

# How to use BBc-1

revision 1

for v1.0

21 May 2018

# 本資料について

- BBC-1の環境構築、利用方法についてまとめた
  - 対象のBBC-1はgithubに公開されたv1.0 (2018/5/1バージョン) である
    - <https://github.com/beyond-blockchain/bbc1>
  - BBC-1のWhitePaper、YellowPaper (Analysys)はgithubリポジトリのdocs/、および下記URLに公開されている
    - <https://beyond-blockchain.org/public/bbc1-design-paper.pdf>
    - <https://beyond-blockchain.org/public/bbc1-analysis.pdf>
  - 本資料の内容に起因するあらゆるトラブルには責任を負わない
- 作成日：2018/5/21
- 作成者：takeshi@quvox.net (t-kubo@zettant.com)

# Collaborators' github account

- junkurihara
- imony
- ks91

# Change log

- 2018/5/21: 初版

# 目次

| タイトル                    | ページ |
|-------------------------|-----|
| システム構成                  | 6   |
| システムの起動と設定              | 11  |
| 起動方法                    | 14  |
| 設定ツール                   | 17  |
| ネットワーク設定                | 32  |
| 履歴交差のためのdomain_global_0 | 37  |

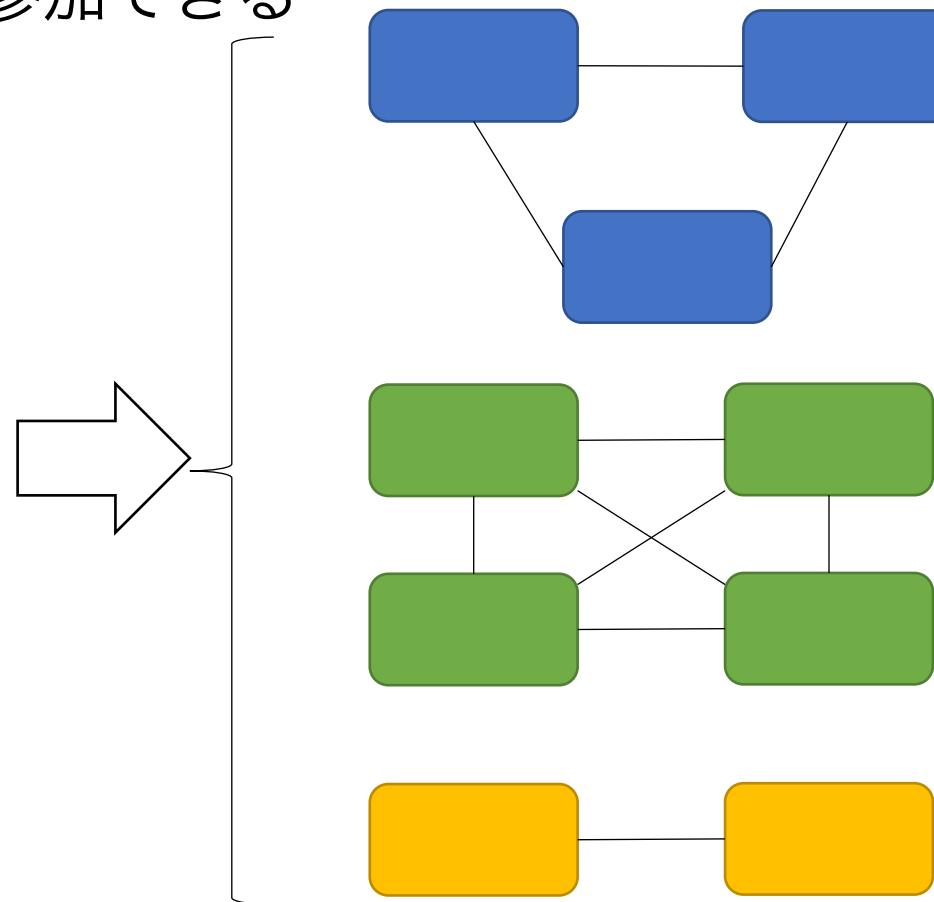
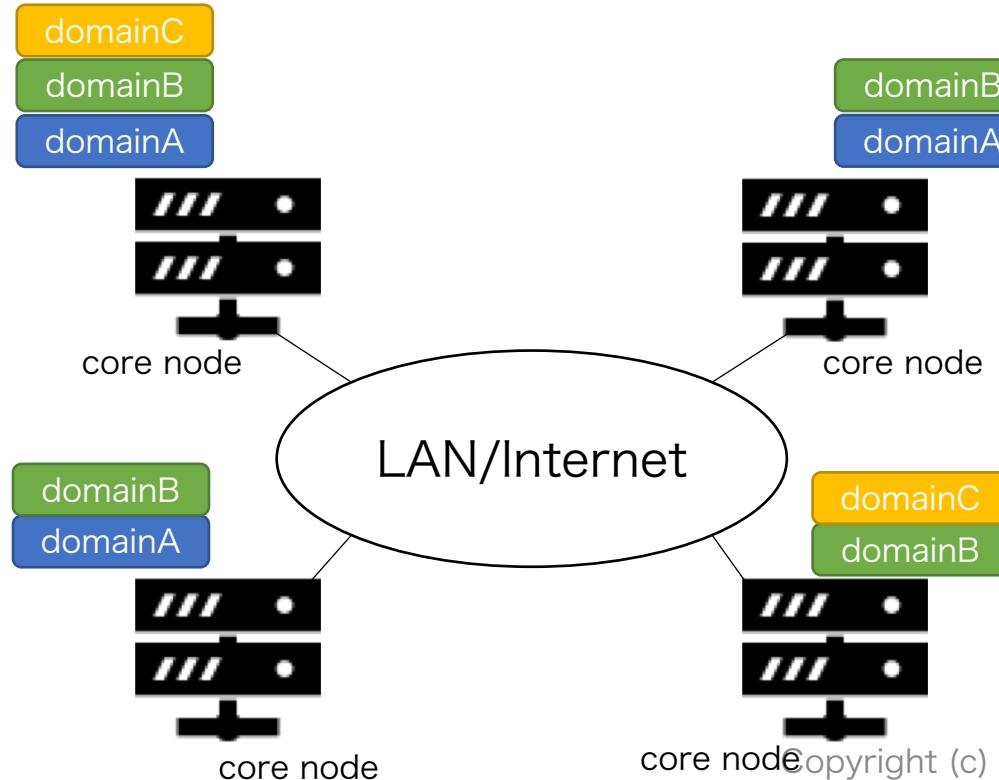
# システム構成

