, tandis que [table\_title](#table_8-2) montre la structure de l'entrée de la transaction coinbase.

La structure d'une entrée de transaction "normale"

| Taille | Champ | La description |
| --- | --- | --- |
| 32 octets | Hachage de transaction | Pointeur vers la transaction contenant l'UTXO à dépenser |
| 4 octets | Indice de sortie | Le numéro d'index de l'UTXO à dépenser, le premier est 0 |
| 1–9 octets (VarInt) | Taille du script de déverrouillage | Longueur du script de déverrouillage en octets, à suivre |
| variables | Déverrouillage-Script | Un script remplissant les conditions du script de verrouillage UTXO |
| 4 octets | Numéro de séquence | Généralement défini sur 0xFFFFFFFF pour désactiver BIP 125 et BIP 68 |

La structure d'une entrée de transaction coinbase

| Taille | Champ | La description |
| --- | --- | --- |
| 32 octets | Hachage de transaction | Tous les bits sont à zéro : pas une référence de hachage de transaction |
| 4 octets | Indice de sortie | Tous les bits sont des uns : 0xFFFFFFFF |
| 1–9 octets (VarInt) | Taille des données coinbase | Longueur des données coinbase, de 2 à 100 octets |
| variables | Données coinbase | Données arbitraires utilisées pour les nonce supplémentaires et les balises minières. Dans les blocs v2 ; doit commencer par la hauteur du bloc |
| 4 octets | Numéro de séquence | Réglé sur 0xFFFFFFFF |

Dans une transaction coinbase, les deux premiers champs sont définis sur des valeurs qui ne représentent pas une référence UTXO. Au lieu d'un "hachage de transaction", le premier champ est rempli de 32 octets tous mis à zéro. "L'index de sortie" est rempli de 4 octets tous mis à 0xFF (255 décimal). Le "script de déverrouillage" (scriptSig) est remplacé par les données coinbase, un champ de données utilisé par les mineurs, comme nous le verrons ensuite.

### Données coinbase

Les transactions coinbase n'ont pas de champ de script de déverrouillage (aka, scriptSig). Au lieu de cela, ce champ est remplacé par des données coinbase, qui doivent être comprises entre 2 et 100 octets. À l'exception des premiers octets, le reste des données de coinbase peut être utilisé par les mineurs comme ils le souhaitent ; ce sont des données arbitraires.

Dans le bloc de genèse, par exemple, Satoshi Nakamoto a ajouté le texte "The Times 03/Jan/2009 Chancellor on bound of second bailout for banks" dans les données coinbase, l'utilisant comme preuve de la date et pour transmettre un message. Actuellement, les mineurs utilisent les données coinbase pour inclure des valeurs nonce supplémentaires et des chaînes identifiant le bassin de minage.

Les premiers octets de la coinbase étaient auparavant arbitraires, mais ce n'est plus le cas. Conformément à BIP-34, les blocs de version 2 (blocs avec le champ de version défini sur 2) doivent contenir l'index de hauteur de bloc en tant qu'opération "push" (pousser) de script au début du champ coinbase.

Dans le bloc 277 316, nous voyons que le coinbase (voir [example\_title](#generation_tx_example)), qui se trouve dans le script de déverrouillage ou le champ scriptSig de l'entrée de transaction, contient la valeur hexadécimale 03443b0403858402062f503253482f. Décodons cette valeur.

Le premier octet, 03, indique au moteur d'exécution de script de pousser les trois octets suivants sur la pile de script (voir [???](#tx_script_ops_table_pushdata)). Les trois octets suivants, 0x443b04, sont la hauteur de bloc encodée au format little-endian (petit-boutiste, octet le moins significatif en premier). Inversez l'ordre des octets et le résultat est 0x043b44, soit 277 316 en décimal.

Les quelques chiffres hexadécimaux suivants (0385840206) sont utilisés pour coder un *nonce* supplémentaire (voir [La solution nonce (nombre aléatoire) supplémentaire](#extra_nonce)), ou valeur aléatoire, utilisée pour trouver une solution de preuve de travail appropriée.

La partie finale de la donnée coinbase (2f503253482f) est une chaîne de caractères ASCII encodées /P2SH/, ce qui indique que le nœud de minage qui a miné ce bloc prend en charge l'amélioration P2SH définie dans BIP-16. L'introduction de la capacité P2SH a nécessité une signalisation par les mineurs pour approuver le BIP-16 ou le BIP-17. Ceux qui approuvaient la mise en œuvre du BIP-16 devaient inclure la chaîne /P2SH/ dans leurs données coinbase. Ceux qui approuvaient l'implémentation BIP-17 de P2SH devaient inclure la chaîne p2sh/CHV dans leurs données coinbase. Enfin, le BIP-16 a été élu vainqueur, et de nombreux mineurs ont continué à inclure la chaîne /P2SH/ dans leur coinbase pour indiquer qu'ils prennent en charge cette fonctionnalité.

[example\_title](#satoshi_words) utilise la bibliothèque libbitcoin introduite dans [???](#alt_libraries) pour extraire les données coinbase du bloc de genèse, affichant le message de Satoshi. Notez que la bibliothèque libbitcoin contient une copie statique du bloc de genèse, de sorte que l'exemple de code peut récupérer le bloc de genèse directement à partir de la bibliothèque.

/\*  
 Display the genesis block message by Satoshi.  
\*/  
#include <iostream>  
#include <bitcoin/bitcoin.hpp>  
  
int main()  
{  
 // Create genesis block.  
 bc::chain::block block = bc::chain::block::genesis\_mainnet();  
 // Genesis block contains a single coinbase transaction.  
 assert(block.transactions().size() == 1);  
 // Get first transaction in block (coinbase).  
 const bc::chain::transaction& coinbase\_tx = block.transactions()[0];  
 // Coinbase tx has a single input.  
 assert(coinbase\_tx.inputs().size() == 1);  
 const bc::chain::input& coinbase\_input = coinbase\_tx.inputs()[0];  
 // Convert the input script to its raw format.  
 const auto prefix = false;  
 const bc::data\_chunk& raw\_message = coinbase\_input.script().to\_data(prefix);  
 // Convert this to a std::string.  
 std::string message(raw\_message.begin(), raw\_message.end());  
 // Display the genesis block message.  
 std::cout << message << std::endl;  
 return 0;  
}

Nous compilons le code avec le compilateur GNU C++ et exécutons l'exécutable résultant, comme indiqué dans [example\_title](#satoshi_words_run).

# Compiler le code  
$ g++ -o satoshi-words satoshi-words.cpp $(pkg-config --cflags --libs libbitcoin)  
# Lancer l&#39;exécutable  
$ ./satoshi-words  
^D��<GS>^A^DEThe Times 03/Jan/2009 Chancellor on brink of second bailout for banks

## Construction de l'en-tête de bloc

Pour construire l'en-tête de bloc, le nœud d'exploration de données doit remplir six champs, comme indiqué dans [table\_title](#block_header_structure_ch10).

La structure de l'entête du bloc

| Taille | Champ | La description |
| --- | --- | --- |
| 4 octets | Version | Un numéro de version pour suivre les mises à niveau du logiciel/protocole |
| 32 octets | Hachage de bloc précédent | Une référence au hachage du bloc précédent (parent) dans la chaîne |
| 32 octets | Racine de Merkle | Un hachage de la racine de l'arbre de Merkle des transactions de ce bloc |
| 4 octets | Horodatage | Le temps de création approximatif de ce bloc (secondes à partir d'Unix Epoch) |
| 4 octets | Cible | La cible de l'algorithme de preuve de travail pour ce bloc |
| 4 octets | Nonce | Un compteur utilisé pour l'algorithme de preuve de travail |

Au moment où le bloc 277 316 a été miné, le numéro de version décrivant la structure du bloc est la version 2, qui est codée au format petit-boutiste sur 4 octets sous la forme 0x02000000.

Ensuite, le nœud de minage doit ajouter le "Previous Block Hash" (hachage de bloc précédent, également appelé prevhash). C'est le hachage de l'en-tête de bloc du bloc 277,315, le bloc précédent reçu du réseau, que le nœud de Jing a accepté et sélectionné comme *parent* du bloc candidat 277,316. Le hachage d'en-tête de bloc pour le bloc 277 315 est :

0000000000000002a7bbd25a417c0374cc55261021e8a9ca74442b01284f0569

En sélectionnant le bloc *parent* spécifique, indiqué par le champ Hash du bloc précédent dans l'en-tête du bloc candidat, Jing engage sa puissance minière à étendre la chaîne qui se termine dans ce bloc spécifique. En substance, c'est ainsi que Jing "vote" avec son pouvoir minier pour la chaîne valide la plus longue.

L'étape suivante consiste à résumer toutes les transactions avec un arbre de Merkle, afin d'ajouter la racine de Merkle à la en-tête de bloc. La transaction coinbase est répertoriée comme la première transaction du bloc. Ensuite, 418 transactions supplémentaires sont ajoutées après, pour un total de 419 transactions dans le bloc. Comme nous l'avons vu dans le [???](#merkle_trees), il doit y avoir un nombre pair de nœuds "feuilles" dans l'arbre, donc la dernière transaction est dupliquée, créant 420 nœuds, chacun contenant le hachage d'une transaction. Les hachages de transaction sont ensuite combinés, par paires, créant chaque niveau de l'arbre, jusqu'à ce que toutes les transactions soient résumées en un nœud à la "racine" de l'arbre. La racine de l'arborescence de Merkle résume toutes les transactions en une seule valeur de 32 octets, que vous pouvez voir répertoriée comme "racine de Merkle" dans [example\_title](#block277316), et ici :

c91c008c26e50763e9f548bb8b2fc323735f73577effbc55502c51eb4cc7cf2e

Le nœud minier de Jing ajoutera alors un horodatage de 4 octets, codé comme un horodatage "d'époque" Unix, qui est basé sur le nombre de secondes écoulées depuis minuit UTC, le jeudi 1er janvier 1970. L'heure 1388185914 est égale à vendredi , 27 décembre 2013, 23:11:54 UTC.

Le nœud de Jing remplit ensuite la cible, qui définit la preuve de travail requise pour en faire un bloc valide. La cible est stockée dans le bloc en tant que métrique de "bits cibles", qui est un codage d'exposant de mantisse de la cible. Le codage a un exposant de 1 octet, suivi d'une mantisse de 3 octets (coefficient). Dans le bloc 277 316, par exemple, la valeur des bits cibles est 0x1903a30c. La première partie 0x19 est un exposant hexadécimal, tandis que la partie suivante, 0x03a30c, est le coefficient. Le concept de cible est expliqué dans [Reciblage pour ajuster la difficulté](#target) et la représentation des "bits cibles" est expliquée dans [Représentation cible](#target_bits).

Le dernier champ est le nonce, qui est initialisé à zéro.

Une fois tous les autres champs remplis, l'en-tête du bloc est maintenant terminé et le processus d'extraction peut commencer. L'objectif est maintenant de trouver une valeur pour le nonce qui se traduit par un hachage d'en-tête de bloc égal ou inférieur à la cible. Le nœud de minage devra tester des milliards ou des billions de valeurs de nonce avant de trouver un nonce qui réponde à l'exigence.

## Miner le bloc

Maintenant qu'un bloc candidat a été construit par le nœud de Jing, il est temps pour la plate-forme de minage matériel de Jing de "miner" le bloc, pour trouver une solution à l'algorithme de preuve de travail qui rend le bloc valide. Tout au long de ce livre, nous avons étudié les fonctions de hachage cryptographique utilisées dans divers aspects du système Bitcoin. La fonction de hachage SHA256 est la fonction utilisée dans le processus d'extraction de bitcoin.

Dans les termes les plus simples, le minage est le processus de hachage répété de l'en-tête de bloc, en modifiant un paramètre, jusqu'à ce que le hachage résultant corresponde à une cible spécifique. Le résultat de la fonction de hachage ne peut pas être déterminé à l'avance, pas plus qu'un modèle ne peut être créé qui produira une valeur de hachage spécifique. Cette caractéristique des fonctions de hachage signifie que la seule façon de produire un résultat de hachage correspondant à une cible spécifique est d'essayer encore et encore, en modifiant de manière aléatoire l'entrée jusqu'à ce que le résultat de hachage souhaité apparaisse par hasard.

### Algorithme de preuve de travail

Un algorithme de hachage prend une entrée de données de longueur arbitraire et produit un résultat déterministe de longueur fixe, c-à-d une empreinte numérique de l'entrée. Pour toute entrée spécifique, le hachage résultant sera toujours le même et peut être facilement calculé et vérifié par toute personne mettant en œuvre le même algorithme de hachage. La principale caractéristique d'un algorithme de hachage cryptographique est qu'il est impossible de trouver deux entrées différentes qui produisent la même empreinte digitale (appelée *collision*). En corollaire, il est également pratiquement impossible de sélectionner une entrée de manière à produire une empreinte digitale souhaitée, autrement qu'en essayant des entrées aléatoires.

Avec SHA256, la sortie a toujours une longueur de 256 bits, quelle que soit la taille de l'entrée. Dans [example\_title](#sha256_example1), nous utiliserons l'interpréteur Python pour calculer le hachage SHA256 de la phrase "Je suis Satoshi Nakamoto".

$ python

Python 3.7.3  
>>> import hashlib  
>>> hashlib.sha256(b"I am Satoshi Nakamoto").hexdigest()  
'5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e'

[example\_title](#sha256_example1) montre le résultat du calcul du hachage de &quot;Je suis Satoshi Nakamoto&quot; : 5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e. Ce nombre de 256 bits est le résultat de *hachage* ou *prétraitement* de la phrase et dépend de chaque partie de la phrase. L'ajout d'une seule lettre, d'un signe de ponctuation ou de tout autre caractère produira un hachage différent.

Maintenant, si nous changeons la phrase, nous devrions nous attendre à voir des hachages complètement différents. Essayons cela en ajoutant un nombre à la fin de notre phrase, en utilisant le script Python simple dans [example\_title](#sha256_example_generator).

# example of iterating a nonce in a hashing algorithm's input  
  
import hashlib  
   
text = "I am Satoshi Nakamoto"  
  
# iterate nonce from 0 to 19  
for nonce in xrange(20):   
   
 # add the nonce to the end of the text  
 input = text + str(nonce)   
   
 # calculate the SHA-256 hash of the input (text+nonce)  
 hash = hashlib.sha256(input).hexdigest()   
   
 # show the input and hash result  
 print input, '=>', hash

L'exécution de ceci produira les hachages de plusieurs phrases, rendues différentes en ajoutant un nombre à la fin du texte. En incrémentant le nombre, nous pouvons obtenir différents hachages, comme indiqué dans [example\_title](#sha256_example_generator_output).

$ python hash\_example.py

I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...  
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...  
I am Satoshi Nakamoto2 => ea758a8134b115298a1583ffb80ae629...  
I am Satoshi Nakamoto3 => bfa9779618ff072c903d773de30c99bd...  
I am Satoshi Nakamoto4 => bce8564de9a83c18c31944a66bde992f...  
I am Satoshi Nakamoto5 => eb362c3cf3479be0a97a20163589038e...  
I am Satoshi Nakamoto6 => 4a2fd48e3be420d0d28e202360cfbaba...  
I am Satoshi Nakamoto7 => 790b5a1349a5f2b909bf74d0d166b17a...  
I am Satoshi Nakamoto8 => 702c45e5b15aa54b625d68dd947f1597...  
I am Satoshi Nakamoto9 => 7007cf7dd40f5e933cd89fff5b791ff0...  
I am Satoshi Nakamoto10 => c2f38c81992f4614206a21537bd634a...  
I am Satoshi Nakamoto11 => 7045da6ed8a914690f087690e1e8d66...  
I am Satoshi Nakamoto12 => 60f01db30c1a0d4cbce2b4b22e88b9b...  
I am Satoshi Nakamoto13 => 0ebc56d59a34f5082aaef3d66b37a66...  
I am Satoshi Nakamoto14 => 27ead1ca85da66981fd9da01a8c6816...  
I am Satoshi Nakamoto15 => 394809fb809c5f83ce97ab554a2812c...  
I am Satoshi Nakamoto16 => 8fa4992219df33f50834465d3047429...  
I am Satoshi Nakamoto17 => dca9b8b4f8d8e1521fa4eaa46f4f0cd...  
I am Satoshi Nakamoto18 => 9989a401b2a3a318b01e9ca9a22b0f3...  
I am Satoshi Nakamoto19 => cda56022ecb5b67b2bc93a2d764e75f...

Chaque phrase produit un résultat de hachage complètement différent. Ils semblent complètement aléatoires, mais vous pouvez reproduire les résultats exacts de cet exemple sur n'importe quel ordinateur avec Python et voir les mêmes hachages exacts.

Le nombre utilisé comme variable dans un tel scénario est appelé un *nonce* (ou nombre aléatoire). Le nonce est utilisé pour faire varier la sortie d'une fonction cryptographique, dans ce cas pour faire varier l'empreinte SHA256 de la phrase.

