クラウド型連動装置のコントローラ機能仕様の検討

寺田　夏樹*　　潮見　俊輔　　遠山　喬（鉄道総合技術研究所）

Functional specification of controller, subsystem of interlocking device on cloud computing environment

Natsuki Terada*, Shunsuke Shiomi, Takashi Toyama (Railway Technical Research Institute)

The cloud interlocking device we are developing, is composed of three layers, field terminals to interface with field devices, interlocking logic units that process interlocking logic, and controllers that allocates process time. In this paper we report the functional specification of controllers, including communications between controllers, and between a controller and logic unit.

キーワード：クラウド型連動装置，コントローラ，監理端末，マスタースレイブ，ネットワークの分離
(cloud interlocking device, controller, observer, master-slave, clustering of network)

1. はじめに

鉄道総研では，連動装置の機器更新の費用削減や制御機能のクラウド化，災害等に対する強靭化，保守作業のクラウドサービス化を目指し，クラウド型の連動装置を開発している。その検討過程で，実際の論理演算を行う連動論理部が，必要となる処理量に応じて柔軟に増やせる構成であることが重要である，という認識に至った。解決策として，現場機器との直接的なインタフェースとなる現場端末と，論理演算を実際に処理する連動論理部のほかに，連動論理部に対して，処理の対象となる連動論理を割り当てるコントローラを設け，３つの階層からなるシステムを現在提案している（図 1）[1]。

本稿では，連動論理データを保持するとともに，連動論理部に対して連動論理の割当処理を行うコントローラの仕様について検討したので報告する。

2. コントローラ仕様

〈2･1〉　コントローラの基本機能　コントローラの基本的な機能は，①制御対象となる各駅の連動論理データをデータベースとして保有する，②個々の連動論理部において，どの連動論理データを扱うかを決定する，③連動論理部に対して処理を指示するとともに，必要な連動論理データをアップロードする，というものである。ここで同一の連動論理データを処理する連動論理部を複数割り当てることで，連動論理処理の多重化が実現できる。どの連動論理部の処理結果を採用するかについては，優先度に関する情報を連動論理部に対して伝送するものの，最終的な出力は，実際に現場機器と連動装置をインタフェースする現場端末で優先度の情報を参考にして行う。

コントローラは複数台設けるものとし，これらコントローラ間で連動論理データおよび連動論理処理の割当情報を共有する。ここで，各連動論理部において，連動論理データを受け取る元となるコントローラはあらかじめ決められた１台とする。すなわちコントローラと連動論理部との対応関係は１対 n となる。

コントローラは，連動論理部に対してデータをアップロードして，処理する連動論理を指示するだけではなく，連動論理部の死活監視を行い，その情報を他のコントローラと共有する。連動論理部に異常があると判断した場合には他の連動論理部に対して連動処理の再割り当てを行うことで連動論理処理の可用性を上げる。

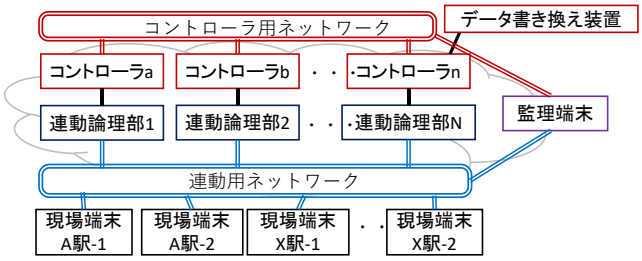

図 1　クラウド連動装置の 3 階層機器構成
Fig. 1.　Structure of processing unit for cloud interlocking device with three layers

〈2･2〉　コントローラ相互の死活監視とデータ共有の仕組み□□連動論理データや連動論理部の動作状態，割当て情報の共有に関しては，相互の死活監視と併せて行うこととした。死活監視と共有にあたっては，図 2 に示すようにマス

タとなるコントローラを１台決め（決め方は任意とする），マスタコントローラから他のコントローラ（スレイブコントローラ）に対し，死活監視に関する問い合わせとともに，現在所有している連動論理データのリストを配信し，その応答に従い，不足するデータを配信するというのが基本的な考え方である。ここで連動論理データはデータ入力端末を通じてコントローラの１台に入力を行うこととするため，その１台がマスタでない限りは，そのスレイブコントローラが一時的に最新の情報を持つことになる。

スレイブコントローラはマスタコントローラからの問い合わせに対して，自身および配下の連動論理部の状態を返すとともに，保有していないデータがマスタからのリストにある場合は，そのデータ本体の送信要求を行い，死活監視とは別にデータ本体の受領をする。逆にリストにないデータを保有している場合には，マスタに対し，そのデータに関する情報を送る。それに伴い，マスタからスレイブに対してデータ本体の要求をし，死活監視とは別のやり取りでデータ本体を取得する。