# BBc-1システム構築・利用時の流れ

- 登場人物
  - コアノード提供者（≒システム管理者）
  - サービス開発・提供者
  - アプリ利用者
- 流れ
  1. BBc-1の環境を準備する (by コアノード提供者)
  2. コアノード群に対してdomainを定義する (by サービス開発・提供者)
  3. 外部コンピュータ（またはコアノード）にアプリケーションを開発し、コアノードに接続させる (by サービス開発・提供者)
  4. アプリケーションを利用しアセットの登録や移転、検索、取得を行う (アプリ利用者)

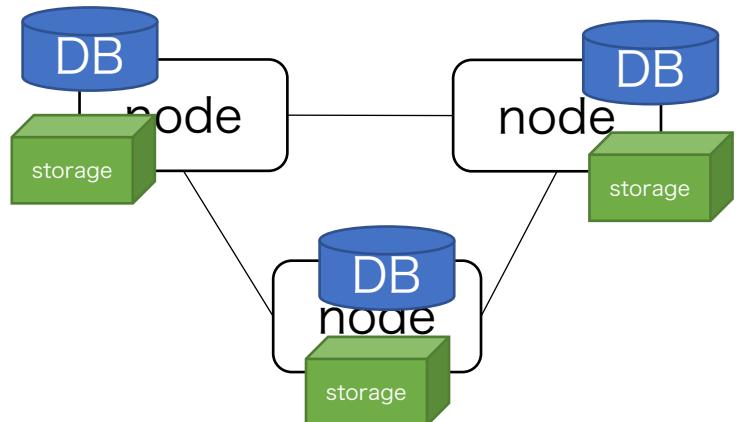
# BBc-1 ネットワークの論理構成

- 物理core nodeは複数のdomainに参加できる

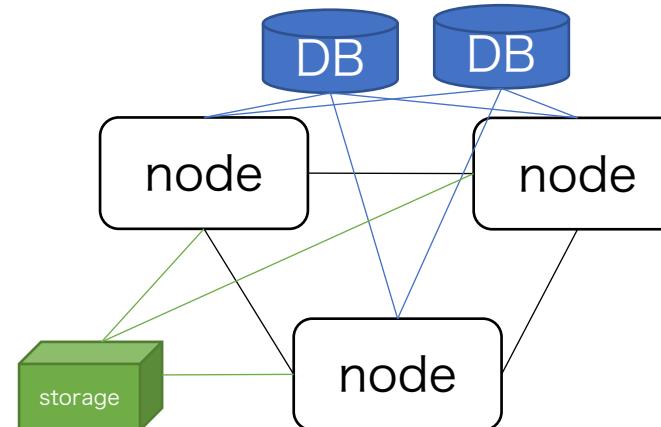


# domain内のシステム構成

- ・ネットワークトポロジ
  - ・フルメッシュ（参加ノードが、他の全ての参加ノードを把握する）
- ・トランザクション等を保持するためのDB
  - ・参加ノード全員がそれぞれDBを持つ/外部のDBを指定する
- ・アセットファイルを保存するためのストレージ
  - ・参加ノード全員がそれぞれストレージを提供する/外部のストレージを指定する



DBもストレージもcore nodeが持つパターン

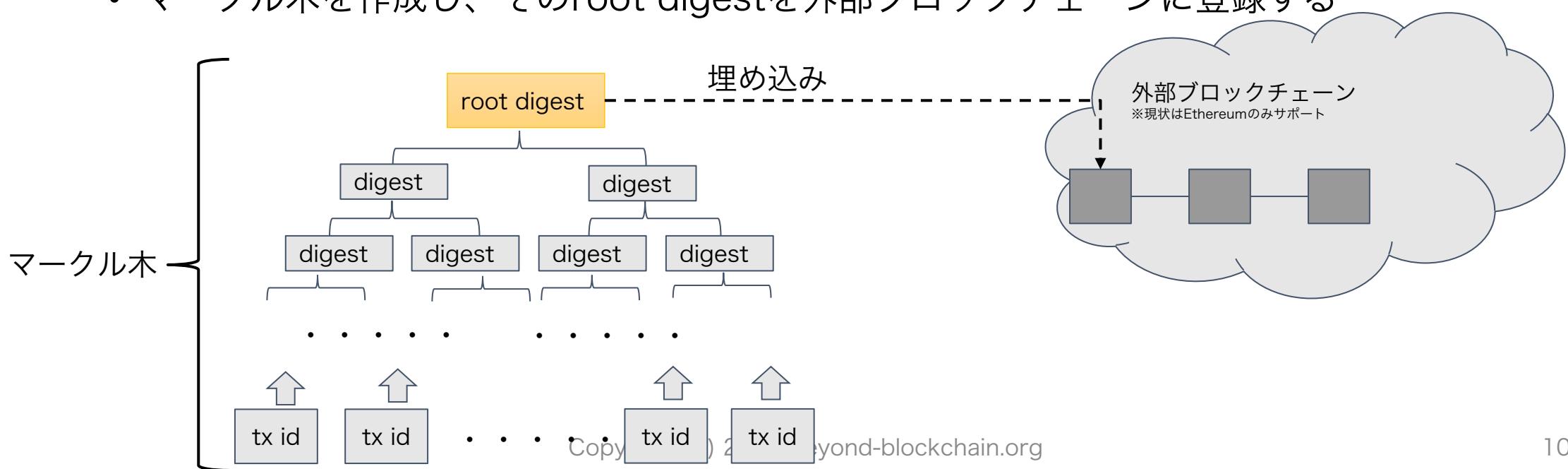


DBもストレージも外部に配置するパターン

※DBとストレージはそれぞれ  
独立に配置場所を選べる

# 外部ブロックチェーンへのアンカリング

- ・アンカリングの目的
  - ・トランザクションがすでに登録されていることを、外部のブロックチェーンにも記録することで、存在の有無を検証できるようにする
- ・方法
  - ・core nodeが
  - ・マーカル木を作成し、そのroot digestを外部ブロックチェーンに登録する



