C-07

MARSの冗長構成及び実現手法の提案

Proposal of MARS redundant configuration and implementation method

新谷 裕太†　　川橋 裕‡
Yuta Shintani　　Yutaka Kawahashi

1. はじめに

近年情報化社会が発展するにつれ，多くの大学機関や研究機関，一般企業などで独自のネットワークであるLAN(Local Area Network)の提供がおこなわれている．それに伴い，ユーザは機関内では事前申請をおこなうことで個々の利用環境に合わせてインターネットを享受できるようになっている．加えて，情報管理のIT化も進行しており，それぞれのLANにおける運用管理の重要性は年々高まっている．

しかしながら，インターネット利用者が増加していくにしたがって，LAN内では多くの障害が発生している．そのため，運用管理ではセキュリティの確保が問題となっている．障害とは，LAN内のサーバへの攻撃やネットワークに接続できないといったユーザからの問い合わせなど様々存在する．

これらに対応するため，ネットワーク管理者（以下，管理者）は，一般的にIPアドレスやサーバのログ情報から障害対応をおこなう．そのため，多くの機関ではIPアドレスの利用を申請制にすることで，そのIPアドレスを使用するユーザを把握している．和歌山大学(以下，本学)でも，紙媒体での利用申請書を用いて，使用するユーザとIPアドレスを対応付けている．しかし，利用申請をせずに使用するユーザや変更申請をせずに過去の申請内容のまま使用するユーザが少なからず存在する．実際に申請された内容とは異なることがあるため，管理者は正確な利用情報を得ることが困難であり，障害対応に多大な労力と時間を要する．

そこで，本学では先行研究としてMARS(Monitoring Analysis and Response System)[1]を運用している．MARSでは，ユーザが使用している端末のIPアドレスやMACアドレス，その端末が設置されている場所を統合的に管理している．端末の情報は，大学内の各所に設置されているエッジスイッチにおける接続情報を，RADIUS認証プロトコルを用いてデータベースに記録することで取得している．その後，ブラウザを通じて管理者に提示するシステムとなっている．これにより，実際に使用されている端末のIPアドレスとその端末，場所が対応付けられ，障害に対して迅速に対応することが可能となった．

しかしながら，現在のMARSでは，ユーザの情報を統合的に管理しているデータベースが単一であるため，そのデータベースに障害が発生した場合，ブラウザから認証情報を閲覧することが困難になる．さらに，フロントエンドであるWebシステムとバックエンドであるデータベースが同一のサーバにあるため，このいずれかのシステムに障害が発生した際も相互に影響を受けてしまい，同様に閲覧することが困難になる．さらに，エッジスイッチはRADIUS認証が成功しない場合，端末は物理的にエッジスイッチに接続されていてもインターネットに接続できない問題がある．したがって，障害が発生した際は，管理者がMARSを利用して障害対応ができないだけでなく，エッジスイッチに接続している端末にも弊害が発生してしまう．

† 和歌山大学大学院システム工学研究科
Graduate School of System Engineering Wakayama University
‡ 和歌山大学システム情報学センター
Wakayama University System Information Informatics Center

そこで，本研究では，MARSをフェイルセーフで冗長的な構成とすることを目的とする．データベースとWebサーバを分割することで相互に影響を受けなくする．加えて，データベースとRADIUSサーバを複数設置することで多重化し，各RADIUSサーバからの認証情報をそれらのデータベースに同一の情報を蓄積することで実現した．さらに，冗長構成にすることでMARSに関連するサーバの数が増えてしまい，管理がしにくくなることを考慮し，関連するサーバの監視をおこなう専用のインタフェースの開発もおこなう．

本論文では，第2章で既存技術とその問題点を述べる．第3章で先行研究について述べ，第4章で研究目的について述べる．第5章及び第6章では提案システムとその実装・実験について述べ，第7章でその評価・考察・今後の課題について述べる．

2. 既存技術

2.1 統合監視システム

統合監視システムとは，ネットワークを介して特定の複数のホストを集中監視するシステムである．このシステムは，監視対象であるホストの死活やそのホスト上で実行されているプロセス，ネットワークなどの様々なものを監視することができる．本節では，代表的な統合監視システムである「Zabbix」，「Nagios」，「Hinemos」について述べる．