Pour relever le défi de cet algorithme, fixons-nous un objectif : trouver une phrase qui produit un hachage hexadécimal commençant par un zéro. Heureusement, ce n'est pas difficile ! [example\_title](#sha256_example_generator_output) montre que la phrase "I am Satoshi Nakamoto13" produit le hash 0ebc56d59a34f5082aaef3d66b37a661696c2b618e62432727216ba9531041a5, ce qui correspond à nos critères. Il a fallu 13 tentatives pour le trouver. En termes de probabilités, si la sortie de la fonction de hachage est uniformément distribuée, nous nous attendrions à trouver un résultat avec un 0 comme préfixe hexadécimal une fois tous les 16 hachages (un sur 16 chiffres hexadécimaux de 0 à F). En termes numériques, cela signifie trouver une valeur de hachage inférieure à 0x100000000000000000000000000000000000000000000000000000000000. Nous appelons ce seuil la *cible* et le but est de trouver un hachage qui est numériquement égal ou inférieur à la cible. Si nous diminuons la cible, la tâche de trouver un hachage inférieur à la cible devient de plus en plus difficile.

Pour donner une analogie simple, imaginez un jeu où les joueurs lancent une paire de dés à plusieurs reprises, en essayant de lancer autant ou moins qu'une cible spécifiée. Au premier tour, l'objectif est de 11. À moins que vous ne lanciez un double-six, vous gagnez. Au tour suivant, l'objectif est de 10. Les joueurs doivent lancer 10 ou moins pour gagner, encore une fois une tâche facile. Disons que quelques tours plus tard, l'objectif est tombé à 5. Maintenant, plus de la moitié des lancers de dés dépasseront l'objectif et seront donc invalides. Il faut exponentiellement plus de lancers de dés pour gagner, plus la cible est basse. Finalement, lorsque l'objectif est de 2 (le minimum possible), un seul lancer sur 36, soit 2 % d'entre eux, produira un résultat gagnant.

Du point de vue d'un observateur qui sait que la cible du jeu de dés est 2, si quelqu'un a réussi à lancer un lancer gagnant, on peut supposer qu'il a tenté, en moyenne, 36 lancers. En d'autres termes, on peut estimer la quantité de travail nécessaire pour réussir à partir de la difficulté imposée par la cible. Lorsque l'algorithme est basé sur une fonction déterministe telle que SHA256, l'entrée elle-même constitue la *preuve* qu'une certaine quantité de *travail* a été effectuée pour produire un résultat égal ou inférieur à la cible. Par conséquent, la *Preuve de travail*.

Même si chaque tentative produit un résultat aléatoire, la probabilité de tout résultat possible peut être calculée à l'avance. Par conséquent, un résultat d'une difficulté spécifiée constitue la preuve d'une quantité de travail spécifique.

Dans [example\_title](#sha256_example_generator_output), le "nonce" gagnant est 13 et ce résultat peut être confirmé par n'importe qui indépendamment. N'importe qui peut ajouter le nombre 13 comme suffixe à la phrase "Je suis Satoshi Nakamoto" et calculer le hachage, en vérifiant qu'il est inférieur à la cible. Le résultat réussi est également une preuve de travail, car cela prouve que nous avons fait le travail pour trouver ce nonce. Alors qu'il ne faut qu'un seul calcul de hachage pour vérifier, il nous a fallu 13 calculs de hachage pour trouver un nonce qui a fonctionné. Si nous avions une cible inférieure (difficulté plus élevée), il faudrait beaucoup plus de calculs de hachage pour trouver un nonce approprié, mais un seul calcul de hachage pour que quiconque puisse le vérifier. De plus, en connaissant la cible, n'importe qui peut estimer la difficulté à l'aide de statistiques et donc savoir combien de travail a été nécessaire pour trouver un tel nonce.

La preuve de travail doit produire un hachage qui est *égal ou inférieur à* la cible. Une cible plus élevée signifie qu'il est moins difficile de trouver un hachage égal ou inférieur à la cible. Une cible inférieure signifie qu'il est plus difficile de trouver un hachage égal ou inférieur à la cible. La cible et la difficulté sont inversement liées.

La preuve de travail de Bitcoin est très similaire au défi présenté dans [example\_title](#sha256_example_generator_output). Le mineur construit un bloc candidat rempli de transactions. Ensuite, le mineur calcule le hachage de l'en-tête de ce bloc et voit s'il est égal ou inférieur à la *cible* actuelle. Si le hachage est supérieur à la cible, le mineur modifiera le nonce (généralement en l'incrémentant simplement de un) et réessayera. À la difficulté actuelle du réseau Bitcoin, les mineurs doivent essayer des quadrillions de fois avant de trouver un nonce qui se traduit par un hachage d'en-tête de bloc suffisamment bas.

Un algorithme de preuve de travail très simplifié est implémenté en Python dans [example\_title](#pow_example1).

#!/usr/bin/env python  
# example of proof-of-work algorithm  
  
import hashlib  
import time  
  
max\_nonce = 2 \*\* 32 # 4 billion  
  
def proof\_of\_work(header, difficulty\_bits):  
   
 # calculate the difficulty target  
 target = 2 \*\* (256-difficulty\_bits)  
   
 for nonce in xrange(max\_nonce):  
 hash\_result = hashlib.sha256(str(header)+str(nonce)).hexdigest()  
   
 # check if this is a valid result, below the target  
 if long(hash\_result, 16) < target:  
 print "Success with nonce %d" % nonce  
 print "Hash is %s" % hash\_result  
 return (hash\_result,nonce)  
   
 print "Failed after %d (max\_nonce) tries" % nonce  
 return nonce  
  
   
if \_\_name\_\_ == '\_\_main\_\_':  
   
 nonce = 0  
 hash\_result = ''  
   
 # difficulty from 0 to 31 bits   
 for difficulty\_bits in xrange(32):  
   
 difficulty = 2 \*\* difficulty\_bits  
 print "Difficulty: %ld (%d bits)" % (difficulty, difficulty\_bits)  
   
 print "Starting search..."  
   
 # checkpoint the current time  
 start\_time = time.time()  
   
 # make a new block which includes the hash from the previous block  
 # we fake a block of transactions - just a string  
 new\_block = 'test block with transactions' + hash\_result   
   
 # find a valid nonce for the new block  
 (hash\_result, nonce) = proof\_of\_work(new\_block, difficulty\_bits)   
   
 # checkpoint how long it took to find a result  
 end\_time = time.time()  
   
 elapsed\_time = end\_time - start\_time  
 print "Elapsed Time: %.4f seconds" % elapsed\_time  
   
 if elapsed\_time > 0:  
   
 # estimate the hashes per second  
 hash\_power = float(long(nonce)/elapsed\_time)  
 print "Hashing Power: %ld hashes per second" % hash\_power

En exécutant ce code, vous pouvez définir la difficulté souhaitée (en bits, combien de bits de tête doivent être nuls) et voir combien de temps il faut à votre ordinateur pour trouver une solution. Dans [example\_title](#pow_example_outputs), vous pouvez voir comment cela fonctionne sur un ordinateur portable moyen.

$ python proof-of-work-example.py\*

Difficulty: 1 (0 bits)  
  
[...]  
  
Difficulty: 8 (3 bits)  
Starting search...  
Success with nonce 9  
Hash is 1c1c105e65b47142f028a8f93ddf3dabb9260491bc64474738133ce5256cb3c1  
Elapsed Time: 0.0004 seconds  
Hashing Power: 25065 hashes per second  
Difficulty: 16 (4 bits)  
Starting search...  
Success with nonce 25  
Hash is 0f7becfd3bcd1a82e06663c97176add89e7cae0268de46f94e7e11bc3863e148  
Elapsed Time: 0.0005 seconds  
Hashing Power: 52507 hashes per second  
Difficulty: 32 (5 bits)  
Starting search...  
Success with nonce 36  
Hash is 029ae6e5004302a120630adcbb808452346ab1cf0b94c5189ba8bac1d47e7903  
Elapsed Time: 0.0006 seconds  
Hashing Power: 58164 hashes per second  
  
[...]  
  
Difficulty: 4194304 (22 bits)  
Starting search...  
Success with nonce 1759164  
Hash is 0000008bb8f0e731f0496b8e530da984e85fb3cd2bd81882fe8ba3610b6cefc3  
Elapsed Time: 13.3201 seconds  
Hashing Power: 132068 hashes per second  
Difficulty: 8388608 (23 bits)  
Starting search...  
Success with nonce 14214729  
Hash is 000001408cf12dbd20fcba6372a223e098d58786c6ff93488a9f74f5df4df0a3  
Elapsed Time: 110.1507 seconds  
Hashing Power: 129048 hashes per second  
Difficulty: 16777216 (24 bits)  
Starting search...  
Success with nonce 24586379  
Hash is 0000002c3d6b370fccd699708d1b7cb4a94388595171366b944d68b2acce8b95  
Elapsed Time: 195.2991 seconds  
Hashing Power: 125890 hashes per second  
  
[...]  
  
Difficulty: 67108864 (26 bits)  
Starting search...  
Success with nonce 84561291  
Hash is 0000001f0ea21e676b6dde5ad429b9d131a9f2b000802ab2f169cbca22b1e21a  
Elapsed Time: 665.0949 seconds  
Hashing Power: 127141 hashes per second

Comme vous pouvez le constater, augmenter la difficulté de 1 bit entraîne un doublement du temps nécessaire pour trouver une solution. Si vous pensez à l'ensemble de l'espace numérique de 256 bits, chaque fois que vous contraignez un bit de plus à zéro, vous réduisez l'espace de recherche de moitié. Dans [example\_title](#pow_example_outputs), il faut 84 millions de tentatives de hachage pour trouver un nonce qui produit un hachage avec 26 bits de tête en zéro. Même à une vitesse de plus de 120 000 hachages par seconde, il faut tout de même 10 minutes sur un portable pour trouver cette solution.

Au moment de la rédaction, le réseau tente de trouver un bloc dont le hachage d'en-tête est égal ou inférieur à :

0000000000000000029AB900000000000000000000000000000000000000000

Comme vous pouvez le voir, il y a beaucoup de zéros au début de cette cible, ce qui signifie que la plage acceptable de hachages est beaucoup plus petite, il est donc plus difficile de trouver un hachage valide. Il faudra en moyenne plus de 1,8 zêta de hachages (mille milliards de milliards de hachages) pour que le réseau découvre le bloc suivant. Cela semble être une tâche impossible, mais heureusement, le réseau apporte 3 exa-hachages par seconde (EH/sec) de puissance de traitement, ce qui permettra de trouver un bloc en environ 10 minutes en moyenne.

### Représentation cible

Dans [example\_title](#block277316), nous avons vu que le bloc contient la cible, dans une notation appelée "bits cible" ou simplement "bits", qui dans le bloc 277 316 a la valeur 0x1903a30c. Cette notation exprime la cible de preuve de travail sous la forme d'un format coefficient/exposant, avec les deux premiers chiffres hexadécimaux pour l'exposant et les six chiffres hexadécimaux suivants pour le coefficient. Dans ce bloc, par conséquent, l'exposant est 0x19 et le coefficient est 0x03a30c.

La formule pour calculer la cible de difficulté à partir de cette représentation est :

cible = coefficient \* 2

(8\*(exposant–3))

En utilisant cette formule et la valeur des bits de difficulté 0x1903a30c, nous obtenons :

cible = 0x03a30c \* 2

0x08\*(0x19-0x03)

=> cible = 0x03a30c \* 2

(0x08\*0x16)

=> cible = 0x03a30c \* 2

0xB0

qui en décimal est :

=> cible = 238 348 \* 2

176

=> cible =

22 829 202 948 393 929 850 749 706 076 701 368 331 072 452 018 388 575 715 328

retour en hexadécimal :

=> cible =

0x0000000000000003A30C000000000000000000000000000000000000000000

Cela signifie qu'un bloc valide pour la hauteur 277 316 est celui qui a un hachage d'en-tête de bloc inférieur à cette cible. En binaire, ce nombre doit avoir plus de 60 bits de tête mis à zéro. Avec ce niveau de difficulté, un seul mineur traitant 1 billion de hachages par seconde (1 terahash par seconde ou 1 TH/sec) ne trouverait une solution qu'une fois tous les 8 496 blocs ou une fois tous les 59 jours, en moyenne.

### Reciblage pour ajuster la difficulté

Comme nous l'avons vu, la cible détermine la difficulté et affecte donc le temps qu'il faut pour trouver une solution à l’algorithme de preuve de travail. Cela conduit à des questions évidentes : pourquoi la difficulté est-elle ajustable, qui l'ajuste et comment ?

Les blocs de Bitcoin sont générés toutes les 10 minutes, en moyenne. C'est le battement de cœur de bitcoin et sous-tend la fréquence d'émission de devises et la rapidité de règlement des transactions. Il doit rester constant non seulement à court terme, mais sur une période de plusieurs décennies. Au cours de cette période, on s'attend à ce que la puissance des ordinateurs continue d'augmenter à un rythme rapide. De plus, le nombre de participants à l'exploitation minière et les ordinateurs qu'ils utilisent changeront également constamment. Pour maintenir le temps de génération de bloc à 10 minutes, la difficulté de minage doit être ajustée pour tenir compte de ces changements. En fait, la cible de preuve de travail est un paramètre dynamique qui est périodiquement ajusté pour atteindre un objectif d'intervalle de bloc de 10 minutes. En termes simples, l'objectif est défini de sorte que la puissance minière actuelle se traduise par un intervalle de bloc de 10 minutes.

Comment, alors, un tel ajustement s'effectue-t-il dans un réseau totalement décentralisé ? Le reciblage se produit automatiquement et sur chaque nœud indépendamment. Tous les 2 016 blocs, tous les nœuds reciblent la preuve de travail. L'équation de reciblage mesure le temps qu'il a fallu pour trouver les 2 016 derniers blocs et le compare au temps prévu de 20 160 minutes (2 016 blocs multipliés par l'intervalle de bloc de 10 minutes souhaité). Le rapport entre la durée réelle et la durée souhaitée est calculé et un ajustement proportionnel (à la hausse ou à la baisse) est apporté à la cible. En termes simples : si le réseau trouve des blocs plus rapidement que toutes les 10 minutes, la difficulté augmente (la cible diminue). Si la découverte de blocs est plus lente que prévu, la difficulté diminue (la cible augmente).

L'équation peut se résumer ainsi :

Nouvel objectif = Ancien objectif \* (Heure réelle des 2016 derniers blocs / 20160 minutes)

[example\_title](#retarget_code) montre le code utilisé dans le client Bitcoin Core.

// Limiter le pas de réglage  
 int64\_t nActualTimespan = pindexLast->GetBlockTime() - nFirstBlockTime;  
 LogPrintf(" nActualTimespan = %d before bounds\n", nActualTimespan);  
 if (nActualTimespan < params.nPowTargetTimespan/4)  
 nActualTimespan = params.nPowTargetTimespan/4;  
 if (nActualTimespan > params.nPowTargetTimespan\*4)  
 nActualTimespan = params.nPowTargetTimespan\*4;  
  
 // Recibler  
 const arith\_uint256 bnPowLimit = UintToArith256(params.powLimit);  
 arith\_uint256 bnNew;  
 arith\_uint256 bnOld;  
 bnNew.SetCompact(pindexLast->nBits);  
 bnOld = bnNew;  
 bnNew \*= nActualTimespan;  
 bnNew /= params.nPowTargetTimespan;  
  
 if (bnNew > bnPowLimit)  
 bnNew = bnPowLimit;

Bien que l'étalonnage cible se produise tous les 2 016 blocs, en raison d'une erreur de un par un dans le client Bitcoin Core d'origine, il est basé sur le temps total des 2 015 blocs précédents (et non 2 016 comme il se doit), ce qui entraîne un biais de reciblage vers une difficulté plus élevée de 0,05 %.

Les paramètres Interval (2 016 blocs) et TargetTimespan (deux semaines en 1 209 600 secondes) sont définis dans *chainparams.cpp*.

Pour éviter une volatilité extrême de la difficulté, l'ajustement de reciblage doit être inférieur à un facteur quatre (4) par cycle. Si l'ajustement cible requis est supérieur à un facteur de quatre, il sera ajusté d'un facteur de 4 et pas plus. Tout autre ajustement sera effectué au cours de la prochaine période de reciblage, car le déséquilibre persistera au cours des 2 016 prochains blocs. Par conséquent, des écarts importants entre la puissance de hachage et la difficulté peuvent nécessiter plusieurs cycles de blocs de 2 016 pour s'équilibrer.

La difficulté d'extraction d'un bloc de bitcoins est d'environ '10 minutes de traitement' pour l'ensemble du réseau, sur la base du temps qu'il a fallu pour extraire les 2 016 blocs précédents, ajusté tous les 2 016 blocs. Ceci est réalisé en abaissant ou en élevant la cible.

Notez que la cible est indépendante du nombre de transactions ou de la valeur des transactions. Cela signifie que la quantité de puissance de hachage et donc d'électricité dépensée pour sécuriser le bitcoin est également entièrement indépendante du nombre de transactions. Bitcoin peut évoluer et parvenir à une adoption plus large et rester sécurisé sans aucune augmentation de la puissance de hachage par rapport au niveau actuel. L'augmentation de la puissance de hachage représente les forces du marché alors que de nouveaux mineurs entrent sur le marché pour concourir pour la récompense. Tant qu'une puissance de hachage suffisante est sous le contrôle des mineurs agissant honnêtement à la recherche de la récompense, cela suffit pour empêcher les attaques de "prise de contrôle" et, par conséquent, il en est suffisant pour sécuriser le bitcoin.

La difficulté du minage est étroitement liée au coût de l'électricité et au taux de change du bitcoin vis-à-vis de la monnaie utilisée pour payer l'électricité. Les systèmes de minage hautes performances sont à peu près aussi efficaces que possible avec la génération actuelle de fabrication de silicium, convertissant l'électricité en calcul de hachage au taux le plus élevé possible. La principale influence sur le marché minier est le prix d'un kilowattheure d'électricité en bitcoin, car cela détermine la rentabilité du minage et donc les incitations à entrer ou à sortir du marché minier.

