JSiSE Research Report
vol.35,no.6(2021-3)

ELECOA における教材オブジェクト間通信削減方式

森本 容介[*1]，仲林 清[*2]
*1 放送大学教養学部, *2 千葉工業大学情報科学部

Reducing Methods of the Amount of Communication between Courseware Objects in ELECOA

Yosuke Morimoto[*1], Kiyoshi Nakabayashi[*2]
*1 Faculty of Liberal Arts, The Open University of Japan,
*2 Faculty of Information and Computer Science, Chiba Institute of Technology

e ラーニングシステムのアーキテクチャである ELECOA において，通信要否の判断やキャッシュの使用により，教材オブジェクト間の通信回数を削減する実装を行った．次に，SCORM のテストケースを用いて，実装前後の通信回数を計測した．その結果，全体として，79.7%の通信が削減できた．また，ELECOA における各処理の特性を考慮した分析を行い，各方式の効果を考察した．本実装により，マルチプラットフォームに配置したコンテンツの動作速度の向上が期待できる．

キーワード: ELECOA，SCORM，分散環境

1. はじめに

著者らは，e ラーニングシステムのアーキテクチャである ELECOA（Extensible Learning Environment with Courseware Object Architecture）を提案している[1]~[3]．ELECOA では，教材オブジェクトと呼ぶ概念を導入した．教材オブジェクトは e ラーニングシステムのプラットフォーム上で動作するプログラム部品である．ELECOA の学習制御機能は，コンテンツに配置された教材オブジェクト同士の連携動作によって実行される．独習型のコンテンツのほか，グループ学習のコンテンツや，両者を組み合わせるコンテンツも実現できる．

現在，教材オブジェクトをマルチプラットフォームに配置する方法の研究を行っている[4]．例えば，学習者の端末上に配置された教材オブジェクトを用いて独習を行った後，サーバに配置された教材オブジェクトを用いてグループ学習を行うコンテンツが実現できる．

教材オブジェクトは木構造に配置され，隣接するノード間で通信を行う．通信のパターンとして，葉ノードから根ノードまでのような大域的な通信が発生する処理を規定している．これまでの設計においては，教材オブジェクトのツリーが単一プラットフォーム上で動作することを前提としていた．また，これまでの実装における教材オブジェクト間通信の実体は，PHP のメソッド呼び出しであり，通信のオーバーヘッドが低い．そのため，通信が発生する範囲や回数は，意識する必要がなかった．

教材オブジェクトをマルチプラットフォームに配置する場合，プラットフォームをまたぐ通信による動作速度の低下が課題となる．特に根ノードまでの通信を削減できれば，プラットフォームをまたぐ通信を削減することにつながる．前述の構成では，学習者の端末とサーバの間の通信回数を削減することが望ましい．

そこで，本研究では，これまでの ELECOA の設計と同等の動作を担保したうえで，教材オブジェクト間の通信回数を削減する方策を策定し，実装する．また，SCORM のテストケース[5]を用いて，削減効果を評価する．本稿では，参考文献(6)における報告を詳述するとともに，ELECOA における各処理の特徴を踏まえて分析した結果を報告する．

以下，2 章で ELECOA の教材オブジェクト間通信の仕組みを述べる．3 章で教材オブジェクト間通信を削減する方式を述べ，4 章でその効果を評価する．5 章

で本研究をまとめる．

2. ELECOA の教材オブジェクト間通信

2.1 ELECOA の動作の概要

ELECOA は図 1 のような階層型（木構造）のコンテンツを対象とする．内部ノードを含むすべてのノードに教材オブジェクトを配置する．教材オブジェクトは，配置されたノードを頂点とするサブツリーを制御する．また，ノード間で情報を共有するために，階層構造とは独立した共有学習目標を持つことができる．本稿においては，“ノード”には共有学習目標を含めない．共有学習目標にも教材オブジェクトを配置する．学習制御機能は，教材オブジェクト間でのコマンド送信により実現される．ここで，直接通信可能な教材オブジェクトは，親子ノード間，およびノードとそれに結びつけられた共有学習目標間に限る．以降の図においては，ノード，および共有学習目標に配置される教材オブジェクトの描画を省略する．また，ノードおよび共有学習目標と，それらに配置された教材オブジェクトとを区別せずに説明する．

