



THE HONG KONG
POLYTECHNIC UNIVERSITY
香港理工大學

FlashyPath: A Flash-based visualization
tool for Internet path measurement

By

Chun Ping Lim

(05765925G)

A Dissertation submitted in fulfillment
of the requirement for the degree of
Master of Science in Software Technologies
Hong Kong Polytechnic University 2008

STATEMENT OF AUTHORSHIP

Except where reference is made in the text of this dissertation, this dissertation contains no material published elsewhere or extracted in whole or in part from a dissertation presented by me for another degree or diploma.

No other person's work has been used without due acknowledgement in the main text of the dissertation.

This dissertation has not been submitted for the award of any other degree or diploma in any other tertiary institution.

Name:

Dated:

ABSTRACT

Abstract of dissertation entitled:

FlashyPath: A Flash-based visualization tool for Internet path measurement

Submitted by Chun Ping Lim

For the degree of MSc in Software Engineering

at The Hong Kong Polytechnic University in July 2008

Visualizing end-to-end path measurement data is important for monitoring network performance, diagnosing network problems, identifying trends and faults. However, the current visualization systems, such as VisoNETUI and Smokeping, suffer from a number of shortcomings. One of the problems is to only provide static presentation on measurement results. Moreover, most systems are short of interactive model that restricts to explore more performance metrics of Internet path. In essence of archive functions, most are reluctant because it does involve too complex technical designs. In this report, we have proposed and developed new monitoring approaches to overcome these shortcomings.

We build new network monitoring tool called FlashyPath for the issues addressed which visualizes interactive measurement results on multiple Internet paths. For the sake of visualizing continuous monitoring results, the tool is required to read the measurement results for every five minutes and compute averaged statistics on

each client request. Having FlashyPath integrated interactive model can provide ISP and institutional operators with

In the report, FlashyPath provides Internet path measurement on Round-trip time (RTT), Time- To-Live (TTL), and packet and path loss. A visual mechanism for network topology would also be useful to better understand and debug network problems with utilizing measurement result in multiple paths. The report will mainly illustrate the worked approach on user interaction and data integrity.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Dr. Rocky Chang, for his guidance, encouragement, and support. Being advised and mentored by Rocky has been a very rewarding learning experience for me.

I am deeply grateful that Edmond Chan provides comments and great efforts with a huge amount of sample Internet measurement data for my dissertation design.

Finally, I would like to dedicate this work to my parents, wife, and son, who have always support me.

TABLE OF CONTENTS

STATEMENT OF AUTHORSHIP.....	I
ABSTRACT	II
ACKNOWLEDGEMENTS	IV
LIST OF FIGURES.....	VII
LIST OF TABLES.....	IX
1. INTRODUCTION.....	1
1.1. BACKGROUND.....	1
1.2. MOTIVATIONS	4
1.3. IMPLMENETATION.....	5
1.4. CONTRIBUTIONS.....	6
1.5. RELATED WORKS.....	7
1.6. OBJECTIVES	10
1.7. DISSERTATION LAYOUT	10
2. LITERATURE REVIEW	12
2.1. E2E ARCHITECTURE	12
2.2. FLASH AS INTERACTIVE PACKAGES	23
2.3. AUTONOMOUS SYSTEM TOPOLOGICAL GRAPH	26
2.4. GRAPH THOERY	33
2.5. FLEX AND FLASH.....	35
2.6. FLASH AND JAVA	38
3. DESIGN AND IMPLMENTATION	41
3.1. OVERALL ARCHITECTURES	42
3.2. DEVELOPMENT CYCLE	43
3.3. WRAPPER DESIGN	46
3.4. ALIGNMENT OF RESULTS.....	52
3.5. DATABASE DESIGN.....	54
3.6. VISUALIZATION MODULE.....	59
4. SOFTWARE DEVELOPMENT	61
4.1. HTTP SERVICE.....	64
4.2. ADOBE FLEX AND PHP	67
4.3. FLEX GRAPH	69

4.4.	CSS & SYTLE	75
4.5.	FLEX APPLICATION WITH RRDTOOL	76
4.6.	RRDTOOL (ROUND-ROBIN DATABASE).....	79
5.	LIMITATIONS.....	82
5.1.	UNUSED RRD SPACE	83
5.2.	TIME LAG.....	85
5.3.	PHP INTERFACE BETWEEN FLASH AND SQL DATABASE.....	86
5.4.	SQL INJECTION	87
5.5.	TCP SOCKET VULNERABLE.....	88
6.	CONCLUSION & FUTURE WORK	90
7.	APPENDIX	93
7.1.	INSTALL APACHE 1.2.2 WITH PHP 5.0.....	93
7.2.	INSTALL MYSQL AND RRDTOOL DATABASES	95
7.3.	INSTALL FLASHYPATH BINARY DISTRIBUTION	96
7.4.	INSTALL FLASH PLAYER 9.0 OR ABOVE.....	97
7.5.	SOURCES CODE.....	99
8.	REFERENCES.....	107

LIST OF FIGURES

Figure 1-1 Perfuse visualization framework	2
Figure 1-2 VisoNETUI as visualization for multiple Internet path measurement ...	8
Figure 1-3 Smokeping for latency visualization	9
Figure 1-4 Cytoscape for network topology	10
Figure 2-1 Rich Internet Application for interaction of measurement results	14
Figure 2-2 Typical three-tier Architecture	16
Figure 2-3 implementation of the MVC pattern	17
Figure 2-4 Visual PerfSONAR on SOA architectures	18
Figure 2-5 Skitter visualizes network connectivity without interaction model	23
Figure 2-6 Animated Atlas - http://www.animatedatlas.com/movie.html	25
Figure 2-7 Comparison of the two AS mapping approaches	27
Figure 2-8 Sample measurement results for Internet path	30
Figure 2-9 The development of RIA application through Flex modules	36
Figure 2-10 Platform comparison (Source: UWEBC)	39
Figure 3-1 Software requirement of FlashyPath	43
Figure 3-2 Development cycle of FlashyPath designed on each Flex application	44
Figure 3-3 Design of wrapper of measurement result collection.....	51
Figure 3-4 Mechanism of alignment of round-robin and MySQL databases	53
Figure 3-5 Database schema is developed for FlashyPath on AS paths and measurement results	55
Figure 3-6 The database of FlashyPath	55
Figure 3-7 RRD control wrapper is to create and update RRD file using RRDTool for each analyze item	57
Figure 3-8 Diagram showing FlashyPath executing topology graph with measurement result chart.....	60
Figure 4-1 Adobe Flex Builder 2 built on Eclipse version 2.0.1.....	61
Figure 4-2 Create Flex application in XML or web service from PHP	62
Figure 4-3 Flex Development Workplace in designer window.....	63
Figure 4-4 Source window can allow adjust the parameter of each Flash component	64
Figure 4-5 login mxml invoke HTTP Service for user authentication.....	65
Figure 4-6 Main FlashyPath application with authentication function.....	67
Figure 4-7 PHP coding for user authentication functions	68
Figure 4-8 The Graph XML result object is data source of measurement points for Visual Graph library	70
Figure 4-9 The AS graph using flexvisgraphlib package.....	71
Figure 4-10 The measurement points for which monthly and daily measurement	

results are visualized	72
Figure 4-11 Create empty Flex Chart for measurement result.....	73
Figure 4-12 Monthly interactive timeseries panel for multiple measurement points	75
Figure 4-13 RRDTool command line.....	77
Figure 4-14 Reload the browser and have a nice XML-version of dataset.....	78
Figure 4-15 Using RRDTool export function, daily measurement result from round-robin database is created on the fly.....	79
Figure 5-1 List of available performance metrics	83
Figure 5-2 Physical size of each Internet path on analyze item	84
Figure 5-3 Percentage of impression of vulnerability (Source: adambarth)	89
Figure 7-1 Binary distribution can be downloaded from Apache project web site	94
Figure 7-2 PHP official web site	94
Figure 7-3 Install Apache 1.2.2 with PHP packages	95
Figure 7-4 MySQL official web page	96
Figure 7-5 RRDTool official web page.....	96
Figure 7-6 rrdtool support for PHP	96
Figure 7-7 Flash component in HTML tag	97
Figure 7-8 Flash Player 9.0 for executing FlashyPath	98

LIST OF TABLES

Table 2-1 Traceroute from 1.1 to the University of Hong Kong.....	29
Table 2-2 Sample traceroute measurement result tuple	29
Table 3-1 Open-source package for FlashyPath development.....	42
Table 3-2 FlashyPath MXML in which measurement tasks are performed.....	45
Table 3-3 Sample meta data description file for each sub experiment.....	46
Table 3-4 Traceroute showing timeout result.....	48
Table 3-5 IP-to-AS mapping function returns XML result storing AS information on prefix, ip address, IPs/Prefix, AS name, AS description, Country and BGP Prefix of IP address.	49
Table 3-6 AS file describes traceroute and AS relationship between two measurement points.....	49
Table 3-7 The XML hierarchy of RRDTool xport function	54
Table 3-8 RRD create function for source and destination nodes for June 2008...	58
Table 4-1 the sample procedures of Flash to be implemented	62
Table 4-2 HTTPService Tag for Flash object communicate data source through PHP gateway	65
Table 4-3 HTTP Service tag can specify any value of parameters for mx:Request tag	66
Table 4-4 Named Object is created for send method of HTTPService class	66
Table 4-5 the function for return object from PHP	67
Table 4-6 the return object in form of XML containing user object	68
Table 4-7 Flex Visual Graph library for creating hierarchical graph for measurement points.....	69
Table 4-8 the procedures of building Flex Graph for measurement results	72
Table 4-9 DateTimeAxis and VerticalAxis.....	74
Table 4-10 Sample CSS and Style on Flex application.....	76
Table 4-11 RRDTool xport command for daily measurement results.....	77
Table 5-1 The file size of RRD for performance metrics.....	84
Table 5-3 absolute URL locator if FlashyPath serves as standalone tool	86
Table 5-4 Drop table SQL command if query value of FlashyPath request is changed	87
Table 5-5 The SQL query will be amended that table is permanently removed. ...	88
Table 7-1 Sample MySQL functionalities.....	95
Table 7-2 SELECT statement for measurement results on MySQL database.....	96
Table 7-3 RRDTool fetch function.....	96

1.INTRODUCTION

1.1. BACKGROUND

Visualizing end-to-end path measurement data is important for monitoring network performance, diagnosing network problems, identifying trends and faults. Internet Service Provider (ISP) and institutional operators exploit monitoring traceroute measurement results for Internet paths with interactive channels that help diagnose Internet paths in a timely manner. More importantly, integration of real-time data with advanced presentation skill is expected for timely responses that the operators are quickly informed the faults and errors of Internet path performance [3].

In this report, we aim at incorporating interactive measurement tool on visualizing continuous measurement results for ISP operators. The term interactive measurement tool [49] generally describes operators can monitor Internet paths through interactive tools to query concrete measurement results on Internet paths. Implementing interactive model would allow operators serve their customers with instantaneous monitoring results, thus enabling the service-level quality to detect

errors or provide correct diagnosis on Internet paths. Heer, Card, and Landay [60] have illustrated multiple views, semantic zooming, data and visual transformations, and application extension and customization in model-view-controller environment in future user interactive tool [61]. Perfuse, in Figure 1-1, enables multiple visualizations of a shared data set by using separate filters, and different views of a specific visualization by reusing the same filtered items.

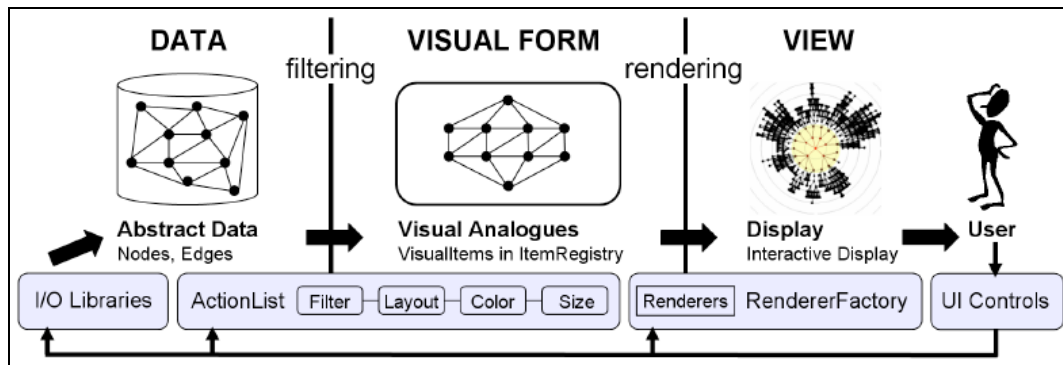


Figure 1-1 Perfuse visualization framework

However, the current visualization systems suffer from a number of shortcomings. Most of the applications, such as VisoNETUI [48], integrated with RRDTTool and server-side scripting that provides limited user interface and static graphical outputs for operators that restrict diagnosis purpose. VisoNETUI visualize traceroute measurement results, which are in advance generated, in limited time basis. Moreover, The RRDTTool provides fixed time interval and measurement unit on X- and Y-axis respectively that cannot be easily adjusted for zooming

function that impedes the flexibility and scalability. In functional, absence of archive functions on most monitoring system that measurement data cannot be compared, which disallows ISP operators distinguish current and previous performance metrics on different months and to determine what has changed on Internet paths. On the other hand, some measurement tools, such as Smokeping, tend to combine all performance metrics in a single graph that ISP operators hardly determine diagnosis results. However, the single graph is displayed for Internet characteristics derived from ping packet responses that hampers operators on further diagnosis. Moreover, some monitoring systems, especially active probing, are seldom to ensure error-free data because the measurement results is displayed in streaming mode, which hardly determines data integrity for good measurement tool.

A new monitoring tool will be therefore designed to address these problems. Compared with current monitoring systems, FlashyPath should be visualized on Internet paths measurement more interactive that operators can easily adjust their needs. If we develop the tool as web-based application, it can greatly enhance to realize interactive manner on monitoring and diagnosing purpose. The experience has also shown that interactive model integrated with server-side

programmin that does not hamper monitoring functionalities. Supposing Flash integrates Flex objects can promote interoperable features that allows others who design network monitoring interactively tasks on their own sake. In light of secured data binding service through HTTP channel, FlashyPath ensures measurement results to be quickly response to multimedia features. Having, though, much similar functionality among existing network monitoring systems, FlashyPath would focus on developing interactive approach that continuous measurement results are visualized on multiple measurement points.

1.2. MOTIVATIONS

Considering the current applications, we would develop new application, named as FlashyPath, which attempts to develop interactive monitoring tool for measurement results, in view of developer's perspective, by following:

- We can provide more interactive visualization on measurement of Internet Paths
- We develop network monitoring function with Flash technologies that builds the application quickly and further develop or deploy easily.
- We build up visual component in which other monitoring systems can be

extended.

- Allows end users or ISPs that can monitor multiple measurement paths continuously and can be compared simultaneously with integration of visual components.
- Synchronize measurement results on the database for interactively displaying very large volume of data.

1.3. IMPLMENETATION

According to motivations we have stated, the design of our measurement tool for e2e topology, the system is to be composed of several major components:

1. Data collection components extracts the measurement results in every ten minutes from source of data that is gathered, computed and stored into the databases in which visualization tool displays the result continuously.
2. Data analysis component reads the measurement results which are valid and well formatted under source directory and stores it into database on each analyze item.

3. Visualization components exhibit monitoring functions in interactive features that is capable of displaying measurement results more efficiently. The module would design the features through hierarchical AS topology and time-series chart for measurement points that enables ISP operators easily navigate.

1.4. CONTRIBUTIONS

Our system makes key contributions to the network monitoring tool:

1. The tool can systemically visualize the measurement results for Internet paths on every ten minutes
2. Show multiple Internet path measurement in same performance metrics that can be compared.
3. Provide interactive monitoring panels for ISP operators who can navigate measurement results for diagnosis purpose.
4. In order to provide measurement tool for monitoring and diagnosis purpose, we work out the guidelines of integration of multimedia packages that monitoring system can be more interactive.

1.5. RELATED WORKS

In this section we briefly present some of the most widely used state-of-the-art networking monitoring tools and provide some indications why the multi-domain performance monitoring scenario might be not well suited for most of them..

1. Cacti [34] is front-end graphing solution of network monitoring tool, which provides features including fast poller, advanced graph templating, multiple data acquisition methods, and user management features. It stores all of the necessary information to create graphs and populate them with data in a MySQL database. It contributes Cacti provides a user friendly interface to RRDTool [31] without requiring users to understand how RRDTool works
2. VisoaNETUI [45] is a visualization tool which visualizes measurement data in structured format like tables and graphs written in server-side scripting PHP languages. Like other tools, VisoNETUI generates too many RRD graphs for different metrics of Internet paths. On the other hand, measurement units and time intervals are fixed that ISP operators cannot adjust measurement items for next navigation.

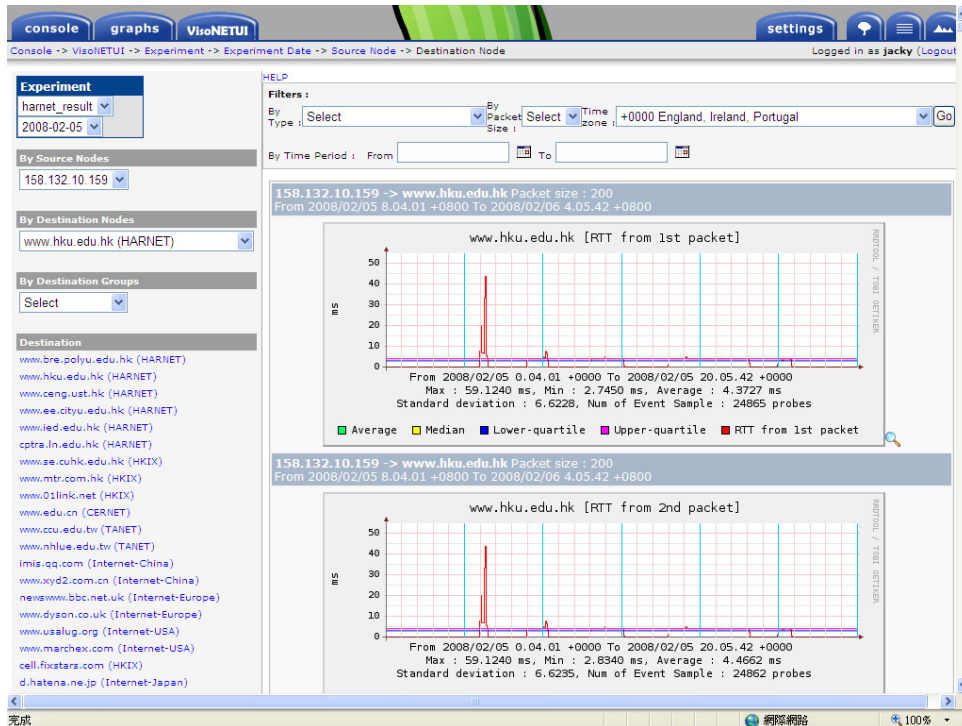


Figure 1-2 VisoNETUI as visualization for multiple Internet path measurement

3. Another popular network performance measurement tool is SmokePing [6]. It measures, stores, and displays latency, latency distribution, and packet loss. SmokePing uses RRDtool for maintaining a long term data-store, as well as for its graphing functions. It also implements a latency measurement plug-in interface for seamless extendibility and features a powerful anomaly detection and alarm reporting mechanism. When used together, CACTI and SmokePing provide a very good summary and detailed overview of network performance metrics like interface rate, round trip time, latency distribution, and packet loss. However, a higher level of integration between metrics reporting across multiple domains is desirable in many cases.

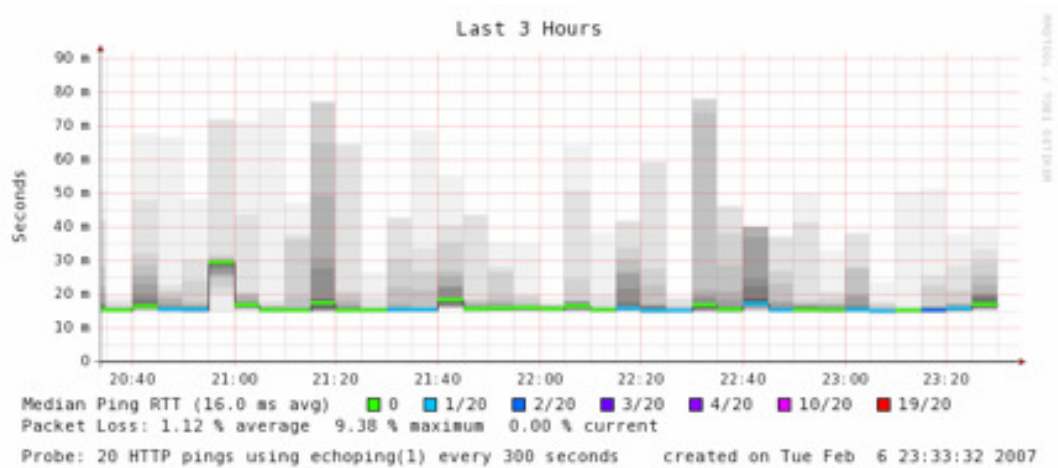


Figure 1-3 Smokeping¹ for latency visualization

4. PerfsonarUI [16] is written as Java and supports several RRDs, shows the network topology using a set of hierarchical topology, supports error handling and reporting functions on link utilization. The application employs database that stores topology and measurement data and also implements wrapper around existing RRD files and SQL databases.
5. Cytoscape [32] works as a web service client, which provides data integration and visualization. It visualizes as cyclic or force-directed graph layout. It can be used to visualize and analyze network graphs of any kind involving nodes and edges.

¹ Smokeping (<http://homepage.mac.com/duling/halfdozen/Smokeping-Howto.html>)

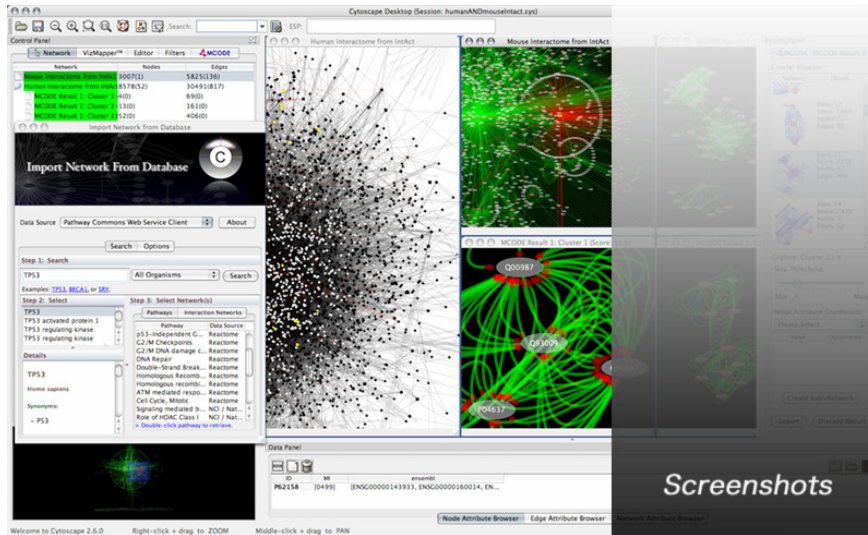


Figure 1-4 Cytoscape for network topology

1.6. OBJECTIVES

The major part of report is to propose design and development of new measurement tool using interactive model with server-side scripting that can be built in order to achieve monitoring and diagnosis purposes. We also suggest methodologies on alignment of continuous measurement results from databases that interactive advantage can be integrated. The report would address how system can gather and display interactive large volume of measurement results.

1.7. DISSERTATION LAYOUT

Besides the chapter 1 of introduction of the dissertation, Chapter 2 provides an introductory background for the readers who can understand FlashyPath design and implementation for e2e topology infrastructure that can develop interactive

model for network monitoring and diagnosis purpose. Chapter 3 describes the proposal of how e2e topology infrastructure is to be constructed using available multimedia techniques and software. Chapter 4 would outline development of FlashyPath and demonstrate network measurement tools in Flash environment in developer's perspective. The limitation and experience of development cycle are also described in Chapter 5. Finally, Chapter 6 summarizes our work and contributions. Possible future research directions for further development of the tool are also discussed.

