第 III 部

ネットワークトラフィック統計情報の収集と解析

第 3 部
ネットワークトラフィック統計情報の収集と解析

第 1 章　MAWI ワーキンググループについて

MAWI（Measurement and Analysis on the WIDE Internet）ワーキンググループは、トラフィックデータの収集と解析を研究対象とした活動を行なっている。

MAWI ワーキンググループでは WIDE プロジェクトの特徴を活かした研究をするため、「広域」「多地点」「長期的」の三つの項目に重点を置いたトラフィックの計測・解析を行っている。広域バックボーンでのデータ収集はバックボーンを持っている WIDE プロジェクトだからできる事である。分散管理されるインターネットの状態を把握するためには、多地点で観測したデータを照らし合わせることが欠かせない。また、長期的にデータを収集し蓄積するために、ワーキンググループとしての継続的な活動が役に立つ。

計測技術はほとんどの研究分野で必要となるため、MAWI ワーキンググループは WIDE プロジェクト内の他のワーキンググループと連係をとりながら活動をしている。具体的には、

- グローバルな視点からの DNS の挙動解析（dns-wg と共同）
- IPv6 普及度の計測（v6fix と共同）
- ネットワークトポロジの観測（netviz-wg と共同）
- 長期的な経路変動の観測（routeview-wg と共同）
- AI3 の衛星トラフィックの計測（ai3-wg と共同）

などがあげられる。

また、国際協調として

- CAIDA (`http://www.caida.org/`)
- CNRS (`http://www.cnrs.fr/`)
- ICANN RSSAC (`http://www.icann.org/committees/dns-root/`)
- ISC OARC (`https://oarc.isc.org/`)
- USC/ISI (`http://www.isi.edu`)

などと共同して研究活動をしている。

第 2 章　MAWI ワーキンググループ 2007 年度の活動概要

今年度の報告書では、まず第 3 章において、計測に関する国際協調について報告する。現在、WIDE プロジェクトでは、CAIDA とフランスの CNRS との間で計測に関する包括的な共同研究を行なっていて、それぞれの組織と複数のテーマについて共同研究を進め、定期的なワークショップの開催や研究者交換を行なっている。

第 4 章では、CNRS の LAAS に交換留学した慶應義塾大学の金井君が留学中の活動およびハニーポット環境の構築研究について報告する。このような学生の交換留学は、本人にとって貴重な経験になるのと同時に、組織間の交流を促進し相互理解を深めるので、共同研究を円滑に進めるためにも有効である。

第 5 章では、計測システムの分散化に関して、その構築方法や評価手法について報告する。ここでは WIDE プロジェクトが開発している N-TAP システムを用いて、集中管理型、ピア・ツー・ピア型、ハイブリッド型の計測環境が構築可能である事を示し、N-TAP を使った評価実験の結果を報告する。

第 6 章では、広域計測基盤として、主に途上国に小型計測箱を設置しているガリバープロジェクトについて報告する。途上国からのデータはグローバルなインターネットの挙動を理解するために必要であるが、実際に途上国から計測するとなると、人的また物理的なリソースの不足のため、環境構築が難しい。そこでガリバープロジェクトでは、小型計測箱を設置し、遠隔管理することで、広域計測基盤の構築を目指している。

第 3 章　計測に関する 2007 年度国際協調活動報告

3.1 はじめに

WIDE プロジェクトは多くの国際協調活動を行なっているが、近年は計測研究の重要性が増している。これは、インターネット研究において、グローバルなレベルでその挙動を把握する必要性と難しさが認識されてきたためである。

現在、WIDE プロジェクトでは、CAIDA（the Cooperative Association for Internet Data Analysis）とフランスの CNRS（The Centre National de la Recherche Scientifique）との間で計測に関する共同研究を行なっている。

3.2 CAIDA との共同研究

CAIDA と WIDE プロジェクトは、2003 年度から計測に関する包括的な共同研究を行なっている。主なテーマは、DNS 計測、トポロジ計測、IPv6 計測、BGP 計測であり、年に 2 回程度ワークショップを開催し、相互の活動を理解し協力体制を作っている。

2007 年度には以下の 2 回のワークショップを開催した。

- 第 8 回 CAIDA-WIDE 計測ワークショップ
  2007 年 7 月 20–21 日 UIC/EVL
- 第 9 回 CAIDA-WIDE 計測ワークショップ
  2008 年 1 月 19–20 日 Univ. Hawaii

2007 年度の主な活動を以下にあげる。

- 計測データの目録化
  計測データの研究利用を促進するため、CAIDA が中心となり各組織の持つ計測データを目録化するプロジェクトを進めている。主に、2007 年 1 月に CAIDA と WIDE プロジェクトが中心になり実施した、インターネット計測デーに収集したデータの目録化を行なった。
- 地理情報を考慮したトポロジ解析
  CAIDA の持つ広域 traceroute データをもとにして、WIDE プロジェクトが地域別の AS トポロジの解析を行なっている。
- 広域計測基盤
  主に開発途上国からの計測を行なう目的で、WIDE プロジェクトが小型計測箱を設置、遠隔管理する計画を進めている。
- 研究者交換
  CAIDA の Brad Huffaker が 2007 年 8 月から 2 カ月間日本に滞在し、広域計測基盤に関して共同研究を行なった。

2008 年度もこれらの共同研究活動を継続する予定である。

3.3 CNRS との共同研究

2006 年より、フランスの大学連合である CNRS と WIDE プロジェクトは、計測とモビリティの 2 つの分野において 3 年間の共同研究を行なっている。共同研究 2 年目の今年は、相互の技術を組合せた研究への取り組みに重点を置いた活動を行なった。

計測グループでは、ゲームや P2P などの新規アプリケーションやセキュリティ攻撃を計測、モデル化することをテーマとして共同研究を行なっている。より具体的には、以下のような研究活動を行なっている。

(1) アプリケーション識別
フランス側 LIP6 の Salamatian 教授のグループが開発した、パケットの先頭数十バイトの情報からアプリケーションのタイプを識別する技術を日本側のデータを使って検証を行なっている。

(2) 時系列データ解析
フランス側 ENS Lyon の Patrice Abry のグループと日本側国立情報学研究所の福田准教授が、時系列トラフィックデータをモデル化し、定常時と異常時のパラメータ変化の違いに着目し、セキュリティ攻撃等を自動で検出する共同研究を行なった。次のステップとして、日本側の蓄積データに含まれる異常トラフィックの目録化に着手している。これにより既存データの研究利用の促進が期待できる。

(3) ハニーポットによるセキュリティ攻撃の検出
フランス側 LAAS Philippe Owezarski のグループのハニーポットを日本側にも設置し、日仏で同時に観測する事によって、広域に渡る攻撃を検出することや、地域差を明らかにする共同研究を実施している。双方の技術を組合せより精度を高める研究を進めている。

(4) 分散計測基盤
広域分散計測基盤について、双方で研究を進めている。

2007年度は、10月にLyonでワークショップを開催し、研究の進捗報告や、学生交換の成果報告等を中心に発表を行ない、今後のスケジュールを確認した。また、2008年3月には、奈良先端科学技術大学院大学でワークショップの開催を予定している。

- 第4回CNRS-WIDEワークショップ
  2007年10月15–16日ENS-Lyon

また、2007年度は以下の研究者交換を行なった。

- LIP6のSalamatian助教授の学生Thomas Silverstonが2007年7月から9月まで日本に滞在。東京大学で受け入れ、ピア・ツー・ピア型IPTVの挙動に関して、日仏双方でデータを収集し解析を行なった。
- 慶應義塾大学の学生金井瑛君が2006年9月から11月までLAASを訪問。ボット検出手法について研究を行なった。
- LAASのPhilippe Owezarski氏の学生Ion Alberdiが2007年12月に3週間日本に滞在、慶應義塾大学の村井研究室で受け入れ、ボット検出手法の研究を行なった。
- ENS LyonのPatrice Abryの研究員Guillaume Dewaeleが2007年8月から3週間、2度目の日本滞在をした。国立情報学研究所福田准教授が受け入れ、時系列解析に関する共同研究を行なった。
- ENS LyonのPatrice Abryの研究員Pierre Borgnatが2007年12月に3週間、日本滞在をした。国立情報学研究所福田准教授が受け入れ、時系列解析、特に長期的なトレンド解析に関する共同研究を行なった。

さらに2007年度は、以下の2本の共同執筆論文を国際会議で発表した。

- Guilaume Dewaele, Kensuke Fukuda, Pierre Borgnat, Patrice Abry, Kenjiro Cho. Extracting Hidden Anomalies using Sketch and Non Gaussian Multiresolution Statistical Detection Procedures. ACM SIGCOMM2007 LSAD Workshop, Kyoto Japan. August 2007.
- Thomas Silverston, Olivier Fourmaux, Kave Salamatian, Kenjiro Cho. Measuring P2P IPTV Traffic on Both Sides of the World. CoNEXT Student Workshop, NY, NY. December 2007.