# システムの起動と設定

# コマンドとツール群

| スクリプト名               | 説明   |
|----------------------|--|
| bbc_core.py          | core node本体                                    |
| bbc_domain_config.py | コンフィグファイル作成/更新ツール                              |
| bbc_domain_update.py | bbc_core.pyプロセスへのコンフィグファイル更新通知ツール              |
| domain_key_setup.py  | domain_keyの生成/通知のためのツール                        |
| bbc_info.py          | bbc_core.pyプロセスの情報取得ツール                        |
| bbc_ping.py          | 隣接core nodeへの送達確認ツール（確認できれば隣接nodeとして認識するようになる） |
| id_create.py         | 256bitのIDのHEX文字列を生成し表示するツール                    |

全てのツールは”-hオプション”を指定することでヘルプが表示される  
以下では、bbc\_info.pyとid\_create.py以外の使い方を説明する

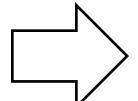
# 2種類の秘密鍵

- core nodeは、通信相手を認証するために2種類の暗号鍵を用いる
- domain\_key (core node同士の認証)
  - core node同士が通信をする際に、その相手が正しいcore nodeであることを示すために、運用者がdomain\_keyを作成し、bbc\_core.pyに読み込ませる
  - domain\_keyはドメインごとに作成する
  - core nodeが他のcore nodeからメッセージを受信すると、正しいdomain\_keyで署名されているかを確認し、不正であればメッセージを破棄する
  - 全てのメッセージに署名を付与するのではなく、ネットワークを構成するためのメッセージにのみ署名付与を義務付ける
- node\_key (クライアントとcore node間の認証)
  - core nodeが生成する
  - core nodeに対してドメイン作成などの管理コマンドをクライアントから投入する際に、メッセージにnode\_keyによる署名を付与しなければならない
  - デフォルト設定では、node\_keyを利用する

# コアノードの起動 (bbc\_core.py)

- bbc\_core.py : BBC-1プロセスとして起動すべきプログラム
  - デフォルトではbbc\_core.pyを起動したディレクトリに.bbc1/ というワーキングディレクトリが作成される (起動引数でパスを変更可能)
  - コンフィグファイルが存在しなければ、デフォルトコンフィグが生成される
- ワーキングディレクトリの中身
  - コンフィグファイル (config.json)
  - node\_key.pem
    - node\_key本体
  - domainごとのワーキングディレクトリ
    - DBファイル (トランザクション等保存用にSQLite3を利用する場合)
    - asset\_groupごとのディレクトリ
      - assetファイル群がここに格納される

BBC-1システムを動作  
させるために必要な作業



- configファイルの作成・更新
- 鍵の設定
- core node間のフルメッシュトポロジの構成

# コアノードの起動（起動オプション）

```
usage: python ../../bbc1/core/command.py [--coreport <number>] [--p2pport <number>] [--workingdir <dir>] [--config <filename>] [--nodekey] [--no_nodekey] [--domain0] [--ledgersubsystem] [--ip4addr <IP addr>] [--ip6addr <IPv6 addr>] [--log <filename>] [--verbose_level <string>] [--daemon] [--kill] [--help]

optional arguments:
  -h, --help            show this help message and exit
  -cp COREPORT, --coreport COREPORT
                        waiting TCP port
  -pp P2PPORT, --p2pport P2PPORT
                        waiting TCP port
  -w WORKINGDIR, --workingdir WORKINGDIR
                        working directory name
  -c CONFIG, --config CONFIG
                        config file name
  --nodekey             use node_key for admin command
  --no_nodekey          don't use node_key for admin command
  --domain0             connect to domain_global_0
  --ledgersubsystem    use ledger_subsystem
  --ip4addr IP4ADDR    IPv4 address exposed to the external network
  --ip6addr IP6ADDR    IPv6 address exposed to the external network
  -l LOG, --log LOG    log filename/"-" means STDOUT
  -d, --daemon          run in background
  -k, --kill             kill the daemon
  -v VERBOSE_LEVEL, --verbose_level VERBOSE_LEVEL
                        log level all/debug/info/warning/error/critical/none
```

- `--nodekey`を指定すると、`node_key`を利用する
- `--no_nodekey`を指定すると、`node_key`を利用しない
  - 上記2つは、コンフィグファイルにもその設定が自動で書き出される
- どちらも指定しなければ、コンフィグファイルの設定に従う

# コアノードの起動（起動オプション）

```
usage: python ../../bbc1/core/command.py [--coreport <number>] [--p2pport <number>] [--workingdir <dir>] [--config <filename>] [--nodekey] [--no_nodekey] [--domain0] [--ledgersubsystem] [--ip4addr <IP addr>] [--ip6addr <IPv6 addr>] [--log <filename>] [--verbose_level <string>] [--daemon] [--kill] [--help]

optional arguments:
  -h, --help            show this help message and exit
  -cp COREPORT, --coreport COREPORT
                        waiting TCP port
  -pp P2PPORT, --p2pport P2PPORT
                        waiting TCP port
  -w WORKINGDIR, --workingdir WORKINGDIR
                        working directory name
  -c CONFIG, --config CONFIG
                        config file name
  --nodekey             use node_key for admin command
  --no_nodekey          don't use node_key for admin command
  --domain0             connect to domain_global_0
  --ledgersubsystem    use ledger_subsystem
  --ip4addr IP4ADDR    IPv4 address exposed to the external network
  --ip6addr IP6ADDR    IPv6 address exposed to the external network
  -l LOG, --log LOG    log filename/"-" means STDOUT
  -d, --daemon          run in background
  -k, --kill             kill the daemon
  -v VERBOSE_LEVEL, --verbose_level VERBOSE_LEVEL
                        log level all/debug/info/warning/error/critical/none
```

