ネットワーク通信特性を考慮した分散制御システムのシミュレーション
— タームワープ機構とプロトコルモデルを用いた通信遅れの実装 —

旭川工業高等専門学校　○戸村 豊明
北 海 道 大 学　金井 理
北 海 道 大 学　岸浪 建史

要 旨

FA や BA の分野で導入され始めている分散制御システム（DCS）において，通信遅れやパケット損失の発生を高精度に予測したいという要求が高まっている．これを実現するために，本報では，これまで提案してきた DCS シミュレーションモデルに対して，並列離散事象シミュレーションの一手法であるタイムワープ機構と LonTalk プロトコルモデルを組み込む事により，各種の通信サービスにおける通信遅れを定量的に評価できるようにする事を目的とする．

1. はじめに

近年，FA や BA（ビルオートメーション）の分野において，図 1 に示すような，多数の制御ノードをオープン規格のネットワークで相互接続した分散制御システム（DCS）が導入され始めている．

制御ノード数が多い DCS では，ネットワーク上での通信トラフィックの集中による制御性能の低下が起こりやすい．ゆえに，各制御ノードに対して通信パラメータを設定した後，通信遅れやパケット損失の発生を高精度に予測したいという要求が高まっており，そのためには，DCS シミュレーションにおいて，送受信時刻を持つ全事象を論理上並列に連鎖させるメカニズムが必要となる．

そこで前報[1] では，並列離散事象シミュレーションの一手法であるタイムワープ機構[2] と LonWorks ネットワーク用の LonTalk[3] プロトコルモデルを，既報[4] の DCS シミュレーションモデルへ組み込む事により，任意時刻におけるネットワーク上の通信状況を予測する手法を提案した．しかしながら，前報は LonTalk 固有の通信サービスを実装していなかったので，充分に高精度な通信状況の予測は不可能であった．ゆえに，LonTalk に準拠した挙動を持つプロトコルモデルが必要となる．

そこで本報では，前報のモデルへ LonTalk Reference Implementation[3] に基づく洗練された LonTalk プロトコルモデルを組み込む事により，各種通信サービスにおける通信遅れの定量的評価を可能にする事を目的とする．

2. 本研究で提案する DCS シミュレーションモデル

本研究で新たに提案する DCS シミュレーションモデルの全体像を図 2 に示す．本研究では，有限状態機械が生成する事象をシグナルと呼ぶ．

図 2 における各モデルは固有の局所仮想時計（LVT）[2] を持っており，それが指す時刻以前のシグナルを受信すると，タイムワープ機構によって局所仮想時計と局所状態[2] が巻き戻され，送信済みシグナルと因果関係を持つモデルの局所仮想時計も連鎖して巻き戻される．この巻き戻しアルゴリズムにより，各モデルの局所仮想時計間の同期機能が実現されている．

図 2 におけるネットワークモデルはチャネルに相当し，各デバイスモデルが送信するフレームを他のデバイスモデルへ伝送し，かつフレームの送受信時刻に基づいて，チャネル状態（アイドルまたはアクティブ）を管理する．