2008年度は、共同研究の最終年度であり、より活発な研究活動を行なって成果を出していく予定である。

3.4 まとめ

インターネットの計測研究では、国際的な協調による広域なデータ収集、しかも長期に渡る地道な努力が重要である。今後は、これまでに築いた関係をベースに、さらに協調の幅を広げると同時に、具体的な成果を出す努力をしていく。

第4章　WIDE-CNRS間の交換留学活動報告

4.1 概要

WIDEプロジェクトはフランス国立科学研究センター（CNRS）との研究協力の一環として、両組織間で人的交流・学術的交流を目的とした、学生の交換留学制度を設けている。私はこの交換留学生として、2007年09月18日から同年11月10日にかけて約2ヶ月間渡仏した。この期間中、私はフランス南部トゥールーズにあるLaboratoire d'Architecture et d'Analyse des Systemes（LAAS）で受け入れていただいた。期間中は主に、LAASの研究者であるPhilippe Owezarski氏と博士課程学生であるIon Alberdi氏に研究の体制を用意していただいた。とくにIon氏と私はボットネットを共通のテーマとして取り扱っているため、多くの知識を共有し、また、今後に向けた研究体制を整えることができた。

4.2 イベントへの参加

本留学では、LAASでの共同研究に加えて、パリで開催されたMozilla24中継スタッフとしての参加およびCNRS-WIDEミーティングへの参加をした。本節ではそれぞれの取り組みについて述べる。

4.2.1 Mozilla24中継スタッフ

私は、期間中の最初の10日間をMozilla24の中継スタッフとして参加するため、パリに滞在した。Mozilla24は、日本、フランス、イギリスとタイの4ヵ国間をネットワークで接続し、各会場からの講演やディスカッションを実現するイベントである。このイベントは、ヨーロッパの大手学術ネットワークの1つであるRenater、パリを拠点とする大手通信事業者TELECOM ParisTech ENST（以下ENST）、

映像のプロフェッショナル組織である CERIMES の協力を得て実現した。

Mozilla24 はパリ市内の ENST で開催された。到着後、ENST の Pierre Beyssac 氏、Renater の Simon Muyal 氏、同組織の Virginie BLANQUART 氏及び Mozilla ヨーロッパ代表の Tristan Nitot 氏らとミーティングを行った。到着後から 1 週間の準備の間、会場の映像環境及び、日本間との通信環境について確認し、低遅延と高品質な映像・音声を提供できる DVTS を利用した環境を整備した。Mozilla24 では日本を拠点として各国と映像と音声の通信を行うため、ENST からは日本との双方向通信が必要であった。

イベント実施日は約 40 人の受講者がフランス会場に集まり、4 名の講演者がフランスより講演を行った。また、受講者は日本を介したタイや、東京からの講演を聴講した。イベント中はとくに大きな障害も発生せず、イベントは成功を収めた。

イベントを通して、Renater や Mozilla、ENST のメンバと交流を深めることができ、今後のフランスで行う CNRS-WIDE の研究活動をより円滑に行えることが期待できる。

4.2.2 CNRS-WIDE ミーティングへの参加

2007 年 10 月 15 日から 16 日にかけて、フランス中部の Lyon で第 4 回の CNRS-WIDE ミーティングが開かれた。会期が留学期間中であったため、私は LAAS から参加した。本ミーティングでは、WIDE から 11 名、CNRS から 22 名の研究者が参加し、measurement と mobility の 2 部構成でそれぞれの研究成果を発表し合った。本ミーティングでは、CNRS 側の measurement の研究者と話す機会が多くあり、インターネット上の攻撃特性に関する情報や、DDoS 検知の手法などについて、有益な議論を交わすことができた。

4.3 成果

本留学中に私は LAAS の Philippe 氏からご指導を頂きながら、Ion 氏との共同研究に取り組んだ。本節では、私が LAAS で取り組んだ活動の成果について述べる。

攻撃者により管理された多数のノードが構成するボットネットはその可変性と分散性などから対応が極めて難しい。私はこれまで、フロー型検知手法を用いたボット検知手法を研究してきた。フロー型検知手法はホスト毎のフローを保持し、その順序を追跡してボットを検知する。また、Ion 氏は世界中に分散したハニーポットを用いて効果的にボットの情報を取得する研究や、取得したボットを安全に実行する研究に携わってきた。Ion 氏や私のようなボットの研究者は、日々進化するボットの知識を得るために、ハニーポットを用いている。**ハニーポット**は脆弱性を持つホストの挙動を振る舞うソフトウェアであり、ボットネットの研究を進める上で有効なツールである。研究では運用コストや法的な問題から低インタラクション型のボットネットが用いられることが多い。しかし、一般的な低インタラクション型の環境から得られる情報は、セッション型ボット検知機構に対して十分な情報とならない。

本留学では、この問題を解決するトラフィック合成機構の実用化に向けた開発を行った。

4.3.1 現在の問題点

まず、一般的な低インタラクション型ハニーポットを用いたボットトラフィック収集の環境を図 4.1 に示す。

図中では、IP アドレス IPa を持つマシン上で低インタラクション型ハニーポットが稼働している。ハニーポットの稼働しているマシンの通信はすべてトラフィックミラーポイント MPa でミラーリングされ、蓄積される。MPa のトラフィックダンプデータにはハニーポットが対応したすべてのボットウェアに対してのトラフィックが蓄積される。ハニーポットで蓄積したボットウェアは手動で IP アドレス IPb を持つ実行マシンにコピーする。法的な問題から実行マシンからのトラフィックは管理者の設定したポリシーによって、フィルタアウトされる。また、実行マシンの通信はトラフィックミラーポイント MPb によってミラーリングされ、蓄積される。

2 つに分かれた PCAP ファイルのパケットは例えば *tcpmerge* などの既存のソフトウェアを利用して合成できる。*tcpmerge* は 2 つの PCAP ファイル内のパケットの時間情報を保持したまま、パケットを結合する。しかし、結合に際して、**PCAP ファイルの持つ情報の差**の問題が発生する。また、問題を解決しトラフィックを結合したのみでは **IP アドレスの差**と**時間情報の差**の 2 つの問題が発生する。これら 3 つの問題は、フロー型ボット検知機構の評価に大きな影響を及ぼす。

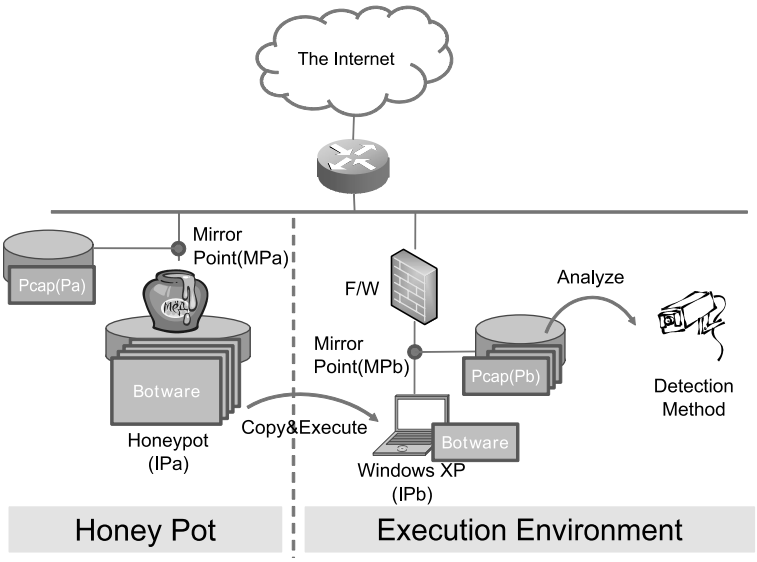

図 4.1. 一般的なハニーポット環境の例

- **保持されているボットネット情報の差**

  MPa のトラフィックダンプデータにはトラフィックをダンプしている期間中のあらゆるボットウェアのトラフィックが含まれる。しかし、MPb のトラフィックダンプデータは実行毎にトラフィックをダンプするため、ボットウェア毎に分かれたダンプファイルとなっている。現在、私が取り組んでいるフロー型ボット検知手法の評価に利用するには、攻撃からボットの活動トラフィックまで全てのトラフィックが必要である。そのため、MPa のデータからボット毎のトラフィックを抜き出す必要がある。

- **IP アドレスの差**

  ハニーポットマシンと実行環境マシンは保持する IP アドレスが異なる。そのため、トラフィックを結合しただけでは実際とは異なったマシンが通信しているようなトラフィックが生成される。たとえば、図 4.1 の例では、ダウンロードまでのトラフィックは IPa が、ボットウェアの実行トラフィックは IPb が外部と通信をしているトラフィックとなる。

  フロー型ボット検知機構では、内部ホストのフローの順序を利用して検知するため、IPa と IPb の行う別の通信として扱われ、正常な評価が実現できない。

- **時間情報の差**

  実行環境トラフィックは、ボットウェアを収集した直後に収集されていないので、実際の動作と時間情報に差が出る。この様子を図 4.2 に示す。ただし、$Tr1$ はハニーポットトラフィック、$Tr2$ は実行環境トラフィック、$Tr3$ は時間情報を保持したまま結合したトラフィック、$Tr4$ は実際のトラフィックとする。また、$ts1$ はハニーポットトラフィックの開始時刻、$te1$ はハニーポットトラフィックの終了時刻、$ts2$ は実行環境トラフィックの開始時刻、$te2$ は実行環境トラフィックの終了時刻とする。図はそれぞれのトラフィックに含まれるパケットを時系列に表現している。Δt は式 (1) のように表わされる。

$$\Delta t = ts2 - te1 \tag{1}$$

一般的なハニーポット環境においては、ボットウェアのダウンロード時間とボットウェアの実行やその準備に時間を要するため、$Tr2$ の全トラフィックは実際に比べて実行までに要した時間 Δt だけ遅れた時間をもつパケットとなる。実際のトラフィックでは、ボットウェアのダウンロード後、即座にボットウェアが実行されるため、$te1$ と $ts2$ の差は極めて小さい。

フロー型ボット検知機構を含む多くのトラフィック監視するセキュリティ機構では、フロー間の間隔が長いとそれらを連続したフローとして扱わない。そこで、$Tr2$ は Δt だけ時間を早くして $Tr1$ と結合し、実際のトラフィックに近づける必要がある。

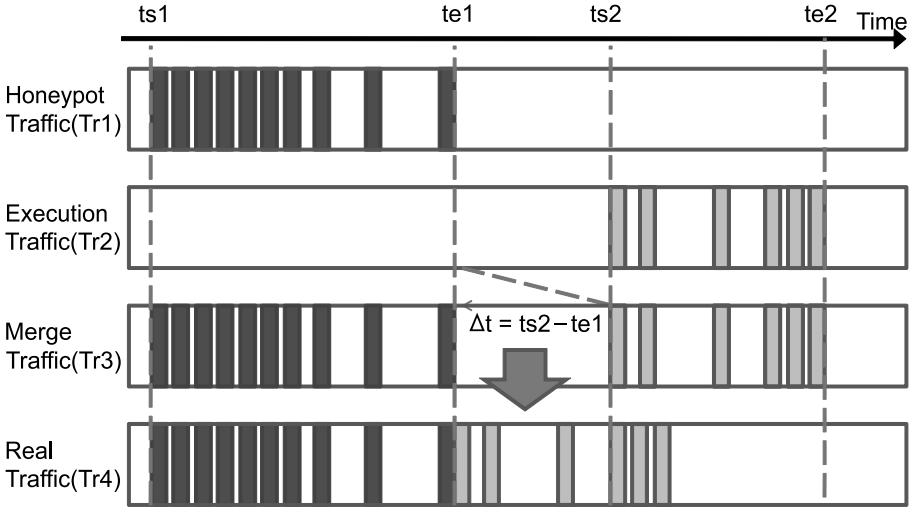

図 4.2.　実機のトラフィックとハニーポット環境のトラフィックに生じる時間情報の差

4.3.2 アプローチと実装

本留学では、前節で挙げた問題を解決するために、ハニーポットへのパッチおよびいくつかのネットワークツールを作成し、それらを組み合わせて問題を解決したトラフィックを合成するプロトタイプ実装を行った。