2.1.1 Zabbix

Zabbix[2]は，監視対象の端末情報を監視サーバのデータベースに保存し，Webインタフェースを利用して管理者に提示する．異常時にはメールや指定のスクリプトを実行させることができる．

2.1.2 Nagios

Nagios[3]は，指定の時間で指定のネットワークサービスの監視をし，リモート監視の場合には，Web インタフェースから主にSSH を使用する．異常時には指定のイベントを実行させることができる．

2.1.3 Hinemos

Hinemos[4]は，上記二つと同様にWeb インタフェースからプロセス管理をはじめ，SNMP(Simple Network Management Protocol）監視，SQL 監視，システムログ監視ができる．SNMPは，ルータ，スイッチ，サーバなどのネットワークの通信機器に対して，ネットワーク経由で監視することができるプロトコルである．加えて，比較的新しいソフトウェアであり，複数の監視対象に対して，バッチの実行などの操作をまとめて実行することができる．

2.2 統合監視システムの問題点

既存の統合監視システムは，それぞれ様々な監視が可能であるが，監視対象の端末を把握することが前提とな

っている．加えて，端末の細かい情報を得るためには，SNMP を利用するため端末に専用のソフトウェアをインストールしなければならない．そのため，ユーザが新たに端末を使用する際には管理者へ利用申請をおこなう必要がある．しかし，利用申請をせず新たに端末を使用するユーザが少なからず存在する．したがって，ネットワークに接続されている端末情報は全て申請通りとは言えず，その実態を正確に把握するのは困難である．

3 章ではこの問題点を解決した先行研究の MARS について詳しく述べる．

3. 既存技術

本章では，端末のネットワーク接続をリアルタイムに監視することにより，管理者の障害対応を支援する先行研究のMARS について述べる．

3.1 MARS(Monitoring Analysis and Response System)

本節では，MARS の動作と問題点について述べる．

3.1.1 動作と端末接続情報

MARSは，RADIUS 認証プロトコル[5] を利用し，エッジスイッチが通知する端末の接続情報を収集するシステムである．ここでエッジスイッチとは，ある機関内でのネットワークの末端にある，ユーザに最も近いスイッチのことである．また，RADIUS 認証プロトコルはRFC2865 で策定されており，ダイヤルアップ・インターネット接続サービスを実現するためのプロトコルである．近年では，ダイアルアップだけでなく，ブロードバンド接続やVPN，VLAN，無線LAN へ接続する際のユーザ認証にも利用されている．ユーザ認証をおこなう際は，エッジスイッチに対して認証の設定をおこない，RADIUS サーバと連携させる必要がある．MARS はRADIUS サーバ群から成る収集部，認証情報を保存するデータベースサーバから成るデータベース部，収集した情報を閲覧するためのインタフェース部の3 つで構成されている．下記と図3.1 にエッジスイッチからの認証情報の収集手順を示す．

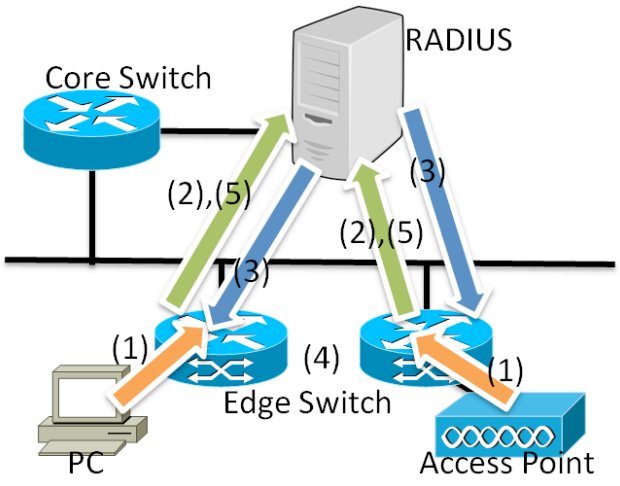

図 3.1:認証情報の収集手順

1. ユーザの端末がエッジスイッチに接続
2. エッジスイッチは接続要求をRADIUS サーバに送信
3. RADIUS サーバは受信した情報を基に認証し，結果をエッジスイッチに通知
4. エッジスイッチは通知された認証結果を基に，端末の接続を許可
5. エッジスイッチは端末の接続が終了した事実をRADIUS サーバに通知

