2010 年度第 6 回情報処理学会東北支部研究会(山形大学)
資料番号 10-6-A3-2

リアルソフトウェアのネットワークシミュレーションへの組み込み方法

○竹田 勇人* 平中 幸雄 武田 利浩

山形大学大学院理工学研究科

1. はじめに

近年ネットワークは急速に発展し，インターネットを始めとするネットワークサービスが普及している．その背景で，あらゆるネットワークシステムの研究開発には大規模化・複雑化が進んでいる．その開発過程にはシミュレーションと実装の段階が存在し，シミュレーションでは，システムを実際に使われているネットワークを使ってシステムの実験をするわけではない．その理由は，まだ開発段階の不安定なもの，不安要素のあるものを実際のネットワークに流すと，そのネットワークに大きな影響を与える可能性があるからである．そこで，実際のネットワークを模した仮想ネットワークを構築して実験を行う必要があり，そこで用いられるのがネットワークシミュレータである．

そしてシミュレーションの段階の後に実利用できる形にするための実装の段階がある．シミュレーション後の実装といっても，シミュレーションに用いたプログラムをそのまま用いることができるわけではない．実利用の環境に合わせた実装やシミュレーションされた機能がシステムの一部分であることもある．そのため，シミュレーションを経ていても実装時にも大規模なプログラミングが必要になる場合が存在する．このとき，シミュレーションと実装の 2 段階で大規模なプログラミングを行うことになる

これはシステム開発にとって開発期間の増大やプログラミングの二度手間といった問題に繋がる．この点に注目し，上記問題が生じないシステム開発が可能なネットワークシミュレーション方法開発の研究 [1] が行われている．

1.1 研究目的

本研究ではシミュレーションしたプログラムをそのまま実装に用いることができる仕組みを開発する．しかし上で述べているようにシミュレーション用に作成したプログラムをそのまま実装するのは難しい．従って、本研究ではシミュレーションと実装をシームレスに行える仕組み，さらに言えば，シミュレータ自体でシステムの開発が行えるようにし，シミュレーションと実装を同時の行うことができる仕組みを開発する．

本研究では実装されるソフトウェアをリアルソフトウェアと呼ぶ．リアルソフトウェアをネットワークシミュレーションに組み込む方法を開発することが目的であり，リアルソフトウェアをネットワークシミュレーションに組み込む際の問題点を提示し，その解決案を提案していく．

1.2 現状の問題点

リアルソフトウェアをネットワークシミュレーションに組み込む際に生じる問題点として，実時計とシミュレーション時計の違いから通信の順序が入れ替わったり，ずれが発生する可能性がある．

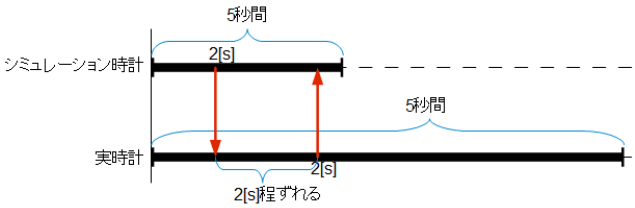
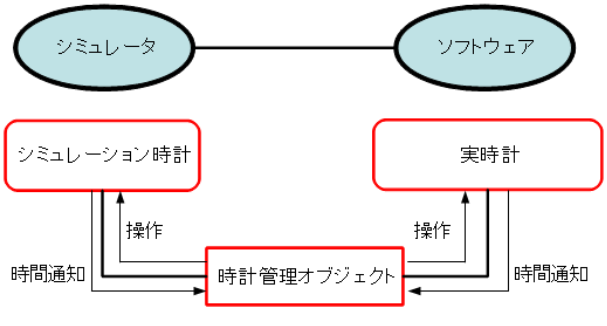


図 1:問題点の例

2. 提案方法

本研究にて提案する方法は，実時計とシミュレーション時計の双方を管理するオブジェクトを作成し，それぞれの時計を同期させることによって時計のずれを無くし整合性の保たれたシミュレーションを行うというものである．



