Broker

Bro’s New Communication Library

Robin Sommer
ICSI / LBNL / Broala

robin@icsi.berkeley.edu
robin@broala.com
http://www.icir.org/robin
Bro Communication
Bro Communication

Exchanging Events
Separates event generation from handling.
Bro Communication

Exchanging Events
Separates event generation from handling.

Logging Remotely
Aggregates logs on remote side.
Exchanging Events
Separates event generation from handling.

Logging Remotely
Aggregates logs on remote side.

Distributing State
Provides shared view across instances.
Bro Communication

Exchanging Events
Separates event generation from handling.

Logging Remotely
Aggregates logs on remote side.

Distributing State
Provides shared view across instances.
Use Case: Bro Cluster

- Bro Manager
- Bro Worker 1
- Bro Worker 2
- Bro Worker 3

Logs & Events

Load-balanced Traffic Stream
Use Case: Bro Cluster

- Bro Manager
- Bro Proxy
- Bro Worker 1
- Bro Worker 2
- Bro Worker 3

Logs & Events

Load-balanced Traffic Stream

&synchronized State

Broker
Use Case: External Integration

Events

Application A

Application B

Application C
Traditional Implementation

Events
Logs
State Updates

Bro A

Bro B
Traditional Implementation

Events
Logs
State Updates

Bro A → Child Process ← Child Process → Bro B

Broker
Traditional Implementation

- Events
- Logs
- State Updates


libbroccoli

C  Python  Ruby  Perl

Broker
Traditional Implementation

Events

It’s showing its age …

Race conditions with &synchronized.
No good persistence story.
Not much control over data flow.
Complex & inefficient protocol.
Implementation deficiencies.
Two independent implementations.
Introducing the Broker Library

Events
Logs
State Updates

Bro A

Bro B
Introducing the Broker Library

Events
Logs
State Updates

Bro A  Broker  Broker  Bro B
Introducing the Broker Library

Events
Logs
State Updates

Bro A Broker Broker Bro B

Bro A

Broker

Application

C
C++
Python
Introducing the Broker Library

Events
Logs
State Updates

Broker

Unified Library
New protocol
Publish/subscribe
Limiting type system’s flexibility.
Explicit state operations.
Exchanging Events with Broker

my_event("Bro", 42)
Exchanging Events with Broker

Topic: “bro/event”
my_event(“Bro”, 42)
Exchanging Events with Broker

Topic: “bro/event”
my_event(“Bro”, 42)

subscribe_to_events(“bro/event”)
Exchanging Events with Broker

subscribe_to_events("bro/event")

Topic: “bro/event”
my_event("Bro", 42)
Exchanging Events with Broker

Topic: “bro/event”
my_event("Bro", 42)

Broker

Sender

my_event("Bro", 42)

Receiver

Broker

subscribe_to_events("bro/event")

my_event("Bro", 42)
Exchanging Events with Broker

Demo
Forwarding Logs with Broker

Log::write(…)

Sender

Bro / Broker

Receiver

Broker / Bro
Forwarding Logs with Broker

Topic: “bro/log”
Log::write(…)

Sender

Receiver
Forwarding Logs with Broker

subscribe_to_logs("bro/log")

Topic: "bro/log"
Log::write(...)
Forwarding Logs with Broker

subscribe_to_logs("bro/log")

Topic: "bro/log"
Log::write(...)
Forwarding Logs with Broker

subscribe_to_logs("bro/log")

Topic: “bro/log”
Log::write(...)

Bro | Broker
---|---
Sender

Log::write(...)
Data Stores with Broker
Data Stores with Broker

Broker maintains global, persistent key/value stores.
Data Stores with Broker

Broker maintains global, persistent key/value stores.
Data Stores with Broker

Broker maintains global, persistent key/value stores.

```
s = create_master("my_store")
```
Data Stores with Broker

Broker maintains global, persistent key/value stores.

```
s = create_master("my_store")
insert(s, 192.150.187.12, 21)
```
Data Stores with Broker

Broker maintains global, persistent key/value stores.

```python
s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
```
Data Stores with Broker

Broker maintains global, persistent key/value stores.

c = create_clone("my_store")

s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.150.187.12</td>
<td>21</td>
</tr>
<tr>
<td>131.159.14.1</td>
<td>5</td>
</tr>
</tbody>
</table>

| Authoritative Version |
Broker maintains global, persistent key/value stores.

```
c = create_clone("my_store")
s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
```
Data Stores with Broker

Broker maintains global, persistent key/value stores.

```
c = create_clone("my_store")

s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
```

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.150.187.12</td>
<td>21</td>
</tr>
<tr>
<td>131.159.14.1</td>
<td>5</td>
</tr>
</tbody>
</table>

{ content }

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.150.187.12</td>
<td>21</td>
</tr>
<tr>
<td>131.159.14.1</td>
<td>5</td>
</tr>
</tbody>
</table>

Authoritative Version

Cached Version
Data Stores with Broker

Broker maintains global, persistent key/value stores.

c = create_clone("my_store")

s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
Broker maintains global, persistent key/value stores.

```python
s = create_master("my_store")
n = lookup(c, 192.150.187.12)
s = insert(s, 192.150.187.12, 21)
s = insert(s, 131.159.14.1, 5)
c = create_clone("my_store")
```
Broker maintains global, persistent key/value stores.

c = create_clone("my_store")
n = lookup(c, 192.150.187.12)
insert(c, 192.150.187.12, 21)
insert(c, 192.150.187.43, 1947)
s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
Data Stores with Broker