- --domain0を指定すると、domain\_global\_0にも接続する

# 設定の流れ

- 初期設定
  - `bbc_domain_config.py`でコンフィグファイルを作成/更新する（テキストエディタで編集しても良い）
    - デフォルトコンフィグであれば、`bbc_core.py`を起動するだけでも作成される
    - IDの値は`id_create.py`を使えば簡単に出せる
  - `domain_key_setup.py`で`domain_key`を生成し、コンフィグファイルの`domain_key→directory`で指定したディレクトリに格納する
  - 生成した`domain_key`を他のノードにも配布する
- コンフィグの更新
  - `bbc_core.py`が起動していない場合
    - `bbc_domain_config.py`でコンフィグファイルを更新する（テキストエディタで編集しても良い）
    - `bbc_core.py`を起動する
  - `bbc_core.py`がすでに起動している場合
    - `bbc_domain_config.py`でコンフィグファイルを更新する（テキストエディタで編集しても良い）
    - `bbc_domain_update.py`でコンフィグが更新されたことを`bbc_core.py`プロセスに通知する
      - ドメインの新規作成と削除のみ、`bbc_core.py`起動中も可能
      - すでに設定済みのドメインのコンフィギュレーションを変更する場合は、`bbc_core.py`プロセスの再起動が必要
- `domain_key`の更新
  - `domain_key_setup.py`で`domain_key`の生成して、他のノードにも配布する（格納するディレクトリは上記と同じ）
    - 鍵のファイル名は同じなので上書きして良い
  - `bbc_core.py`が起動中なら`domain_key_setup.py -n`で`domain_key`の更新をcore nodeに伝える
    - `bbc_core.py`が起動していないければ、起動するだけで良い
- `node_key`の更新
  - `node_key`に更新の概念はなく、手動で削除すれば、`bbc_core.py`を起動時に新規生成される

# コンフィグファイル(config.json)

- 設定ツール (bbc\_domain\_config.py) があるので、ファイルは直接編集する必要はない
  - コンフィグファイルを読み込むタイミングは、bbc\_core.pyプロセスの起動時のみであるため、コンフィグを修正した場合には再起動が必要となる
  - ただし、ドメインの新規追加/削除だけは、再起動せずにbbc\_domain\_update.pyで更新を取り込ませることができる
- 設定可能な項目を示すために、次ページ以降にコンフィグファイルを例示する

# コンフィグファイル(config.json)

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "00000000000000000000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ]  
            },  
            "static_nodes": {},  
            "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
        }  
    },  
    "ethereum": {  
        "chain_id": 15,  
        "port": 30303,  
        "log": "geth.log",  
        "account": "",  
        "passphrase": "",  
        "contract": "BBcAnchor",  
        "contract_address": ""  
    }  
}
```

- workingdir
  - カレントディレクトリ (bbc\_core.pyを起動したときのカレントディレクトリ) からの相対パスまたは絶対パスを指定する
  - 省略した場合はデフォルト値(.bbc1)が作成される
- client
  - クライアント待受ポート番号 (TCP)
  - node\_keyを使うかどうか
    - デフォルトはtrue
- network
  - 他のcore nodeとP2Pトポロジを形成する際に利用するポート番号 (TCPおよびUDP)
  - 受け入れ可能なTCPコネクション数 (=最大の隣接ノード数)

# コンフィグファイル(config.json)

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "00000000000000000000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ]  
            },  
            "static_nodes": {},  
            "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
        }  
    "ethereum": {  
        "chain_id": 15,  
        "port": 30303,  
        "log": "geth.log",  
        "account": "",  
        "passphrase": "",  
        "contract": "BBcAnchor",  
        "contract_address": ""  
    }  
}
```

- domain\_key

- 他のcore nodeに対してドメインに関する各種設定を行う際に、その設定メッセージの送り主が管理者であることを示すために署名付加を課すことができる（有効な署名のないものは破棄される）
- 署名に用いる秘密鍵のファイル名はdomain\_idのHEX文字列とし、PEM形式とする
  - 例：bcd5428b941(..中略..)b9216fc9a39f92f721f89a0f4c.pem
- use
  - trueにセットすると、署名付加が必要になる
  - デフォルトはfalse
- directory
  - 署名用の秘密鍵の置き場所
- obsolete\_timeout
  - 鍵を更新したときに、古い方の鍵を消さずに置いておく秒数

# コンフィグファイル(config.json)

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "00000000000000000000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ],  
                "static_nodes": {},  
                "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
            }  
        },  
        "ethereum": {  
            "chain_id": 15,  
            "port": 30303,  
            "log": "geth.log",  
            "account": "",  
            "passphrase": "",  
            "contract": "BBcAnchor",  
            "contract_address": ""  
        }  
    }  
}
```

- domains

- ドメインごとの設定
  - domain\_global\_0 (00000(..中略..)0000)はデフォルトで生成される
- node\_id
  - core nodeのそのドメインにおけるnode\_id
- static\_nodes
  - 隣接ノード情報 (隣接ノードを発見する度にコンフィグファイルも更新される)
- storage
  - type: internal/external
  - internalの場合は、ワーキングディレクトリの中にアセットファイルが格納される
  - externalの場合は、bbc\_core.pyプロセスではアセットファイルを格納せず、クライアント自身で何処かに保存する

# コンフィグファイル(config.json)

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "00000000000000000000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ],  
                "static_nodes": {},  
                "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
            }  
        },  
        "ethereum": {  
            "chain_id": 15,  
            "port": 30303,  
            "log": "geth.log",  
            "account": "",  
            "passphrase": "",  
            "contract": "BBCAnchor",  
            "contract_address": ""  
        }  
    }  
}
```