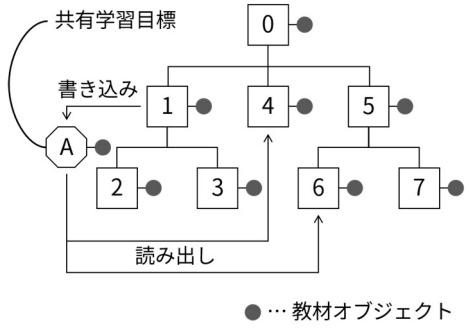

図 1　ELECOA のコンテンツの構成
（参考文献(2)の図 3 を改変）

コンテンツの起動時に，コンテンツの階層構造に従った教材オブジェクトのツリーを作る．ある時点で学習中のノードをカレントノードという．カレントノードに対応する学習リソースが学習者の端末に配信される．学習者からの学習制御の要求は，“学習コマンド”としてカレントノードに送られる．学習コマンドを受け取ったカレントノードは，次の処理を順に行う．まず，カレントノードから根ノードまで，各ノードの教材オブジェクトが持つ学習状態を更新する．次に，コンテンツに定義されたルールが成立すれば，学習者が指定した学習コマンドを別の学習コマンドに置き換える．次に，学習コマンドを実行し，次の配信ノード（新しいカレントノード）を決定する．そして，カレントノードの移動後に，その時点で実行可能な学習コマンドのリストを生成する．これら 4 つの処理を ELECOA の基本的な通信パターンとして規定し，それぞれをロールアップ処理，ポストコンディションルール処理，シーケンシング処理，学習コマンドリスト生成処理と呼ぶ．それぞれの処理の詳細を，次節以降に示す．

2.2 ロールアップ処理

まず，カレントノードが自身の学習状態を更新する．次に，親ノードにロールアップコマンドを送り，親ノードは自身の学習状態を更新する．これを根ノードまで繰り返す．カレントノードが共有学習目標を持つときの動作は次の通りである．図 2 の例では，ノード 2 から共有学習目標 A に学習状態を書き込み，ノード 4 とノード 5 から読み出している．ノードにおける共有学習目標の参照は，主学習目標に対応するものと，そうでないものの 2 種類に分けられる．ノード 4 においては共有学習目標 A が主学習目標に対応しており，ノード 5 においてはそうではないとする．ノード 2 における共有学習目標 A にも，主学習目標に対応しているか否かが決められているが，以降の説明に影響を及ぼさないため，この例では特定していない．

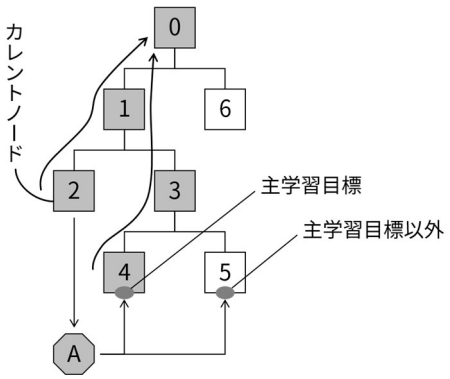

図 2　共有学習目標経由でのロールアップ処理の例

カレントノードからロールアップ処理を行うとき，そのノードが書き込む共有学習目標を，主学習目標に対応する共有学習目標として読み込むノードからもロールアップが発生する．図 2 の例では，ノード 2 からに加えて，ノード 4 からもロールアップが発生する．このとき，ノード 1 には，ノード 2 を起点とするロールアップコマンド，およびノード 4 を起点とするロー

ルアップコマンドが送られる．ノード 1 では，両者が届くのを待ち合わせてから，ノード 0 にロールアップコマンドを送る．

図 2 で背景色をつけたような，ロールアップ処理によって学習状態を更新するノードと学習目標をロールアップセットと呼ぶ．ELECOA のロールアップ処理においては，まずロールアップセットを求める処理を行い，次にロールアップセット内にロールアップのためのコマンドを伝播させる[1]．

