Contentos技術白皮書

Contentos是一個聚焦於內容服務的區塊鏈系統。系統提供了完善的帳號權限等級和授權管理功能，能夠最大程度的保護帳號安全，同時滿足靈活的運營需求。內置的經濟規則鼓勵用戶向系統貢獻價值內容、加強互動和參與度。在底層區塊鏈共識機制方面，Contentos採用DPoS，相比於PoW大幅提升了TPS。在DPoS基礎上，還引入了BFT機制進一步提升服務響應速度，並實現iBFT算法，能夠根據系統的動態參數自動適配最佳的共識模式。

Contentos系統提供基於JSON RPC的接口服務，支持前端應用通過HTTP、HTTPS或Web socket接入。系統還提供智能合約功能，允許用戶自行開發dApp，最大化的滿足個性化需求。為了進一步提升智能合約系統的易用性，Contentos向智能合約開放豐富的API，不僅支持鏈上數據查詢和轉賬交易，還支持全面的內容管理，以及合約之間的靈活調用。針對常見的用戶業務需求，Contentos還提供一批智能合約模版，降低智能合約應用的技術門檻，方便更多用戶快速部署。

1. 共識機制

1.1 設計理念

受制於計算機科學中的CAP理理論： 一個分佈式系統最多只能同時滿 足 一致性（Consistency）、可用性（Availability）和分區容錯性（Partition tolerance）這三項中的兩項。 Contentos 借鑒了眾多公鏈的底層設計及核心理念，研究顯式處理網絡分區案，通過優化數據的一致性算法及可用性，取得三者之間的平衡。進而使整個Contentos 網絡的效率、一致性、不可逆性及開放性得到極大提升。基於鏈的DPoS算法傾向於選擇可用性高的而不選擇一致性高的，因為可用性高意味著所有的交易都能被處理，不過要以犧牲整個網絡中一致性狀態復制為代價。基於BFT的卻相反，會傾向於選擇高一致性。

Contentos基於BFT實現了一種智能的、動態調整的共識算法 iBFT（*Intelligence Byzantine Fault Tolerance*） 。

iBFT共識算法有三種共識模式：

- 線性（Linear）模式 -- 大幅降低通信消息，節省網絡資源，處理更多的區塊鏈交易；
- 及時（Timely）模式 -- 犧牲一定的網絡資源，區塊能夠比較快的達成共識，進而縮短區塊鏈交易的確認時間；
- 混合（Mixing）模式 -- Contentos 網絡能夠在比較節省網絡資源的情況下保證較短的確認時間，集成兩種共識模式的優點；

Contentos的共識模式是組件化設計， 支持熱插拔、可定制、可配置為智能動態選擇。在智能選擇的模式下，智能算法會根據整個 網絡的算力、帶寬、交易待確認量、 網絡擁堵情況等等一系列的因素智能選擇最合適的共識模式。

1.2 記賬人選舉規則

Contentos基於DPoS算法，實現了記賬 人的公開選舉。 Contentos 網絡中所有持有COS的用戶都有權申請競選記賬人，同時所有持有COS的用戶有資格投票。每人最多投30票，得票最多的前20個節點負責生產區塊，還有一個名額是得票20名以外的節點輪流 生產。這21個節點被稱作記賬人。 Contentos每3秒產生一個區塊，並且在任一時間內僅有一個記賬人有權生產區塊。每一輪產生21個區塊。每輪開始時，根據COS持有者選出新的21個記賬人，記賬人的生產順序由2/3或更更多的記賬人同意決定。