- **ハニーポットへのパッチ**

  現在、私が WIDE ネットワーク上で用いているハニーポットと Ion 氏が LAAS ネットワーク上で用いているハニーポットはオープンソースソフトウェアの低インタラクション型ハニーポット Nepenthes である。保持されているボットネット情報の差の問題を解決するためには、ハニーポットトラフィックからボット毎のトラフィックを抜き出せなければならない。これには、ハニーポットの出力するログを利用する。しかし、Nepenthes はボット毎のトラフィックを抜き出すのに必要な情報を出力しない。そのため、私は Nepenthes に対して必要な情報を出力するように変更を加えた。変更が必要な情報を挙げる。

  - **UNIXTIME**

    各ログは発生した時間を現地時間で保持している。しかし、PCAP ファイルは UNIX タイムでパケットの発生時間を保持している。そのため、時間情報の統一のためにログに UNIXTIME を保持させるようにした。

  - **IP アドレスとポート番号の情報**

    ログからパケットを抜き出すには通信に関する IP アドレス情報とポート番号が必要となる。そのため、各ログに関連する IP アドレスとポート情報を保持させるようにした。

  - **識別子**

    Nepenthes は攻撃以降のイベントがどの攻撃イベントによって発生したかを保持しない。たとえば、ボットの本体をダウンロードするイベントがどの攻撃によって発生したかが不明である。あるボットウェアのトラフィックのみを適切に抜き出すにはログの関係が分かる必要がある。そのため、各ログ間の関係をユニークな識別子を各ログに保持するようにした。

  この結果、図 4.3 のようなログが出力される。この例では、各ログに UNIXTIME が記述され、ネットワーク通信を行ったログには、通信先の IP アドレスと使用したポート番号が記述されている。また、それぞれのログは UniqueId と定義されたユニークな識別子を保持しており、これらが関連したログであることが分かる。

- **PCAP Address Replacer**

  PCAP Address Replacer は C 言語で記述されたパケット内部の IP アドレスと MAC アドレスを書き換えるコマンドラインソフトウェアである。パケットに含まれるコマンドラインから指定したアドレスを置き換え、また、IP チェックサムおよび、L4 のチェックサムを再計算する。IP は IPv4 および IPv6 をサポートする。また、L4 プロトコルは TCP と UDP をサポートする。

```
(12102007 22:21:36 info handler dia) Time: 1192195296,
Info: KNOWN DCOM(2) ATTACK RECEIVED, UniqueId: 1,
Local: 203.178.xxx.xxx:135, Remote: 203.174.yyy.yyy:1258
12102007 info mgr submit Time: 1192195331

(12102007 22:21:36 info handler dia) Time: 1192195296,
Info: BLINK DOWNLOAD START, UniqueId: 1,
Local: 203.178.xxx.xxx:33286, Remote: 203.174.yyy.yyy:54791

(12102007 22:22:11 info mgr submit) Time: 1192195331,
Info: SUBMITED, UniqueId: 1,
FileName: 2aa59ba4251795deda72738d1c67be7c,
FileType: MS-DOS executable PE for MS Windows (GUI) Intel 80386 32-bit
```

図 4.3. パッチを当てた Nepenthes のログ例

- **PCAP Appender**
  PCAP Appender は C 言語で記述された 2 つの PCAP ファイル内のパケットの時間を調整し、それぞれの PCAP ファイルを結合するソフトウェアである。このソフトウェアは、infile1 のトラフィックを $Tr1$、infile2 のトラフィックを $Tr2$ として扱い、$Tr2$ の各パケットの時間軸を $-\Delta t$ ずらすことで、$Tr4$ に近いトラフィックを生成して、outfile に書き出す。
- **PCAP Slicer**
  PCAP Slicer は保持されているボットネット情報の差を解決するためのシェルスクリプトである。本スクリプトはパッチを当てた Nepenthes のログから、ボット毎の攻撃フローとダウンロードフローの通信先 IP アドレスとポート番号を抜き出す。そして、ハニーポットのトラフィックからそれぞれのトラフィックを抜き出し、PCAP Address Replacer と PCAP Appender を用いてトラフィックを合成する。

4.3.3 現状と今後

前節の実装は大きくパッチを当てたハニーポットと、トラフィック合成プログラムの 2 つに分かれる。このうち、パッチを当てたハニーポットは、WIDE ネットワーク内のハニーポットおよび、Ion 氏の協力を得た LAAS ネットワーク内のハニーポットで稼働中である。

我々は、将来的には収集したボットウェアを自動的に実行しトラフィック合成できることを期待している。しかし、ボットウェアの自動的な実行は、法的な側面や、外部への攻撃を防ぐ難しさの障害から、今回取り組むことが出来なかった。

現在、Ion 氏は自動なボットウェアの実行に利用できるファイアウォールの研究に取り組んでいる。今後も共同研究を進めていき、本成果と Ion 氏の成果を組み合わせて、自動的なボットウェアトラフィックの収集機構の構築を目指したい。

4.4 まとめ

本留学では、Mozilla24 への中継スタッフとしての参加及び、LAAS での共同研究に取り組んだ。Mozilla24 では Renater、ENST や Mozilla のメンバと共に作業に取り組み、また交友を深めた。これにより、フランスにおける研究だけではない様々な活動において、WIDE プロジェクトがより円滑に協力できる関係を築けた。LAAS での共同研究では、Ion 氏との共通の問題点に取り組み、また LAAS のスタッフと交友を深めることができた。トラフィック合成に関する問題に対する取り組みは、今後の双方の研究活動に有用なものであり、その方向性について議論できたことは非常に有意義な経験であった。現在、Ion 氏は WIDE ネットワークにハニーポット環境を構築しており、また、私も今回の留学で LAAS の実験ネットワークにハニーポット環境を構築させていただいている。トラフィックを監視するという活動の性質から、双方の交友が深まったことは今後の活動の際にも貴重な機会であった。今後もお互いの情報を共有しつつ、研究を進めていきたい。

第 5 章　A Role-Based Peer-to-Peer Apprach to Application-Oriented Measurement Platforms

Abstract

The importance of large-scale measurement infrastructures for grasping the global state of the Internet is recently strongly emphasized. However, a fundamental analysis of these infrastructures has not yet been conducted. In this report, we highlight the formation of measurement networks and provide a first look at measurement activities performed on those networks. We also propose a scheme for constructing a measurement network, which divides the measurement agent's roles into core agent and stub agent. This scheme entails only simple adjustment for changing the formation of the measurement network. Through the transition from a centralized system to hybrid and pure peer-to-peer networks, we visualize the flow of measurement procedures and explore the factors that have an influence on the overall performance of measurement systems.

Key words: peer-to-peer network, network measurement platform

5.1 Introduction

Large-scale network systems that include an overlay network application and a distributed computing environment have not yet fully succeeded in obtaining network characteristics on the Internet. Network characteristics are necessary information for sustaining and scaling the services of these systems. For example, an IP level topology and round trip time (RTT) information between two nodes are used as the metrics of the proximity among overlay nodes, and overlay network applications perform their optimization procedures based on these metrics. However, due to the complicated procedures of measurement and data processing, network characteristics are not yet widely utilized by the applications.

Given this situation, application-oriented measurement services[97, 100, 187] have been appearing. In these systems, measurement procedures are typically packaged into one independent network service, and applications need only issue a request to the systems in order to obtain network characteristics. Monitoring nodes are located in multiple administrative domains, and the systems manage and control them to obtain the requested data. These systems have emerged as a way for the applications to grasp the global state of the Internet.

This tendency has also brought a radical shift in the architecture of measurement systems. Traditional measurement infrastructures generally prepare a central management plane. In such systems, respective monitoring nodes perform measurement independently or according to the decision made by a central control plane. Then the system aggregates collected data in the central storage. This scheme worked well within the statistical observation of the Internet for mid-to-long-term. In the case of application-oriented measurement, measurement targets (e.g., nodes and links) disperse widely and change dynamically depending on the structure of application networks. In addition, collecting a large number of network characteristics with one control point causes a heavy load on specific nodes. For these reasons, the architecture of measurement systems has become more decentralized, and measurement methodologies performed in a decentralized manner have been studied. Some of these measurement methodologies are called "cooperative measurement,"[29, 38] in which a monitoring node shares collected network characteristics and/or communicates with other agents for more efficient collection of network characteristics.

Though a variety of measurement systems has been proposed, and these systems focus on the sorts of network characteristics that they can collect, their architectures are yet to be sufficiently explored. We do understand the superficial

indices, such as the capability and efficiency of measurement methodologies, that are implemented on the systems; however, we do not know which aspects of the architectures bring such results and how they influence actual deployment. This problem cannot be left unsolved before a large-scale measurement service is widely deployed, because such analysis could reveal fundamental drawbacks and advantages of the measurement infrastructures.

In this report, we focus on the structure of a measurement network as one aspect of the architecture. The measurement network is a network in which measurement procedures are performed and measurement-related information is managed. One of the structures focused on is a centralized structure, where a specific management node manages all of the management information and controls the other monitoring nodes. Another is a pure peer-to-peer structure, where all of the monitoring nodes take partial charge of the management node's tasks. Moreover, we propose a hybrid structure, where some of the nodes work as management nodes and the others work as ordinary monitoring nodes. We explain how the respective measurement networks and the monitoring nodes in the networks actually work, and we investigate their basic characteristics related to their responsiveness and load distribution.

The rest of this report is organized as follows: Section 5.2 describes measurement network models including a centralized model, a pure peer-to-peer model and our proposed hybrid model. We describe the experiment in Section 5.3, and in Section 5.4, we look into the experimental results and investigate the basic characteristics of the respective measurement networks. Section 5.5 presents a discussion on application-oriented measurement services and the formation of their networks, based on evaluation in the previous section. We refer to related work in Section 5.6, and finally conclude this report in Section 5.7.

5.2 Measurement Network Models

In this section, we first define the components of a measurement network and how they work and interact with other entities. In Section 5.2.2, we describe two existing models of measurement networks — centralized and pure peer-to-peer models. We also refer to a hybrid measurement network model in the same section. Finally in Section 5.2.3, we propose a methodology to allow shifting a measurement network between these models, and we describe its implementation on an actual measurement system.