2.3 ポストコンディションルール処理

カレントノードから根ノードまで，各ノードが持つルールが評価される．ルールが成立した場合，学習コマンドを置き換える．その際，そのノードを新しい学習コマンドの実行ノードとする．複数のノードでルールが成立する場合，より根ノードに近い方を優先する．

図 3 の状況を考える．ノード 4 で，“次へ”コマンドを発行した．ポストコンディションルール処理により，ノード 2 では“やり直し”に置き換えるルールが，ノード 1 では“次へ”に置き換えるルールが成立した．この場合，より根ノードに近いノード 1 が優先され，ポストコンディションルール処理の結果は，ノード 1 を実行ノードとする“次へ”となる．

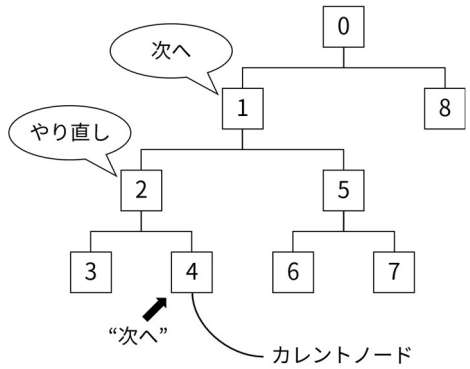

図 3　ポストコンディションルール処理の例

2.4 シーケンシング処理

自ノードを頂点とするサブツリー内で次の配信ノードを探し，決定できなければ，親ノードに処理を依頼する．図 4 は，ノード 3 で“次へ”を実行した例である．ノード 3 は子ノードを持たず，“次へ”に対応する次の配信ノードを決定できない．そこで，親ノードであるノード 1 に処理を依頼した（図 4 ①）．ノード 1 も自ノードの配下から決定できないため，ノード 0 に処理を依頼した（図 4 ②）．ノード 0 は，ノード 4 に配信可能か問い合わせた（図 4 ③）．ノード 4 は，まず，ノード 5 に配信可能かを問い合わせた（図 4 ④）．ノード 5 には習得済みならば飛ばすというルールが設定されており，すでに習得済みであったため，配信できない旨の結果が返った（図 4 ⑤）．次に，ノード 4 がノード 6 に問い合わせた（図 4 ⑥）ところ，配信可能という結果が返った（図 4 ⑦）．ノード 6 が配信できるという情報が，ノード 3 まで戻された（図 4 ⑧～⑩）．その結果，ノード 6 が新しいカレントノードとなる．なお，①と⑩は 1 回の通信におけるコマンド送信とその戻りであり，その他も同様である．

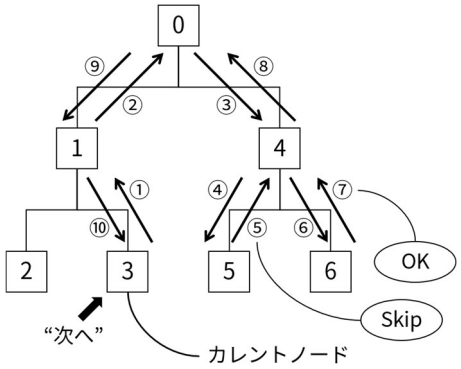

図 4　シーケンシング処理の例

ポストコンディションルール処理により，学習コマンドの実行ノードがカレントノードでない場合も，シーケンシング処理の起点はカレントノードである．自身が実行ノードでなければ，親ノードに学習コマンドを送る．

2.5 学習コマンドリスト生成処理

学習コマンドはカレントノードから根ノードに向かって伝播されるため，両者を結ぶパス上にあるノードが実行できる学習コマンドを集約し，カレントノードが実行可能な学習コマンドを生成する．

2.6 制御状態の更新

各ノードは，現在試行中であることを示すアクティブフラグと，親ノードの学習状態の決定に影響を及ぼすカレントフラグを有している．図 5 の構造を持つコンテンツにおいて，ノード 3 からノード 4 に進み，その後ノード 3 に戻ることを考える．ノード 1 を頂点とするサブツリーは，ノード 4 に進んだときに試行が終