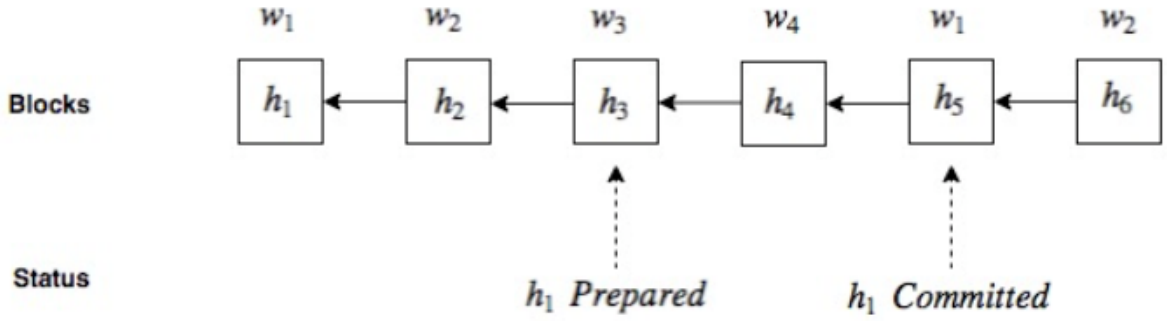
1.3 共識模式設計

在Contentos區塊鍊網絡中，共有n個記賬人，輪詢表 $S(w_1, w_2, \ldots, w_n)$上的每 一個 $slot$ 就是 一個視圖，稱之為s。另外區塊為b, 區塊高度為h, 父區塊Hash為H , 第i個記賬人節點對區塊的簽名為$\langle b \rangle_{\sigma i}$ 。

Linear共識模式

1. 在某一 $timestamp$, 記賬人w_1 生產高度為 h_1 的區塊;
2. 後續記賬人繼續生產區塊$h_2, h_3, h_4 \ldots$，每生產一個區塊相當於當前記賬人對從自己上一次生產區塊高度到當前區塊高度之間所有區塊做了一次確認；注意記賬人不對自己生產的區塊做確認；
3. 當對某一高度h_i 的區塊的確認數達到$\frac{2}{3}n + 1$時， 高度為h_i 的區塊進 入$Prepared$ 狀態;
4. 後續記賬人繼續生產區塊, 當$Prepared$ 狀態的區塊數達到$\frac{2}{3}n + 1$時，按照進入$Prepared$ 狀態的順序排序，前$\frac{1}{3}$ 的區塊鏈進入$Committed$ 狀態，成為不可逆區塊持久化到區塊鏈數據庫中.

假設Contentos網絡中有4個記賬人：

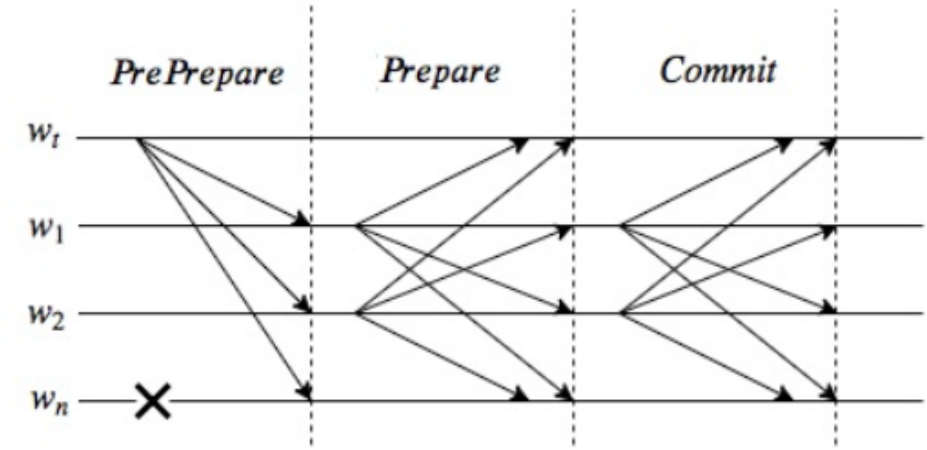


Timely共識模式

1. 在某一$timestamp$，記賬人w_t 生產區塊 b，簽名後廣播。預請求消息: $\langle PrePrepare\langle s, b, H, h, \langle b \rangle_{\sigma t} \rangle$
2. 記賬人節點w_i 第一次收到預請求消息後，驗證消息有效性，然後廣播準備消息 $\langle Prepare\langle s, b, H, h, \langle b \rangle_{\sigma i} \rangle$
3. 當記賬人節點w_i 收集到至少$2f$ 個其他記賬人節點簽名的準備消息後，$block$進入