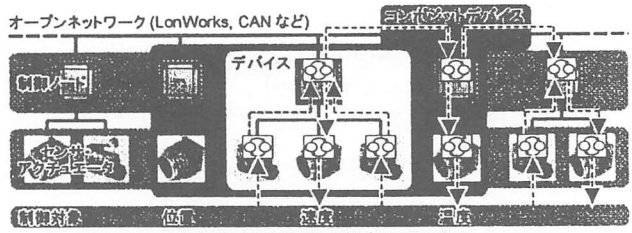

図 1. DCS の構造と挙動

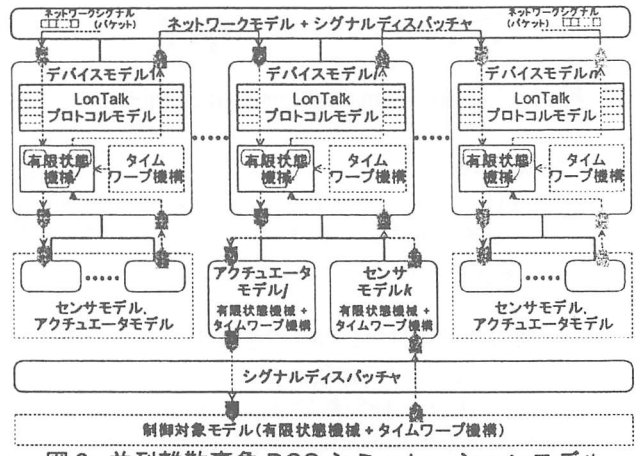

図 2. 並列離散事象 DCS シミュレーションモデル

3. 本研究で提案する LonTalk プロトコルモデル

デバイスモデル間のフレーム通信を実現するために，本研究では，前報の LonTalk プロトコルモデルを図 3 のように拡張する．図 3 のモデルは，LonTalk を搭載した LonWorks 用 VLSI（Neuron Chip）に基づきモデル化された，以下の 3 つの CPU モデルから構成されている．

- **Application CPU モデル**: 有限状態機械から出力されたシグナルを APDU（Application Protocol Data Unit）へ変換し，それをアプリケーションバッファへ格納する．
- **Network CPU モデル**: APDU を NPDU（Network PDU）へ変換し，それをネットワークバッファへ格納する．送信リトライや複製検出を行なうために，トランザクションレコードと送信・受信レコードの管理も行う．
- **MAC CPU モデル**: NPDU をフレームへ変換した後，*p*-persistent CSMA[3] に基づいてフレームの送信時刻を計算し，フレームをネットワークモデルへ送信する．

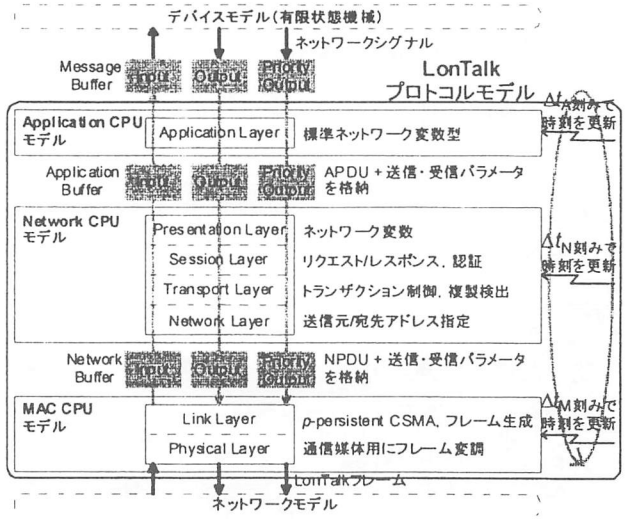

図 3. 本研究で提案する LonTalk プロトコルモデル

図 3 における各バッファは，デバイスモデルや CPU モデル間で共有されるキューであり，各バッファ内の PDU は送受信時刻順に格納される．各 CPU モデルは局所仮想時計を持っており，その時刻を Δt_A，Δt_N，Δt_M 刻みで更新する事によってパイプライン動作をシミュレートする．

4. タイムワープ機構による DCS の並列離散事象シミュレーション

並列離散事象 DCS シミュレーションを実現するため，本研究では，図 4 に示すように，LonTalk プロトコルモデルとタイムワープ機構をデバイスモデルへ組み込む．議論を簡略化するために，図 4 では優先出力バッファを省略している．本研究において，送信済みのシグナルの作用を取り消すシグナルは，アンチシグナルと呼ばれる．

図 4 において，各モデルの LVT は互いに異なる時刻を刻むが，PDU の通信によって，ゆるやかに同期する．

巻き戻し発生によるシグナルの取り消しに備えるために，各 CPU モデルとデバイスモデルは固有の入力キューと出力キューを持っている．ここで，巻き戻しを引き起こすような LonTalk フレームをデバイスモデルが受信した時の巻き戻しアルゴリズムは，以下のようになる．