5.2.1 Components of the Measurement Network

A measurement network is a network in which measurement procedures are performed according to predefined sequences. Here we define the entities that appear in a measurement network and its operation.

The first entity is a "monitoring node," which performs measurement procedures in order to collect network characteristics. The second entity is a "management node," which is responsible for coordinating other entities so that the intended measurement can be performed. For example, the management node inspects and updates "management information," such as the list of monitoring nodes, and commands some of the monitoring nodes to perform measurement procedures. Collectively, we call a system that is composed of management information and management nodes a "control plane." A control plane is, so to speak, an entity where decisions for measurement procedures are made. "Control messages" are exchanged among the monitoring and the management nodes to achieve the intended measurement features. The control messages include a measurement command to the monitoring nodes and the node list in the measurement network, but do not contain the network traffic derived from the measurement procedures themselves. We note that one physical node may simultaneously

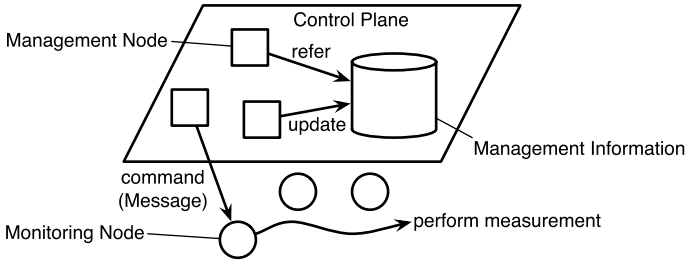

Fig. 5.1. Components in the measurement network and relationships among them.

play the roles of both management and monitoring. Figure 5.1 shows the relationship among the entities described in this paragraph.

5.2.2 Three Types of Models

Existing measurement networks are categorized mainly into two models — centralized and pure peer-to-peer models. In the centralized model, one management node or a cluster of replica nodes manages all of the management information and issues control messages to the monitoring nodes. On the other hand, in the pure peer-to-peer model, all of the nodes take the roles of both monitoring and measurement. Therefore each node has to maintain the measurement network and has also to perform the necessary measurement procedures. The merit of the centralized model is that the responsibilities of the respective nodes are clear, and it is easy to follow the sequence of measurement operations. At the same time, a central management node has to tolerate a heavy load caused by all the management operations, otherwise the measurement system will not function. In the pure peer-to-peer model, we can distribute such loads to all nodes; hence this model is considered appropriate for a large-scale measurement system. However, a frequent change in the state of the measurement network, such as nodes joining and leaving, will result in poor stability of the control plane. These trade-offs are also discussed as a general problem existing between centralized and peer-to-peer systems.

As a middle course between these models, we now consider a hybrid measurement network model. In the hybrid model, management operations are divided among some management nodes, while other nodes behave as monitoring nodes. The difference between the hybrid model and the centralized model is that, in the hybrid model, multiple management nodes each perform a different management operation, whereas the management operations are not clearly divided in the centralized model even if there are multiple management nodes. By adopting this model, we can expect to moderate both the load concentration and the instability of the measurement network, which are the problems in the first two models. This model is similar to that of the Kazaa[83] network, in which stable nodes (called "super nodes") construct an overlay network in a peer-to-peer manner, and ordinary nodes join the overlay network through the super nodes.

5.2.3 N-TAP and its Extension

N-TAP[1][101, 102] is a distributed measurement infrastructure that provides an application-oriented measurement service. In N-TAP, a program called an "N-TAP agent" performs measurement procedures. The N-TAP agents also construct a pure peer-to-peer measurement network (called the "N-TAP network") that is based on the technique of Chord[158]. In the context of Section 5.2.1, the N-TAP agent corresponds to a node that works as both a monitoring node and a management node. The management information in N-TAP is stored and shared in a shared database that the N-TAP agents construct upon their peer-to-peer network. Besides

1　Available at http://www.n-tap.net/.

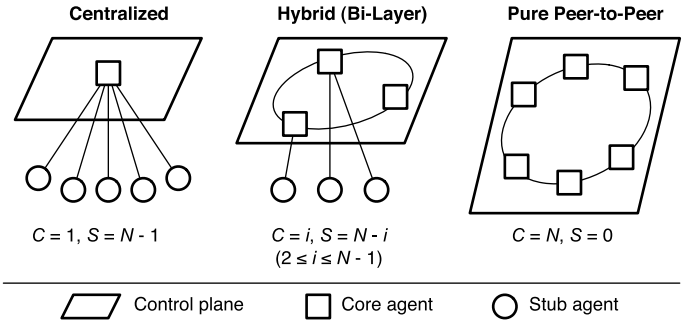

Fig. 5.2. Measurement network formations with the scheme of core and stub agents ($N = 6$, $i = 3$).

the nodes (agents) list being maintained in the shared database as the management information, collected network characteristics are also stored in the same database so that the agents can share them in cooperative measurement. The N-TAP agents decide measurement tactics according to the "local-first and remote-last" rule, which improves the responsiveness to measurement requests from applications. In order to create a situation of a hybrid measurement network on an actual measurement system, we made some modifications on N-TAP.

The key idea of the extension to N-TAP is the division of the agent's roles into core agent and stub agent. The core agent, which corresponds to the management node, constructs the measurement overlay network, called the N-TAP network, as conventional agents did: it maintains its own routing table in the Chord ring and stores some of the shared data in a local database as a part of the shared database. The core agent also performs measurement as a monitoring node if necessary. The stub agent, which is equivalent to the monitoring node, does not perform the operations related to the construction of the N-TAP network. For joining the N-TAP network, the stub agent inserts its information in the shared agent list so that other agents can find it. It performs measurement only when a core agent sends a request to it or when it knows that the measurement procedures that are requested directly from applications should be done by itself. In the case that the stub agent needs to do the operations related to the N-TAP network, it sends a request to one of core agents, and the core agent responds to the request. For example, suppose that a stub agent wants to find a core agent that is responsible for a specified ID in the Chord ring so as to retrieve a shared data entry that has this ID; the stub agent asks a core agent to find the responsible agent, and the core agent performs the procedure of finding it. After the core agent obtains a result, it sends the result to the stub agent. In this way, even a stub agent, which does not perform the management procedures for the N-TAP network, can know the state of the N-TAP network.

By adopting the scheme of core and stub agents, we can also easily form the centralized and pure peer-to-peer measurement networks. Figure 5.2 shows the transitions of measurement networks according to the allocation of the respective numbers of core and stub agents. Now we have N agents, and C of N agents are assigned as core agents; i.e., S $(= N - C)$ agents are stub agents. The N-TAP network where $C = 1$ is equivalent to a centralized measurement network because all of the management information is concentrated in one core agent. If we take the value of $C = N$, all of the agents are core agents; therefore the N-TAP network in this situation is a pure peer-to-peer network, which is same as the conventional N-TAP network. In case of $C = i$ $(2 \leq i \leq N - 1)$, we can regard the N-TAP network as the hybrid measurement network.

As described in this section, we can now have three types of measurement networks on the actual measurement system. In the following

sections, we investigate the basic characteristics of these measurement networks.

5.3 Experiment

For this experiment, we used 128 homogeneous nodes in StarBED[114], which is a large-scale network experiment facility. Each node had an Intel Pentium III 1 GHz CPU, 512 MB memory and a 30 GB ATA hard drive. These nodes were connected through 100 Mbps Ethernet links in the same network. The Debian GNU/Linux operating system with the 2.6-series kernel was installed on each node.

We had one N-TAP agent run on each node; therefore we constructed a measurement network with 128 agents (i.e., $N = 128$). An N-TAP ID, which puts an agent in the Chord ring, was randomly assigned to each agent with no overlaps. The reason we chose random IDs was to distribute the load derived from maintaining the N-TAP network among the core agents in the hybrid and pure peer-to-peer measurement networks. After the N-TAP network was constructed, we ran a client program on one node that is in the same experimental network and did not have an agent. The program sequentially issued 2000 requests to one of the core agents for the RTT information between two randomly chosen experimental nodes. The program also issued the same number of the requests to one of the stub agents if the N-TAP network had stub agents. The request messages were exchanged based on the XML-RPC protocol between an agent and the client program. We note that an N-TAP agent usually tries to reuse RTT data previously collected and stored in the shared database if a client program specifies the request on the freshness of the RTT data. However, for simplicity in this experiment, we forced the agents not to reuse the RTT data but to perform the actual measurement. The agents logged their operations with time stamps, and N-TAP related packets were captured on the nodes, so we were able to analyze the behavior of the measurement network. We selected the values of 1, 2, 4, 8, 16, 32, 64 and 128 for C (the number of core agents) to shift a measurement network from the centralized one to the decentralized one. For convenience, we numbered the respective agents from 1 to 128 according to the following rules: (a) The first agent was a core agent that accepted and processed the above requests. (b) If there were other core agents, they were numbered from 2 to C. (c) If there were one or more stub agents, we set a stub agent that accepted and processed the above requests as the 128th agent. (d) If there were other stub agents, they were numbered from $C + 1$ to 127. Also note that the 128th stub agent was configured to issue a request related to the N-TAP network to the first core agent.

The procedures carried out by a core agent when it accepted an RTT measurement request are as follows (see [101, 102] for detailed operational flows on an N-TAP agent):

1. The core agent searches the source node in the requested RTT measurement. In this procedure, the core agent issues a request to find a core agent that is responsible for storing the data entries on the source node in the shared database. After it finds a responsible agent, it asks the agent to send the information on the source node (for instance, whether the source node is alive or not).
2. If the source node is alive (this condition is always true in this experiment), the core agent asks the source node to measure the RTT. Then the source node sends the measurement result to the core agent.
3. On receiving the result, the core agent responds to a client program with this result.
4. The core agent stores the collected RTT information in the shared database. It finds another core agent, one that is responsible for storing this data entry, and sends the entry to the responsible agent.

In the case of a stub agent, a control message to find a responsible agent was always sent to a specific core agent because the stub agent did not

have a routing table in the N-TAP network but only knew the core agent that bridged between the N-TAP network and the stub agent itself. Apart from this messaging manner, the stub agent behaved in a same way as a core agent.

After the experiment, we confirmed that no measurement error had occurred and that all of the N-TAP related packets had been correctly captured during the experiment. The evaluation carried out in the following section is based on the recorded behavior of the agents after the measurement network became stable, i.e., no change in the agents' routing tables were made.