連動論理処理の割当データは，セグメント割当データと称する。セグメントとは，各連動論理部の１周期のプロセス時間を分割した処理単位であり，セグメントを１つもしくは複数占有して，１駅の連動論理の処理をさせる。このセグメント割当データについては原則としてマスタが管理するものとし，常にマスタが最新のものを保有する形をとる。セグメント割当はマスタにて自動で行うことを基本とする。例えば連動論理の処理の新規／追加の割当が必要な場合は，連動論理を指定したセグメント割当要求をデータ入力端末からスレイブ経由でマスタに対して行い，これを受けて，マスタが割当てを行う。一旦処理対象を割当てられたセグメントは最終的に解放処理が必要となるが，実際にはセグメント割当案から対象となる連動論理データの指定を外すことで自動的に解放される。この解放作業もマスタにて実施する。そのほかに保守の際に特定の連動論理部を物理

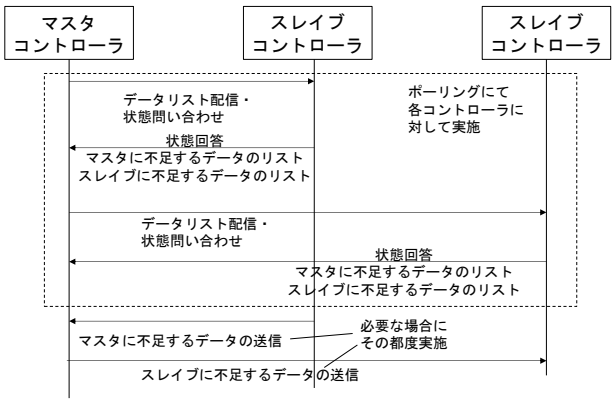

図２　死活監視と共有データの配信

Fig. 2.　Alive monitoring of controllers and delivery of shared data

的に停止する場合には，連動論理部を指定してセグメント割当の解放あるいは再割当を要求する。

コントローラ間の死活監視において，マスタからの問い合わせに対し，スレイブから反応が一定時間返ってこない場合は，マスタがスレイブの故障と判断する。一方，スレイブにおいてマスタからの問い合わせが一定時間来ない場合は，マスタの故障と判断し，残されたスレイブの中から新たにマスタを選定する。なお，コントローラの中の１台が他のコントローラから分離して単独となった場合は，そのコントローラは単独マスタとなる。

ここで，コントローラの故障判定を，通信の有無で判断した場合，ネットワークの障害と装置の故障とが区別できない。そのため，ネットワークの障害が発生し，例えばコントローラが２つのクラスタに分離した場合は，実際にはそれまでのマスタが動作し続けているのにもかかわらず，マスタから分離されたクラスタにおいて別途マスタが立ち上がることになる。つまりネットワーク分離の際のマスタ選出，再統合の際のマスタ統合処理について検討が必要となることが分かる。

〈2･3〉　マスタ選出作業および統合作業　ネットワーク分離時，マスタ故障時の新マスタ選出にあたっては，選出ルール（例えば，ID が若いサーバを選定する）を決めたうえで，その選出ルールで最優先となるサーバがマスタとして立候補し，それを承認することで，以後，そのサーバがマスタとして動作するというのが基本的な考え方である。また初期状態において，１台しかコントローラが動作していない場合は，その１台がマスタコントローラとなる。

ただし，新マスタ選出にあたって，一部のコントローラが動作していない，またはネットワークで結合されていないなどの理由で，そのコントローラからのマスタ承認が得られない可能性も想定される。そのため，全コントローラからの承認が得られなくても，マスタの否認が来ない限りは，立候補後一定時間でマスタとして動作することとする。

マスタ立候補に対して，自身の方がマスタとして優先されると判断される場合は，マスタ選出の調停という形で要求を行う。その結果，調停の要求を出したものがマスタとなる場合は，改めてマスタとして他のコントローラに対してマスタ宣言を行う。

ネットワーク分離が発生すると，分断されたネットワーク毎にマスタが立ち上がる。そのため，ネットワークが再結合するとマスタが２台以上並立することとなるが，その場合は，マスタを統合させる。

上記のプロトコルを検討する前提として，マスタコントローラはリストにあるコントローラに対して，リストから削除されない限り，反応がないスレイブコントローラに対しても死活監視の問合せをし続ける。一方，スレイブコント

