ブロックチェーン上のアプリケーション移植性に向けて

首藤 一幸　神田 伶樹　斉藤 賢爾

本稿で我々は，経済的なインセンティブに支えられたパブリックブロックチェーンが抱える根本的な問題，**インセンティブ不整合**を論じる．これは未解決問題であるが，アプリケーション移植性が暫定解となる．移植性はプライベートブロックチェーンのアプリケーションにとっても望ましい性質である．本論文で，アプリケーション移植性，特にブロックチェーン間移行をサポートするミドルウェアの設計を示す．

We discuss the issue of what we call *incentive mismatch*, a fundamental problem with public blockchains supported by economic incentives. This is an open problem, but one potential solution is to make application portable. Portability is desirable for applications on private blockchains. Then, we present examples of middleware designs that enable application portability and, in particular, support migration between blockchains.

1　はじめに

ブロックチェーンは、政府といった中央的な存在なしに（図 1）ある種の信頼を提供し，その応用は暗号通貨以外にも広がりつつある．

インセンティブ不整合は，パブリックブロックチェーンが抱える根本的な問題である．ブロックチェーンを支えるノード群のインセンティブは経済的なもの，つまりコインの取得であり，ブロックチェーン上アプリケーションのそれとは異なる．経済的な魅力が消失してブロックチェーンが機能不全に陥った場合，その上のアプリケーションも機能不全に陥る．ブロックチェーンとアプリケーションのインセンティブを揃えることは容易ではなく，未解決の課題である．

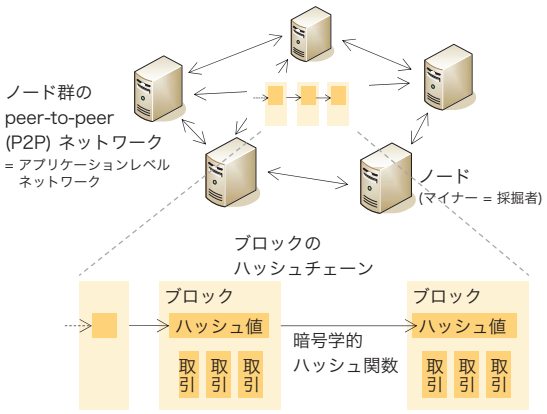

図 1　分散システムとして見たブロックチェーン

それでも，もしアプリケーションに移植性があれば，ブロックチェーンの機能不全に際してもアプリケーションは守ることができる．移植性は，プライベートブロックチェーン上のアプリケーションにとっても望ましい性質である．ミドルウェアへの依存を最小限とすることは知られたベストプラクティスの 1 つ

Towards Application Portability on Blockchains.

Kazuyuki Shudo, Reiki Kanda, 東京工業大学 情報理工学院 数理・計算科学系, Dept. of Mathematical and Computing Science, Tokyo Institute of Technology.
Kenji Saito, 慶應義塾大学 SFC 研究所, Keio Research Institute at SFC, Keio University.

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

であろう．

しかし，今日，各ブロックチェーンミドルウェアはそれぞれ固有の API を提供しており，あるミドルウェア向けに書かれたアプリケーションは他のミドルウェアでは動作しない．各ブロックチェーンは，例えば Ethereum の Solidity 言語のように，独自の抽象化を行っているため，機能する共通 API を策定できるかどうかも定かではない．

本論文にて我々は，インセンティブ不整合問題を提起し，暫定解としてアプリケーション移植性を論じる．続いて，アプリケーションのミドルウェア間移行を可能とするソフトウェアアーキテクチャと技法を述べる．

本論文は我々の以前の論文 [10] の日本語版である．

2 アプリケーション移植性の恩恵

移植性は，パブリックとプライベートブロックチェーン，双方のアプリケーションにとって，それぞれ理由は異なれど，望ましい性質である．

2.1 パブリックブロックチェーンのアプリケーション

パブリックブロックチェーンは我々が**インセンティブ不整合**と呼ぶ根本的な問題を抱えている．ブロックチェーンを支えるノード群とブロックチェーン上のアプリケーションに，共通の動機がないのである．パブリックブロックチェーンでは，トランザクションの承認は Proof of Work [2] やその派生形である Proof of Stake [8]，Proof of Elapsed Time [5] 等によって達成されており，そうした機構はノード群の経済的な動機に基づく．ノード群はただコインを稼ごうとしているだけであり，アプリケーションを支えるという動機は持たない．