5.4 Evaluation

In this section, we investigate the basic characteristics of the respective measurement networks shown in Section 5.2. Our focus is the load distribution and the responsiveness to a measurement request in measurement networks.

5.4.1 Load Distribution

First we investigate the flow of control messages in the N-TAP network. Since the N-TAP agents have to carry out procedures according to the control messages, we can determine the distribution of loads among the agents by seeing this flow. Figure 5.3 depicts the distribution of exchanged control messages among the agents. Its horizontal axis denotes the assigned numbers of source agents in control messages, and the vertical axis denotes the assigned numbers of destination agents. The colored squares in the graphs show the number of the messages by their darkness: dark gray indicates more messages were exchanged and light gray means fewer. Specifically, where we define M as the logarithm of the number of exchanged messages, we divide the zone of the values of M into four even intervals and assign four shades of gray to the respective intervals so that the zone of the largest value of M is the darkest; a white area means that no message was exchanged between the agents. The horizontal and vertical dotted lines indicate the borders between the core agents and stub agents; therefore the bottom-left area shows the messages exchanged between two core agents, the bottom-right and top-left areas are for the messages between a core agent and a stub agent, and the top-right area is for the messages between two stub agents.

In any case, we can confirm that the squares are plotted more densely in the bottom-left area than in other areas, and the grays there are mostly dark. This shows that the burden of maintaining the measurement network was concentrated on the core agents, and the stub agents were relatively freed from such tasks. Additionally, no message was exchanged between two stub agents except for the cases of involving the 128th agent. The reason why the number of the messages to/from the first and 128th agents is large is that these agents had to ask other agents to perform the RTT measurement when they accepted measurement requests. For example, these agents asked the 10th agent to obtain the RTT between the 10th agent and the other agents. Moreover, after they obtained the RTT information, the first and the 128th agents had to store the measured RTT information in the shared database, as described in Section 5.3.

Secondly, we look into the exact number of exchanged messages and its tendency. Figure 5.4 shows the total number of exchanged messages during the 4000 requests in the respective cases of the C values. We can find that the number of messages exchanged between two core agents increases proportionally as the logarithm of the number of core agents grows. This number is zero where $C = 1$ because there is only one core agent and it does not need to issue a control message to another core agent. Meanwhile, the number of messages exchanged between a core agent and a stub agent changes slightly, though it becomes zero in the case of no stub agent ($C = 128$). The number of messages exchanged between two stub agents decreases as the number of stub agents decreases. The total number of messages tends to

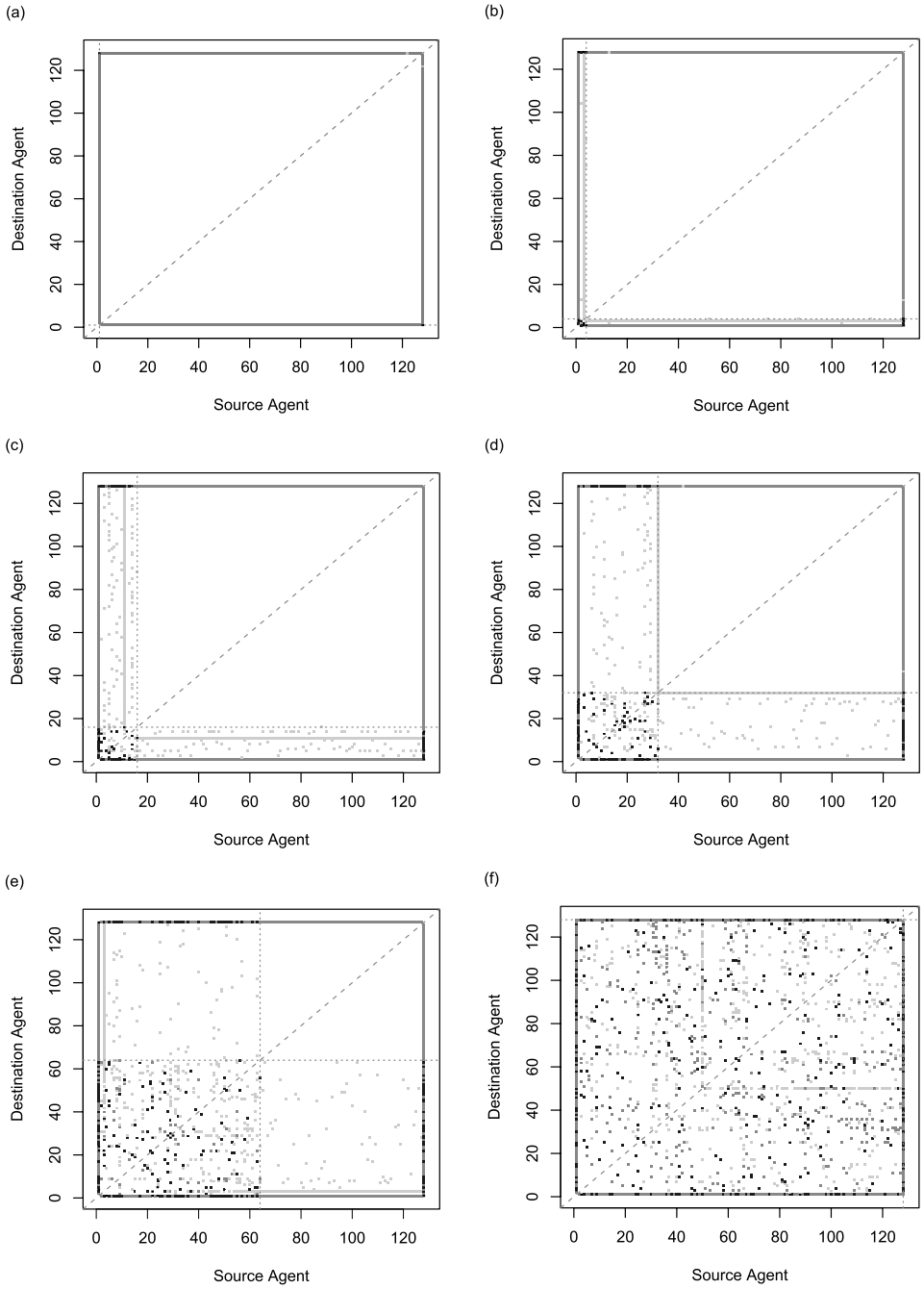

Fig. 5.3. Distribution of exchanged messages among 128 N-TAP agents where (a) $C = 1$, (b) $C = 4$, (c) $C = 16$, (d) $C = 32$, (e) $C = 64$, and (f) $C = 128$.

be larger as the number of core agents increases.

From these tendencies, let us see the number of messages that an agent of each role has to process as a metric of loads. It appears that the number of messages that one core agent has to process is reduced when the proportion of core agents to the total number of agents is large, because the growth order of the summation of the core-core and core-stub messages is lower than that of the number of core agents. On the other hand, when this proportion of core agents is large, the number of messages that one stub agent has to process increases but is still smaller than the number of messages that one core agent has to process.

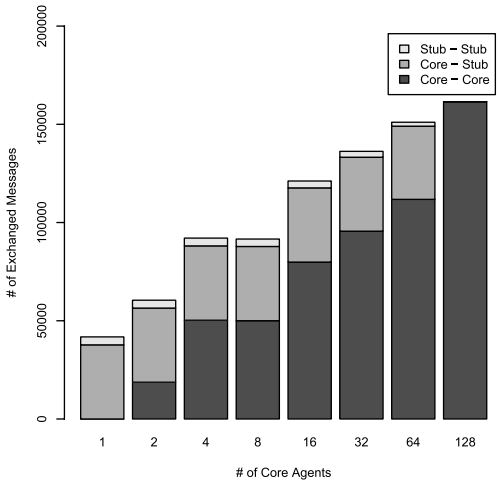

Fig. 5.4. Number of exchanged messages.

These facts indicate that the scheme of core and stub agents works just as we had intended, that is, that the loads should be distributed among the core agents, and the stub agents should have less burden. The maintainer of the measurement network can easily adjust the load distribution with the proportion of core agents as he or she intends.

5.4.2 Responsiveness

Next we compare the responsiveness to a measurement request in the cases of a request to a core agent and to a stub agent. Responsiveness is an important factor as an application-oriented measurement service, because it has an effect on optimization procedures performed by emerging applications that need network characteristics. In Figure 5.5, the boxplots that represent the distribution of the turn-around time for a measurement request are depicted. The left graph represents the turn-around time in the case that a client program issued requests to a core agent, and the right graph represents the turn-around time in the case of issuing requests to a stub agent. In both graphs, the horizontal axis denotes the number of core agents and the vertical axis denotes the turn-around time for one request. In the case of sending a request to a stub agent, the boxplot where $C = 128$ is not given because we have no stub agent in the measurement network.

We can find that, in both cases, the turn-around time where we adopted the centralized model is shorter than the turn-around time with other models. The difference between the centralized model and other models is that the agent that accepts a request must perform the procedures for finding a responsible agent, retrieving the information on agents in the N-TAP network from the shared database, and requesting other core agents to store the collected data as an entry in the shared database. Inspecting the log files of the agents that accepted measurement requests from a client program, we calculated the mean of required time for each procedure, and the result is shown in Table 5.1. From this table, we see that, between the centralized model and the other models, a considerable difference of the time required

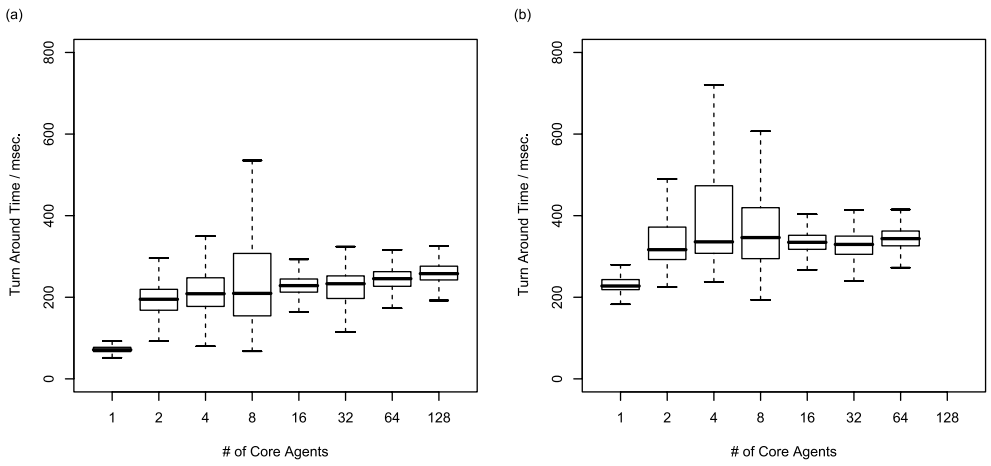

Fig. 5.5. Turn-around time for a measurement request to (a) a core agent/(b) a stub agent.

