PEPINO: PEer-to-Peer network INspectOr [*]

Donatien Grolaux, Boris Mejías, and Peter Van Roy
Université catholique de Louvain, Belgium
firstname.lastname@uclouvain.be

Abstract

PEPINO is a simple and effective peer-to-peer network inspector. It visualises not only meaningful pointers and connections between peers, but also the exchange of messages between them, providing a useful tool for debugging purposes. It can monitor running networks, simulate them and log them in order to reproduce interesting case scenarios. Failures can be explicitly introduced to study fault tolerant algorithms. The graphical representation of the network uses a physical model to attract or repel peers, allowing the user to study the system from different points of view. This demo aims to present the use of PEPINO in the development of a novel relaxed-ring topology for fault tolerant networks, where the representation of the ring based on predecessors may differ from the ring based on successors. We show how PEPINO is also useful for visualising other network topologies such as perfect ring or unstructured networks.

1. Introduction

Developing peer-to-peer systems requires the ability of visualising the network in terms of the designed algorithms. The more dynamic the case-studies become, the harder is to keep track of all interaction between peers. The obvious solution is to use a software to graphically represent the network, and thus, nearly every developer group creates its own network viewer to study their algorithms. Then, why do we present yet another viewer? The reason to do it is because existing tools are so ad-hoc to each network that studying a different network topologies in such tools did not allowed us to visualise our concrete issues.

We developed PEPINO, a PEer-to-Peer network INspectOr that adapts its graphical representation to several network topology such as ring, relaxed-ring, unstructured networks or even client-server. The graphical representation uses a physical model to attract and repel connected nodes depending on the weight of each kind of connection. This is how the viewer is able to dynamically adapt itself to different network architectures, without making any previous assumption on the topology.

Another crucial information for debugging is the exchange of messages between peers. PEPINO displays simultaneously with the representation of the network, the communication between peers and the events triggered by each one of them. Messages and events are annotated with different categories, allowing filters in order to get more meaningful information.

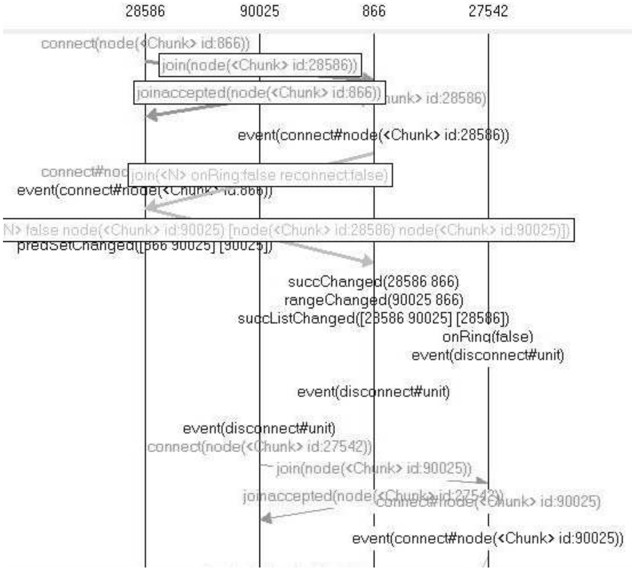

28586
90025
866
27542
connect(node(<Chunk> id:866))
join(node(<Chunk> id:28586))
joinaccepted(node(<Chunk> id:866))
event(connect#node(<Chunk> id:28586))
connect#nod
join(<N> onRing:false reconnect:false)
event(connect#node(<Chunk> id:866))
N> false node(<Chunk> id:90025) [node(<Chunk> id:28586) node(<Chunk> id:90025)])
predSetChanged([866 90025] [90025])
succChanged(28586 866)
rangeChanged(90025 866)
succListChanged([28586 90025] [28586])
onRing(false)
event(disconnect#unit)
event(disconnect#unit)
event(disconnect#unit)
connect(node(<Chunk> id:27542))
join(node(<Chunk> id:90025))
joinaccepted(node(<Chunk> id:27542))
connect#node(<Chunk> id:90025)
event(connect#node(<Chunk> id:90025))

Figure 1. Message exchange between peers

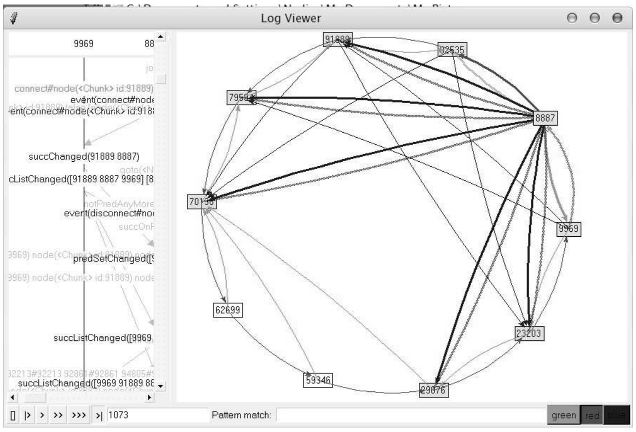
2. PEPINO and P2PS

We use PEPINO in the development of P2PS*v*3 [2], a Chord-like platform to develop fault-tolerant peer-to-peer applications. P2PS uses a network topology based on a relaxed-ring where only the predecessor links form a perfect ring, guaranteeing a correct distribution the responsi-

[*] This research is mainly funded by EVERGROW (contract number:001935) and SELFMAN (contract number: 034084), with additional funding by CoreGRID (contract number: 004265).

bility of the keys. Successors follow a different invariant allowing the presence of branches in the ring. The topology and the algorithms of P2PS provide a network that can survive efficiently to failures of nodes and also to broken links (inaccurate failure detection), which are often ignored.