パブリックブロックチェーンは，ノード群に対して充分な経済的動機を与えそこねると，トランザクションを安全に承認することができなくなる．例えば，コイン価値の低下はノード群の離脱を招き，安全な承認を行う能力が低下する．ブロックチェーンは，支えるノードが少ないほど，majority (51%) attack や eclipse attack [4] [6] といった攻撃に対して脆弱になる．2018 年 5 月には実際に，暗号通貨を支えるパブリックブロックチェーンへの攻撃が立て続けに成功した．Monacoin の場合，攻撃者は，block withholding 攻撃によって，一度は承認された 20 ブロック以上を無効化した．Bitcoin Gold の場合，攻撃者は，22 ブロック以上を無効化し，コインの二重使用に成功した．これは，ブロックチェーン上のアプリケーションは，その基礎となる承認能力に対して何の制御もできず，経済環境が原因で機能不全に陥り得るということである．

ブロックチェーンを支えるノード群とブロックチェーン上アプリケーションの動機を揃えることは可能だろうか？あるパブリックブロックチェーンを支える貨幣システムは，それ自体が，ブロックチェーンの承認能力を活用するアプリケーションである．この場合，動機は整合しているが，しかし，これは特殊ケースである．同一の人や組織がブロックチェーンノード群とアプリケーションの両方を配備，運用すれば，動機は整合する．こうなると，それはプライベートブロックチェーンである．

さもなくば，パブリックブロックチェーンとその上のアプリケーションは一般には共通の動機は持たない．それらをどう整合させるか，換言すると，アプリケーション群が自ら承認能力を支える機構を作れるのか否か，は未解決問題である．アプリケーション移植性は，この問題への暫定解となる．もし，移植性，特に移行可能性（3 章）があれば，ブロックチェーンが機能不全に陥りそうな状況でも，アプリケーションを他のブロックチェーンに移行できる．

2.2 プライベートブロックチェーンのアプリケーション

移植性は，プライベートブロックチェーン上のアプリケーションにとっても望ましい性質である．移植性が低いと，いわゆるベンダロックインを被ることとなる．例えば，アプリケーションとその利用者は，より良い他のミドルウェアの恩恵を受けられない．悪いと，ミドルウェアと共にアプリケーションも死んでしまう．ここで言う死とは，古さが原因でミドルウェアを実行できくなる，という意味である．

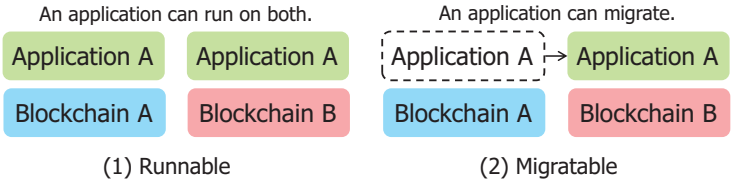

図 2　アプリケーション移植性の 2 レベル

3　アプリケーション移植性

アプリケーション移植性にはいくつかのレベルがある．以降，次の 2 レベル（図 2）を区別して扱う．

1. “**動作可能**（*runnable*）”：　アプリケーションは異なるブロックチェーンの上で動作し得る．あるブロックチェーンから他への移行はできない．
2. “**移行可能**（*migratable*）”：　アプリケーションとそのデータをあるブロックチェーンから異なるブロックチェーンへ移行できる．

ブロックチェーン群に共通する API があれば，“**動作可能**” レベルの移植性が達成される．また，同一のミドルウェアで運用される別ブロックチェーンの間で移植性があることは自明であろう．一方，異種ミドルウェアに対して，機能する共通 API を策定できるかどうかは定かではない．各ミドルウェアはそれぞれの抽象化を採用しているからである．例えば，あるミドルウェアはブロックのハッシュチェーンの代わりに有向非巡回グラフ（DAG）を採用しているし，各ミドルウェアはそれぞれ自身のスマートコントラクト機構—例えば Ethereum の Solidity 言語—を持っている．しかし，もし，アプリケーションが利用するブロックチェーンの機能を限定し，限定した機能に対して共通 API を用意できたなら，移植性を達成できる．どのようなブロックチェーンもハッシュ値をタイムスタンプ付きで格納する程度の機能は持っており，このための共通 API くらいは用意できる．