先述したように既存技術では，監視する際はユーザ側に専用の設定をおこなう必要がある．しかし，MARS の上記の認証では，RADIUS サーバ側で端末の接続に対して全ての接続を許可するという設定にしておくことでユーザに認証を要求せず監視が可能な設計としている．これにより，MARS を導入することでユーザに新たな負担がかからないようにし，導入の敷居を下げている．

次に，エッジスイッチがRADIUS サーバに通知する情報を下記と図3.2 に示す．

```
Thu Dec  1 [redacted] 2016
        Packet-Type = Access-Request
        Framed-IP-Address = 172.16.[redacted]
        Calling-Station-Id = "742b.627d.24a9"
        Service-Type = Call-Check
        NAS-Port-Type = Ethernet
        Message-Authenticator = [redacted]
        NAS-Port-Type = Ethernet
        NAS-Port = 50102
        NAS-Port-Id = "GigabitEthernet1/0/2"
        NAS-IP-Address = 172.16.[redacted]
        Acct-Session-Id = "0000008B"
```

図 3.2:RADIUS が受信する接続情報のログの例

- セッションID
- 端末のIP アドレス
- エッジスイッチのIP アドレス
- エッジスイッチのポート番号
- 接続開始もしくは終了状態
- 接続開始もしくは終了時刻

MARS はエッジスイッチから収集した情報と，エッジスイッチと設置場所が対応付けられたパッチ情報を連携させ，端末が接続された部屋名を特定する．端末の接続を開始してから終了するまでの通信をひとつのセッションと定義し，収集される端末の接続情報を各セッションごとに管理している．端末の接続情報として管理される項目を下記に示す．データベース部はこれらの情報をひとつのカラムとして格納している．

- セッションID
- 端末のDNSホスト名
- 端末のIPアドレス
- 端末のMACアドレス
- エッジスイッチのIPアドレス
- エッジスイッチのポート番号
- 棟名
- 部屋名
- 接続開始時刻
- 接続終了時刻

次に，MARS の収集部，データベース部，インタフェース部の全体の動作について下記と図3.3 に示す．以降，簡略化のため，データベース部とインタフェース部のあるサーバを「MARS サーバ」と呼称することとする．

1. エッジスイッチがRADIUS サーバに対して，認証情報を通知
2. RADIUS サーバは認証情報をMARS サーバに対して送信

3. MARS サーバは認証情報を上記カラムに成型し，データベース部に格納
4. 管理者はMARS サーバにあるインタフェース部を利用して，認証情報を閲覧

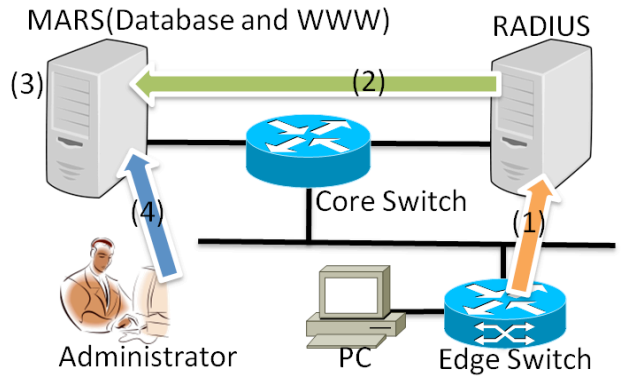

図 3.3:MARS の動作手順

図3.4 のようにこれらの情報を管理インタフェースから閲覧することができる．指定のエッジスイッチのIP アドレスやMAC アドレス，特定の部屋などでの検索も可能である．