わり，ノード 3 に戻ったときに新しい試行が始まる．その後，ノード 1 の学習状態を決定するため，ノード 2 について現在の試行，つまり新しい試行の学習状態を用いるか，最新の試行，つまり前回またはそれ以前の学習状態を用いるかの 2 通りが考えられる．これらの 2 通りの値を選択できる状態にあることを表すフラグがカレントフラグである．

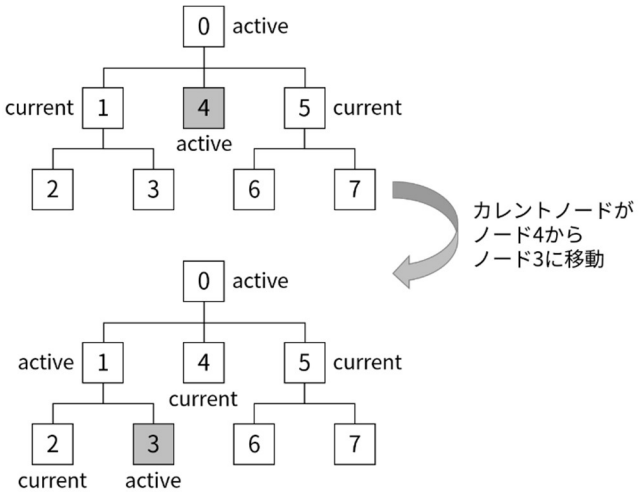

図 5　アクティブフラグとカレントフラグ

図 5 において，カレントノードがノード 4 からノード 3 に戻ったときを例に，アクティブフラグ，およびカレントフラグの更新方法を説明する．なお，図 5 には，カレントノードの移動開始前，および移動完了後において，アクティブフラグが真であるノードに“active”，カレントフラグが真であるノードに“current”と記載している．

ロールアップ処理のタイミングで，葉ノードのアクティブフラグを偽にする．これによりノード 4 のアクティブフラグが偽に変わる．

シーケンシング処理において，次の配信ノードが決定できず，親ノードに処理を依頼した場合は，アクティブフラグを偽にする．ノード 0 は子孫ノードから次の配信ノードが決定できたため，アクティブフラグは真のままである．

カレントノードが移動した後，学習コマンドリスト生成のタイミングで，新しいカレントノードの先祖ノードの中から，アクティブフラグが偽である最上位のノードを見つける．図 5 の例ではノード 1 である．そのノードの子孫ノードのカレントフラグ，および先祖ノードの子ノードのカレントフラグを真にする．これにより，ノード 1 ～ノード 5 のカレントフラグが真となる．次に，カレントノードのアクティブフラグを真にし，アクティブフラグを真にする役割を持つコマンドを根ノードまで伝播する．このとき，各ノードにおいて，アクティブフラグが真に変わるタイミングで，つまり新しい試行が始まる場合は，カレントフラグを偽にする．これにより，ノード 3 とノード 1 のアクティブフラグが真に，カレントフラグが偽に変わる．ノード 0 までコマンドが送られるが，アクティブフラグは真，カレントフラグは偽のまま変化しない．

以上が，各フラグの更新方法である．子ノードの学習状態を取得する通信には，現在の試行を用いるか，最新の試行を用いるかの指定を含める．子ノードでは，その指定とカレントフラグにより，適切な学習状態を返却する．

3.　教材オブジェクト間通信の削減方式

前章で述べた ELECOA の通信を，次のような観点から削減する．

- コンテンツの状態によらず静的に決まる事柄を求める処理は，教材の実行時ではなく事前に行う．
- 教材オブジェクト間の通信を行っても，教材オブジェクトの状態や，以降の処理に影響を及ぼさないことがわかっている場合は，通信を省略する．
- 通信相手の教材オブジェクトから取得すべきデータをキャッシュとして保持し，通信を省略する．

