Future Network Monitoring for IXPs

Saleem Bhatti  
<S.Bhatti@cs.ucl.ac.uk>

Felipe Huici  
<F.Huici@cs.ucl.ac.uk>

Networks Research Group
Department of Computer Science, UCL
http://nrg.cs.ucl.ac.uk/
Outline of talk

1. Problem space and requirements
   - John Souter
2. Networks Research Group at UCL
3. The RMF Architecture
4. Demo
5. Opportunities for collaboration
6. Current status and Next steps
7. Questions and Discussion
Problem Space and Requirements
Outline problem space [1]

- Many IXPs have similar monitoring requirements
- All have semantically similar tools
- The tools often differ in the presentation of information, e.g.:
  - different hardware logs for data source
  - “home brew” scripts for processing logs
  - different visualisation front ends
Outline problem space [2]

- This makes it tricky to:
  - share information directly
  - use common information for troubleshooting
  - make comparisons of multi-site data
  - perform analysis using multi-site data

- IXPs recognise this is a growing problem
Outline requirements

- Based on discussion + email:
  - John Souter and Saleem Bhatti
- Allow IXPs to share data easily
- Devise a system for:
  - representing similar data in a common format
  - allowing easy, **secure**, remote access to data
  - common APIs
  - still make use of the “normal” data/log files
  - still make use of the existing tool base if possible
Networks Research Group at UCL
Research Agenda

- Internet Architecture and Evolution:
  - networking in the large
  - routing
  - protocols
  - QoS
  - congestion control
  - high-speed networking
  - control and management

- Internet Applications:
  - multimedia

- Mobile and Wireless Networked Systems

- Practical, experimental, collaborative research
Examples of interesting research problems

- Interaction between BGP convergence process and route flap damping
- Large-scale effects of interaction between inter-domain and intra-domain routing
- Congestion control in the large:
  - synchronisation effects, stability, macro effects, etc.
- Denial of service:
  - what is happening? how do we spot it? effects on network?
- Traffic modelling and performance analysis:
  - topology vs. routing
The RMF Architecture for network monitoring of IXPs
All the solid-shaded parts are “standardised”. All protocols (except G-RP) are “standardised”.

G-RI  glue - resource independent part
G-RS  glue - resource specific part
G-RP  glue/resource protocol
P-GP  PoD/glue protocol
RMF   resource monitoring function
RR    real resource
PoD   processing of data
VF    visualisation function
V-PP  visualisation/PoD protocol
The story so far - Architecture

- Multi-site, configurable, **secure**, remote monitoring
- Modular system
- IXP and hardware-independent architecture
- Extensible:
  - uses existing back-end tools and scripts
- Scaleable through encapsulation:
  - a PoD can use other PoDs as back-ends - a **SuperPoD**
- **Secure:**
  - uses SSL, using X.509 certificates
  - mutual authentication between front-end and PoD
The story so far - Implementation

- Modular and platform independent
- Language independence - currently Java:
  - but could be python/Tk, perl/Tk, C++/Qt, ...
  - front-ends can be text-based, of course 😊
- Client / PoD independence
- PoD yields actual data, not just a graph
- Some new visualisation of data
- ‘Real deployment’ at LINX
Opportunities for collaboration
IXPs looking after the network

- Care of the network
- Different timescales:
  - different tools
  - different information
  - different actions
- A more unified, valuable view of the network:
  - not just individual points in the network
- Allow IXPs to help each other more
Spatial monitoring advantage [1]
Spatial monitoring advantage [2]
Temporal monitoring advantage [1]

Health monitoring
- delay
- throughput
- link status
- alarms

$t = \text{now}$
Temporal monitoring advantage [2]

- Health monitoring
- Emergency situations
  - fault management
  - error reporting
  - diagnostics
  - routing problems

$t = \text{now}$

time, t
Temporal monitoring advantage [3]

- Health monitoring
- Emergency situations

Post hoc analysis
- connectivity logs
- routing logs
- routing policy

$t = \text{now}$

$t = \text{time, } t$
Temporal monitoring advantage [4]

Health monitoring

Emergency situations

Post hoc analysis

Long term analysis
- routing changes
- traffic matrices
- topology information

$t = \text{now}$

$\text{time, } t$
Temporal monitoring advantage [5]

- **Health monitoring**
- **Emergency situations**
- **Pro-active tools**
  - routing changes
  - traffic changes
  - fault pre-emption
  - historical data
- **Post hoc analysis**
- **Long term analysis**

$t = \text{now}$

time, $t$
Current status and Next Steps
Current status

- Working prototype: architectural proof-of-concept
- Reasonably stable:
  - running at LINX since 27 August 2004
- Software engineering:
  - needs some tidying up
  - needs packaging (release end Jan 2005)
- Software will be released as open source:
  - can provide remote help with installation
- Need to build more functionality
- Architectural refinement
Next steps

- **Look at the routing information:**
  - ‘in-the-wild’ behaviour of routing
  - this will give us huge insights

- **Engage with IXP community:**
  - examine the problem space in more detail

- **Deploy the monitoring more widely:**
  - information from more of the Euro-IX network

- **Further development**
How can we make progress? [1]

Get involved!

- Join the monitoring deployment:
  - use the tools
  - we are happy to help with configuration of tools
- Provide feedback on use of tools
- Provide the “really interesting” data:
  - iBGP/IGP, BGP, other routing info such as policies
  - filtered packet traces
- Understanding of problems and requirements
- Contribute to the system
How can we make progress? [2]

- Get in touch with us:
  - S.Bhatti@cs.ucl.ac.uk  F.Huici@cs.ucl.ac.uk

- Current software available end Jan 2005:
  - set-up distributed monitoring across Euro-IX

- Set-up data feeds for routing information

- What do you need at your site to take part?
  - a modest linux box with Java, gcc/g++, fping
  - future: someway of accessing routing-packet exchanges (e.g. a log written to the linux box)
Questions and discussion
Additional Slides
Example: RTT Threshold RMF

VF

DISPLAYS

GUI.java

PoD

PoDThresholdRTT.java

PoD

ANALYSES

PARSES

GlueIPpingmon.java

G-RI

G-RS

G-RP

alarms.log

RR

Vector | line1 | line2 | line3 | ...
[0] DATE | DATE | DATE
[1] TIME | TIME | TIME
[2] ADDR | ADDR | ADDR
[3] COLOR | COLOR | COLOR

Vector | line1 | line2 | line3 | ...
[0] DATE | DATE | DATE
[1] TIME | TIME | TIME
[2] ADDR | ADDR | ADDR
[3] RTT | RTT | RTT

64.3.7.109

64.3.7.110

64.3.7.134

64.3.8.222

time

Example: RTT Threshold RMF
PoD Registry and PoD Init [1]

1. Client/VF needs to know what PoDs exist at a site:
   - need configuration info for client/VF
   - (PoD meta-data)

2. Need to start PoDs at site:
   - site-specific start-up configuration for PoDs
PoD Registry and PoD Init [2]

1. PoD registry reads local config
2. Local PoDs are instantiated
3. PoD Registry is updated with PoD info:
   • PoD type
   • PoD addr/port/proto
PoD Registry and PoD Init [3]

1. PoD Registry listens on “well-known” port
2. Client/VF contacts PoD:
   a. requests PoD meta-data
   b. PoD responds with info on all instantiated PoDs
3. Client/VF can then contact PoDs to complete RMF