## Réussir à miner le bloc

Comme nous l'avons vu précédemment, le noeud de Jing a construit un bloc candidat et l'a préparé pour le minage. Jing possède plusieurs plates-formes de minage matériel avec des circuits intégrés spécifiques à l'application, où des centaines de milliers de circuits intégrés exécutent l'algorithme SHA256 en parallèle à des vitesses incroyables. Beaucoup de ces machines spécialisées sont connectées à son nœud minier via USB ou un réseau local. Ensuite, le nœud de minage exécuté sur le bureau de Jing transmet l'en-tête de bloc à son matériel de minage, qui commence à tester des milliards de nonces par seconde. Parce que le nonce n'est que de 32 bits, après avoir épuisé toutes les possibilités de nonce (environ 4 milliards), le matériel de minage modifie l'en-tête de bloc (en ajustant l'espace ou l'horodatage supplémentaire du nonce coinbase) et réinitialise le compteur de nonce, en testant de nouvelles combinaisons.

Près de 11 minutes après avoir commencé à miner le bloc 277 316, l'une des machines de minage de matériel trouve une solution et la renvoie au nœud de minage.

Lorsqu'il est inséré dans l'en-tête de bloc, le nonce 924 591 752 produit un hachage de bloc de :

0000000000000001b6b9a13b095e96db41c4a928b97ef2d944a9b31b2cc7bdc4

qui est inférieur à la cible :

0000000000000003A30C000000000000000000000000000000000000000000

Immédiatement, le nœud minier de Jing transmet le bloc à tous ses pairs. Ils reçoivent, valident puis propagent le nouveau bloc. Au fur et à mesure que le bloc se répand sur le réseau, chaque nœud l'ajoute à sa propre copie de la chaîne de blocs, l'étendant à une nouvelle hauteur de 277 316 blocs. Au fur et à mesure que les nœuds miniers reçoivent et valident le bloc, ils abandonnent leurs efforts pour trouver un bloc à la même hauteur et commencent immédiatement à calculer le bloc suivant dans la chaîne, en utilisant le bloc de Jing comme "parent". En construisant sur le bloc nouvellement découvert de Jing, les autres mineurs "votent" essentiellement avec leur puissance minière et approuvent le bloc de Jing et la chaîne qu'il prolonge.

Dans la section suivante, nous examinerons le processus utilisé par chaque nœud pour valider un bloc et sélectionner la chaîne la plus longue, créant ainsi le consensus qui forme la chaîne de blocs décentralisée.("", startref="jingtentwo")

## Validation d'un nouveau bloc

La troisième étape du mécanisme de consensus de bitcoin est l’indépendante validation de chaque nouveau bloc par chaque nœud du réseau. Au fur et à mesure que le bloc nouvellement résolu se déplace sur le réseau, chaque nœud effectue une série de tests pour le valider avant de le propager à ses pairs. Cela garantit que seuls les blocs valides sont propagés sur le réseau. La validation indépendante garantit également que les mineurs qui agissent honnêtement obtiennent leurs blocs incorporés dans la chaîne de blocs, gagnant ainsi la récompense. Les mineurs qui agissent de manière malhonnête voient leurs blocs rejetés et non seulement perdent la récompense, mais gaspillent également les efforts déployés pour trouver une solution de preuve de travail, encourant ainsi le coût de l'électricité sans compensation.

Lorsqu'un nœud reçoit un nouveau bloc, il valide le bloc en le comparant à une longue liste de critères qui doivent tous être remplis ; sinon, le bloc est rejeté. Ces critères peuvent être vus dans le client Bitcoin Core dans les fonctions CheckBlock et CheckBlockHeader et incluent :

* La structure de données du bloc est syntaxiquement valide
* Le hachage de l'en-tête de bloc est égal ou inférieur à la cible (applique la preuve de travail)
* L'horodatage du bloc est inférieur à deux heures dans le futur (en tenant compte des erreurs de temps)
* La taille du bloc est dans les limites acceptables
* La première transaction (et seulement la première) est une transaction coinbase
* Toutes les transactions dans le bloc sont valides en utilisant la liste de contrôle des transactions décrite dans [Vérification indépendante des transactions](#tx_verification)

La validation indépendante de chaque nouveau bloc par chaque nœud du réseau garantit que les mineurs ne peuvent pas tricher. Dans les sections précédentes, nous avons vu comment les mineurs peuvent écrire une transaction qui leur attribue le nouveau bitcoin créé dans le bloc et réclamer les frais de transaction. Pourquoi les mineurs n'écrivent-ils pas eux-mêmes une transaction pour mille bitcoins au lieu de la bonne récompense ? Parce que chaque nœud valide les blocs selon les mêmes règles. Une transaction coinbase invalide rendrait l'ensemble du bloc invalide, ce qui entraînerait le rejet du bloc et, par conséquent, cette transaction ne ferait jamais partie du grand livre. Les mineurs doivent construire un bloc parfait, basé sur les règles partagées que tous les nœuds suivent, et l'exploiter avec une solution correcte à la preuve de travail. Pour ce faire, ils dépensent beaucoup d'électricité dans l'exploitation minière, et s'ils trichent, toute l'électricité et les efforts sont gaspillés. C'est pourquoi la validation indépendante est un élément clé du consensus décentralisé.

## Assemblage et sélection de chaînes de blocs

La dernière étape du mécanisme de consensus décentralisé de Bitcoin est l'assemblage de blocs en chaînes et la sélection de la chaîne avec le plus de preuves de travail. Une fois qu'un nœud a validé un nouveau bloc, il tentera alors d'assembler une chaîne en connectant le bloc à la chaîne de blocs existante.

Les nœuds maintiennent trois ensembles de blocs : ceux connectés à la chaîne de blocs principale, ceux qui forment des branches de la chaîne de blocs principale (chaînes secondaires) et enfin, les blocs qui n'ont pas de parent connu dans les chaînes connues (orphelins). Les blocs invalides sont rejetés dès que l'un des critères de validation échoue et ne sont donc inclus dans aucune chaîne.

La "chaîne principale" à tout moment est la chaîne de blocs *valide* à laquelle est associée la preuve de travail la plus cumulative. Dans la plupart des cas, il s'agit également de la chaîne contenant le plus de blocs, à moins qu'il n'y ait deux chaînes de longueur égale et que l'une ait plus de preuve de travail. La chaîne principale aura également des branches avec des blocs qui sont des "frères et sœurs" des blocs de la chaîne principale. Ces blocs sont valides mais ne font pas partie de la chaîne principale. Elles sont conservées pour référence future, au cas où l'une de ces chaînes serait étendue au-delà de la chaîne principale en cours. Dans la section suivante ([Fourches de chaîne de blocs](#forks)), nous verrons comment des chaînes secondaires se produisent à la suite d'une extraction quasi simultanée de blocs à la même hauteur.

Lorsqu'un nouveau bloc est reçu, un nœud essaie de l'insérer dans la chaîne de blocs existante. Le nœud examinera le champ "hachage du bloc précédent" du bloc, qui est la référence au parent du bloc. Ensuite, le nœud tentera de trouver ce parent dans la chaîne de blocs existante. La plupart du temps, le parent sera la "pointe" de la chaîne principale, ce qui signifie que ce nouveau bloc étend la chaîne principale. Par exemple, le nouveau bloc 277,316 a une référence au hachage de son bloc parent 277,315. La plupart des nœuds qui reçoivent 277 316 auront déjà le bloc 277 315 comme pointe de leur chaîne principale et relieront donc le nouveau bloc et étendront cette chaîne.

Parfois, comme nous le verrons dans [Fourches de chaîne de blocs](#forks), le nouveau bloc prolonge une chaîne qui n'est pas la chaîne principale. Dans ce cas, le nœud attachera le nouveau bloc à la chaîne secondaire qu'il prolonge, puis comparera le travail de la chaîne secondaire à la chaîne principale. Si la chaîne secondaire a plus de travail cumulé que la chaîne principale, le nœud *reconvergera* sur la chaîne secondaire, ce qui signifie qu'il sélectionnera la chaîne secondaire comme sa nouvelle chaîne principale, faisant de l'ancienne chaîne principale une chaîne secondaire. Si le nœud est un mineur, il construira maintenant un bloc étendant cette nouvelle chaîne plus longue.

Si un bloc valide est reçu et qu'aucun parent n'est trouvé dans les chaînes existantes, ce bloc est considéré comme "orphelin". Les blocs orphelins sont enregistrés dans le bassin de blocs orphelins où ils resteront jusqu'à ce que leur parent soit reçu. Une fois que le parent est reçu et lié aux chaînes existantes, l'orphelin peut être retiré du bassin d'orphelins et lié au parent, ce qui en fait une partie d'une chaîne. Les blocs orphelins se produisent généralement lorsque deux blocs extraits dans un court laps de temps sont reçus dans l'ordre inverse (enfant avant parent).

En sélectionnant la chaîne valide de plus grand travail cumulé, tous les nœuds finissent par parvenir à un consensus à l'échelle du réseau. Les divergences temporaires entre les chaînes sont finalement résolues au fur et à mesure que du travail est ajouté, étendant l'une des chaînes possibles. Les nœuds de minage « votent » avec leur puissance de minage en choisissant quelle chaîne étendre en minant le bloc suivant. Lorsqu'ils exploitent un nouveau bloc et étendent la chaîne, le nouveau bloc lui-même représente leur vote.

Dans la section suivante, nous verrons comment les divergences entre les chaînes concurrentes (les fourches) sont résolues par la sélection indépendante de la chaîne de plus grand travail cumulatif.

### Fourches de chaîne de blocs

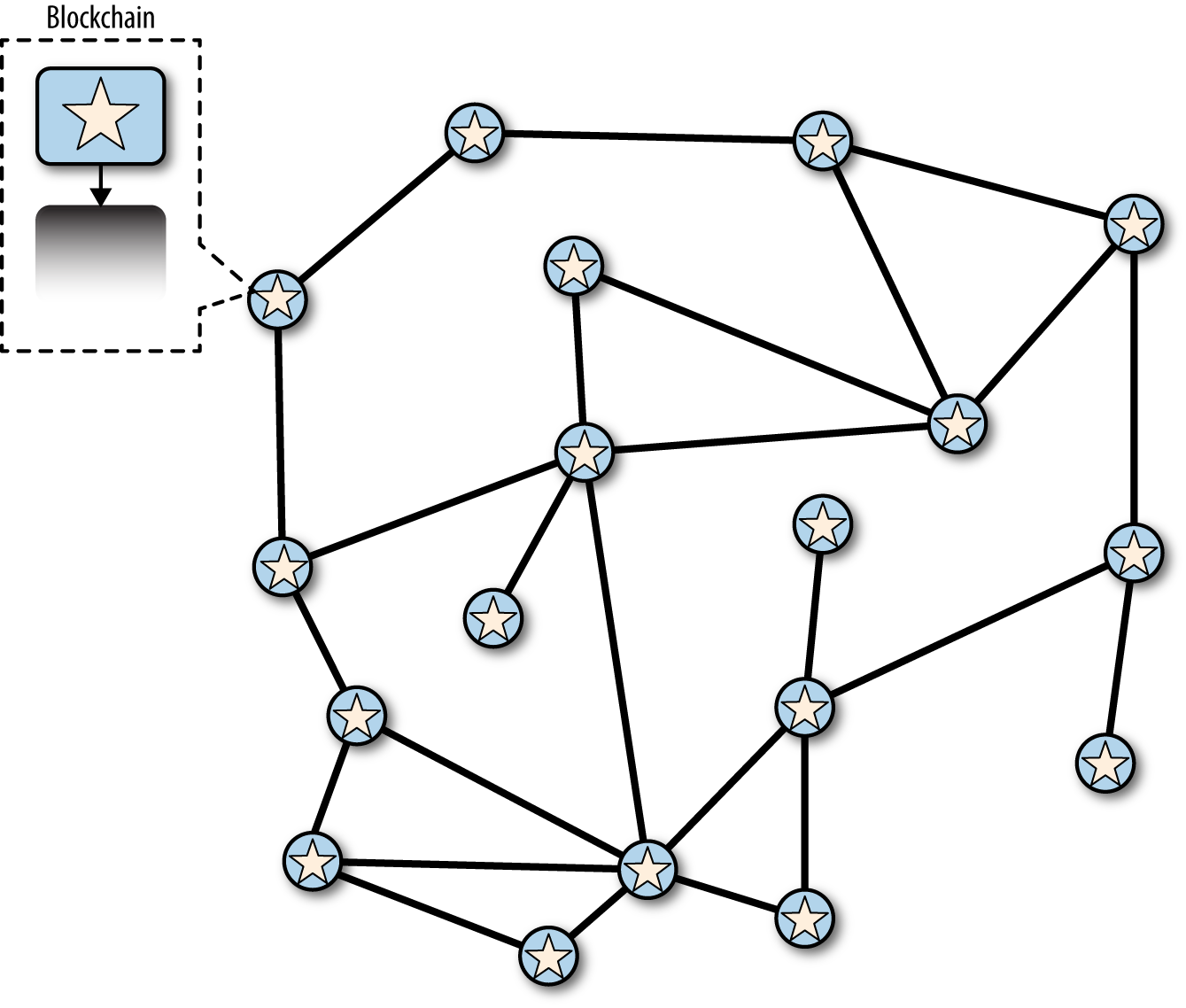
Parce que la chaîne de blocs est une structure de données décentralisée, différentes copies de celle-ci ne sont pas toujours cohérentes. Les blocs peuvent arriver à différents nœuds à des moments différents, ce qui amène les nœuds à avoir différentes perspectives de la chaîne de blocs. Pour résoudre ce problème, chaque nœud sélectionne et tente toujours d'étendre la chaîne de blocs qui représente le plus de preuve de travail, également connue sous le nom de chaîne la plus longue ou de chaîne de travail cumulative la plus importante. En additionnant le travail enregistré dans chaque bloc d'une chaîne, un nœud peut calculer la quantité totale de travail qui a été dépensée pour créer cette chaîne. Tant que tous les nœuds sélectionnent la plus grande chaîne de travail cumulatif, le réseau mondial Bitcoin finit par converger vers un état cohérent. Les fourches se produisent sous forme d'incohérences temporaires entre les versions de la chaîne de blocs, qui sont résolues par une éventuelle reconvergence à mesure que d'autres blocs sont ajoutés à l'une des fourches.

Les fourches de chaîne de blocs décrites dans cette section se produisent naturellement (accidentellement) en raison de retards de transmission dans le réseau mondial. Plus loin dans ce chapitre, nous verrons également les fourches délibérément induites (embranchements divergents et embranchements convergents), qui sont utilisés pour modifier les règles de consensus.

Dans les quelques diagrammes suivants, nous suivons la progression d'un événement d'"embranchement" (ou de fourche) à travers le réseau. Le diagramme est une représentation simplifiée du réseau Bitcoin. À des fins d'illustration, différents blocs sont représentés sous différentes formes (étoile, triangle, triangle inversé, losange), répartis sur le réseau. Chaque nœud du réseau est représenté par un cercle.

Chaque nœud a sa propre perspective de la chaîne de blocs mondiale. Au fur et à mesure que chaque nœud reçoit des blocs de ses voisins, il met à jour sa propre copie de la chaîne de blocs, en sélectionnant la plus grande chaîne de travail cumulée. À des fins d'illustration, chaque nœud contient une forme qui représente le bloc qui, selon lui, est actuellement la pointe de la chaîne principale. Donc, si vous voyez une forme d'étoile dans le nœud, cela signifie que le bloc en étoile est la pointe de la chaîne principale, en ce qui concerne ce nœud.

