# Self-Organizing Publish/Subscribe

Michael A. Jaeger

Berlin University of Technology, Germany
Institute for Telecommunication Systems
Communications and Operating Systems Group (KBS)

E-mail: michael.jaeger@acm.org

$2^{nd}$ International Middleware Doctoral Symposium
2005

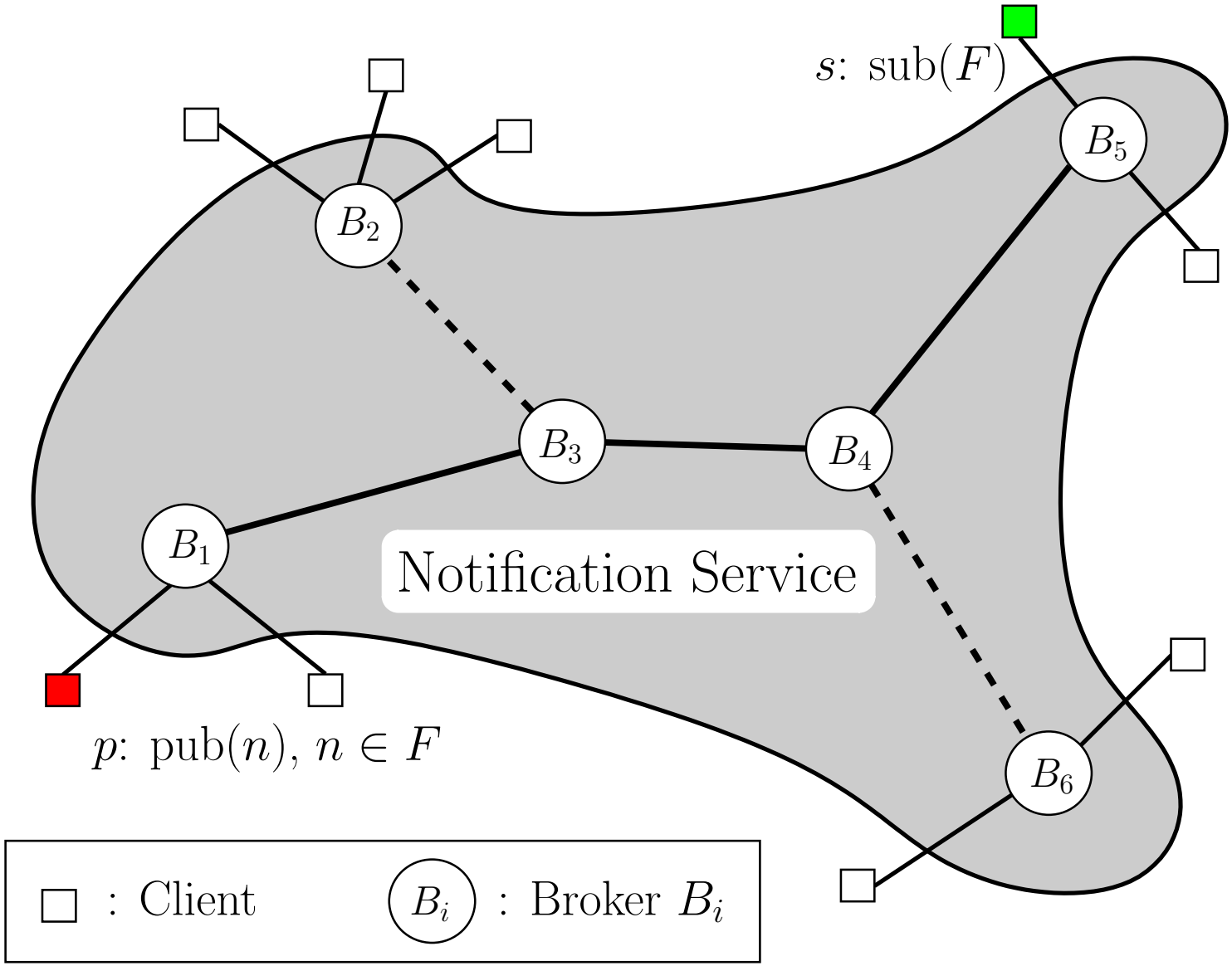
# Motivation

- The Internet is almost omnipresent today and an important source for information
- Broadband Internet access and flatrates getting cheaper and thus more common
- Many users are willing to share (legal) resources to improve their QoS ($\rightarrow$BitTorrent)
- An increasing number of consumers from the past become publishers and cooperate