移植性が “**動作可能**” レベルにとどまる場合，別ブロックチェーンへ移行するアプリケーションはその初期状態での再起動しかできず，蓄積したデータは捨てざるを得ない．データの存在証明と状態変化の検証はブロックチェーンの特徴的な機能であり，それを可能にするログ（4.1 節）を伴わない移行ではほとんど意味がない．“**移行可能**” レベルの移植性が，プライベートブロックチェーンミドルウェアの喪失（2.2 節）やパブリックブロックチェーンの機能不全（2.1 節）を乗り越えてのアプリケーションの生存を可能にする．

4　アプリケーションのブロックチェーン間移行

本章は，どのようにしてアプリケーションのブロックチェーン間移行を達成するかを論じ，移行を可能にするミドルウェアの設計案を示す．

4.1　移行対象データ

どういったデータをブロックチェーン間移行の対象とすべきか？アプリケーションがブロックチェーンに依存しないものであって，ミドルウェア間で移行する場合，たいていは，アプリケーションの最新状態，例えば関係データベース中のレコード，さえ移行すれば充分である．しかし，ブロックチェーンを採用する強い動機の 1 つは，過去のある時点であるデータが存在したことを証明できることや，状態（データ）の変化を検証できること，である．この場合，最新状態だけを移行したのでは不足である．存在証明や検証に必要なログの移行も必要である．

つまり，移行の対象となるデータには次の 2 つのクラスがある．

1. アプリケーションの**最新状態**
2. **ログ** – 状態についてのメタデータであり，状態変化とそのタイムスタンプから成る．

例えば，Bitcoin のような貨幣アプリケーションでは，前者は口座残高である．もっとも，これは，Bitcoin では明には記録されず関連するトランザクションを集計することで算出できる．後者はトランザクションで

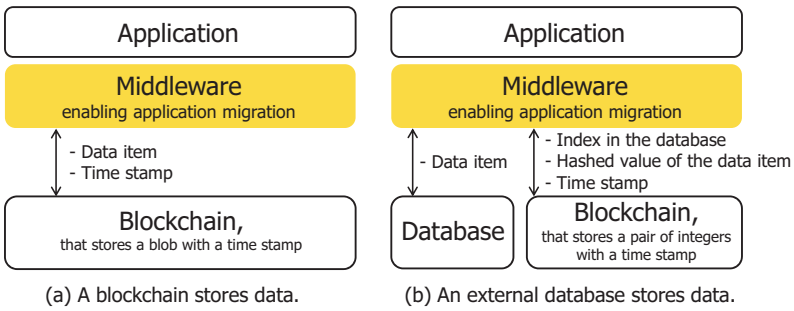

図 3　ブロックチェーンへの依存を制限することでアプリケーション移植性を向上させるソフトウェアアーキテクチャ

あり，口座残高の変化とタイムスタンプから成る．

4.2　移行を可能とするミドルウェア設計

移行を可能にするミドルウェアを設計するにあたり，次の方針を採る．

- 特定のブロックチェーンミドルウェアへの依存を最小限にする．
  単純なデータ，つまり数値の組やバイト列等をタイムスタンプと共に記録する機能だけを前提とする．
- 移行元ブロックチェーンを信用し続けられるとは期待しない．
  あるブロックチェーンから脱出しようという理由の 1 つは，そのブロックチェーンの差し迫った機能不全である．例えば，パブリックブロックチェーンは，ノードを失い過ぎるとそうした機能不全に陥る（2.1 節）．この場合，そのブロックチェーンに依拠し続けることはできない．

図 3 に設計の候補を示す．図中の “middleware” がブロックチェーン間移行を行う．上述の依存を最小限にするため，設計案では，ブロックチェーンにはいくつかの値とタイムスタンプの組を検索キー（例：Bitcoin アドレスといったアカウント ID）とともに記録する機能しか求めない．それ以上を求める場合も，せいぜいブロブ（バイト列）の記録までとする．それ以上の機能をサポートしつつの移行は今後の課題である．