Broker maintains global, persistent key/value stores.

```python
c = create_clone("my_store")
n = lookup(c, 192.150.187.12)
insert(c, 192.150.187.43, 1947)
s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
```
Broker maintains global, persistent key/value stores.

```python
c = create_clone("my_store")
n = lookup(c, 192.150.187.12)
insert(c, 192.150.187.43, 1947)

s = create_master("my_store")
insert(s, 192.150.187.12, 21)
insert(s, 131.159.14.1, 5)
```

### Data Stores with Broker

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.150.187.12</td>
<td>21</td>
</tr>
<tr>
<td>131.159.14.1</td>
<td>5</td>
</tr>
<tr>
<td>192.150.187.43</td>
<td>1947</td>
</tr>
</tbody>
</table>

**Cached Version**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.150.187.12</td>
<td>21</td>
</tr>
<tr>
<td>131.159.14.1</td>
<td>5</td>
</tr>
</tbody>
</table>

**Authoritative Version**

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.150.187.12</td>
<td>21</td>
</tr>
<tr>
<td>131.159.14.1</td>
<td>5</td>
</tr>
<tr>
<td>192.150.187.43</td>
<td>1947</td>
</tr>
</tbody>
</table>
Data Store Features
Data Store Features

Data Types
Supports Bro data types for keys & values.
Data Store Features

Data Types
Supports Bro data types for keys & values.

Operations
  Increment/decrement.
  Set insert/delete.
  Vector push/pop.
  Automatic expiry.
Data Store Features

Data Types
Supports Bro data types for keys & values.

Operations
Increment/decrement.
Set insert/delete.
Vector push/pop.
Automatic expiry.

Persistence
Choice of SQLite or RocksDB.
Broker Bits & Pieces
Fine-granular Subscription Model

All elements come with a topic or name.
Prefixed-based subscription for topics.
Fine control what’s published and subscribed to.
Fine-granular Subscription Model
All elements come with a topic or name.
Prefixed-based subscription for topics.
Fine control what’s published and subscribed to.

Security Model
Peers are assumed to be trustworthy.
Currently no encryption/authentication; SSL later.
Broker’s Foundation: CAF

C++ Actor Framework

http://actor-framework.org
Broker’s Foundation: CAF

C++ Actor Framework
ConcURRENCY through Erlang-style lightweight actors.
Concise messaging API.
Network-transparency abstracting transport.
Efficient and adaptive.
BSD license.

http://actor-framework.org
Broker’s Foundation: CAF

C++ Actor Framework
Concurrency through Erlang-style lightweight actors.
Concise messaging API.
Network-transparency abstracting transport.
Efficient and adaptive.
BSD license.

Trade-Offs
Needs a C++11 compiler.
Introduces a new, non-standard dependency.

http://actor-framework.org
The Future: A “Stateful Deep Cluster”
The Future: A “Stateful Deep Cluster”

Example: Geographically distributed organization.
The Future: A “Stateful Deep Cluster”

Example: Geographically distributed organization.
The Future: A “Stateful Deep Cluster”

Example: Geographically distributed organization.

Global Master

Regional Heads

Local Clusters
The Future: A “Stateful Deep Cluster”

Example: Geographically distributed organization.

- **Global Master**: Corp/GLOBAL/
- **Regional Heads**: Corp/US/, Corp/UK/, Corp/FR/, Corp/CN/
- **Local Clusters**: Corp/US/..., Corp/UK/..., Corp/FR/..., Corp/CN/...
The Future: A “Stateful Deep Cluster”

Example: Geographically distributed organization.
Broker, beyond Bro
Broker, beyond Bro

Distributed, real-time publish/subscribe platform, with a data model.
What This All Means for You Right Now
What This All Means for You Right Now

Bro 2.4

Legacy communication still the primary mechanism. Broker optional, 

`-enable-broker` to compile support. (CAF 0.13, C++11)
What This All Means for You Right Now

Bro 2.4
Legacy communication still the primary mechanism.
Broker optional, \texttt{--enable-broker} to compile support. (CAF 0.13, C++11)

Git master, as of today
Legacy communication still the primary mechanism.
Broker built by default, \texttt{--disable-broker} turns off. (CAF 0.14, C++11)
What This All Means for You Right Now

Bro 2.4
Legacy communication still the primary mechanism.
Broker optional, `--enable-broker` to compile support. *(CAF 0.13, C++11)*

Git master, as of today
Legacy communication still the primary mechanism.
Broker built by default, `--disable-broker` turns off. *(CAF 0.14, C++11)*

Bro 2.5 (tentative)
Broker provides primary communication mechanism.
Legacy communication deprecated, but remains available.
What This All Means for You Right Now

**Bro 2.4**
Legacy communication still the primary mechanism.
Broker optional, `--enable-broker` to compile support. (CAF 0.13, C++11)

**Git master, as of today**
Legacy communication still the primary mechanism.
Broker built by default, `--disable-broker` turns off. (CAF 0.14, C++11)

**Bro 2.5 (tentative)**
Broker provides primary communication mechanism.
Legacy communication deprecated, but remains available.

**Bro 2.6 (even more tentative)**
Broker provides primary communication mechanism.
Legacy communication no longer available.
What This All Means for You Right Now

Bro 2.4
Legacy communication still the primary mechanism. Broker optional, `-enable-broker` to compile support. (CAF 0.13, C++11)

Git master, as of today
Legacy communication still the primary mechanism. Broker built by default, `-disable-broker` turns off. (CAF 0.14, C++11)

Bro 2.5 (tentative)
Broker provides primary communication mechanism. Legacy communication deprecated, but remains available.

Bro 2.6 (even more tentative)
Broker provides primary communication mechanism. Legacy communication no longer available.

Credit for Broker goes to Jon Siwek!
Questions?

Robin Sommer
ICSI / LBNL / Broala

robin@icsi.berkeley.edu
robin@broala.com
http://www.icir.org/robin