Don’t Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations

Ryan Beckett (Princeton, MSR)
Ratul Mahajan (MSR)
Todd Millstein (UCLA)
Jitu Padhye (MSR)
David Walker (Princeton)
Configuring Networks is Error-Prone

~60% of network downtime is caused by human error

- Yankee group 2002

50-80% of outages from configuration changes

- Juniper 2008
Configuring Networks is Error-Prone

YouTube/Pakistan incident: Could something similar whack your site?

Configuring BGP properly is key to avoidance, 'Net registry official says

By Carolyn Duffy Marsan
Network World  |  Mar 10, 2008 1:00 AM PT

In light of Pakistan Telecom/YouTube incident, Internet registry official explains how you can avoid having your web site victimized by such an attack.

When Pakistan Telecom blocked YouTube's traffic one Sunday evening in February, the ISP created an international incident that wreaked havoc on the popular video site for more than two hours.

RIPE NCC, the European registry for Internet addresses, has conducted an analysis of what happened during Pakistan Telecom's hijacking of YouTube's traffic and the steps that YouTube took to stop the attack.

We posed some questions to RIPE NCC's Chief Scientist Daniel Karrenberg about the YouTube incident. Here's what he had to say:

How frequently do hijacking incidents like the Pakistan Telecom/YouTube incident happen?

Misconfigurations of IBGP (internal BGP, the protocol used between the routers in the same Autonomous System) happen regularly and are usually the result of an error. One such misconfiguration caused the Pakistan Telecom/YouTube incident. It appears that the Pakistan Telecom/YouTube incident was not an "attack" as some have labeled it, but a configuration error. (See Columnist Johna Till Johnson's take on the topic.)

What is significant about the YouTube incident?
Configuring Networks is Error-Prone

By Carolyn Duffy Marsan
Network World | Mar 10, 2008 1:00 AM PT

In light of Pakistan Telecom/YouTube incident, your web site victimized by such an international incident that weakened network availability.

When Pakistan Telecom blocked YouTube, it triggered a large international incident that weakened network availability.

RIPE NCC, the European registry for connectivity, said it was aware of the attack.

We posed some questions to RIPE NCC:

Here’s what he had to say:

How frequently do hijacking incidents happen?

Misconfigurations of IBGP (inter-BGP) are a regular occurrence, but misconfiguration caused the Pakistan Telecom/YouTube incident was not.

What is significant about the YouTube outage?

I think it’s significant that users and networks that connect through Time Warner were unable to reach the supply chain portal, indicating the issue was in the Time Warner network. In this case, it’s a brief service interruption while traffic routed through their other upstream BGP AS. However, the availability issues continued for the entire duration of the outage. I was surprised to see issues, so took a look at the path calculation view to figure out exactly where traffic was getting dropped. On the way to this site, normally, two locations (Prague and Dublin) would connect through the main router (Time Warner) to reach this supply chain portal, while the next one through AT&T (Figure 2).
Configuring Networks is Error-Prone

In light of Pakistan Telecom/YouTube incident, are you ready to protect your site?

Configuring BGP properly is not easy, and it can be error-prone. Time Warner Outages, for example, highlight the importance of having a robust configuration in place.

Time Warner Outages
The service started crapping out in a blink of an eye. Many ISPs, including many of those who have the same BGP routing configuration, experienced service disruption.

When Pakistan Telecom blocked YouTube/Google traffic, it created an international incident that widened the scope of the problem.

RIPE NCC, the European registry for IP and AS numbers, raised questions about the incident and the hijacking process.

We posed some questions to RIPE NCC, and here’s what they had to say:

How frequently do hijacking incidents happen?
Misconfigurations of IBGP (Interior Border Gateway Protocol) Autonomous System) happen regularly, and some configurations caused the Pakistan Telecom/YouTube incident.

What is significant about the YouTube/Google incident?
It is significant because it is an example of how misconfigurations can cause wide-scale disruptions. It also highlights the importance of having robust monitoring and control mechanisms in place.

China routing snafu briefly mangles Internet
Coup, not conspiracy

China Telecoms, AT&T, France Telecom, and Deutsche Telekom all lost internet traffic for 30 minutes, due to a routing configuration error.

The error was caused by a misconfiguration of BGP (Border Gateway Protocol) that affected multiple internet service providers.