以降，本研究で設計した通信削減方式を述べる．

3.1　コンテンツの起動時の処理

① 各ノードにおけるロールアップセットはコンテンツの実行中に変化しない．そこで，コンテンツの起動時に，各ノードに対してロールアップセットを求め，各ノードが保持する．これにより，ロールアップ処理の際の，ロールアップセットを求めるための通信が省略できる．

② 各ノードのポストコンディションルールの有無は，実行時に変化しない．そこで，コンテンツの起動時に，各ノードに対して先祖ノードのポストコンディションルールの有無を求め，各ノードが保持する．また，各ノードにおいて，自ノードの学習状態が変化しなければ，ポストコンディショ

ンルール処理を省略できることを確定できる場合は，その旨を保持する．

3.2 根ノードまでの通信の省略

③ あるノードでロールアップ処理を行った結果，学習状態が変化しなかった場合，先祖ノードの学習状態が変化することはない．そこで，親ノードへのロールアップコマンドの送信を行わない．

④ 3.1 節の②で述べた方法により，親ノードへのポストコンディションルール処理のコマンド伝播が不要と判断できる場合は，自ノードで止める．

⑤ 学習コマンドリスト生成処理のタイミングにおいて，アクティブフラグとカレントフラグが更新されるのは，カレントノードから根ノードのうち，アクティブフラグが真である最下位のノードを頂点とするサブツリー内のみである．そこで，アクティブフラグが真の場合，親ノードへのコマンド伝播を省略する．

3.3 キャッシュの使用

親ノードへのロールアップコマンドの送信時に，自ノードの学習状態，および“次へ（Continue）”と“前へ（Previous）”コマンドによる自ノードを頂点とするサブツリー内のノードからの配信可否を親ノードに伝える．親ノードはそれらをキャッシュする．

⑥ 内部ノードにおける典型的なロールアップ処理では，子ノードの学習状態から自ノードの学習状態を求める．ここで，子ノードの学習状態を取得する通信を省略し，キャッシュを用いる．学習状態の決定に子ノードの現在の試行を用いる場合は，自ノードの試行を終える際，つまりアクティブフラグが偽に変わるタイミングで，子ノードの学習状態のキャッシュを初期状態に戻す．

⑦ 配信可否のキャッシュは，シーケンシング処理において，子ノードに配信可否を問い合わせる通信の省略に用いる．例えば，図 4 における④・⑤の通信が省略できる．子ノードの配信可否の状態が変わるときはロールアップ処理が発生し，キャッシュが更新されるため，キャッシュの無効化は考慮しなくてよい．

4. 通信削減方式の実装と効果の評価

4.1 通信削減方式の実装

ELECOA の実装として，PHP を用いた php-elecoa[2]，および php-elecoa を JavaScript で再実装した js-elecoa が存在する．前章で述べた教材オブジェクト間通信の削減方式を js-elecoa に実装した．同一のコンテンツに対する同一の操作を，php-elecoa と js-elecoa の両方で行い，教材オブジェクト間の通信回数を比較することにより，設計した通信削減方式の効果を測定できる．php-elecoa と js-elecoa に，教材オブジェクト間通信のログを出力する機能を組み込んだ．

4.2 通信削減効果の評価方法

SCORM 2004 3rd Edition[7]には，適合性テストに用いるテストケース（テスト用コンテンツ）が 100 種類用意されている[5]．それぞれのテストケースには，テストを進めるためのステップ（手順）が定義されている．SCORM のテストケースは，独習型の教材として妥当な構成と考えられることから，これを用いて通信削減効果を評価する．

詳細な条件は次の通りである．

ELECOA では，SCORM のナビゲーションリクエスト（学習コマンド）のうち，Abandon All，および Abandon を実装していない．これらを使ったテストケースである SX-04a と SX-04b を除外する．