P2PS is implemented using a software architecture based on tiers, where the lowest tier implements point-to-point communication. The relaxed-ring maintenance is another layer placed upper in the architecture. As we previously mentioned, PEPINO can display messages between peers in different categories. In the particular case of P2PS, every tier is represented by a category. Figure 1 depicts how messages are represented, and how one category is highlighted. The figure is shown in grey scale, but it is possible to distinguish that every category has its own colour. Every category can be enabled or disabled in order to avoid unnecessary verbosity.

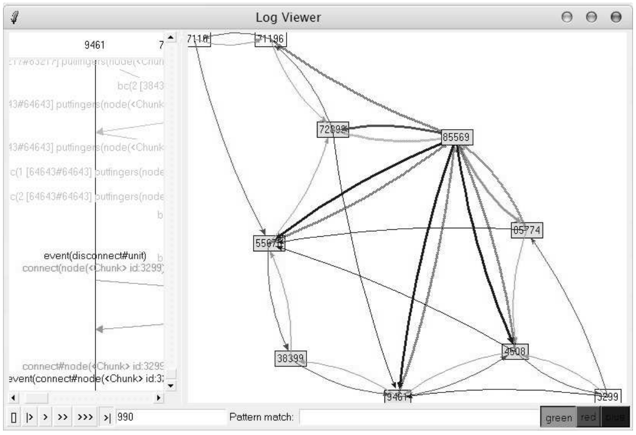


```
connect#node(<Chunk> id:91889)
event(connect#node
ent(connect#node(<Chunk> id:918
succChanged(91889 8887)
goto(<N
cListChanged([91889 8887 9969] [8
notPredAnyMore
event(disconnect#no
succOnF
9969) node(<Chu
predSetChanged([9
9969) node(<Chunk> id:91889) node
succListChanged([9969
32213#92213 92861#92861 94805#
succListChanged([9969 91889 88
```

Figure 2. Fingers on a ring network

During the demonstration, different networks will be presented in order to observe their dynamic behaviour. Figure 2 is a screenshot of PEPINO visualising a network using ring topology. On the left side of the screenshot is possible to observe the frame with messages between peers. On the right side, in the graphical representation of the ring, the finger of a particular peer are highlighted. On the bottom right corner, there is a set of buttons allowing the election of the strongest connector between peers for the graphical representation. The underlay physical model will adapt the parameter for attraction or repulsion of nodes according to these settings. Like this, it is possible to observe the network from different points of view. For instance, putting the focus on the predecessors or successor links.

To check how the network reacts to network failures, it is possible to explicitly inject temporary or permanent failures on nodes. Failures can also be injected in the communication channel between peers, which is one way to study inaccuracy of failure detection.



```
217#63217] putfingers(node(<Chun
bc(2 [3843
43#64643] putfingers(node(<Chunk
43#64643] putfingers(node(<Chunk
c(1 [64643#64643] puttingers(node
c(2 [64643#64643] puttingers(node
event(disconnect#unit)
connect(node(<Chunk> id:3299
connect#node(<Chunk> id:3299
event(connect#node(<Chunk> id:3
```

Figure 3. Relaxed-ring topology

Figure 3 depicts the visualisation of the relaxed-ring topology of P2PS, our main focus of interest for the demonstration. The information given by PEPINO helps the developer to understand why branches are created, and how the network recovers from crashed peers and broken links. P2PS implements different algorithms for the election of fingers such as Chord [5], Tango [1] and DKS [3], allowing the comparison between them.

PEPINO is also useful for bug reports. The history of a visualised network can be saved in a log file to be sent to developers. The log can be visualised at different speeds. In figures 2 and 3, a set of arrows can be seen at the bottom left corner. The speed of visualisation can be tuned with those buttons. It is also possible to run the visualisation until a particular event identified by a number, or matching a pattern.

PEPINO is implemented with Mozart [4], and it can be run on Linux, MacOSX and other Unix systems. It also runs on Windows 98/NT/XP.

References

[1] B. Carton and V. Mesaros. Improving the scalability of logarithmic-degree dht-based peer-to-peer networks. In M. Danelutto, M. Vanneschi, and D. Laforenza, editors, *Euro-Par*, volume 3149 of *Lecture Notes in Computer Science*, pages 1060–1067. Springer, 2004.

[2] DistOz Group. P2PS: A peer-to-peer networking library for Mozart-Oz. *http://gforge.info.ucl.ac.be/projects/p2ps*, 2007.

[3] A. Ghodsi. *Distributed k-ary System: Algorithms for Distributed Hash Tables*. PhD dissertation, KTH — Royal Institute of Technology, Stockholm, Sweden, Dec. 2006.

[4] Mozart Community. The Mozart-Oz programming system. http://www.mozart-oz.org, 2007.

[5] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.