修 士 論 文

大規模シミュレータ実現に向けた
軽量クランアントモジュールの作成

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

鈴木　章弘

2015 年 3 月

修 士 論 文

大規模シミュレータ実現に向けた軽量クランアントモジュールの作成

指導教員　丹 康雄 教授

審査委員主査　丹 康雄 教授
審査委員　篠田 陽一 教授
審査委員　知念 賢一 特任准教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

1310033 鈴木　章弘

提出年月: 2015 年 2 月

概 要

M2M や IoT 等、多数の機器が接続・通信を行いサービスと連携するシステムの運用が目前に迫っている。その様なシステムの中には、一つの街全体がネットワークで繋がり、互いに通信を行う様な非常に大規模なシステムが存在する。上記のような大規模システムの実際の運用前には、同様の規模かつ精度の高いシミュレーションを実施する必要がある。しかし、大規模なシミュレーション対象の環境の模擬には多大な計算資源が必要とされるため、シミュレーションの実施には膨大な費用を要する。そのため、シミュレーションに必要な資源が少ないシミュレータの開発が求められている。

また、本研究室では、数万世帯のスマートハウスを含む大規模スマートコミュニティシミュレータの開発が行われた。スマートコミュニティ内には数多くのクライアントが存在するため、正確なシミュレーションを行う場合、個々の特徴を模擬したクライアントのモデルを作成する必要があるが、必要な計算資源が膨大になるという問題がある。そこで、先行研究では、同様の特徴を持つ一定数のスマートコミュニティを一つのモジュールとして扱い、モジュール内をモデル化して計算することによりシミュレータのスケーラビリティを実現した。しかし、通信システムにおけるサーバ・クライアントの外部通信から得られる情報のみを利用し、クライアントの複雑なアプリケーションレイヤプロトコルの通信を適切に推測することは困難であった。

本研究では大規模な通信システムのシミュレーションに必要な計算資源の低減を実現する軽量クライアントモジュールの作成を目的とする。軽量クライアントモジュールを適用するシミュレーションの対象として想定する大規模通信システムは、数万台のクライアントと複数のサーバから構成され、機器同士が互いに通信を行うシステムである。また、このような大規模システムの高精度なシミュレーションを行うためには、複雑な動作を要するアプリケーションレイヤプロトコルの通信の正確な模擬が求められる。本研究では大規模システムの高精度なシミュレーションに用いることの出来る軽量クライアントモジュールを以下の方法で作成した。大規模システムのシミュレーションの低資源化を実現するために、軽量クライアントモジュールは模擬対象アプリケーションのクライアントの外部通信動作のみを模擬する。模擬アプリケーションの外部通信以外の内部的処理を省くことで低資源化を図った。軽量クライアントモジュールを作成する際には、模擬対象のアプリケーションの状態機械を作成し、そこに入出力における特徴量を適用し、アプリケーションのモデル化を行った。このモデルからクライアントモジュールを作成することで模擬対象のアプリケーションレイヤプロトコルの通信動作を再現可能した。この方法により、軽量クライアントモジュールを使用したシミュレーションの精度を向上させる。状態機械の作成においては、模擬対象アプリケーションの外部通信観測結果のみに基づいて作成した場合、クライアントモジュールの模擬通信の精度が低く、作成にも多くの時間を要し非効率的であるため、模擬対象アプリケーションの作成者からの情報提供を受けて状態機械の作成を行った。状態機械に適用する特徴量とは、モデルの入力情報と出力情報の間に

現れる関係を示すものであり、ネットワークトラフィックモデルにおけるスループットやパケットサイズの分布等を示す。この特徴量は模擬対象アプリケーションの外部通信観測結果から得たものを利用した。また、対象アプリケーションの動作基板として大規模ネットワーク実証環境 StarBED を利用した。エミュレーションベースで動作するアプリケーションの模擬を行うことで、高い精度のモジュールを実現する。

以上の方法によって作成した軽量クライアントモジュールの軽量化についての評価を行った。評価のため、作成したクライアントモジュールと模擬対象アプリケーションのプロファイリングを行った。プロファイリングにはパケットキャプチャツールや Java 仮想マシンのモニタリングツールを用いた。パフォーマンスの指標として CPU 使用率、ヒープメモリ使用量、スループット、レスポンスタイム等の調査を行った。結果として、サーバとの通信が可能な軽量なクライアントモジュールを実現することができた。元のアプリケーションのメモリ使用量を軽量クライアントモジュールによって削減できた。

作成した軽量クライアントモジュールをプロセス・スレッドで一つのマシン上で並列動作させた。その結果、スレッドを利用して軽量クライアントモジュールを動作させることでメモリ 48GB のマシン 5 台で 5 万世帯の大規模シミュレーションが可能であることが確認できた。

クライアントモジュールのさらなる軽量化が今後の課題である。Java ではガベージコレクションが自動で行われるため、メモリ解放を自主的に管理するプログラムを作成することで軽量化が実現できる可能性がある。作成したモデルは他のアプリケーションにも適用可能な設計であるため、今後は様々なアプリケーションに適用させることも課題である。

目 次

図 目 次

表 目 次

第1章　はじめに

本章では研究背景、研究目的、本論文の構成を以下に示す。

1.1　研究背景

M2M や IoT 等、多数の機器が接続・通信を行いサービスと連携するシステムの運用が目前に迫っている。その様なシステムの中には、一つの街全体がネットワークで繋がり、互いに通信を行う様な非常に大規模なシステムが存在する。例示した様な大規模システムの実際の運用前には、同様の規模かつ精度の高いシミュレーションを実施する必要がある。しかし、シミュレーション対象の環境の模擬には多大な計算資源が必要とされるため、シミュレーションの実施には膨大な費用を要する。そのため、シミュレーションに必要な資源を低減したシミュレータの開発が求められている。

また、本研究室では、数万世帯のスマートハウスを含む大規模スマートコミュニティシミュレータの開発が行われた。スマートコミュニティ内には数多くのクライアントが存在するため、正確なシミュレーションを行う場合、個々の特徴を模擬したクライアントのモデルを作成する必要があるが、必要な計算資源が膨大になるという問題がある。そこで、先行研究では、同様の特徴を持つ一定数のスマートコミュニティを一つのモジュールとし、て扱い、モジュール内をモデル化して計算することによりスケーラビリティを実現した。しかし、通信システムにおけるサーバ・クライアントの外部通信から得られる情報のみを利用し、クライアントの複雑なアプリケーションレイヤプロトコルの通信を適切に推測することは困難であった。

1.2　研究目的

本研究では、大規模な通信システムのシミュレーションの計算資源低減を実現する軽量クライアントモジュールの作成を目的とする。

シミュレーションの対象として想定する大規模通信システムは数万台のクライアントとサーバから構成されており、機器が互いに通信行うシステムであり、そのシミュレーションに必要な計算資源量は膨大になる。そこで、模擬対象システムにおけるサーバ・クライアントの外部通信のみを模擬する軽量クライアントモジュールを実現し、シミュレーションに必須となる計算資源量を低減する。また、大規模システムの高精度なシミュレーショ

ンを行うためには、複雑な動作を含むアプリケーションレイヤプロトコルの通信の正確な模擬が求められる。本研究では、その問題に対処する手段として、実際のソースコードやプロトコルの仕様書の提供をシステム作成者から受け、その情報を基にアプリケーションレイヤプロトコルに至るまでのシステムの通信の模擬を行う。更に、作成する軽量クライアントモジュールに、模擬対象システムの通信における状態機械と特徴を適用することで、より精度の高いクライアントモジュールを実現する。

以上の方法を用いて、軽量かつ精度の高いクライアントモジュールの作成を行う。また、現在は数万台規模のシミュレーションが求められているため、そのような大規模シミュレーションが可能で、必要な計算資源が少ないクライアントモジュールの作成を行う。

1.3 本論文の構成

本論文は以下に示す構成である。

- 第 1 章
  - 研究の背景と目的、本論文の構成を示す。
- 第 2 章
  - 軽量クライアントモジュールの設計についての説明を行う。
- 第 3 章
  - シミュレータの作成の際に利用した大規模ネットワーク実証環境である StarBED に関しての説明を行う。
- 第 4 章
  - 軽量クライアントモジュールの適用対象となるアプリケーションについて述べる。
- 第 5 章
  - 軽量クライアントモジュールの実装に関して述べる。。
- 第 6 章
  - 作成した軽量クライアントモジュールについての評価と考察を述べる。
- 第 7 章
  - 本論文におけるまとめを述べる。

第2章　軽量クライアントモジュールの設計

本章では、作成する軽量クライアントモジュールの設計に関して記述する。

2.1　作成手順

本研究で作成する軽量クライアントモジュールの作成手順を以下に示す。

- 模擬対象アプリケーションのモデル作成
- 状態機械の作成
- 模擬対象の通信観測から特徴量抽出
- モジュール作成

これらの手順によって軽量クライアントモジュールの作成を行い、実際のサーバとの通信が可能かつ、必要となるリソースが少ないモジュールを実現する。

上記手順の詳細について以下に示す。

2.2　模擬対象アプリケーションのモデル作成

アプリケーションモデルの作成を行う。まず、模擬対象の選定を行う。対象のアプリケーションが決定した後は、

そのモデルの作成を行う。モデルの作成には、アプリケーション作成者からのアプリケーションに関する情報と通信の観測結果を用いる。ここで、作成者から提供される情報は、アプリケーションのプロトコルの仕様書等である。

通信の観測結果のみを利用したアプリケーションの通信動作の正確な模擬は、アプリケーションレイヤプロトコルの動作の詳細を明確にすることが困難なため、難しい。そのため、本研究では情報提供を受けることを前提とする。

アプリケーションの通信動作の状態遷移図の作成を行う。この状態遷移図を作成することで、モジュールで特定のパケットを受信した際に送信すべきパケットの決定を行い、モジュールの正確な通信の模擬を実現する。