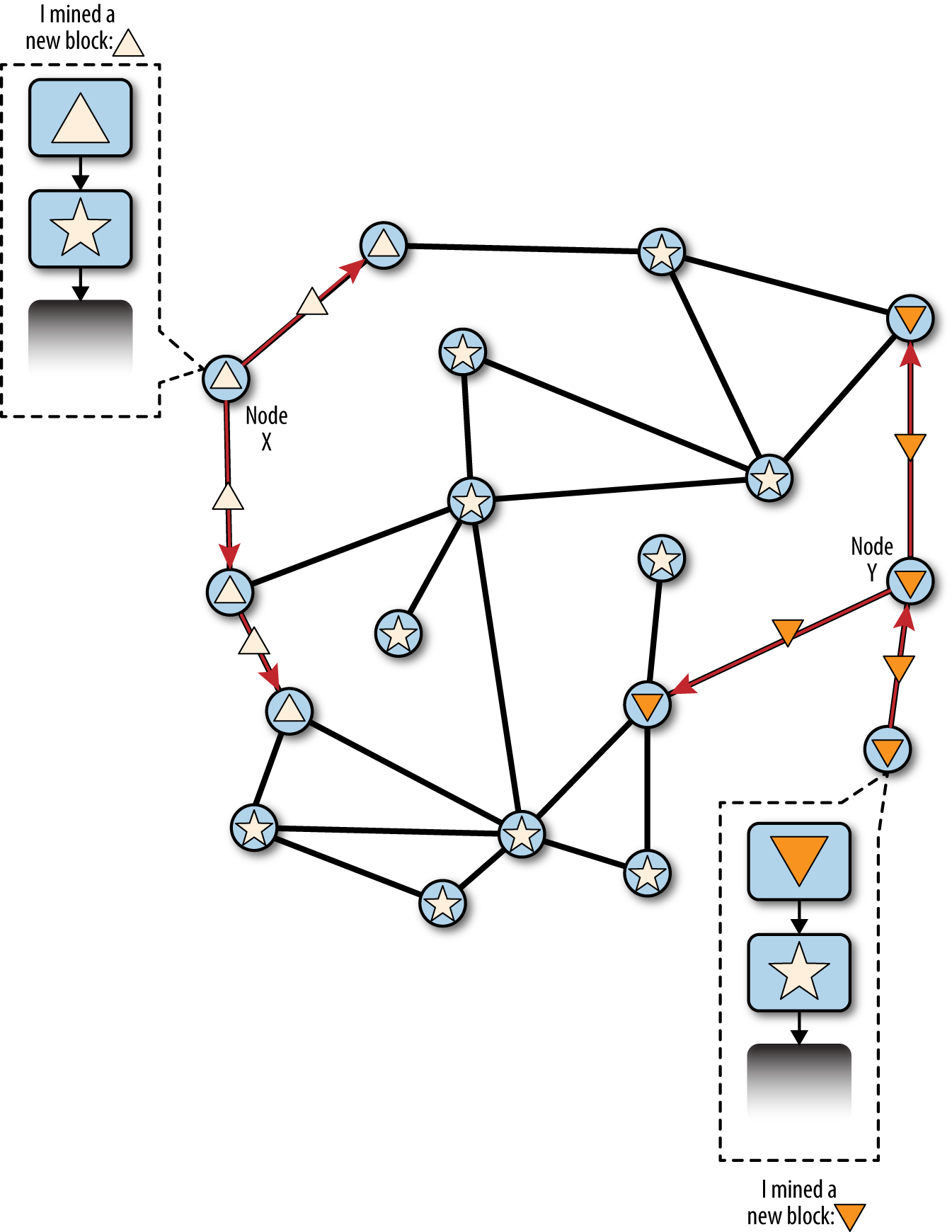
Dans le premier diagramme ([figure\_title](#fork1)), le réseau a une perspective unifiée de la chaîne de blocs, avec le bloc étoile comme pointe de la chaîne principale.



Avant la bifurcation, tous les nœuds ont la même perspective

Un "embranchement" (ou une "fourche") se produit chaque fois qu'il y a deux blocs valides différents à la même hauteur de bloc en compétition pour former la chaîne de blocs la plus longue. Cela se produit dans des conditions normales chaque fois que deux mineurs résolvent l'algorithme de preuve de travail dans un court laps de temps l'un de l'autre. Alors que les deux mineurs découvrent une solution pour leurs blocs candidats respectifs, ils diffusent immédiatement leur propre bloc "gagnant" à leurs voisins immédiats qui commencent à propager le bloc sur le réseau. Chaque nœud qui reçoit un bloc valide l'incorpore dans sa chaîne de blocs, étendant la chaîne de blocs d'un bloc. Si ce nœud voit plus tard un autre bloc valide étendre le même parent (à la même hauteur de bloc), il connecte le deuxième bloc sur une chaîne secondaire, bifurquant sa chaîne principale. En conséquence, certains nœuds "verront" un bloc gagnant en premier, tandis que d'autres nœuds verront l'autre bloc gagnant en premier, et deux versions concurrentes de la chaîne de blocs émergeront.

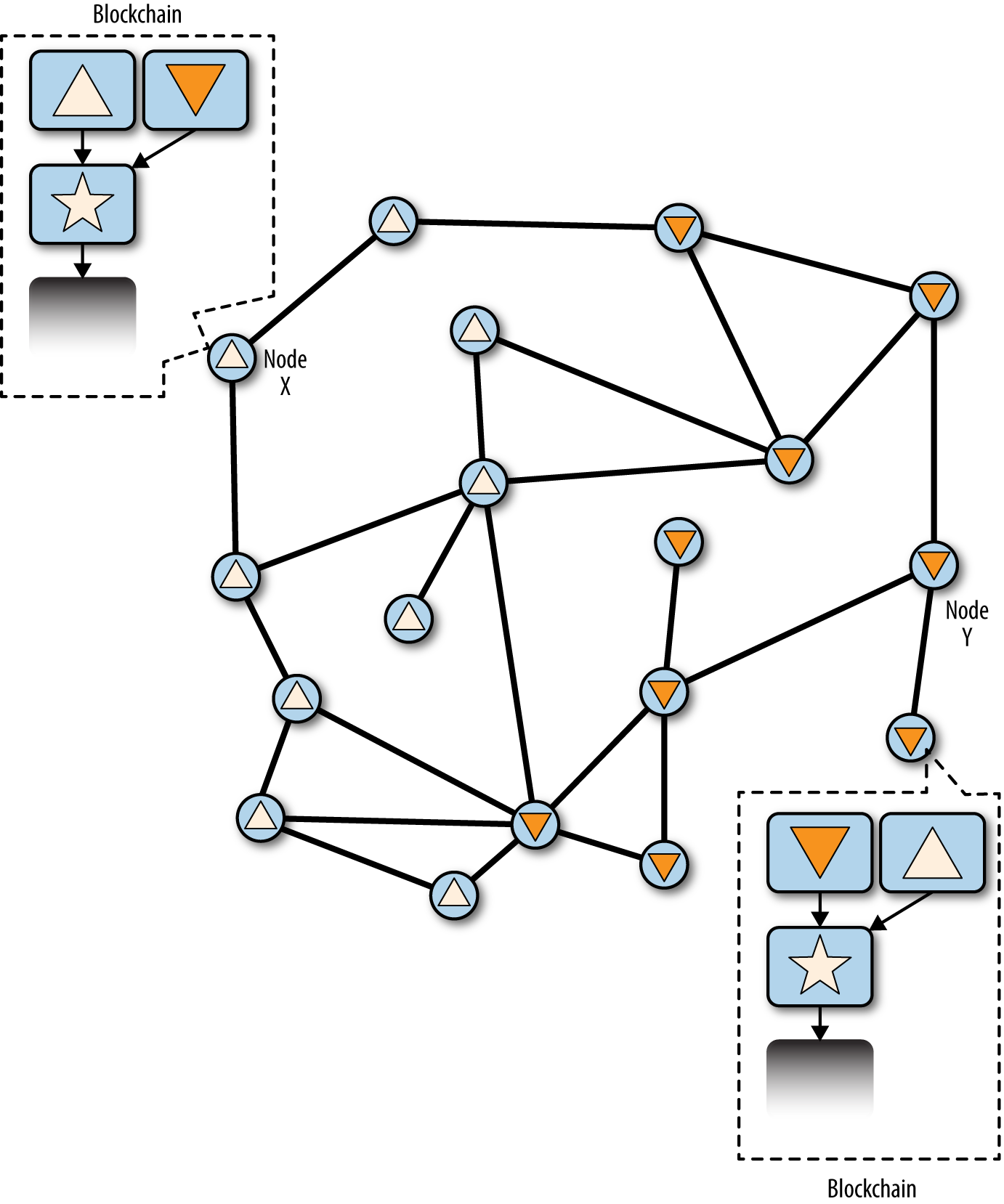
Dans [figure\_title](#fork2), on voit deux mineurs (Noeud X et Noeud Y) qui minent deux blocs différents presque simultanément. Ces deux blocs sont des enfants du bloc étoile et étendent la chaîne en s'appuyant sur le bloc étoile. Pour nous aider à le suivre, l'un est visualisé comme un bloc triangulaire provenant du nœud X, et l'autre est représenté comme un bloc triangulaire inversé provenant du nœud Y.



Visualisation d'un événement d’embranchement (fourche) de la chaîne de blocs: deux blocs trouvés simultanément

Supposons, par exemple, que le mineur Noeud X trouve une solution de preuve de travail pour un bloc "triangle" qui étend la chaîne de blocs, en s'appuyant sur le bloc parent "étoile". Presque simultanément, le mineur Noeud Y qui étendait également la chaîne du bloc "étoile" trouve une solution pour le bloc "triangle inversé", son bloc candidat. Maintenant, il y a deux blocs possibles ; un que nous appelons "triangle", originaire du Nœud X ; et un que nous appelons "triangle inversé", originaire du nœud Y. Les deux blocs ont été extraits avec succès, les deux blocs sont valides (contiennent une solution valide à la preuve de travail), et les deux blocs étendent le même parent (bloc "étoile"). Les deux blocs contiennent probablement la plupart des mêmes transactions, avec seulement peut-être quelques différences dans l'ordre des transactions.

Au fur et à mesure que les deux blocs se propagent, certains nœuds reçoivent d'abord le bloc "triangle" et d'autres reçoivent le bloc "triangle inversé" en premier. Comme indiqué dans [figure\_title](#fork3), le réseau se scinde en deux perspectives différentes de la chaîne de blocs; un côté surmonté du bloc triangle, l'autre du bloc triangle renversé.



Visualisation d'un événement d’embranchement de la chaîne de blocs: deux blocs se propagent, divisant le réseau

Dans [figure\_title](#fork3), le mineur Noeud X a extrait (créé) le bloc triangulaire et a étendu la chaîne d'étoiles avec. Par conséquent, le Nœud X considère initialement la chaîne avec le bloc "triangle" comme la chaîne principale. Plus tard, le Nœud X a également reçu le bloc "triangle inversé" qui a été extrait par le Nœud Y. Puisqu'il a été reçu en deuxième, il est supposé avoir "perdu" la course. Pourtant, le bloc "triangle inversé" n'est pas écarté. Il est lié au bloc parent "étoile" et forme une chaîne secondaire. Alors que le Nœud X suppose que sa chaîne principale est la chaîne gagnante, il conserve la chaîne "perdante" afin qu'il dispose des informations nécessaires pour reconverger si la chaîne "perdante" finit par "gagner".

De l'autre côté du réseau, le mineur Noeud Y construit une blockchain basée sur sa propre perspective de la séquence des événements. Le mineur Noeud Y a extrait (créé) le "triangle inversé" et considère initialement cette chaîne comme la chaîne principale (la chaîne "gagnante"). Lorsqu'il a ensuite reçu le bloc "triangle" extrait par le Nœud X, il l'a connecté au parent du bloc "étoile" en tant que chaîne secondaire.

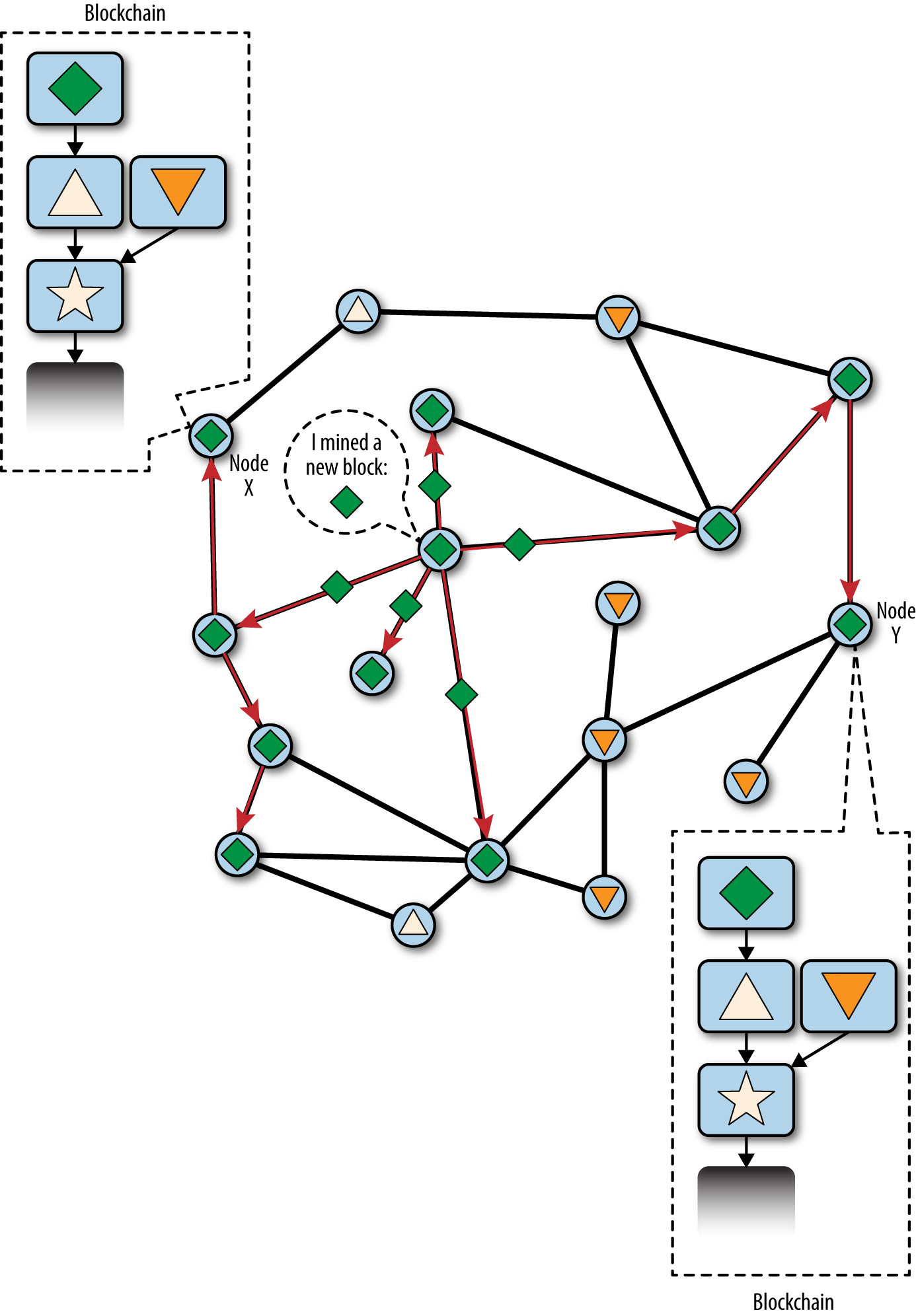
Aucun des deux côtés n'est "correct" ou "incorrect". Les deux sont des perspectives valables de la chaîne de blocs. Ce n'est qu'avec le recul qu'une seule l'emportera, en fonction de la façon dont ces deux chaînes concurrentes sont prolongées par des travaux supplémentaires.

Chaque nœud de minage dont la perspective ressemble au Nœud X commencera immédiatement à miner un bloc candidat qui étend la chaîne avec un "triangle" comme pointe. En liant "triangle" en tant que parent de leurs blocs de candidats, ils votent avec leur pouvoir de hachage. Leur vote soutient la chaîne qu'ils ont élue comme chaîne principale.

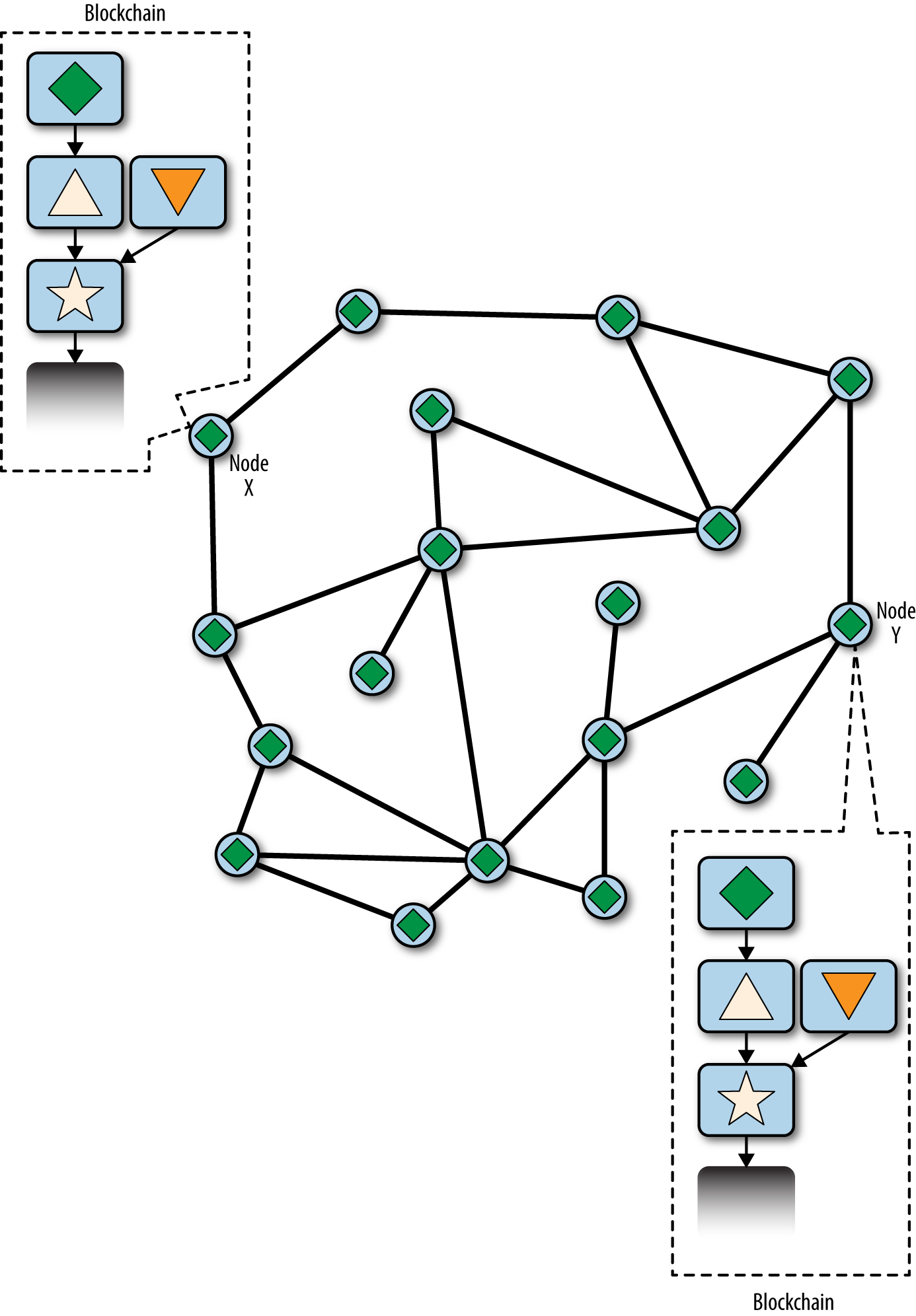
Tout nœud minier dont la perspective ressemble au Nœud Y commencera à construire un bloc candidat avec un "triangle inversé" comme parent, étendant la chaîne qu'ils croient être la chaîne principale. Et voilà, la course recommence.