- domains(つづき)

- db

- トランザクション等の情報を保持するDBの設定
    - db\_type: sqlite/mysql
    - db\_name: データベース名
      - sqliteの場合はDBファイル名
      - mysqlの場合はDB名
    - replication\_strategy: all/p2p/external
      - allの場合は、全てのcore nodeに複製を配布する
      - p2pの場合は、実装されたP2Pアルゴリズムに依存する
      - externalの場合は、指定された外部DBに書き込む
    - db\_servers (mysqlの場合に利用。複数指定可)
      - db\_addr: mysqlサーバのアドレス
      - db\_port: mysql待受ポート
      - db\_user: mysqlのユーザ名
      - db\_pass: パスワード

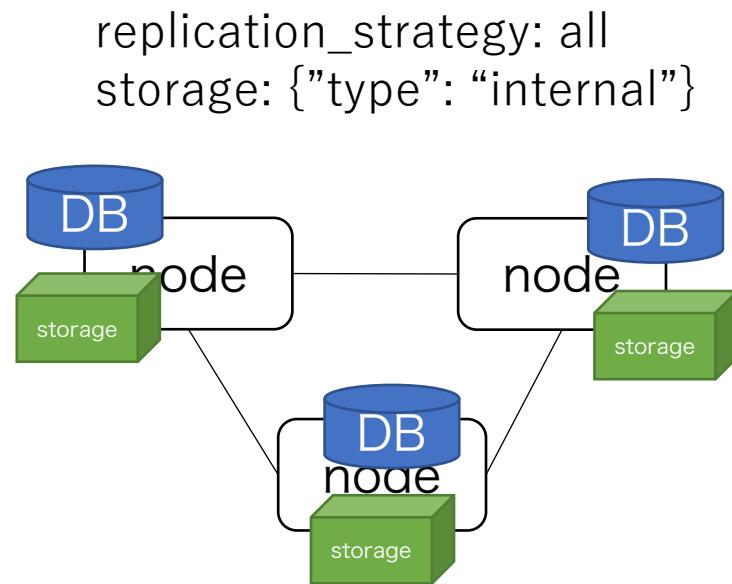
# コンフィグファイル(config.json)

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "00000000000000000000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ]  
            },  
            "static_nodes": {},  
            "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
        }  
    },  
    "ethereum": {  
        "chain_id": 15,  
        "port": 30303,  
        "log": "geth.log",  
        "account": "",  
        "passphrase": "",  
        "contract": "BBCAnchor",  
        "contract_address": ""  
    }  
}
```

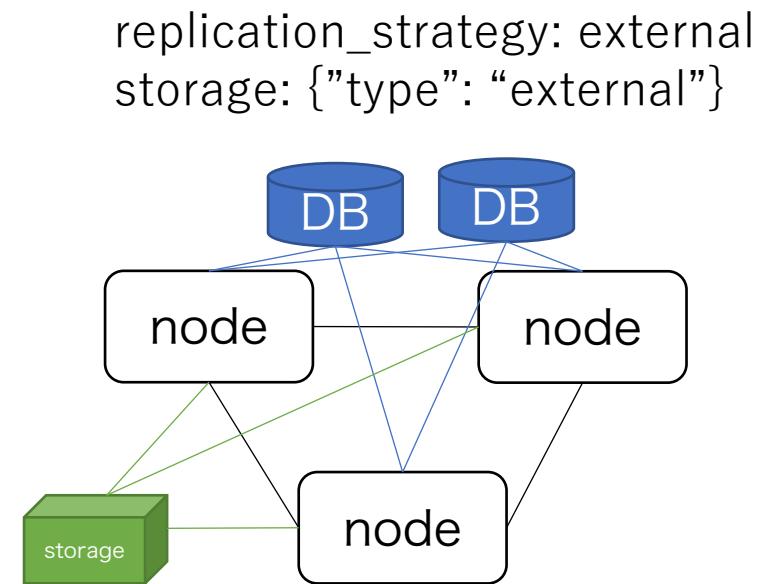
- domains(つづき)
  - use\_ledger\_subsystem
    - ledger\_subsystem (外部ブロックチェーンへのアンカリング) を行うかどうか
    - デフォルトはfalse
  - ledger\_subsystem
    - subsystem: ethereum/bitcoin
      - v1.0時点ではethereumのみ
    - max\_transactions: マークル木に含める最大のトランザクション数
    - max\_seconds: マークル木に最大数までトランザクションを含められなくても、ここで指定した秒数ごとにアンカリングを行う

# replication\_strategyとstorage

- コンフィグファイルの各ドメインに設定するreplication\_strategyとstorageによって下記のようなシステム構成となる



DBもストレージもcore nodeが持つパターン



DBもストレージも外部に配置するパターン

※DBとストレージはそれぞれ独立に配置場所を選べる

# コンフィグファイル(config.json)

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "00000000000000000000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ]  
            },  
            "static_nodes": {},  
            "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
        }  
    },  
    "ethereum": {  
        "chain_id": 15,  
        "port": 30303,  
        "log": "geth.log",  
        "account": "",  
        "passphrase": "",  
        "contract": "BBCAnchor",  
        "contract_address": ""  
    }  
}
```

- ethereum
  - chain\_id: 接続する Ethereum ネットワークの ID
  - port: 接続に用いるポート番号
  - log: ログファイル名
  - account: 使用する Ethereum アドレス (16進文字列)
  - passphrase: 当該アカウントのパスフレーズ
  - contract: v1.0 では “BBCAnchor” を指定
  - contract\_address: デプロイされた “BBCAnchor” コントラクトのアドレス (16進文字列)
- 以上は bbc1/core/ethereum/setup.py により設定できる