China Telecoms, AT&T, France Telecom, and Deutsche Telekom all lost internet traffic for 30 minutes, due to a routing configuration error. The error was caused by a misconfiguration of BGP (Border Gateway Protocol) that affected multiple internet service providers.

China Telecoms, AT&T, France Telecom, and Deutsche Telekom all lost internet traffic for 30 minutes, due to a routing configuration error. The error was caused by a misconfiguration of BGP (Border Gateway Protocol) that affected multiple internet service providers.

The error was caused by a misconfiguration of BGP (Border Gateway Protocol) that affected multiple internet service providers. The error was caused by a misconfiguration of BGP (Border Gateway Protocol) that affected multiple internet service providers.

The error was caused by a misconfiguration of BGP (Border Gateway Protocol) that affected multiple internet service providers.
Configuring Networks is Error-Prone

YouTube/Pakistan incident interferes with your site?

Configuring BGP properly says

By Carolyn Duffy Marsan
Network World | Mar 10, 2008 1:00 AM PT

In light of Pakistan Telecom/YouTube’s recent incident that wreaked havoc on your web site victimized by such an attack, it’s important to ask:

When Pakistan Telecom blocked YouTube traffic a few weeks ago, it was part of an international incident that wreaked havoc on the Internet. The incident took place during Pakistan Telecom’s hijacking of domain names.

We posed some questions to RIPE NCC, the European registry for domain names during Pakistan Telecom’s hijacking incident. Here’s what he had to say:

How frequently do hijacking incidents occur?

Misconfigurations of IBGP (Interior Border Gateway Protocol) Autonomous System) happen regularly, causing Internet-wide outages. The Pakistan Telecom/YouTube incident was not an isolated case.

What is significant about the YouTube/Pakistan incident?

In this case, the hijacking lasted for 24 hours and affected approximately 10% of the Internet. This is significant because of the scale of the impact. Additionally, the hijacking was part of a larger cyber attack, which highlights the need for robust security measures.

What can the Internet community do to prevent such incidents?

To prevent such incidents, the Internet community can take several steps:

1. Implement robust security measures to prevent unauthorized access to infrastructure.
2. Regularly monitor and audit network configurations to detect and correct errors promptly.
3. Establish clear communication channels among network operators to quickly respond to incidents.
4. Encourage the adoption of best practices and standards, such as the RFC 6480 “Network Security Considerations.”

By taking these steps, the Internet community can work together to mitigate the risk of such incidents and maintain the Internet’s stability.

References:


Further Reading:

<table>
<thead>
<tr>
<th>Distributed</th>
<th>Centralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed</td>
<td>Centralized</td>
</tr>
<tr>
<td>Control Mechanism</td>
<td>Configuration</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Distributed</td>
<td>OSPF, RIP, BGP</td>
</tr>
<tr>
<td>Centralized</td>
<td></td>
</tr>
</tbody>
</table>

- **OSPF**: Scalability, Complexity
- **RIP**: Robustness
- **BGP**: Complexity
Fundamental Tradeoff?

Control Mechanism

Configuration

Distributed

Centralized

OSPF
RIP
BGP

Scalability
Robustness
Complexity

100,000s of lines of config

100,000s of lines of config
Fundamental Tradeoff?

- Distributed
- Centralized
- Configuration

- OSPF | Scalability
- RIP   | Robustness
- BGP   | Complexity

- Distributed
- Centralized
- Control Mechanism

- SDN | Scalability
-     | Robustness
-     | Complexity
Fundamental Tradeoff?

- **Configuration**
  - Distributed
  - Centralized

- **Control Mechanism**
  - Distributed
  - Centralized

- **OSPF**
  - Scalability
  - Robustness
  - Complexity

- **RIP**

- **BGP**

- **SDN**
  - Scalability
  - Robustness
  - Complexity

- **Ideal**
Propane Overview

Propane → Compiler → BGP Configs

Topology

Propane

Compiler

BGP Configs
Propane System

I) Language for expressing network-wide objectives with:

- Path **constraints** and **preferences** in case of failures
- Uniform abstractions for **intra**- and **inter**-domain routing
Propane System

2) Compiler for a purely distributed implementation
2) Compiler for a purely distributed implementation

- Generate **BGP** configs for each router
- Compiler guarantees **policy-compliance** for all failures
Example: A DC network with traditional configs

Goals

- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer1 over Peer2
- Prevent transit traffic between peers
Example: A DC network with traditional configs