Les fourches sont presque toujours résolues en un bloc de temps (10 minutes en moyenne). Alors qu'une partie de la puissance de hachage du réseau est dédiée à la construction au-dessus du "triangle" en tant que parent, une autre partie de la puissance de hachage est concentrée sur la construction au-dessus du "triangle inversé". Même si la puissance de hachage est presque également répartie, il est probable qu'un groupe de mineurs trouvera une solution et la propagera avant que l'autre groupe de mineurs ait trouvé des solutions. Disons, par exemple, que les mineurs construisant au-dessus du "triangle" trouvent un nouveau bloc "losange" qui prolonge la chaîne (par exemple, étoile-triangle-losange). Ils propagent immédiatement ce nouveau bloc et l'ensemble du réseau le considère comme une solution valide, comme indiqué dans [figure\_title](#fork4).

Tous les nœuds qui avaient choisi "triangle" comme vainqueur au tour précédent étendront simplement la chaîne d'un bloc supplémentaire. Cependant, les nœuds qui ont choisi "triangle inversé" comme gagnant verront désormais deux chaînes : étoile-triangle-losange et étoile-triangle inversé. La chaîne étoile-triangle-losange est maintenant plus longue (plus de travail cumulé) que l'autre chaîne. En conséquence, ces nœuds définiront la chaîne étoile-triangle-losange comme chaîne principale et transformeront la chaîne étoile-triangle inversé en une chaîne secondaire, comme indiqué dans [figure\_title](#fork5). Il s'agit d'une reconvergence en chaîne, car ces nœuds sont obligés de revoir leur vision de la chaîne de blocs pour intégrer les nouvelles preuves d'une chaîne plus longue. Tous les mineurs travaillant à l'extension de la chaîne étoile-inversée-triangle arrêteront désormais ce travail car leur bloc candidat est désormais considéré comme un enfant d'un bloc périmé, car son parent "triangle-inversé" n'est plus sur la chaîne la plus longue . Étant donné que le bloc triangle inversé est désormais obsolète, le mineur Noeud Y (qui a miné ce bloc) ne pourra pas dépenser la récompense de minage pour ce bloc, même si ce bloc était valide et a été miné avec succès. Les transactions dans le "triangle inversé" qui ne sont pas dans le "triangle" sont réinsérées dans le mempool pour être incluses dans le bloc suivant pour devenir une partie de la chaîne principale. L'ensemble du réseau reconverge sur une seule blockchain étoile-triangle-losange, avec "losange" comme dernier bloc de la chaîne. Tous les mineurs commencent immédiatement à travailler sur des blocs candidats qui font référence à "losange" comme parent pour étendre la chaîne étoile-triangle-losange.



Visualisation d'un événement d’un embranchement de la chaîne de blocs: un nouveau bloc étend une fourche, reconvergeant le réseau. Le Nœud X et le Nœud Y considèrent maintenant le bloc "triangle inversé" comme un bloc obsolète.



Visualisation d'un événement d’embranchement de la chaîne de blocs: le réseau reconverge sur une nouvelle chaîne la plus longue

Il est théoriquement possible qu'une fourche s'étende à deux blocs, si deux blocs sont trouvés presque simultanément par des mineurs sur les "côtés" opposés d'une fourche précédente. Cependant, la probabilité que cela se produise est très faible. Alors qu'une fourche à un bloc peut se produire tous les jours, une fourche à deux blocs se produit au plus une fois toutes les quelques semaines.

L'intervalle de bloc de Bitcoin de 10 minutes est un compromis de conception entre des temps de confirmation rapides (règlement des transactions) et la probabilité d'une fourche. Un temps de bloc plus rapide rendrait les transactions plus rapides mais conduirait à des fourches de chaîne de blocs plus fréquentes, tandis qu'un temps de bloc plus lent réduirait le nombre de fourches mais ralentirait le règlement.

## L'exploitation minière et la course au hachage

L'exploitation minière de Bitcoin est une industrie extrêmement compétitive. La puissance de hachage a augmenté de façon exponentielle chaque année d'existence du bitcoin. Certaines années, la croissance a reflété un changement complet de technologie, comme en 2010 et 2011, lorsque de nombreux mineurs sont passés de l'exploitation de processeurs CPU à l'exploitation de GPU et à l'exploitation de réseau prédiffusé programmable par l’utilisateur (FPGA). En 2013, l'introduction de l'exploitation minière ASIC a conduit à un autre bond de géant dans la puissance minière, en plaçant la fonction SHA256 directement sur des puces de silicium spécialisées à des fins d'exploitation minière. Les premières puces de ce type pourraient fournir plus de puissance de minage dans une seule boîte que l'ensemble du réseau Bitcoin en 2010.

La liste suivante montre la puissance de hachage totale du réseau Bitcoin en terahachages/sec (TH/sec), depuis sa création en 2009 (source : Blockchain.com) :

2009

0,000004 – 0,00001 TH/sec (croissance 2,40×)

2010

0,00001 – 0,14 TH/sec (14 247× croissance)

2011

0,14 – 9,49 TH/sec (croissance de 63,92×)

2012

9,49 – 22 TH/sec (croissance 2,32×)

2013

22.04 – 15 942 TH/sec (croissance de 723,32×)

2014

15 942 – 306 333 TH/sec (croissance de 19,21×)

2015

306 333 – 881 232 TH/sec (croissance de 2,87×)

2016

881 232 – 2 807 540 TH/sec (croissance 3,18×)

2017

2 807 540 – 18 206 558 TH/sec (6,48× croissance)

2018

18 206 558 – 41 801 528 TH/sec (croissance de 2,29×)

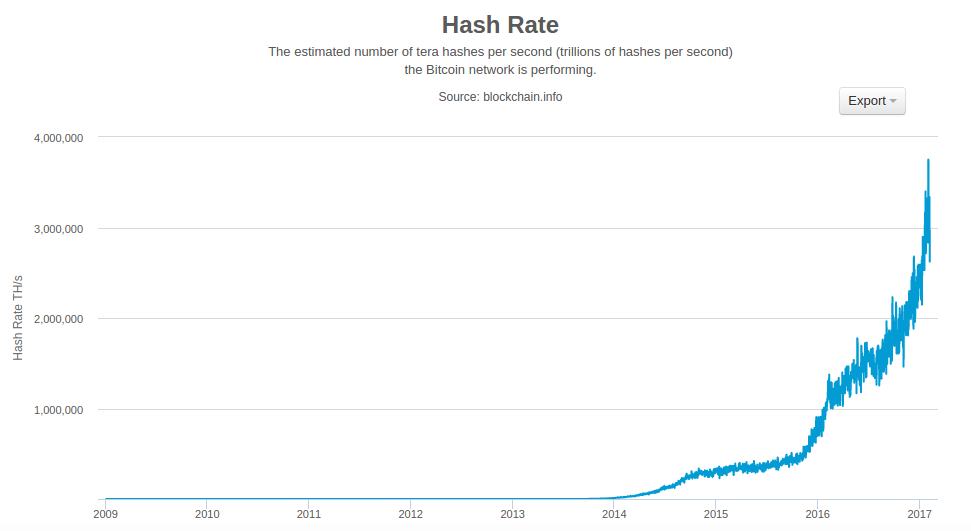
2019

41 801 528 – 109 757 127 TH/sec (croissance de 2,62×)

2020

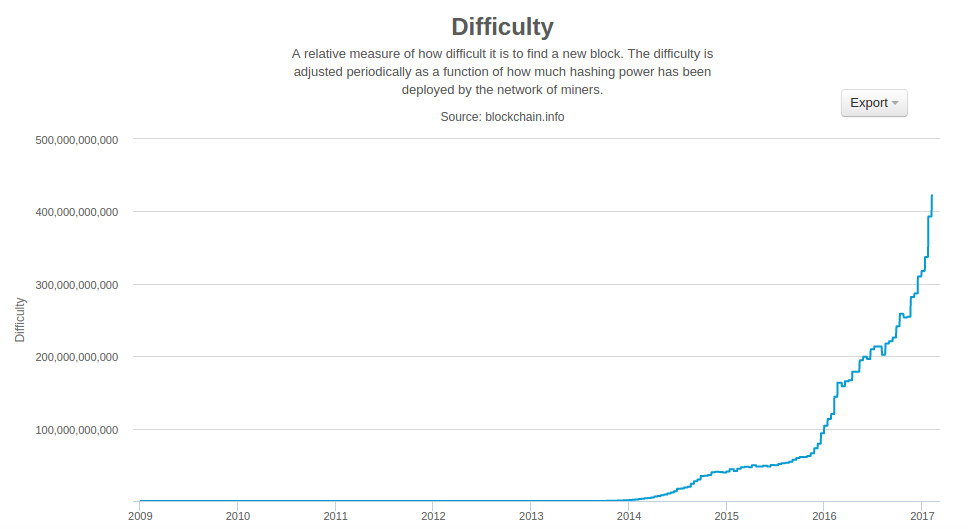
109 757 127 – 149 064 869 TH/sec (croissance de 1,35×)

Dans le graphique en [figure\_title](#network_hashing_power), nous pouvons voir que la puissance de hachage du réseau Bitcoin a augmenté au cours des deux dernières années. Comme vous pouvez le constater, la concurrence entre les mineurs et la croissance du bitcoin ont entraîné une augmentation exponentielle de la puissance de hachage (total des hachages par seconde sur le réseau).



Puissance de hachage totale, terahachages par seconde (TH/sec) (graphique sur une échelle linéaire)

Alors que la quantité de puissance de hachage appliquée à l'extraction de bitcoins a explosé, la difficulté a augmenté pour l'égaler. La métrique de difficulté dans le graphique affiché dans [figure\_title](#bitcoin_difficulty) est mesuré comme un rapport de la difficulté actuelle sur la difficulté minimale (la difficulté du premier bloc).



Métrique de la difficulté de minage de Bitcoin (graphique sur une échelle logarithmique)

Au cours des deux dernières années, les puces minières ASIC sont devenues de plus en plus denses, se rapprochant de la pointe de la fabrication de silicium avec une taille de caractéristique (résolution) de 7 nanomètres (nm). Actuellement, les fabricants d'ASIC visent à dépasser les fabricants de puces CPU à usage général, en concevant des puces avec une taille de fonctionnalité de 5 nm, car la rentabilité de l'exploitation minière fait avancer cette industrie encore plus rapidement que l'informatique générale. Il n'y a plus de pas de géant dans l'extraction de bitcoins, car l'industrie a atteint l'avant-garde de la loi de Moore, qui stipule que la densité de calcul doublera environ tous les 18 mois. Pourtant, la puissance minière du réseau continue de progresser à un rythme exponentiel alors que la course aux puces à plus haute densité s'accompagne d'une course aux centres de données à plus haute densité où des milliers de ces puces peuvent être déployées. Il ne s'agit plus de savoir combien d'extraction peut être effectuée avec une seule puce, mais combien de puces peuvent être insérées dans un bâtiment, tout en dissipant la chaleur et en fournissant une puissance adéquate.

### La solution nonce (nombre aléatoire) supplémentaire

Depuis 2012, l'extraction de bitcoins a évolué pour résoudre une limitation fondamentale dans la structure de l'en-tête de bloc. Au début du bitcoin, un mineur pouvait trouver un bloc en parcourant le nonce jusqu'à ce que le hachage résultant soit égal ou inférieur à la cible. Au fur et à mesure que la difficulté augmentait, les mineurs parcouraient souvent les 4 milliards de valeurs du nonce sans trouver de bloc. Cependant, cela a été facilement résolu en mettant à jour l'horodatage du bloc pour tenir compte du temps écoulé. Étant donné que l'horodatage fait partie de l'en-tête, la modification permettrait aux mineurs de parcourir à nouveau les valeurs du nonce avec des résultats différents. Cependant, une fois que le matériel de minage a dépassé 4 GH/sec, cette approche est devenue de plus en plus difficile car les valeurs nonce ont été épuisées en moins d'une seconde. Alors que l'équipement de minage ASIC commençait à pousser puis à dépasser le taux de hachage TH/sec, le logiciel de minage avait besoin de plus d'espace pour les valeurs nonce afin de trouver des blocs valides. L'horodatage pourrait être un peu étiré, mais le déplacer trop loin dans le futur rendrait le bloc invalide. Une nouvelle source de "changement" était nécessaire dans l'en-tête du bloc. La solution consistait à utiliser la transaction coinbase comme source de valeurs nonce supplémentaires. Étant donné que le script coinbase peut stocker entre 2 et 100 octets de données, les mineurs ont commencé à utiliser cet espace comme espace nonce supplémentaire, leur permettant d'explorer une gamme beaucoup plus large de valeurs d'en-tête de bloc pour trouver des blocs valides. La transaction coinbase est incluse dans l'arborescence de Merkle, ce qui signifie que tout changement dans le script coinbase entraîne la modification de la racine de Merkle. Huit octets de nonce supplémentaires, plus les 4 octets de nonce "standard" permettent aux mineurs d'explorer un total de 296 (8 suivis de 28 zéros) possibilités *par seconde* sans avoir à modifier l'horodatage. Si, à l'avenir, les mineurs pouvaient parcourir toutes ces possibilités, ils pourraient alors modifier l'horodatage. Il y a également plus d'espace dans le script coinbase pour l'expansion future de l'espace nonce supplémentaire.

### Groupes miniers

Dans cet environnement hautement concurrentiel, les mineurs travaillant seuls (également appelés mineurs solo) n’ont aucune chance. La probabilité qu'ils trouvent un bloc pour compenser leurs coûts d'électricité et de matériel est si faible que cela représente un pari, comme jouer à la loterie. Même le système minier ASIC grand public le plus rapide ne peut pas suivre les systèmes commerciaux qui empilent des dizaines de milliers de ces puces dans des entrepôts géants à proximité de centrales hydroélectriques. Les mineurs collaborent désormais pour former des groupes de minage, mettant en commun leur pouvoir de hachage et partageant la récompense entre des milliers de participants. En participant à un groupe, les mineurs obtiennent une plus petite part de la récompense globale, mais sont généralement récompensés tous les jours, ce qui réduit l'incertitude.

Prenons un exemple précis. Supposons qu'un mineur ait acheté du matériel de minage avec un taux de hachage combiné de 14 000 gigahachages par seconde (GH/s), soit 14 TH/s. En 2017, cet équipement coûte environ 2 500 $ US. Le matériel consomme 1375 watts (1,3 kW) d'électricité lorsqu'il fonctionne, 33 kW-heures par jour, à un coût de 1 $ à 2 $ par jour à des tarifs d'électricité très bas. À la difficulté actuelle du bitcoin, le mineur pourra extraire un bloc en solo environ une fois tous les 4 ans. Comment calcule-t-on cette probabilité ? Il est basé sur un taux de hachage à l'échelle du réseau de 3 EH/sec (en 2017) et le taux du mineur de 14 TH/sec :

P = (14 \* 10

12

/ 3 \* 10

18

) \* 210240 = 0,98

…​où 210240 est le nombre de blocs en quatre ans. Le mineur a une probabilité de 98 % de trouver un bloc sur quatre ans, sur la base du taux de hachage global au début de la période.

Si le mineur trouve un seul bloc dans ce délai, le paiement de 6,25 bitcoins, à environ 1 000 $ par bitcoin, se traduira par un paiement unique de 6 250 $, ce qui produira un bénéfice net d'environ 750 $. Cependant, la chance de trouver un bloc dans une période de 4 ans dépend de la chance du mineur. Il pourrait trouver deux blocs en 4 ans et faire un profit plus important. Ou il pourrait ne pas trouver de bloc pendant 5 ans et subir une grosse perte financière. Pire encore, la difficulté de l'algorithme Bitcoin de preuve de travail est susceptible d'augmenter considérablement au cours de cette période, au rythme actuel de croissance de la puissance de hachage, ce qui signifie que le mineur a au plus un an pour atteindre le seuil de rentabilité avant que le matériel soit effectivement obsolète et que ce dernier soit remplacé par du matériel de minage plus puissant. Financièrement, cela n'a de sens qu'à très faible coût d'électricité (moins de 1 cent par kWh) et uniquement à très grande échelle.