移行のために必要ではないものの，図 3 (b) のように，データはブロックチェーン内ではなく外部のデータベースに保存する，という設計もあり得る．ブロックチェーン上ミドルウェア Beyond Blockchain One (BBc-1) [9] がこうした設計を採っている．この設計はブロックチェーンのサイズを削減する．ブロックチェーンは全ブロックチェーンノードにコピーされて，全ノード上のストレージを占有するため，サイズの削減にはメリットがある．加えて，もしアプリケーションが状態（データ）変化の検証までは必要としないという場合，データベース上のデータをただ上書きすれば済み，ストレージ専有量はさらに減る．

(1) 動作可能（Runnable）
共通 API および共通言語によって、可能となる

(2) 移行可能（Migratable）
状態（データ）変化の検証まで
(2-1) 必要
図3 (a): 移行元ブロックチェーンを保存し、維持する必要がある
図3 (b): 更新時にデータの全バージョンを保存していく必要がある
(2-2) 不要
図3 (a): データの存在証明に必要なメタデータをコピーした後、移行元ブロックチェーンは捨てることができる
図3 (b): データは上書きしてよい

図 4　アプリケーションの移行を可能にするブロックチェーンの設計選択肢

図 3 (b) の設計では，充分なデータ可用性を達成するためには，データベースの耐故障性が充分に高い必要がある．このためには，たいていの分散データベースがサポートしている複製や消失訂正符号（erasure coding） [3] を用いることができる．耐故障性の程度は，複製数や消失訂正符号のパラメータを設定することで調整できる．このように調整できる柔軟性は，全ブロックチェーンノードへの単純なコピー（図 3 (a)）

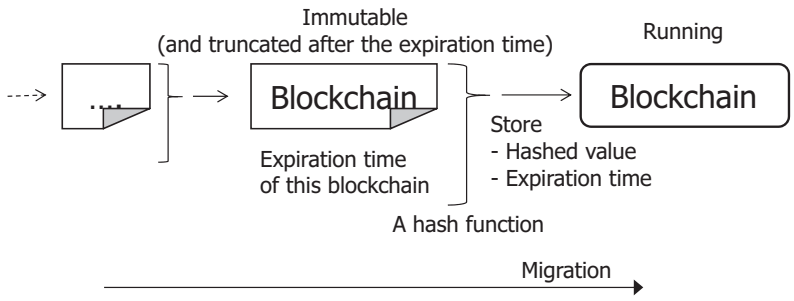

図 5　ブロックチェーン間でアプリケーションを移行させる方法

に対する利点であろう．現在，そうした可用性の高いデータベースがパブリッククラウドのサービスとして提供されており，さらには我々自身でクラウド提供者をまたいで複数のデータベースに記録することもできる．

図 3 (b) の設計では，アプリケーションが状態（データ）変化の検証までは必要としない場合は，更新時にデータをデータベース上で上書きしてしまうこともできる．これによって，ストレージの占有量をさらに減らすことができる．

図 4 に，アプリケーションの移行を可能にするブロックチェーン設計の選択肢をまとめる．

4.3　移行の手順

アプリケーションが移行元ブロックチェーン中のログ（4.1 節）へのアクセスを必要とするなら，ミドルウェアはアクセスを提供する必要がある．しかし我々は，セキュリティ的な理由から，移行元ブロックチェーンを信用し続けられると期待するわけにはいかない（4.2 節）．さらに，移行元ブロックチェーンのミドルウェアが実行を続けていてアクセスできるとは期待しない方が安全である．なぜなら，一般にパブリックブロックチェーンを制御できないのであるし，プライベートブロックチェーンのミドルウェアを実行させ続けるためには手間と CPU 能力が要る．つまり，移行元ブロックチェーンを信用することなく，実行が続いていることを仮定することなく，その中のログにアクセスする方法が必要である．

図 5 はこれらの要求を満たす移行手順である．ミドルウェアは，移行元ブロックチェーンの静的なコピーを自身のストレージに保存し，有効期限以降を切り捨て，それをアクセスする．ミドルウェアは，移行直前の時刻を移行元ブロックチェーンの有効期限と設定する．タイムスタンプはある程度操作できてしまうので，有効期限はブロック高で指定することとなろう．静的保存によって，ブロックチェーンミドルウェア実行の必要性をなくせる．切り捨てによって，有効期限以降にそのブロックチェーンを信用する必要をなくせる．