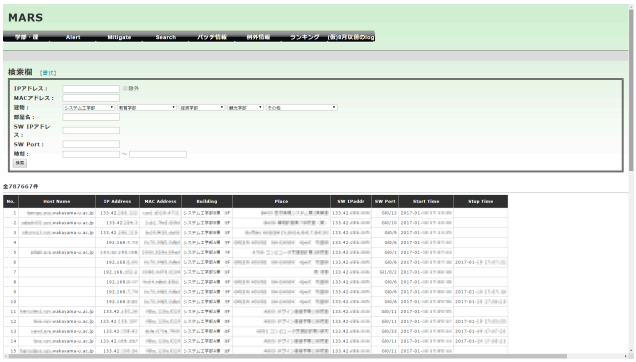

図 3.4:管理インタフェース

3.1.2 先行研究の問題点

これらの手法を使用することで，MARS では正確な認証情報を管理者のもとに収集し閲覧することが可能となり，管理者の障害対応に役立っている．その一方で，現在のMARSはMARSサーバが1 台しかなく，障害が発生した場合，障害対応に使用できない．加えて，先述の通り，MARS サーバ上にデータベース部とインタフェース部がともにあるため，そのいずれかに障害が発生した場合，相互に影響を受けてしまい，認証情報を閲覧することも，収集することもできなくなる．さらに，収集部であるRADIUS サーバに障害が発生した際は，同様に認証情報を収集できなくなる．それだけではなく，エッジスイッチはRADIUS 認証が成功しない場合，端末は物理的にエッジスイッチに接続されていてもインターネットに接続できない問題が発生する．したがって，RADIUS サーバは常に応答できる状態でなければネットワーク内のユーザの利便性が損なわれてしまう．

以上の問題点より，現在のMARS は一部に障害が発生した際にも，運用を継続できる冗長構成にする必要があると考える．

4. 研究目的

管理者は自身が管理するネットワークで障害が発生した際，解決のために迅速に原因の究明をおこなわなければならない．そのため，本学では先行研究であるMARS を運用している．MARS を運用することで，管理者はIP アドレスからそのIP アドレスを使用しているMAC アドレスや設置場所を特定することができ，障害の対応に役立てることができる．しかし，MARS の収集部，データベース部，インタフェース部のいずれかに障害が起こる可能性も存在する．その際，管理者は端末の接続情報の閲覧や，障害対応に使用することができなくなる．さらに，3章で述べたように，エッジスイッチがRADIUS 認証に成功しない場合，端末がそのエッジスイッチに物理的に接続されていてもインターネットに接続できない．そのため，MARS の収集部であるRADIUS サーバに障害が発生した場合，接続情報が取得できないだけでなく，ユーザの端末もインターネットに接続できなくなる．加えて，現在のMARS のデータベース部は単一であり，データベースに障害が発生した場合にも，管理者は接続情報を閲覧できなくなるだけでなく，管理している接続情報が消えてしまう可能性も存在する．

そこで，本研究では，現在のMARS を改良し，MARS 内に障害が発生した際にも管理者が接続情報を閲覧することが可能である冗長的な構成を提案及び構築することを目的とする．冗長的な構成とは，MARS の一部に障害が発生した際，運用を停止させることなく継続して接続情報を取得および閲覧できる構成のことを指す．これにより，いかなる状態であっても，管理者はMARS を利用して障害対応にあたることができる．しかし，冗長構成にすることでMARS に関連するサーバの数が増えてしまい，その数だけpingやSSH 接続を利用して死活監視，プロセス監視をしなければならなくなる．本研究で提案するインタフェースを利用することで，管理者はMARS のすべてのサーバの状態をまとめて確認できるようになり，動作確認のたびにそれらのサーバにアクセスする必要がなくなると考える．

5. 提案システム

本章では，MARS の冗長化のための構成と，従来の管理インタフェースの修正，サーバをまとめて監視できるインタフェースの作成について，それぞれの提案手法およびその実装について述べる．

5.1 MARSの冗長化手法

本節では，従来のMARSの問題点を解決するための冗長化手法について述べる．

5.1.1 RADIUSサーバの複製

RADIUS サーバに障害が発生した場合，認証情報の収集がおこなえないだけでなく，エッジスイッチが認証をおこなえなくなるため，ネットワークの末端のユーザの利便性が損なわれてしまう．そこで，RADIUS サーバを複製し，すべてのエッジスイッチが2 つのRADIUS サーバを指定するようにした．エッジスイッチは，認証をおこなうRADIUS サーバを複数指定できる．そのため，プライマリとして指定されたRADIUS サーバに認証が通らなければ，セカンダリとして指定されたRADIUS サーバに対して，認証をおこなうことができる．そのため，複製したすべてのRADIUS サーバに障害が起きていない限り，エッジスイッチは常に認証することができる．