また、状態遷移図を作成することで取得すべき特徴量に関しても明らかになる。

2.3　状態機械の作成

アプリケーションの通信動作の状態機械の作成を行う。作成した状態機械をモジュールに適用することで、モジュールで特定のパケットを受信した際に送信すべきパケットの決定を行い、正確な通信の模擬を実現する。また、状態遷移図を作成することで取得すべき特徴量を明確にすることが可能である。

状態機械の作成には通信観測から得られる情報だけでなく、アプリケーションの仕様書からの情報を利用し、効率的に作成を行う。

以下に状態機械の作成方法と各方法に関する利点と欠点を示す。

- ソースコード・バイナリコードの解析から状態機械を作成する方法
  - 利点
    解析ツール等を使用することによりソースコード・バイナリコードから状態機械を自動的に抽出可能である。
  - 欠点
    ソースコード・バイナリコードを解析するために特別な解析技術を用いる必要がある。
- アプリケーションの仕様書から状態機械を作成する方法
  - 利点
    ソースコード・バイナリコードを解析するための特別な解析技術を必要としない。
  - 欠点
    アプリケーションの仕様書から新たに別のプログラムを作成する必要がある。

上記の各状態機械作成方法の利点と欠点を考慮し、本研究ではソースコード等の特別な解析技術を必要としない、アプリケーションの仕様書の情報から状態機械を作成する方法を用いた。

2.4　模擬対象における特徴量の抽出

本研究では模擬対象のアプリケーションの通信観測や、アプリケーションの仕様書等の情報から特徴量の抽出を行い、軽量クライアントモジュールに適用する。図**??**に模擬対象の特徴量について示す。図に示すように特徴量とは、入力情報と出力情報の間の関係性を示すものである。この特徴量を模擬し、作成するクライアントモジュールに適用する。

模擬対象アプリケーションの特徴量について以下の図 2.1 示すようにいて以下に示す。

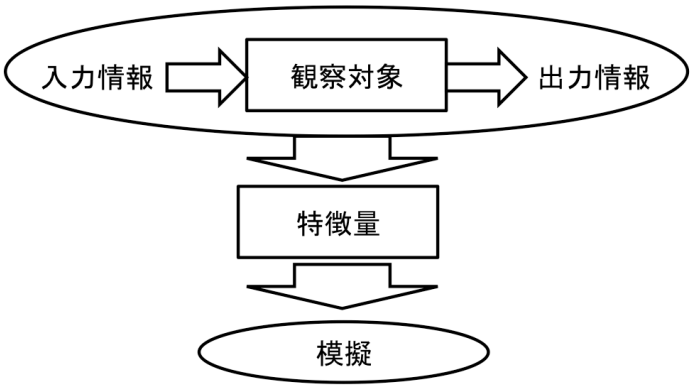

図 2.1: 模擬対象の特徴量

- 通信順序

  模擬対象が行う通信の順序である。通信における要求と応答では、特定のパケットを受信した際に、送信すべき特定のパケットを送る。この特徴量はそのような通信フローである。

- スループット

  単位時間あたりに送信するパケット数、または転送量である。

- パケットサイズ分布

  通信パケットにおけるパケットサイズの分布である。

- プロトコル分布

  使用されるプロトコルの種類の分布である。HTTP、STUN、TCP 等のプロトコルがその一例である。

- 通信頻度

模擬対象アプリケーションでは特定の通信を一定間隔で行う場合がある。そのような通信等の実施頻度である。

これらの特徴量をアプリケーションの通信の観察等から抽出し、軽量クライアントモジュールに適用する。

2.5 軽量クライアントモジュールが再現する通信

軽量クライアントモジュールは模擬対象となるクライアント上で動作するアプリケーションの通信動作を再現する。そのため、上記のような通信における特徴量等を機器間の通信観測から得て、モジュールで模擬する。このように軽量クライアントモジュールを作成し、模擬対象の通信観測から得られる通信のみを再現する。モジュール実装の際には、設定ファイルの読み込み、通信内容を解析するパーサ、ロギングや軽量クライアントモジュールに不要なデータを削除、利用頻度を少なくする。

また、軽量クライアントモジュールが発生させる通信のタイミングは外部観測結果から決定する。通信のタイミングは時間経過や、特定のパケットの受信に依存するものであると考えられる。

第3章　大規模ネットワーク実証環境 StarBED

本章では、軽量クライアントモジュールの作成に使用した大規模ネットワーク実証環境 StarBED について記述する。

3.1　StarBED の概要

StarBED は、ネットワーク上で動作するサービスを検証することを目的に開発された大規模なテストベッドである。StarBED は 1100 台の物理ノードとそれらを接続しているスイッチ群から構成される。StarBED には複数のノードを一括りとしたグループが複数存在し、各グループ毎にノードの機器の性能が統一されている。この環境上で、更に VM 等の仮想化技術を使用することにより、数万台規模のノードを有する環境のシミュレーションが実施可能である。また、各ノードは実験用ネットワークと管理用ネットワークの双方に属しており、各ネットワークに接続するためのインターフェースを有している。これらのネットワークは互いに独立しているため、通信による干渉が発生しない構成となっている。実験用ネットワークは実験を実施する際に用いられ、管理用ネットワークは OS やソフトウェアの導入する際に用いられる。シミュレーションを行う際にはシステムのネットワークトポロジを模倣する必要があるが、StarBED のネットワークは VLAN によって分割することが可能なため、シミュレーション対象となるシステムがどのようなトポロジを有していた場合にも対応可能である。また、StarBED では実験支援用ソフトウェアである SpringOS が導入されており、ノード環境の保存や配布が容易に行える。

3.2　ノードの構成

StarBED における各ノードの構成について、以下の表 3.1 に示す。表に示す通り、各ノードは名称がアルファベッドのグループに属しており、グループ毎にノードの性能が統一されている。

本研究で使用したノードはグループ I に属するノードである。

表 3.1: StarBED ノードの構成

グループ	F	G	H	I	J	K	L	N
モデル	NEC Express 5800/ 110Rg-1	Proside Amaze Blast neo920	HP DL320 G5	Cisco UCS C200 M2				DELL PowerEdge C6220
CPU	Intel Pentium4 3.2GHz	AMD Opteron 2.0GHz	Intel QuadCore Xeon X3350 2.66GHz	Intel6-Core Xeon X5670 2.93GHz×2				Intel 8-Core Xeon E5 2650 2.00GHz×2
メモリ	8GB	8GB/4GB	2GB×4	48GB				128GB
ストレージ	HDD: SATA 80GB×2	- - - -	HDD: SATA 160GB	HDD: SATA 500GB×2				HDD: 500GB×1 SSD: 200GB×4
NIC管理用 NIC実験用	1GbE×1 1GbE×4	1GbE×1 1GbE×1	1GbE×1 1GbE×2	1GbE×1 1GbE×4				1GbE×1 10GbE×1
台数	168	150	240	192	96	144	120	224
導入年	2006年	2007年	2009年	2011年				2013年

3.3 スイッチの構成

StarBED で使用されるスイッチについて以下の表 3.2、表 5.5 に示す。管理用ネットワークで使用されているスイッチ、及び実験用ネットワークで使用されるスイッチについてそれぞれを表に示す。

表 3.2: 管理用スイッチの一覧

スイッチ名	mggw1	mggw3	mggw4	mggw5	mggw6	mggw7
型式	7050Q-16	DGS3427 + DGS3450 ×10	DGS3427 + DGS3450 ×10	Nexus5010 + Nexus2248TP ×12	Nexus5010 + Nexus2248TP ×12	EX4200-24T + EX4200-48T ×9
収容機器	Mggw (3～7)	グループF	グループH	グループ I～L	グループ I～L	グループ N

表 3.3: 実験用スイッチの一覧

スイッチ名	exsw4	exsw6	exsw7	exsw8	exsw9	exsw10	exsw11
型式	Bigiron RX 16	Bigiron RX 16	Bigiron RX 16	Bigiron RX 32	MLXe-32	MLXe-32	QFX3100 x2 + QFX3600I x4 + QFX3500 x16
収容機器	グループ G	exsw 4,7,8, 9,10	グループ F	グループ F,H	グループ I〜J, exsw11	グループ K〜L, exsw11	グループ N

3.4 ネットワークの構成

StarBED におけるネットワークの構成について以下の図 3.1 に示す。機密性の高い実験を行う際には、ウイルス等による影響を防ぐため、外部のネットワークを遮断した上で行う必要がある。StarBED は StarBED の外部ネットワークとの接続がない、閉鎖的なネットワークを有しており、信頼性、機密性の高い実験の施行が可能である。StarBED の各グループのノードは管理用ネットワークと実験用ネットワークの２つのネットワークに属している。管理用ネットワークはノードに OS や、ソフトウェアを導入する際等に用いられ、このネットワークを使用して実験の準備を行う。一方実験用ネットワークは、実験を行う際に使用されるネットワークであり、管理用ネットワークやその他のネットワークから独立しているため、干渉を受けることなく信頼性の高い実験を実施可能にしている。