Les "pools" ou groupes de minage coordonnent plusieurs centaines ou milliers de mineurs, sur des protocoles de minage de groupe spécialisés. Les mineurs individuels configurent leur équipement minier pour se connecter à un serveur de pool et spécifient une adresse Bitcoin, qui recevra leur part des récompenses. Leur matériel de minage reste connecté au serveur du pool pendant le minage, synchronisant leurs efforts avec les autres mineurs. Ainsi, les mineurs du pool partagent l'effort d'extraction d'un bloc, puis partagent les récompenses.

Les blocs réussis paient la récompense à une adresse Bitcoin de groupe, plutôt qu'à des mineurs individuels. Le serveur de groupe effectuera périodiquement des paiements aux adresses Bitcoin des mineurs, une fois que leur part des récompenses aura atteint un certain seuil. En règle générale, le serveur de groupe facture un pourcentage des récompenses pour la fourniture du service de minage de groupe.

Les mineurs participant à un groupe se partagent le travail de recherche d'une solution à un bloc candidat, gagnant des "actions" pour leur contribution minière. Le groupe de minage fixe un objectif plus élevé (difficulté moindre) pour gagner une part, généralement plus de 1 000 fois plus facile que l'objectif du réseau Bitcoin. Lorsqu'un membre du groupe exploite avec succès un bloc, la récompense est gagnée par le groupe, puis partagée avec tous les mineurs proportionnellement au nombre d'actions qu'ils ont contribué à l'effort.

Les groupes sont ouvertes à tout mineur, grand ou petit, professionnel ou amateur. Un groupe aura donc certains participants avec une seule petite machine de minage, et d'autres avec un garage rempli de matériel de minage haut de gamme. Certains exploiteront quelques dizaines de kilowatts d'électricité, d'autres exploiteront un centre de données consommant un mégawatt d'électricité. Comment un groupe de minage mesure-t-il les contributions individuelles, de manière à répartir équitablement les récompenses, sans possibilité de tricherie ? La réponse est d'utiliser l'algorithme de preuve de travail de Bitcoin pour mesurer la contribution de chaque mineur du groupe, mais avec une difficulté inférieure afin que même les plus petits mineurs du groupe gagnent une part suffisamment fréquemment pour qu'il soit intéressant de contribuer au groupe. En fixant une difficulté inférieure pour gagner des actions, le groupe mesure la quantité de travail effectué par chaque mineur. Chaque fois qu'un mineur de groupe trouve un hachage d'en-tête de bloc égal ou inférieur à la cible du groupe, il prouve qu'il a effectué le travail de hachage pour trouver ce résultat. Plus important encore, le travail de recherche de parts contribue, de manière statistiquement mesurable, à l'effort global pour trouver un hachage égal ou inférieur à la cible du réseau Bitcoin. Des milliers de mineurs essayant de trouver des hachages de faible valeur finiront par en trouver un suffisamment bas pour satisfaire la cible du réseau Bitcoin.

Revenons à l'analogie d'un jeu de dés. Si les joueurs de dés lancent des dés avec un objectif de lancer égal ou inférieur à quatre (la difficulté globale du réseau), un groupe fixerait un objectif plus facile, en comptant combien de fois les joueurs du groupe ont réussi à lancer égal ou inférieur à huit. Lorsque les joueurs du pool lancent un nombre égal ou inférieur à huit (l'objectif de partage du groupe) mais supérieur à quatre (supérieur à la difficulté globale du réseau), ils gagnent des parts, mais ni eux ni le groupe ne gagnent la partie car ils n'y parviennent pas à atteindre la cible (égale ou inférieure à quatre). Les joueurs de groupe atteindront beaucoup plus souvent l'objectif de groupe le plus facile, leur faisant gagner des parts très régulièrement, même s'ils n'atteignent pas l'objectif plus difficile de gagner la partie. De temps en temps, l'un des joueurs du groupe lancera les dés combinés égal ou inférieur à quatre, le joueur du groupe gagne une part et l'ensemble du groupe gagne la partie. Ensuite, les gains peuvent être distribués aux joueurs du groupe en fonction du nombre de parts que chacun a gagné. Même si l'objectif de huit ou moins n'était pas gagnant, c'était un moyen juste de mesurer les lancers de dés pour les joueurs, et cela produit parfois un lancer de quatre ou moins.

De même, un groupe de minage définira une cible de groupe (plus élevée et plus facile) qui garantira qu'un mineur de groupe individuel peut trouver des hachages d'en-tête de bloc qui sont souvent égaux ou inférieurs à la cible du pool, gagnant ainsi des actions. De temps en temps, l'une de ces tentatives produira un hachage d'en-tête de bloc égal ou inférieur à la cible du réseau Bitcoin, ce qui en fait un bloc valide et l'ensemble du groupe gagne.

#### Groupes gérées

La plupart des groupes de minage sont "gérés", ce qui signifie qu'une entreprise ou un individu gère un serveur de groupe. Le propriétaire du serveur de groupe est appelé *opérateur de pool* (ou *opérateur de groupe*), et il facture aux mineurs du groupe un pourcentage des gains.

Le serveur de groupe exécute un logiciel spécialisé et un protocole de minage en groupe qui coordonnent les activités des mineurs en groupe. Le serveur de groupe est également connecté à un ou plusieurs nœuds Bitcoin complets et a un accès direct à une copie complète de la base de données, c-à-d la chaîne de blocs. Cela permet au serveur de groupe de valider les blocs et les transactions au nom des mineurs en groupe, les soulageant ainsi du fardeau d'exécuter un nœud complet. Pour les mineurs en groupe, il s'agit d'une considération importante, car un nœud complet nécessite un ordinateur dédié avec au moins 300 à 350 Go de stockage persistant (disque) et au moins 2 à 4 Go de mémoire (RAM). De plus, le logiciel bitcoin exécuté sur le nœud complet doit être surveillé, entretenu et mis à jour fréquemment. Tout temps d'arrêt causé par un manque d'entretien ou un manque de ressources nuira à la rentabilité du mineur. Pour de nombreux mineurs, la possibilité d'exploiter sans exécuter un nœud complet est un autre grand avantage de rejoindre un groupe géré.

Les mineurs en groupe se connectent au serveur de groupe à l'aide d'un protocole de minage tel que Stratum (STM) ou GetBlockTemplate (GBT). Une norme plus ancienne appelée GetWork (GWK) est pour la plupart obsolète depuis fin 2012, car elle ne prend pas facilement en charge l'extraction à des taux de hachage supérieurs à 4 GH/s. Les protocoles STM et GBT créent des blocs *modèles* qui contiennent un modèle d'en-tête de bloc candidat. Le serveur de groupe construit un bloc candidat en agrégeant les transactions, en ajoutant une transaction coinbase (avec un espace nonce supplémentaire), en calculant la racine de Merkle et en se liant au hachage de bloc précédent. L'en-tête du bloc candidat est ensuite envoyé à chacun des mineurs du groupe en tant que modèle. Chaque mineur du groupe exploite ensuite le modèle de bloc, à une cible plus élevée (plus facile) que la cible du réseau Bitcoin, et renvoie tous les résultats réussis au serveur du groupe pour gagner des parts.

#### Groupe de minage pair à pair (P2Pool)

Les groupes gérés créent la possibilité de tricherie par l'opérateur du groupe, qui pourrait diriger l'effort du groupe pour doubler les transactions ou invalider les blocs (voir [Attaques par consensus](#consensus_attacks)). De plus, les serveurs de groupe centralisés représentent un point de défaillance unique. Si le serveur du groupe est en panne ou est ralenti par une attaque par déni de service, les mineurs du groupe ne peuvent pas exploiter. En 2011, pour résoudre ces problèmes de centralisation, une nouvelle méthode de minage en groupe a été proposée et mise en place: P2Pool, un pool de minage pair à pair sans opérateur central.

P2Pool fonctionne en décentralisant les fonctions du serveur de groupe, en implémentant un système parallèle de type chaîne de blocs appelé *chaîne de partage*. Une chaîne de partage est une chaîne de blocs fonctionnant à une difficulté inférieure à la chaîne de blocs Bitcoin. La chaîne de partage permet aux mineurs en groupe de collaborer dans un groupe décentralisé en extrayant des parts sur la chaîne de partage à raison d'un bloc de partage toutes les 30 secondes. Chacun des blocs de la chaîne de partage enregistre une récompense de part proportionnelle pour les mineurs du groupe qui contribuent au travail, se reportant aux parts du bloc de partage précédent. Lorsque l'un des blocs de partage atteint également l'objectif du réseau Bitcoin, il est propagé et inclus dans la chaîne de blocs Bitcoin, récompensant tous les mineurs du groupe qui ont contribué à toutes les parts qui ont précédé le bloc de partage gagnant. Essentiellement, au lieu qu'un serveur de groupe garde une trace des actions et des récompenses des mineurs en groupe, la chaîne de partage permet à tous les mineurs en groupe de suivre toutes les actions en utilisant un mécanisme de consensus décentralisé comme le mécanisme de consensus de chaîne de blocs de Bitcoin.

L'extraction de P2Pool est plus complexe que l'extraction de groupe car elle nécessite que les mineurs en groupe exécutent un ordinateur dédié avec suffisamment d'espace disque, de mémoire et de bande passante Internet pour prendre en charge un nœud Bitcoin complet et le logiciel de nœud P2Pool. Les mineurs P2Pool connectent leur matériel de minage à leur nœud P2Pool local, qui simule les fonctions d'un serveur de pool en envoyant des modèles de bloc au matériel de minage. Sur P2Pool, les mineurs de pool individuels construisent leurs propres blocs candidats, agrégeant les transactions un peu comme les mineurs en solo, mais exploitent ensuite en collaboration sur la chaîne de partage. P2Pool est une approche hybride qui présente l'avantage de paiements beaucoup plus granulaires que le minage en solo, mais sans donner trop de contrôle à un opérateur de pool comme les groupes gérés.

Même si P2Pool réduit la concentration du pouvoir par les opérateurs de groupes de minage, il est concevable qu'il soit vulnérable à 51 % des attaques contre la chaîne de partage elle-même. Une adoption beaucoup plus large de P2Pool ne résout pas le problème d'attaque de 51% pour le bitcoin lui-même. Au contraire, P2Pool rend le bitcoin plus robuste dans l'ensemble, dans le cadre d'un écosystème minier diversifié.

## Attaques par consensus

Le mécanisme de consensus de Bitcoin est, du moins théoriquement, vulnérable aux attaques des mineurs (ou les groupes) qui tentent d'utiliser leur pouvoir de hachage à des fins malhonnêtes ou destructrices. Comme nous l'avons vu, le mécanisme de consensus dépend du fait qu'une majorité de mineurs agissent honnêtement dans leur propre intérêt. Cependant, si un mineur ou un groupe de mineurs peut atteindre une part importante de la puissance minière, il peut attaquer le mécanisme de consensus afin de perturber la sécurité et la disponibilité du réseau Bitcoin.

Il est important de noter que les attaques de consensus ne peuvent affecter que le consensus futur, ou au mieux, le passé le plus récent (dizaines de blocs). Le registre de Bitcoin devient de plus en plus immuable au fil du temps. Alors qu'en théorie, une fourche peut être réalisé à n'importe quelle profondeur, en pratique, la puissance de calcul nécessaire pour forcer une fourche très profonde est immense, rendant les anciens blocs pratiquement immuables. Les attaques par consensus n'affectent pas non plus la sécurité des clés privées et de l'algorithme de signature (ECDSA). Une attaque par consensus ne peut pas voler des bitcoins, dépenser des bitcoins sans signatures, rediriger des bitcoins ou autrement modifier des transactions passées ou des enregistrements de propriété. Les attaques par consensus ne peuvent affecter que les blocs les plus récents et provoquer des interruptions par déni de service sur la création de futurs blocs.

Un scénario d'attaque contre le mécanisme de consensus est appelé "l'attaque à 51 %". Dans ce scénario, un groupe de mineurs, contrôlant la majorité (51%) de la puissance de hachage totale du réseau, s'entend pour attaquer le bitcoin. Avec la capacité d'exploiter la majorité des blocs, les mineurs attaquants peuvent provoquer des "fourches" délibérés dans la chaîne de blocs et doubler les transactions ou exécuter des attaques par déni de service contre des transactions ou des adresses spécifiques. Une attaque fourche/double-dépense est l'endroit où l'attaquant provoque l'invalidation de blocs précédemment confirmés en forgeant en dessous d'eux et en reconvergeant sur une chaîne alternative. Avec une puissance suffisante, un attaquant peut invalider six blocs ou plus à la suite, provoquant l'invalidation de transactions considérées comme immuables (six confirmations). Notez qu'une double dépense ne peut être effectuée que sur les propres transactions de l'attaquant, pour lesquelles l'attaquant peut produire une signature valide. Dépenser deux fois ses propres transactions est rentable si, en invalidant une transaction, l'attaquant peut obtenir un paiement ou un produit d'échange irréversible sans le payer.

Examinons un exemple pratique d'une attaque à 51 %. Dans le premier chapitre, nous avons examiné une transaction entre Alice et Bob pour une tasse de café. Bob, le propriétaire du café, est prêt à accepter le paiement pour des tasses de café sans attendre de confirmation (extraction en bloc), car le risque de double dépense pour une tasse de café est faible par rapport à la commodité d'un service client rapide . Ceci est similaire à la pratique des cafés qui acceptent les paiements par carte de crédit sans signature pour des montants inférieurs à 25 $, car le risque de rétrofacturation par carte de crédit est faible tandis que le coût de retarder la transaction pour obtenir une signature est comparativement plus élevé. En revanche, vendre un article plus cher pour du bitcoin présente le risque d'une attaque à double dépense, où l'acheteur diffuse une transaction concurrente qui dépense les mêmes intrants (UTXO) et annule le paiement au commerçant. Une attaque par double dépense peut se produire de deux manières : soit avant qu'une transaction ne soit confirmée, soit si l'attaquant profite d'une fourche de chaîne de blocs pour annuler plusieurs blocs. Une attaque à 51 % permet aux attaquants de doubler leurs propres transactions dans la nouvelle chaîne, annulant ainsi la transaction correspondante dans l'ancienne chaîne.

Dans notre exemple, l'attaquant malveillant Mallory se rend dans la galerie de Carol et achète un magnifique triptyque représentant Satoshi Nakamoto en Prométhée. Carol vend les peintures "The Great Fire" pour 250 000 $ en bitcoins à Mallory. Au lieu d'attendre six confirmations ou plus sur la transaction, Carol emballe et remet les peintures à Mallory après une seule confirmation. Mallory travaille avec un complice, Paul, qui exploite un grand groupe minier, et le complice lance une attaque à 51% dès que la transaction de Mallory est incluse dans un bloc. Paul ordonne au groupe minier de reminer la même hauteur de bloc que le bloc contenant la transaction de Mallory, en remplaçant le paiement de Mallory à Carol par une transaction qui double la même entrée que le paiement de Mallory. La transaction à double dépense consomme le même UTXO et le rembourse au portefeuille de Mallory, au lieu de le payer à Carol, permettant essentiellement à Mallory de conserver le bitcoin. Paul dirige ensuite le groupe de minage pour exploiter un bloc supplémentaire, de manière à rendre la chaîne contenant la transaction à double dépense plus longue que la chaîne d'origine (provoquant une bifurcation sous le bloc contenant la transaction de Mallory). Lorsque la fourche de la chaîne de blocs se résout en faveur de la nouvelle chaîne (plus longue), la transaction à double dépense remplace le paiement initial à Carol. Carol manque maintenant les trois peintures et n'a pas non plus de paiement en bitcoin. Tout au long de cette activité, les participants au groupe de minage de Paul pourraient parfaitement ignorer la tentative de double dépense, car ils minent avec des mineurs automatisés et ne peuvent pas surveiller chaque transaction ou chaque bloc.

Pour se protéger contre ce type d'attaque, un commerçant vendant des articles de grande valeur doit attendre au moins six confirmations avant remettre le produit à l'acheteur. Alternativement, le commerçant doit utiliser un compte multisignature à garantie bloqué, attendant à nouveau plusieurs confirmations après le financement du compte à garantie bloqué. Plus le nombre de confirmations s'écoule, plus il devient difficile d'invalider une transaction avec une attaque à 51 %. Pour les articles de grande valeur, le paiement par bitcoin sera toujours pratique et efficace même si l'acheteur doit attendre 24 heures pour la livraison, ce qui correspondrait à environ 144 confirmations.

En plus d'une attaque à double dépense, l'autre scénario d'une attaque par consensus consiste à refuser le service à des participants Bitcoin spécifiques (adresses Bitcoin spécifiques). Un attaquant disposant de la majorité de la puissance minière peut simplement ignorer des transactions spécifiques. S'ils sont inclus dans un bloc miné par un autre mineur, l'attaquant peut délibérément bifurquer et réminer ce bloc, en excluant à nouveau les transactions spécifiques. Ce type d'attaque peut entraîner un déni de service prolongé contre une adresse spécifique ou un ensemble d'adresses aussi longtemps que l'attaquant contrôle la majorité de la puissance de minage.

Malgré son nom, le scénario d'attaque à 51 % ne nécessite pas réellement 51 % de la puissance de hachage. En fait, une telle attaque peut être tentée avec un pourcentage plus faible de la puissance de hachage. Le seuil de 51% est simplement le niveau auquel une telle attaque est presque garantie de réussir. Une attaque consensuelle est essentiellement un bras de fer pour le bloc suivant et le groupe "le plus fort" a plus de chances de gagner. Avec moins de puissance de hachage, la probabilité de succès est réduite, car d'autres mineurs contrôlent la génération de certains blocs avec leur puissance de minage "honnête". Une façon de voir les choses est que plus un attaquant a de puissance de hachage, plus la fourche qu'il peut délibérément créer est long, plus il peut invalider de blocs dans le passé récent ou plus il peut contrôler de blocs dans le futur. Des groupes de recherche en sécurité ont utilisé la modélisation statistique pour affirmer que divers types d'attaques par consensus sont possibles avec aussi peu que 30 % de la puissance de hachage.