5.1.2 MARSサーバの分割

3.1.2 節で述べたように，MARS サーバ上にデータベー

ス部とインタフェース部が共存しているため，障害が発生した際に相互に影響を受けてしまう．そこで，MARS サーバにおけるデータベース部とインタフェース部を別々のサーバに分割することで，影響を受けなくする．しかし，サーバの分割をおこなうことにより，ファイアウォールの問題が発生する．従来のMARS では，MARS サーバにあるインタフェースから自身のサーバ内にあるデータベースにアクセスしていたためファイアウォールに阻止されることはなかった．しかし，分割することでインタフェースのある外部のサーバからデータベースにアクセスすることになるため，ファイアウォールに阻止されてしまう．そのため，データベース部のあるサーバではデータベースの通信で標準的に使用する3306 番のポートを開放する．

従来のMARS サーバはデータベース部の一台として今後運用に使用するため，新たにインタフェース部のサーバを作成した．

5.1.3 データベース部の複製

MARSサーバの分割ができていても，データベース部は単一のため，データベース部に障害が発生した場合，認証情報の収集や閲覧ができなくなるのは変わらない．そのため，データベース部を複製することで常に収集や閲覧が可能な構成とする．

5.1.4 提案システムの構成

上記の手法を従来のMARSの構成に適用した際の構成と動作手順を下記と図5.1に示す．

1. エッジスイッチは設定されたRADIUS サーバに対して，認証情報を通知
2. RADIUS サーバは認証情報をデータベース部の全てのサーバに対して送信
3. データベース部は認証情報をそれぞれのデータベースに格納
4. 管理者はインタフェース部で指定の情報を要求
5. インタフェース部は要求に応じ，データベース部から情報を取得

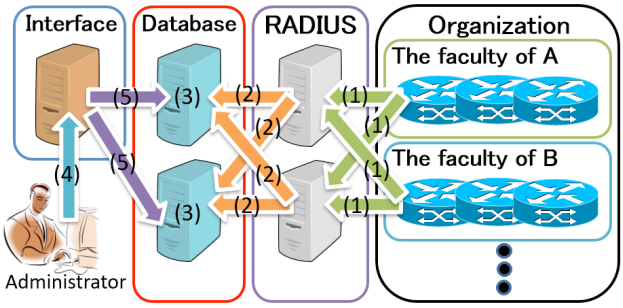

図 5.1:提案システムの構成

エッジスイッチは複数のRADIUS サーバを指定して認証が可能なサーバのひとつに対して認証情報を通知する．すなわち一度の認証に対して，複数のRADIUSサーバを利用しない．手順(3) の際の格納方法は従来の手法と同様で，3章で説明したカラムに成型したのち保存しておく．手順(4)(5) では管理者はどのデータベースで情報の検索をおこなうかを指定できる．

5.2 インタフェースの修正および追加

本節では，従来の管理インタフェースの修正と新たに作成する監視インタフェースについて述べる．

5.2.1 新たな管理インタフェース

インタフェース部に表示する新たな管理インタフェースを図5.2 に示す．データベース部を複製することで，従来の管理インタフェースはどのデータベースにアクセスしているかが確認できなくなる．そこで図5.2 の赤丸で囲んだ部分のように対象のデータベースを選択し，検索に組み込めるようにした．検索に使用するデータベースのIP アドレスはブラウザのセッションごとにphp の変数内に保存されている．そのため，個々の端末で別のデータベースを選択し，閲覧することができる．はじめのアクセスの際は，指定されているデータベースのIP アドレスの中から死活を確認し，稼働しているうちのひとつのIP アドレスを使用する．