各テストケースの最初のステップ（ステップ 1）はコンテンツの起動である．php-elecoa と js-elecoa では，コンテンツの起動方法が異なるため，通信回数の単純な比較が行えない．そこで，ステップ 1 は除いて集計する．これにより，ステップ数が 1 しかない OB-03c は除外される．ここまでの条件により，対象となるテストケース数は 97，総ステップ数は 419 である．

js-elecoa では，コンテンツの起動時に学習コマンドリストを静的に生成し，実行時には学習コマンドリストの生成を行わない．php-elecoa では実行時に行っているが，これを行わないとしても通信回数は変わらない．そこで，この違いは考慮しない．

php-elecoa，js-elecoa ともに，意図しない教材オブジェクト間通信が発生する不具合があった．これらは無害な通信であるが，それぞれの実装の設計とは異な

る．そこで，これらの不具合による通信は除いて集計する．なお，集計から除いた通信は，php-elecoa では全通信回数 25,671 回のうち 134 回（0.5%），js-elecoa では 5,216 回のうち 41 回（0.8%）であった．

4.3 通信回数の計測結果

テストケース OB-01a のステップ 2 における通信回数の計測結果を，表 1 に示す．各通信を，2 章で述べた 4 種類の通信パターンに分類した．ロールアップ処理は php-elecoa の 40 回に対して, js-elecoa は 14 回に削減できている．合計で 46 回から 17 回に減少し，63.0%の通信を削減できた．

表 1　OB-01a のステップ 2 における通信回数

	php-elecoa	js-elecoa
ロールアップ	40	14
ポストコンディションルール	1	0
シーケンシング	4	2
学習コマンドリスト生成	1	1
計	46	17

評価対象とする 419 ステップにおける通信回数の総計，および通信回数の削減率を，表 2 に示す．通信回数の総計は, php-elecoa の 25,537 回に対して, js-elecoa は 5,175 回であり，79.7%削減できた．ロールアップ処理における通信回数は全体に占める割合が大きく，削減率への寄与度が高い．

表 2　総通信回数，および削減率

	php-elecoa	js-elecoa	削減率
ロールアップ	21,697	2,374	89.1%
ポストコンディションルール	770	230	70.1%
シーケンシング	1,915	1,610	15.9%
学習コマンドリスト生成	1,155	961	16.8%
計	25,537	5,175	79.7%

次節以降で，各処理の特性を踏まえた分析を行う．

4.4 ロールアップ処理の分析

ロールアップ処理は，起点ノードから根ノードに向かって ROLLUP コマンドを伝播することにより行われる．3.2 節の③により，状態変化が起こらなかったノードで，親ノードへの ROLLUP コマンドの伝播を止める．そこで，起点ノードの深さ，およびそのノードからの距離ごとに，ROLLUP コマンドによる通信の削減率を計算した．ノード間の距離は，隣接していれば 0 で，エッジが 1 つ増えるたびに距離も 1 増えると定義する．図 2 の例では，ノード 2 の深さは 2 である．また，ノード 2 を起点とする ROLLUP コマンドの伝播において，ノード 2 からノード 1 への通信は距離 0，ノード 1 からノード 0 への通信は距離 1 である．ノード 1 からノード 0 への通信は，ノード 2 を起点とみると距離 1，ノード 4 を起点とみると距離 2 であり，一意に定まらない．しかし，今回用いたテストケースにはこのようなケースがなく，ROLLUP コマンドの距離はすべて一意に定まった．結果を表 3 に示す．

表 3　ROLLUP コマンドの削減率

		計	距離			
			0	1	2	3
深さ	1	30.0%	30.0%			
	2	31.0%	7.7%	56.5%		
	3	54.7%	33.9%	64.1%	76.9%	
	4	39.3%	0.0%	48.6%	54.3%	54.3%
総計		35.4%	21.0%	56.6%	66.2%	54.3%

深さ 2 のノードを起点とする ROLLUP コマンドは，距離 0 の通信が 7.7%，距離 1 の通信が 56.5%削減でき，距離を考慮しない削減率は 31.0%であった．設計から想定される通り，起点からの距離が離れるほど削減できていることが分かる．なお，深さを考慮しない総計行の結果は, 距離 2 より距離 3 の削減率が少ない．これは，深さ 4 のノードを持つコンテンツが少なく，それらのコンテンツにおいては削減率が低かったためである．