Table 5.1. Required time for the procedures (in milliseconds).

The number of core agents ($= C$)	2	4	8	16	32	64	128
Find a responsible agent (core)	1.0	2.5	6.2	4.8	6.4	7.5	7.8
Retrieve from the shared DB (core)	5.5	5.3	7.6	3.5	3.4	3.0	3.0
Store in the shared DB (core)	52.7	53.7	49.4	58.2	55.5	60.5	62.6
Find a responsible agent (stub)	2.1	7.1	9.6	8.3	9.8	11.2	—
Retrieve from the shared DB (stub)	10.0	9.1	9.7	5.5	5.2	4.6	—
Store in the shared DB (stub)	56.5	58.5	55.0	60.3	56.4	61.5	—

for a measurement request is dominated by the time required for these procedures. The time required for finding a responsible agent is expected to increase linearly depending on the logarithm of the number of core agents. This is because, given the nature of Chord, the number of times a control message to find a responsible agent is forwarded among the core agents is proportional to the logarithm of the number of core agents. In Table 5.1, the required time for finding a responsible agent seems to follow this expectation. On the other hand, the required time for the database-related procedures would not significantly change while the size of local databases in respective core agents is small, and we can confirm such a tendency from the table. We also note that the distribution of the core agents' IDs also has an effect on the topology of the measurement overlay network, which results in the fluctuation of the required time, as in the above table. In this experiment, the IDs were randomly assigned; therefore we suppose that the required time for these procedures is almost the same among the agents.

The required time for finding a responsible agent in the case of sending a request to a stub agent is longer in the order of a few milliseconds than in the case of sending a request to a core agent. This can be explained by considering that a stub agent first needs to send a control message to a core agent, while a core agent can send a message directly to the next hop's agent in its own routing table. We can suppose that this additional procedure for a stub agent increases the required time in the case of sending a request to a stub agent.

According to the discussion in this section, a measurement network with the centralized model is superior to one with the hybrid or the pure peer-to-peer model in its responsiveness to a measurement request. In this experiment, communication delay between the agents is short enough to be ignored, however, the communication delay will range approximately from tens to thousands of milliseconds when the measurement network is deployed in a wide-area network. This will have significant influence on the measurement network with the hybrid or the pure peer-to-peer model because a larger number of control messages must be exchanged through networks in these measurement networks. However, the centralized measurement network always has to struggle with load concentration at a core agent. These factors should be considered in constructing a measurement network.

5.5 Discussion

So far we have described the trade-offs among measurement networks with three different models based on the agents' behavior in respective networks. The centralized measurement network can get the best responsiveness in exchange for the heavy loads, which may bring a decrease in responsiveness. In the hybrid measurement network, we can select multiple core agents according to our purposes, and the processing loads can be distributed among the core agents. The load on one core agent will be the minimum on an average in the case of the pure peer-to-peer measurement network. However, in the hybrid and pure peer-to-peer measurement networks, the responsiveness will go down depending on the size of the control planes of these networks.

The ease of adjusting the formation of a measurement network will be important in the actual deployment of a measurement service. In this report, we first proposed the scheme of core and stub agents in a measurement network. With this scheme, we can easily shift the measurement network among the centralized network, the hybrid network and the pure peer-to-peer network by adjusting the proportion of core and stub agents. In the case that we can control a measurement network (e.g., when we monitor network facilities with such measurement systems), administrators should design the measurement network to meet their requirements. They will benefit from the ease of adjustment to the measurement network. In the case that we cannot know beforehand what types of agents will join a measurement network, we cannot create a clear plan for constructing the network. One of the cases is that the agents run on the same nodes as the applications (an overlay network application, etc.), whose nodes will arbitrarily join and leave. Even in such cases, role-based adjustment will work with the application nodes. For example, in order to improve the responsiveness to a measurement request, we would choose agents that are connected with a high-speed link and have high performance as core agents. Other metrics, like the continuous running time of nodes, will also be helpful in constructing the desired measurement network.

Focusing on the application-oriented measurement service, quick responsiveness to a measurement request is indispensable in a measurement system. To improve the responsiveness in a hybrid or a pure peer-to-peer measurement network, some possible refinements of a measurement system can be pointed out. One is to let an agent cache the results of finding a responsible agent so as to decrease the number of exchanged control messages. From the results in Section 5.4.2, in a large-scale core network, we can expect that the required time for finding a responsible agent will become dominant in the turn-around time for a measurement request. Caching the results of this procedure will improve the responsiveness, but the agents will need to handle the inconsistency between the cache and the actual topology of a measurement network, and we will pay a waiting time penalty when such inconsistency occurs. Moreover, as described before, choosing core agents based on the capability of agents will also be effective. In the case of choosing core agents dynamically, we will also have to handle the migration of key-value pairs in a distributed hash table (DHT), which is expected to be a considerable burden.

5.6 Prior Work

Some application-oriented measurement systems have been proposed. The S^3[187]'s network is similar to our hybrid measurement network in terms of having multiple roles for the entities in its network. On the other hand, considering that these entities are connected in a peer-to-peer manner, the S^3 network can be regarded as a pure peer-to-peer measurement network. iPlane[100] forms a centralized measurement network and provides a variety of network characteristics including an IP level topology, packet loss rate and available bandwidth. pMeasure[97] leverages the technique of Pastry[136] to form its own pure peer-to-peer measurement network and manage monitoring nodes in this network. In application-oriented measurement, the responsiveness to a measurement request is emphasized. To improve the responsiveness in these systems, an inference algorithm for network characteristics is sometimes utilized instead of performing actual measurement procedures. For example, iPlane estimates the RTT between two nodes based on an AS path. Alternatively, research efforts have produced effective measurement methodologies in large-scale networks called "cooperative measurement." As one example of the cooperative measurement methodologies, Vivaldi[29] lets us calculate the RTT between two nodes from their locations and distance in Euclidean space. Some

researchers have adopted an approach of optimizing overlay networks for a specific measurement purpose. For example, MIND[93] focuses on the indexing and query processing in order to make its overlay network suitable for the distributed monitoring of anomalous traffic.

Other measurement infrastructures, e.g., DIMES[36] and NETI@home[148], whose main purpose is the statistical analysis of network characteristics, basically construct centralized measurement networks. They aggregate the collected data to a central server for performing their own analysis. These infrastructures do not need to consider responsiveness as strictly as application-oriented measurement services do. Hence the simple formation of a centralized measurement network seems to be suitable for analyzing the collected data.

In a hybrid peer-to-peer network, each overlay node is assigned one or more node roles and is managed in a hierarchical structure as described already in this report. Kazaa[83], a peer-to-peer file sharing application, utilizes this scheme to connect between its unstructured peer-to-peer network and ordinary nodes. Though the details of its protocol and structure are not officially unveiled, some measurement-based work[92, 94] has already been done. The extension to N-TAP that we have added in this report is unique in applying this scheme to a structured measurement overlay network in which measurement procedures different from the ones of ordinary file sharing applications are performed.

5.7 Conclusions

Analysis of the behavior and characteristics of measurement networks was an unexplored field. In this report, we proposed a methodology for constructing a measurement network, which can easily change its network formation, alternating between centralized, hybrid and pure peer-to-peer models. By adopting this scheme and modifying an existing measurement agent, we investigated the operational flow in each of the measurement networks. As a result, we were able to confirm that exchanging control messages through networks has an appreciable effect on the turnaround time for a measurement request in the hybrid and pure peer-to-peer measurement networks. At the same time, the processing loads were successfully distributed among core agents in these networks. The consideration of such tradeoffs is important in constructing a desired measurement network.

More measurement networks of a decentralized type will appear, and their importance will grow in the future, as large-scale network services and emerging applications are developed in the Internet. In further research and development of the N-TAP project, we aim to construct a practical measurement network that can provide network characteristics indispensable for these applications.

第 6 章　Gulliver Project: Building Distributed Active Measurement Appliances

6.1 Overview

This report describes the design and implementation of a plug-and-measure appliance and its remote management framework. We use a small disk-less box as an active measurement appliance, and the remote management framework is built into the firmware of the appliance. A management server, by both a GUI or scripts, remote appliances can be instructed to preform measurements, current appliance status, and apply firmware updates. As a case study of this system, we show long-term DNS measurements obtained from the appliances deployed around the world.

The purpose of this project is to build a measurement appliance and framework which can easly be deployed in a developing world instructor. The core goals of the framework are (1) distributing probe boxes all over the world, (2) managing the boxes easily, and (3) getting measured

results easily. We call our framework the "Gulliver Framework". In this section the overview of the appliance and the framework is described.

In order to build a distributed active measurement framework, there are several requirements to achieve. The requirements for the framework are (1) easy installation and robustness of probe boxes, (2) security of the probe box and communication, (3) low costs for management, and (4) applicability of various active measurements.

In consideration of the above requirements, we have selected to use the SEIL appliance, rather then a PC as a probe box. SEIL is a commercial product produced by the Internet Initiative Japan (IIJ) Inc, developed for use as a SOHO router. The advantages of using this platform for us are: (1) the hardware has been proven to work for a 24x7 operation, (2) a remote management system for a SOHO router use is already developed, and (3) we can easily import future bug or security fixes from the commercial version of the product. We have customized the SEIL's firmware in order to support our measurement activities. Should the need arise we could easily port the framework to some other hardware.

Most of the existing measurement efforts use PCs as measurement boxes. In constast to a PC an applications: is cheaper to purchase, cheaper to send, take up less space at the hosting institution, and is has a lower power consumption. In addition SEIL's disk-less storage means improved reliablity in the face of unreliable power supplies, common some developing regions. makes it a more superior choice given our target hosting sights.

As for security, the communication between the probe boxes and the servers should be encrypted. The probe box should accept only connections from our specified servers. Even in the unlikely case that one one of the box is physically stolen, the over all security of the framework should not be threatened.