図 5.2:新たな管理インタフェース

5.2.2 監視インタフェース

MARS では，ネットワーク内の端末の認証情報はインタフェースで管理できるが，MARS を構成するサーバは，手動でping コマンドを利用して死活を確認するかSSH 接続を使用してプロセスの動作を確認するといった手法のどちらかで管理している．ping コマンドとは，対象からの応答を確認できるコマンドである．そのため，サーバの数だけその手順を繰り返さなければならない．そこで，それらのサーバをまとめて監視できるインタフェースを実装する．一般的に，サーバの管理はSNMP を用いて監視する．しかし，2.1.4 節で述べたように，このプロトコルは監視対象に対し，専用のSNMPマネージャをインストールしなければならない．今後サーバを追加する可能性があることからSNMP は使用せず，従来SSH 接続を使用して管理をおこなっていたことを考慮し，このインタフェースにはlibssh2[6] を使用する．libssh2 とは，php ファイルでSSH接続をおこなうためのライブラリのひとつである．これを用いて，php ファイルからSSH 接続をおこなう．従来のMARSの管理インタフェースはphp ファイルで作成されているため監視インタフェースはphp ファイルを用いて作成した．SNMP を利用する際と異なる点は，監視対象のサーバすべてにlibssh2 をインストールする必要はなく，インタフェースを設置するサーバにインストールされていれば良い点である．これにより，今後MARSに関するサーバが新たに設置された場合でも，そのサーバに対して専用エージェントをインストールする必要はなく，このインタフェース上への追加を容易におこなうことができる．すなわち，SSH 接続の許可とユーザ名およびパスワードがあれば，容易にこのインタフェース上に追加することができる．このインタフェースは管理インタフェースのあるインタフェース部のサーバに実装した．図5.3 のように，指定したIP アドレスへのping コマンドの結果，そのサーバのOS 名と

ホスト名，そのサーバ上で実行されているすべてのプロセスを取得し表示する．プロセスは指定したプロセス名と一致するものを色付けすることで見やすくした．

IPアドレス	ping疎通確認			uname -sn(OS host)	稼働プロセス
	min	avg	max		
133.42.	0.223	0.258	0.293	Linux wakayama-u.ac.jp	auditd crond httpd mysqld master rsyslogd
133.42.	0.168	0.202	0.236	Linux wakayama-u.ac.jp	auditd crond mysqld master rsyslogd
133.42.	0.257	0.416	0.576	Linux wakayama-u.ac.jp	acpid atd automount crond cupsd hald lldpad messagebus rpc.statd ntpd pcscd radiusd rpcbind rpc.idmapd rsyslogd
133.42.	0.543	1.119	1.696	FreeBSD wakayama-u.ac.jp	hostid hostid_save cleanvar ip6addrctl devd newsyslog syslogd dmesg virecover motd ntpd radiusd sshd sendmail cron mixer gptboot
133.42.	0.017	0.020	0.024	Linux wakayama-u.ac.jp	auditd crond httpd master rsyslogd

図 5.3:監視インタフェース

6. 実装・実験

6.1 実装

5 章の提案手法を現行のMARS への実装をおこなった．本節では，その構成について述べる．

6.1.1 ソフトウェア構成

5 章で述べた提案手法をCentOS[7]，Apache[8]，MySQL[9] を用いて実装した．従来のMARS サーバで使用しているOS はCentOS のため，新たに作成するデータベース部のサーバ，インタフェース部のサーバ，2 台目のRADIUS サーバは全てCentOS で作成した．Apache とMySQL はともに一般に広く普及している無償のオープンソースソフトウェアであり，信頼性が高いため利用した．従来と同様にデータベース部で認証情報の挿入のためのプログラムにPerl[10] を利用した．使用したソフトウェアのバージョンは表6.1 の通りである．

表 6.1: ソフトウェア

OS	CentOS 6.5
Web Server	Apache 2.2.15
Database Server	MySQL 5.1.73
Perl	Perl 5.10.1

6.1.2 エッジスイッチの設定

エッジスイッチには図6.1 の赤丸で囲んだ部分のように従来のRADIUS サーバをプライマリ，新たに作成したRADIUS サーバをセカンダリとして登録した．しかし，すべてのエッジスイッチがプライマリとして従来のRADIUS サーバを指定すると負荷に偏りが発生する．そのため，従来のRADIUS サーバをプライマリとするエッジスイッチと新たなRADIUS サーバをプライマリとするエッジスイッチを分けて設定した．

```
!
radius-server attribute 44 include-in-access-req
radius-server attribute 8 include-in-access-req
radius-server host 133.42.    auth-port 1812 acct-port 1813
radius-server host 133.42.    auth-port 1812 acct-port 1813
radius-server key 7
!
```