表 2 に示した通り，ロールアップ処理における通信は大幅に削減できている．これには，3.1 節の①で述べたコンテンツの起動時におけるロールアップセットの作成，3.3 節の⑥で述べたキャッシュの使用，およ

び本節で評価した ROLLUP コマンドの省略（3.2 節の③）が寄与している．

4.5 ポストコンディションルール処理の分析

ポストコンディションルール処理は，カレントノードから根ノードに向かって，EXITCOND コマンドを伝播することにより行われる．3.2 節の④により，親ノードへの EXITCOND コマンドの伝播が不要と判断できるノードで伝播を止める．そこで，前節におけるロールアップ処理と同様の分析を行う．結果を表 4 に示す．

表 4　EXITCOND コマンドの削減率

		計	距離			
			0	1	2	3
深さ	1	85.6%	85.6%			
	2	69.1%	38.7%	99.4%		
	3	77.8%	53.8%	82.1%	97.4%	
	4	47.9%	34.3%	48.6%	54.3%	54.3%
総計		70.1%	58.7%	89.5%	77.0%	54.3%

設計から想定される通り，カレントノードからの距離が離れるほど削減できていることがわかる．

4.6 シーケンシング処理の分析

シーケンシング処理における学習コマンドの伝播は，探索範囲を広げるための子ノードから親ノードに向かっての通信と，自ノードを頂点とするサブツリー内で次に配信するノードを探すための親ノードから子ノードに向かっての通信に分けられる．3.3 節の⑦により，親ノードから子ノードへの通信が削減されることが期待できる．計測の結果，親ノードから子ノードに学習コマンドを送る通信の削減率は 21.9%であった．

通信回数は削減できているが，ロールアップ処理やポストコンディションルール処理と比較して削減率は低い．図 4 の例では，新旧カレントノードを結ぶパス上のノードは，ノード 3，ノード 1，ノード 0，ノード 4，ノード 6 である．キャッシュにより，ノード 4 からノード 5 への学習コマンドの伝播を省略できる可能性があるが，新旧カレントノードを結ぶパス上の学習コマンドの伝播は省略できない．削減率が相対的に低い理由として，通信を省略できる可能性があるエッジが限られることがあげられる．

なお，シーケンシングコマンドによる子ノードから親ノードへの通信は省略できず，計測結果は全く同じであった．

4.7 学習コマンドリスト生成処理の分析

3.2 節の⑤では，学習コマンドリスト生成処理に付随するアクティブフラグの更新処理における通信を削減した．⑤の方法を実装しない場合は根ノードまでの通信が発生するのに対して，実装する場合は新旧カレントノードの最小共通祖先ノードより上位の通信が省略できる．ここでは，新しいカレントノードの深さ，および新しいカレントノードと通信元ノードの距離ごとに，学習コマンドリスト生成処理におけるアクティブフラグとカレントフラグを更新するための通信の削減率を求めた．例えば，新しいカレントノードから親ノードへの通信は距離 0，逆の通信は距離 1 である．結果を表 5 に示す．

新しいカレントノードの深さが 4 となるステップにおいて，根ノードから子ノードへの通信が発生すれば，その距離は 4 である．今回用いたテストケースでは，通信削減方式の実装有無にかかわらず距離 4 の通信はなかったため，表 5 には記載していない．

表 5　制御状態更新処理の通信削減率

		計	距離			
			0	1	2	3
深さ	1	0.0%	0.0%	0.0%		
	2	17.3%	0.0%	25.3%	0.0%	
	3	16.0%	0.0%	14.6%	26.5%	0.0%
	4	30.1%	0.0%	25.4%	33.3%	53.1%
総計		16.8%	0.0%	22.6%	28.9%	47.3%

距離 0 とは，新しいカレントノードから親ノードへの通信である．また，深さと距離が同じ通信は，根ノードから子ノードへの通信である．本研究で設計した通信削減方式では，これらの通信は削減できない．後者の通信は，回数が少なく，総計に及ぼす影響は小さい．距離が離れるほど削減率が高いことがわかる．