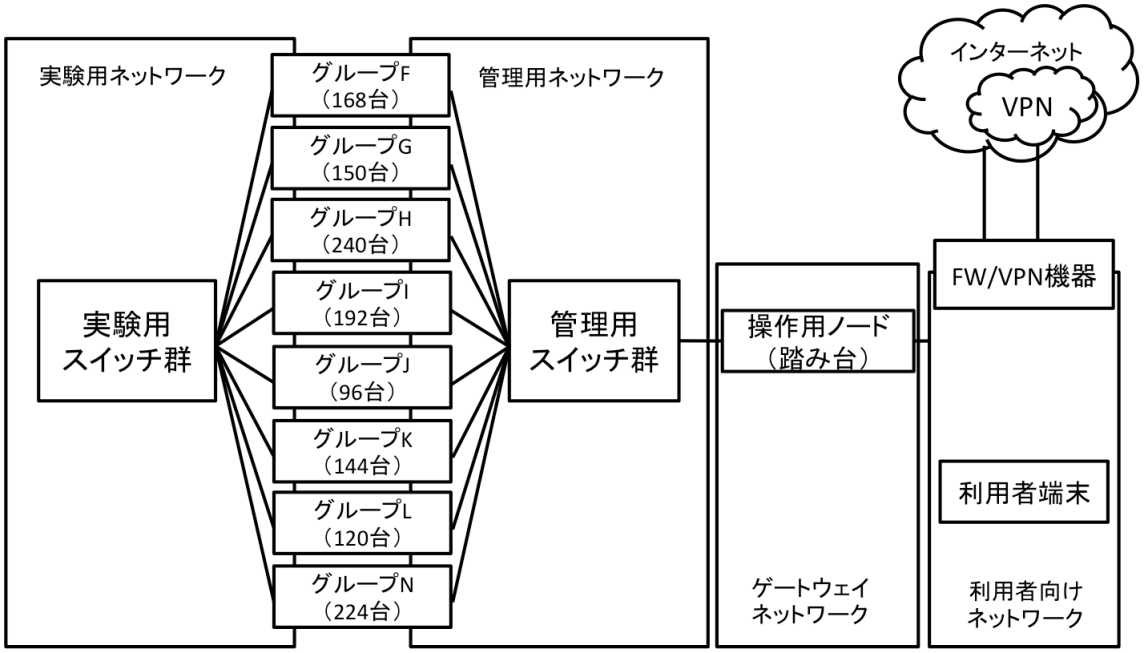

図 3.1: スターベッドのネットワーク構成

3.5 実験支援用ソフトウェア SpringOS

SpringOS とは、StarBED での実験を支援するソフトウェア群の総称である。StarBED で行われる大規模なシミュレーション等には数百台、数千台に及ぶノードを使用する場合があり、実験に必要な環境を各ノード上に構築する必要がある。そのような実験環境を作成する際の労力や手間を省くため、StarBED では実験支援用ソフトウェア SpringOS が導入されている。SpringOS を用いることで、雛形となるノードの環境を一台分作成した後、複製を行い他ノードへ配布する作業が容易になる。また、各ノードのネットワーク設定や仮想ネットワーク設定、ノードの電源管理等のノードの遠隔管理機能や、実験の設定を記述したシナリオに基づいてノードへの設定を自動的に実行する機能を提供する。

SpringOS が提供する機能を以下に示す。

- ノードの電源管理
- ネットワークの設定 (仮想ネットワーク設定)
- ノードのディスクイメージ複製
- ノードへのディスクイメージの配布
- ノードでのシナリオ実行

3.5.1 ユーザインターフェース blancket

blancket とは、SpringOS の機能を利用するためのユーザインターフェースである。blancket はユーザ設定ファイルとファシリティ設定ファイルからの情報を読み込み、SpringOs の各機能を実行する。blancket で使用するユーザ設定ファイルには動作させるノードのユーザ名やパスワード、作成するディスクイメージ名等が記述されており、ファシリティ設定ファイルには StarBED を使用するユーザが共通で利用する内容が記述されている。以下に blancket で使用可能な SpringOS の機能を示す。

(1) ノードの電源管理

SpringOS には多数のノードの電源管理を一括に行う機能が存在する。SpringOS のユーザインターフェースである blancket におけるノードの電源管理に使用するコマンドの一覧を以下に示す。

- poweron

  ノード名を指定し、ノードの電源を入れる。

- poweroff

  ノード名を指定し、ノードの電源を切る。

- powerstatus

  ノード名を指定し、ノードの電源状態の確認を行う。

- powerreset・reboot

  ノード名を指定し、ノードの再起動を行う。

(2) ネットワークの設定

StarBED で実験を行う際には、実験環境を構築するために事前に各ノードに対してネットワークの設定を行う必要がある。ネットワーク設定に使用したネットワークスイッチの設定コマンドについて以下に示す。

- activateport

  ノード名とインターフェースの番号を指定し、ノードの実験用ネットワークインターフェースに接続される実験用スイッチのポートを有効にする。

- deactivateport

  ノード名とインターフェースの番号を指定し、ノードの実験用ネットワークインターフェースに接続される実験用スイッチのポートを無効にする。

- mkvlan

  指定した VLAN ID の VLAN を新規作成する。

- rmvlan

  指定した VLAN ID の VLAN を削除する。

- joinvlan

  VLAN ID とノード名、実験用ネットワークインターフェースを指定し、ノードの実験用インターフェースが接続されるスイッチのポートを指定番号の VLAN に所属させる。VLAN におけるタグの有無を指定可能である。

- leavevlan

  VLAN ID とノード名、実験用ネットワークインターフェースを指定し、ノードの実験用インターフェースが接続されるスイッチのポートを指定番号の VLAN から除外する。

(3) ノードの複製、ディスクイメージの配布

SpringOS の機能を利用するためのユーザインターフェースである blancket によって、ノードの複製、ディスクイメージをノードへ配布可能である。この作業を行う際には、ノードを複製する getdisk、複製イメージを配布する distdisk コマンドを使用する。各コマンドの詳細について以下に示す。

- getdisk

  指定されたノードのディスクイメージを保存するコマンドであり、ストレージ内のパーティションを指定したディスクイメージの複製、またはストレージ全体の複製を行う。

- distdisk

  getdisk 等で作成されたディスクイメージを指定のノードに配布するコマンドである。指定ノードのパーティションへのディスクイメージの書き込み、ストレージ全体への書き込みを選択可能である。distdisk を使用することで複数のノードへイメージの一括配布が可能となる。

3.6 実験環境構築手順

前述のネットワーク設定やノードの複製、ディスクイメージの配布等の SpringOS の機能を用いて実験環境の構築を行った。実際には以下の手順で実験環境の作成を行った。

1. OS のインストール

   ディスクイメージの雛形を作成するノードへ OS のインストールを行う。

2. 雛形ノードの環境調整

   実験に使用するソフトウェアの導入やパッケージ管理ソフトのアップデートをイメージ配布の事前に行い、実験環境構築に要する時間を削減する。

3. getdisk・distdisk によるノードへのディスクイメージの配布

   getdisk で作成した雛形のディスクイメージを取得し、distdisk によって各ノードへディスクイメージを配布する。

これらの手順によって実験環境の作成を行った。

第4章　模擬対象アプリケーション

軽量クライアントモジュールが通信動作を模擬する対象となるアプリケーションの詳細について記述する。

4.1　模擬対象システム

本研究において作成する軽量クライアントモジュールが動作を模倣するシミュレーションアプリケーションについて述べる。アプリケーションの概要図を図 4.1,4.2 に示す。

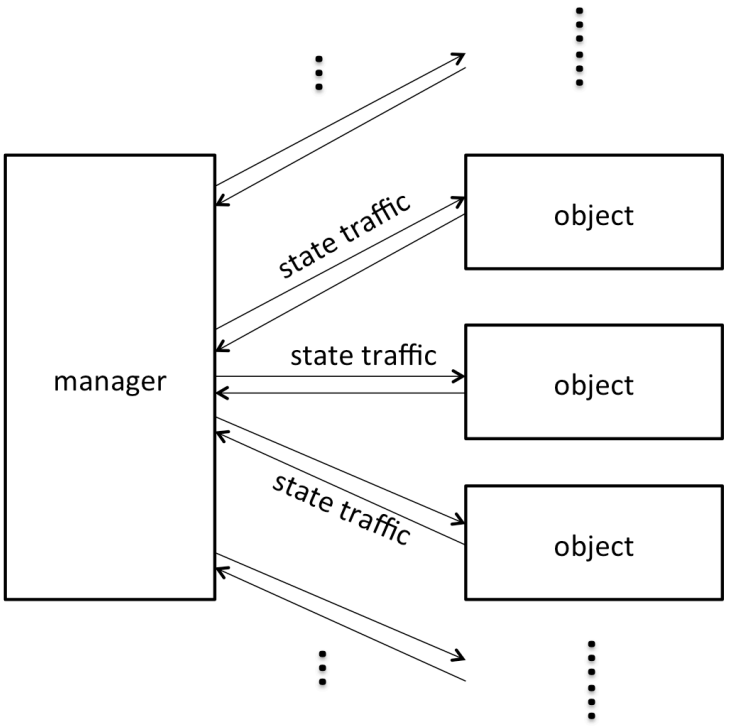

図 4.1: アプリケーションシステム概要図

システムの動作を以下に示す。

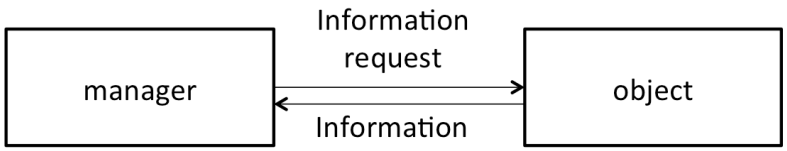

図 4.2: 模擬対象アプリケーションシステム

- マネージャ動作マネージャはオブジェクトに対して情報の要求を行う。この要求間隔時間は設定可能である。
- オブジェクト動作オブジェクトはマネージャから情報の要求を受信すると、要求された情報を HTTP で xml ファイルを送信する。また、NAT 越えのための STUN を使用し、その後はマネージャからの認証に応答し、それが完了後にファイルを送信する。具体的な動作については 4.4 のアプリケーションの通信動作に示す。このオブジェクトの通信動作をクライアントモジュールで模擬する。
- 通信動作アプリケーションは図 4.1 のような構成で使用されるが、作成する軽量クライアントモジュールは図 4.2 のようにマネージャとオブジェクトが一対一におけるオブジェクト側の通信を模擬する。この通信を模擬する軽量クライアントモジュールを作成することで、図 4.1 の大規模なシミュレーションにも対応可能になる。

4.2　アプリケーションで使用されるプロトコル・認証方式

模擬対象アプリケーションでは様々なプロトコルが使用されるが、以下に示すものが通信において主に使用される。作成する軽量クライアントモジュールはこれらの通信を模倣する。