# 起動時の指定と設定ツールによる設定

```
{  
    "workingdir": ".bbc1-9000/",  
    "client": {  
        "port": 9000,  
        "use_node_key": false  
    },  
    "network": {  
        "p2p_port": 6641,  
        "max_connections": 100  
    },  
    "domain_key": {  
        "use": false,  
        "directory": ".bbc1/domain_keys",  
        "obsolete_timeout": 300  
    },  
    "domains": {  
        "0000000000000000000000000000000000000000000000000000000000000000": {  
            "module": "p2p_domain0",  
            "static_nodes": {},  
            "use_ledger_subsystem": false,  
            "ledger_subsystem": {  
                "subsystem": "ethereum",  
                "max_transactions": 4096,  
                "max_seconds": 3600  
            }  
        },  
        "62272e72024dcea491080b8c1361d5d3a824927f5959211c949d84bc6b261e59": {  
            "storage": {  
                "type": "internal"  
            },  
            "db": {  
                "db_type": "sqlite",  
                "db_name": "bbc_ledger.sqlite",  
                "replication_strategy": "all",  
                "db_servers": [  
                    {  
                        "db_addr": "127.0.0.1",  
                        "db_port": 3306,  
                        "db_user": "user",  
                        "db_pass": "pass"  
                    }  
                ]  
            },  
            "static_nodes": {},  
            "node_id": "653f0e7ad28ac666018f2d523ae9ca339b8c87f471f29f2ecfcc750b48a08f09"  
        }  
    },  
    "ethereum": {  
        "chain_id": 15,  
        "port": 30303,  
        "log": "geth.log",  
        "account": "",  
        "passphrase": "",  
        "contract": "BBCAnchor",  
        "contract_address": ""  
    }  
}
```

- bbc\_core.py起動時の引数で指定することで書き出されるconfig
  - workingdir
  - client
  - network
- ツールで設定できる項目
  - domainsのドメインエントリ
    - ただし、node\_idはbbc\_core.pyが自動生成するので設定不要

※ いずれにしても設定を変更する度にコンフィグファイルも更新される

# bbc\_domain\_config.pyの使い方

```
usage: python -t <generate|write|delete> -d <DOMAIN_HEX> -k1 <K1NAME> -v <K1VALUE> [-k2 <K2NAME>] [-v2 <K2VALUE>] -w <WORKINGDIR>

optional arguments:
  -h, --help            show this help message and exit
  -t {generate,write,delete}, --type {generate,write,delete}
                        operation type
  -d DOMAINHEX, --domainhex DOMAINHEX
                        domain hex
  -k1 KEY1NAME, --key1name KEY1NAME
                        key1 name
  -v VALUE, --value VALUE
                        value
  -k2 KEY2NAME, --key2name KEY2NAME
                        key2 name
  -w WORKINGDIR, --workingdir WORKINGDIR
                        working directory
```

- -wでワーキングディレクトリを相対パス指定
  - 存在しなければ、ディレクトリを作成する
  - ワーキングディレクトリの下にあるconfig.jsonという名のjsonファイルが操作対象のコンフィグファイル
- -t generateは、config.jsonが存在しなければデフォルトファイルを作成する
- -t writeは、config.jsonに指定したkeyを追加または更新する
- -t deleteは、config.jsonの指定したkeyを削除する

# bbc\_domain\_config.pyの使い方(続き)

```
usage: python -t <generate|write|delete> -d <DOMAIN_HEX> -k1 <K1NAME> -v <K1VALUE> [-k2 <K2NAME>] [-v2 <K2VALUE>] -w <WORKINGDIR>

optional arguments:
  -h, --help            show this help message and exit
  -t {generate,write,delete}, --type {generate,write,delete}
                        operation type
  -d DOMAINHEX, --domainhex DOMAINHEX
                        domain hex
  -k1 KEY1NAME, --key1name KEY1NAME
                        key1 name
  -v VALUE, --value VALUE
                        value
  -k2 KEY2NAME, --key2name KEY2NAME
                        key2 name
  -w WORKINGDIR, --workingdir WORKINGDIR
                        working directory
```

- -dはdomain\_idをHEX文字列表現で指定する
  - HEX文字列は、id\_create.pyツールでも簡単に生成できる
- -k1は1階層目のkey、-k2は2階層目のkeyを指定し、-vはその値である
  - 2階層目が存在しない場合は、-k1 use\_ledger\_subsystem -v true のようにする
  - また、-k1 db -v '{"db\_type": "sqlite", "db\_name": "test.db"}' のように-k2を使わずに連想配列全体を直接指定することもできる

# bbc\_domain\_update.pyの使い方

```
usage: bbc_domain_update.py [-h] [-4 IP4ADDRESS] [-6 IP6ADDRESS] [-p PORT]
                            [-d DOMAIN_ID] [-a] [-r] [-k NODE_KEY]

Online domain setting update tool

optional arguments:
  -h, --help            show this help message and exit
  -4 IP4ADDRESS, --ip4address IP4ADDRESS
                        bbc_core address (IPv4)
  -6 IP6ADDRESS, --ip6address IP6ADDRESS
                        bbc_core address (IPv6)
  -p PORT, --port PORT  port number of bbc_core
  -d DOMAIN_ID, --domain_id DOMAIN_ID
                        domain_id to setup
  -a, --add_domain      add a new domain
  -r, --remove_domain   remove a domain
  -k NODE_KEY, --node_key NODE_KEY
                        path to node key
```

- -aはbbc\_core.pyにdomain追加を指示する
  - 予め、bbc\_domain\_config.pyなどでコンフィグファイルに該当するdomainの設定をかけておかなければならない
- -rはbbc\_core.pyにdomainを削除を指示する
  - コンフィグファイルからも削除される
- node\_keyが必要な場合は-kオプションでpemファイルを指定する
  - デフォルトでbbc\_core.pyはnode\_keyを利用する