4.8 通信が行われたエッジ数

同一プラットフォーム内での通信よりも，プラットフォームをまたぐ通信に時間がかかる．また，プラットフォーム間通信は，接続の確立に時間がかかる．プラットフォームをまたぐ通信は，回数に加えて，有無も重要な指標となる．そこで，各テストケースの各ステップについて,通信が行われたエッジ数を調査した．

根ノードからの距離ごとに集計した結果を，表 6 に示す．ここでは，根ノードからの距離を，親子ノード間の通信は親の深さ，ノードと共有学習目標間の通信はノードの深さと定義する．例えば，php-elecoa による OB-01a のステップ 2 では，距離 0 で 3 エッジ，距離 1 で 2 エッジにおいて通信が発生した．これらをそれぞれ，php-elecoa の距離 0，距離 1 のエッジ数に加算する．

表 6　通信が行われたエッジ数の削減率

距離	php-elecoa	js-elecoa	削減率
0	1,276	653	48.8%
1	1,206	873	27.6%
2	352	291	17.3%
3	277	231	16.6%
計	3,111	2,048	34.2%

通信が行われたエッジの延べ数は，php-elecoa の 3,111 に対して，js-elecoa は 2,048 であり，34.2%削減できた．なお，この延べ数を評価対象とする 419 ステップで割ると，1 ステップあたりの平均が求まる．各ステップにおける平均は，php-elecoa が 7.4，js-elecoa が 4.9 であった．平均して，2.5 のエッジにおいて，通信回数が 1 以上から 0 になった．距離別に見ると，距離が短いほど，つまり根ノードに近い位置であるほど削減率が高い．本研究における通信削減方式の実装により，通信が行われるエッジ数が減少し，その割合は根ノードに近いほど高いことがわかった．

5. まとめ

本研究では，ELECOA における教材オブジェクト間通信を削減する方式を設計し，実装した．そして，SCORM のテストケースを用いて通信回数の削減効果を評価した．その結果，全体として，79.7%の通信を削減できたことを確認した．また，各処理やプラットフォーム間通信の特徴を考慮した分析を行った．その結果,根ノードに近づくほど通信が削減できることや,通信が発生するエッジ数を減らせることを確認した．カレントノードから距離が離れたノード間での通信が削減できたことにより，プラットフォームをまたぐ通信の削減に有効であることを示した．

謝辞

本研究は，JSPS 科研費 17H00774 の助成を受けた．

参　考　文　献

(1) 仲林清, 森本容介: “拡張性を有する適応型自己学習支援システムのためのオブジェクト指向アーキテクチャの設計と実装”, 教育システム情報学会誌, Vol.29, No.2, pp.97-109 (2012)

(2) 森本容介, 仲林清, 芝崎順司: “ELECOA における教材オブジェクト・プラットホーム間インタフェースの設計と実装”, 電子情報通信学会論文誌, Vol.J98-D, No.6, pp.1033-1046 (2015)

(3) 仲林清, 森本容介: “拡張性を有する学習支援システムにおける再利用性向上のための教材オブジェクトデザインパターンの設計と実装”, 教育システム情報学会誌, Vol.35, No.3, pp.248-259 (2018)

(4) 仲林清, 森本容介, 池田満, 瀬田和久, 田村恭久: “拡張性を有する学習支援システムアーキテクチャに基づく分散マルチプラットフォーム学習環境の検討と試作”, 教育システム情報学会研究報告, Vol.33, No.5, pp.59-66 (2019)

(5) Advanced Distributed Learning Initiative: “SCORM ® 2004 (3rd Edition) Conformance Test Suite”, https://adlnet.gov/projects/scorm-2004-3rd-edition-conformance-test-suite/ (参照 2021.2.11)

(6) 森本容介, 仲林清: “ELECOA における教材オブジェクト間通信削減方式の実装と評価”, 教育システム情報学会第 45 回全国大会講演論文集, pp.233-234 (2020)

(7) Advanced Distributed Learning: “SCORM 2004 3rd Edition Specification”, https://adlnet.gov/projects/scorm-2004-3rd-edition/ （参照 2021.2.11）