L'augmentation massive de la puissance de hachage totale a sans doute rendu le bitcoin insensible aux attaques d'un seul mineur. Il n'y a aucun moyen pour un mineur solo de contrôler plus qu'un petit pourcentage de la puissance minière totale. Cependant, la centralisation du contrôle causée par les groupes de minage a introduit le risque d'attaques à but lucratif par un opérateur de groupe de minage. L'opérateur d’un groupe géré contrôle la construction des blocs candidats et contrôle également les transactions incluses. Cela donne à l'opérateur du groupe le pouvoir d'exclure des transactions ou d'introduire des transactions à double dépense. Si un tel abus de pouvoir est fait de manière limitée et subtile, un opérateur de groupe pourrait éventuellement profiter d'une attaque par consensus sans se faire remarquer.

Cependant, tous les attaquants ne seront pas motivés par le profit. Un scénario d'attaque potentiel est celui où un attaquant a l'intention de perturber le réseau Bitcoin sans avoir la possibilité de profiter d'une telle perturbation. Une attaque malveillante visant à paralyser le bitcoin nécessiterait d'énormes investissements et une planification secrète, mais pourrait éventuellement être lancée par un attaquant bien financé, très probablement parrainé par l'État. Alternativement, un attaquant bien financé pourrait attaquer le consensus de bitcoin en amassant simultanément du matériel de minage, en compromettant les opérateurs de groupe et en attaquant d'autres groupes par déni de service. Tous ces scénarios sont théoriquement possibles, mais de plus en plus impraticables à mesure que la puissance de hachage globale du réseau Bitcoin continue de croître de façon exponentielle.

Sans aucun doute, une attaque sérieuse du consensus éroderait la confiance dans le bitcoin à court terme, provoquant éventuellement une baisse significative des prix. Cependant, le réseau et les logiciels Bitcoin évoluent constamment, de sorte que les attaques par consensus se heurteraient à des contre-mesures immédiates de la part de la communauté bitcoin, rendant le bitcoin plus robuste.

## Modification des règles de consensus

Les règles de consensus déterminent la validité des transactions et des blocs. Ces règles constituent la base de la collaboration entre tous les nœuds Bitcoin et sont responsables de la convergence de toutes les perspectives locales en une seule chaîne de blocs cohérente sur l'ensemble du réseau.

Alors que les règles de consensus sont invariables à court terme et doivent être cohérentes à travers tous les nœuds, elles ne sont pas invariables à long terme. Afin d'évoluer et de développer le système Bitcoin, les règles doivent changer de temps en temps pour s'adapter aux nouvelles fonctionnalités, améliorations ou corrections de bogues. Contrairement au développement de logiciels traditionnels, cependant, les mises à niveau vers un système de consensus sont beaucoup plus difficiles et nécessitent une coordination entre tous les participants.

### Embranchements divergents

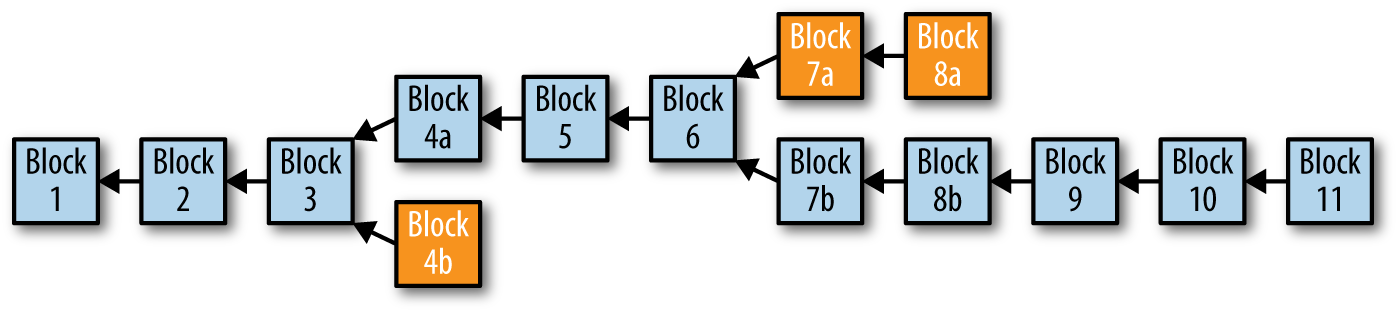
Dans [Fourches de chaîne de blocs](#forks) nous avons examiné comment le réseau Bitcoin peut brièvement diverger, avec deux parties du réseau suivant deux branches différentes de la blockchain pendant une courte période. Nous avons vu comment ce processus se produit naturellement, dans le cadre du fonctionnement normal du réseau et comment le réseau reconverge sur une chaîne de blocs commune après le minage d'un ou plusieurs blocs.

Il existe un autre scénario dans lequel le réseau peut diverger en deux chaînes suivantes : un changement dans les règles de consensus. Ce type de fourche ou embranchement est appelé *embranchement divergent* (*hard fork*), car après l’embranchement ou la fourche, le réseau ne reconverge pas sur une seule chaîne. Au lieu de cela, les deux chaînes évoluent indépendamment. Les embranchements divergents se produisent lorsqu'une partie du réseau fonctionne selon un ensemble de règles de consensus différent du reste du réseau. Cela peut se produire à cause d'un bogue ou à cause d'un changement délibéré dans l'implémentation des règles de consensus.

Les embranchements divergents peuvent être utilisés pour modifier les règles du consensus, mais ils nécessitent une coordination entre tous les participants au système. Tous les nœuds qui ne se mettent pas à niveau vers les nouvelles règles de consensus ne peuvent pas participer au mécanisme de consensus et sont forcés à une chaîne distincte au moment de l’embranchement divergent. Ainsi, un changement introduit par un embranchement divergent peut être considéré comme non « compatible en aval », dans la mesure où les systèmes non mis à niveau ne peuvent pas traiter les nouvelles règles de consensus après l'événement d’embranchement divergent.

Examinons les mécanismes d'un embranchement divergent avec un exemple spécifique.

[figure\_title](#blockchainwithforks) montre une chaîne de blocs avec deux fourches. À la hauteur de bloc 4, une fourche à un bloc se produit. C'est le type de fourche spontané que nous avons vu dans [Fourches de chaîne de blocs](#forks). Avec le minage du bloc 5, le réseau reconverge sur une chaîne et l’embranchement est résolu.



Une blockchain avec des fourches

Plus tard, cependant, à la hauteur de bloc 6, une bifurcation divergente se produit. Supposons qu'une nouvelle implémentation du client soit publiée avec un changement dans les règles de consensus. À partir de la hauteur de bloc 7, les mineurs exécutant cette nouvelle implémentation accepteront un nouveau type de signature numérique, appelons-le une signature "Smores", qui n'est pas basée sur ECDSA. Immédiatement après, un nœud exécutant la nouvelle implémentation crée une transaction qui contient une signature Smores et un mineur avec le logiciel mis à jour mine le bloc 7b contenant cette transaction.

Tout nœud ou mineur qui n'a pas mis à jour le logiciel pour valider les signatures Smores est désormais incapable de traiter le bloc 7b. De leur point de vue, la transaction qui contenait une signature Smores et le bloc 7b qui contenait cette transaction sont invalides, car ils les évaluent sur la base des anciennes règles de consensus. Ces nœuds rejetteront la transaction et le bloc et ne les propageront pas. Tous les mineurs qui utilisent les anciennes règles n'accepteront pas le bloc 7b et continueront à exploiter un bloc candidat dont le parent est le bloc 6. En fait, les mineurs utilisant les anciennes règles peuvent même ne pas recevoir le bloc 7b si tous les nœuds auxquels ils sont connectés sont obéissant également aux anciennes règles et donc ne propageant pas le bloc. À terme, ils pourront exploiter le bloc 7a, qui est valide selon les anciennes règles et ne contient aucune transaction avec des signatures Smores.

Les deux chaînes continuent de diverger à partir de ce point. Les mineurs de la chaîne "b" continueront d'accepter et d'exploiter les transactions contenant des signatures Smores, tandis que les mineurs de la chaîne "a" continueront d'ignorer ces transactions. Même si le bloc 8b ne contient aucune transaction signée Smores, les mineurs de la chaîne "a" ne peuvent pas le traiter. Pour eux, il semble être un bloc orphelin, car son parent "7b" n'est pas reconnu comme un bloc valide.

### Embranchements divergents: logiciel, réseau, exploitation minière et chaîne

Pour les développeurs de logiciels, le terme "fourche" a une autre signification, ajoutant de la confusion au terme "fourche divergente". Dans les logiciels à source libre, une fourche se produit lorsqu'un groupe de développeurs choisit de suivre une feuille de route logicielle différente et démarre une implémentation concurrente d'un projet à source libre. Nous avons déjà discuté de deux circonstances qui conduiront à un embranchement divergent en Bitcoin: un bogue dans les règles de consensus et une modification délibérée des règles de consensus. Dans le cas d'une modification délibérée des règles de consensus, une fourche logicielle précède l’embranchement divergent. Cependant, pour que ce type d’embranchement divergent se produise, une nouvelle implémentation logicielle des règles de consensus doit être développée, adoptée et lancée.

Des exemples de fourches logicielles qui ont tenté de modifier les règles de consensus incluent Bitcoin XT, Bitcoin Classic et, plus récemment, Bitcoin Unlimited. Cependant, aucun de ces fourches logiciels n'a abouti à un embranchement divergent. Bien qu'une fourche logicielle soit une condition préalable nécessaire, il n'est pas suffisant en soi pour qu'un embranchement divergent se produise. Pour qu'un embranchement divergent se produise, l'implémentation concurrente doit être adoptée et les nouvelles règles activées, par les mineurs, les portefeuilles et les nœuds intermédiaires. A l'inverse, il existe de nombreuses implémentations alternatives de Bitcoin Core, et même des fourches logiciels, qui ne changent pas les règles de consensus, et sauf par bogue, peuvent coexister sur le réseau et interagir sans provoquer d’embranchement divergent.

Les règles de consensus peuvent différer de manière évidente et explicite, dans la validation des transactions ou des blocs. Les règles peuvent également différer de manière plus subtile, dans la mise en œuvre des règles de consensus telles qu'elles s'appliquent aux scripts bitcoin ou aux primitives cryptographiques telles que les signatures numériques. Enfin, les règles de consensus peuvent différer de manière imprévue en raison des contraintes de consensus implicites imposées par les limitations du système ou les détails de mise en œuvre. Un exemple de ce dernier a été vu dans l’embranchement divergent imprévu lors de la mise à niveau de Bitcoin Core 0.7 vers 0.8, qui a été causé par une limitation de l'implémentation de Berkeley DB utilisée pour stocker les blocs.

Conceptuellement, nous pouvons considérer qu'un embranchement divergent se développe en quatre étapes : une fourche logiciel, une fourche du réseau, une fourche de minage et une fourche de chaîne.

Le processus commence lorsqu'une implémentation alternative du client, avec des règles de consensus modifiées, est créée par les développeurs.

Lorsque cette implémentation fourchée est déployée dans le réseau, un certain pourcentage de mineurs, d'utilisateurs de portefeuille et de nœuds intermédiaires peuvent adopter et exécuter cette implémentation. Une fourche résultante dépendra de l'application ou non des nouvelles règles de consensus aux blocs, aux transactions ou à tout autre aspect du système. Si les nouvelles règles de consensus concernent les transactions, un portefeuille créant une transaction selon les nouvelles règles peut précipiter une fourche réseau, suivi d'un embranchement divergent lorsque la transaction est extraite dans un bloc. Si les nouvelles règles concernent les blocs, le processus d’embranchement divergent commencera lorsqu'un bloc sera miné selon les nouvelles règles.

Tout d'abord, le réseau bifurquera. Les nœuds basés sur l'implémentation d'origine des règles de consensus rejetteront toutes les transactions et tous les blocs créés selon les nouvelles règles. De plus, les nœuds qui suivent les règles de consensus d'origine seront temporairement bannis et déconnectés de tous les nœuds qui leur envoient ces transactions et blocs invalides. En conséquence, le réseau va se partitionner en deux : les anciens nœuds ne resteront connectés qu'aux anciens nœuds et les nouveaux nœuds ne seront connectés qu'aux nouveaux nœuds. Une transaction ou un bloc unique basé sur les nouvelles règles se répercutera sur le réseau et entraînera la partition en deux réseaux.

Une fois qu'un mineur utilisant les nouvelles règles exploite un bloc, la puissance et la chaîne de minage bifurqueront également. Les nouveaux mineurs exploiteront le nouveau bloc, tandis que les anciens mineurs exploiteront une chaîne distincte basée sur les anciennes règles. Le réseau partitionné fera en sorte que les mineurs opérant selon des règles de consensus distinctes ne recevront probablement pas les blocs les uns des autres, car ils sont connectés à deux réseaux distincts.

### Mineurs divergents et difficulté

Alors que les mineurs divergent pour exploiter deux chaînes différentes, la puissance de hachage est répartie entre les chaînes. La puissance minière peut être répartie dans n'importe quelle proportion entre les deux chaînes. Les nouvelles règles ne peuvent être suivies que par une minorité, ou par la grande majorité de la puissance minière.

Supposons, par exemple, une répartition de 80 % à 20 %, la majorité de la puissance minière utilisant les nouvelles règles de consensus. Supposons également que la fourche se produise immédiatement après une période de reciblage.

Les deux chaînes hériteraient chacune de la difficulté de la période de reciblage. Les nouvelles règles consensuelles auraient 80% de la puissance minière précédemment disponible qui leur serait engagée. Du point de vue de cette chaîne, la puissance minière a subitement reculé de 20% par rapport à la période précédente. Des blocs seront trouvés en moyenne toutes les 12,5 minutes, ce qui représente la baisse de 20 % de la puissance minière disponible pour prolonger cette chaîne. Ce taux d'émission de blocs se poursuivra (sauf modification de la puissance de hachage) jusqu'à ce que les blocs de 2016 soient extraits, ce qui prendra environ 25 200 minutes (à 12,5 minutes par bloc), soit 17,5 jours. Après 17,5 jours, un reciblage se produira et la difficulté s'ajustera (réduite de 20%) pour produire à nouveau des blocs de 10 minutes, en fonction de la quantité réduite de puissance de hachage dans cette chaîne.

La chaîne minoritaire, minant selon les anciennes règles avec seulement 20% de la puissance de hachage, sera confrontée à une tâche beaucoup plus difficile. Sur cette chaîne, les blocs seront désormais minés toutes les 50 minutes en moyenne. La difficulté ne sera pas ajustée pour les blocs de 2016, qui prendront 100 800 minutes, soit environ 10 semaines à miner. En supposant une capacité fixe par bloc, cela se traduira également par une réduction de la capacité de transaction par un facteur de 5, car il y a moins de blocs par heure disponibles pour enregistrer les transactions.

### Embranchements divergents litigieux

C'est l'aube du développement de logiciels consensuels. Tout comme le développement à source libre a changé à la fois les méthodes et les produits des logiciels et a créé de nouvelles méthodologies, de nouveaux outils et de nouvelles communautés dans son sillage, le développement de logiciels consensuels représente également une nouvelle frontière en informatique. Des débats, des expériences et des tribulations de la feuille de route de développement du bitcoin, nous verrons émerger de nouveaux outils, pratiques, méthodologies et communautés de développement.

Les fourches dures sont considérées comme risquées car elles obligent une minorité à se mettre à niveau ou à rester sur une chaîne minoritaire. Le risque de scinder l'ensemble du système en deux systèmes concurrents est considéré par beaucoup comme un risque inacceptable. En conséquence, de nombreux développeurs hésitent à utiliser le mécanisme d’embranchement divergent pour mettre en œuvre des mises à niveau des règles de consensus, à moins qu'il n'y ait un soutien quasi unanime de l'ensemble du réseau. Toutes les propositions d’embranchement divergent qui ne bénéficient pas d'un soutien quasi unanime sont considérées comme trop "controversées" à tenter sans risquer une partition du système.

La question des embranchements divergents est très controversée dans la communauté du développement de Bitcoin, en particulier en ce qui concerne toute modification proposée aux règles de consensus contrôlant la limite de taille de bloc maximale. Certains développeurs sont opposés à toute forme d’embranchement divergent, le considérant comme trop risqué. D'autres considèrent le mécanisme d’embranchement divergent comme un outil essentiel pour mettre à niveau les règles de consensus d'une manière qui évite la "dette technique" et offre une rupture nette avec le passé. Enfin, certains développeurs voient les embranchements divergents comme un mécanisme qui devrait être utilisé rarement, avec beaucoup de planification préalable et uniquement sous un consensus quasi unanime.

Nous avons déjà vu l'émergence de nouvelles méthodologies pour faire face aux risques d’embranchements divergents. Dans la section suivante, nous examinerons les embranchements convergents et les méthodes BIP-34 et BIP-9 pour la signalisation et l'activation des modifications de consensus.