$\Rightarrow$ Cooperative event-based computing can be a building block to support this development

# Outline

# Basics


$s$: sub($F$)
$B_1$
$B_2$
$B_3$
$B_4$
$B_5$
$B_6$
Notification Service
$p$: pub($n$), $n \in F$
: Client
$B_i$ : Broker $B_i$

# So far...

- Most pub/sub research has abstracted from fault-tolerance issues
- Topology of the notification service is mostly assumed to be static
- Message ordering and completeness during reconfiguration has not been considered as a main issue

# Contributions of My Thesis

## Self-Stabilizing Publish/Subscribe

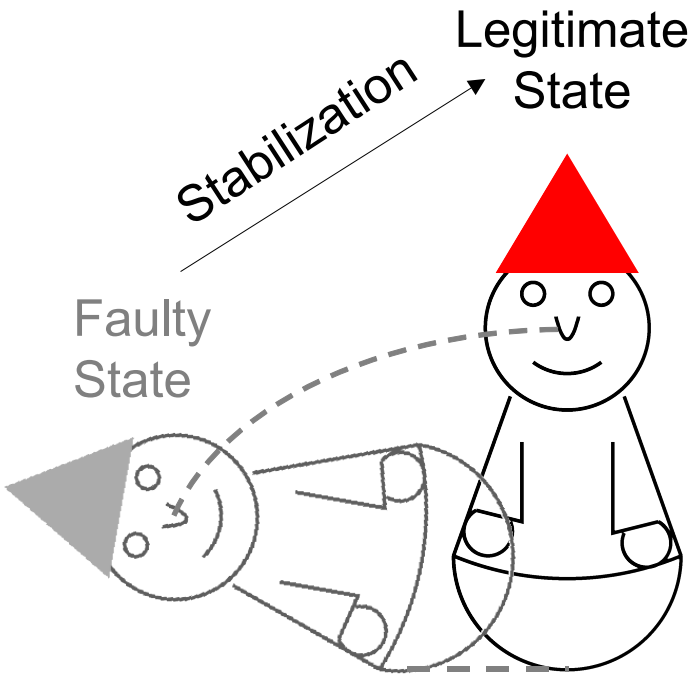
Add self-stabilization to publish/subscribe systems, analyze efficiency, and incorporate external reconfiguration requests.

+

## Adapt Structure of Notification Service

Improve system performance by optimizing the inner structure according to usage patterns of the clients.

# Self-Stabilizing Publish/Subscribe



# State of The Art

- Most current approaches focus only on certain failures (like link failures) [CFMP04]
- Message ordering in combination with fault-tolerance has not been an issue in past research

# State of The Art

- Pub/sub layered on peer-to-peer routing substrates inherits fault-tolerance mechanisms from lower layers, but does not *guarantee* stabilization nor an uninterrupted service
- Self-stabilization does not make any assumption regarding the faults that might happen, as long as they are *transient*

# Goal of Research

- Add provable and efficient self-stabilizing features to pub/sub
  - *Small scale:* Scenarios where no (skilled) administrators are available (*e.g.*, e-home environment)
  - *Large scale:* Essential because human centered fault-management is nearly impossible here
- Evaluation through stochastic analysis and simulation
  - Formalism to compare efficiency with flooding
  - Discrete event simulator to evaluate the performance of self-stabilization