The probe boxes should be managed and operated in an integrated fashion by the management server. One of the goals of the project is to deploy more than 100 probe boxes world wide. If a vulnerability is found in the firmware or if we want to build new measurement tools into the firmware, it should be upgraded in all the probe boxes without pains and troubles. In addition to the firmware upgrading, daily monitoring of operations should be provided.

The firmware of our appliances is a derivative of NetBSD, so measurement tools which run on NetBSD can be easily be deployed in future firmware updates. This provides an easy mechanism for development and debugging for measurement tools. The firmware also includes a remote management system. All the appliances can be controlled and monitored from the management server through the remote management system, called SEIL Management Framework (SMF).

6.2 Framework Design

The framework consists of probe appliances, management servers and measurement collection servers. We have designed the framework based on the client-server model. The set of servers manage, control and monitor all probe appliances. The appliance does not start any measurement action until it is instructed to do so by the management server.

6.2.1 Architecture Goals

As described before, the appliance has the remote management system built into the firmware. The Gulliver Framework has the following architectural goals.

(a) plug-and-measure of appliances; even if the network environment which the appliance is located in is changed and an administrator can not login the appliance remotely, the configuration of the appliance can be changed at the management server and just cycling power of the appliance, the appliance will boot with new applicable configuration. We want to distribute appliances without circumstance in developing countries.

(b) low operational costs and effective measurements; when the number of activated appliances

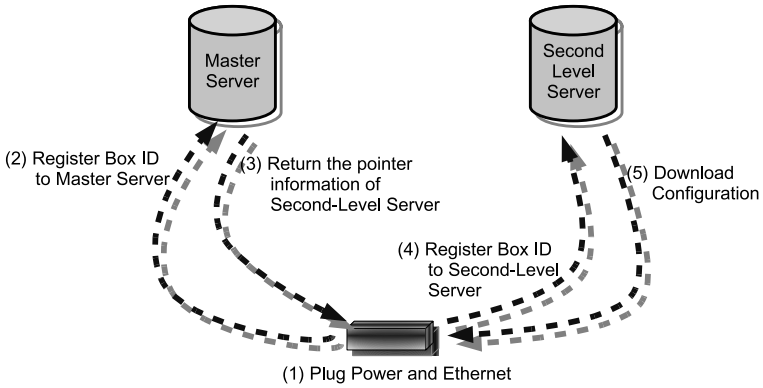

Fig. 6.1. The flow diagram of bootstrap

is increased, the operational costs is not increased proportionately. An administrator can control a number of appliances as controlling just an appliance. However, the measured results are increased proportionately and an administrator can collect the results easily.

(c) secure and safe against snooping and cracking; the appliance talks to the servers with authenticated and encrypted datagrams, and the measured results also should be retrieved securely.

6.2.2 Appliance bootstrap

When the appliance is connected to a UTP cable and power source, it obtains an IP address using DHCP, and then the appliance bootstraps as shown in figure 6.1. The appliance computes its own unique ID from its MAC address. The appliance knows only the IP address of a master server and the server's SSL certificate.

The master server's only job is to authenticate the appliance from its ID and tell the appliance about a second-level server. At bootstrap, the appliance contacts the master server using XML-RPC over SSL, verifies the server using the built-in certificate, and passes its unique ID to the server. The master server checks the appliance's ID, and returns the IP address and the SSL certificate of a second-level server.

The appliance contacts the second-level server using XML-RPC over SSL. The reasoning behind this server hierarchy is to allow a set of second-level servers for future scalability as well as to have completely different services for the appliances. The second-level server returns the appropriate configuration for the appliance, if the authentication is succeeded. The configuration includes the SSL certificates of the servers, and the SSH public key of a measurement collection server. The appliance keeps this configuration on memory, and reboots itself using this configuration.

6.2.3 Appliance management

When controlling the appliances after a reboot, firmware upgrading or requesting status reports, an administrator does not need to send the requests directly to every appliance. Rather, all he or she needs to do is send the requests to the second-level server as shown in figure 6.2.

For example, if an administrator wants to reboot Box B and C, she can send a reboot request to the second-level server with the IDs of Box B and C, then the second-level server sends reboot requests to Box B and C.

When an appliance talks to a server, it uses XML-RPC over SSL. On the other hand, when a server talks to an appliance, it sends a command over SSH. In addition to these communications, appliances can periodically send heartbeats to a server; a heartbeat message includes the status of the interfaces, CPU and memory so that administrators can monitor the status of the deployed appliances.

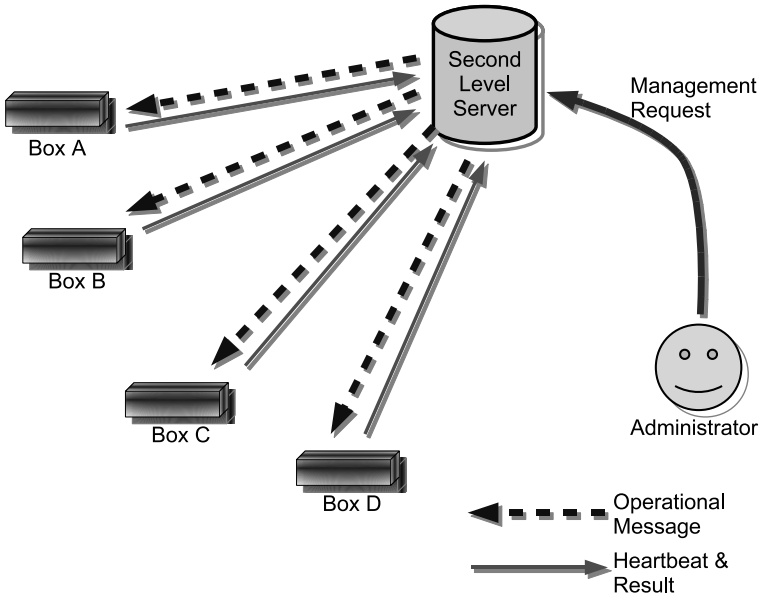

Fig. 6.2. The flow diagram of management

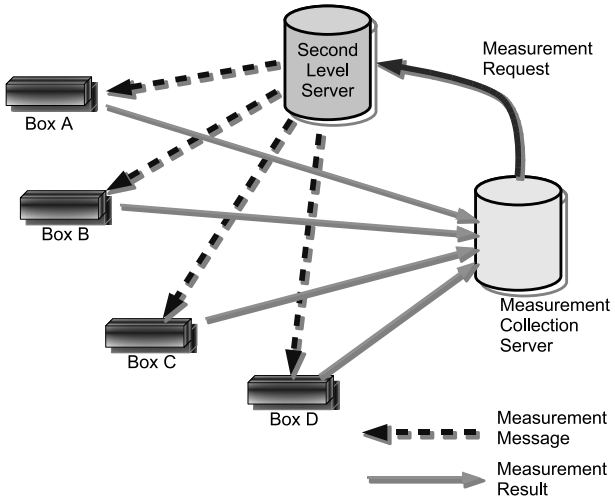

Fig. 6.3. The flow diagram of measurement

6.2.4 Appliance measurement

Administrators can send start and stop requests for measurement programs to the appliances in a similar manner as the appliance management shown in figure 6.3. When an appliance receives a start request from second-level server, it starts the measurement. When the measurement is completed, the appliance sends the results to the measurement collection server specified by the administrator.

6.3 Framework implementation

There are a number of challenges in building the appliances and framework. In this section, we have described details of our implementation, especially about our modifications to the original SMF and challenges encountered.

The first challenege we faced was to reduce firmware memory usage. The appliance has only 64 MB memory. In order to save memory we deleted unused functions for active measurements from the original firmware.

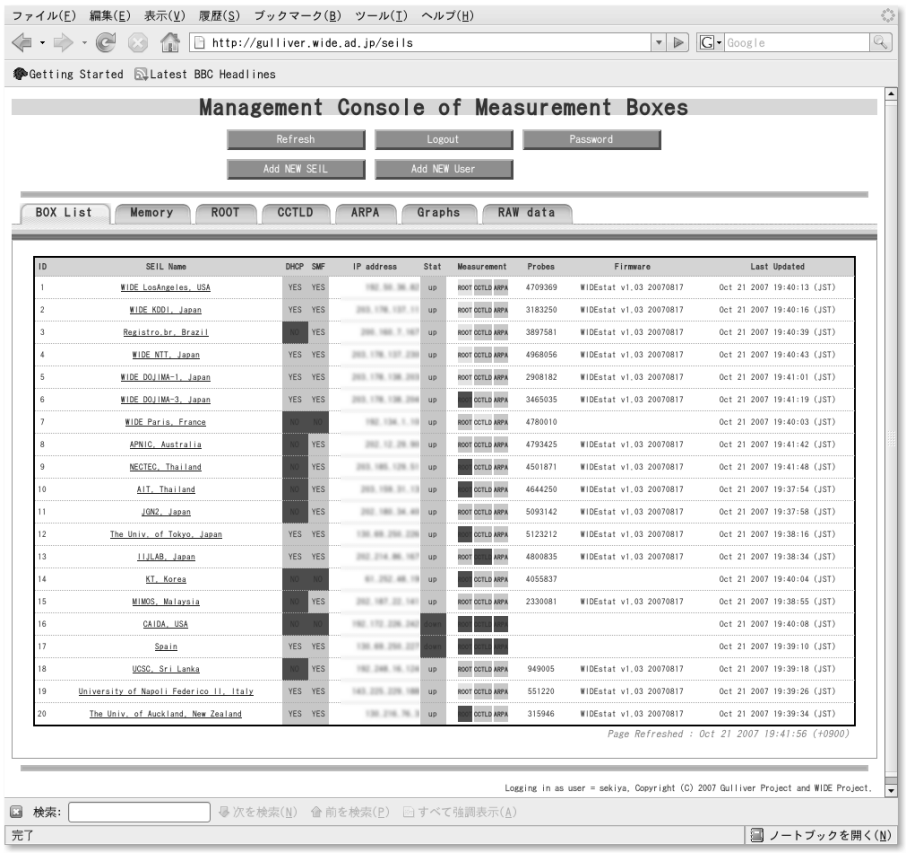

Fig. 6.4. Web GUI for the managements

The second challenge was building a wrapper software to run the existing measurement tools without modifications. The wrapper was integrated into the management system firmware.