2.LITERATURE REVIEW

This chapter mainly provides overview of e2e topology that can be worked out by interactive components with server-side scripting languages. The reader should already be familiar with basic networking terminology, including terms such as ASes, links and IP addresses. Moreover, the readers should also be professional on software development that can shorten the learning time for integration of e2e topology.

2.1. E2E ARCHITECTURE

Current network monitoring tool are built as Internet path measurement over e2e infrastructure. Challenges have been addressed that are to simplify the design of monitoring tool and exchange meta-information about testing packets [59], in which current monitoring systems do not support. If meta information from other domains is available, ISP operators enable to eliminate useless packets. The e2e infrastructure generally refers to network components, which include routers, switches, or end-to-end paths, between the endpoints could be monitored. The framework facilities precious measurement result of each component, in which

standard schemas, discovery mechanisms, and access policies for monitoring data, can be exchanged. On the other hand, historical data as data archive can be managed to establish a baseline to compare current and predict future performance in the framework. Summing up all features in the framework can visualize on the display windows for their customers. In this report, we would propose architecture in which the monitoring tool may extend from rich Internet application (RIA) framework with three-tier architecture integrated with model-view-controller (MVC) technologies.

2.1.1. RICH INTERNET APPLICATION (RIA)

Technically, monitoring tools may work with employing RIA allows ISP operators who explore the Internet paths issues interactive approach, through generation of different diagnosis graphs. One of the obvious differences between interactive application and conventional monitoring system is that any button or hyperlink of the monitoring tool is clicked which doesn't reload. Only the measurement results or other network information related to Internet paths is updated. This produces responsive and seamless interfaces more effectively enables the operators to concentrate on the diagnosis task that the results are never delayed or distracted by the mechanics of the interface itself. It indicates that visualizations

are processed more effectively when key structural elements in the display do not change from one view to the next, illustrated in Figure 2-1. This gives operators a set of continuously visible reference points, which prevent ISP operators from becoming confused or disoriented as they navigate around the application.

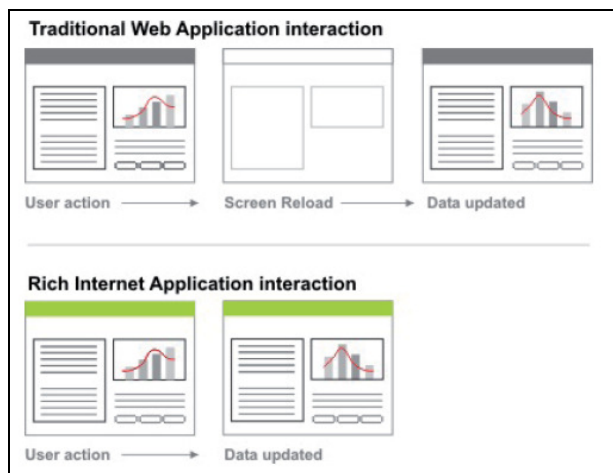


Figure 2-1 Rich Internet Application for interaction of measurement results

Rich Internet Applications² (RIA) are defined as the combination of the best user interface functionality of desktop software applications with the broad reach and low-cost deployment of Web applications and the best of interactive, multimedia communication. RIA technologies such as Adobe Flex and Asynchronous Java and XML (AJAX) are to view data without refreshing the screen or streamline interactive process. The important differentiator between an RIA and a more traditional website is that an RIA is a true application that allows you to perform a

² RIA http://www.azavar.com/solutions/rich_internet_applications.aspx

task. This task can include finding a product, customizing a service, learning new information, playing a game, or mixing information to create something new. Another difference with RIAs is the way they handle and process information. Traditional desktop applications rely exclusively on client-side processing. When a task is initiated, the local system's resources are leveraged to process the request. In contrast, web application on the server technology depends on client request. With RIAs the load is shared by both client- and server-side tasks. With an HTML website, when a user fills in data, changes options, or checks boxes and hits, the page must be submitted to the server for data validation and then the screen is reloaded with the new data incorporated.

2.1.2. THREE-TIER ARCHITECTURE

Interactive monitoring framework can be extended from traditional measurement design over e2e topology that could be exploited, which allows end-users who explore network issues from different angles of measurement results on Internet paths. The traditional framework is mainly integrated with programming skills in three-tier architecture implementing Model-view-controller (MVC) approach for monitoring framework. Three-tier is a client-server architecture in which the user interface, business process and data storage and data access are developed and

maintained as independent modules or most often on separate platforms. Basically, there are 3 layers, tier 1 (presentation tier, GUI tier), tier 2 (business objects, business logic tier) and tier 3 (data access tier). These tiers can be developed and tested separately.

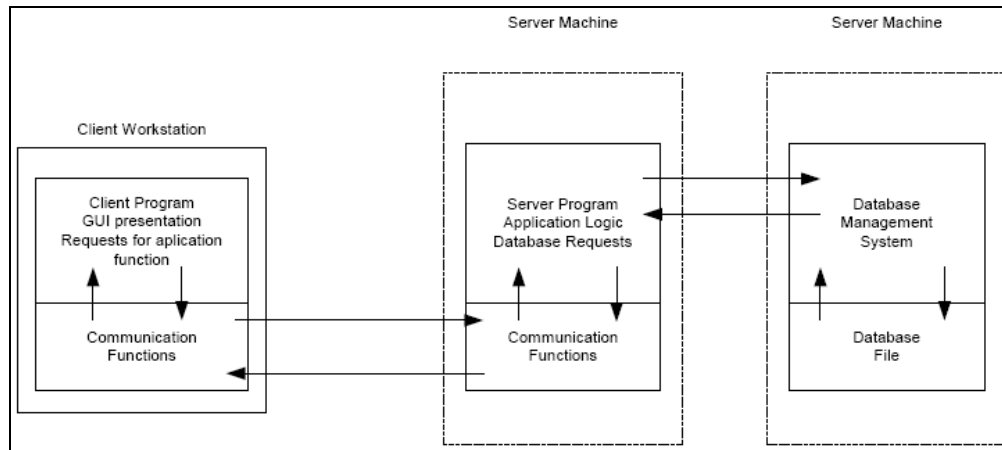


Figure 2-2 Typical three-tier Architecture

2.1.3. MODEL-VIEW-CONTROLLER (MVC)

In the MVC paradigm the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object, each specialized for its task. The view manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate. Finally, the model manages the behavior and data of the application domain, responds to requests for information

about its state (usually from the view), and responds to instructions to change state (usually from the controller).

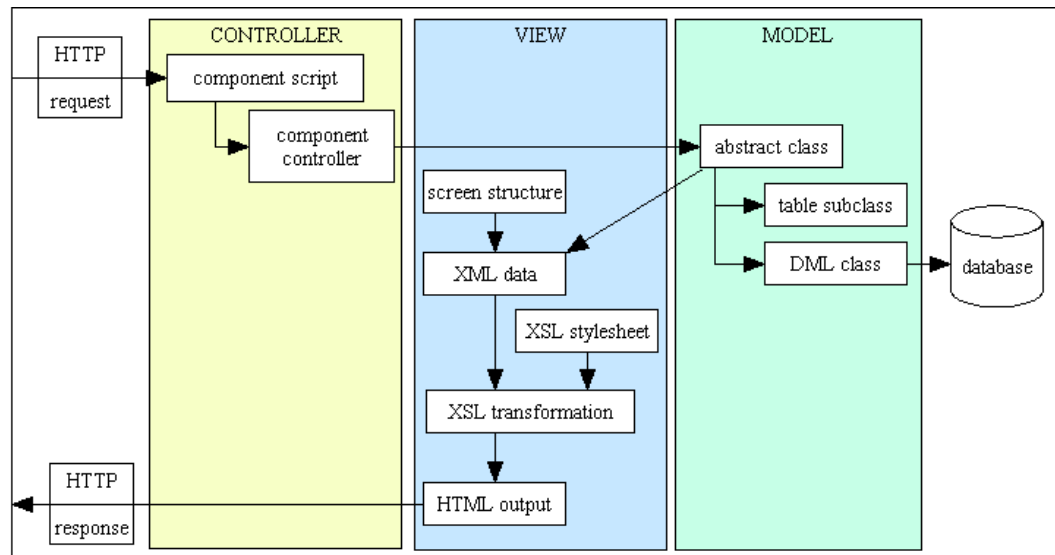


Figure 2-3 implementation of the MVC pattern

Compared with other development tool, such as Java, multimedia package is not well defined for monitoring features [62]. PerfSONAR³ is implemented as Service Oriented Architecture (SOA) for data exchange. However, the application seldom integrates with visual effect and difficult to deploy with multimedia packages on this stage. If we extend SOAP⁴ and XML schema with visual and audio effect on data exchange, monitoring and diagnosis would become more interested. In the face of challenges on visualized network monitoring, measurement data can be joyfully compared in same network property.

³ PerfSONAR is now released as v3.0 and downloaded from [here](#)

⁴ SOAP, abbreviation of Service Oriented Access Protocol)

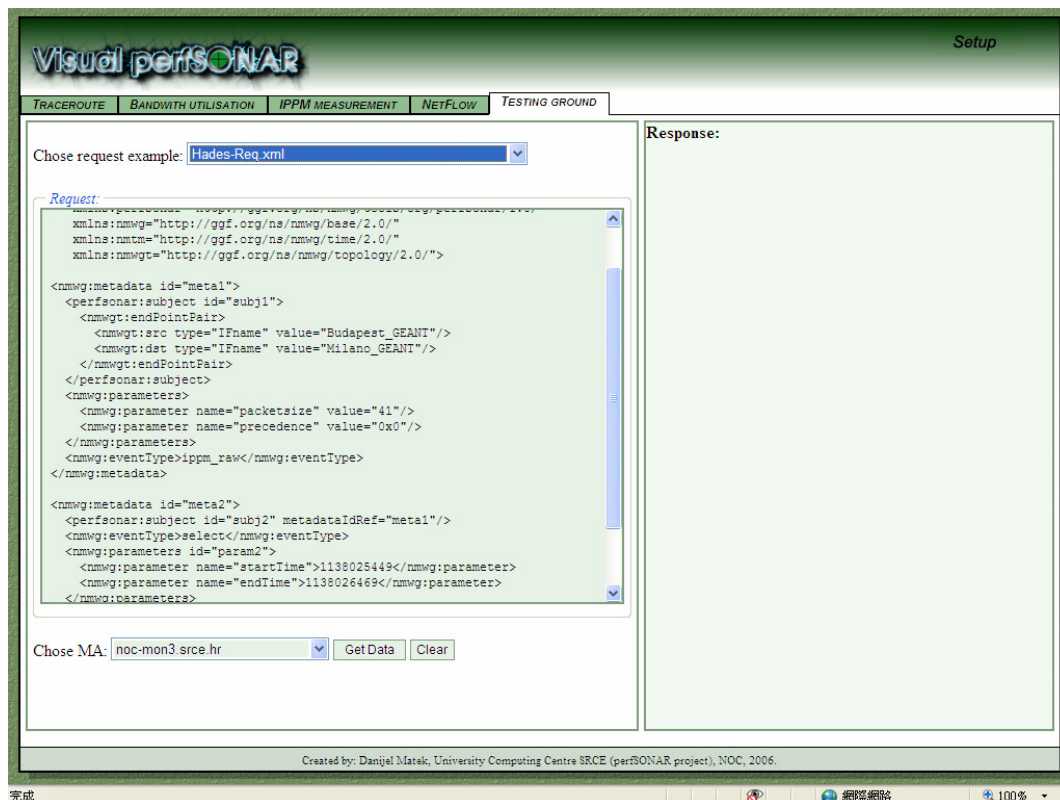


Figure 2-4 Visual PerfSONAR on SOA architectures

There are much ongoing visualization efforts for e2e topology infrastructures in which develop interactive methods of implementing measurement results in monitoring system. These efforts are directed towards scalability and efficient tool. For network monitoring application to be able to use smart interactive technique that visualizes measurement points, the requirement is that the technique should provide more interaction that measurement results can be explored for diagnosis purpose. Alternatively, if it is capable of providing rich information from original traceroute results between two endpoints, then monitoring measurement tool can be built.

Assuming e2e topology having built over web environment, interactive programming languages, such as Flash, is considered and questioned to visualize network measurement results.

1. Does Flash development tool primarily help minimize a barrier to the deployment of network monitoring tool?
2. Can Flash application retrieve relevant set of measurement results in a timely manner?
3. May Flash-based monitoring system serve as inter-domain machine?
4. Have Flash been better visualization effect on network monitoring tool?

Flash presentations are assumed to be interactive model and can incorporate with measurement results that can be integrated. It can be also integrated into Web site to display continuously for better visualization effect. For network monitoring purpose, flash-based application is to visualize measurement points interactive with multimedia visual effect. For example, we can display interactive and lively sequences on how Internet paths perform between two endpoints in multimedia manner. If any of the paths operate below threshold, special visual effects, such as blurring, will be displayed that end-users can catch the issue. With multimedia features, flash application can have faster responses on measurement results than

other non-multimedia languages, such as Java. Moreover, flash application can allow multiple paths to be compared in form of movie show on basis of measurement metrics, such as round-trip time, forward and backward packet loss rate, and reordering rate with different probe and response size. In the paper on algorithmic mechanism design, Nilsan and Ronen [18] advocate combining an economic approach with the more traditional protocol-design approach to the networking problem. Having monitoring system been designed as multimedia package, people are more fun and interested on networking issues than traditional design.

Having FlashyPath been considered as interactive diagnose purpose; traditional measurement function may be extended with the following interactions between user interface. Given the web technologies, standard diagnosis functions for a pair of measurement points are generally included in the following:

1. Looks up available source and destination measurement points from data store.
2. Retrieve monitoring characteristics between a particular source and destination and the statistics that the framework can apply to the result
3. Query particular source, destination, characteristic and statistic

4. Add visual effects, such as flow, fade, mask and tween properties⁵, on characteristic and statistics information
5. Display the result on the AS topological graph.

In essence these monitoring activities may be separated on the basis of time. Background monitoring looks over days, weeks and months may be animated in different speed or velocity at the behaviour of the network whilst immediate monitoring provides a snapshot of existing conditions within the network.

1. Looks up available source and destination measurement points from data store.
2. Retrieve monitoring results in days, weeks or months between a particular source and destination.
3. Add animation effects, such as Zoom, Dissolve, and Blur properties, for different Internet path
4. Display the measurement result on the chart

Given Flash as interactive visualization tool, e2e topology can be visually worked with connectivity of graph in Autonomous Systems (ASes). The hierarchical

⁵ More visual effect can be referenced from flash visual effect packages on <http://livedocs.adobe.com/flex/2/langref/mx/effects/package-detail.html>

graph layout is merely constructed with IP address nodes, which may incur long animation processing time on the graph in multiple Internet paths, thus making monitoring tools complicated. If we construct the e2e topology in term of IP address, it would cause complex and time-consuming approach. The novel example on IP-based graph is that Skitter [36] probes the Internet paths to many destination IP addresses spread throughout the IPv4 address space and visualizes the directed graph from a source on the Internet in static manner. Reducing processing time would forgo the global network information. Cheswick and Burch [37] has examined IP address level graph layout would restrict coverage of a globally diverse set of network. Therefore, connectivity of graph in ASes is easily deployed and animated in the Internet environment. Generally, Internet connects thousands of ASes operated by different Internet Service Providers (ISPs). Routing within an AS is easily controlled by interdomain protocols such as static routing, OSPF, IS-IS and RIP. Border Gateway Protocol (BGP) is an interdomain routing protocol that allows ASes to select routes and propagate routing information. Depending on the connectivity of upstream ISPs and traffic patterns, ISP will therefore suit the available bandwidth of the respective connections to varying degrees. However, the current AS topological graphs seem to be short of interactive features for end-users. Skitter, in Figure 2-4, is famous connectivity tool for visualizing IP addresses in graphical representation. Getting much IP

addresses construct complex spherical layout. Therefore, it does also not provide interactive feature for ISP operators who easily locate the Internet path measurement between two endpoints.

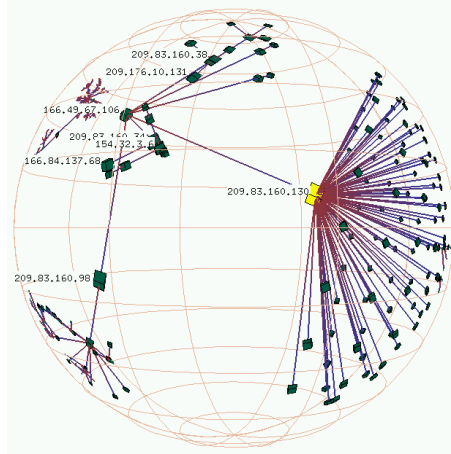


Figure 2-5 Skitter visualizes network connectivity without interaction model

2.2. FLASH AS INTERACTIVE PACKAGES

In order to visualize interactive networking monitoring purpose, Flash is assumed to be good multimedia package for e2e topology design. Flash, by definition, is a multimedia graphics program especially for use on the Web. Specifically, it enables to create interactive movies on the Web and uses vector graphics, which means that the graphics can be scaled to any size without losing clarity/quality.

Integration with multimedia of Flash is to natively support streaming audio and video. If we are working with multimedia, Flash is definitely the Rich Internet Application (RIA) platform to build network monitoring application. Flash comes

with UI toolkits for such multimedia object as tree views and data grids. It is important to Flash is a much better option compared with other programming packages.

On the other hand, Flash can integrate with Actionscript languages for enhancement of visual effect. ActionScript is the scripting language of Macromedia Flash. A scripting language is a way to communicate with monitoring system; you can use it to tell Flash what to do and to ask Flash what is happening as a movie runs. This two-way communication lets ISP operate create interactive images on Internet path measurement. For example, Flash can capture measurement results and show on the Internet map with different color and animation that ISP operators can easily navigate the Internet paths. Therefore, the operators can drill down the red-colored state to look thoroughly at the current state of each Internet path. Figure 2-5 is an example that Internet map could exploits the interactive model that end-users can zoom the affected area. Moreover, the map also provides color scheme for threshold that end-users can navigate different scenario for Internet paths.



Figure 2-6 Animated Atlas - <http://www.animatedatlas.com/movie.html>

In brief, Flash can be summarized as following benefits on interactive monitoring tool.

1. Flash can make a web site more attractive, interactive and dynamic.
2. Within a 300 second period Flash can show measurement result on Internet paths.
3. Flash has now become a very well-recognized format on the internet, and it is estimated that over 90% of web users now have the Flash Player installed on their computers
4. Flash distribution object can be stored in very small file sizes, so they can be downloaded rapidly, thus achieving the interactive model for continuous Internet path measurement

2.3. AUTONOMOUS SYSTEM TOPOLOGICAL GRAPH

For interactively visualizing measurement results, AS graph is constructed to describe Internet paths for connectivity from router-level information. An autonomous system (AS) is known as either a single network or a group of networks under the control of a single administrative entity, typically an ISP or large organization with independent connections to multiple networks. ASes in the topology may differ which varies in size, type and number of relationships with their neighbours. The ASes graph [55] better visualizes Internet path measurement from traceroute data due to unreachable IP addresses on Internet space. Khalifa [57] describes two techniques, namely BGP-based and probing-based, on AS topology from router-level data. The former continuously update best routes information in BGP routing tables, which can be publicly announced. The latter constructs ASes connectivity relationships from traceroute utility on all possible forwarding paths. The two techniques have been summarized for collection of AS topological data for graph in Figure 2-1. For completeness of topology information, the probing-based technique may provide complex relationship between each router on physical connectivity. Moreover it also provides low-level directed reachability graph on Internet path.

	BGP-based techniques	Probing-based techniques
Algorithm complexity	Straight forward	Complex
Output data	High-level connectivity graph	Low-level directed reachability graph
Availability	Publicly available (WWW)	Available upon request
Completeness	Less interconnections (edges)	Less ASes (vertices)
Graph edges	Represent peering relationships	Represent physical connectivity

Figure 2-7 Comparison of the two AS mapping approaches

Studies have previously shown that mapping IP addresses to AS number is not simple because of incomplete and out of date in WHOIS database. Mao et al [58] identified that 10% of traceroute paths contained one or more hops that did not map to a single AS number. Furthermore, mapping IP addresses to AS numbers paths resulted in loops in the inferred AS-path for about 15% of the node-level paths examined.

2.3.1. TRACEROUTE

For probing-based technique, AS connectivity data is mapping from traceroute program, which returns the path taken by packets sent to a particular IP address with time-to-live (TTL) counter. In theory, the TTL field is given an initial value which is decreased by one every time it is forwarded by a router and eventually dropped when its value reaches zero. This ensures that the packet will be dropped

once the maximum number of hops is reached. The router that discards the packet returns an ICMP time exceeded message to the packet source, thus unveiling its IP address.

The traceroute starts by sending a packet to a given destination with a TTL equal to one. The first router reached by the packet will then discard it and reply by a time exceeded packet showing in its source field one of the router's IP addresses. This IP address corresponds to the interface on which the reply packet was sent, most probably the interface through which the traceroute source address is routed. An IP address of the first router is thus known, but not necessarily the destination address of the probe packets. Traceroute then sends a packet with a TTL of 2 to the same destination, discovers an IP address of the second router, and so on. The probing ends when the maximum number of hops is reached or when a reply indicates the destination has been reached. Its output shows up as a list of IP addresses belonging to routers which responded to the probe packets at each TTL with their response times.

1	158.132.10.28	0.426 ms	0.299 ms	0.262 ms
2	158.132.254.65	0.972 ms	0.882 ms	0.620 ms
3	158.132.254.38	1.903 ms	1.002 ms	1.098 ms
4	158.132.12.20	1.525 ms	4.875 ms	4.465 ms
5	203.188.117.69	6.402 ms	4.107 ms *	

6	203.188.117.6	98.433 ms	18.120 ms	10.407 ms
7	147.8.239.15	5.336 ms	5.650 ms	17.551 ms
8	147.8.240.228	11.314 ms	4.310 ms *	
9	147.8.240.238	34.478 ms * *		
10	* * *			
11	www.hku.hk (147.8.145.43) [open]	4.552 ms	13.732 ms	11.576 ms

Table 2-1 Traceroute from 1.1 to the University of Hong Kong

2.3.2. TRACEROUTE MEASUREMENT RESULT

The traceroute measurement result is assumed to collect performance metrics between two endpoints, constructed with specified tuple, which illustrated in Figure 2-7. The tuple of each line in a file represents measurement result on particular analyze item between Internet paths in which probe and response sizes are specified.

<monitoring timestamp>,<measurement analyze item>,<measurement result>
--

Table 2-2 Sample traceroute measurement result tuple

```

1208874425.350936,49,1208874425.350936
1208874425.350936,37,1536514702
1208874425.350936,39,4110353350
1208874425.350936,29,1208874425.589025
1208874425.350936,33,52
1208874425.350936,7,4110355669
1208874425.350936,11,1536514702
1208874425.350936,23,0.238089
1208874425.350936,50,1208874425.350990
1208874425.350936,38,1536514704
1208874425.350936,40,4110353350
1208874425.350936,30,1208874425.591136
1208874425.350936,34,52
1208874425.350936,8,4110355669
1208874425.350936,12,1536514704
1208874425.350936,24,0.240146
1208874425.350936,75,3
1208874425.350936,72,-1.000000

```

Figure 2-8 Sample measurement results for Internet path

The measurement results from the input file is analyzed and computed for Internet path measurement. The source code of function is to be listed in Appendix 7.5.1.

For each source node, the traceroute measurement files will be looked up in which filename pattern is extracted, assuming all filename path remains constant. The filename tells us three properties of measurement

1. The date in which traceroute measurement is to be implemented
2. The probeSize and responseSize of each Internet path
3. The measurement result for the Internet path on performance metrics

The implementation of traceroute measurement result for each Internet path

measurement would be described later.

