Erweiterungen für VAST
Importer, Middleware und Web-Applikation

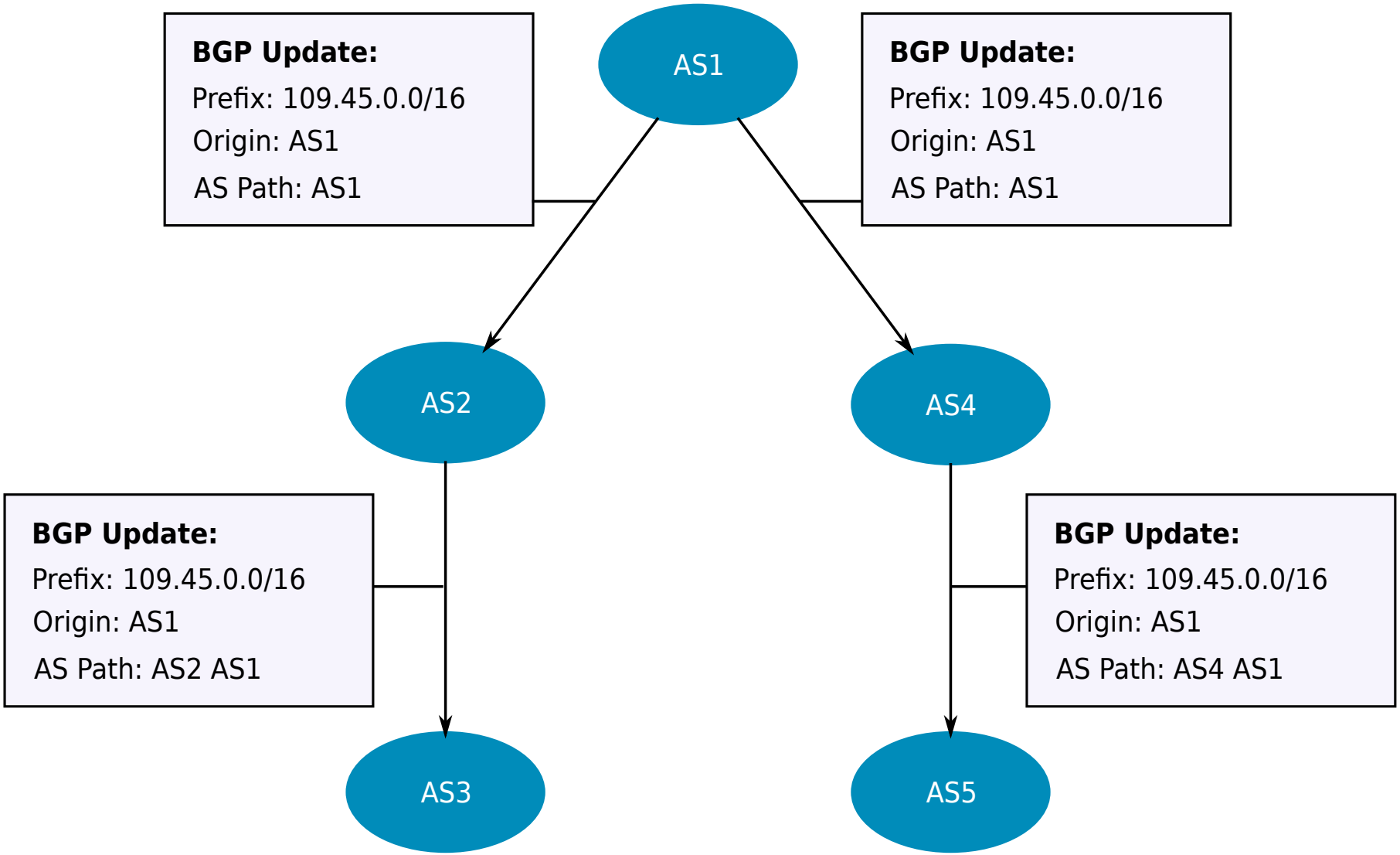
Samir Al-Sheikh, Andreas Reuter, Fabrice Ryba, Robert Schmidt
Freie Universität Berlin