4.2.1　TLS

模擬対象アプリケーションでは HTTP による通信を TLS（Transport Layer Security）によって暗号化し、通信内容の改竄や盗聴等を防いでいる。模擬対象アプリケーションでは SSL ソケットを作成し、HTTP による通信を行う。

4.2.2　STUN

STUN（Sinmple Traversal of UDP Through NATs）とは NAT 越えの際に用いられるプロコトルである。模擬アプリケーションでは NAT 越えを行う際に STUN プロトコルを使用する。UDP の STUN リクエストを送信するクライアントは STUN クライアントであり、それに対して応答を行うインターネット上のサーバが STUN サーバである。

模擬アプリケーションにおける STUN の動作を以下に示す。

1. クライアントが STUN リクエストを送信する。
2. STUN サーバがリクエストを受信する。
3. サーバはリクエスト内容を引用し、パブリック IP アドレスとポート番号、更に NAT の存在をクライアントに返す。
4. クライアントは応答を受信することで、NAT の種類と IP アドレスを学習する。

STUN では時間の経過によってバインディングがタイムアウトする。そのため適用アプリケーションにおける STUN クライアントはバインディングし続けるため、定期的に STUN リクエストの送信を行う。

4.2.3　基本認証

アプリケーションの用途によって認証が必要な通信を行う場合が存在する。選択した模擬対象のアプリケーションでは用途上、通信内容の改竄や盗聴等を防ぐ必要があるため、認証を行う。アプリケーションでは HTTP で定義される認証方式である基本認証を用いて通信を行う。基本認証は以下に示す順序で行われる。

1. クライアントがサーバに対して HTTP リクエストを行う。

2. サーバは、基本認証が必要であることをクライアントに通知する。

3. クライアントはユーザ名とパスワードが記述された認証ヘッダを再送信する。

4. 認証が成功した場合、サーバはクライアントのリクエストに対して応答する。
   認証が失敗した場合、クライアントは再度ユーザ名、パスワードを送信する。

この基本認証によって信頼性の高い通信を行っている。

4.3 アプリケーションの通信動作

模擬対象となるアプリケーションの通信動作について述べる。

アプリケーションは以下の図 4.3 に示す状態遷移図に沿った通信動作を行う。

アプリケーションは起動後に STUN をマネージャに対して送信するループに遷移する。この STUN の送信は、クライアントのバインディングに関する情報を保持するために一定間隔で再送信を行う。マネージャから接続要求を受信すると、クライアントとマネージャとのコネクションが開始される。その後マネージャはクライアントに対して認証要求を行う。それに対してユーザ名、パスワードを送信し認証を行う。認証は一定回数繰り返される。認証が成功するとシナリオ実行に遷移する。この状態では要求のあった xml ファイルをマネージャに向けて送信する。送信後はバインド待ちの状態に再び戻る。

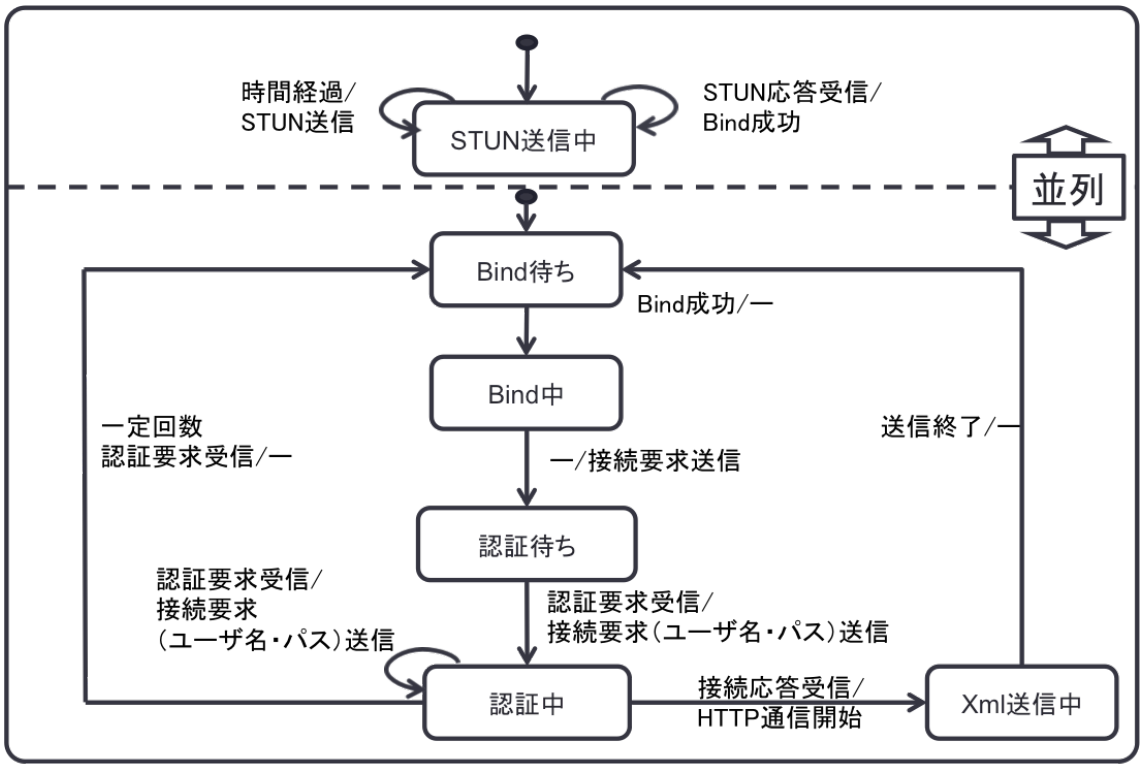

図 4.3: アプリケーションの状態遷移図

4.4　パケットキャプチャツール

適用アプリケーションの通信パケットを収集する際には、CUI パケットキャプチャツールである tcpdump を用いた。UnixOS 上で動作し、コマンドラインで使用可能である等の理由から　tcpdump を用いてパケットキャプチャを行った。

- tcpdump の概要

  tcpdump はネットワークトラフィックのパケットキャプチャを行うソフトウェアである。パケットキャプチャの際にコマンドオプションとしてフィルターを使用することが可能であり、特定のホスト、特定の宛先、特定のプロトコル等、収集を行うパケットの種類を限定することが可能である。フィルタオプションを併用することも可能であり、必要なパケットのみを収集できる。膨大な数の通信を行うアプリケーションを対象として　下記オプション w でパケットキャプチャを行う場合、フィルタリングを行わずにパケットの収集を行った際に出力ファイルのサイズが過大になるためフィルタリングの選定が重要となる。

- 使用したオプション

  - i

    観察を行うインターフェースを限定してパケットキャプチャを行う。

  - w

    キャプチャしたパケットを標準出力する代わりに、ファイル名を指定してパケットキャプチャの結果をファイルに出力する。このオプションで作成したファイルを wireshark 等の GUI ネットワークアナライザに読み込ませ、様々な分析を行うことが可能である。

  - r

    オプション w で作成したキャプチャデータが記述されているファイルを解析する際に使用する。

  - t

    パケットキャプチャの出力結果にタイムスタンプを表示しない。

- 使用した条件式

  - host

    host 以降に IP アドレスを指定することにより、指定の IP アドレスから送信されるパケットと、host が宛先となるパケットをキャプチャするようフィルタリングを行う。

- port
  ポート番号を指定し、そのポート番号で受信、送信するパケットのキャプチャを行う。
- src
  IP アドレスを指定し、その IP アドレスが送信元であるパケットをフィルタリングする。
- dst
  IP アドレスを指定し、その IP アドレスが宛先であるパケットをフィルタリングする。
- proto
  プロトコルを指定し、該当するプロトコルのパケットをフィルタリングする。

4.5 Java 仮想マシンのモニタリングツール

適用アプリケーションが通信を行っている間の Java 仮想マシン（JVM）のパフォーマンスを確認するため、Java 仮想マシンのパフォーマンスモニタリングツールである jstat を用いた。

- jstat の概要

  jstat とは、JVM のパフォーマンスをモニタリングするツールである。Jps コマンド等で確認した Java プロセスの VMID を指定し、特定のプロセスのヒープに関する情報を取得することが可能である。インターバルタイムを追記することで、統計情報の取得間隔を指定できる。jstat を使用することにより、JVM におけるヒープメモリの使用領域を詳細に知ることができる。

- オプション

  jstat を使用した際に用いたオプションについて以下に示す。

  - gc
    Java オブジェクトが使用するヒープメモリの使用量等を表示する。現在使用しているヒープメモリの容量から、現在使用している使用量を表示可能であり、ヒープメモリにおける、Suviver,Eden 領域から成る New 領域と Old 領域についての領域使用量も測定することができる。
  - gcutil
    Java オブジェクトが使用するヒープメモリの使用率等を表示する。gc では絶対量でヒープメモリの使用量を表示するが、gcutil では使用割合を表示する。

- jstat で取得可能なデータ

  jstat に gc オプションを付随させた場合に取得可能なヒープメモリの統計データの一例を以下の表 4.1 に示す。

表 4.1: jstat で取得可能なヒープメモリに関するデータ

データ名	データ内容
S0C	Suviver0 領域の容量
S1C	Suviver1 領域の容量
S0U	Suviver0 領域の使用量
S1U	Suviver1 領域の使用量
EC	Eden 領域の容量
EU	Eden 領域の使用量
OC	Old 領域の容量
OU	Old 領域の使用量
FGC	フルガベージコレクションの回数

4.6　アプリケーションの通信観測

軽量クライアントモジュールで模擬を行うアプリケーションの特徴量を抽出するために通信観測を行った。抽出する特徴量を以下に示す。

- STUN の送信間隔
- パケットサイズの分布

上記の tcpdump を用いたパケットキャプチャから特徴量を抽出した。
以下の図にアプリケーションの STUN の送信間隔、図にパケットサイズの分布を示す。

4.7　アプリケーションのプロファイリング

作成した軽量クライアントモジュールとのパフォーマンス比較を実施するためにアプリケーションのプロファイリングを行った。以下の項目についてプロファイリングを行った。プロファイリング結果は、上記の jstat や tcpdump によってキャプチャした情報を wireshark で解析したものである。