2.3.3. IP-TO-AS MAPPING FOR E2E TOPOLOGICAL GRAPH

Autonomous system number (ASN) for each router-level IP could map traceroute data in forward path from external lookup algorithm. Govindan and Tangmunarunkit [35] have developed Internet discovery tool, but does not use animated effects on client side. In our implementation, we repeatedly convert all IP addresses on Internet paths and show visually in the AS topology graph. Minimizing monitoring workload should FlashyPath convert IP addresses into AS numbers and drawn as Internet topology graph. In face of AS topological graph layout design, we can build multiples paths between animated source and destination nodes. Usually, the AS graph is reconstructed by merging information collected by a number of repositories managed by private and public research organizations. IP-to-ASN mapping tool⁶ has been developed that provides quick, summarized, view of prefixes seen in an entire set of measurement results. Theoretically, traceroute IP addresses on Internet path are an approximation of the hop-by-hop router-level forward path a packet would take to a destination. Nevertheless, there is no direct method of obtaining ASes path from IP path,

⁶ The IP to ASN translation in RouteViews project is undertaken by RIPE in 2003

external tasks on IP-to-ASN mapping approach would be employed that convert traceroute IP addresses to AS path information for each hop. Traceroute is widely used to detect routing problems, characterize end-to-end paths, and discover the Internet topology. Providing an accurate list of the ASes along the dual paths would make traceroute even more valuable to researchers and network operators.

A workable algorithm for IP-to-ASN translation has been developed. Given IP addresses, we need gather routing tables from several routers all over the world and extract a view of the Internet from each of them. The following procedures would be then executed for mapping for IP-to-AS for each import. A third-party Perl script is to implement the translation for IP addresses in each Internet path on measurement results. The perl script describes that it collects the measurement result files from the wrapper program and performs IP-to-AS mapping algorithm. This indicates the ASes that announce a given prefix as its origin ASes.

The IP-to-ASN algorithms draw on analysis of traceroute probes, reverse DNS lookups, BGP routing tables, and BGP update messages collected from multiple locations. The mapping allows us to home in on cases where the BGP and traceroute AS paths differ for legitimate reasons. Much research has experienced

that the algorithm is to reduce the initial mismatch ratio of 15% between BGP and traceroute AS paths to 5% while changing only 2.9% of the assignments in the initial IP-to-AS mappings. The algorithm is robust and can yield near-optimal results even when the initial mapping is corrupted or when the number of probing sources or destinations is reduced.

2.4. GRAPH THOERY

In Autonomous Systems (ASes) topology, we design visually directed graphs reflecting Internet connectivity of each measurement point. The hierarchical AS graph is a layout algorithm that portraits the precedence relation of directed graphs. The layout algorithm aims to highlight the main direction or flow within a directed graph. Cyclic dependencies of nodes will be automatically detected and resolved. Nodes will be placed in hierarchically arranged layers. Additionally, the ordering of the nodes within each layer is chosen in such a way that the number of line (or edge) crossings is small. For Actionscript-based graph package, which names Flex Visual Graph Library [10], it is available for our graph layout algorithm. The Flex library is to advance the design and development of an open source data visualization library and component suite for Adobe Flex. Enabling to create complex data visualization interfaces for the analysis of relational data sets using

texts and images, the library is purposely extendable and provides for separation of base, interface, and layout code. Additional layout algorithms can be readily integrated as an extended class containing only the mathematical calculations and controls needed specifically for the layout.

Upon completion of IP-to-AS mappings mechanism, the visually directed graph can be drawn. When the measurement points have been selected from monitoring system, the Actionscript will invoke HTTPService object which retrieves the AS data from domain database. The XML objects containing AS file and path information will then be returned. FlashyPath will execute Flex Visual Graph Library and draw the hierarchical layout for end-users. Moreover, the library will also provide other information, such as source IP, destination IP, experiment ID, that users can select measurement results with probeSize and responseSize. The Flex Visual Graph support the creation of vertices and edge connected with Edge. As usual, we build the graph of the ASes starting from a set of AS paths generated from measurement results. Each AS path is associated with a number of paths and sequence of ASes traversed by the traceroute data to be delivered.

2.5. FLEX AND FLASH

In a multi-tiered model, FlashyPath applications can be served as the presentation tier, which is mainly integrated with Flash player and Actionscript programming. The Flash applications in form of Flex architectures can take advantage of data streaming and rich media integration to provide compelling functionality that would be inexpensive to develop e2e technologies. Since Flex applications are accessed via standard web browsers, they can take advantage of the web deployment model's traditional benefits such as single-sign-on and secure data transfer.

Flash integrating Flex is a framework that helps build dynamic, interactive rich Internet applications. Flex applications are delivered on the web via the Flash Player or to the desktop. In Flex, a higher-level language named MXML, an XML language that sits on top of ActionScript. Because MXML is a domain-specific language targeting the rapid development of Flex user interfaces, you can quickly become productive with MXML. The MXML syntax allows inserting complex component into network monitoring application with limited lines of code. MXML's XML tags can have children, and it's through XML child elements that you can do layouts inside containers. Unlike Java, Flex provides well-structured layout, such as VBox and HBox, components, which is a container that lays out its

components vertically and horizontally respectively. You specify child elements of the VBox XML tag to put child components inside the box, and the box knows how to lay those components out. Moreover, the containers, such as Panel, Canvas, and TitleWindows are supplied that developers can develop application-like measurement tool.

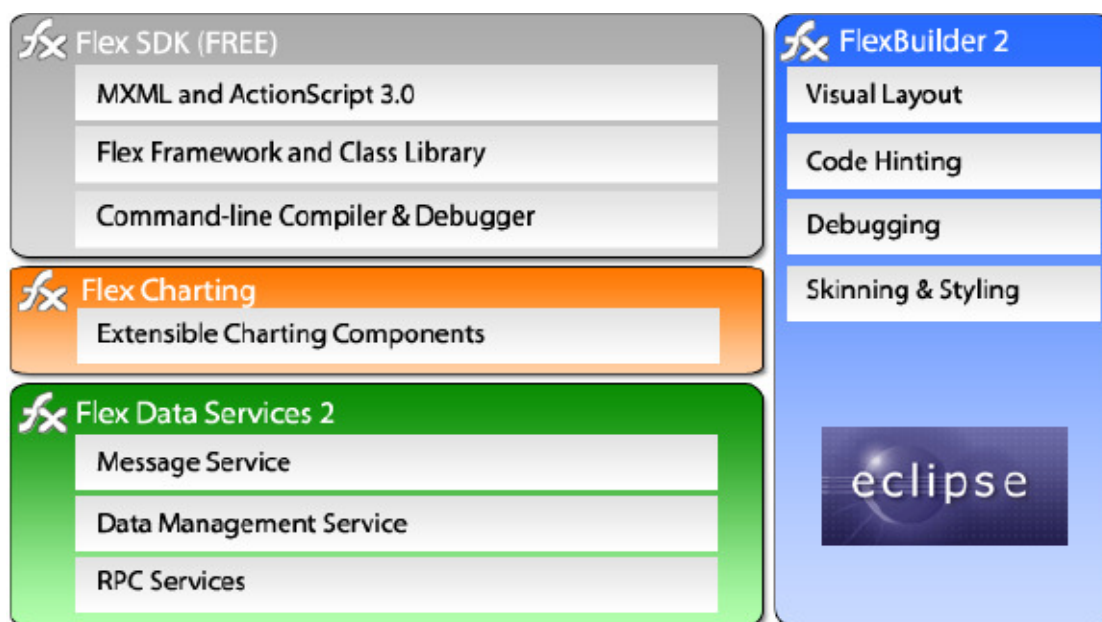


Figure 2-9 The development of RIA application through Flex modules

Although a Flex client can use direct socket connections, the most common way to communicate between a Flex client and a back-end service is through HTTP. HTTP communication is very easy with Flex, because we provide a simple HTTP client API with the HTTPService component. HTTPService is an API on top of the Flash VM's HTTP connectivity that, in turn, uses the browser's HTTP library.

HTTPService provides a higher-level abstraction for Flex developers to have a nice API to make HTTP requests. In most cases, the only things you need to specify to an HTTPService are a URL and a call-back handler function. The callback function will be invoked when the HTTPService gets back a response.

The HTTPService supports REST-style HTTP URLs as well as SOAP URLs. If you organize your back-end API such that the action you want to perform come in as a parameter to a single URL, then you can just have one HTTPService for all those calls. With multiple URLs for the various requests to the server, you can easily set the URL value on one HTTPService. In larger applications, you typically write a singleton model locator object to access all your data objects.

Adobe Flex relies on the Flash 9 browser plug-in, which needs to be present in the browser of the website visitor. Ajax uses the various Internet browsers as its runtime. Some Ajax Frameworks have an additional JavaScript engine that abstracts away from differences in browser implementations: developers use the engine which in turn communicates with the browser. The benefit of using a proprietary plug-in as Flash 9 is the controlled runtime environment, which is identical across all web browsers. This makes development easier and it allows Adobe to add additional

features and to improve performance. The downside is that the plug-in needs to be installed, which can pose a problem in environments with locked-down operating systems. Adobe periodically publishes data on the market penetration of the Flash Player. Some critics of Flash state that reliance on a plug-in is a break with web standards, as the web browser is only used to launch the player which does not use web standards such as HTML, CSS and JavaScript. The Flash plug-in offers some support for HTML, CSS and an extended version of JavaScript

2.6. FLASH AND JAVA

Most of network monitoring tools are preferable using Java because Java supports multi-threading and remote procedures on data-binding services. The application of Java technology helps greatly in making truly portable and less dependent on slow Internet collection [4]. For concerns of visual consistency, Java serves as certain function with user-customized parameters, and displays both the original and the resultant images. It also allows end-users to choose operating parameters according to their own wish. Java moreover can be used either as client-side applet for rich application or can be used on the server-side for writing the application code delivered to browsers through HTML/PHP. However, Java runtime object on the separate environments cannot be equally operated because

Java built as desktop application must be redesigned in form of client-side applet because of different development cycle.

From the source of UWEBC, Flash is obviously better than Java on the area of graphical richness, various computing environment and audio/video support. It is fairly easy development tool on XML, DOM, Javascript and Actionscript that meet open standard requirement. As Flash executable files are binary compressed, Flash application can be developed as sandbox that network measurement results are security protected.




























	AJAX	Macromedia Flash	Java
Graphical Richness	 Average (Same as HTML)	 Very Rich	 Rich
Container/Engine Footprint	 Very Light (browser built-in)	 Light	 Heavy
Application Download	 Fast	 Slow	 Slow
Audio/Video Support	 Poor (unless use ActiveX)	 Excellent	 OK
Consistency on Different Computing Environments	 Varies	 Very consistent	 Relatively consistent
Server Requirements	 None or very minimal (TIBCO General Interface)	 Yes (Flex or Open Laszlo)	 Yes or No (Nexaweb, Java Web Start)
Plug-in/Runtime Requirement on Client	 No	 Flash (Player)	 Java Runtime (JRE)
Development Challenge	 Very complex without tools such as TIBCO, and high skills required (JavaScript, CSS, XML, XSLT, DOM, ActiveX...)	 Relatively easy with tools such as Flex or Open Laszlo (XML, DOM, JavaScript, Flash, ActionScript)	 Relatively easy with tools such as Nexaweb (XML, JavaScript, Java)
Security Concerns	 JavaScript codes are open to public Everybody can see source codes if desire	 Flash files (compressed binary) are created Flash Player becomes a sandbox	 Class/Jar compressed binary files are created JVM (Java Runtime) becomes a sandbox

Figure 2-10 Platform comparison (Source: UWEBC⁸)

⁸ UWEBC: Report on platform comparison of Rich Internet Application (RIA) published by UW

Let us turn the page on deficiency of Flash application. Because Flash applications require fully download, we experience low response time when network applications operate low-bandwidth network environment. On the other hand, Flash requires a flash player as plugin for activation within the web page that is highly vulnerable due to frequent upgraded from Adobe development team. Unlike Java, Flash cannot be edited and duplicated because the source is a multimedia file that is loaded onto the web page. Aside from interoperability features on each kind of components, experimental statistics has shown that Flash application achieves better average file size including images and average load time than Java runtime [27]. It also concludes that Flash is most practiced for network transmission.

In short, Flash application is relatively light-weighted solution for the development of monitoring measurement and will be illustrated later in more detail.

3.DESIGN AND IMPLEMENTATION

The core components of FlashyPath that we, as developer's perspective, have implemented are interactive model of visualizing measurement result collected by external wrapper. In this section we complete the description of FlashyPath by showing how the interactive application is to be designed. Moreover, we have shown FlashyPath of how visualizes the measurement results are to be post-processed in Flash environment.

Throughout the report, Flash integrating Flex with Actionscript 2.0 will be mainly used to illustrate the development of FlashyPath application. FlashyPath will provide a set of API which implements core part on AS-level topology with measurement results. For other programming languages, XML as common data interchange format between the application and server-side programming.

FlashyPath will employ PHP programming to interface with alignment of MySQL and round robin database which store continuous measurement data. It helps improve the speed and integrity of data and relieve the job of round-robin database.

The Open-source packages are proposed in the application and illustrated in Table

- | |
|--|
| <ol style="list-style-type: none"> 1. Apache 1.2.2 with Actionsript 2.0 2. PHP 5.0 and Perl 5.8.8 programming 3. Flash player 9.0 or above 4. MySQL 5.0.32 5. RRDTool 1.2.15 6. Flex Visual Graph Library package 7. Flex Chart Component 8. XML and DOM |
|--|

Table 3-1 Open-source package for FlashyPath development

3.1. OVERALL ARCHITECTURES

The following open-source components are used for developing applications for FlashyPath. Figure 4-1 illustrates FlashyPath architecture that can be performed under web technologies. Apache HTTP server is primarily to serve measurement results over the Internet. Integrating PHP and Perl script packages, FlashyPath can communicate backend database through the package that implement measurement results to be worked. MySQL and round-robin databases are used to gather autonomous systems (ASes) and timeseries results respectively.

FlashyPath Software Development Modules (Version 1)

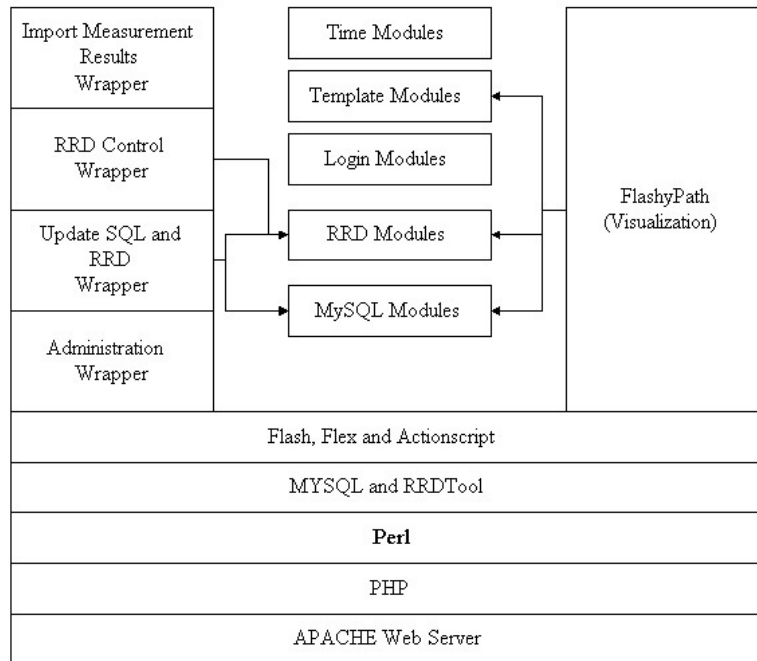


Figure 3-1 Software requirement of FlashyPath

3.2. DEVELOPMENT CYCLE

For FlashyPath, development cycle is firstly determined for entire architecture on programming level that builds network monitoring system. A development cycle is the sequence of events in the development of network application. Figure 3-2 illustrates each MXML operating specific function on monitoring system.

Flow of development cycle of FlashyPath

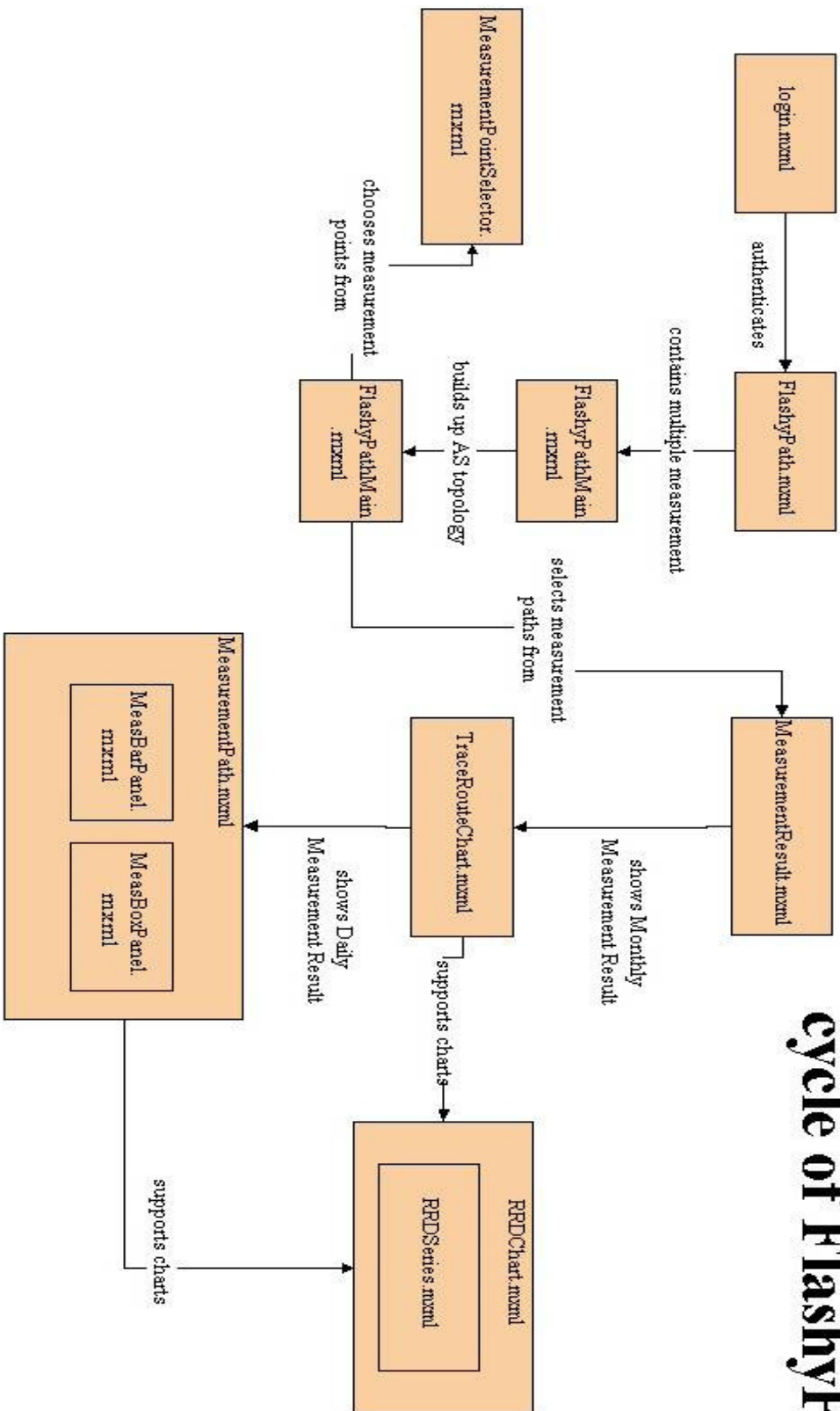


Figure 3-2 Development cycle of FlashyPath designed on each Flex application

9.

The following table shows description of each MXML file.

	File name	Description
1.	Login.mxml	It is to authenticate end-users who have access right of FlashyPath and moreover load the template end-users already saved
2.	FlashyPath.mxml	Core container which operates one or multiple measurement results.
3.	FlashyPathMain.mxml	Main measurement panel
4.	MeasurementPointSelector.mxml	End-users can select multiple measurement points for which AS graph is to be built
5.	MeasurementResult.mxml	Having AS graph built can select one or multiple measurement results on monthly data
6.	TraceRouteChart.mxml	The traceroute chart for monthly measurement results that end-users can drill down daily results
7.	MeasurementPath.mxml	The function is to visualize daily measurement results on multiple paths. It contains measurement boxes and bars for users who choose each measurement time.
8.	RRDChart.mxml	The RRDChart for time-sensitive data from HTTPService result object
9.	RRDSeries.mxml	It is dynamically to draw series of measurement results depending on the selection of measurement points
10.	MeasBarPanel.mxml	Shows measurement results in bar-style format
11.	MeasBoxPanel.mxml	Show measurement results in box-style format, which determines from experiment meta-data

Table 3-2 FlashyPath MXML in which measurement tasks are performed

3.3. WRAPPER DESIGN

Having considered each functionality on FlashyPath, the architecture is also designed the wrappers which manage measurement results from other applications within fixed interval. Due to FlashyPath developed as non-active probing system, wrapper programs are necessary to absorb new measurement results continuously. Figure 4-2 illustrates wrapper architectures for data management. The object-oriented wrapper is useful when a legacy program becomes the server in a client/server application. The measurement result control wrapper is the component which implements new measurement results in every ten minutes, controlled by crontab scheduling. The crontab job runs as background job that starts importing experiment results in every ten minutes.

```
start_time:1212609900
end_time:1213214700
expr_name:Harnet
expr_hrs:168
expDurPerSite_min:2
numSubExp:1008
curr_subExp:994
numMinReturnResult:10
data_path:/home/usthk_measure1/EXP0000018/analysis
src_list:/home/usthk_measure1/EXP0000018/analysis/opairlist
dest_list:/home/usthk_measure1/EXP0000018/analysis/targetlist
```

Table 3-3 Sample meta data description file for each sub experiment

The main wrapper control program, which is written in Perl script, executing a list

of job for new measurement results and summarized below

1. Given import directory, the wrapper reads measurement results in form of meta-data description file and experiment files from BACKUP directory.
2. Execute meta-data description file on each sub-experiment.
3. Read Traceroute result and execute IP-to-AS mappings that translate to Autonomous System (ASes) data.
4. Read and Update measurement result file for each target node.
5. Execute Housekeeping functions to clean up measurement result for next scheduling.

For wrapper design, we have performed seven days of experiments between 5 June and 12 June 2008. Each experiment contains traceroute and measurement results among measurement points. For each measurement result, we have traceroute result which contains hop-by-hop information between source and target nodes shown on Table 4-2. As we draw autonomous systems (ASes) graph layout, the traceroute result must be translated as AS-like topology structure that contains AS information on each IP address. Executing IP-to-AS mapping for traceroute file has shown in Table 4-3. For each IP address on the file, the perl script will translate

ASNum by using ip address through the URL

<http://eu.asnumber.networx.ch/asnumber/asnum?ip=158.132.10.28>. Therefore, the AS Num

will be returned as 4616 when ip address is registered and found, which is shown on

Table 4-4. Upon completion of IP-to-AS mappings, the output of AS file is stored

on the database for AS graph. Beginning at line 10 in figure 3-4, there is a sudden

timeout means packet does not return within the expected timeout window. For

IP-to-AS mapping, no AS path is returned because IP does not guarantee that all

the packets take the same route

```
#200806156201106
158.132.10.28  0.493 ms  1.049 ms  0.422 ms
158.132.254.65  0.706 ms  0.641 ms  0.672 ms
158.132.254.38  0.923 ms  0.720 ms  0.692 ms
158.132.12.20  1.324 ms  0.915 ms  1.047 ms
203.188.117.69  2.152 ms  1.994 ms *
203.188.117.6  3.097 ms  2.714 ms  2.724 ms
147.8.239.15  3.447 ms  3.557 ms  4.153 ms
147.8.240.228  2.778 ms  2.798 ms *
147.8.240.238  18.582 ms  18.623 ms  18.893 ms
* * *
www.hku.hk (147.8.145.43) [open]  3.448 ms  3.182 ms *
```

Table 3-4 Traceroute showing timeout result

```
<span xmlns="http://www.w3.org/1999/xhtml" id="asresponse" asnumber="AS4616"><div class="asinfo"><div class="title">AS 4616</div><table> <tr><td>Prefixes</td><td>:</td><td>22</td></tr><tr> <td>IP addrs</td><td>:</td><td>70912</td></tr> <tr><td>IPs/Prefix</td> <td>:</td><td>3223</td></tr><tr><td>AS name</td><td>:</td><td>PRISTINE-COMM</td></tr><tr><td>AS descr</td><td>:</td><td>Pristine Communications Limited Information Technology Services</td></tr><tr><td>Country</td><td>:</td><td>HK</td></tr><tr><td>Allocated</td><td>:</td><td>19950707</td></tr><tr><td>RIR</td><td>:</td><td>APNIC</td></tr><tr><td /><td /><td /></tr></table><div class="title">BGP Prefix</div><table><tr><td>Prefix</td> <td>:</td><td>158.132.0.0/16</td></tr></table></div></span>
```

Table 3-5 IP-to-AS mapping function returns XML result storing AS information on prefix, ip address, IPs/Prefix, AS name, AS description, Country and BGP Prefix of IP address.