Softwareprojekt Technische Informatik, SS 2015

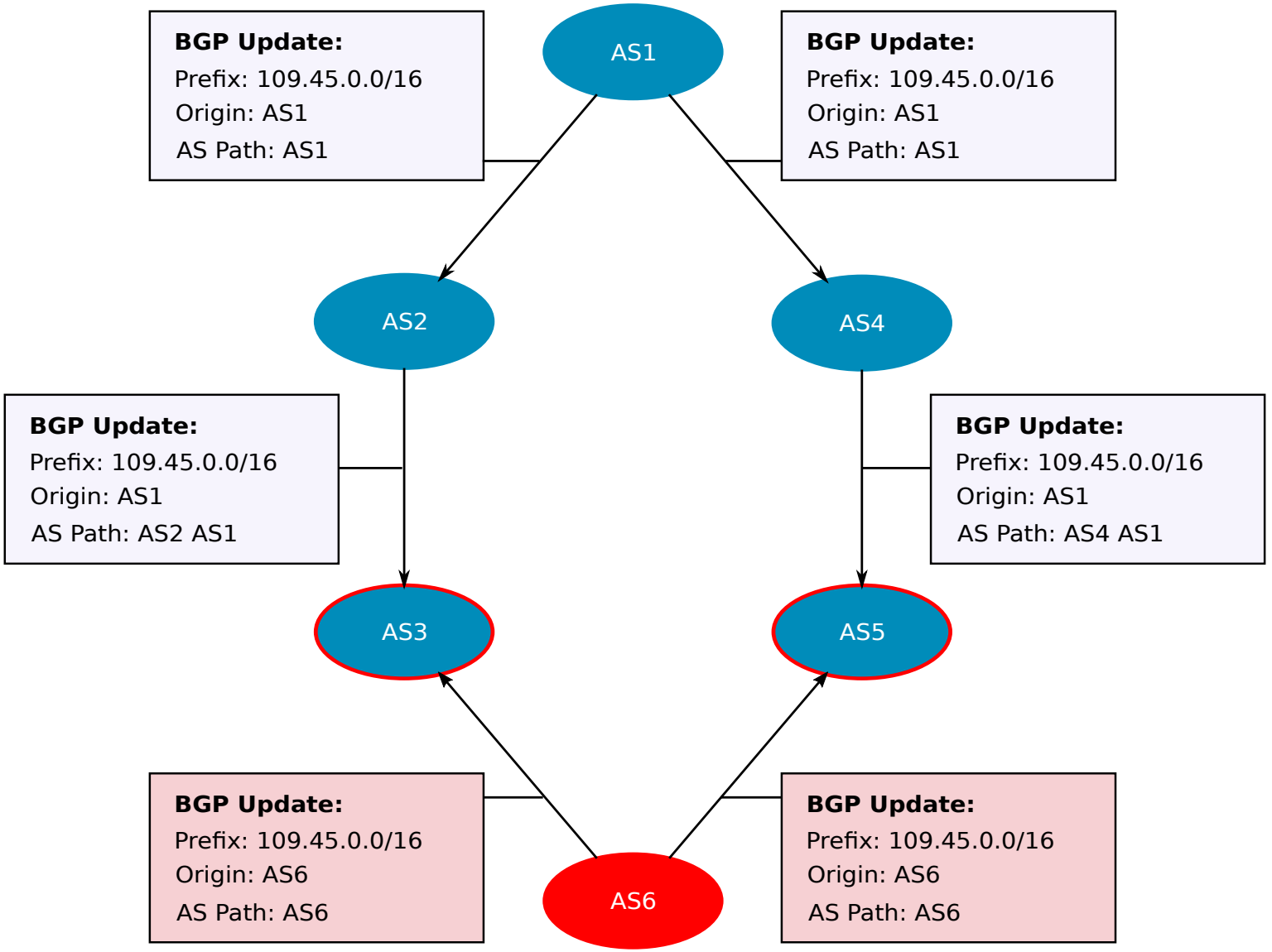
Gliederung

Gliederung

BGP Funktionsweise



BGP Hijacking



VAST = Visibility Across Space and Time

Anwendungsfall: Netzwerk-Forensik

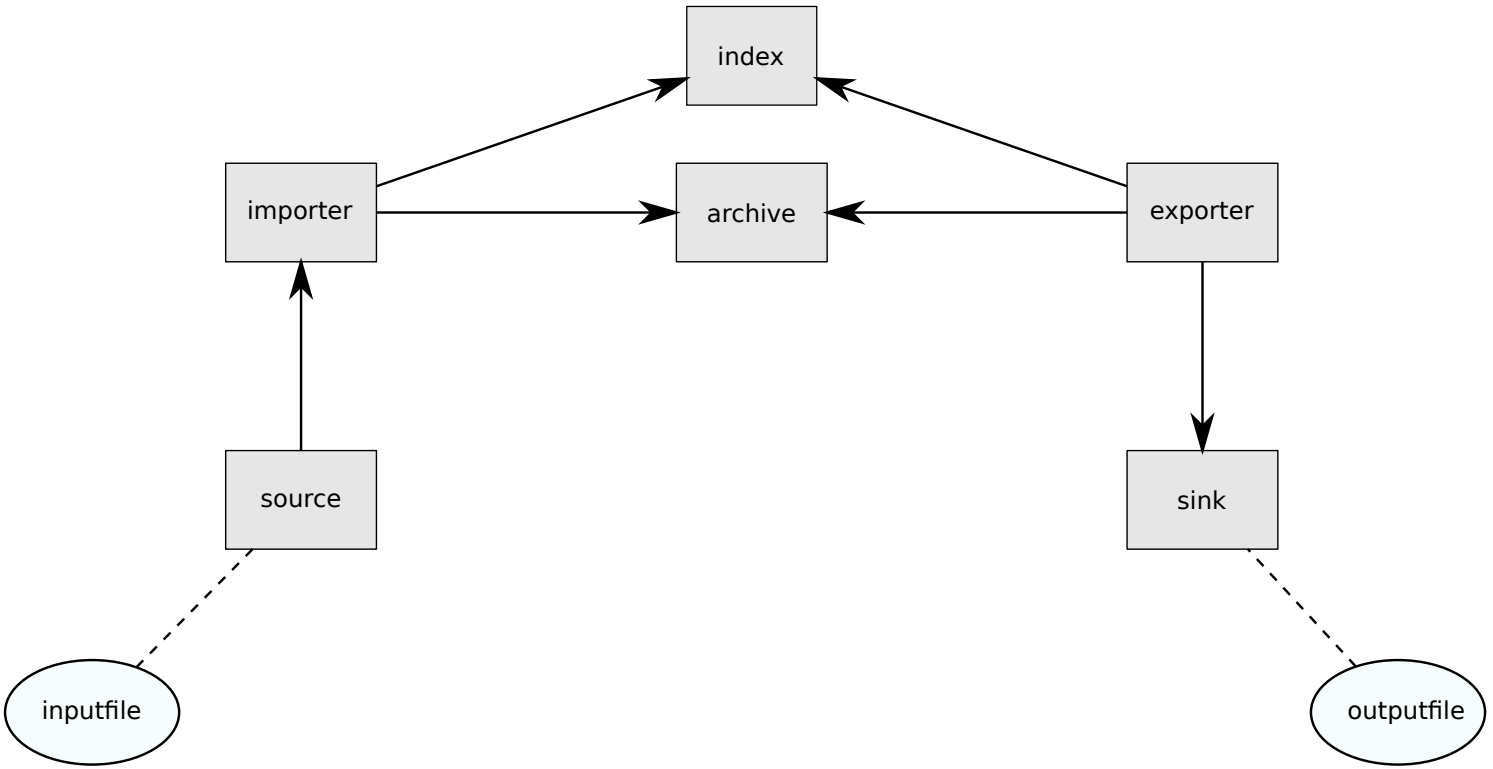
- Sehr viele Log-Dateien von verschiedenen Systemen/Protokollen
- Bei Problem: Durchsuchen und Analysieren großer Datenmengen
- Mit herkömmlichen Mitteln sehr Zeitaufwendig

VAST:

- Bietet eine sehr schnelle Durchsuchung von Massendaten
- Explorative Suchen können schnell ausgeführt werden (wichtig bei forensischen Analysen)

VAST - Architektur

- Setzt auf das Actor-Modell
  - Nebenläufige Einheiten (Actor)
  - Kommunikation ausschließlich über Nachrichten