図 6.1:エッジスイッチへの RADIUS サーバの指定の例

6.2 実験

6.2.1 RADIUSサーバの故障時の正常運用

3.1.2 節で述べたとおり，RADIUS サーバは常に応答を返す状態でなければならない．そこで，片方のRADIUS サーバに障害が発生した際にも運用が継続できるのかを検証するため，故意にRADIUS サーバに障害を発生させる実験をおこなった．この実験の際，実環境での実験をおこなうと万が一の場合データの紛失も考えられるため仮想環境での実験をおこなった．仮想環境で使用したスイッチを下記に示す．これは実環境でも使用されているスイッチを使用した．

- Catalyst 2960

上記のスイッチに対して図6.1 の設定をおこなった．指定したRADIUS サーバ
も，表6.1 のソフトウェアを使用し仮想環境内に作成したものを使用した．この環
境で以下の実験をおこなった．

- RADIUS サーバのプライマリを停止させる
- RADIUS サーバのプライマリのRADIUS のプロセスのみを停止させる

このどちらの場合においても，スイッチはプライマリに対して認証が通らないと判断し，セカンダリを自動的に選択し，認証することを確認した．これにより，双方のRADIUS サーバがともに障害が発生して停止しない限り，エッジスイッチは認証が可能であることが確認できた．よって，RADIUS サーバのどちらかに障害が発生した場合でも，実環境のエッジスイッチも認証が可能であると考えられる．

6.2.2 データベースの整合性の確認

本研究では，データベースを複製することで障害が発生した際においても認証情報を閲覧できるようにした．そのため，複数あるデータベースはすべて同じデータを保持しておくことが重要となる．従来のデータベースサーバをデータベースA，新たに作成したデータベースサーバをデータベースB として，同時刻の取得状況を図6.2 と図6.3 に示す．データベースAとデータベースB での情報が一致していることが確認できた．データベースA とデータベースB の件数が一致しない点は，データベースA が以前から使用されていたもので蓄積されたデータが多いためである．この点については7.2 節で述べる．

図 6.2:データベース A のデータの取得状況

図 6.3:データベース B のデータの取得状況

6.2.3 サーバの障害発生時の監視インタフェースの表示

監視インタフェースには，サーバに障害が発生した際に視覚的に表示する機能がある．そこで，サーバに対して故意に障害を発生させ検証した．その際の表示についてを図 6.4 に示す．赤枠の部分のようにping コマンドで応答が確認できない際と，SSH 接続に失敗した際にその旨が表示されることを確認した．

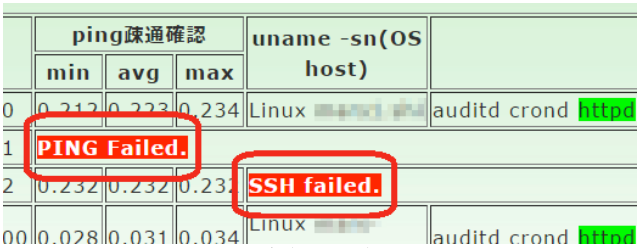

図 6.4:障害発生時の表示

7. 評価・考察・今後の課題

7.1 評価・考察

本節では，本研究で実装した冗長化構成のための手法の評価・考察について述べる．

7.1.1 従来のMARSとの比較

3.1.2 で述べたように，従来のMARS ではMARS サーバとRADIUS サーバは一台しかなく，そのどちらに障害が発生しても，障害対応に使用できなくなる．さらに，エッジスイッチが認証できなくなることで端末がインターネットに接続できないという弊害も発生する．しかし，本研究で冗長構成にしたことで，RADIUSサーバの一台に障害が発生した場合でもセカンダリで認証できるため，エッジスイッチに影響を与えることなく，運用を続けることができる構成となった．加えて，データベース部とインタフェース部を分割したことで，相互に影響を受けることなく認証情報の収集および閲覧を継続することが可能となった．よって，以前よりも障害に対して強固な構成になったと考える．