(1) ネットワークモデルから仮想受信時刻 T_F のフレームが MAC CPU モデルへ到着する．ここでは $T_F < LVT_M$ なので，LVT_M を T_F に最も近い Δt_M 刻みの時刻へ巻き戻した後，仮想受信時刻 T_{Ni} の NPDU を生成し，そのアンチ NPDU を入力キューへ格納しておく．
(2) NPDU をネットワーク入力バッファへ格納する．
(3) $T_{Ni} < LVT_N$ なので，Network CPU モデルは，LVT_N を T_{Ni} に最も近い Δt_N 刻みの時刻へ巻き戻した後，仮想受信時刻 T_{Ai} の APDU を生成し，アンチ APDU を入力キューへ格納しておく．
(4) APDU をアプリケーション入力バッファへ格納する．
(5) $T_{Ai} < LVT_A$ なので，Application CPU モデルは，LVT_A を T_{Ai} に最も近い Δt_A 刻みの時刻へ巻き戻した後，仮想受信時刻 T_{Mi} のシグナルを生成し，アンチシグナルを入力キューへ格納しておく．
(6) シグナルをメッセージ入力バッファへ格納する．

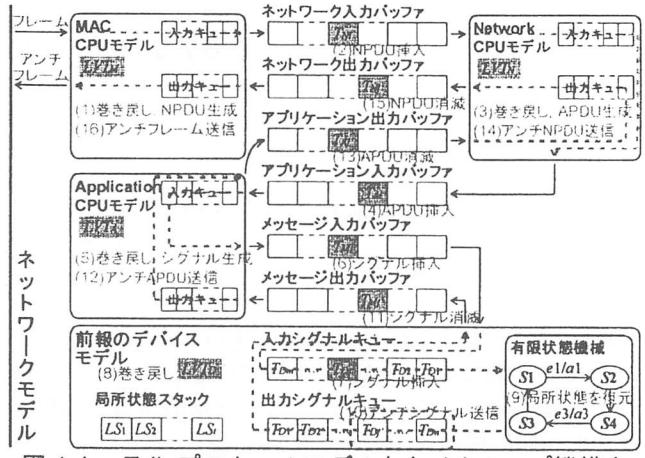

図 4. LonTalk プロトコルモデルとタイムワープ機構を持つデバイスモデル

(7) デバイスモデルはメッセージ入力バッファの先頭にあるシグナルを，入力シグナルキューへ格納する．
(8) $T_{Di} < LVT_D$ なので，デバイスモデルは，LVT_D を T_{Di} に巻き戻す．
(9) 有限状態機械の状態を T_{Di} 以前の状態へ復元する．局所状態スタック内の T_{Di} 以降の状態は破棄される．
(10) T_{Di} 以降に送信されたネットワークシグナル（仮想送信時刻 $T_{Di'} \sim T_{Dn'}$）のアンチシグナルを送信する．
(11)〜(16) 各 CPU モデルが出力キューのアンチ APDU・NPDU・フレームを送信する事により，各出力バッファの PDU が連鎖的に消滅し，最終的にネットワークモデルへアンチフレームが送信される．

DCS シミュレーションモデルを図 4 のように改良する事により，前報のモデルよりも高精度に（論理時刻の単位は[ns]），Acknowledgement /Reminder[3] 送信によるトラフィック増大，同時送信によるパケット衝突，抽選漏れによる通信遅れなどの特性が定量的に評価可能となった．

5. まとめと今後の課題

本報では，前報で提案した DCS シミュレーションモデルの LonTalk プロトコルモデルとタイムワープ機構を改良する事により，各種通信サービスにおけるシミュレーションの精度を向上させる事ができた．今後は，実機規模の DCS に関する通信特性を評価する予定である．

参考文献

[1] 戸村，金井，岸浪，“デザインパターンとタイムワープ機構に基づく FA 用分散制御システムの並列離散事象シミュレーション”, 2003 年度精密工学会春季大会学術講演会講演論文集, p.155, 2003.
[2] D. R. Jefferson, “Virtual Time”, ACM Transactions on Programming Languages and Systems, Vol. 7, No. 3, pp. 404-425, 1985.
[3] “EIA/CEA-709.1-B Control Network Protocol Specification”, Electronic Industries Alliance, 2002.
[4] T. Tomura, S. Kanai, T. Kishinami, “A Cooperative Simulation Mechanism of Distributed Control Systems Based on Object-Oriented Design Patterns”, ISORC 2003 Proceedings (ISBN 0-7695-1928-8), pp. 83-90, 2003.