$Prepared$ 狀態，對所有節點廣播確認消息$\langle Commit\langle s,b,H,h,\langle b\rangle_{\sigma i}\rangle$

4. 記賬人節點w_i 收集到$2f$ 個確認消息後，就認為對該區塊的共識已經達成一致，區塊進入$Committed$ 狀態，並將其持久化到區塊鏈數據庫中。



Mixing共識模式

1. 在某一 $timestamp$, 記賬人w_1 生產高度為 h_1 的區塊;
2. 後續記賬人繼續生產區塊$h_2, h_3, h_4 \ldots$，每生產一個區塊，當前記賬人對從自己上一次生產區塊高度的下一個區塊開始到當前區塊高度所有區塊做一次確認;
3. 當對某一高度h_i的區塊的確認數達到 $\frac{2}{3}n+1$ 時，高度為h_i的區塊進入$Prepared$狀態;
4. 當前記賬人廣播高度為h_i的區塊的準備消息$\langle Commit\langle s,H_i,H_p,h,\langle b\rangle_{\sigma}\rangle$，所有記賬人收到 $langleCommit\rangle$消息後，校驗區塊，更改高度為h_i 的區塊進入$Prepared$狀態，並轉發$\langle Commit\rangle$消息;
5. 所有記賬人至少收集到$2f$個確認後，就認為對該區塊的共識已經達成一致，區塊進入$Committed$狀態，並將其持久化到區塊鏈數據庫中。

拜占庭懲罰

拜占庭問題之所以難解，在於任何時候系統中都可能存在多個提案，並且要完成最終的一致性確認過程十分困難，容易受干擾。而在提案時，記賬人付出的成本極低，加劇了惡意拜占庭節點的滋生。在Contentos 網絡中，記賬人在出塊或者在驗證區塊時的拜占庭行為很容易被檢測。例如（包括但不不限於）：

- 記賬人不合法的出塊
- 記賬人多次無效出塊或者沒有出塊
- 記賬人無序的交織出塊或者共識投票
- 記賬人對同一高度不同區塊的重複共識投票

記賬人任何無效的行為後，將被Cententos管控，限制其記賬權。除此之外，Contentos引入經濟懲罰進一步限制記賬人的行為。記賬人必須先繳納保證金才可以參與出塊和共識形成。如果一個記賬人產生Contentos認為是作惡的行為，保證金將被罰沒。保證 金的加入確保了記賬人作惡的成本。