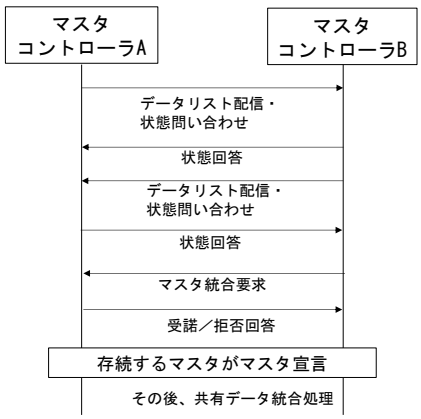

図 3　ネットワーク結合時のマスタ統合作業
Fig. 3.　Merging master controller function when two networks are connected

ローラも，自身がマスタと認識しているコントローラ以外の死活監視の問合せに対して，死活監視情報の共有の観点から，自身の死活状態を返すこととする。

この前提に従うと，ネットワークが再結合した場合に，図 3 に示すように 2 台以上のマスタが互いに死活監視の問合せを行うことになるため，それによりマスタ統合処理を開始することになる。2 回目の死活監視問い合わせに対して調停要求を出せば，その時点では互いのスレイブコントローラの状態が取得できているため，円滑に死活監視情報を引き継ぐことが可能である。その調停時における優先度は，ネットワーク分離時等と同じ考えに従えばよい。

マスタ統合に際し，互いの配下にあるスレイブコントローラおよびさらにその配下の連動論理部の死活情報は，ネットワークが再結合した時点で，それぞれで取得できるため，あとは連動論理データと連動処理割当を統合すればよい。連動論理データの統合については，マスタースレイブ間の共有処理と同じ形で行える。連動処理の割当については，スレイブに回る旧マスタから，配下の連動論理部に対する割当を受け取り，新旧マスタで同一の連動論理部に対して，競合しているものがあれば，それまでの制御状態に従った割当を採用することとする。

〈2･4〉　連動論理データおよび連動処理割当データの構造

ここで連動論理データやセグメント割当データについて述べる。なお，連動論理部が現場端末に対してネットワーク越しに配置されるため，非同期論理処理を行うことにより，ネットワーク遅延の問題を解決しようとしているが，その論理そのものは結線論理，マトリックス論理のどちらでも適用可能と考える。

連動論理データについては，同じ駅を対象にしたデータであっても，制御論理が異なる複数のデータを異なるバージョンにすることで同時に共有することを可能とする。これは，セグメント割当において，どのバージョンのデータを処理すればよいかを指定することによる。これにより，例えば連動論理の変更を想定した場合，新旧バージョンを同時に処理し，現場機器と接続する処理をあるタイミングで切り替えるということも可能となる。

一方，セグメント割当データについては，どこまでが固定情報なのか，状況に合わせて変更が必要な動的な情報なのかを考える必要がある。セグメント割当データにはバージョンを付与するものとし，バージョンの新規性に関する順序を決めることで，複数のバージョン間でどちらが最新かを決定可能とする。1 つのバージョンの中で，個々の割当案に含む情報は，連動論理データの対象駅およびそのバージョン（データ本体はこの情報をもとに別途読み込むのでここでは含まない）およびその優先度とした。優先度を変更する場合は，セグメント割当データを更新する必要がある。一方，その連動論理データの実行／停止については，動的に変わりうるものとして，バージョン管理の対象からは外すこととした。

コントローラと連動論理部のやり取りとしては以下の通りとなる。コントローラから連動論理部に対しては，連動処理割当データを送り，どのデータを処理するかの指示を行う。連動論理部はその指示にしたがい，必要な連動論理データをコントローラからダウンロードし，処理を開始する。また，コントローラは定期的に連動論理部に死活監視の問い合わせを行い，連動論理部はそれに回答する。なお，コントローラは連動論理部に対し，処理の開始／停止の指示も可能である。

3.　監理端末の仕様

〈3･1〉　監理端末の機能□□コントローラはアップローダおよびロードバランサの一種であり，コントローラが停止してもその配下の連動論理部は動作し続ける前提としている。しかし，コントローラが停止すると，その配下の連動論理部の死活監視も停止してしまう。これを補うのが監理端末である。監理端末はコントローラにより構成されるネットワークと連動論理部で構成されるネットワークの双方に接続し，連動論理部の死活監視を行うものである。仮にコントローラが停止しても，監理端末が連動論理部の状態を取得し，他のコントローラにその情報を送れば，連動論理部が停止しても，それを検出することが可能となる。

なおコントローラから見た場合，監理端末は連動論理部の死活監視に特化し，連動論理データやセグメント割当データを持たない特殊なコントローラであり，あくまでも連動論理部のバックアップとしての機能しか持たない。