- CPU 使用率

  アプリケーションの CPU の使用率を示す。

- 通信中の CPU 使用率

  マネージャからの状態通知要求が 1 秒間に 1 回送信される際の CPU の使用率を示す。

- メモリ使用量

  アプリケーションのメモリの使用量を示す。

- 通信中のメモリ使用量

  マネージャから状態通知要求が 1 秒間に 1 回送信される際のアプリケーションのメモリ使用量を示す。

- レスポンスタイム

  状態遷移図におけるシナリオ実行状態では HTTP パケットを送信する。その HTTP のレスポンスタイムを示す。レスポンスタイムの最大値、最小値、平均値、標準偏差についても表に示す。

- STUN 送信間隔

  STUN の送信間隔を示す。

模擬対象アプリケーションのプロファイリング結果を以下に示す。

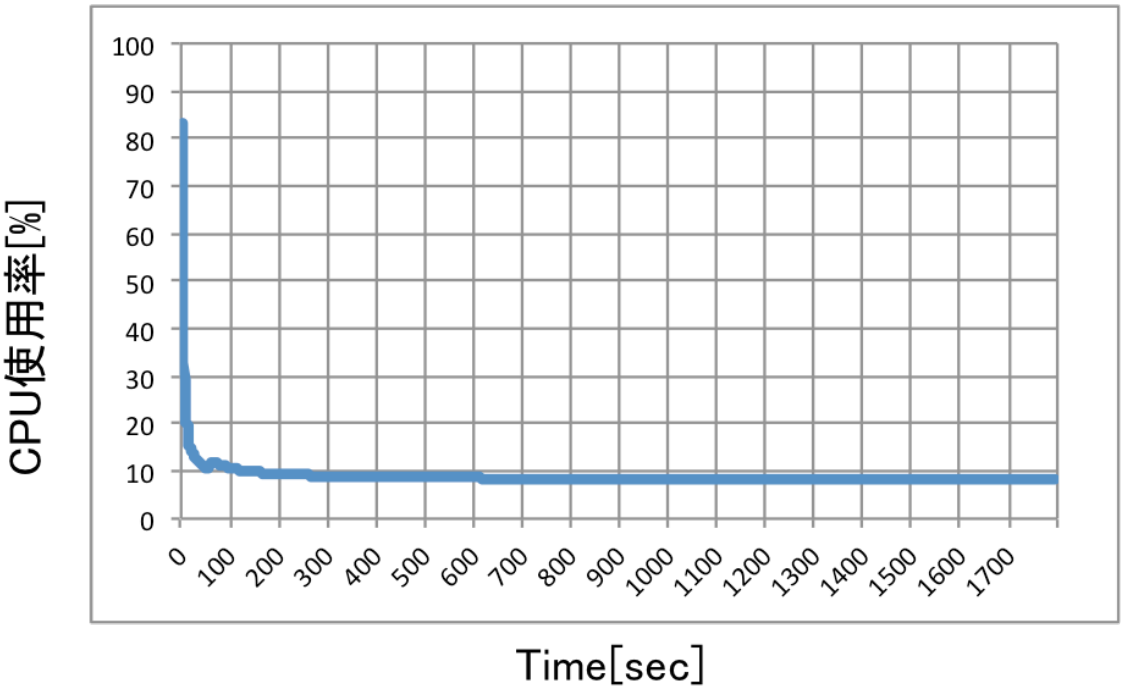

図 4.4: アプリケーションの CPU 使用率

表 4.2: アプリケーションの HTTP レスポンスタイム

項目	レスポンスタイム [μ sec]
最大値	189344
最小値	129244
平均値	152349
標準偏差	6275

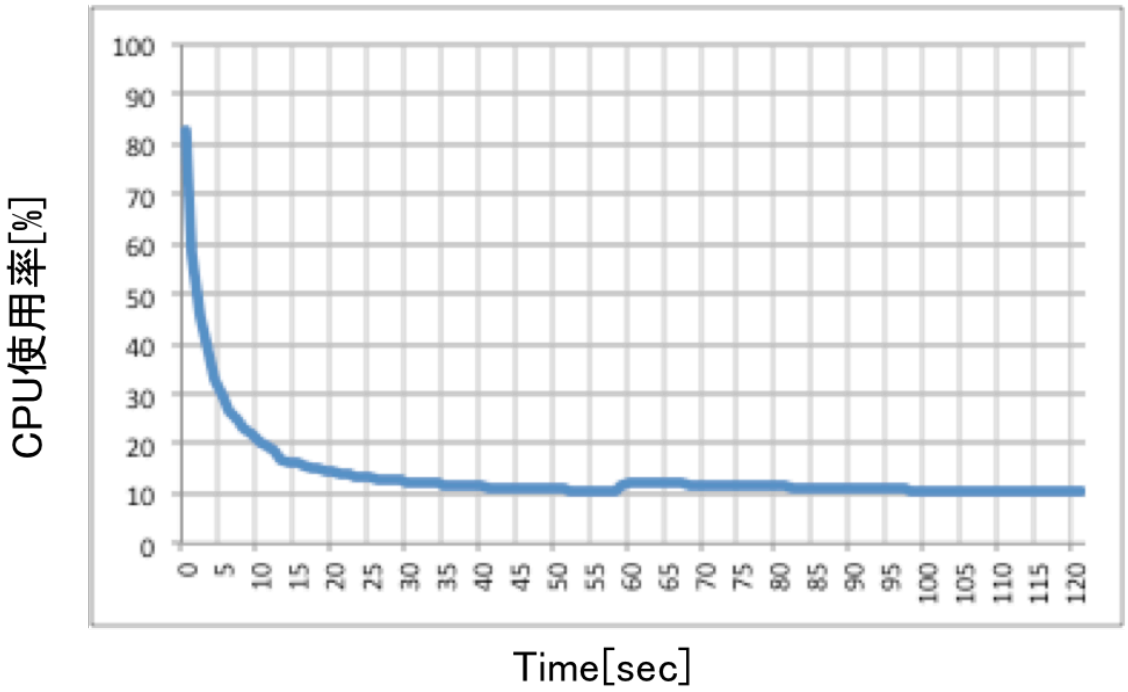

図 4.5: 通信中のアプリケーションの CPU 使用率

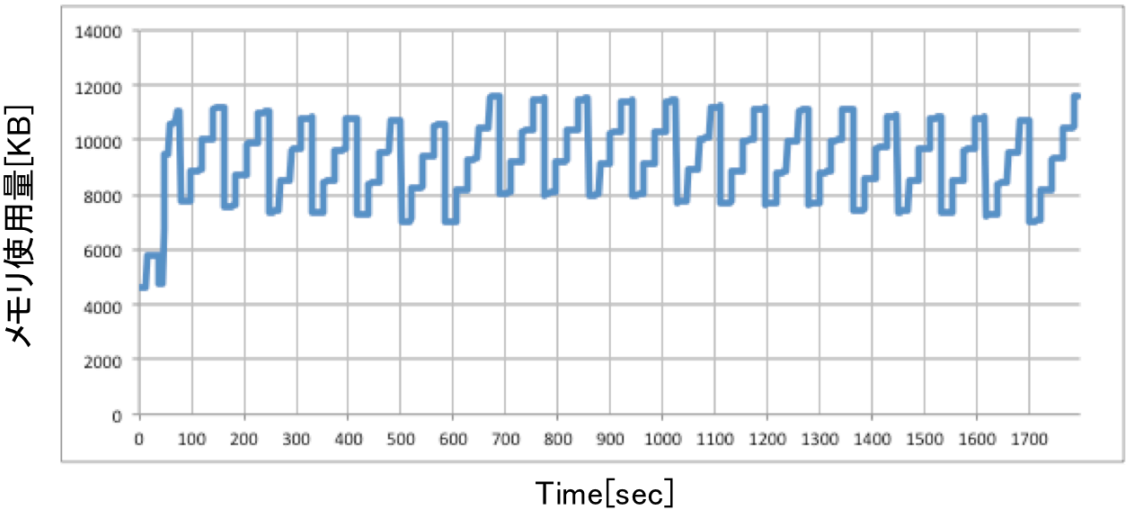

図 4.6: アプリケーションのメモリ使用量

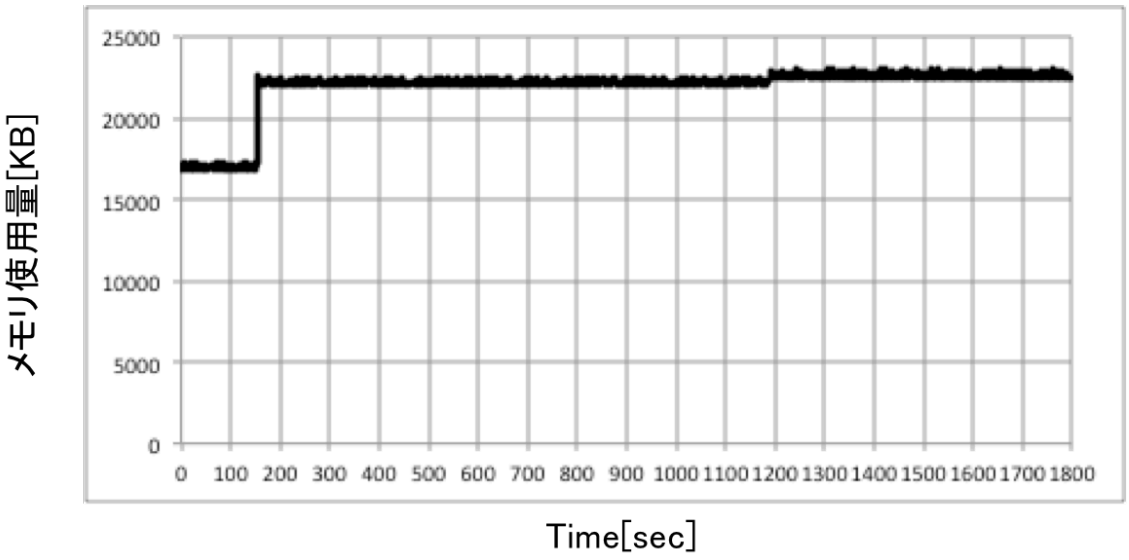

図 4.7: 通信中のアプリケーションのメモリ使用量

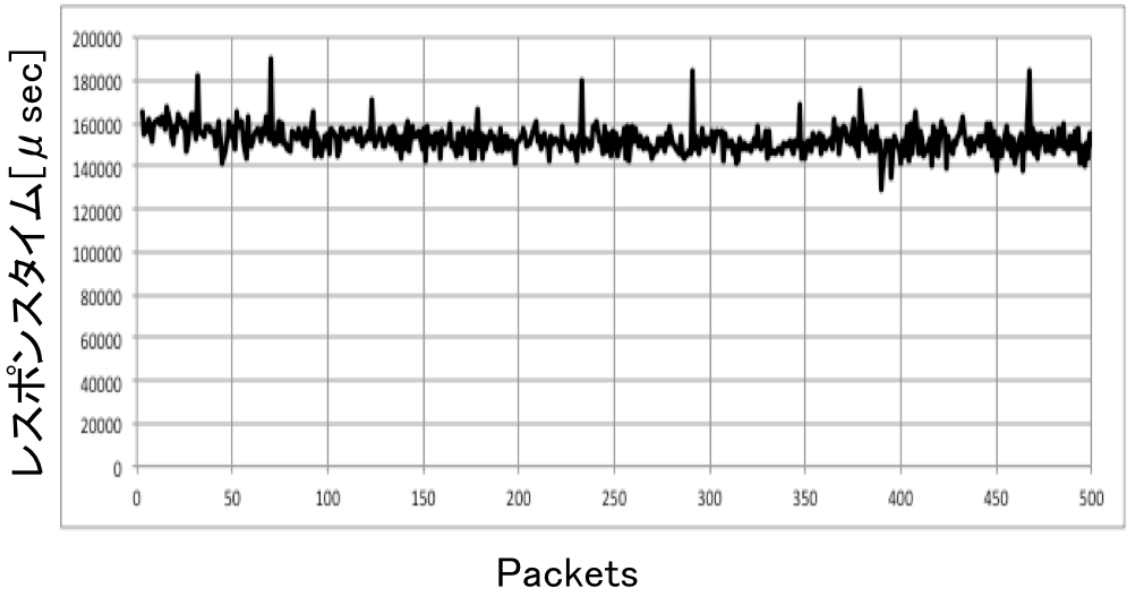

図 4.8: アプリケーションにおける HTTP のレスポンスタイム