2. 經濟系統

- Contentos 系統內通貨標識為 COS，COS 參與一切流通性行為，在目前版本下，流通特指錢包之前的轉賬行為。
- 持有通貨並不會得到額外的權益，如果需要獲取系統權利，需要對通貨進行凍結，以換取無利率憑據。系統內，該憑據的標識為 VESTS。 VESTS 持有的數量直接決定了系統內操作的影響力，比如可以改變獎勵池的分配關係。足夠的 VESTS 數量也提供了競選為 Witness 的資格。
- COS 和 VESTS 的比例恆定為 1:1
- Contentos 系統每三秒產生一個新的區塊，每個區塊產生時，新的 VESTS 將會被追加進獎勵池中
- 每個區塊產生的 VESTS 數目由數學模型所決定。必要時，所有理事會成員可以通過投票干預模型，以調節新增 VESTS 數目，從而使系統內經濟能滿足生產力需求
- 所有用戶通過參與系統內活動，以從獎勵池中分享 VESTS。
- Dapp的開發者會拿到整個獎勵池中不多於10%的獎勵。
- 目前系統主要活動是“創造內容”，詳細來說包括上傳視頻以及發表評論。其中，前者分享 63% 獎勵池中收益，後者分享 13.5% 獎勵池的收益。獎勵池是累計的，每個新區塊產生的時候，會在鏈上去查找是否有滿足獎勵結算的對象，如果沒有，本次新產生的獎勵會累加進獎勵池。
- 每個新區塊產生獎勵的 9% 會直接作為激勵進入 Witness 的賬戶
- 最後 4.5% 的收益結算比較特殊，這部分是對舉報的獎勵。用戶通過抵押一部分 VESTS ，可以對視頻或者評論進行舉報。舉報提交後，會有仲裁者在隨後查看是否這是一個有效的舉報。如果是，那麼抵押的 VESTS 返還，並且從舉報獎勵池中按照比例分享獎勵。如果不是，那麼這部分抵押的 VESTS 會加入舉報獎勵池。舉報獎勵池不是永久累加的，每 1000 個塊會重置掉舉報獎勵池。如果重置時間點有滿足獎勵的對象，那麼會進行獎勵，否則，池中 VESTS 會對半放入創作獎勵池和評論獎勵池。該機制用來確保“有對舉報激勵的手段，同時防止投機式舉報”。特別的，如果一個視頻或者評論已經被仲裁者仲裁過，那麼舉報請求會直接駁回。
- 創造者和評論者獎勵遵循同樣的算法，稱為”過去窗口權重累計算法“。具體到某個創作品和評論，其在結算時間點分得的VESTS 數主要由三個因素決定，第一個是當前的“過去積累rshares ”，第二個是當前獎勵池的大小，第三個是該作品的rshares。
- rshares 用來描述一個作品被投票（點贊）之後獲取的權重。該值由兩個變量決定，一個是投票用戶的 VESTS，一個是該用戶的 VOTE POWER。後者由一個複雜的公式計算出來。 VESTS 和 VOTE POWER 進行一個計算之後，可以得出該用戶的 rshare。而所有對該作品進行投票用戶的 rshare 之和，即為 rshares。
- 每當一個作品被結算，其 rshares 會被累加進“過去積累 rshares”，即 `p_rshares` 中。為了防止 `p_rshares` 只增不減，額外規定了 `p_shares` 的衰減函數。準確來說，每一個區塊產生的時候， `p_shares` 會衰減掉 `3 / (86400 * 17) p_shares` 的數量。也就是說，某一個 rshares 會持續影響 17 天內的結算。
- 如果已知 `p_shares` , `rewards` , 與一個作品在結算點的 `shares` ，那麼該作品作者能得到的獎勵為 `shares * (rewards / p_shares)` 。
- 特別的，雖然創作品和評論遵循同樣的算法，但是其 p_shares 的累計函數並不相同，創

作品是簡單的線性累計，評論獎勵池為平方根算法。
- 每個作品發佈到結算的窗口期為 7 天，當前時間 + 7 * 86400 即為結算時間，結算有且只有一次。如果對超過窗口期的作品進行投票，該投票會被直接忽略。
- COS 與 VESTS 可以互相轉化。前者到後者的過程稱為 Power up，因為權利增加了。反之後者向前者的轉化稱為 Power down。
- power up 轉化是瞬間的，並且可以把自己的 COS 轉給別人的 VESTS 賬戶。
- power down 過程是漸進的，申請提交後，每周可以獲取到轉化額的 1/13 的部分，13 週才能完整拿到全部 COS。並且只能轉化自己的 VESTS 到自己的 COS 賬戶。每個用戶同時只能有一個 power down 請求進行。

3. 帳號系統

帳號名是Contentos區塊鏈帳號的唯一標識。有效的帳號名包含3～16個字符，每個字符可以是任意的小寫英文字母、阿拉伯數字或句點。長度和字符集的限制條件，幾乎不會增加用戶選取合適名稱的難度，但Contentos系統卻可以在工程上大幅降低存儲和提高名稱查找匹配效率。

3.1 權限和授權

Contentos帳號有3個內建的權限等級：Owner，Active，Posting。

- Owner權限：對帳號的絕對控制權，可以進行所有操作。
- Active權限：可以進行除修改帳號信息之外的所有操作。
- Posting權限：只可以進行社交操作，如發表內容或投票，不可以進行轉賬、買賣等影響帳號資產的操作。