# Goal of Research

- Incorporate external reconfiguration requests into self-stabilizing mechanism
  - Important for combination with (manual/automated) optimization
  - Prevent negative influence of stabilization on system performance

# Self-Organizing Broker Topology

# State of The Art

- Most approaches assume a static notification service
- Some approaches (*e.g.*, HERMES [Pie04]) employ peer-to-peer routing substrates and inherit their flexibility and self-organizing capabilities
- However, the structure of the broker topology is not adapted to usage patterns of the clients
  - $\Rightarrow$ System performance is suboptimal
  - $\Rightarrow$ Not well suitable for dynamic environments like the Internet (*e.g.*, slashdot effect, changing behavior due to changing global interests)

# State of The Art

- Another approach determines broker interests based on filters they manage [BBQV04]
- Problems with this approach:
  - Common interests are hard to determine
  - Implementation depends on the filter model used
  - Implicit assumption on distribution of notifications
  - Possibility of message loss during reconfiguration

# Goal of Research

- Enable brokers to “measure” their interests and compare it with that of other brokers
  - → How can interests be determined (efficiently)?
- Bring brokers with common interests closer to each other
  - → Consider the effect on other brokers in the system!
- Use local information to maintain scalability
  - Local decisions can have a global impact

# Goal of Research

- Consensus of the affected brokers is needed to prevent degredation
- Uninterrupted service while optimizing the system
  - No message loss or duplicates
  - Keep message ordering
    (*e.g.*, producer FIFO and causal ordering)
  - Minimize delay caused by reconfigurations

# Combination With Self-Stabilization

- Optimizer works on top of self-stab routing layer
  - Prevent oscillation due to competing layers!
- Issues reconfiguration requests to lower layers
  - Coordinate actions on different layers!
- Keep configuration easy
  - Not: Self-organization by configuration parameter adjustment!

# Methodology and Evaluation

- Evaluation through simulation
- Comparison of self-organizing notification service with static approach (random, MST, MIT) regarding:
  - Message complexity
  - Delay
  - Cost of reconfiguration (messages)
  - Comparison
- Integration into the REBECA pub/sub system

# Applications

# State of The Art

- RSS is the most prominent publish/subscribe application on the Internet today
- Relies on “push via pull”
- Scalability problems already play a role
- “Content syndication” based on explicit knowledge of RSS sources

# State of The Art

- Recent work on filtering graph-based structures and CMS-integration [PLJ05]
- FeedTree uses application-layer multicast to disseminate RSS micronews [SMPD05]

# Goal

- Support push-RSS via publish/subscribe
- Increase scalability, timeliness, and power by content-based filtering
- Use “collaborative polling” to lower load on RSS-server
  - $\Rightarrow$ Easy integration of legacy RSS
- Consider consequences for integrity

# Thank you for your attention!


R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito.
A self-organizing crash-resilient topology management system for content-based publish/subscribe.
In *3rd International Workshop on Distributed Event-Based Systems (DEBS'04)*, Edinburgh, Scotland, UK, May 2004.


Gianpaolo Cugola, Davide Frey, Amy L. Murphy, and Gian Pietro Picco.
Minimizing the reconfiguration overhead in content-based publish-subscribe.
In *Proceedings of the 2004 ACM symposium on Applied computing (SAC'04)*, pages 1134–1140, New York, NY, USA, 2004. ACM Press.


P. R. Pietzuch.
*Hermes: A Scalable Event-Based Middleware*.
PhD thesis, Computer Laboratory, Queens' College, University of Cambridge, February 2004.


Milenko Petrovic, Haifeng Liu, and Hans-Arno Jacobsen.
G-ToPSS: Fast filtering of graph-based metadata.
In *Proceedings of the World Wide Web Conference*, pages 539–547, Chiba, Japan, 2005.


Dan Sandler, Alan Mislove, Ansley Post, and Peter Druschel.
FeedTree: Sharing web micronews with peer-to-peer event notification.
In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, Ithaca, NY, February 2005.