The third challenge was handling static configuration of IPv4 addresses. The original remote management system, SMF only supports DHCP environments. This means that if a DHCP server is not available, the appliance can not get IPv4 address or communicate with the master server. Depending on the policy of the instillation site this can be problematic, because there exist situations were only a static IPv4 address is available. In order to adapt to a static IPv4 address, we modified the firmware so that the IPv4 address and default gateways could be pre-configured. If the IPv4 address is pre-configured, the appliance communicate to the master server using the address, then it boots in a manner similar to a DHCP environment.

The fourth challenge was the to build a more user-friendly administrative interface. Thus reducing the cost to manage and monitor the appliances. It is not user-friendly to send messages using XML-RPC with SSL for users who would like to just run active measurements on this framework, so it is very important to deploy and generalize the framework for various active measurements. The interface is shown in figure 6.4.

In the figure there is a list of the registered appliances. By clicking the name of appliance on the list, the detail of the appliance is shown as in figure 6.5. By clicking “reboot” button in the figure, the reboot request will be sent to the appliance through the second-level server and the appliance will be rebooted.

Figure 6.6 shows the memory usages of each appliance. The status is monitored by heartbeats which every appliance send to the second-level server. Figure 6.7 also shows the status of the

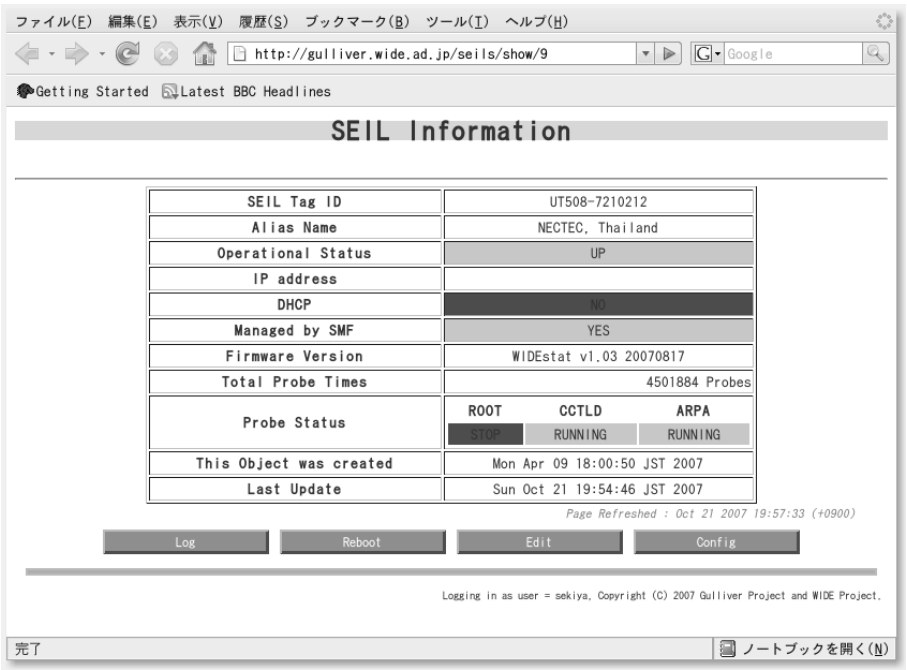

Fig. 6.5. Details of the appliance

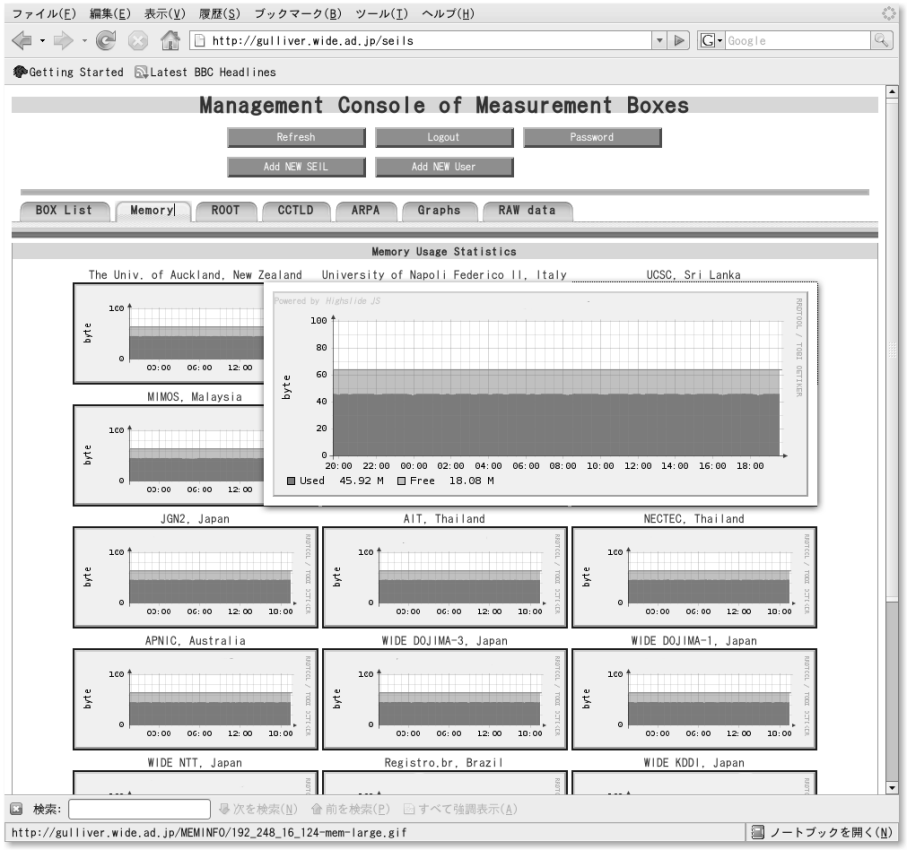

Fig. 6.6. Monitoring memory usages

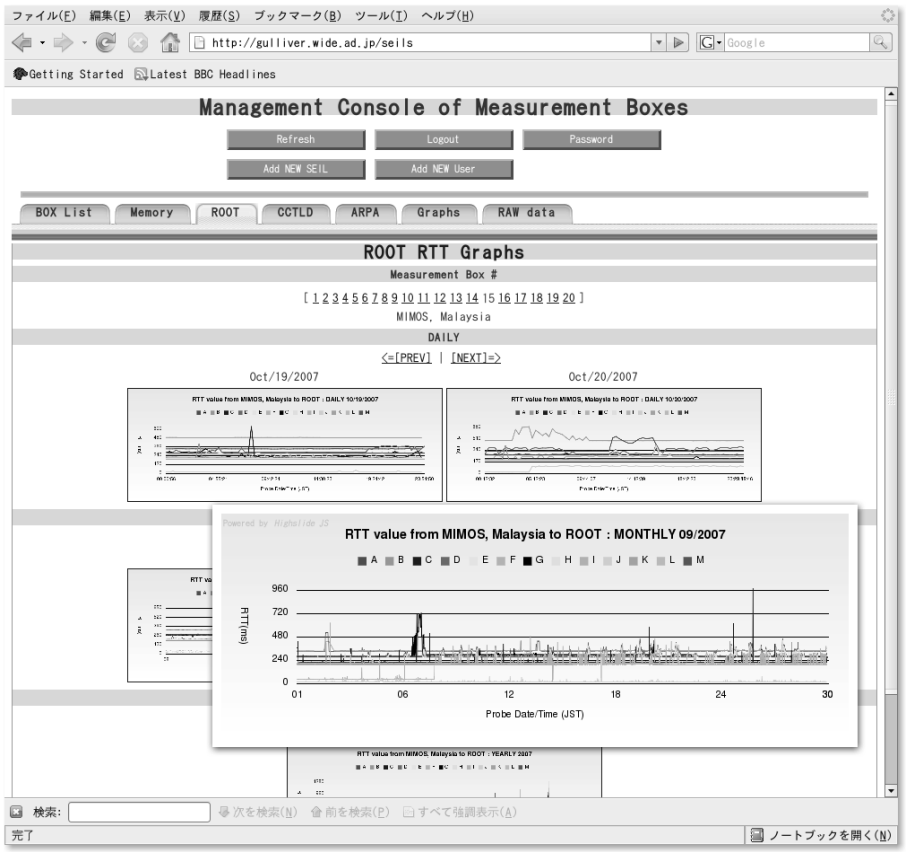

Fig. 6.7. Monitoring measurement status

Table 6.1. Location of measurement boxes

Box ID	Organization	City	Country
1	WIDE	Los Angenes	U.S.A.
2	WIDE	Tokyo	Japan
3	Registro.br	Sao Paulo	Brazil
4	WIDE	Tokyo	Japan
5	WIDE	Osaka	Japan
6	WIDE	Osaka	Japan
7	WIDE	Paris	France
8	APNIC	Brisbane	Australia
9	NECTEC	Bangkok	Thailand
10	AIT	Pathumthani	Thailand
11	JGN2	Tokyo	Japan
12	The University of Tokyo	Tokyo	Japan
13	IIJ Inc.	Tokyo	Japan
14	Korea Telecom	Seoul	Korea
15	MIMOS	Kuala Lumpur	Malaysia
16	CAIDA	San Diego	U.S.A.
17	University of Colombo School of Computing	Colombo	Sri Lanka
18	University of Napoli Federico II	Napoli	Italy
19	The University of Auckland	Auckland	New Zealand

measured results collected by measurement collection server. As shown in the figures, administrators can manage and monitor the appliances using web GUI.

6.4 Measurements

As of October 2007, 18 appliances are distributed and activated. Table 6.1 summarizes the locations. Only one box is located in United States, while the majority are located in Asian countries reflecting the project aims to deploy the appliances in developing regions.

Two measurement tools are implemented in the appliance at this time, dnsprobe and scamper. As a case study of the Gulliver framework, we run two measurements and validate and prove advantages of the Gulliver Framework. The results are described in this section.

6.5 Project Status

A further directions of this study are (1) distributing more appliances all over the world at least 100 boxes, especially developing regions, (2) implementing module mechanism in the firmware in order to load and unload measurement tools dynamically, and (3) enabling managements even if the appliance is located behind NAT.

The status of gulliver project is available on `http://gulliver.wide.ad.jp/`.

第 7 章　まとめ

インターネットの研究において、計測はますます重要視されてきていて、国際協調の機会も増している。そのような状況のなかで、WIDE プロジェクトの計測活動は、グローバルな視点を持った継続的な計測活動として国際的にも認知されてきている。2008 年度は、国際協調を実りある研究に結びつける事を目標に置いている。