



- The Message Queue project for PHP

Soenke Ruempler (soenke@ruempler.eu)

and Boris Erdmann (
boris.erdmann@gmail.com)

PHP Unconference Hamburg
26-27.04.2008

What is a Message Queue

- A FIFO buffer (first in / first out)
- Asynchronous push / pull paradigm
- Types of Message Queues
 - Local message queues (Inter Process Communication, Pipes)
 - Network message queues (Middleware) like ActiveMQ, MSMQ, Websphere MQ and dropr :)
- What is NOT a Message Queue
 - message bus / group communication systems like spread
 - high level frameworks like RPC / RMI etc

Ok, so what is dropr and why?

- Dropr is a PHP-only peer-2-peer Message Queue with a local queue storage at each node
- Why reinvent the wheel?
 - There's no plain PHP message queue
 - Jimdo has a distributed architecture with servers in Germany, USA and China so network outtakes are daily business
 - You could use things like ActiveMQ, but a Java Message System at every node – and connectors are rare

Requirements and features

- **RELIABLE** and **DURABLE** (failsafe)-messaging over networks
- decentralized architecture without a single (point of failure) server instance
- easy to setup and use
- **modularity** for queue storage and message transports

What's currently implemented

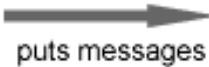
- A filesystem storage (no database needed)
- A CURL-Upload transport (can handle multiple parallel uploads)
- Channels for messages
- Durability and Reliability
- The client daemon with an angel process if PHP dies :)
- Monitoring and Queue cleaning scripts
- Sorry, no real documentation yet, but a Howto

Ok, too much text

Client

```
<?php  
$queue->sendMessage('test');
```

Your sender application

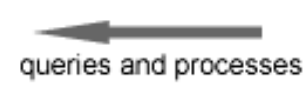


puts messages

API



Client Database



queries and processes



Client Queue Daemon

Transport



Server(s)

```
<?php  
$msg = $queue->poll();
```

Your receiver application
normally a daemon process

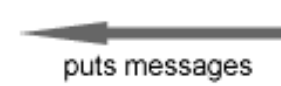


polls messages

API



Server Database



puts messages



Server

Your application

Message queue

Use-Cases at Jimdo

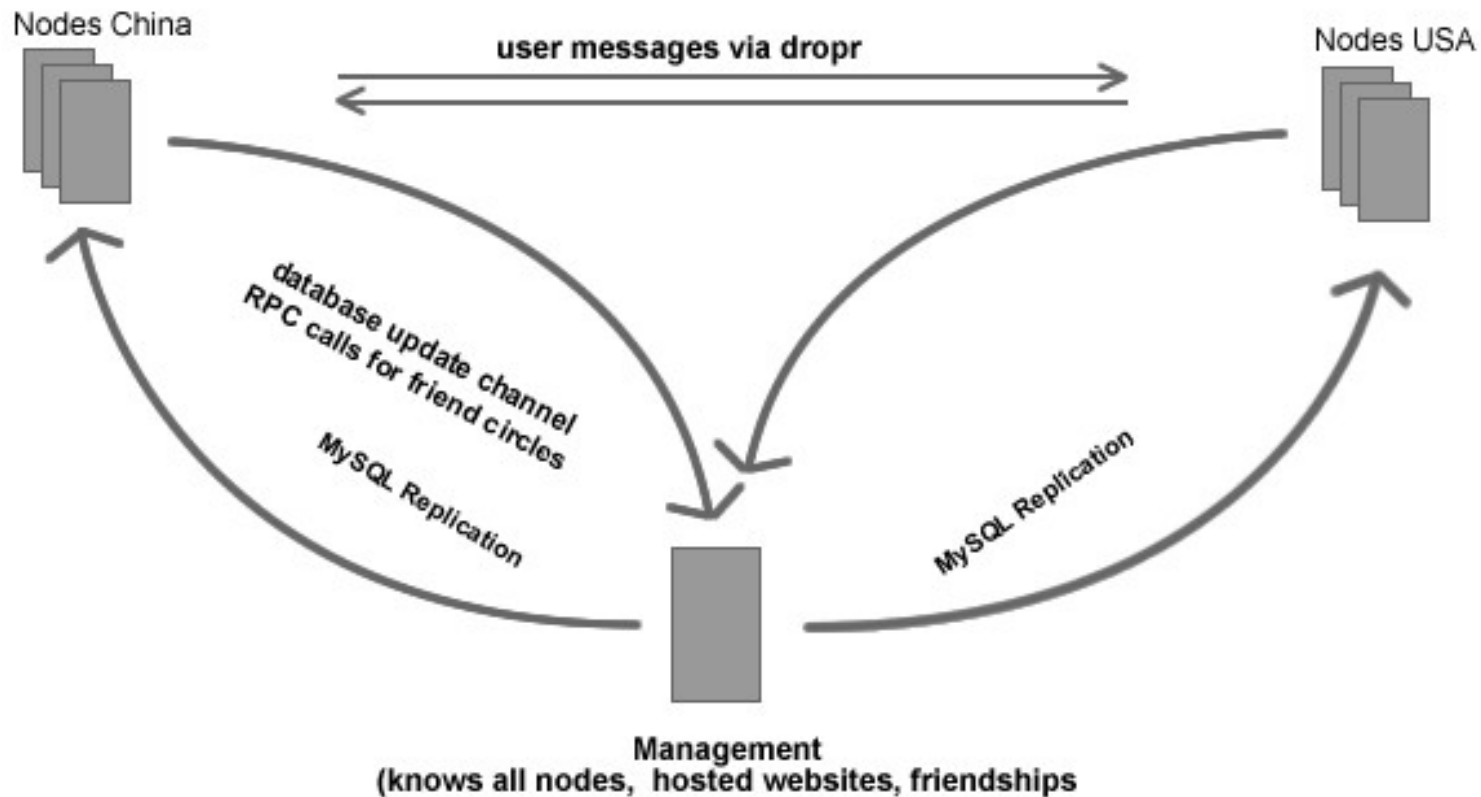
- Jimdo is a online service that lets you
 - easily create beautiful websites with a drag-and-drop interface
 - social networking: friendship, user messages, directory of websites etc.
 - And this with a distributed architecture?

The challenge

- Challenge: Distributed world-wide architecture
- Jimdo has a central database that knows all nodes, users and websites
 - But Websites and Website Data are stored locally within the local nodes
 - Data has to be synchronized with the central management database
- So you need the Google borg-network, a worldwide MySQL-Master-Master Ring or a Message Queue

How is dropr integrated?

Distributed Architecture of Jimdo



Pros and Cons

- Vorteile:
 - Nachrichten über Objekt-Updates werden im Hintergrund abgearbeitet
 - Lokale Server sind autark
 - Migrationen sind leichter (einfach den client oder server stoppen)
- Nachteile
 - Keine direkte Rückmeldung von Remote-Calls, Status muss z.B. per ACK Message transportiert werden

Live-Presentation

- Shell :)

Further Information, References

- dropr: <https://www.dropr.org>
- Jimdo :) <http://www.jimdo.com/>
- Draft for dropr:
- Wikipedia
 - http://en.wikipedia.org/wiki/Message_queue