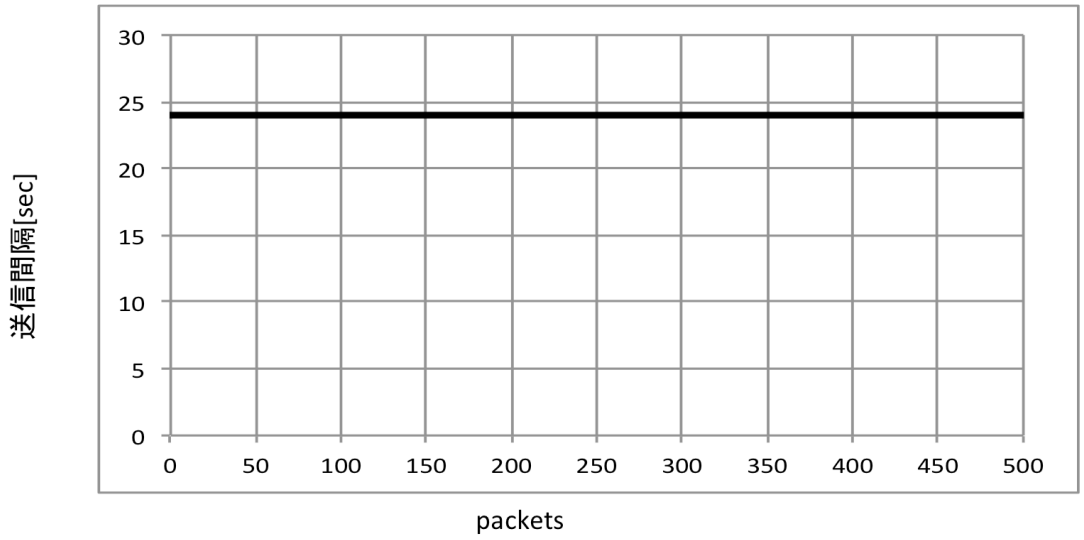

図 4.9: アプリケーションにおける STUN 送信間隔

第5章　軽量クライアントモジュールの実装

軽量クライアントモジュールの実装を行った。軽量クライアントモジュールの実装の一連の流れを以下に示す。

- 状態機械の作成
- 特徴量抽出
- モジュールの作成

このようなプロセスで軽量クライアントモジュールを作成し、その後にプロファイリングを行い、模擬対象アプリケーションとの比較を行う。

5.1　状態機械の作成

本研究では、マネージャとオブジェクトからなるシステムが通信を行うアプリケーションを模擬対象と決定した。その状態機械をアプリケーション制作者からの仕様書情報等から作成した。作成した状態遷移図は 4 章の 4.3 に示した。

5.2　特徴量抽出・適用

4 章で抽出を行ったアプリケーションの特徴量を軽量クライアントモジュールに適用した。

5.3　モジュール作成

4 章で作成した状態機械と特徴量を用いて、モジュールの作成を行った。

5.4　軽量クライアントモジュールのプロファイリング結果

作成した軽量クライアントモジュールのプロファイリングを行った。プロファイリング対象について以下に示す。

- CPU 使用率

  CPU の使用率を示す。

- 通信中の CPU 使用率

  マネージャからの状態通知要求が 1 秒間に 1 回送信される際の CPU の使用率を示す。

- メモリ使用量

  メモリの使用量を示す。

- 通信中のメモリ使用量

  マネージャから状態通知要求が 1 秒間に 1 回送信される際のメモリ使用量を示す。

- レスポンスタイム

  状態遷移図におけるシナリオ実行状態では HTTP パケットを送信する。その HTTP のレスポンスタイムを示す。レスポンスタイムの最大値、最小値、平均値、標準偏差についても表に示す。