# domain\_key\_setup.pyの使い方

```
usage: domain_key_setup.py [-h] [-4 IP4ADDRESS] [-6 IP6ADDRESS] [-p PORT]
                           [-k NODE_KEY] [-dir DIRECTORY] -d DOMAIN_ID [-g]
                           [-n]

Domain_key manager

optional arguments:
  -h, --help            show this help message and exit
  -4 IP4ADDRESS, --ip4address IP4ADDRESS
                        bbc_core address (IPv4)
  -6 IP6ADDRESS, --ip6address IP6ADDRESS
                        bbc_core address (IPv6)
  -p PORT, --port PORT  port number of bbc_core
  -k NODE_KEY, --node_key NODE_KEY
                        node_key of the bbc_core.py
  -dir DIRECTORY, --directory DIRECTORY
                        Directory for domain_keys
  -d DOMAIN_ID, --domain_id DOMAIN_ID
                        Domain_id
  -g, --generate         Generate a domain_key
  -n, --notify           Notify update of domain_key
```

- -gオプションでdomain\_keyを生成する
  - -dirで保存先ディレクトリ、-dでドメイン名を指定する必要がある
  - 保存先ディレクトリは、bbc\_core.pyのconfig.jsonの"domain\_key"のdirectoryで指定している場所にしなければならない
- -nオプションは、domain\_keyが更新されたことをbbc\_core.pyに通知し、再読み込みを促す
  - なお、暫くの間マイグレーション期間として、古いdomain\_keyも有効である
  - マイグレーション期間の長さは、bbc\_core.pyのconfig.jsonの"domain\_key"のobsolete\_timeoutで秒単位で指定する
- node\_keyが必要な場合は-kオプションでpemファイルを指定する
  - デフォルトでbbc\_core.pyはnode\_keyを利用する

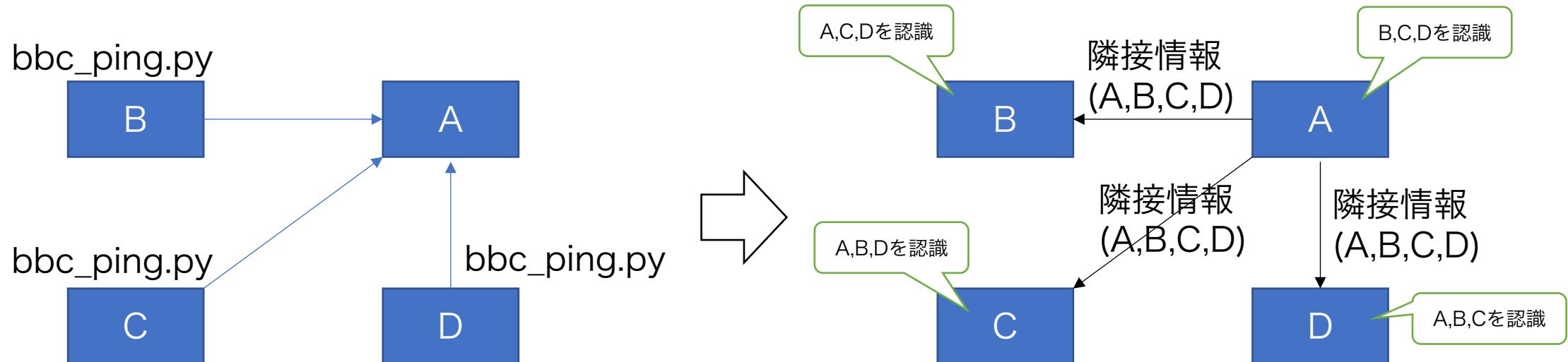
# node\_keyの設定

- コンフィグの "node\_key" の "use" エントリを true にすると、ドメイン管理用の機能を利用する場合に、クライアントからのメッセージに node\_key による署名を付加しなければならなくなる（デフォルトは true）
  - ドメイン管理機能：
    - ドメイン作成/破棄、neighbor list 取得、bbc\_ping、neighbor 設定、接続ユーザリスト取得、登録完了通知取得、統計情報取得、コンフィグ取得、ledger\_subsystem 起動/終了
- 鍵の生成
  - core node が初回起動時に、"node\_key" の中の "directory" で指定されたディレクトリに、node\_key.pem というファイルを生成する
  - このファイルをクライアントでも読み込む (set\_node\_key() を使う) ことで、ドメイン管理用のメッセージに署名が付加されるようになる
    - もちろん、ファイルをコピーして利用してすればよい
    - ファイルが存在しなければ（つまり "use" エントリが false なら）署名は付加されない
- 鍵の更新
  - 鍵を更新する場合は、一度 node\_key.pem を削除して、bbc\_core.py を再起動する必要がある

# ネットワーク設定

# フルメッシュトポロジーの構成

- ドメインのネットワーク設定手順
  - core node同士にお互いを認識させて、メッセージをやり取りできるようにする
    - bbc\_ping.py というユーティリティを使う( $B \rightarrow A$ にpingを投げる)
  - ドメインに加入する全てのcore node同士を認識させる
    - $C \rightarrow A$ 、 $D \rightarrow A$ にもbbc\_ping.pyを使ってpingを投げる
    - 自動的にAからB,C,Dに隣接情報が配布される → フルメッシュトポロジーが完成する