The processASmain perl script will reformat the XML result object and print below AS path file format. The * symbol represents line 10 in traceroute results mentioned above shows timeout mechanism.

```
11|158.132.10.28, 4616, PRISTINE-COMM, HK
158.132.254.65, 4616, PRISTINE-COMM, HK 158.132.254.38, 4616, PRISTINE-COMM, HK
158.132.12.20, 4616, PRISTINE-COMM, HK 203.188.117.6, 3662, HARNET, HK9
203.188.117.6 147.8.239.15, 4528, HKU-AS-AP, HK
147.8.240.228, 4528, HKU-AS-AP, HK 147.8.240.238, 4528, HKU-AS-AP, HK *
147.8.145.43, 4528, HKU-AS-AP, HK
##DistinctPaths:1
##AllPaths:1
```

Table 3-6 AS file describes traceroute and AS relationship between two measurement points

For each experiment, the measurement results are controlled with a meta-data description file in which it describes experiment activity.

Having completed importing AS measurement results, the wrapper implements the measurement result data with respect to analyze items, such as RTT and forward and backward loss. The PHP script read the results and computes average,

minimum and maximum values for each item and update SQL database. Vern Paxson warns that collection of Internet measurement data will affect performance of wrapper program depends on system limitations, such as maximum file sizes and number of result data file. Therefore, the wrapper program would take a few seconds before starting importing measurement result data. Moreover, the program enables to be executed with MySQL configuration data that users can adjust wrapper result.

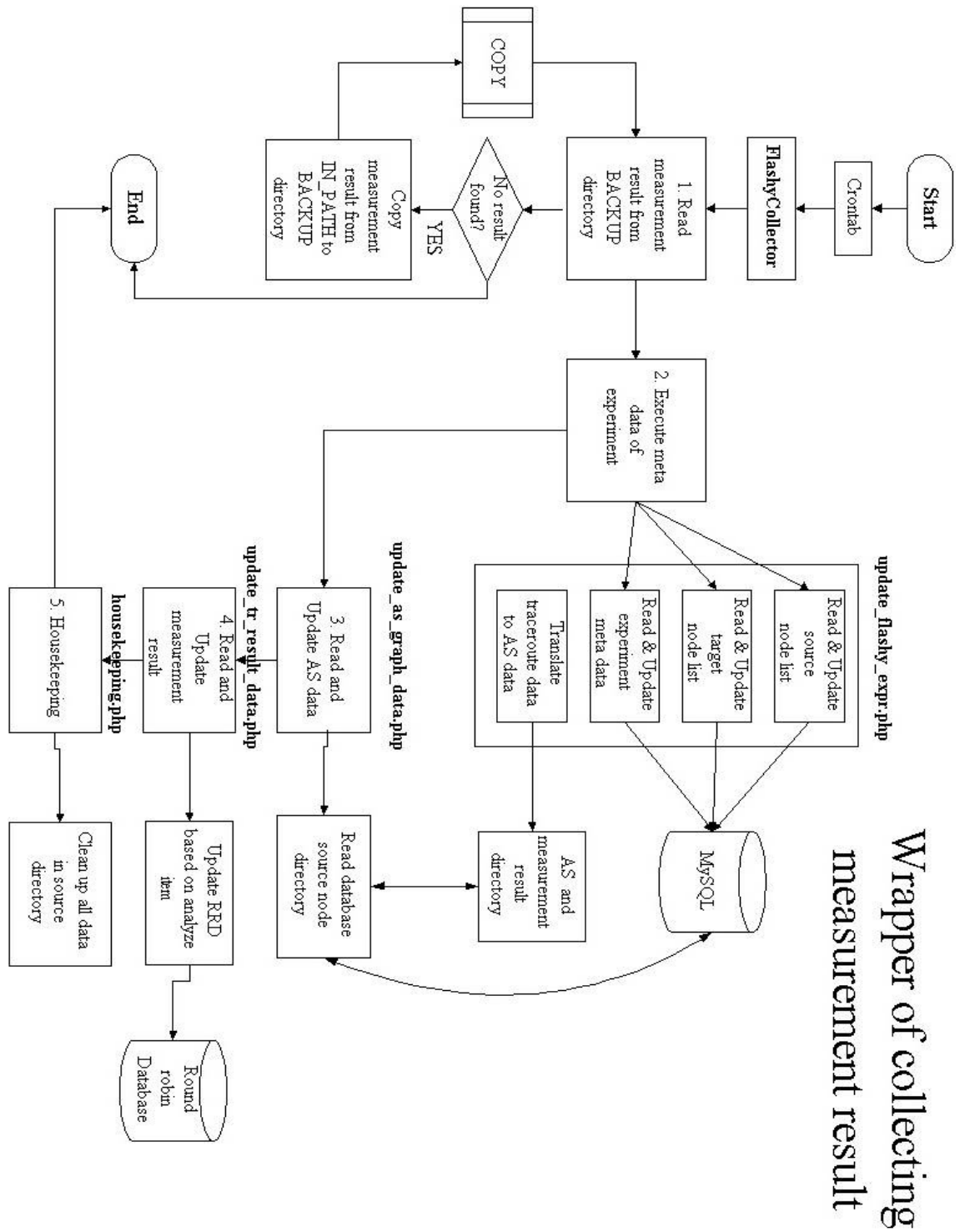
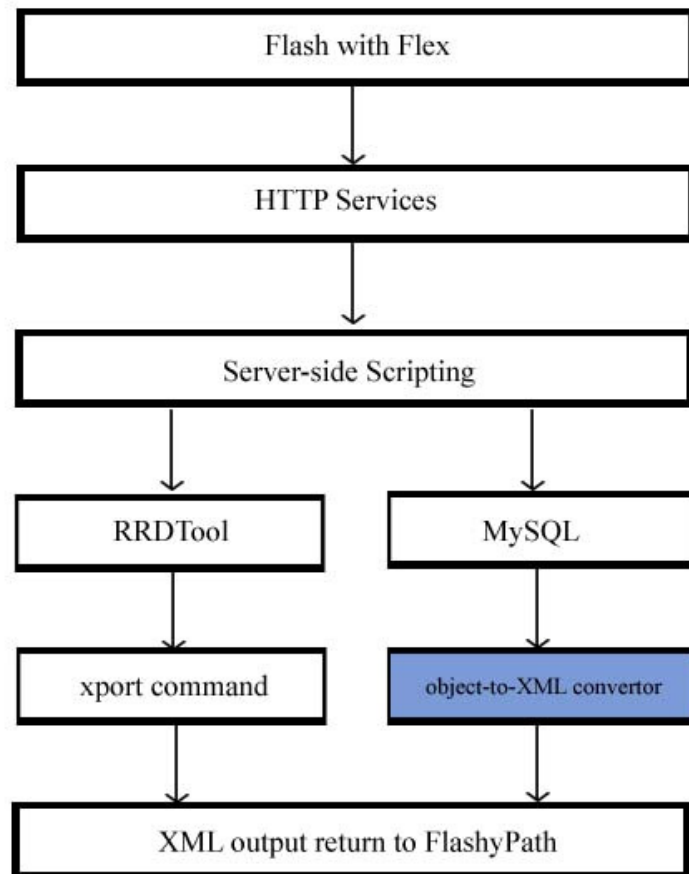


Figure 3-3 Design of wrapper of measurement result collection

The performance of wrapper program would be affected by physical file environment. Having created RRD file, the size approximates more 20M bytes on each analyze item.

3.4. ALIGNMENT OF RESULTS

Alignment of measurement results on source of data in different databases must be aligned. Recalling FlashyPath is to keep continuous measurement result stored on two databases, which output results are different. For this purpose, FlashyPath is proposed to only receive XML data wherever the measurement result is stored. It is known that RRDTool already provides standard xport command, which dynamically export measurement results as XML format, as visualization purpose for e2e infrastructures. Therefore, the SQL control wrapper for constructing XML data from SQL database is required. Figure 3-4 illustrates the flow of measurement results between FlashyPath and SQL database. It is found that object-to-XML wrapper for the database is to be built. The wrapper would store measurement results as XML format from database and send back to FlashyPath.



Mechanism of alignment of Round-robin and MySQL databases

Figure 3-4 Mechanism of alignment of round-robin and MySQL databases

Figure 7-1 in Appendix illustrates object-to-XML converter as PHP script which converts SQL result object to XML output. Having embedded root element, each result value will be quoted with user-defined child element. The advantage is not to restrict that enrich flexibility for the developers. Started with HTML Content-type as text/xml, the result will be thought of XML object and sent back to FlashyPath for visualization.

RRDTool export function is executed by PHP function, which control the

parameters sent from HTTP service in FlashyPath. The xport command provides description of RRD such as

1.	Start time
2.	End time
3.	Number of rows
4.	Number of columns to be displayed
5.	The names of legend
6.	The time and value of each row

Table 3-7 The XML hierarchy of RRDTool xport function

3.5. DATABASE DESIGN

For database design for FlashyPath, we build a list of core tables which stores ASes and timeseries information about measurement results. Generally, the user table is end-users whose userID are placed on each experiment that can place different measurement result templates that have created. FlashyPath defines template for the experiment for which exprID is placed the measurement has taken place. The userID and exprID are then actually derived from templateID field. For importing function, the measurement will be stored when we have computed the statistics, it will then update d_to_graph_item table with other important parameters, such as destID, sourceID, probeSize, responseSize and itemID. Figure 3-5 shows complete relationship between each table for measurement result for template management.

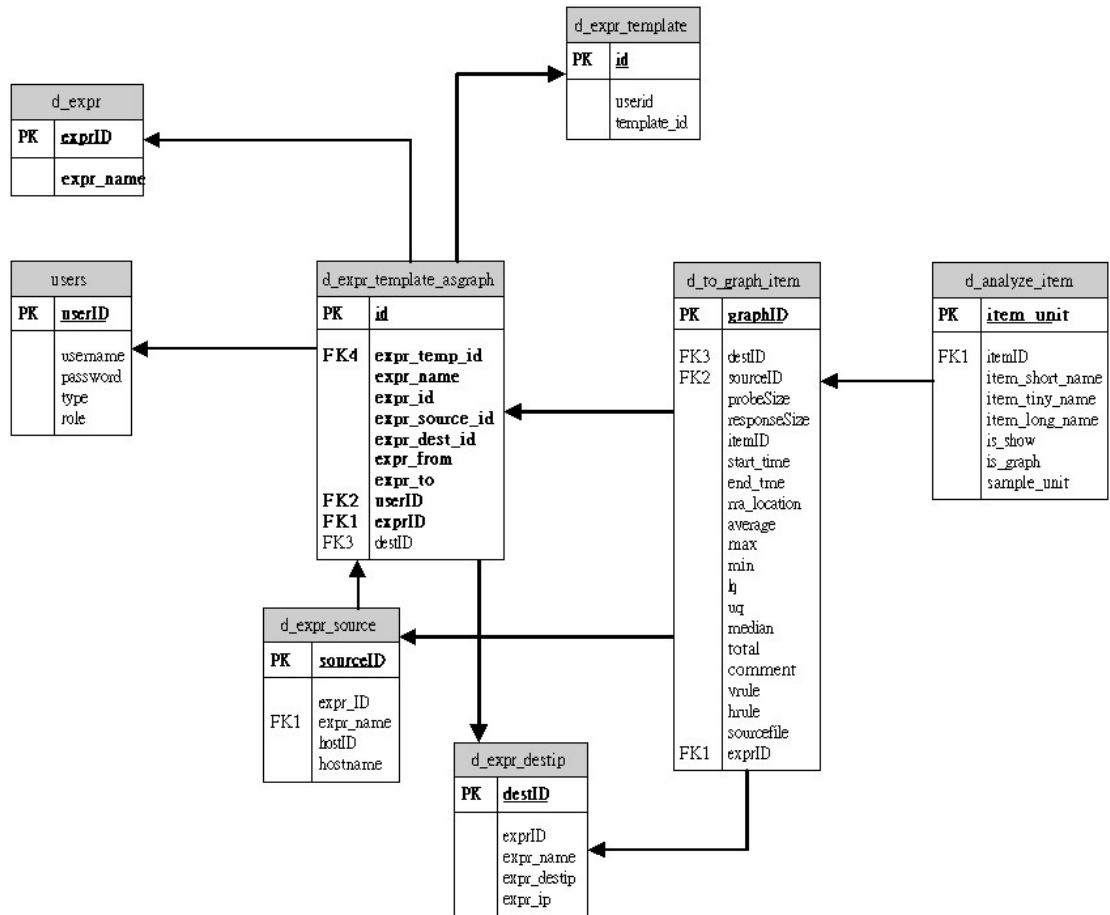


Figure 3-5 Database schema is developed for FlashyPath on AS paths and measurement results

The screenshot shows the FlashyPath database web interface. The table displays the following data:

graphID	destID	sourceID	probeSize	responseSize	itemID	instance_no	start_time	end_time	graph
10973	85	358	1460	1460	23	NULL	1212609843	1212609963	
10974	85	358	1460	1460	24	NULL	1212609843	1212609963	
10975	85	358	1460	1460	33	NULL	1212609843	1212609963	
10976	85	358	1460	1460	34	NULL	1212609843	1212609963	
10977	85	358	1460	1460	1010	NULL	1212609843	1212609963	

Figure 3-6 The database of FlashyPath

For the database design, the wrapper monitors specified directory hierarchy and automatically restarts PHP programs which update measurement result to the table `d_to_graph_item`. Supposed result data existed in default directory, a Perl program called `create_new_rrd.perl` would create new RRD file storing one-month data if it does not exist. Previous RRD file would be removed from directory. It ensures that the RRD file would not be continuously grown. Moreover, the backup directory will also be created when the measurement data has been updated for later stage. Note that the create wrapper is to be executed once on each month and the size approximately estimates 20M for each analyze item. Figure 3-7 has clearly shown that the flow of create and update RRD for measurement results. The 2-minute aggregated values will be stored on MySQL database for monthly measurement result. The timeseries data with respect to aggregated value will be separately stored on each itemized RRD files for daily measurement purpose. The RRD file is updated continuously until the end of month.

Create and Update Wrapper on result data for every 10 minutes to RRD file (Version 1)

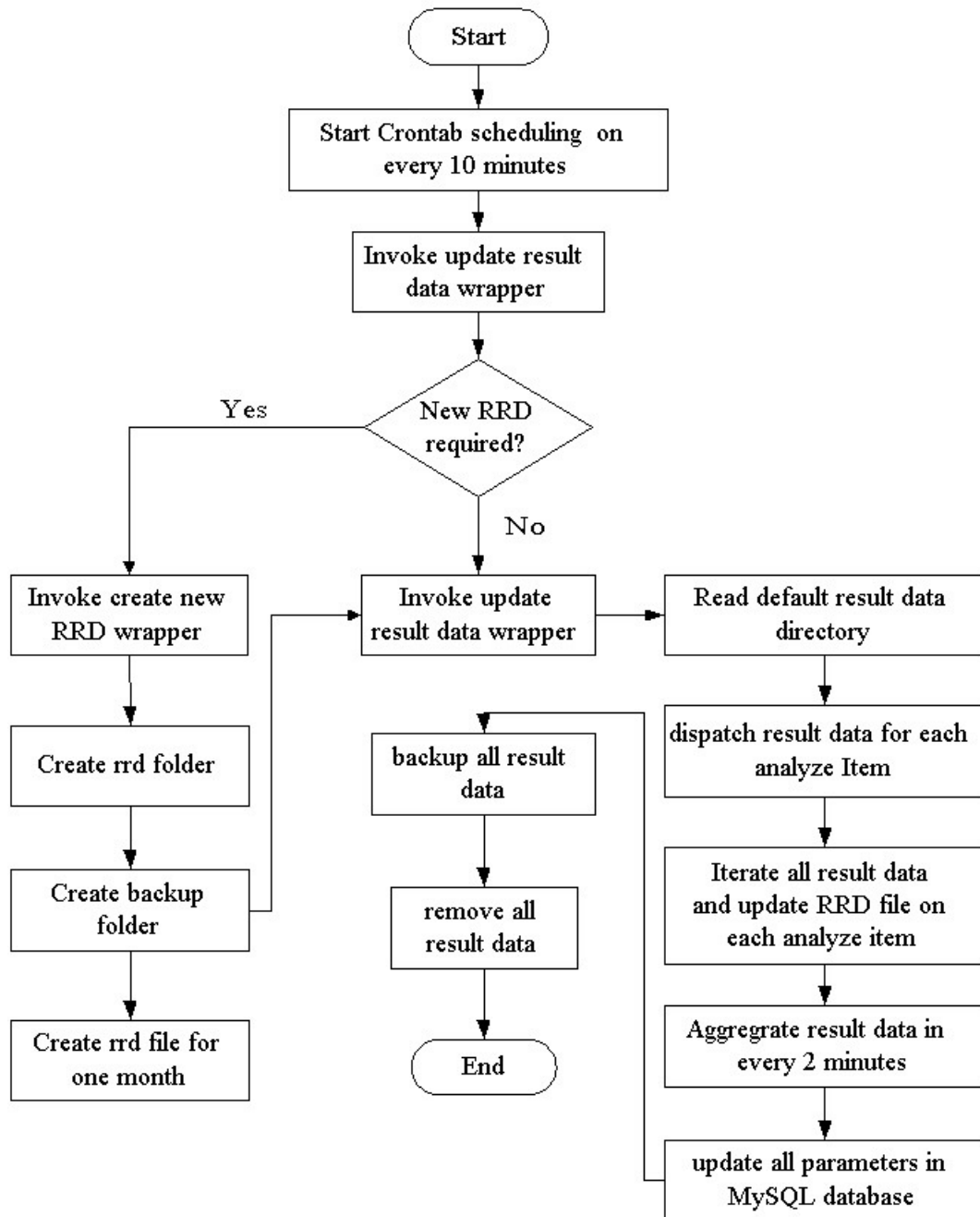


Figure 3-7 RRD control wrapper is to create and update RRD file using RRDTool for each analyze item

In detail, the round-robin database is to store measurement results on each item among measurement points on every ten minutes continuously. On first day of

each month, round-robin database for source and destination points is newly created using RRDTool standard command. For example, a measurement result file name, 20080604200401836181_www.bre.polyu.edu.hk-80_0.measure.result, is stored under a directory 1.1/result/20080604_result_1/1460_1460. The control wrapper [create_new_rrd.php] will get information from the filename. The directory 1.1 tells us source of measurement result. The date 20080604 knows the experiment is to be done. The target www.bre.polyu.edu.hk indicates the endpoint is to be measured. Moreover, 1460_1460 shows the probeSize and responseSize on each experiment. Therefore, RRDTool would create monthly RRD file using RRD create function and illustrates below

```
rrdtool create rrd/1.1/1460/1460/2008-06-30_www.bre.polyu.edu.hk-80_23.rrd
-s 1 -b 1212249600
DS:ID_RTT_BY_PCAP_0:GAUGE:2592002:0:10000
RRA:AVERAGE:0.5:1:2592003
```

Table 3-8 RRD create function for source and destination nodes for June 2008

In other words, the RRD file is stored under the directory rrd/1.1/1460/1460 in which the file is named 2008-06-30_www.bre.polyu.edu.hk-80_23.rrd. The file name tells us that measurement results from the target www.bre.polyu.edu.hk are stored between 2008-06-01 and 2008-06-30. The analyze item, 23, would be appended at the end of the file in which data source (DS) is specified by ID_RTT_BY_PCAP_0. The RRD would compute average on each second.

Moreover, the parameter `-s` and `-b` are the step between measurement results and start time of RRD file respectively.

3.6. VISUALIZATION MODULE

FlashyPath application is mainly designed as a Graphical User Interface so that we can visualize measurement results by calling it. The visualization module is mainly made up by Flash and Actionscript programming. The GUI allows the end users to specify the AS paths between two measurement points to be measured, together with traceroute result for the paths. That is, the user is requested login that the system can load the template of measurement points.

FlashyPath requires authentication requests from end users. The login module returns `USER_ID` of user upon the completion of authentication. The `USER_ID` is primary key for users that not only determines the role in application; the key is also used for linkage of other module, such as measurement template. The system would automatically retrieve templates from database by SQL query upon successful authentication. The measurement template, on the other hand, for measurement points can be newly created, users can select measurement templates stored on the system that minimizes the selection of measurement points every time.

Figure 3-8 illustrates the flow of FlashyPath from AS topology infrastructures to measurement results on multiple Internet paths.

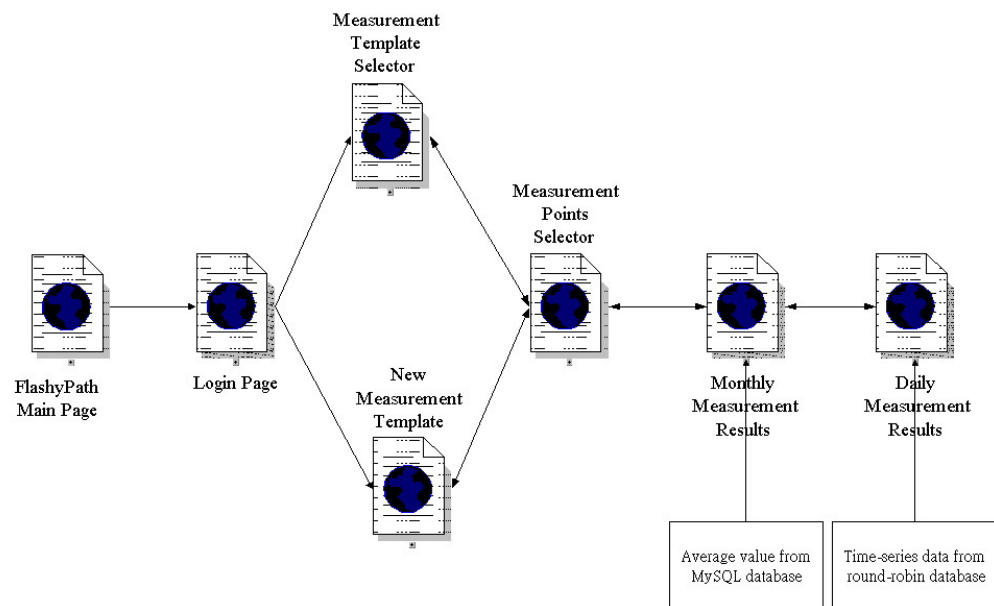


Figure 3-8 Diagram showing FlashyPath executing topology graph with measurement result chart

4.SOFTWARE DEVELOPMENT

In this section, we demonstrate how flash object as function or plugin is to be created for FlashyPath application. As part of that demonstration I generally develop a simple application that consists of a DataGrid, a Button and a WebService. This is great for showing layout management, events, RPC calls and data binding.



Figure 4-1 Adobe Flex Builder 2 built on Eclipse version 2.0.1

The development tool enables intelligent coding, interactive step-through debugging, and visual design of the user interface layout, appearance, and behavior of rich Internet applications. In detail, Flex SDK comes with a set of user interface components including buttons, list boxes, trees, data grids, several text controls, and various layout containers. Charts and graphs are available as an

add-on. Other features like web services drag and drop, modal dialogs, animation effects, application states, form validation, and other interactions round out the application framework. We outlined the basic skill that the simple application by the Flex builder can create application quickly.