ミドルウェアは，図 5 の通り，保存されたブロックチェーン群からハッシュチェーンを構成する．ハッシュチェーンによって，我々は保存されたブロックチェーンへの改ざんを検知でき，改ざんされていないことを信じられる．ハッシュチェーンによって，次のブロックチェーンに格納された有効期限が正しいことの確認もできる．

ここで，パブリックブロックチェーンではいったん承認されたブロックも無効化される可能性があることを思い出さねばならない．それは，攻撃（2.1 章）によっても，また，自然にも発生する．そのブロックチェーン上のアプリケーションは，そうした無効化を単に無視できるように思われる．なぜなら，アプリケーションは，実行を続けている古いブロックチェーンではなく，静的に保存したブロックチェーン群にのみ依拠するからである．とはいえ，そうしたブロック無効化を考慮せねばならない状況もあるかもしれない．その場合，静的に保存したブロックチェーンも更新する必要がある．最低でも、保存したブロックチェーンの有効期限を，未だ有効なブロックまで巻き戻す必要はある．ともあれ，そうした更新は時間を食

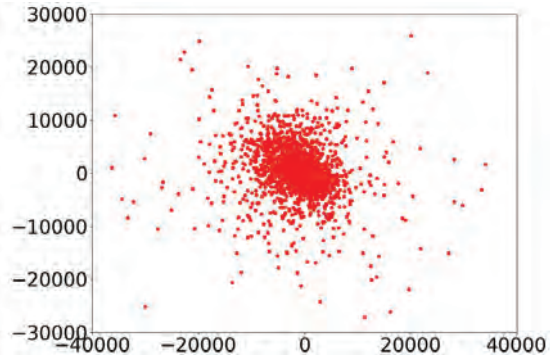

図 6　Bitcoin の約 10,000 ノードを表したネットワーク座標系（単位はミリ秒）

い，その間アプリケーションを止めてしまうので，避けたい．我々は，自然発生するブロック無効化の原因である，ノード間ブロック遅延を減らそうと試みている．削減のための手法は，隣接ノード選択か経路選択 [7] となるだろう．そうした，性能のための隣接ノード選択や経路選択には，ノード間の近接性情報が必要である．そのために我々はまず，Vivaldi [1] といったネットワーク座標系を用いてノード間の通信遅延を見積もる手法を確立しようと試みている．図 6 は，2 次元ユークリッド平面上で Vivaldi を用いて推測した Bitcoin ノード間の通信遅延である．この取り組みにおける次の一歩は，精度の向上である．

このミドルウェアはすべての移行元ブロックチェーンを解釈できる必要がある．それらブロックチェーンは，それぞれが自身の形式を持っているだろう．そうした解釈機能の開発は大きな労力を要し，また，ソフトウェアは複雑で多くの誤りを含みがちとなる．もしアプリケーションが許すなら，移行元ブロックチェーンは単純に捨てることが望ましい．アプリケーションが，存在証明は必要とするけれど状態（データ）変化の検証までは必要としないという場合，存在を証明するメタデータ（と図 3 (a) ならデータ自体）を次のブロックチェーンにコピーし，移行元ブロックチェーンは捨てることができる．この場合，コピー元メタデータなしにコピー先メタデータが正しいことをどう信じたらよいだろうか？ブロックチェーンが提供する信用は，データが検証可能であることに基礎を置く．しかし，このコピーを行う設計では，コピー先メタデータは元データと明示的な関係を持たず，検証ができない．アプリケーションはそもそもコピーを行うミドルウェアを信用しているという前提ではあるが，そうしたソフトウェアの正しさに依拠する信用は検証可能データによる信用より弱い．なぜなら，例えばソフトウェアは人的誤りを含むからである．次善の策は，コピーを行うプログラムの検証であろう．コピーを行うプログラムとそのハッシュ値をどこかに保存しておくことで，検証可能にできるだろう．

5　まとめ

ブロックチェーンのアプリケーションには，通常のアプリケーションと同様に移植性への動機がある一方，パブリックブロックチェーン上ではインセンティブ不整合という特有の動機もある．つまり，パブリックブロックチェーンはアプリケーションとは異なる経済的なインセンティブで駆動されており，アプリケーションから制御できるものではない．そのため，移植性，もっと言うと移行可能性がない限り，ブロックチェーンが機能不全になるとアプリケーションも機能不全に陥ってしまう．