### Embranchements convergents

Tous les changements de règles de consensus ne provoquent pas un embranchement divergent. Seuls les changements de consensus qui sont incompatibles en aval provoquent un embranchement ou fourche. Si la modification est implémentée de telle manière qu'un client non mis à niveau voit toujours la transaction ou le bloc comme valide selon les règles précédentes, la modification peut se produire sans fourche.

Le terme *embranchement convergent* (*soft fork*) a été introduit pour distinguer cette méthode de mise à niveau d'un "embranchement divergent" (hard fork). En pratique, un embranchement convergent n'est pas du tout un embranchement ou une fourche. Un embranchement convergent est une modification compatible avec les règles de consensus qui permet aux clients non mis à niveau de continuer à fonctionner en consensus avec les nouvelles règles.

Un aspect des embranchements convergents qui n'est pas immédiatement évident est que les mises à niveau d’embranchement convergent ne peuvent être utilisées que pour restreindre les règles de consensus, pas pour les étendre. Afin d'être compatibles en aval, les transactions et les blocs créés selon les nouvelles règles doivent également être valides selon les anciennes règles, mais pas l'inverse. Les nouvelles règles ne peuvent que limiter ce qui est valide; sinon, ils déclencheront un embranchement divergent s'ils sont rejetés selon les anciennes règles.

Les embranchements convergents peuvent être implémentés de plusieurs façons - le terme ne spécifie pas une méthode particulière, mais plutôt un ensemble de méthodes qui ont toutes une chose en commun : elles ne nécessitent pas la mise à niveau de tous les nœuds ou ne forcent pas les nœuds non mis à niveau hors de consensus.

#### Embranchements convergents redéfinissant les opcodes NOP

Un certain nombre de embranchements convergents ont été implémentés dans bitcoin, sur la base de la réinterprétation d'opcodes NOP. Bitcoin Script avait dix opcodes réservés pour une utilisation future, NOP1 à NOP10. Selon les règles de consensus, la présence de ces opcodes dans un script est interprétée comme un opérateur nul, ce qui signifie qu'ils n'ont aucun effet. L'exécution continue après l'opcode NOP comme s'il n'était pas là.

Un embranchement convergent peut donc modifier la sémantique d'un code NOP pour lui donner un nouveau sens. Par exemple, BIP-65 (CHECKLOCKTIMEVERIFY) a réinterprété l'opcode NOP2. Les clients implémentant BIP-65 interprètent NOP2 comme OP\_CHECKLOCKTIMEVERIFY et imposent une règle de consensus de verrouillage absolu sur UTXO qui contient cet opcode dans leurs scripts de verrouillage. Ce changement est un embranchement convergent car une transaction valide sous BIP-65 est également valide sur tout client qui n'implémente pas (ignore) BIP-65. Pour les anciens clients, le script contient un code NOP, qui est ignoré.

#### Autres façons de mettre à niveau l’embranchement convergent

La réinterprétation des opcodes NOP était à la fois prévue et un mécanisme évident pour les mises à niveau consensuelles. Récemment, cependant, un autre mécanisme d’embranchement convergent a été introduit qui ne repose pas sur les opcodes NOP pour un type très spécifique de changement de consensus. Ceci est examiné plus en détail dans [???](#segwit). Segwit est une modification architecturale de la structure d'une transaction, qui déplace le script de déverrouillage (témoin) de l'intérieur de la transaction vers une structure de données externe (en la séparant). Segwit a été initialement envisagé comme une mise à niveau de embranchement divergent, car il modifiait une structure fondamentale (transaction). En novembre 2015, un développeur travaillant sur Bitcoin Core a proposé un mécanisme par lequel segwit pourrait être introduit en tant que embranchement convergent. Le mécanisme utilisé pour cela est une modification du script de verrouillage d'UTXO créé selon les règles segwit, de sorte que les clients non mis à niveau voient le script de verrouillage comme échangeable avec n'importe quel script de déverrouillage. En conséquence, segwit peut être introduit sans nécessiter que chaque nœud soit mis à niveau ou séparé de la chaîne : un embranchement convergent.

Il est probable qu'il existe d'autres mécanismes, encore à découvrir, par lesquels les mises à niveau peuvent être effectuées de manière compatible avec les versions ultérieures en tant qu’embranchement convergent.

### Critiques d’embranchements convergents

Les embranchements convergents basés sur les opcodes NOP sont relativement peu controversés. Les opcodes NOP ont été placés dans Bitcoin Script dans le but explicite de permettre des mises à niveau non perturbatrices.

Cependant, de nombreux développeurs craignent que d'autres méthodes de mises à niveau d’embranchements convergents ne fassent des compromis inacceptables. Les critiques courantes des changements d’embranchement convergent incluent:

Dette technique

Étant donné que les embranchements convergents sont plus complexes techniquement qu'une mise à niveau de embranchements divergents, ils introduisent la *dette technique*, un terme qui fait référence à l'augmentation du coût futur de la maintenance du code en raison des compromis de conception effectués dans le passé. La complexité du code augmente à son tour la probabilité de bogues et de failles de sécurité.

Assouplissement de la validation

les clients non mis à niveau voient les transactions comme valides, sans évaluer les règles de consensus modifiées. En effet, les clients non mis à niveau ne valident pas en utilisant la gamme complète des règles de consensus, car ils sont aveugles aux nouvelles règles. Cela s'applique aux mises à niveau basées sur NOP, ainsi qu'aux autres mises à niveau d’embranchements convergents.

Mises à niveau irréversibles

Étant donné que les embranchements convergents créent des transactions avec des contraintes de consensus supplémentaires, elles deviennent des mises à niveau irréversibles dans la pratique. Si une mise à niveau d’embranchement convergent devait être annulée après avoir été activée, toute transaction créée selon les nouvelles règles pourrait entraîner une perte de fonds selon les anciennes règles. Par exemple, si une transaction CLTV est évaluée selon les anciennes règles, il n'y a pas de contrainte de timelock et elle peut être dépensée à tout moment. Par conséquent, les critiques soutiennent qu'un embranchement convergent défaillant qui devait être annulé à cause d'un bogue entraînerait presque certainement une perte de fonds.

## Signalisation d’embranchement convergent avec la version du bloc

Puisque les embranchements convergents permettent aux clients non mis à jour de continuer à fonctionner dans le cadre du consensus, le mécanisme pour « activer » un embranchement convergent consiste à signaler que les mineurs sont prêts : une majorité de mineurs doit convenir qu'ils sont prêts et disposés à appliquer les nouvelles règles consensuelles. Pour coordonner leurs actions, il existe un mécanisme de signalisation qui leur permet de montrer leur soutien à un changement de règle consensuel. Ce mécanisme a été introduit avec l'activation de BIP-34 en mars 2013 et remplacé par l'activation de BIP-9 en juillet 2016.

### Signalisation et activation BIP-34

La première implémentation, dans BIP-34, utilisait le champ de version de bloc pour permettre aux mineurs de signaler qu'ils étaient prêts pour un changement de règle de consensus spécifique. Avant le BIP-34, la version du bloc était définie par la valeur "1" par *convention* non appliquée par *consensus*.

BIP-34 a défini un changement de règle de consensus qui exigeait que le champ de données coinbase d'une entrée de transaction coinbase contienne la hauteur du bloc. Avant le BIP-34, les données coinbase pouvaient contenir toutes les données arbitraires que les mineurs choisissaient d'inclure. Après l'activation de BIP-34, les blocs valides devaient contenir une hauteur de bloc spécifique au début des données coinbase et être identifiés par un numéro de version supérieur ou égal à "2".

Pour signaler le changement et l'activation de BIP-34, les mineurs ont défini la version du bloc à "2" au lieu de "1". Cela n'a pas immédiatement rendu les blocs de la version "1" invalides. Une fois activés, les blocs de la version "1" deviendraient invalides et tous les blocs de la version "2" devraient contenir la hauteur du bloc dans la coinbase pour être valides.

BIP-34 a défini un mécanisme d'activation en deux étapes, basé sur une fenêtre glissante de 1000 blocs. Un mineur signalerait sa préparation individuelle pour le BIP-34 en construisant des blocs avec "2" comme numéro de version. À proprement parler, ces blocs n'avaient pas encore à se conformer à la nouvelle règle de consensus consistant à inclure la hauteur de bloc dans la transaction coinbase car la règle de consensus n'avait pas encore été activée. Les règles de consensus activées en deux étapes :

* Si 75% (750 des 1000 blocs les plus récents) sont marqués avec la version "2", alors les blocs de la version "2" doivent contenir la hauteur du bloc dans la transaction coinbase ou ils sont rejetés comme invalides. Les blocs de version "1" sont toujours acceptés par le réseau et n'ont pas besoin de contenir de hauteur de bloc. Les anciennes et les nouvelles règles de consensus coexistent pendant cette période.
* Lorsque 95 % (950 des 1 000 blocs les plus récents) sont de version "2", les blocs de version "1" ne sont plus considérés comme valides. Les blocs de la version "2" ne sont valides que s'ils contiennent la hauteur de bloc dans le coinbase (selon le seuil précédent). Par la suite, tous les blocs doivent respecter les nouvelles règles de consensus et tous les blocs valides doivent contenir la hauteur de bloc dans la transaction coinbase.

Après une signalisation et une activation réussies selon les règles BIP-34, ce mécanisme a été utilisé deux fois de plus pour activer les embranchements convergents :

* [BIP-66](https://github.com/bitcoin/bips/blob/master/bip-0066.mediawiki) Le codage DER strict des signatures a été activé par une signalisation de style BIP-34 avec une version de bloc "3" et invalidant la version "2" des blocs.
* [BIP-65](https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki) CHECKLOCKTIMEVERIFY a été activé par la signalisation de style BIP-34 avec une version de bloc "4" et invalidant la version "3" des blocs.

Après l'activation de BIP-65, le mécanisme de signalisation et d'activation de BIP-34 a été retiré et remplacé par le mécanisme de signalisation BIP-9 décrit ci-après.

La norme est définie dans [BIP-34 (Block v2, Height in Coinbase)](https://github.com/bitcoin/bips/blob/master/bip-0034.mediawiki).

### Signalisation et activation BIP-9

Le mécanisme utilisé par BIP-34, BIP-66 et BIP-65 a réussi à activer trois embranchements convergents. Cependant, il a été remplacé car il présentait plusieurs limitations :

* En utilisant la valeur entière de la version du bloc, un seul embranchement convergent pouvait être activé à la fois, il fallait donc une coordination entre les propositions d’embranchements convergents et un accord sur leur hiérarchisation et leur séquencement.
* De plus, étant donné que la version de bloc était incrémentée, le mécanisme n'offrait pas de moyen simple de rejeter une modification et d'en proposer une autre. Si d'anciens clients étaient toujours en cours d'exécution, ils pourraient confondre la signalisation d'une nouvelle modification avec la signalisation de la modification précédemment rejetée.
* Chaque nouveau changement réduit irrévocablement les versions de blocs disponibles pour les changements futurs.

Le BIP-9 a été proposé pour surmonter ces défis et améliorer le rythme et la facilité de mise en œuvre des changements futurs.

BIP-9 interprète la version de bloc comme un champ de bits au lieu d'un entier. Étant donné que la version de bloc était à l'origine utilisée comme entier, versions 1 à 4, seuls 29 bits restent disponibles pour être utilisés comme champ de bits. Cela laisse 29 bits qui peuvent être utilisés pour signaler indépendamment et simultanément la disponibilité sur 29 propositions différentes.

BIP-9 fixe également un temps maximum pour la signalisation et l'activation. De cette façon, les mineurs n'ont pas besoin de signaler indéfiniment. Si une proposition n'est pas activée dans le délai TIMEOUT (défini dans la proposition), la proposition est considérée comme rejetée. La proposition peut être soumise à nouveau pour signalisation avec un bit différent, renouvelant la période d'activation.

De plus, après que le TIMEOUT est passé et qu'une fonctionnalité a été activée ou rejetée, le bit de signalisation peut être réutilisé pour une autre fonctionnalité sans confusion. Ainsi, jusqu'à 29 changements peuvent être signalés en parallèle et après TIMEOUT les bits peuvent être "recyclés" pour proposer de nouveaux changements.

Alors que les bits de signalisation peuvent être réutilisés ou recyclés, tant que la période de vote ne se chevauche pas, les auteurs du BIP-9 recommandent que les bits ne soient réutilisés que lorsque cela est nécessaire ; un comportement inattendu peut se produire en raison de bogues dans des logiciels plus anciens. En bref, nous ne devrions pas nous attendre à voir une réutilisation tant que les 29 bits n'auront pas été utilisés une fois.

Les modifications proposées sont identifiées par une structure de données contenant les champs suivants :

name

Une brève description utilisée pour distinguer les propositions. Le plus souvent, le BIP décrivant la proposition, comme "bipN", où N est le numéro du BIP.

bit

0 à 28, le bit dans la version de bloc que les mineurs utilisent pour signaler l'approbation de cette proposition.

starttime

Le temps (basé sur le temps passé médian, ou MTP) auquel la signalisation commence après lequel la valeur du bit est interprétée comme signalant la préparation de la proposition.

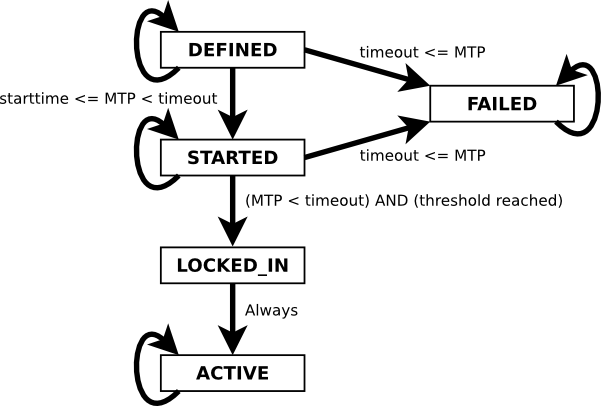
endtime

Le temps (basé sur MTP) après lequel le changement est considéré comme rejeté s'il n'a pas atteint le seuil d'activation.

Contrairement à BIP-34, BIP-9 compte la signalisation d'activation dans des intervalles entiers en fonction de la période de reciblage de difficulté de 2016 blocs. Pour chaque période de reciblage, si la somme des blocs signalant une proposition dépasse 95 % (1 916 sur 2016), la proposition sera activée une période de reciblage plus tard.

BIP-9 propose un diagramme d'état de proposition pour illustrer les différentes étapes et transitions d'une proposition, comme indiqué dans [figure\_title](#bip9states).

Les propositions commencent dans l'état DEFINED, une fois que leurs paramètres sont connus (définis) dans le logiciel bitcoin. Pour les blocs avec MTP après l'heure de début, l'état de la proposition passe à STARTED. Si le seuil de vote est dépassé dans une période de reciblage et que le délai d'attente n'a pas été dépassé, l'état de la proposition passe à LOCKED\_IN. Une période de reciblage plus tard, la proposition devient ACTIVE. Les propositions restent perpétuellement dans l'état ACTIVE une fois qu'elles atteignent cet état. Si le délai d'attente s'est écoulé avant que le seuil de vote n'ait été atteint, l'état de la proposition passe à FAILED, indiquant une proposition rejetée. Les propositions FAILED restent perpétuellement dans cet état.



BIP-9 diagramme de transition d'état

BIP-9 a d'abord été implémenté pour l'activation de CHECKSEQUENCEVERIFY et des BIP associés (68, 112, 113). La proposition nommée "csv" a été activée avec succès en juillet 2016.

La norme est définie dans [BIP-9 (Version bits with timeout and delay)](https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki).

## Développement de logiciel de consensus

Le logiciel de consensus continue d'évoluer et il y a beaucoup de discussions sur les divers mécanismes pour changer le règles consensuelles. De par sa nature même, le bitcoin place la barre très haute en matière de coordination et de consensus pour les changements. En tant que système décentralisé, il n'a pas d'"autorité" qui puisse imposer sa volonté aux participants du réseau. Le pouvoir est réparti entre plusieurs groupes tels que les mineurs, les développeurs principaux, les développeurs de portefeuilles, les échanges, les commerçants et les utilisateurs finaux. Les décisions ne peuvent être prises unilatéralement par aucun de ces groupes. Par exemple, alors que les mineurs peuvent théoriquement changer les règles à la majorité simple (51%), ils sont contraints par le consentement des autres circonscriptions. S'ils agissent unilatéralement, le reste des participants peut simplement refuser de les suivre, gardant l'activité économique sur une chaîne minoritaire. Sans activité économique (transactions, marchands, portefeuilles, échanges), les mineurs exploiteront une pièce sans valeur avec des blocs vides. Cette diffusion du pouvoir signifie que tous les participants doivent se coordonner, sinon aucun changement ne peut être apporté. Le statu quo est l'état stable de ce système avec seulement quelques changements possibles s'il y a un fort consensus à une très large majorité. Le seuil de 95% pour les embranchements convergents reflète cette réalité.

Il est important de reconnaître qu'il n'y a pas de solution parfaite pour le développement d'un consensus. Les embranchements divergents et les embranchements convergents impliquent des compromis. Pour certains types de changements, les embranchements convergents peuvent être un meilleur choix ; pour d'autres, les embranchements divergents peuvent être un meilleur choix. Il n'y a pas de choix parfait; les deux comportent des risques. La seule caractéristique constante du développement de logiciels consensuels est que le changement est difficile et que le consensus oblige à faire des compromis.

Certains y voient une faiblesse des systèmes de consensus. Avec le temps, vous finirez peut-être par le voir comme moi, comme la plus grande force du système.