1. Define an application interface using a set of pre-defined components (forms, buttons, and so on)
2. Arrange components into a user interface design
3. Use styles and themes to define the visual design
4. Add dynamic behavior (one part of the application interacting with another, for example)
5. Define and connect to data services as needed
6. Build the source code into an SWF file that runs in the Flash Player

Table 4-1 the sample procedures of Flash to be implemented

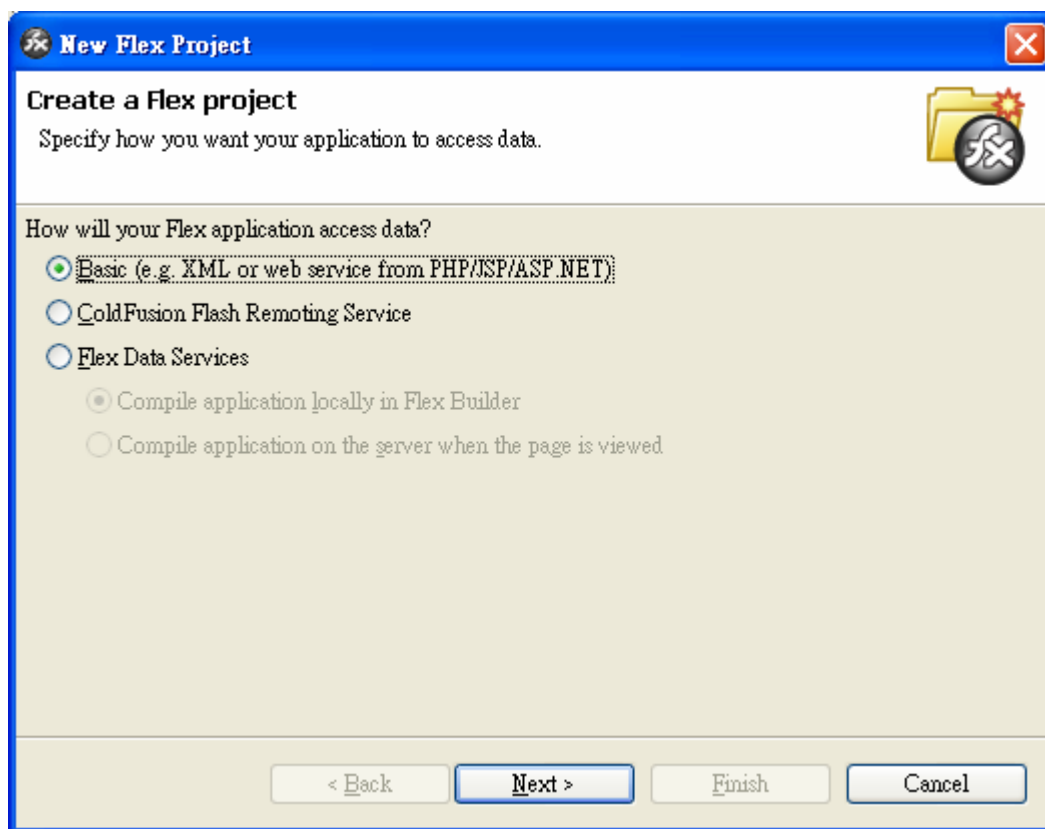


Figure 4-2 Create Flex application in XML or web service from PHP

We start developing FlashyPath with Flash integrating Flex with the builder is used.

Having launched successfully, Flex application can be built different level of data management. For simplicity, FlashyPath is now picked the basic level on XML or web service from PHP or JSP, shown on Figure 4-2. The other two options are more complicated for advanced data binding services on large-scale infrastructure.

When the project is newly created, the source and design workplaces are displayed in Figure 4-3, which designs MXML-based components. In designer view, the Flash object can select and drag component directly from components list.

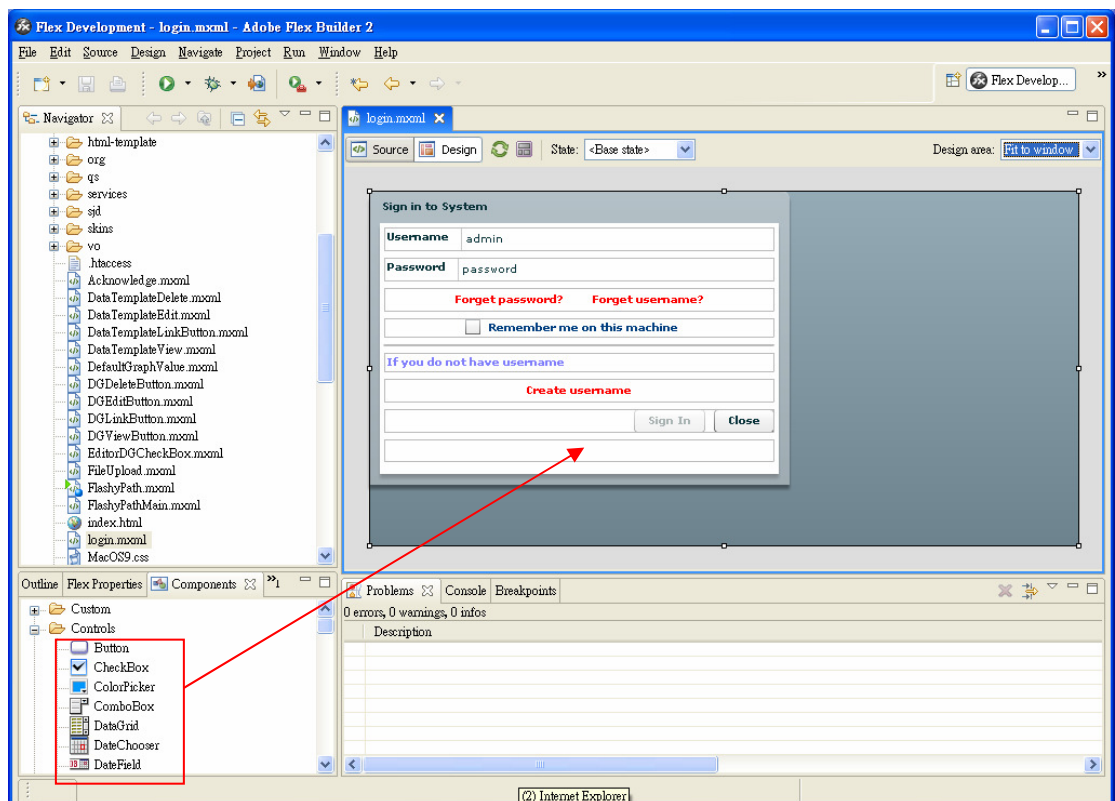


Figure 4-3 Flex Development Workplace in designer window

Alternatively, the designer may adjust the parameter of each component in source

windows that can efficiently fine tune the property of components in making Flash objects.

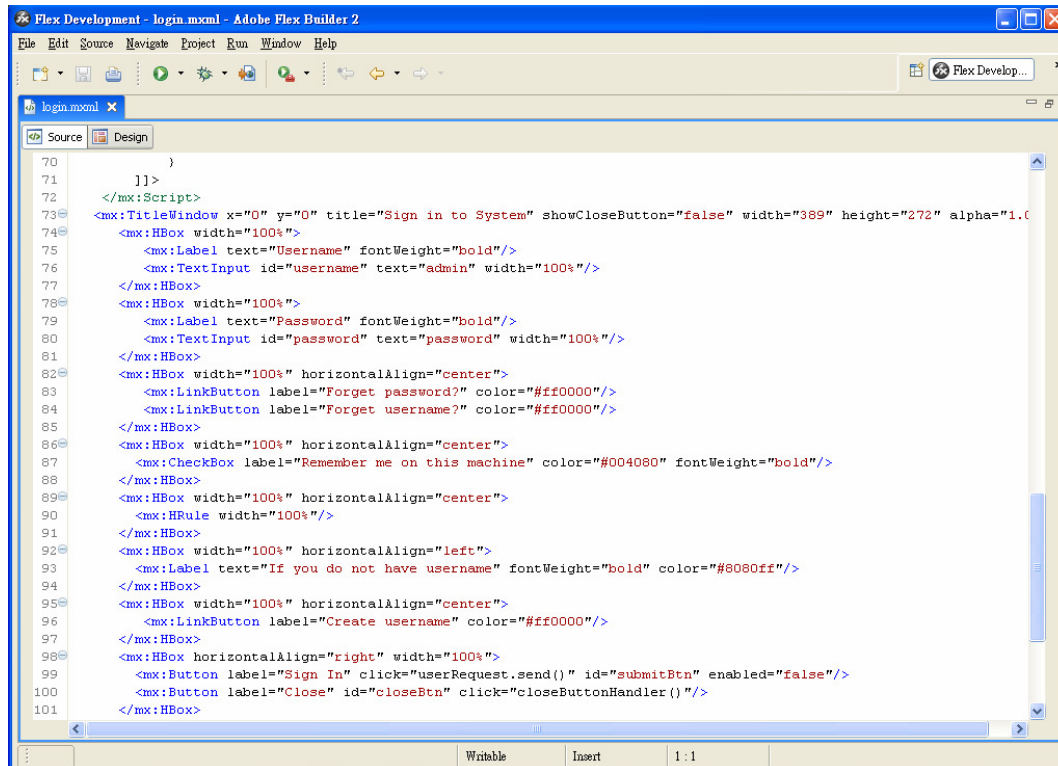


Figure 4-4 Source window can allow adjust the parameter of each Flash component

4.1. HTTP SERVICE

For data binding service, Table 4-2 is a simple example that shows how to use the HTTPService tag within the Flex Builder to load data from a third domain.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.macromedia.com/2005/mxml" xmlns="*">
<mx:Script>
<![CDATA[
import mx.controls.Alert;
private var url:String = "http://localhost/yourapp/getapp.php";

private function submitHandler ():void { }

]]>
```

```

</mx:Script>

<mx:HTTPService id="httpserv" url="example.php" useProxy="false" resultFormat="xml"
method="POST" result="httpserv_result(event);" fault="fault(event);"/>

<mx:Panel width="400" height="350" label="HTTPService Remote Data Test" >
<mx:TextArea id="outputField" width="100%" height="100%" />
<mx:Button label="Submit" id="mxSelectButton" click="submitHandler()"/>
</mx:Panel>
</mx:Application>

```

Table 4-2 HTTPService Tag for Flash object communicate data source through PHP gateway

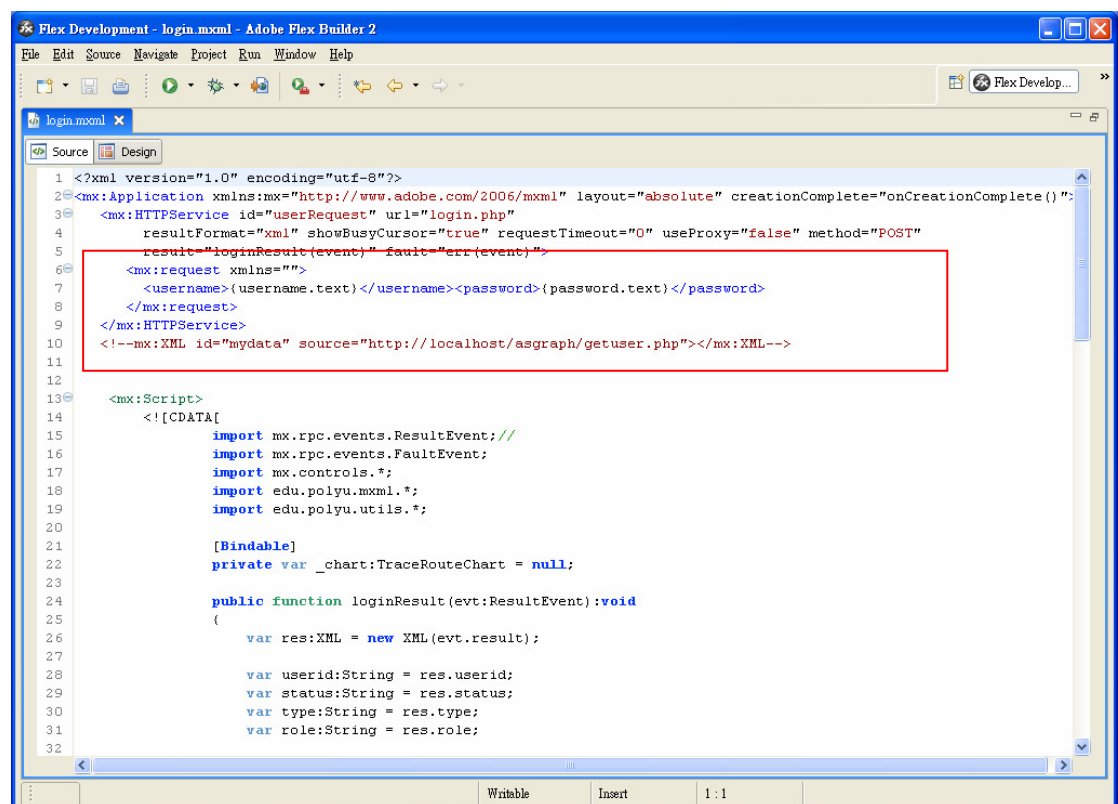


Figure 4-5 login mxml invoke HTTP Service for user authentication

Flex has an MXML tag named `mx:HTTPService` that can also be used to connect to a URL and get the data the provided by that URL. As with the `HTTPService` ActionScript class, you use the `URL` property to identify the resource to connect to and the `send` method to make the connection. You can embed an `mx:Request` tag inside the `mx:HTTPService` to specify the values for any parameters the web service

requires. The value for the mx:Request tag can be bound to a Flex control such as a bindable variable or combo box. Nested inside the mx:Request tag will be tags that have the name of the parameter and these tags will contain the values. For example:

```
<mx:request>
  <username>{_username}</username>
  <password>{_password}</password>
</mx:request>
```

Table 4-3 HTTP Service tag can specify any value of parameters for mx:Request tag

Alternatively, we could pass data to the URL web service is to create an Object that includes the name-value pairs and use this Object as the argument for the send method as we did with class HTTPService. Table 4-3 is code fragment for sending request for authentication using named object value.

```
private function submitHandler():void{
var param:Object = new Object();
param["username"] = _username;
param["password"] = _password;
httpserv.send(param);
}
```

Table 4-4 Named Object is created for send method of HTTPService class

When the data is returned to the Flex application, a result object is available. If you set the result Format property of the mx:HTTPService tag to "xml", the result object is recognized as XML for further processing in Table 4-5. You can bind the result XML or a node value to other Flex controls.

```

public function httpserv_result (evt:ResultEvent):void{
var res:XML = new XML(evt.result);

.....

}

```

Table 4-5 the function for return object from PHP

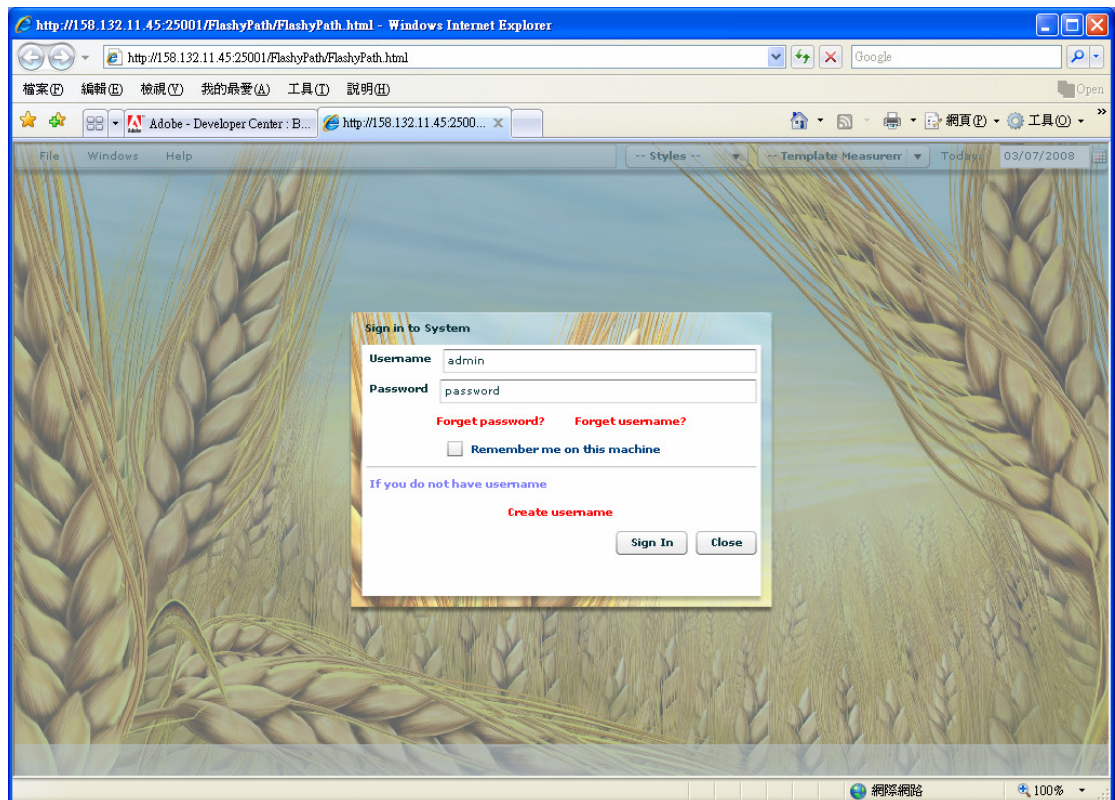


Figure 4-6 Main FlashyPath application with authentication function

4.2. ADOBE FLEX AND PHP

In PHP script, it will response the result on the request though HTTP services from FlashyPath that create XML-format result. Figure 4-7 is code fragment managing the authentication functions which the parameters of username and password are sent from FlashyPath. If users are registered, the USER_ID and role of end-users will be quoted and sent back to FlashyPath directly.

```

<?php
    header("content-type: text/xml");

    include("../include/config.php");

    if (isset($_post["username"])) {
        $username = $_post["username"];
    }
    if (isset($_post["password"])) {
        $password = $_post["password"];
    }

    $query = sprintf("select id as userid, type, role from users where username = '%s'
and password = '%s'", $username, $password);

    $result = mysql_query( $query );
    $num_rows = mysql_num_rows($result);

    ?>
<user>
<?php if ($num_rows > 0) {
    $user = mysql_fetch_object( $result );
    <username><![CDATA[ <?php echo $username ?>]]></username>
    <userid><![CDATA[ <?php echo $user->userid ?>]]></userid>
    <role><![CDATA[ <?php echo $user->role ?>]]></role>
    <status><![CDATA[true]]></status>
<?php
    }
?>
</user>
<?php mysql_free_result( $result ); ?>

```

Figure 4-7 PHP coding for user authentication functions

As a result, FlashyPath would receive returning XML looks something like this:

```

<user>
<username>admin</username>
<userid>1</userid>
<role>A</role>
<status>Y</status>
</user>

```

Table 4-6 the return object in form of XML containing user object

4.3. FLEX GRAPH

As mentioned before, FlashyPath design AS-level graph to construct the Internet Paths for measurement points. The Flex Visual Graph Library (flexvisgraphlib) package is used for the FlashyPath, which is Flash object that enables to create hierarchical graph for measurement points. As the library can be initialized as form of MXML, the parameter of library can be adjusted for requirement of autonomous system (AS) graph.

```
<un:VisualGraph id="vgraph" paddingBottom="0" backgroundColor="#ffffff" alpha="0.8"
    itemRenderer="org.un.flex.unComponentToolbox.renderers.PrimitiveIconRenderer"
    vgraphChanged="refreshUI()" resize="handleResize()"
    visibilityLimitActive="true" width="100%" height="100%"
    horizontalScrollPolicy="auto" verticalScrollPolicy="auto"
    borderStyle="solid" borderColor="#000000">
</un:VisualGraph>
```

Table 4-7 Flex Visual Graph library for creating hierarchical graph for measurement points

Having the Internet paths from database selected, the Actionscript will generate XML as the input of visual graph class. Each GraphXML file contains <Graph> tag of which is a list of <Node> tag and <Edge>. ID attributes of nodes are used as references in edge definitions, edges carry an attribute that defines directed edge in which two nodes are connected. Node attribute contains description of node in which whether node is host type.