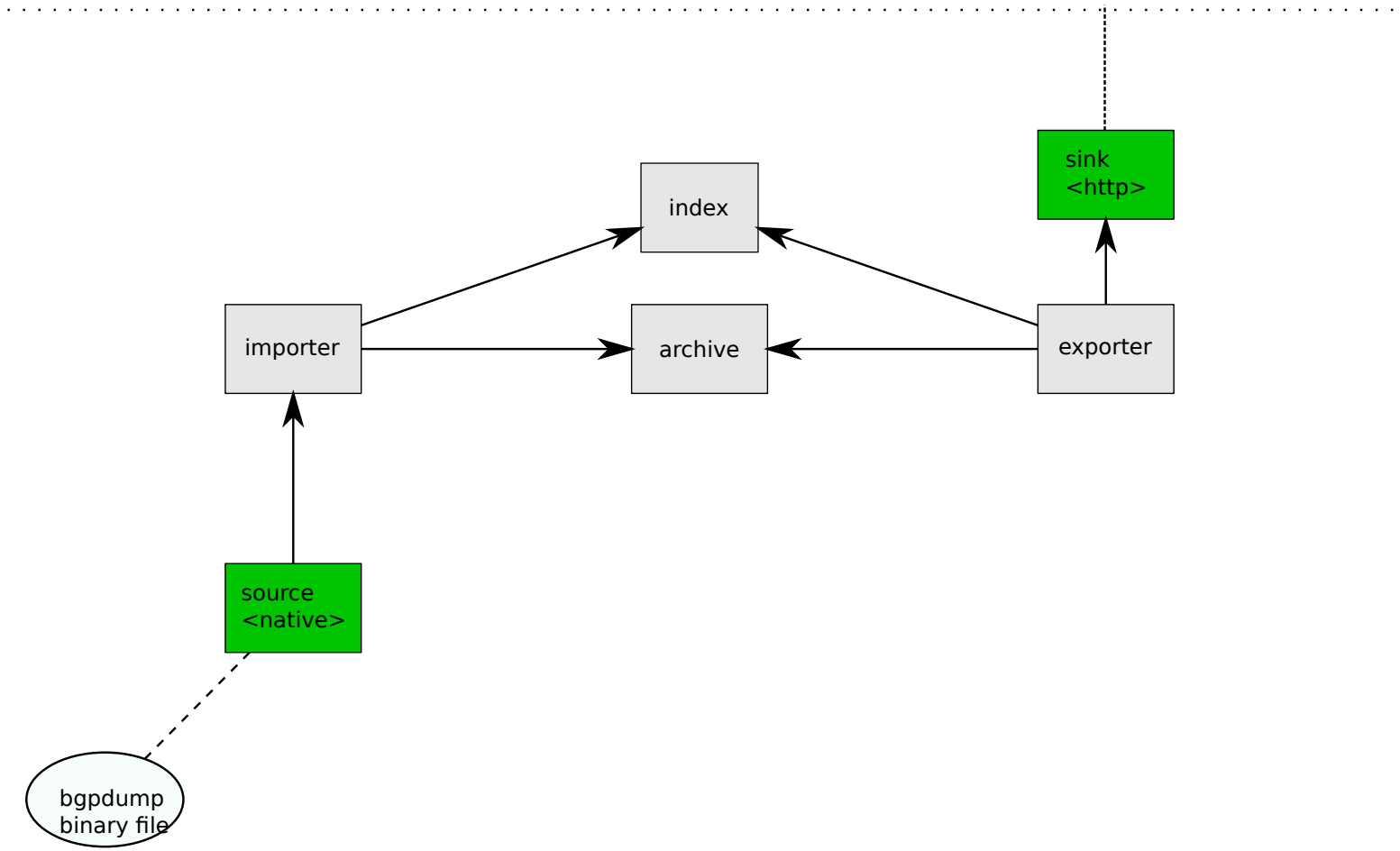
Gliederung

Motivation und Ziele

- Bisheriger BGP-Importer liest nur Text-Format ein –> langsam bei vielen Dateien
- Erweiterung des BGP-Importers, um gängige Binär-Formate nativ einzulesen
- User-Interface bisher nur über Konsole
- Erweiterung um Graphische Web-Oberfläche zur Datenvisualisierung
- Dazu:
  - Entwicklung einer REST-Schnittstelle (Middleware)
  - Entwicklung einer Web-Anwendung für die Analyse von BGP-Daten

Gliederung

Web Interface



Importer

Importer

- Low-Order Parser interpretieren eingelesene Bytes
- High-Order Parser fassen anschließend Bytes zu Paketfeldern zusammen
- Gesamtes BGP-Paket als Event übergeben

Importer - Stand und Probleme

Heutiger Stand:

- BGP-Dumps können eingelesen werden

Probleme:

- häufige Änderungen durch aktive Entwicklung
- ungenaue Definition der Paketfelder

Nächster Milestone:

- Parallelität
- Performance-Test

Middleware

- HTTP über CAF-Nachrichten (Broker)
- Webserver als eigener Actor
- Erzeugt bei jeder Verbindung neuen Actor zur Verarbeitung und Weiterleitung der Query

Heutiger Stand:

- Query empfangen und antworten

Probleme:

- Projekt ist in aktiver Entwicklung, häufige Änderungen
- Einige Fehler

Nächster Milestone:

- Schnittstelle erweitern
- HTTP-Parser

Web-Anwendung

Motivation:

- Einfaches exploratives Durchsuchen von großen Datenmengen
  $\rightarrow$ Daten visualisieren und einfaches Filtern ermöglichen

Technologien:

- Python 3 + Django
- Kommunikation zu VAST: HTTP + JSON

Web-Anwendung - Stand und Probleme

Heutiger Stand:

- Nimmt VAST-Anfragen entgegen
- Schickt Anfrage an VAST-API
- Empfängt JSON Antwort
- Erstellt eine filterbare Tabelle

Nächster Milestone:

- Visualisieren der Routenänderungen in einem Graphen
- VAST-Anfragenerstellung durch graphische Elemente erleichtern
- Weitere Anwendungsfälle implementieren

Gliederung

Demo

...