7.1.2 監視インタフェースの有用性

従来の手法では，サーバの数だけSSH 接続などを使用して死活監視やプロセス監視をおこなっていた．本研究でMARS を冗長化構成にしたことでにより，サーバの数が以前より増加した．しかし，監視インタフェースを導入したことでそのインタフェースにアクセスするだけで監視が可能になったため，サーバが増加したことによる管理者の負担を軽減することができた．

7.2 今後の課題

本節では，本研究の考察と今後の課題について述べる．

7.2.1 障害発生時のデータベースの整合性の保持手法の検討

データベースA とデータベースB がともに正常に稼働している場合には，6.2.2節で述べたように整合性が確認できた．しかし，このデータベースのいずれかに障害が発生した場合，その間の認証情報は障害が発生したサーバには保存されないため整合性が保持できなくなる．そのため，障害が発生したか発生していないかに関わらずある特定のタイミングでデータベースの内容を同期させる設計が必要であると考える．

7.2.2 データベースを複製したことによるフィルタリングの不一致

MARS のデータベースには，管理者が不正接続と判断したIP アドレスやセッションID などを保存しておくテーブルが存在する．データベースを複製したことにより，そのテーブルが複数できてしまい管理者はそのすべてのテーブルに，不正接続についての情報を書き込まなければならない．そのため，そのテーブルに変更があった際に他のデータベースにある同じテーブルにも反映させる設計が必要であると考える．

7.2.3 監視インタフェースの死活監視の精度向上

6.2.3 節で述べたように，監視インタフェースにはpingコマンドに失敗した際にその旨を表示する機能がある．しかし、この場合サーバ本体に障害が発生しているのかその経路に障害が発生しているのかはこの表示だけでは見分けることができない．この場合の経路とは，監視インタフェースのあるサーバから監視対象のサーバまでの経路のことである．そこで，川岸らの手法[11] を複合して用いることで，原因の所在を推定した尤度という値を管理者に提示し，経験の浅い管理者が使用する際にも，迅速に障害対応を可能にする設計が必要であると考える．

7.2.4 監視インタフェースのプロセス監視の精度向上

5.3 節で述べたように，監視インタフェースには指定のサーバ上で稼働しているプロセスを取得し表示する機能がある．しかし，プロセスは稼働していてもサーバ上にエラーのログが出力されることがある．この際，このインタフェースのみではログのエラーには対応できない．そこで，犬束らの手法[12] のようにログの解析をおこなうことでプロセスの稼働状況だけではなくログの状態も通知できる設計が必要であると考える．

8. おわりに

本論文では，先行研究であるMARS の障害発生時の問題やその冗長構成による管理の問題について言及した．既存の状態は，障害が発生した際に認証情報の閲覧ができなくなる可能性が存在した．そこで，MARS の各部について冗長化構成の提案と実装をおこなった．さらに，冗長化構成を実装したことで増加したサーバ群を，まとめて管理できる監視インタフェースの提案および実装をした．データベース部の複製した環境においても同一のデータの収集も確

認することができた．しかし，冗長化構成を実装したことによる新たな問題が発生した．例として，各データベースの認証情報の整合性の保持やフィルタリングテーブルの不一致などの問題がある．今後の課題として，認証情報やフィルタリングテーブルの整合性の保持についての対処が必要であると考える．

文　献

[1] 吉田祐亮 “ネットワーク接続監視システムMARS の構築”
2012 年度修士論文　和歌山大学大学院システム工学研究科
[2] Zabbix
http://www.zabbix.com/
[3] Nagios
https://www.nagios.org/
[4] Hinemos
http://www.hinemos.info/
[5] Remote Authentication Dial In User Service (RADIUS)
http://tools.ietf.org/pdf/rfc2865.pdf
[6] Secure Shell2
http://php.net/manual/ja/book.ssh2.php
[7] The CentOS Project
http://www.centos.org/
[8] The Apache HTTP Server Project
https://httpd.apache.org/
[9] MySQL
https://www-jp.mysql.com/
[10] The Perl Programming Language
https://www.perl.org/
[11] 川岸諒子，魚住光成
監視経路の冗長性に基づく障害原因箇所推定手法の提案
FIT2012（第11 回情報科学技術フォーラム），2012 年9 月
[12] 犬束敏信，浜田雅樹
ログ情報解析に基づくネットワークサービス監視システム
情報処理学会第58 回(平成11 年前期) 全国大会