```

<Graph>
| <Node id="vizgraph" name="" nodeType="" description="Flex Visual Graph Library" hostNode="true" sourceNode="false" targetNode="false"
  <Node id="132.10.159" name="132.10.159" nodeType="PrimitiveGear" description="132.10.159" hostNode="false" sourceNode="false" targetNode="false"
  <Edge fromID="vizgraph" toID="132.10.159" parentID="" targetID="" linkcolor="0xffff" srctoas="true" astoas="false" astotarget="false"
  <Node id="www.bre.polyu.edu.hk-80" name="www.bre.polyu.edu.hk-80" nodeType="PrimitivePolygon" description="www.bre.polyu.edu.hk-80"
  <Node id="www.se.cuhk.edu.hk-80" name="www.se.cuhk.edu.hk-80" nodeType="PrimitivePolygon" description="www.se.cuhk.edu.hk-80"
  <Node id="www.mtr.com.hk-80" name="www.mtr.com.hk-80" nodeType="PrimitivePolygon" description="www.mtr.com.hk-80"
  <Node id="www.cpce-polyu.edu.hk-80" name="www.cpce-polyu.edu.hk-80" nodeType="PrimitivePolygon" description="www.cpce-polyu.edu.hk-80"
  <Node id="www.hku.edu.hk-80" name="www.hku.edu.hk-80" nodeType="PrimitiveSquare" description="www.hku.edu.hk-80"
  <Node id="AS4616" name="AS4616" nodeType="AS_LEVEL" description="AS4616" parentID="132.10.159" lastNode="132.10.159"
    <Graph id="gp_AS4616" edgeDefault="undirected"/>
  </Node>
  <Edge fromID="132.10.159" toID="AS4616" parentID="132.10.159" targetID="www.bre.polyu.edu.hk-80:www.se.cuhk.edu.hk-80:www.mtr.com.hk-80:www.cpce-polyu.edu.hk-80:www.hku.edu.hk-80"
  <Edge fromID="AS4616" toID="www.bre.polyu.edu.hk-80" parentID="" targetID="" linkcolor="0xccc000" srctoas="false" astoas="false" astotarget="false"
  <Node id="AS3662" name="AS3662" nodeType="AS_LEVEL" description="AS3662" parentID="132.10.159" lastNode="AS4616"
    <Graph id="gp_AS3662" edgeDefault="undirected"/>
  </Node>
  <Edge fromID="AS4616" toID="AS3662" parentID="132.10.159" targetID="www.se.cuhk.edu.hk-80:www.mtr.com.hk-80:www.cpce-polyu.edu.hk-80:www.hku.edu.hk-80"
  <Node id="AS3661" name="AS3661" nodeType="AS_LEVEL" description="AS3661" parentID="132.10.159" lastNode="AS3662"
    <Graph id="gp_AS3661" edgeDefault="undirected"/>
  </Node>
  <Edge fromID="AS3662" toID="AS3661" parentID="132.10.159" targetID="www.se.cuhk.edu.hk-80" linkcolor="0xccc000" srctoas="false" astoas="false" astotarget="false"
  <Edge fromID="AS3661" toID="www.se.cuhk.edu.hk-80" parentID="" targetID="" linkcolor="0xccc000" srctoas="false" astoas="false" astotarget="false"
  <Node id="AS9925" name="AS9925" nodeType="AS_LEVEL" description="AS9925" parentID="132.10.159" lastNode="AS3661"
    <Graph id="gp_AS9925" edgeDefault="undirected"/>
  </Node>
  <Edge fromID="AS3662" toID="AS9925" parentID="132.10.159" targetID="www.mtr.com.hk-80" linkcolor="0xccc000" srctoas="false" astoas="false" astotarget="false"
  <Edge fromID="AS9925" toID="www.mtr.com.hk-80" parentID="" targetID="" linkcolor="0xccc000" srctoas="false" astoas="false" astotarget="false"
  <Node id="AS9304" name="AS9304" nodeType="AS_LEVEL" description="AS9304" parentID="132.10.159" lastNode="AS9925"
    <Graph id="gp_AS9304" edgeDefault="undirected"/>
  </Node>

```

Figure 4-8 The Graph XML result object is data source of measurement points for Visual Graph library

When visual graph object has been initialized in the application, the last thing is the AS result object for drawing graph in Figure 4-9. The top of graph shows the source node which the Internet Path comes from. The bottom is the destination node the measurement results are monitored. Between the source and destination nodes, the AS-level node will be hierarchically drawn. The red line represents the measurement points to be examined from the end-users.

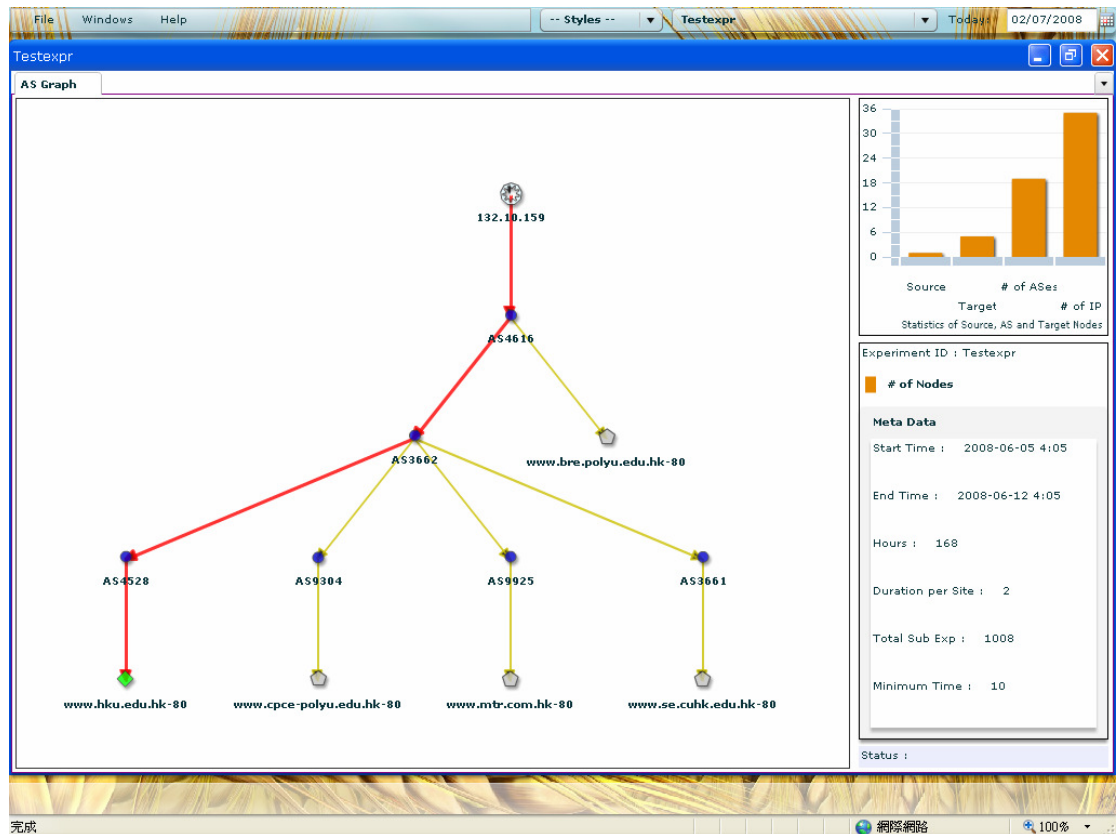


Figure 4-9 The AS graph using flexvisgraphlib package

For the measurement result on Internet Paths, the dialogue box in Figure 4-9 shows the selection with probeSize and responseSize. End-users may select all or some of the Internet paths for showing measurement results separately.

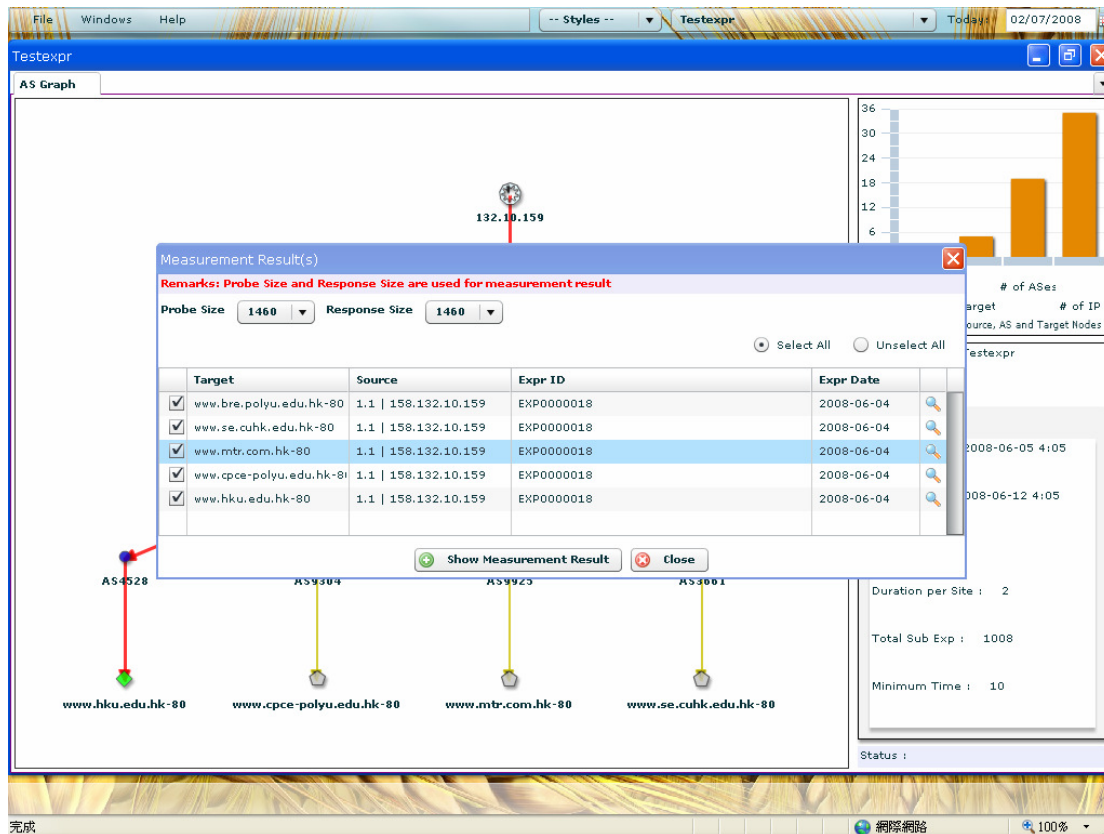


Figure 4-10 The measurement points for which monthly and daily measurement results are visualized

As measurement results are drawn as Time sensitive graph, Flex Chart provides an interface for developers binding the results. The following example is to demonstrate how Flex Chart is to be created in easier way.

1. Create a new file and save it as RRDChart.mxml in the project
2. At the beginning of the file, insert a basic application skeleton with a reference to the ActionScript file that is part of this tutorial's sample files:
3. Add a simple Line Chart control inside the application and connect it to a simple dataProvider object.
4. Save your application and load it in the browser. Your chart should look similar to Figure 4-11

Table 4-8 the procedures of building Flex Graph for measurement results

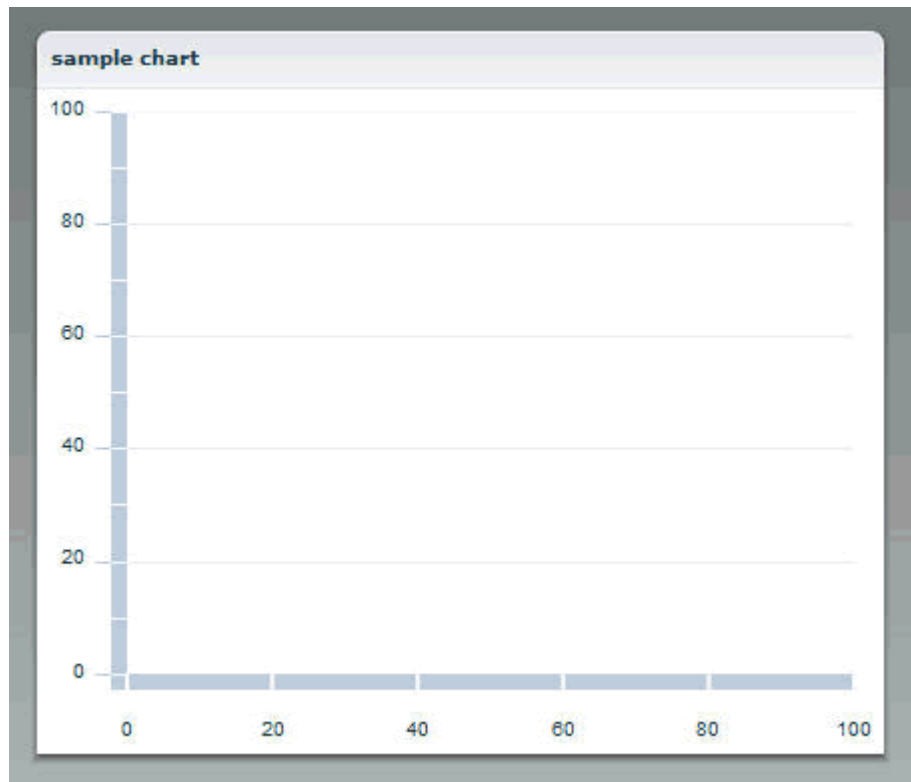


Figure 4-11 Create empty Flex Chart for measurement result

To map the column's position by category, place a `CategoryAxis` object in the chart's `horizontalAxis` property. Use a `CategoryAxis` object when you want to map a series of discrete categories evenly along the axis onscreen. The `CategoryAxis` object draws the categories from its own `dataProvider` property. In this tutorial, you use the same `dataProvider` property to generate the category labels and specify which field of the `dataProvider` property's content it should draw from by setting the `categoryField` property.

1. Add a `DateTimeAxis` object to the `horizontalAxis` property inside the Line Chart tag which can display time-sensitive data
2. Add a `LinearAxis` object to the `verticalAxis` property inside the Line Chart tag

```

<mx:horizontalAxis>
<mx:DateTimeAxis id="dtAxis" baseAtZero="false" displayLocalTime="true"
dataUnits="months"
dataInterval="1" parseFunction="dateParse" labelFunction="formatDateLabel" title="Time" />
</mx:horizontalAxis>
<mx:verticalAxis >
<mx:LinearAxis id="myLinearAxis" baseAtZero="false" autoAdjust="true"
alignLabelsToInterval="true" labelFunction="formaLabel" minimum="0"
maximum="100" interval="0.1"/>
</mx:verticalAxis>

```

Table 4-9 DateTimeAxis and VerticalAxis

A chart's relationship to its series is similar to a DataGrid control's relationship to its columns: just as the DataGridColumn control indicates which fields of the dataProvider property the grid should display, a series object indicates which fields of the dataProvider property to render as columns, lines, plots, and so on. In Figure 4-12, the monthly measurement results have been displayed by using the Flex chart component. The legend of each measurement points will be dynamically created that can be compared.

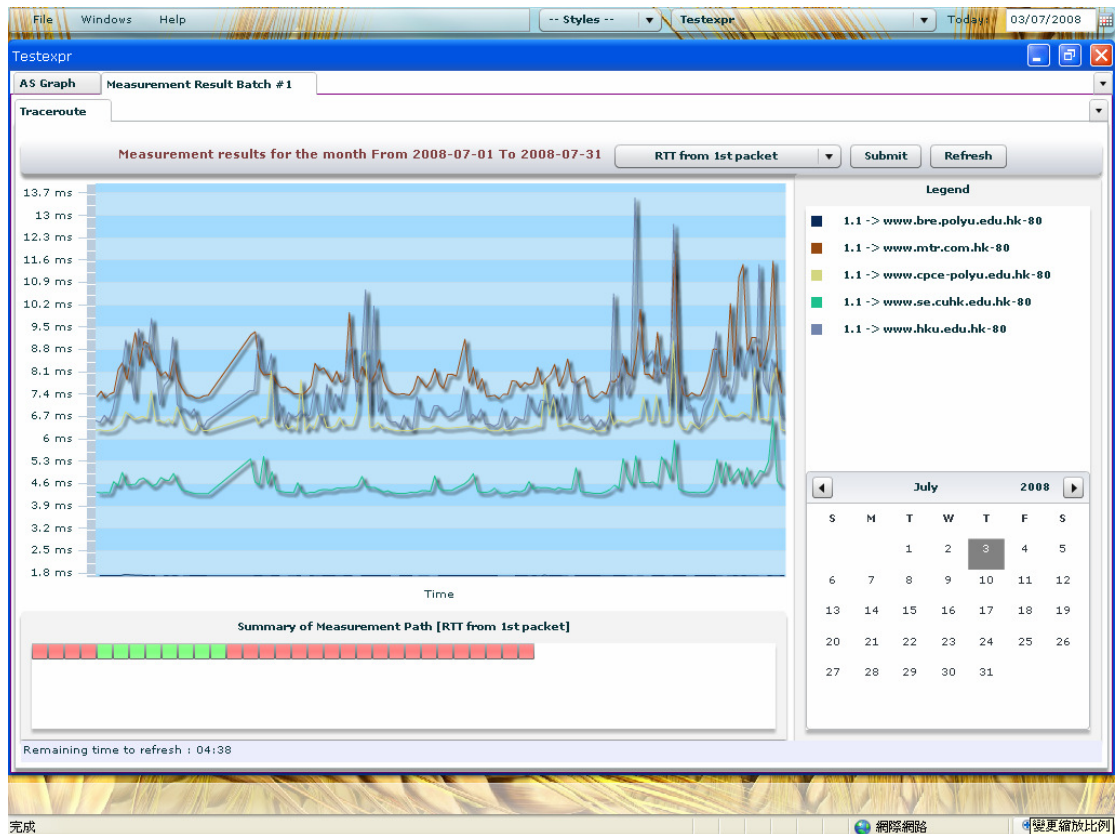


Figure 4-12 Monthly interactive timeseries panel for multiple measurement points

4.4. CSS & SYTLE

Take a look at some styling code I added to my final version of the application (if you've coded any CSS before, this will look familiar):

```

Panel
{
borderStyle: solid;
headerColors: #e7e7e7, #d9d9d9;
backgroundAlpha: 100;
paddingTop: 10;
}
List
{
paddingLeft: 10;
paddingRight: 10;
}

```

```
paddingTop: 10;  
paddingBottom: 10;  
}
```

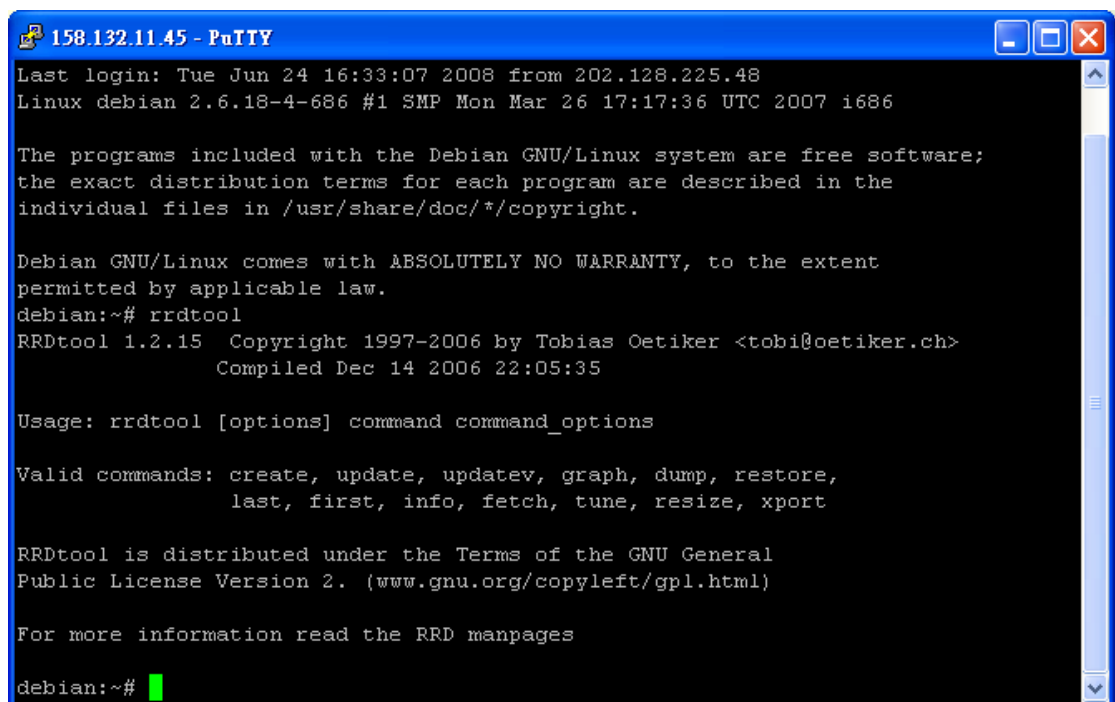
Table 4-10 Sample CSS and Style on Flex application

Flex's style system includes many different properties, and by creating your own assets inside Flash or Photoshop, you can change the appearance of your application entirely. Imagine changing the skin to match your company's branding and marketing materials. With a little work, you could even create a skin to match the default theme of Windows or Mac OS X. The possibilities are endless!

4.5. FLEX APPLICATION WITH RRDTOOL

To draw time series chart for daily measurement result that needs RRDTool and PHP.

Start by setting up server with PHP. Create a PHP, e.g., `t_select_rrd_data.php`, in your directory. It is note that RRDTool has been installed in the server. You may type the command “`rrdtool`” to check the version



```
158.132.11.45 - PuTTY
Last login: Tue Jun 24 16:33:07 2008 from 202.128.225.48
Linux debian 2.6.18-4-686 #1 SMP Mon Mar 26 17:17:36 UTC 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
debian:~# rrdtool
RRDtool 1.2.15 Copyright 1997-2006 by Tobias Oetiker <tobi@oetiker.ch>
Compiled Dec 14 2006 22:05:35

Usage: rrdtool [options] command command_options

Valid commands: create, update, updatev, graph, dump, restore,
                last, first, info, fetch, tune, resize, xport

RRDtool is distributed under the Terms of the GNU General
Public License Version 2. (www.gnu.org/copyleft/gpl.html)

For more information read the RRD manpages

debian:~#
```

Figure 4-13 RRDTool command line

The PHP script invokes system command on RRD like running in terminal. The RRDTool would export the measurement results for two minutes started from 2008-06-05 08:06:06 and finished on 2008-06-05 08:08:06 from RRD file 2008-06-30_www.bre.polyu.edu.hk-80_23.rrd with respect to probeSize and responseSize.