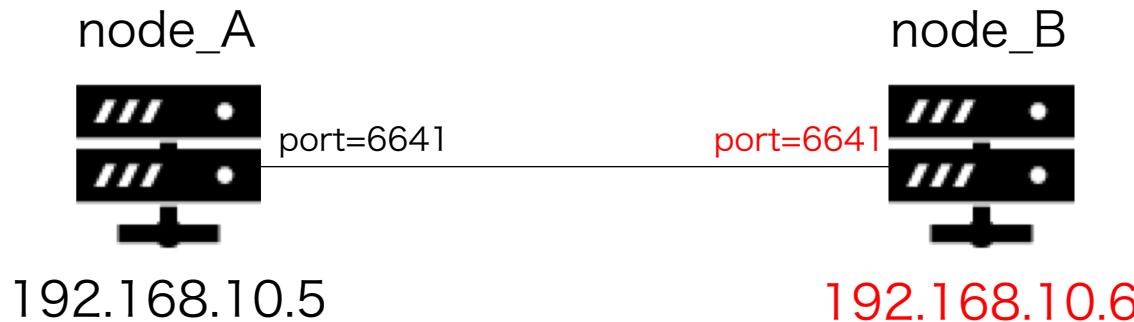


# NATと隣接情報

- 2つのcore nodeの間にNATがあると、自分から見える相手のIPアドレス/ポート番号と実際の相手のIPアドレス/ポート番号とは異なる
  - クラウド上や、オフィス、自宅内にcore nodeを配備した場合に発生しうる



# bbc\_ping.pyの使い方



```
usage: bbc_ping.py [-h] [-4 IP4ADDRESS] [-6 IP6ADDRESS] [-p PORT]
                   [domain_id] [dst_address] [dst_port]

Send domain ping to create domain and configure static neighbor nodes.

positional arguments:
  domain_id           Hex string of the domain_id
  dst_address         destination IPv4/v6 address of the neighbor node
  dst_port            destination port number of the neighbor node

optional arguments:
  -h, --help          show this help message and exit
  -4 IP4ADDRESS, --ip4address IP4ADDRESS
                      bbc_core address (IPv4) to connect
  -6 IP6ADDRESS, --ip6address IP6ADDRESS
                      bbc_core address (IPv6) to connect
  -p PORT, --port PORT port number of bbc_core
```

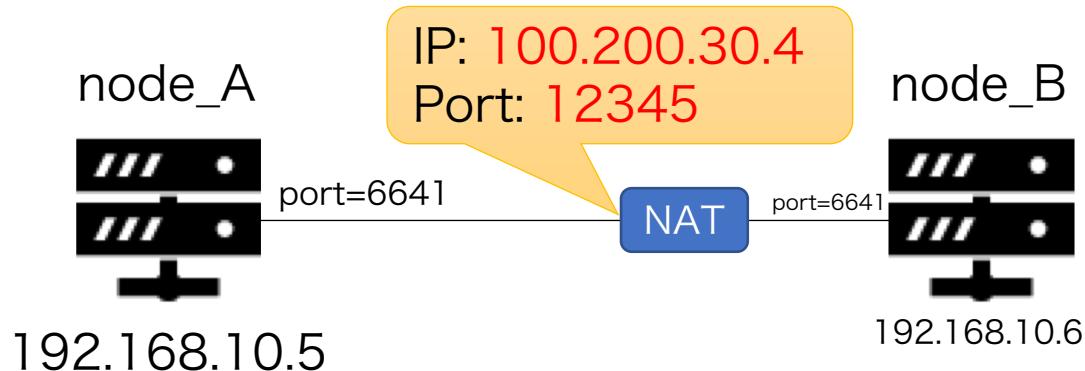
- node\_Aからnode\_Bにbbc\_pingを送る
  - domain\_idは112233xxxxxとする（実際には64文字のHEX文字列）
  - node\_Aにて以下のコマンドを実行する

```
python bbc_ping.py -4 localhost -p 9000 112233xxxx 192.168.10.6 6641
```

- IPv6アドレスを持っている場合は-6オプションで指定可能

bbc\_pingがnode\_Bに届くと、自動的にnode\_Bは返答を返し、その結果node\_Aおよびnode\_Bはお互いのIPアドレスとポート番号を認識する

# bbc\_ping.pyの使い方



```
usage: bbc_ping.py [-h] [-4 IP4ADDRESS] [-6 IP6ADDRESS] [-p PORT]
                   [domain_id] [dst_address] [dst_port]

Send domain ping to create domain and configure static neighbor nodes.

positional arguments:
  domain_id           Hex string of the domain_id
  dst_address         destination IPv4/v6 address of the neighbor node
  dst_port            destination port number of the neighbor node

optional arguments:
  -h, --help          show this help message and exit
  -4 IP4ADDRESS, --ip4address IP4ADDRESS
                      bbc_core address (IPv4) to connect
  -6 IP6ADDRESS, --ip6address IP6ADDRESS
                      bbc_core address (IPv6) to connect
  -p PORT, --port PORT port number of bbc_core
```

- node\_Aからnode\_Bにbbc\_pingを送る
  - domain\_idは112233xxxxxとする（実際には64文字のHEX文字列）
  - node\_Aにて以下のコマンドを実行する（NATの外側のアドレス、ポートを指定）

```
python bbc_ping.py -4 localhost -p 9000 112233xxxx 100.200.30.4 12345
```

- NATがIPv6アドレスを持っている場合は-6オプションで指定可能

bbc\_pingがnode\_Bに届くと、自動的にnode\_Bは返答を返し、その結果node\_Aおよびnode\_Bはお互いのIPアドレスとポート番号を認識する

# 履歴交差のためのdomain\_global\_0

# domain\_global\_0とは

- domain\_global\_0はドメイン同士を接続するための特別なドメインである
  - この特別なドメインでは、履歴交差情報(BBcCrossRefオブジェクト)をやり取りする
  - 履歴交差情報のやり取りはcore nodeが自動で行うため、利用者は意識する必要はない
- 各ドメインのcore nodeの一部(または全部)がdomain\_global\_0にも参加する
  - 参加方法は本資料P.15に示したように、bbc\_core.py --domain0 のようにオプションを指定すれば良い
  - 参加した後、bbc\_pingを使っていざれかの隣接ノードに接続する

以上