TMOが双方の時計の時間を通知してもらい、操作する

図 2:提案方法

時計管理に伴う時刻同期の方法について考察すると，

- 実時計がシミュレーション時計より早い
- シミュレーション時計が実時計より早い

* 現在，株式会社エクサ

の 2 つの場合が考えられる．前者の場合，シミュレーション時計に合わせようとし時計をロールバックするのはリアルソフトウェアの場合現実的ではない．そのため，シミュレーション時計が追いつくまで実時計を停止することによって時刻同期を行う．後者の場合では，シミュレーション時計はシミュレータで管理されるもので実時計の時間までロールバックすることは可能である．しかし，リアルソフトウェアと相互通信を行う場合，シミュレーションの正確性に影響が出てしまう．そのため，シミュレーション時計も前者と同様に実時計が追いつくまでシミュレーション時計を停止する．このように考えると時計はシミュレーション中に最も遅い時計に合わせて同期していくことになる．詳しい同期方法については第 4 章にて記す．

3. 実現方法提案

本提案方法は実時計を止めることによってリアルソフトウェアを止める．そこで，リアルソフトウェアを動作させる専用のプラットフォームを仮想マシン上に用意する．そして，仮想マシン上のゲスト OS の時計を実時計と見立てて操作していく．

本研究方法に仮想マシンを用いるに当たって，仮想マシンに求める 2 つの必要条件がある．

- ゲスト OS の時計がホスト OS の時計と同期しないこと
- 仮想マシンを外部制御できること

1 つ目の必要条件は，ホスト OS とゲスト OS の時計が同期してしまうと，時計の一時停止・再開を行うときにゲスト OS の時計が一時停止した時刻から再開されずに，再開した時のホスト OS の時刻から再開されてしまうことになる．これでは，本提案方法のおいてシミュレーションを行った際に空白の時間が存在してしまう可能性があるそれではシミュレーションの信頼性低下に繋がってしまう．ホスト OS とゲスト OS の間で時計が非同期であればこのような問題は発生しない．

2 つ目の必要条件は，本提案手法ではシミュレータ側から仮想マシンを操作できることを前提としているので，仮想マシンと通信できるだけでなく操作する必要がある．

3.1 シミュレーションアーキテクチャ

シミュレーションアーキテクチャは，先行研究[1]を元にリアルソフトウェアを組み込む形になる．具体的にはリアルソフトウェアを仮想マシン上，それ以外を実 PC 上に配置する．そして，時間管理オブジェクトはシミュレーション核：sim と仮想マシンの間に配置する(図 3)．

4. 量子同期方法[2]

提案方法では時間を量子という単位に分割し，その量子毎に時刻同期を行う量子同期を用いる．量子同期を用

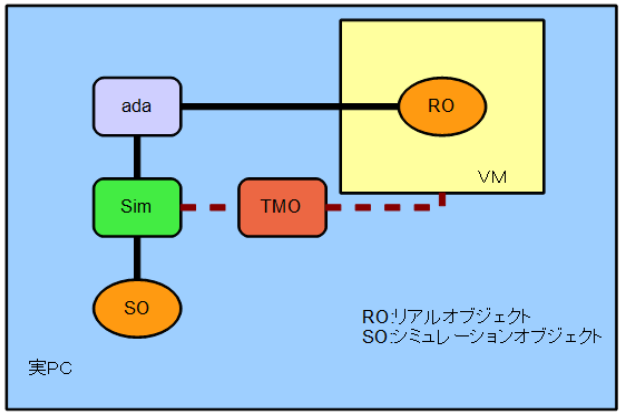


図 3:シミュレーションアーキテクチャ

いると以下に示すような効果が期待できる．

- ノード間の時間誤差の上限を定めたシミュレーションが可能
- 量子時間を短くすれば時間誤差をいくらでも小さくできる

図 4 は量子化を行ったときの例である．量子化前はずれが大きくなっていたが，量子化を施すことによってずれの最大値は量子時間となり，ずれを軽減できる．

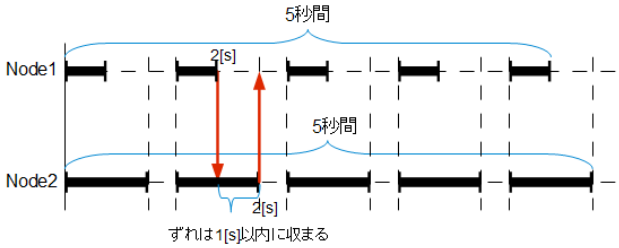


図 4:量子化

4.1 量子化時間長⊿T

量子化時間長とは時間を量子として区切る時間の長さのことである．シミュレーションするノードは量子化時間長で設定した時間分まで時計を進行することができる．

4.2 量子同期での通信方法

量子同期での通信は，通信が発生した直後の量子の始まりに伝達されるものとする．このようにすることで通信の扱いがシミュレーション動作途中で伝達しなくてよいので実現が容易になる．