```
<?php
header("content-type: text/xml");
system('rrdtool xport --step 1 --start 1212624366 --end 1212624486
DEF:xx=rrd/1.1/1460/1460/2008-06-30_www.bre.polyu.edu.hk-80_23.rrd:ID_RTT_BY_PCAP_0:AVERAGE
XPORT:xx:"1.1-www.bre.polyu.edu.hk-80");
?>
```

Table 4-11 RRDTool xport command for daily measurement results.

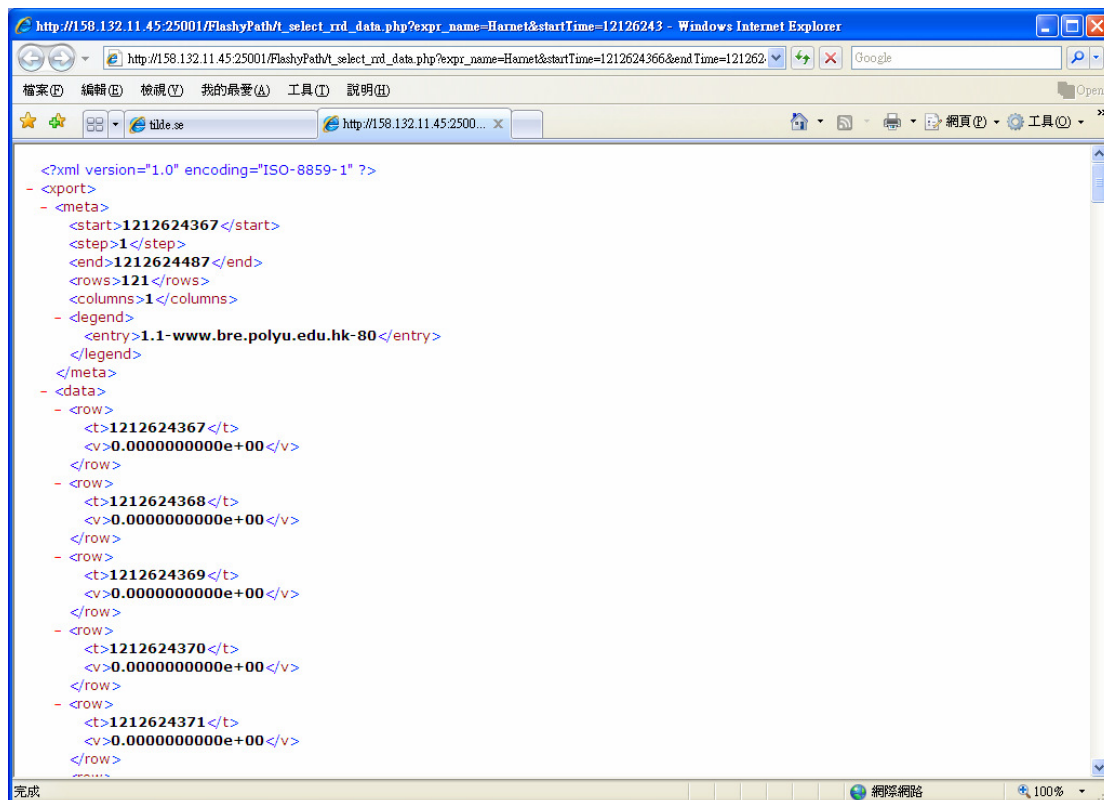


Figure 4-14 Reload the browser and have a nice XML-version of dataset.

For daily measurement result for date 2008-06-06 has been created using RRDTool result record. The right-top side of screen shows measurement boxes showing the selection of time frame for user request. If the users click the period between 7:14 and 7:16, the timeseries chart for two-minute graph will be generated.

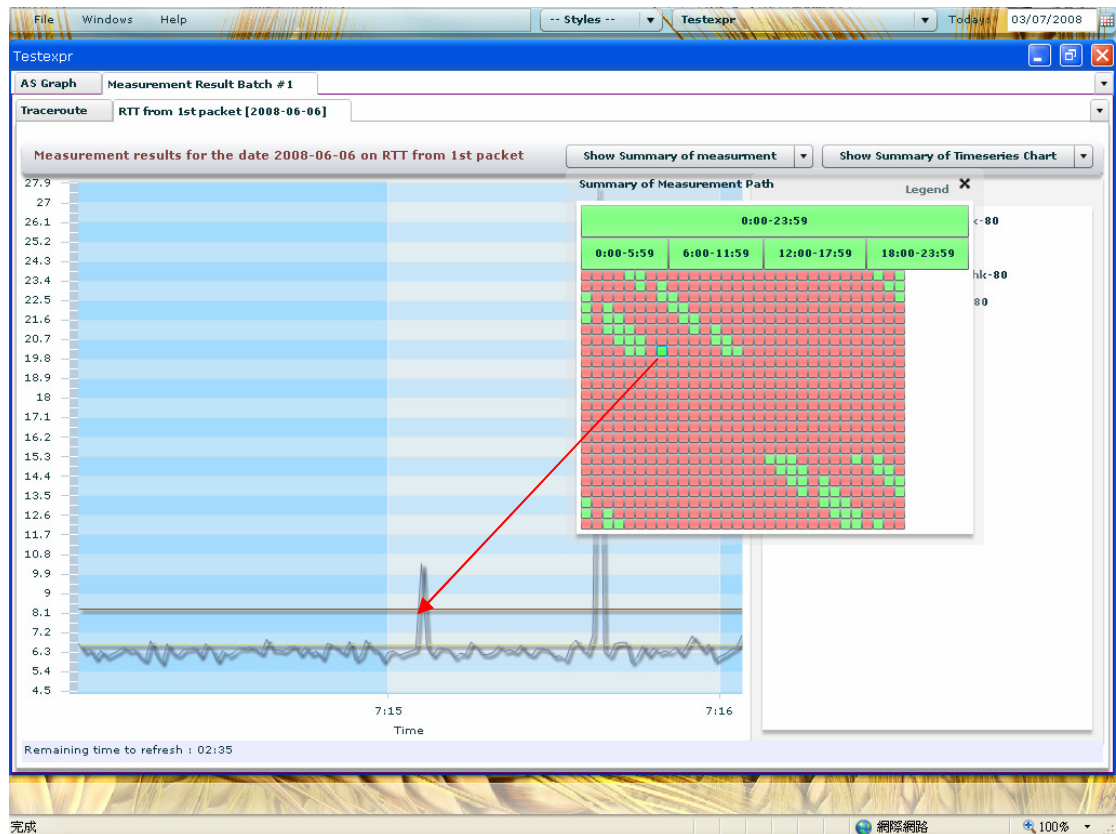


Figure 4-15 Using RRDTool export function, daily measurement result from round-robin database is created on the fly.

4.6. RRDTOOL (ROUND-ROBIN DATABASE)

RRDTool provides visualization features on displaying time-series measurement result using standard *xport* command. It is a tool which uses RRD storing time-series data in a round-robin format. Older data is average and finally moved out as newer data is inserted. The theory behind is that granular data is helpful when looking at recent events, but when looking at longer history, taking two-minute averages is more than adequate. Another advantage of RRDs is that old data is continually being removed. The size of the RRD does not change, which means that each measurement in the RRD has a timestamp associated with it.

Old items cannot be removed; you cannot enter items out of sequence. Based on the features, users should carefully determine how large RRD adequate for data storage. Experience has shown that small-sized RRD results recent data, such as 1 hour earlier, will be quickly removed while too large RRD may increase processing and searching time.

For example, PHP scripting file named `t_select_rrd_data` is to query RRD files and provide the value output of the files as a XML formatted file. It uses the `rrdtool xport` function; although it polls RRD files for the values it has some "features" which need to be implemented. However, the `rrdtool xport` function is restricted by default to 400 rows. This means that querying an RRD for a timespan that returns more than 400 rows (e.g. 5 minute Average RRA for a timespan of 2 weeks) `rrdtool` transforms the results to fit the 400 rows restriction (changing the stepping of the result, so that the results drop to 400). This behavior can be changed by using the `--maxrows` flag, by using a custom value for the `maxrows` flag and overriding the default `maxrows` setting of `RRDTool` we are able to produce the results with the exact points needed (instead of the reduced that the default settings provide). However, `FlashyPath` does not restrict maximum rows requirement. It is because the measurement results are kept in each second with 2 minutes.

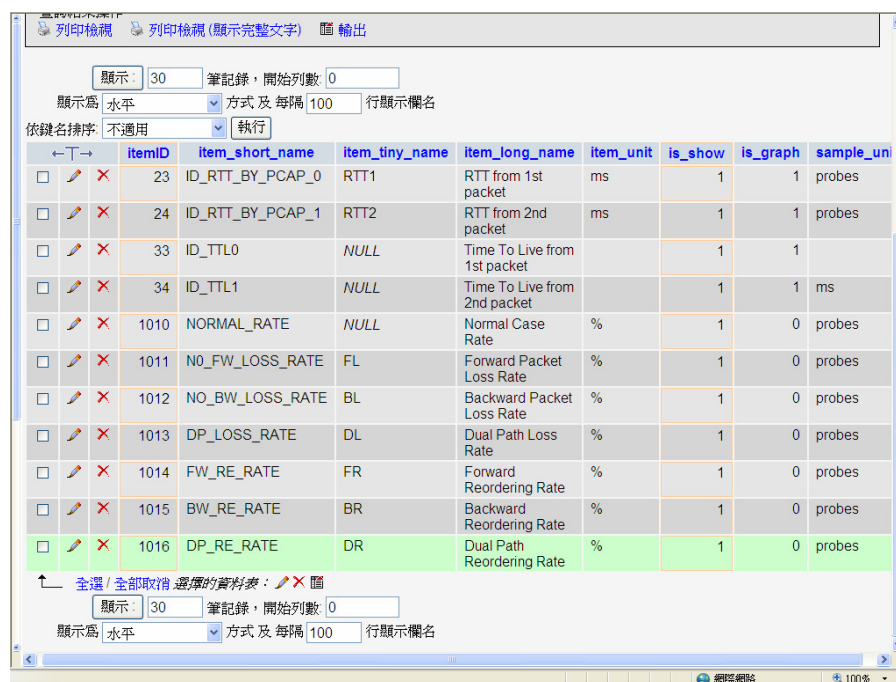
I would suggest modifying the RRDTool xport function inside the t_select_rrd_time.php to compute the rows needed for a provided RRDs file including the timespan (e.g. 2 minutes AVERAGE for the timespan of 1 month) and dynamically producing the needed --maxrows number.

5. LIMITATIONS

During the development of FlashyPath, we have encountered difficulties for usage of components on data-binding services and large volume of measurement results. Moreover, possible vulnerability issues of Flash-based application will be also outlined. This section would outline the difficulties we faced that can solve in the future days.

5.1. UNUSED RRD SPACE

FlashyPath is to read and visualize daily measurement results from RRD file of Internet paths. Not only does the objective alleviate workload of MySQL database, it also minimizes the database size and velocity of visualization between client and server. However, we have experienced that RRD file size will vary directly with the increasing number of analyze item and Internet paths. For the start date of each month, round-robin database will be newly created on performance metrics of two endpoints. In the project, ten analyze items are available for Internet path measurement and shown in Figure 5-1. The statistics has shown that the size is approximately computed as 20 Mbyte and 200 Mbyte for one Internet path on monthly and yearly respectively.



←T→	itemID	item_short_name	item_tiny_name	item_long_name	item_unit	is_show	is_graph	sample_un
<input type="checkbox"/>	23	ID_RTT_BY_PCAP_0	RTT1	RTT from 1st packet	ms	1	1	probes
<input type="checkbox"/>	24	ID_RTT_BY_PCAP_1	RTT2	RTT from 2nd packet	ms	1	1	probes
<input type="checkbox"/>	33	ID_TTL0	NULL	Time To Live from 1st packet		1	1	
<input type="checkbox"/>	34	ID_TTL1	NULL	Time To Live from 2nd packet		1	1	ms
<input type="checkbox"/>	1010	NORMAL_RATE	NULL	Normal Case Rate	%	1	0	probes
<input type="checkbox"/>	1011	NO_FW_LOSS_RATE	FL	Forward Packet Loss Rate	%	1	0	probes
<input type="checkbox"/>	1012	NO_BW_LOSS_RATE	BL	Backward Packet Loss Rate	%	1	0	probes
<input type="checkbox"/>	1013	DP_LOSS_RATE	DL	Dual Path Loss Rate	%	1	0	probes
<input type="checkbox"/>	1014	FW_RE_RATE	FR	Forward Reordering Rate	%	1	0	probes
<input type="checkbox"/>	1015	BW_RE_RATE	BR	Backward Reordering Rate	%	1	0	probes
<input type="checkbox"/>	1016	DP_RE_RATE	DR	Dual Path Reordering Rate	%	1	0	probes

Figure 5-1 List of available performance metrics

2008-06-30_clug.org.au-80_1.md	20,736,568	12.6.2008 下午 ...
2008-06-30_clug.org.au-80_1010.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_1011.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_1012.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_1013.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_1014.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_1015.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_1016.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_23.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_24.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_33.md	20,736,568	13.6.2008 上午 ...
2008-06-30_clug.org.au-80_34.md	20,736,568	13.6.2008 上午 ...
2008-06-30_cptra.hk-80_1.md	20,736,568	12.6.2008 下午 ...
2008-06-30_cptra.hk-80_1010.md	20,736,568	13.6.2008 上午 ...

Figure 5-2 Physical size of each Internet path on analyze item

On the other hand, the number of RRD file largely depends on number of Internet path that is to be measured. Therefore, the total capacity of RRD will be progressively increased for previous and current measurement results. Table 5-1 estimates the file size of RRD keeps growing when the number of analyze items increased.

	10	15	20
Monthly	200M	300M	400M
Yearly	2400M	3600M	4800M

Table 5-1 The file size of RRD for performance metrics

Besides this, the current practice is to measure twenty-four Internet path on every 2 minute sequentially in an hour. Therefore, it only contains 86400 ($2 * 60 * 24 * 30$ days) seconds measurement results on each RRD file. Recalling the RRD

create function, it however already create provide 25920000-second (60*60*24*30 days) space on each month. Therefore, less than 1/30 (86400/25920000) file space would only be used for measurement results and others are not used.

```
rrdtool create rrd/1.1/1460/1460/2008-06-30_www.bre.polyu.edu.hk-80_23.rrd
-s 1 -b 1212249600
DS:ID_RTT_BY_PCAP_0:GAUGE:2592002:0:10000
RRA:AVERAGE:0.5:1:2592003
```

5.2. TIME LAG

FlashyPath is originally to visualize the measurement results, which have been collectively updated and computed, that exist time lag between collection and visualization, incurring delays around ten minutes. ISP operators may only monitor the performance metrics of Internet path from last update time that affects the operation of diagnosis purposes. Most Internet monitoring systems are streaming for any probing results from existing tools, such as Ping, Traceroute, and display on the output screen. However, it would not ensure the data integrity and avoid error-free data because the tools cannot verify the measurement results instantaneously on the fly. Therefore, the time gap seems to act as buffer for FlashyPath which has enough adequate resources for computation of measurement results. However, it would determine wrong diagnosis decisions for Internet

paths because of synchronization issues.

5.3. PHP INTERFACE BETWEEN FLASHYPATH AND SQL DATABASE

In order to retrieve large volume of measurement results on the fly, FlashyPath transmits the XML result data through PHP interface, which directly connect to the MySQL and round-robin database. Using POST/GET method, FlashyPath sends request through HTTPService and invoke the PHP interface, which read the database and create XML result object. Therefore, in Figure 5-1, FlashyPath considers data binding over PHP interface that restricts the URL locator in absolute mapping

```
<mx:HTTPService id="as_graph_item_serv"
url="http://10.20.30.40/FlashyPath/t_as_graph_item.php" useProxy="false" resultFormat="xml"
showBusyCursor="true" method="POST" result="as_graph_item_result(event);"
fault="as_graph_item_fault(event);"/>
```

Table 5-2 absolute URL locator if FlashyPath serves as standalone tool

On the other hand, large-sized measurement results harm FlashyPath stands Idle. During the development, FlashyPath sends one-month data retrieval request to the server, where XML result is to be constructed according to the response from MySQL server. If the multiple requests send continuously, status of database

becomes busy while XML is constructing. Therefore, FlashyPath needs awaiting the response from server but server keeps busy. Consequently, FlashyPath becomes white screen wait too long to refresh on the browser. The shortcoming also appears on round-robin database, which shows daily measurement results.

5.4. SQL INJECTION

PHP interface between FlashyPath and database as backend infrastructure may be susceptible to SQL injection. During the development, it is found that FlashyPath visualizes incorrect measurement result if the SQL query is incorrect. FlashyPath now sends HTTP request in plain text by POST method when I accidentally pass wrong URL parameters. Therefore, it is further thought that attackers can inject SQL by terminating the intended SQL statement with the single quote character followed by a semicolon character to begin a new command, and then executing the command of their choice.

If attacker amend the exprID as following value

```
1; DROP TABLE d_to_graph_item --
```

Table 5-3 Drop table SQL command if query value of FlashyPath request is changed

This results in the following statement being submitted to the database for execution.

Consequently, the table d_to_graph_item will then be permanently removed by the

amended SQL query shown on Table 5-3. It notes that the double dash (--) denotes a SQL comment and is used to comment out any other characters added by the programming, such as the trailing quote.

```
SELECT start_time, end_time, stat_avg FROM v_to_graph_item_23 where  
date_format( from_unixtime( start_time ) , '%Y-%m-%d' ) = '2008-08-06' AND  
exprID = 1; DROP TABLE d_to_graph_item --‘
```

Table 5-4 The SQL query will be amended that table is permanently removed.

5.5. TCP SOCKET VULNERABLE

Flash application would be vulnerable if the TCP socket connection is alive. It is because plug-in does not retrieve FlashyPath directory from the network. Instead, the browser downloads the application and spawns Flash, transferring origin by host name. When the attackers crack the application, which attempts to open a socket, Flash does a second DNS resolution and would pin to the target's IP address. FlashyPath restrict to employ URLLoader class for the loading image and data because the class is not vulnerable to attacks which the browser to request the URL. However, the Socket class on most application could still be used to read and write on arbitrary TCP sockets. Figure 5-1 has shown that Flash player version 9 stands high vulnerability on network attack.

Vulnerability	Impressions
Flash 9	86.9%
LiveConnect	24.4%
Java+Proxy	2.2%
Total Multi-Pin	90.6%

Figure 5-3 Percentage of impression of vulnerability (Source: adambarth⁹)

⁹ Experiment for dnrebinding.net Flash 9 advertisement on 2007

6. CONCLUSION & FUTURE WORK

In this paper, we have demonstrated FlashyPath developed as network monitoring system which can visualize measurement results on e2e topology on multimedia concepts. We have moreover designed methodologies on how Flash multimedia objects integrate for monitoring functions that help display large multiple measurement points on the web browsers. In order to accelerate design monitoring tools on measurement points, FlashyPath employs IP-to-ASN mapping task that translates IP address of traceroute result as AS-level nodes for measurement points. Visual graph components can then efficiently display monthly and daily measurement results between the source and destination nodes on continuous purpose. For the sake of processing large volume of measurement result, FlashyPath provides efficient visual chart components, which import XML-style data, that performance of Internet paths can be continually shown.

Experience has shown that FlashyPath can be efficient on e2e topology if interactive components for network monitoring data are well integrated.

Moreover, the design of monitoring tool is also important when economical and efficient advantages can be taken of. Therefore, one of the future improvement is to revamp RRD module for new measurement results, which can minimize the disk usage and capacity of Internet paths. For example, RRD file can eliminate the unused space of RRD file that increases fetch time. We may newly create RRD file for Internet path for 2-minute measurement results on demand and store the meta information on MySQL database. One the other hand, FlashyPath integrates with PHP interface is regarded as barrier, which restricts to develop interoperability feature. Through the interface, the result from domain database to XML object must be manually constructed. Moreover, the large-sized measurement results for multiple points simultaneously would bring FlashyPath become idle. One of the solutions is to strictly fragment the requests and time controlled, which each fragmented requests is sent in fixed period, say, 5 seconds. However, the alternative will be foreseeable that FlashyPath becomes slower on multimedia components. For each request, it is highly recommended that unique sequence number should be inserted that avoid out-of-order issue. Otherwise, the measurement results will be reordered that diminishes the performance of FlashyPath.

For sake of developing interactive network monitoring tool, FlashyPath may be inadequate model for ISP operators who can diagnose Internet paths that are efficiently informed. Therefore, the tool should be integrated more interactive model, such as radar graph, for visualization of Internet paths. Moreover, FlashyPath currently supports XML-style format on exchange of measurement results between client and server architecture. Therefore, it is highly recommended that interactive monitoring tool support portable data type, such as SOAP messages, for ease of data-binding feature, thus allowing the tool handle measurement results more efficiently.

7.APPENDIX

In the section, we would briefly illustrate how FlashyPath can be executed in your environment with following software packages.

7.1. INSTALL APACHE 1.2.2 WITH PHP 5.0

Apache 1.2.2 is an open-source HTTP server for modern operating systems including UNIX and Windows NT version. The binary distribution can be downloaded in official website¹⁰ in Figure 7-1 and install in the home directory. Having Apache been installed, PHP [48] can be integrated that enables to produce dynamic web pages. PHP is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. For FlashyPath, PHP is to create XML object of measurement results with user request and receive response from databases, and finally send back as data-binding services.

¹⁰ The latest version of Apache HTTP server can be downloaded from <http://httpd.apache.org/download.cgi>

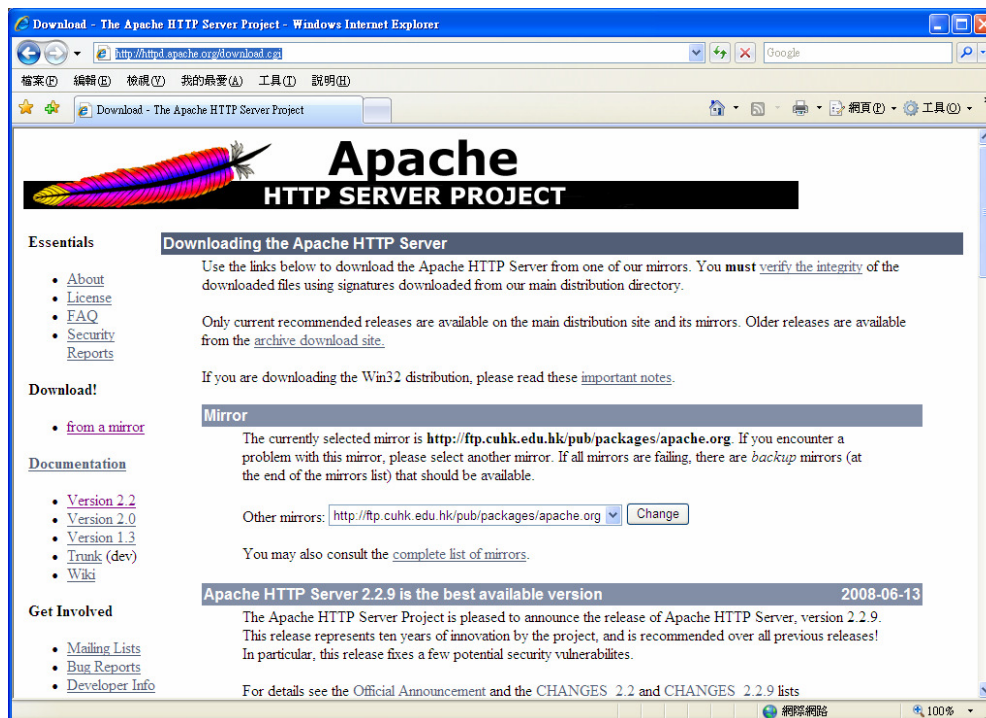


Figure 7-1 Binary distribution can be downloaded from Apache project web site



Figure 7-2 PHP official web site

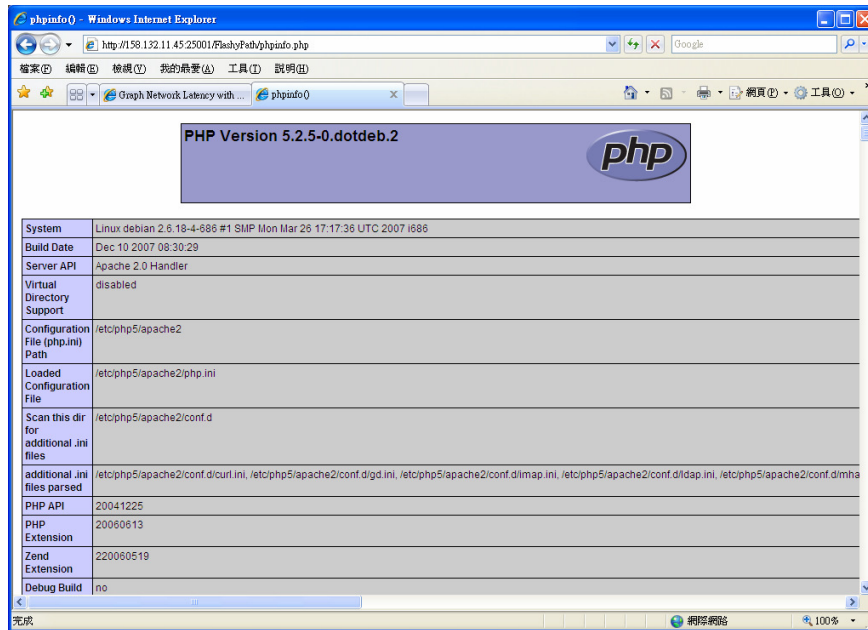


Figure 7-3 Install Apache 1.2.2 with PHP packages

7.2. INSTALL MYSQL AND RRDTOOL DATABASES

In order to store measurement results for FlashyPath, databases are required to be stored. MySQL is a relational database management system (RDBMS) worked on many platforms. Simple functions are included

1. Cross-platform support
2. Stored procedures
3. Triggers
4. Cursors
5. Updatable Views

Table 7-1 Sample MySQL functionalities

On the other hand, RRDTool is the industry standard, high performance data logging and graphing system for time series data. MySQL differs from RRDTool in command syntax on data management and illustrate sample for selection of measurement results from each database.

```
SELECT * from d_to_graph_item;
```

Table 7-2 SELECT statement for measurement results on MySQL database