Goals
- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer₁ over Peer₂
- Prevent transit traffic between peers

Configuration Attempt
- Don’t export from G, H to external
- Aggregate externally as GP
Example: A DC network with traditional configs

**Goals**
- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer\(_1\) over Peer\(_2\)
- Prevent transit traffic between peers

**Configuration Attempt**
- Don’t export from G, H to external
- Aggregate externally as GP
Example: A DC network with traditional configs

Goals

- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer\(_1\) over Peer\(_2\)
- Prevent transit traffic between peers

Configuration Attempt

- Don’t export from G, H to external
- Aggregate externally as GP
- \(X, Y\) block routes through each other
Example: A DC network with traditional configs

**Goals**
- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer\(_1\) over Peer\(_2\)
- Prevent transit traffic between peers

**Configuration Attempt**
- Don’t export from G, H to external
- Aggregate externally as GP
- X,Y block routes through each other
**Example: A DC network with traditional configs**

**Goals**

- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer\(_1\) over Peer\(_2\)
- Prevent transit traffic between peers

**Configuration Attempt**

- Don’t export from G, H to external
- Aggregate externally as GP
- X,Y block routes through each other
Example: A DC network with traditional configs

Goals
- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer₁ over Peer₂
- Prevent transit traffic between peers

Configuration Attempt
- Don’t export from G, H to external
- Aggregate externally as GP
- X, Y block routes through each other

Aggregation-Induced Black Hole!
Example: A DC network with traditional configs

Goals

- Local prefixes reachable only internally
- Global prefixes reachable externally
- Aggregate global prefixes as GP
- Prefer leaving through Peer\textsubscript{1} over Peer\textsubscript{2}
- Prevent transit traffic between peers
**Example: A DC network with Propane**

```plaintext
define Destination =
{GP1  =>  end(A)
 GP2  =>  end(B)
 LP1  =>  end(E)
 LP2  =>  end(F)
 true  =>  exit(Peer1 >> Peer2)}
```
Example: A DC network with Propane

`define` Destination =
{GP1 => end(A)
   GP2 => end(B)
   LP1 => end(E)
   LP2 => end(F)
   true => exit(Peer1 >> Peer2)}

`define` Locality =
{LP1 | LP2 => internal}
Example: A DC network with Propane

**define** Destination =
{GP1 => end(A),
  GP2 => end(B),
  LP1 => end(E),
  LP2 => end(F),
  true => exit(Peer1 >> Peer2)}

**define** Locality =
{LP1 | LP2 => internal}

**define** transit(X,Y) =
enter(X|Y) and exit(X|Y)
Example: A DC network with Propane

```plaintext
define Destination =
{GP1 => end(A)
GP2 => end(B)
LP1 => end(E)
LP2 => end(F)
true => exit(Peer1 >> Peer2)}

define Locality =
{LP1 | LP2 => internal}

define transit(X,Y) =
enter(X|Y) and exit(X|Y)

define NoTransit =
{true => !transit(Peer1,Peer2)}
```
**Example: A DC network with Propane**

\[
\text{define} \; \text{Destination} = \\
\{\text{GP1} \Rightarrow \text{end}(A) \}
\]

\[
\{\text{GP2} \Rightarrow \text{end}(B) \}
\]

\[
\{\text{LP1} \Rightarrow \text{end}(E) \}
\]

\[
\{\text{LP2} \Rightarrow \text{end}(F) \}
\]

\[\text{true} \Rightarrow \text{exit}(\text{Peer1} >> \text{Peer2})\}

\[
\text{define} \; \text{Locality} = \\
\{\text{LP1} | \text{LP2} \Rightarrow \text{internal} \}
\]

\[
\text{define} \; \text{transit}(X,Y) = \\
\text{enter}(X|Y) \; \text{and} \; \text{exit}(X|Y)
\]

\[
\text{define} \; \text{NoTransit} = \\
\{\text{true} \Rightarrow \neg \text{transit}(\text{Peer1}, \text{Peer2})\}
\]

\[
\text{define} \; \text{Main} = \\
\text{Destination} \; \& \; \text{Locality} \; \& \; \text{NoTransit} \; \& \; \text{agg}(\text{GP, in} \rightarrow \text{out})
\]
Compilation