ブロックチェーンアプリケーションの生存能力を高めるべく，移植性と移行について論じた．ブロックチェーン間移行を可能にするミドルウェアの設計を示し，要求の少ないアプリケーション向けにはより効率的な代替案を示した．

アプリケーション移植性と移行は，一般に望ましい機能ではあると同時に，パブリックブロックチェーンのインセンティブ不整合問題に対する暫定解となる．それでも，ブロックチェーンは，ノードの経済的インセンティブではなく，アプリケーション群自身によって支えられることが望ましい．もし，パブリックブロックチェーンが常にアプリケーション群だけに支えられるなら，それは**インセンティブ整合ブロックチェーン**であると言える．しかし，そうした機構をどう作るかは未解決問題である．

ISO 等で分散台帳分野の標準化が始まっている．IETF や W3C もすでに議論を持った．機能する共通 API の策定でさえ挑戦的であり，ブロックチェー

ン間移行はそのさらに先にある．今日，まずソフトウェアが実装され，仕様書の作成はソフトウェアが多大な人気を得た後で行われる．実際，最初の Bitcoin Improvement Proposal（BIP）は Bitcoin ネットワークの立ち上げから 2 年半以上後だった．本論文に続く実効性のある次の一歩は，移植性のある，もしくは，移行可能なアプリケーションのデモンストレーションだろう．

あまり論じなかったものの，スマートコントラクトのプログラムコードとデータの移植性も今後の課題である．同一ミドルウェア上でさえ，既存データを維持したままでのデータ形式やプログラムコードの更新は困難である．

謝辞

インセンティブ不整合問題，インセンティブ整合した将来のパブリックブロックチェーンなど多くのトピックを議論した竹井悠人氏，北條真史氏，鈴木茂哉氏に感謝します．プライベートブロックチェーンにおける移植性の意義を指摘してくれた小出俊夫氏に感謝します．本研究はカウラ (株) の支援を受けたものです．本研究は（公財）セコム科学技術振興財団 一般研究助成の支援を受けたものです．本研究の一部は，国立研究開発法人新エネルギー・産業技術総合開発機構（NEDO）の委託業務として行われました．本研究は JSPS 科研費 25700008 および 16K12406 の助成を受けたものです．

参 考 文 献

[1] Dabek, F., Cox, R., Kaashoek, F., and Morris, R.: Vivaldi: A Decentralized Network Coordinate System, *Proc. ACM SIGCOMM 2004*, September 2004.

[2] Dwork, C. and Naor, M.: Pricing via Processing or Combatting Junk Mail, *Proc. CRYPTO'92*, August 1992, pp. 139–147.

[3] Ford, D., Labelle, F., Popovici, F. I., Stokely, M., Truong, V.-A., Barroso, L., Grimes, C., and Quinlan, S.: Availability in Globally Distributed Storage Systems, *Proc. OSDI'10*, October 2010.

[4] Gervais, A., Ritzdorf, H., Karame, G. O., and Čapkun, S.: Tampering with the Delivery of Blocks and Transactions in Bitcoin, *Proc. ACM CCS 2015*, October 2015.

[5] : Hyperledger Sawtooth. https://sawtooth.hyperledger.org/.

[6] Heilman, E., Kendler, A., Zohar, A., and Goldberg, S.: Eclipse Attacks on Bitcoin's Peer-to-Peer Network, *Proc. USENIX SEC'15*, August 2015, pp. 129–144.

[7] Miyao, T., Nagao, H., and Shudo, K.: A Method for Designing Proximity-aware Routing Algorithms for Structured Overlays, *Proc. IEEE ISCC'13*, 2013.

[8] QuantumMechanic: Proof of stake instead of proof of work. https://bitcointalk.org/?topic=27787.0.

[9] Saito, K. and Kubo, T.: BBc-1 : Beyond Blockchain One – An Architecture for Promise-Fixation Device in the Air –. Available electronically at https://beyond-blockchain.org/public/bbc1-design-paper.pdf.

[10] Shudo, K., Kanda, R., and Saito, K.: Towards Application Portability on Blockchains, *Proc. IEEE HotICN 2018*, August 2018.