```
rrdtool fetch rrdfile AVERAGE -r 900 1216537200 1216537800
```

Table 7-3 RRDTool fetch function



Figure 7-4 MySQL official web page

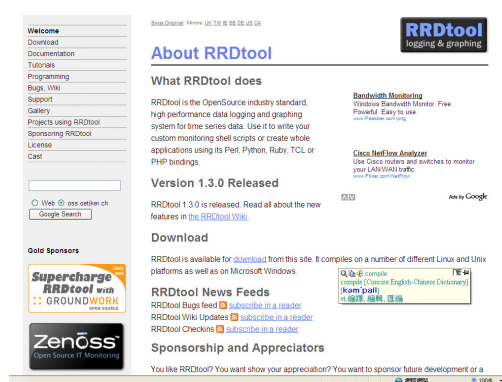


Figure 7-5 RRDTool official web page

Having RRDTool successfully installed in your environment, you may check by phpinfo.php and find that RRDTool extension for PHP environment is supported.

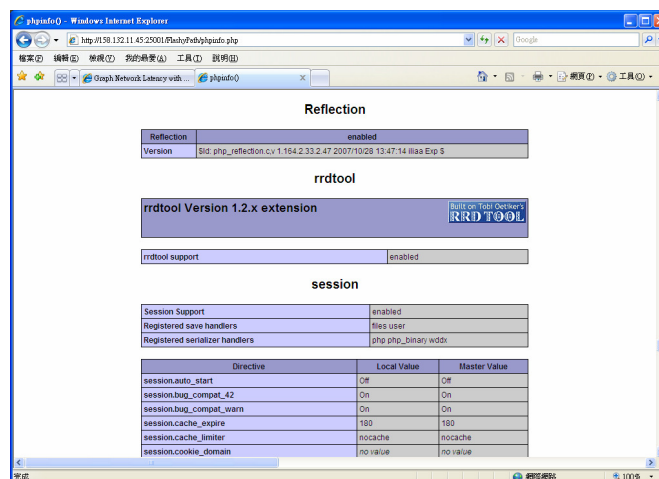


Figure 7-6 rrdtool support for PHP

7.3. INSTALL FLASHYPATH BINARY DISTRIBUTION

Create source directory in Apache directory in which FlashyPath can be executed

Flashypath-bin-1.0 zip is latest distribution of FlashyPath including the binary component with PHP scripting for wrapper programs and data-binding services.

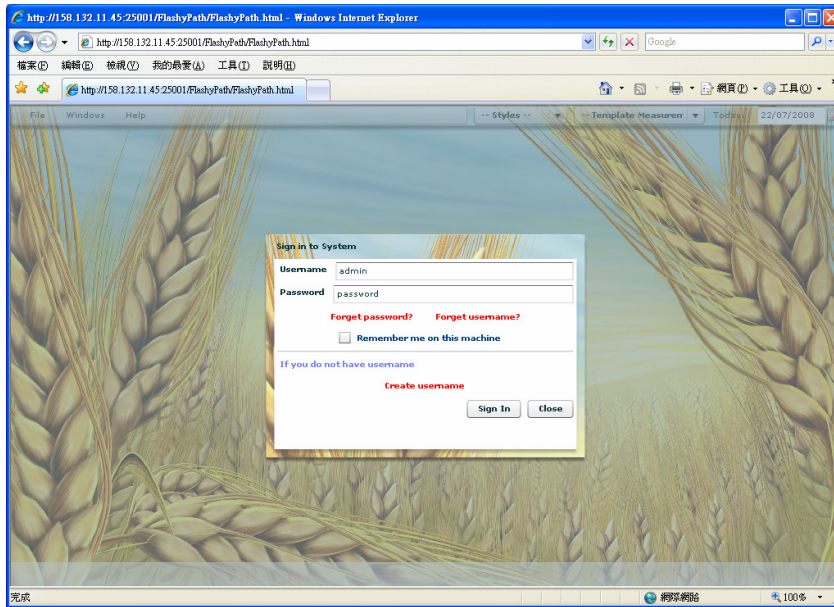


Figure 7-7 Flash component in HTML tag

7.4. INSTALL FLASH PLAYER 9.0 OR ABOVE

In order to execute FlashyPath object on the web, you should download flash player 9.0 in client machine. Flash player 9.0 is minimum requirement that FlashyPath can only be executed.

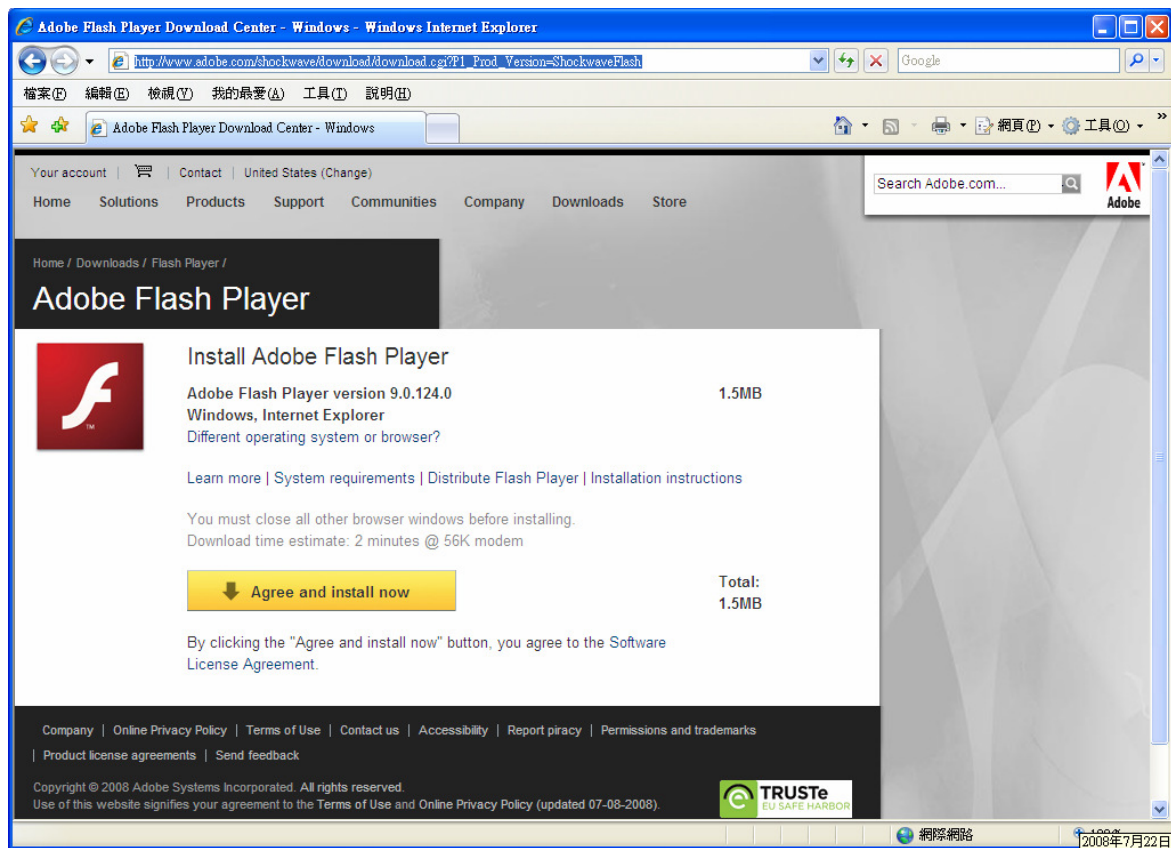


Figure 7-8 Flash Player 9.0 for executing FlashyPath

7.5. SOURCES CODE

7.5.1. TRACEROUTE MEASUREMENT RESULTS FOR

PERFORMANCE METRICS

```
if (count($res_expr) > 0){
    foreach ($res_expr as $expr) {
        $path = $dir . $sep . $planetLab["hostID"];
        if (is_dir($path) && is_readable($path)) {
            $cur_subExp_dir = $dir . $sep . $planetLab["hostID"] . $sep . "result" . $sep .
$expr["folder"] . "_result_" . $expr["curr_subExp"];
            $curDir = scanDirectories($cur_subExp_dir, array(".result"));
            foreach ($curDir AS $file){
                list($h, $var1, $www, $asgraph, $pid, $result, $result_dir, $probing,
$result_file) = split("/", $file);
                list($date, $result_id, $counter) = split('_', $result_dir);
                list($probeSize, $responseSize) = split('_', $probing);
                list($date, $destip, $inst) = split('_', $result_file);
            /* -----
            Get planetlab source id
            ----- */
            $sql = sprintf("SELECT s.expr_source_id, s.expr_ID, s.hostID, s.hostname FROM
d_expr_source s, d_expr e WHERE s.expr_ID = e.id and e.exprID = '%s' and s.hostID = '%s'",
$curr_exp, $pid);
            $expr_source_id = db_fetch_cell($sql, "expr_source_id");
            /* -----
            Get destination id
            ----- */
            $sql = sprintf("SELECT d.expr_ID, d.expr_destip_id , d.expr_destip, d.expr_ip FROM
d_expr_destip d, d_expr e WHERE d.expr_ID = e.id and e.exprID = '%s' and d.expr_destip = '%s'",
$curr_exp, $destip);
            $expr_destip_id = db_fetch_cell($sql, "expr_destip_id");
            $fp = fopen($file, 'r') or exit("Unable to open file!");
            $results = array();
            while (!feof($fp)){
                $s = fgets($fp);
```

```

        if ($s != ""){
            $results[] = $s;
        }
    }
    fclose($fp);

    $result = array(); // $result[$type][$time][$count] = value
    $result_time = array(); // $result_time[$type] = time
    foreach ($results AS $res){
        $data = split(',', $res); // 0 - time, 1 - eventid, 2 - value
        if (!isset($data[1]) || !isset($data[2])){
            continue;
        }
        $sec = floor($data[0]);
        // check event id
        if (checkExist($TypeID_list, $data[1], "itemID")){
            if (!isset($result[$data[1]][$sec])){
                $result[$data[1]][$sec][0] = $data[2];
                if (!isset($result_time[$data[1]])){
                    $result_time[$data[1]] = array();
                }
                array_push($result_time[$data[1]], $sec);
            }else{
                array_push($result[$data[1]][$sec], $data[2]); // insert to the end
            }
        }else{
            handle_special_type($result, $result_time, $data);
        }
    }
    aggregate_special_case($result, $result_time);

    //-----
    // Each item
    //-----
    foreach ($TypeID_list as $type){
        if (!isset($result[$type["itemID"]]) || !isset($result_time[$type["itemID"]])){
            continue;
        }
    }

```

```

sort($result_time[$type["itemID"]]);
reset($result_time[$type["itemID"]]);
$dates = getDateV($date);

$range = extract_trial($result_time[$type["itemID"]]);

//for a combined graph of this type
$type_length = 0;
$type_start = 0;
$type_end = 0;

$rrdFile_type = $dates["format"] . "_" . trim($destip) . "_" . $type["itemID"] ;
$cmd_type = "rrdtool update " . DIR_RRA . $sep. $pid . $sep . $probeSize . $sep .
$responseSize . $sep . $rrdFile_type . ".rrd ";
$stat_type = cal_statistics($result[$type["itemID"]], $result_time[$type["itemID"]],
in_array($type["itemID"], $type_multiple), in_array($type["itemID"], $type_percent),
in_array($type["itemID"], $type_ignore_zero), in_array($type["itemID"], $type_ignore_negative));

$vrule = "";

$trial_count = 0;
$rra_value_type_count = 0;

foreach ($range as $trial){
    //----- create rrd file
    if ($type_start == 0){
        $type_start = $trial["start"] - 1;
    }

    $rra_value_count = 0;
    $trial_count++;

    //Create the period separator, use blue color to indicate every 5 period
    if ($vrule != "")
        $vrule .= ',';
    $vrule .= ($type_length + $type_start + $expect_expr_length + 1);
    if (($trial_count % 5) == 0){
        $vrule .= '#00CCFF';
    }else{

```

```

        $vrule .= '#EDD3D5';
    }

    if (!isset($trial["keys"])){
        $cmd_type .= ($type_length + $type_start + 1) . ':0'; // set null period as 0
        $type_length += $expect_expr_length + 1;
        $cmd_type .= ($type_length + $type_start - 1) . ':0'; // set null period as 0
        continue;
    }

    $rrdFile = $update . " " . trim($destip) . " " . $type["itemID"] . " " . $trial_count;
    $rrd_create = array("filename" => (DIR_RRA . $sep . $pid . $sep . $probeSize . $sep .
$responseSize . $sep . $rrdFile . ".rrd"), "step" => 1, "start" => ($trial["start"] - 1));
    $rrd_create["ds"][0] = array("name" => $type["item_short_name"], "type" => "GAUGE",
"length" => ($expect_expr_length + 2), "min" => 0, "max" => 10000000);
    $rrd_create["rra"][0] = array("type" => "AVERAGE", "step" => 1, "length" =>
($expect_expr_length + 3));
    create_rrd_dbase($rrd_create);
    $cmd = "rrdtool update " . DIR_RRA . $sep . $pid . $sep . $probeSize . $sep . $responseSize .
$sep . $rrdFile . ".rrd ";

    $type_end = $trial["end"];
    $stat = cal_statistics($result[$type["itemID"]], $trial["keys"], in_array($type["itemID"],
$type_multiple),

    in_array($type["itemID"], $type_percent), in_array($type["itemID"], $type_ignore_zero),
in_array($type["itemID"], $type_ignore_negative));

    foreach ($trial["keys"] as $time){
        if ($time-$trial["start"] > $expect_expr_length){ // ignore those data outside expected
experiment length
            $result[$type["itemID"]][$time] = -10000; // indicate the end of period
            break;
        }
        $row_count = 0;
        $total = 0;
        foreach ($result[$type["itemID"]][$time] as $row){
            if (in_array($type["itemID"], $type_multiple)){ //convert floating point number
like ms to integer
                $row *= 1000;

```

```

    }

    if (in_array($type["itemID"], $type_percent)){
        $row *= 100;
    }

    if (in_array($type["itemID"], $type_ignore_zero)){
        if (round($row) == 0){
            continue;
        }
    }

    if (in_array($type["itemID"], $type_ignore_negative)){
        if ($row < 0){
            continue;
        }
    }
}

// find out those values that > 3sd or <3sd and ignore it
if ($row > ($stat["average"] + ($stat["sd"] * 3)) || $row < ($stat["average"] -
($stat["sd"] * 3))){
    $content = "In file : " . $path . $each . " Key = " . $time . "\n" . "Type : " .
$type["itemID"] . " Average : " . $stat["average"] . " Standard deviation = " . $stat["sd"] . " value
= " . $row . "\n";

    if (fwrite($log_file, $content) === FALSE) {
        echo "Cannot write to log file";
        exit;
    }

    if (in_array($type["itemID"], $type_need_filter)){
        continue;
    }
}

if ($stat["max"] < $row){
    $stat["max"] = $row;
}

if ($stat["min"] > $row){
    $stat["min"] = $row;
}

```

```

        if ($stat_type["max"] < $row){
            $stat_type["max"] = $row;
        }
        if ($stat_type["min"] > $row){
            $stat_type["min"] = $row;
        }

        if (in_array($type["itemID"], $type_average)){
            $total += $row;
            $row_count++;
        }else{
            $total += $row;
        }
    }

    if (in_array($type["itemID"], $type_ignore_zero)){
        if ($total == 0){
            continue;
        }
    }

    if ($row_count > 0){
        $sec_value = ($total/$row_count);
    }else{
        $sec_value = $total;
    }

    if (in_array($type["itemID"], $type_integer)){
        $sec_value = round($sec_value);
    }

    $rra_value_count++;
    $cmd .= $time . ':' . $sec_value . ' '; // concat time and value into rrdtool update command
    $cmd_type .= ($time - $trial["start"] + $type_length + $type_start) . ':' . $sec_value . ' ';
}

$type_length += $expect_expr_length + 1;
if ($rra_value_count == 0){

```

```

        continue;
    }
    $rra_value_type_count += $rra_value_count;

    cal_special_type($result, $type["itemID"], $trial["keys"], $stat);
    $hrule = "";

    // put the period graph detail into mysql and input data into rrd format file
    $sql = "insert into d_to_graph_item (destID, sourceID, probeSize, responseSize,
itemID, graph_start_time, graph_end_time, start_time, end_time, rra_location, image_location,
instance_no, expr_ID, stat_max, stat_min, stat_avg, stat_sd, num_of_sample, num_total,
other_comment, hrule) values (" . $expr_destip_id . ", " . $expr_source_id . ", " . $probeSize . ", " .
$responseSize . ", " . $type["itemID"] . ", " . $trial["start"] . ", " . ($trial["start"] +
$expect_expr_length) . ", " . $trial["start"] . ", " . ($trial["start"] + $expect_expr_length) . ", "" .
$rrdFile . ".rrd", "" . $rrdFile . ".png", 0, "" . $curr_exp . "", " . $stat["max"] . ", " . $stat["min"] . ", " .
$stat["average"] . ", " . $stat["sd"] . ", " . $stat["count"] . ", " . $null_value[$trial_count] . ", "", "" .
$hrule . "")";

    shell_exec($cmd);
    mysql_query($sql);
}

$name = $dates["format"] . " _ " . trim($destip) . " _ " . $type["itemID"] ;

cal_special_type($result, $type["itemID"], $result_time[$type["itemID"]], $stat_type);
$hrule = "";

// create horizontal lines on rrd graph
if (in_array($type["itemID"], $type_integer)){
    $hrule = round($stat_type["average"]) . '#00FF66:Average,' .
round($stat_type["median"]) . '#FFFF00:Median,' .
round($stat_type["lq"]) . '#0000FF:Lower-quartile,' . round($stat_type["uq"]) .
'#FF00FF:Upper-quartile';
} else {
    $hrule = ($stat_type["average"]) . '#00FF66:Average,' . ($stat_type["median"]) .
'#FFFF00:Median,' . ($stat_type["lq"]) . '#0000FF:Lower-quartile,' . ($stat_type["uq"]) .
'#FF00FF:Upper-quartile';
}

```

```

// update statistical information in MYSQL database

$sql = "insert into d_to_graph_item (destID, sourceID, probeSize, responseSize,
itemID, start_time, end_time, graph_start_time, graph_end_time, rra_location, image_location,
instance_no, stat_max, stat_min, stat_avg, stat_sd, stat_lq, stat_uq, stat_median, num_of_sample,
num_total, other_comment, vrule, hrule, source_file,backup, expr_ID) values (" . $expr_destip_id .
", " . $expr_source_id . ", " . $probeSize . ", " . $responseSize . ", " . $type["itemID"] . ", " .
$type_start . ", " . ($type_start + $type_length) . ", " . $type_start . ", " . $type_end . ", " . $fname .
".rrd', " . $fname . ".png', 0, " . $stat_type["max"] . ", " . $stat_type["min"] . ", " .
$stat_type["average"] . ", " . $stat_type["sd"] . ", " . $stat_type["lq"] . ", " . $stat_type["uq"] . ", " .
$stat_type["median"] . ", " . $stat_type["count"] . ", 0, " . "From " . date('Y/m/d G.i.s O', $first) . "
To " . date('Y/m/d G.i.s O', $last) . ", " . $vrule . ", " . $hrule . ", " . addslashes($result_file) .
"',N',' . $curr_exp . "')";

if ($debug){
    //print "\n [update_result_data.php] DEBUG : sql = $sql\n";
}

// update rrd file with data
shell_exec($cmd_type);
mysql_query($sql);
}
unset($nall_value);
unset($result);
unset($result_time);
}
}
}
}

```

8. REFERENCES

1. Adobe's Flex SDK <http://www.adobe.com/products/flex/>
2. Adobe Flex http://en.wikipedia.org/wiki/Adobe_Flex
3. Asgarit, P. Trimintzios, M. Irons, G. Pavlou, R. Egan, S. V. D. Berghe, A Scalable Real-time Monitoring System for Supporting Traffic Engineering
4. Ahluwalia A K, Jonker P, Young I T, 2000, An interactive image processing course for the web, Proceedings of First International conference on Image and Graphics, pp. 589-593
5. Hussain, G. Bartlett, Y. Pryadkin, J. Heidemann, C. Papadopoulos, and J. Bannister. 2005, *Experiences with continuous network tracing infrastructure*, In Proceeding of the 2005 ACM SIGCOMM workshop on Mining Network Data, pages 185-190, 2005.
6. Donnet, P. Raoult, T. Riedman, M. Crovella, Efficient Algorithms for Large-Scale Topology Discovery, In Proc. ACM Sigmetrics. Jun. 2005
7. Courcoubetis and V. A. Siris, 1999, *Measurement and analysis of real network traffic*, University of Crete and Institute of Computer Science, Greece.
8. Plonka, 2000, Flowscan: A network traffic flow reporting and visualization tool. In LiSA '00: Proceedings of the 14th USENIX conference on System Administration, pages 205-318: USENIX Association, 2000.
9. N. Hu, 2006, Network Monitoring and Diagnosis Based on Available bandwidth Measuremnt, Carnegie Mellon University.
10. Flex Visual Graph Library from <http://code.google.com/p/flexvizgraphlib/>
11. F. Qi, J. Zheng, W. Jia, and G. Wang, Available Bandwidth Measurement Schemes over Networks, Central South University, China
12. Inferring AS-level Internet Topology from Router-Level Path Traces, in Proc. of SPIE ITCOM, 2001.
13. J. Alberi, T. Chen, S. Khurana, A. Mcintosh, M. Pucci and R. Vaidyanathan, 2001, Using Real-Time Measurements in Support of Real-Time Network Management, Applied Research at Telcordia Technologies, Inc.
14. J. Oberheide, M. Karir and D. Blazakis, 2006, VAST: Visualizing Autoonomous System Topology
15. N. Hu and P. Steenkiste, Emodis – An End-Based Network Monitoring and Diagnosis System, 2005 (<http://www.cs.cmu.edu/~hnn/thesis/>)

16. N. Brwonlee and K.C., Claffy, Internet Measurement, IEEE, 2004
17. perfSONAR (Performance focused Service Oriented Network monitoring ARchitecture) <http://www.perfsonar.net/>
18. T. Hansen, J. Otero, T. McGregor, H Braun, 2000, Active Measurement Data Analysis Techniques, University of Waikato, New Zealand
19. N. Nilsan and A. Ronen, 2001, Algorithmic mechanism design, Games and Economic behaviour, pages 166-196.
20. P. Sessini, 2005, Internet Traffic Measurement, Department of Computer Science, University of Calgary, Canada.
21. R. Caceres, N. Duffield, A. Feldmann, J. Friedmann, A. Greeberg, R. greer, T. Johnson, C. kalamaneek, B Krishnanam, F True, and J Merwe, 2000, Measurement and Analysis of IP Network Usage and Behavior, IEEE Communications Magazine, 38(5)
22. S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, 2004, Inferring TCP Connection Characteristics Through Passive Measurement, IEEE
23. V. Paxson, Strategies for Sound Internet Measurement, Berkeley, USA
24. Continuous online extraction of HTTP traces from packet traces
25. Visualizing Internet Topology at a Macroscopic Scale http://www.caida.org/research/topology/as_core_network
26. P. Saraiya, P. Lee, C. North, Visualization of Graphs with Associated Timeseries Data, Department of Computer Science, Virginia Polytechnic Institute an State University, USA
27. V. Paxson, [Strategies for Sound Internet Measurement](#)
28. W. Yurick, Visualizing NetFlows for Security at Line Speed, NCSA
29. Xiao Qing Zhu, Yu Jin Zhang, 2001, Wei Jin Liu, Evaluation and Comparison of Web Developmental Tools and Technology, Tsinghua University, Beijing
30. Y. Zhao, Y. Chen and D. Bindel, 2006, Towards Unbiased End-to-End Network Diagnosis, SIGCOMM
31. RRDTTool <http://oss.oetiker.ch/rrdtool>
32. Cytoscape: Analyzing and Visualizing Network Data <http://www.cytoscape.org>
33. SmokePing <http://oss.oetiker.ch/smokeping>
34. Cacti: The Complete RRDTTool-based Graphing Solution <http://www.cacti.net>
35. R. Govindan, H. Tangmunarunkit, 2000, Heuristics for Internet map discovery. In Proceeding of IEEE Infocom, Tel Aviv, Israel.
36. Skitter: Macroscopic Topology Measurements <http://www.caida.org/tools/measurement/skitter/>
37. B.Cheswick, H. Burch, 2000, Internet Mapping Project, <http://cm.bell-labs.com/who/ches/map>

38. IP-to-ASN Translation
<http://rainbow.mimuw.edu.pl/SR/prace-mgr/szymaniak/node35.html>
39. R. Siamwalla, R. Sharma, and S. Keshav, 2006, Discovering Internet Topology, Cornell network Research Group, Cornell University, Ithaca, NY
40. Neil Spring, 2004, Efficient discovery of network topology and routing policy in the Internet, PhD Thesis, University of Washington, 2004
41. B. Lowekamp, B. Tierney and L. Cottrell, 2003, Enabling Network Measurement Portability through a Hierarchy of Characteristics, IEEE
42. A. Ciuffoletti and Y. Marchetti, 2008, End-to-end Network Monitoring Infrastructure, INFN-CNAF
43. A. Phipps, 2005, Network performance monitoring architecture. Technical Report EGEEJRA4-TEC-606702-NPM NMWG Model Design, JRA4 Design Team, September.
44. EGEE Network Performance Monitoring - Diagnostic Tool
<http://www.egee-npm.org/dt/>
45. VisoNETUI
http://158.132.10.164:25001/cacti/plugins/visonetui/main_visonetui.php
46. D. Agarwal, J. González, G. Jin, B. Tierney, 2003, An Infrastructure for Passive Network Monitoring of Application Data Streams, Proceedings of the 2003 Passive and Active Monitoring Workshop
47. Multimedia Definition <http://en.wikipedia.org/wiki/Multimedia>
48. PHP <http://en.wikipedia.org/wiki/PHP>
49. A list of Interactive tools <http://www.vbns.net/stats/flows/html/level0/>
50. Rich Internet Applications: Extraordinary interactive experiences, Gartner RAS Core Research
51. L. Colitti, G. Battista, F. Mariani, M. Patrignani, M. Pizzonia, 2005 Visualizing Interdomain Routing with BGPlay, Journal of Graph Algorithms and Applications, vol. 9, no. 1, pp. 117–148
52. Z. Mao, J. Rexford Jia, W. Randy and H. Katz, 2003, Towards an Accurate AS-Level Traceroute Tool, SIGCOMM'03
53. Visualizing Internet topology at a macroscopic scale,
http://www.caida.org/analysis/topology/as_core_network/.
54. H. Chang, S. Jamin, and W. Willinger, 2001, Inferring AS-level internet topology from router-level path traces, In Processing of Workshop on Scalability and Traffic Control in IP Networks, SPIEITCOM Conference
55. R. Mahajan, D. Wetherall, and T. Anderson, Understanding BGP misconfigurations, in Proc. ACM SIGCOMM, August 2002.
56. Z. Mao, D. Johnson, J. Rexford, J Wang and R. Katz, 2004, Scalable and Accurate Identification of AS Level Forwarding Paths, IEEE

57. I. Khalifa, 2002, Characterization of the Internet at the AS Level, School of Engineering Science, Simon Fraser University
58. Z. Mao, J. Rexford, J. Wang, and R. Katz, 2003, Towards an accurate as-level traceroute tool, Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, pages 365–378, New York, NY, USA
59. F. Zhao, V. R. Vemuri, S. F. Wu, 2006, Interactive Informatics on Internet Infrastructure
60. J. Heer, S. Card, J. A. Landay, 2005, prefuse: a toolkit for interactive information visualization, ACM 1-58113-998-5/05/0004
61. B. Myers, S. Hudson, and R. Pausch, 2000, Past, Present, and Future of User Interface Software Tools, ACM Transactions on Computer-Human Interaction
62. P. Gestwicki and B. Jayaraman, Interactive Visualization of Java Programs, Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY, USA