內建權限各自對應一對公私鑰。這3對公私鑰在帳號創建時產生。用戶使用任何一把私鑰都可以成功登錄，擁有對應的權限等級。例如，使用Posting私鑰登錄後，只能發表社交內容，無法轉賬和更改帳號信息。根據PoLP（權限最小化原則），用戶使用合適權限的私鑰登錄，將極大降低帳號的安全風險。在絕大多數使用場景下，都不應該使用Owner私鑰登錄。

Contentos使用數字簽名進行權限鑑定。用戶發布一個transaction前，必須用自己的登錄私鑰對其簽名。 witness節點在收到transaction後，會首先驗證數字簽名是否合法，然後檢查簽名鑰匙是否具備執行這個transaction的權限等級。如果檢查不通過，transaction將被拒絕執行。

有時，用戶會有把自己的權限和一個或多個其他帳號分享——即授權的需求。例如，一個組織O的官方blog有多個維護人員，每人都可以以O的名義發佈內容。要實現這個群組blog功能，帳號O需要把它的posting權限授權給一組維護帳號。某個維護人員M發布一篇文章，聲明作者是O，並用自己的（M的）私鑰簽名。 Contentos將會檢查簽名鑰匙是否具備O的posting權限，由於它已經授權給M，所以M的簽名可以被接受。通過授權的方式，M在不知道O的私鑰的情況下，能夠以O的身份發佈內容。

Contentos帳號權限的授權是始終存在的，默認情況下，Owner/Active/Posting權限分別授權給了對應的帳號公鑰，即：只有用戶自己擁有這些權限。對於任何一種權限等級，用戶都可以更改授權信息，指定一個或多個授權對象。授權對像有2種：

- 其他帳號的公鑰
- 其他帳號的用戶名和權限

第2種情況有些複雜，但非常靈活。例如，user_a把自己的active權限授權給user_b@active，user_b又把自己的active權限授權給user_c@active。這樣就形成了一條授權鏈。使用active私鑰登錄的user_c擁有user_a的active權限，雖然user_a沒有直接授權給他。 Contentos支持授權鏈的鑑定，但出於性能考慮限制了授權鏈的最大長度。幸運的是，大多數使用場景不會用到或者只需要很短的授權鏈。

Contentos還支持對授權對象分配權重，用於更加複雜的授權需求。比如，admin的owner權限授權給3個帳號user_a, user_b, user_c，他們的權重都是0.5，即每個人都只有“一半”權限，需要2個人的共同簽名才能使用admin的owner權限。單個授權用戶的私鑰洩漏，並不會影響admin帳號的安全。這種多用戶簽名機制，能夠大幅降低授權引入的安全風險。

3.2 並行權限鑑定

Contentos的帳號授權管理，和基於數字簽名的權限鑑定機制，是與上層應用邏輯解耦的，並且是個“只讀”過程。即使有些transaction會更改授權信息，改動在block生成之後才能生效。因此，所有transaction的權限鑑定可以並行進行，不必依賴複雜的應用邏輯。而且，p2p節點可以在剛剛接收到transaction時就開始鑑權操作，這些transaction被正式打包到block後需要執行時，不必重新鑑權。

權限鑑定是一個計算密集的過程，需要較多的運算時間。並行化的權限鑑定能夠大幅提高Contentos整體的運行效率。

通過replay blocks重新生成區塊鏈狀態時，是可以省略權限鑑定的。因為一個合法block包含的所有transaction，在進入block時已經做過權限鑑定。這也可以顯著提升replay的速度。

3.3 帳號恢復

用戶的Owner私鑰被盜並被修改時，可以通過Contentos的帳號恢復功能重新找回帳號。用戶需要提供最近30天內有效的任何一個Owner私鑰，由指定的恢復輔助帳號批准並發起帳號恢復申請，來重置用戶帳號的Owner key。所有帳號都有自己的恢復輔助帳號，默認情況下，恢復輔助帳號是用戶帳號的創建者。用戶可以修改自己的恢復輔助帳號，一般設置成自己信任的好友帳號。