- 同期タイミング
  通信の順序制御を行ったり，前量子から次量子への移行処理を行う．同期タイミングで掛かった時間はシミュレーションのオーバヘッドとなる．

4.3 通信の順序制御

通信されるパケットには送信元ノード名，送信先ノード名，送信時刻が含まれている．シミュレータはノード間遅延時間，量子化時間長といった情報を持っており，これらの情報から送信されたパケットの受信タイミングを予測し，渡される量子と順番を決定する．

$$\text{受信される量子No} = \frac{\text{送信時刻}+\text{遅延時間}}{\text{量子化時間長}} + 1$$

(小数点以下第1位切り上げ)

4.4 本提案方法の特徴

本提案方法は通信の送信時刻と遅延時間から受信時刻の理想値を算出し，受信される順序を判断しているので基本的には受信順序の入れ替わりが発生しない．しかし送受信の繰り返しが起こる場合には通信順序が入れ替わる可能性がある．

また，シミュレーション中のノードはそれぞれが依存する時計のスピードの違いを意識する必要はない．しかし，量子化時間長の長さによって正確性や所要時間などシミュレーション結果に与える影響が変わってくる．

5. 検証

量子化時間長を変化させ，提案方法のシミュレーションの忠実性，正確性にどのように影響を与えるか検証する．

- 忠実性
  通信順序の入れ替わりが発生していないことを忠実性が高いとする．通信の入れ替わり度を測定し，忠実性を計る．入れ替わり度が低ければ忠実性が高い．

$$\text{入れ替わり度} = \frac{\text{入れ替わった通信の数}}{\text{全通信の数}} \times 100[\%]$$

- 正確性
  通信の受信時間の正確性を計る．受信時間の誤差を二乗誤差を用いて算出する．誤差が小さければそれだけ正確性が高い．

$$Z = \sqrt{\frac{\sum_{i}^{n}(R_i - S_i)^2}{n}}$$

(Z: 平均二乗誤差平方根, R_i: 理想値, S_i: シミュレーション値, n: 通信数)

5.1 検証方法

検証に用いるネットワークモデルは図 5 に示すようなノード数 3 のネットワークである．

Node1-2 間遅延時間 L1_2 と Node3-2 間遅延時間 L3_2 の遅延時間は以下の示す 4 通りの組み合わせである．

(L1_2, L3_2) = (90, 60), (120, 60), (120, 30), (120, 10)

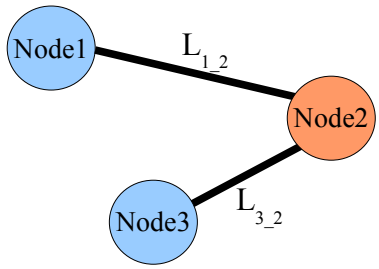


図 5:モデルネットワーク

量子化時間長⊿T は以下のように変化させる．

⊿T = 1, 10, 20, 30, 50, 60, 75, 100, 150, 300[ms]

<シナリオ>

- 300[ms]までに送信されたパケットが受信されるまでを測定
- Node1→2: 20[ms]毎に通信
- Node3→2: 40[ms]毎に通信

5.2 検証結果

検証結果を表 1，図 7 に示す．表 1 から量子化時間長が変化しようとも通信順序の入れ替わりが発生していないことがわかる．本提案方法では通信の入れ替わりが発生せずに忠実性の高いシミュレーションが可能であることを期待しており，その期待通りの高い忠実性を確認することができた．

また，シミュレーションに掛かる時間も測定した結果量子化時間長が最短の 1[ms]のときに最も時間が掛かり 1557[ms]掛かる結果となった．量子化時間長が最大の 300[ms]のときは 603[ms]掛かっており，量子化時間長は短すぎても長すぎても所要時間が掛かり過ぎる結果となった．しかし，20≦⊿T≦150 の範囲では所要時間は大きな差がない結果となった．