Propane → Compiler → Topology → BGP Configs
Compilation

Constraints on Policy

Topology

State Machines

Propane -> Compiler -> BGP Configs
Compilation

Topology

Jointly analyze with topology

State Machines

Propane

Product Graph

Compiler

BGP Configs
end(Y) & (path(A, C, D) >> any)
Compilation: A simple Example

\[
\text{end}(Y) \& (\text{path}(A, C, D) \gg \text{any})
\]

Convert to Regex

\[
XACDY \gg (\Sigma^*) Y
\]
Reversed Automata from Policies

Policy:
1. XACDY
2. $\Sigma^* Y$

More preferred paths
Less preferred paths
Reversed Automata from Policies

Policy:
1. XACDZY
2. ($\Sigma^*$)Y
Reversed Automata from Policies

Reversed automata tracks BGP message flow

Policy:

1. XACDY
2. (Σ*) Y
Constructing the Product Graph (PG)
Constructing the Product Graph (PG)

Topology Location

Automata States

(X,5,1)
Constructing the Product Graph (PG)

Topology Location

Automata States

Path preferences

(X, 5, 1)

{1, 2}
Constructing the Product Graph (PG)
Constructing the Product Graph (PG)
Constructing the Product Graph (PG)

Graph capturing all possible policy-compliant paths through the topology
Constructing the Product Graph (PG)

Preferences

\[
\begin{align*}
\Sigma &: 0 \\
0 &: Y \\
1 &: D \\
2 &: C \\
3 &: A \\
4 &: X \\
5 &: X, 5, 1 \\
& \quad \{1, 2\} \\
& \quad \{2\} \\
\end{align*}
\]
Constructing the Product Graph (PG)

Preferences
Constructing the Product Graph (PG)

Accept: Y D C A X

Preferences

{2}

{1, 2}
Constructing the Product Graph (PG)
Idea 1: Restrict advertisements to edges

- Encode state in a BGP community tag
- Incoming edges — import filters
- Outgoing edges — export filters

Let BGP find some allowed path dynamically
Compilation to BGP:

D allows import matching regex(Y)

{2} {1, 2}
Compilation to BGP:

D exports to C with tag (2,1)
Compilation to BGP:

C allows import from D with tag (2,1)
Compilation to BGP:

C exports to A, B, D, E with tag (3, 1)
Compilation to BGP:

Idea 2: Find preferences
- Direct BGP towards best path
- Under all combinations of failures

Let BGP find **the best** path dynamically
Compilation to BGP:

Router C
match peer = D ...
match peer = E ...

\(\text{start} \rightarrow (Y,1,1) \rightarrow (E,-,1) \rightarrow (C,-,1)\)
Compilation to BGP:

Router C

match peer = D ...
local-pref ← ???

match peer = E ...
local-pref ← ???
Compilation to BGP:

Efficient algorithm to assign preferences that forces BGP to find the best paths for all possible failures

See the paper for details!
Compilation to BGP:

Implementation: Local preference

Less preferred import

More preferred import
Compilation to BGP:

Implementation: MED/Prepending

Less preferred import

More preferred import
Compilation: A simple Example

\[
\text{end}(Y) \& (\text{path}(A, C, D) \gg \text{any})
\]
Implementation

- Written in 7000 lines of F#
- Generates Quagga configurations
- A number of other analyses & features
Benchmarks

- Configurations from a large cloud provider
- Policy described in English documents
- Datacenter and Backbone policies
Policy Size

Without prefix/peer definitions

- Datacenter policy: 31 lines of Propane
- Backbone policy: 43 lines of Propane

Conventional BGP configurations are 1000s of lines
Compilation Time

- Compile for each prefix equivalence class
- Compile for each equivalence class in parallel
- 8 core, 3.6 GHz Intel Xeon processor

![Graphs showing compilation time for Data center (< 9 min) and Backbone (< 3 min)]
Configuration Size

Optimizations

- Avoid using community tags when unambiguous
- Reuse community values across peers
- Merge import/export behaviors across peers

Results

- Optimizations yield 50-100x decrease in config size
- Configurations ~1000-10000 lines per router
Propane: Summary

High-level language

- **Centralized** network programmability
- Constraints specify preferred paths and backups in case of failure
- Uniform abstractions for **Inter**- and **Intra**-domain routing
- Core policy in 30-50 lines of Propane vs. 1000s

Compiler

- **Distributed** implementation via BGP
- Static analysis guarantees policy compliance for **all failures**
- **Scales** to reasonably sized network topologies