図 1 では監理端末は 1 台となっているが，前節で述べたようなコントローラネットワークが分離しうるという前提

に立てば，そのコントローラの一種である監理端末が常にすべてのコントローラと接続されているという想定は難しい。また，連動論理部の死活監視能力も考え合わせると，監理端末は複数台配置して，それぞれが，数台のコントローラ配下の連動論理部の状態を監視するという形が現示的である。ただし，複数の監理端末が同じ連動論理部を監視することはあってもよいものとする。

コントローラによる死活監視を行えない場合の，配下の連動論理部のセグメント再割当は，マスタコントローラが当該コントローラと接続できない場合に実施するのではなく，監理端末からの情報により停止と判断された場合に実施するものとする。

単にコントローラが通信不能であることをもって，その配下の連動論理部に割り当てていたセグメントの再割当を行ってしまうと，連動論理部が動作しているのにもかかわらず，それに対する追加の割当を行うことになる。極端な例として，コントローラが１台だけ分離した状況を考えると，他のコントローラ配下のリソースを無理して追加で再配置を試みることになる。このことから，スレイブコントローラ配下の連動論理部に割り当てられていたセグメントを，コントローラ停止（あるいは通信不能時）の場合に別のコントローラ配下に再割当する際は，監理端末からその連動論理部の状態が取得できることを条件とする。

4　コントローラ機能の検証について

コントローラ機能の検証については，実際の装置をシミュレータとして模擬して検証する準備を行っている。本格的なシミュレータによる検証を行う場合，シミュレータが大規模になるとともに，仕様の問題点を見つけるまでに時間がかかるだけでなく，もし問題点が見つかった場合には，それをシミュレータにフィードバックするのは手戻りとなってしまう，という課題がある。

そこで，もう少し簡便にコントローラの個々の機能について検証するために，モデル検査（model checking）を使った機能検証を試みた。

モデル検査とは，システムを記述したモデルが，与えられた条件を満たすかどうかを網羅的に調べる手法の総称である。モデル検査の手法にも様々なものが挙げられるが，今回はコントローラ間の通信プロトコルの検証という側面があるため，古典的なモデル検査手法ではあるが，元々が通信プロトコルの検証手法として開発が進められた SPIN(2)を使用した。

SPIN の記述言語である promela（process meta language）では，複数のプロセスが任意のタイミングで実行権が移動しながら実行されていく（インターリーブする）形で，並行プロセスが表現される。それとは別にプロセス間通信を記述するチャネルという概念があり，これを用いて２つのプロセス間の同期通信が記述可能である。１つのコントローラを１つのプロセスで表現し（実際には大域変数を共有する複数のプロセス群で記述する），そのプロセス間の同期通信という形でコントローラ間の通信を表現した。実際には通信が失敗することもあるが，これについては，受信側は架空の通信プロセスとして，その結果について何も処理を行わない，という形で表現を行った。

SPIN のようなモデル検査では，処理の分岐部において選択可能な選択肢が複数あればそのうちの１つが非決定的に選択される，という形をとる。これにより，例えば新しくマスタを選定する場合には，具体的な選定アルゴリズムを記載しなくても，どれか１つのスレイブがマスタの候補として選ばれるといった記述が可能となっている。

複数のコントローラがマスタースレイブ形式で死活監視を行い，マスタが故障した場合には残されたスレイブの中から新たなマスタが選定されるモデルを作成した。

SPIN では，モデルのランダムな自動実行，対話的実行，条件式（時相論理によるもので，不変条件や到達性が表現できる）の検査，デッドロックなどの検査が可能であるが，ランダム実行において故障発生時に新たなマスタが選ばれる機能を確認した。そのほか，ネットワーク再結合時におけるマスタの調停およびその引継ぎについてもモデルを作成して確認を行っている。今後，データの共有についても検証を行う予定である。

モデル検査においては，検査対象の状態はメモリに収まる範囲に限定されるため，コントローラの処理全体をそのままモデル化するのではなく，検査する項目に合わせてモデルを記述する，という考え方を採る。そのため，検査した各機能が組み合わさったときの影響については必ずしも評価できるものではないが，その点に関しては現在作成しているシミュレータにて確認を行う予定である。

5.　おわりに

クラウド型連動装置のコントローラ機能仕様について，検討した結果を報告した。現在，矛盾点などを洗い出して，より実用的な仕様になる検討を実施中である。

文　　献

（１） S. Shiomi, et. al: “A study of processing methods for interlock logic on the cloud interlocking device”, Proc. Of J-RAIL2019, pp.378-381, 2019,
潮見他：「クラウド型連動装置の連動論理処理に関する検討」, J-Rail 2019, S2-5-5, pp378-381, 2019

（２） G. J. Holzman. The SPIN Model Checker, Addison-Wesley, 2004.