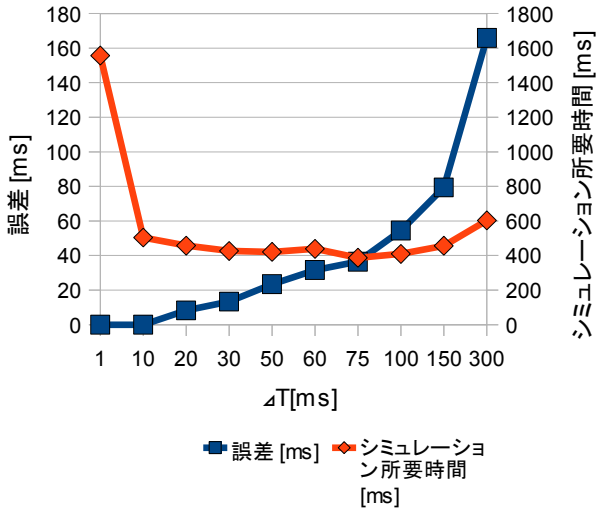


図 6:正確性とシミュレーション所要時間

表 1:検証結果　(L_{1_2},L_{3_2})=(90, 60)の場合

理想値	⊿T=1	⊿T=10	⊿T=20	⊿T=30	⊿T=50	⊿T=60	⊿T=75	⊿T=100	⊿T=150	⊿T=300
1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3
	中省略									
21	21	21	21	21	21	21	21	21	21	21
22	22	22	22	22	22	22	22	22	22	22
入れ替わり度[%]	0	0	0	0	0	0	0	0	0	0
受信時間誤差[ms]	0	0	8.25	13.31	23.45	31.56	36.46	54.39	79.37	165.84
所要時間[ms]	1557	504	457	426	421	438	387	409	456	603

本提案方法の受信時間誤差を考える上で，最初に誤差が発生した以降に特に大きく誤差が発生している量子化時間長を誤差増加点として考え，発生した誤差増加点の順番に第 1 誤差増加点，第 2 誤差増加点，としていく．今回の一例では⊿T=50，60，100，150，300[ms]が誤差増加点にあたる．⊿T=100[ms]以降では，全ての量子化時間長が誤差増加点にあたる．

次に正確性の結果を表 2 に示す．遅延時間(L1_2，L3_2)の 4 つの組み合わせによる受信時間誤差の比較である．これらを見ると，遅延時間の組み合わせによって大きく受信時間誤差の増加傾向が変わるわけではないことがわかる．

表 2:遅延時間の組み合わせと正確性の比較

	誤差[ms]			
⊿T[ms]	(90,60)	(120,60)	(120,10)	(120,30)
1	0	0	0	0
10	0	0	0	0
20	8.25	0	1.78	1.78
30	13.31	10.66	12.79	13.31
50	23.45	23.46	23.45	25.67
60	31.56	25.58	27.88	29.77
75	36.45	36.64	40.37	41.69
100	54.39	48.43	50.05	52.87
150	79.37	79.37	84.1	81.74
300	165.84	159.77	160.84	167.81

そのため，最適な量子化時間長の長さは，遅延時間の違いによらず定めることができると考えられる．各遅延時間の組み合わせから，それぞれの遅延時間の最大公約数に当たる量子化時間長の長さであるときが，第 1 もしくは第 2 の誤差増加点の手前になっており，最適な量子化時間長を考察する上で 1 つの指標になると考えられる．

6. まとめ

本研究ではリアルソフトウェアをネットワークシミュレーションに組み込むことを実現するために，その障害となる問題点を挙げ，その問題を解決すべく研究を行ってきた．実時計とシミュレーション時計に着目し，双方の時間の進むスピードが違うことからその時計の同期が必要であり，同期方法として時間を量子化しながらシミュレーションを進めていく量子同期方法を検証した．

量子同期方法を用いるのに当たって，その量子に区切る単位である量子化時間長の大きさをどのようにするのが最適なのか，様々な量子化時間長の長さを使って検証を行い，最適な量子化時間長を決定する上での 1 つの指標を見つけることができた．また，検証の結果から量子化同期方法を用いることによって，シミュレーションがとても高い忠実性を持てることが分かった．

参 考 文 献

[1]竹田勇人，武田利浩，平中幸雄，”UCF オブジェクトで構成するネットワークシミュレータ”，FIT2009（第 8 回情報科学技術フォーラム），第 4 分冊，L-015，pp.165-166，2009.9.3

[2]Ayose Falc´on, Paolo Faraboschi, Daniel Ortega, “An Adaptive Synchronization Technique for Parallel Simulation of Networked Clusters”, IEEE International Symposium on Performance Analysis of Systems and software, ISPASS 2008., pp.22-31, 2008.