- STUN 送信間隔

  STUN の送信間隔を示す。

軽量クライアントモジュールのプロファイリング結果を以下に示す。

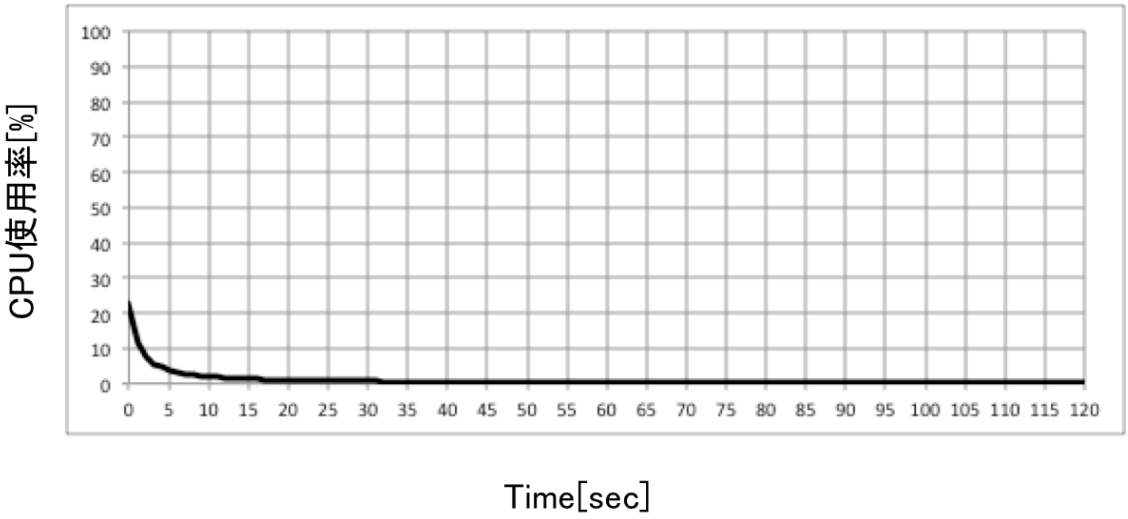

図 5.1: 軽量クライアントモジュールの CPU 使用率

表 5.1: 軽量クライアントモジュールの HTTP レスポンスタイム

項目	レスポンスタイム [μ sec]
最大値	389352
最小値	61803
平均値	83810
標準偏差	17670

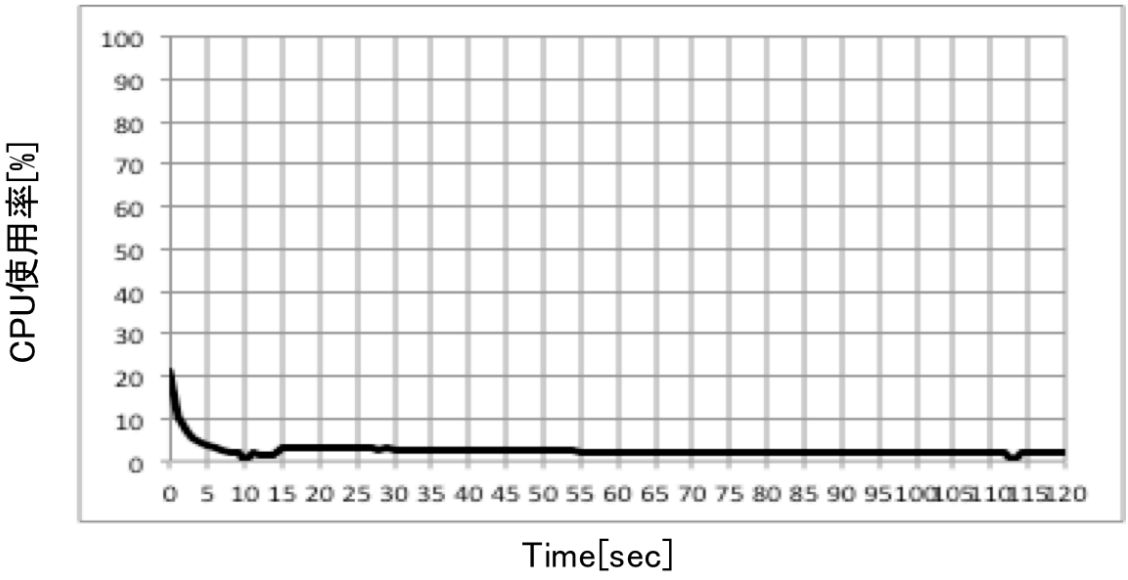

図 5.2: 通信中の軽量クライアントモジュールの CPU 使用率

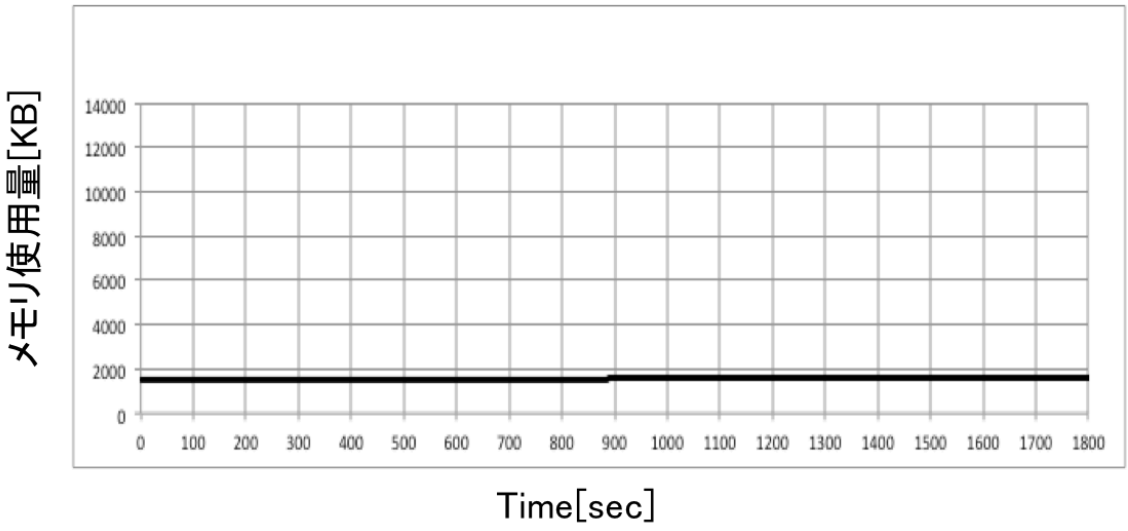

図 5.3: 軽量クライアントモジュールのメモリ使用量

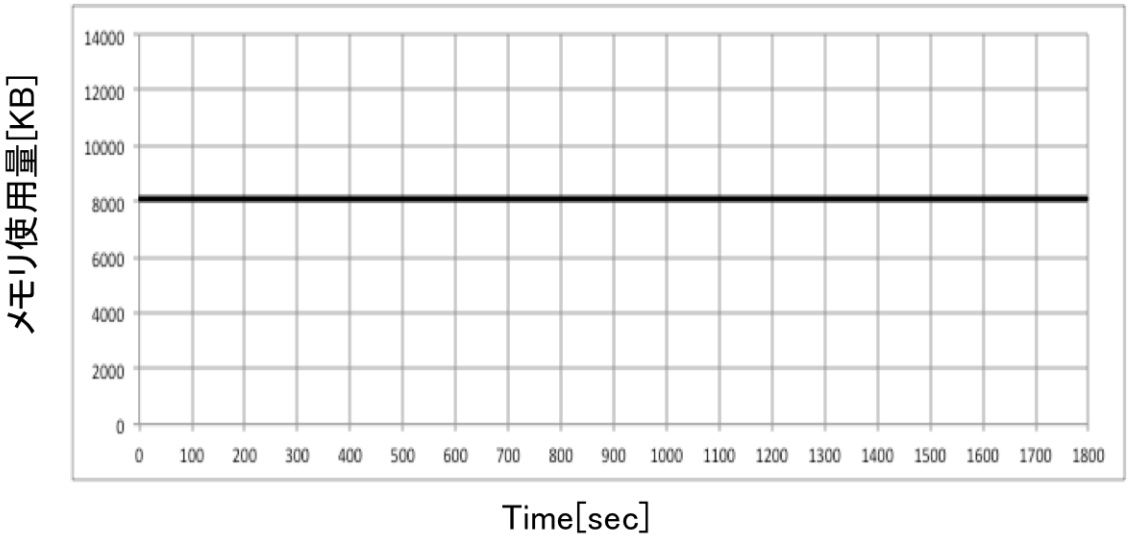

図 5.4: 通信中の軽量クライアントモジュールのメモリ使用量

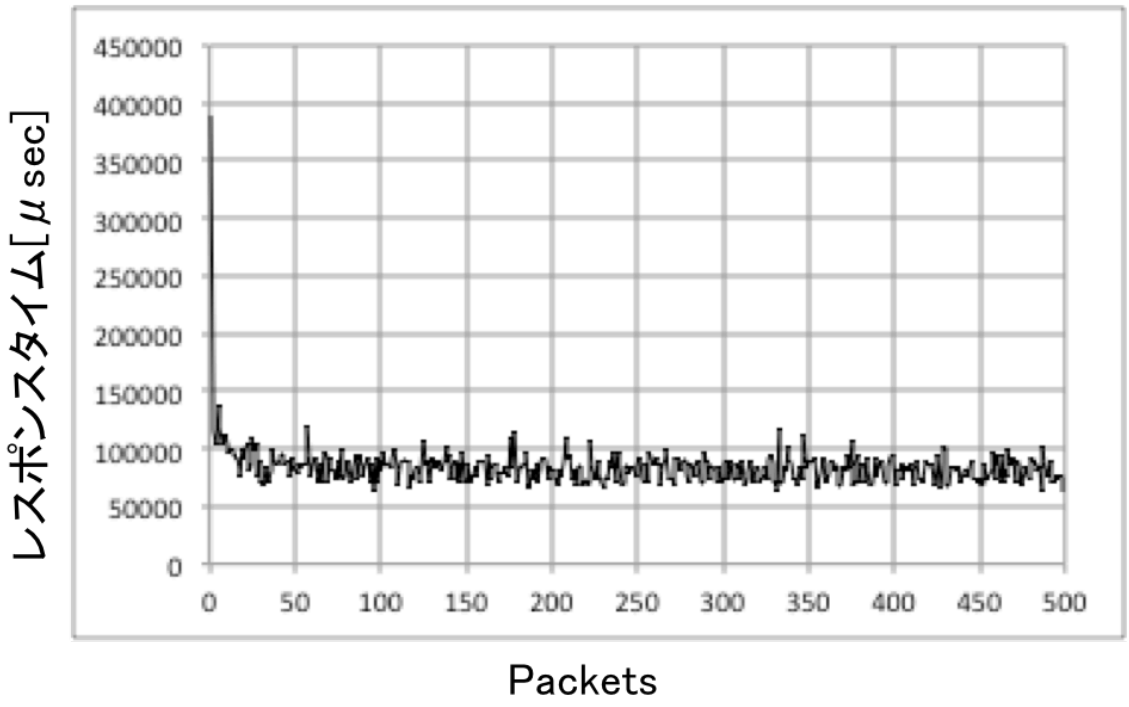

図 5.5: 軽量クライアントモジュールのレスポンスタイム

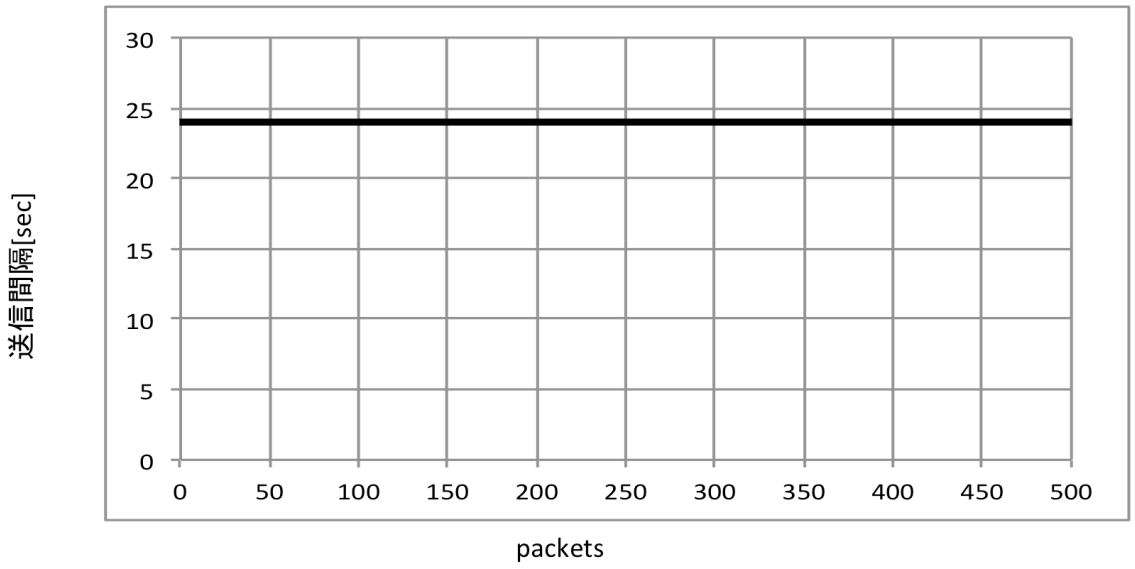

図 5.6: 軽量クライアントモジュールにおける STUN 送信間隔

5.5 軽量クライアントモジュールの並列起動実験

5.5.1 プロセスによる並列起動

作成した軽量クライアントモジュールを並列プロセスにより複数起動させた場合、一つの物理マシンで何台分のモジュールを起動可能か検証を行った。使用したマシンのスペックは本論文 3 章 3.2 のノード I のマシンである。実験方法は、マシン上で複数の軽量クライアントモジュールを並列プロセスで立ち上げ、次に正確にサーバと通信を行っているか確認を行った。その実験を 5 回行った結果、1 台のマシン上で平均的に 785 個の軽量クライアントモジュールの起動を行えることが確認できた。また、tcpdump での通信観測の結果より、サーバとの通信を正確に行っていることが確認できた。5.4 のメモリ使用量測定実験の結果から、これ以上のモジュールの起動が可能であると予想したが、OS 単位でのメモリの使用量がヒープメモリの使用量より比較的に大きく、この結果に至ったものと考えられる。