恢復輔助帳號不同於將Owner權限授權給其他帳號，因為恢復輔助帳號只能參與帳號恢復過程，它並沒有用戶帳號的任何權限，無法以用戶身份發起任何transaction。

4. 智能合約和虛擬機

技術背景

COS智能合約使用C/C++做為主要編程語言，底層採用WebAssembly虛擬機。WebAssembly(WASM)是一個已嶄露頭角的Web標準，受到Google, Microsoft, Apple及其他大公司的廣泛支持。目前為止，最成熟的用於構建應用及WASM代碼編譯的工具鍊是CLANG/LLVM及其C/C++編譯器。第三方開發的其他工具鏈包括：Rust，Python和Solidity。雖然這些其他語言看起來可能更簡單，但它們的性能可能會影響您可以構建的應用程序的規模。

合約能力

用戶可以把自己開發的合約預編譯成WASM字節碼部署到區塊鏈中，Contentos會提供一套API支持合約的外部調用。 COS智能合約提供類似以太坊ERC20規範下的開發能力，也可以通過系統提供的API進行發帖，點贊，數據統計等操作。

合約支持熱更新，可以幫助用戶快速修正BUG。

合約限制

考慮到用戶使用Contentos區塊鏈服務的公平性，每個用戶智能合約執行時長，使用CPU、網絡、內存資源都會被限制。

開發編譯

使用C/C++做為主要編程語言，我們會提供一套封裝好的COS C/C++ API支持合約的開發，這套API有更強大的類型安全性，並且更易於使用和閱讀。

提供合約編譯腳本，可以很方便的編譯合約、生成WASM字節碼文件和ABI接口文件。

安全機制

首先，合約是運行在一個沙箱化的執行環境中。並且WASM通過減少其語義中較為危險的特性並同時保持對C/C++在語法上的最大兼容性來保證一個較高的安全性和可用性。我們在合約運行前，預先對合約本身做嚴格的權限檢查和資源限制，更好的保障安全。

5. 虛擬機內容API

除了虛擬機自身的能力之外，contentos的虛擬機還提供足夠豐富的原生鏈的api來幫助內容開發者，這樣開發者可以通過編寫dApp來實現各種高級的功能。規劃中會開放的API包括但不限於以下方面：

- 用戶在鏈上的任何raw操作，譬如：獲取某個交易的具體內容
- 鏈上產生的虛擬操作數據，譬如：得到某個時段內witness獲得的獎勵值
- 向native鏈提交寫操作，譬如：進行發帖或者進行點贊
- 合約之間相互交互的操作，譬如：根據另外一個合約的執行結果來決定本合約的執行流程

示例

舉一個具體的例子：譬如推廣方A需要宣傳某款物品，進而和網紅B進行合作，和B約定，7天之內如果B創作的宣傳視頻得到10000人點贊，那麼A會給予B 100個COS作為獎勵。這樣一個流程就可以採用合約API很好的進行實現：

1. 首先A部署合約，將100個COS質押到合約中
2. 當B發完視頻後觸發合約啟動
3. 7天過後，合約自動執行，通過內容api獲取到視頻的點贊人數，如果結果符合協議，那麼調用轉賬API將COS轉入B的賬戶中，如果不符合協議，則根據合約程序返還給A賬戶

6. 合約模版

目前大部分公鏈在提供合約功能的同時並沒有很好的合約模版和自動化部署合約的功能，這使得普通用戶根本無法享受到合約帶來的好處。 Contentos公鏈在這個方面會實現的非常易用，公鏈自身會集成幾十種常用的合約，並且為這些合約開發易用的API接口。 Dapp的開發者可以提交模版集成建議，在被開發團隊採納後會直接更新到公鏈的核心代碼中。