5.5.2 スレッドによる並列起動

5.5.1 の実験は、並列プロセスによって軽量クライアントモジュールを一つのマシン上で多数動作させるという実験であった。並列にアプリケーションを動作させる手段にはプロセスでの並列処理の他にプロセス内で複数のスレッド生成し、並列処理を行う方法がある。このスレッドで軽量クライアントモジュールを多数起動させ、5 万台規模のシミュレーションを行うためには何台のマシンが必要か検証を行った。使用したマシンのスペックは本論文 3 章 3.2 のノード I のマシンである。実験方法は、物理マシンを起動し、そこで軽量クライアントモジュールのスレッドを 1 万個生成した。その結果マシン 1 台で 1 万個の軽量クライアントモジュールの動作が可能なことが、確認できた。そのため、StarBED におけるノード i のマシン 5 台があれば 5 万台規模のシミュレーションが可能であるといえる。また、スレッド上の軽量クライアントモジュールが正確な情報を正確な相手に送信しているか、確認するため tcpdump コマンドによって STUN パケットの観測を行った。その結果、適切な相手に STUN を送信していることが、tcpdump の出力結果により確認できた。図 5.7 に軽量クライアントモジュールを 1 万台動作させた際の CPU の使用率のグラフを示す。この図 5.7 は、ps コマンドを用いて得た結果であり、論理プロセッサ 1 台あたりの CPU 時間を 100 %としている。

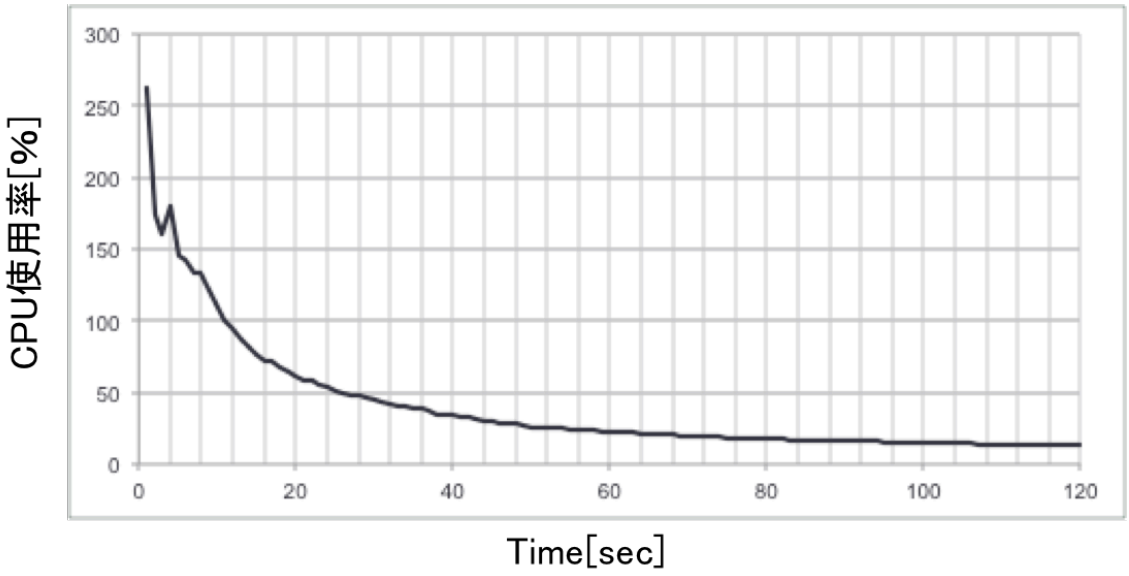

図 5.7: 軽量クライアントモジュールを 10000 台動作させた際の CPU 使用率

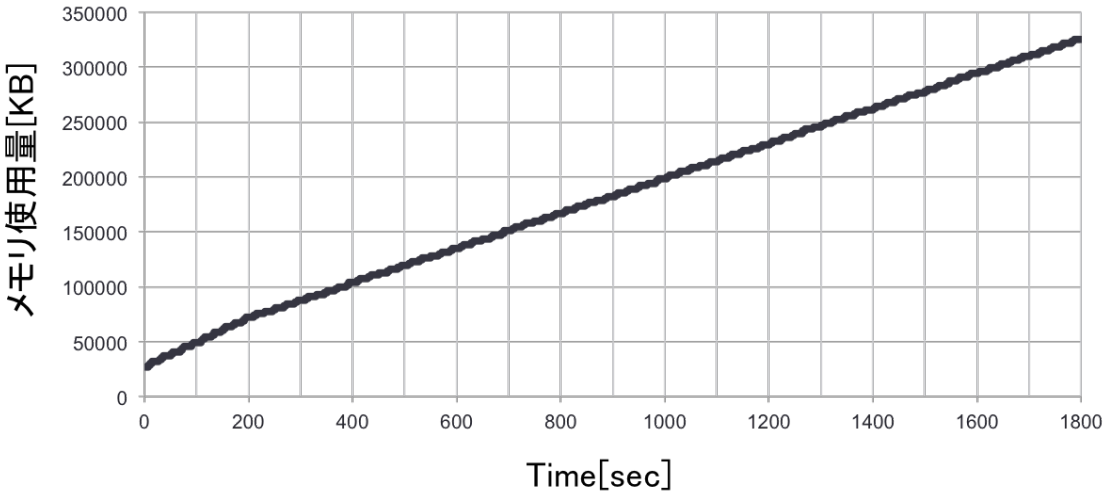

図 5.8: 軽量クライアントモジュールを 10000 台動作させた際のメモリ使用量

第6章　評価・考察

本章では作成した軽量クライアントモジュールについての評価・考察を述べる。プロファイリングの結果より、メモリの使用量に着目すると、作成した軽量クライアントモジュールによって模擬対象アプリケーションが消費するメモリ量を低減できた。

軽量クライアントモジュールによって、メモリの使用量を削減できた。本研究では Java を用いて実装を行ったため、ガベージコレクションが自動で行われる。そのため、メモリの開放を手動で行う言語での実装をすることで、より軽量なモジュールを作成できる可能性がある。

プロセスによって軽量クライアントモジュールを一つのマシン上で複数立ち上げる際には、平均 785 個の軽量クライアントモジュールの起動・通信が可能であることが確認できた。5.4 の実験結果から予測していた数よりも少なかった理由として、OS 単位でのメモリの使用量が大きかったことが考えられる。スレッドを用いることで一つのマシン上で 1 万個の軽量クライアントモジュールの起動が可能であることが確認できた。従って、5 台のマシンがあれば 5 万台規模の大規模シミュレーションが可能である。

第7章　まとめ

大規模シミュレーションの必要な計算資源の低減実現のために、軽量クライアントモジュールの設計、実装、検証を行った。本研究では、必要とする計算資源が少なく、実際のサーバとの通信が行える軽量クライアントモジュールの実現を図った。設計に沿ったモジュールの作成ではまず模擬対象アプリケーションの状態機械を作成し、さらに模擬対象の通信観測から得られる特徴量を利用して、クライアントモジュールの作成を行った。

設計に基づいて作成した軽量クライアントモジュールによって、メモリの使用量を削減可能であることが確認できた。今回の軽量クライアントモジュールの開発ではガベージコレクションを自動で行う言語を用いて行った。ガベージコレクションを手動で適切に行うことによってより軽量なクライアントモジュールの作成が行える可能性があり、今後の課題である。また、実際に設計に基づいて軽量クライアントモジュールの作成を行ったが、性能の検証実験に要する時間が足りず、十分な性能評価を行うことができなかった。しかし、設計に基づいたクライアントモジュールによってサーバとの通信が可能で、必要な計算資源を低減できることが判明した。そのため今後は、本手法でモジュールを作成した際の低計算資源化のボトルネックとなる要因を洗い出し、より大規模なシミュレーションに用いることが可能な軽量クライアントモジュールの作成を行いたい。

また、プロセス・スレッドで軽量クライアントモジュールを複数動作させた際に、1台のマシンで動作可能なモジュールの数を検証できた。プロセスで動作させた場合には予想よりも少ない個数のモジュールしか起動できなかった。そのため、今後はその原因の調査と、プロセスとして動作させた際にも多数の並列動作が行う方法を検討する必要がある。1台のマシン上での並列動作時の軽量クライアントモジュールの個数を増加させることも今後の課題である。

謝辞

本稿を執筆するに当たり、研究に関するご指導を賜りました丹康雄教授に心から感謝するとともに、ここに深くお礼申し上げます。多くの助言を頂きました、リム勇仁准教授に深く感謝いたします。また、研究や公私の面でサポートを頂きました丹研究室、リム研究室の皆様に感謝の言葉を申し上げます。

参考文献

[1] 曽川貴裕, スマートコミュニティシミュレータに向けた階層モジュール化技術の研究, , 北陸先端科学技術大学院大学博士論文 2014,3.

[2] 岡田崇, ホームネットワークサービス及びそのシステムの実証的検証に関する研究, 北陸先端科学技術大学院大学博士論文 2011,9.