Peer-to-Peer Protocols based on asynchronous message communication with prototype-like implementation of a Pastry network over the EventChannelNetwork

Alexander Liebrich

2nd February 2006

1 Motivation

Peer-to-peer networks (P2P) are characterized by the fact that all subunits (nodes) are equal and allow both sending and receiving of data. The ID-based addressing abstracts from conventional IP addresses and can be adapted thereby to the size of the logical network. In contrast to the common client-server communication pattern between few servers and many clients, typical P2P-connection consists of many individual point-to-point connections between equivalent nodes. This fact can be used for distribution of load or for redundancy. P2P networks use broadcast protocols to find and join a network and subsequently to observe and search for new or disappeared nodes. Thus a decentralized administration is realized, which dynamically adapts to the currently available peers. These possibilities of dynamics, scalability and reliability of P2P networks are suitable for realization of pervasive and durable services.

After a short introduction, this abstract describes the implementation of a Pastry[4] API on top of the EventChannelNetwork (ECN)[1] and discusses the protocol conformity and integration. The implementation is build within the FreePastry-API[2] and it shows the reusability of existing applications with an preexisting library example in section 3. An examination of the resulting complexity and further possiblities to reduce these costs cover the following part and finally ends in the summarizing conclusions.

1.1 Pastry: The Specifications

Within the domain of embedded systems, networks usually use a many-too-many communication scheme according to the broadcast principle. A passing of messages over several point-to-point links is not necessary. Depending upon the kind of service, certain response time or data rates must be hold and requires real time guaranties. Pastry is a concrete P2P-API and it is a common standard, which makes it possible to reuse code of existing applications. The transport of messages within the network takes place in the submodule Sockets of the FreePastry library and is based on a point-to-point communication. Each node has a unique identity, which enables abstraction from IP-based addressing. An internal forwarding algorithm controls the routing of messages via multiple nodes in decentralized way. If a node receives a message addressed to another node-ID, it forwards this on the shortest path to the addressed node. The number of forwarding steps is limited to O(logN) in a network with N nodes. Each Pastry node observes a certain number of next neighbors within the node-ID area and considers the distances (measured by the physical connection) between individual nodes.

1.2 Why P2P over RMI resembles the tower of Babel

Remote method invocation (RMI) communicates by means of direct point-to-point connections. Such a remote method call is coupled to a unique object reference and blocks the calling thread.

But the bigger disadvantage is the protocol stack which is necessary to communicate with RMI as an interface layer for a Peer-to-Peer network. Following figure shows this resulting stack.

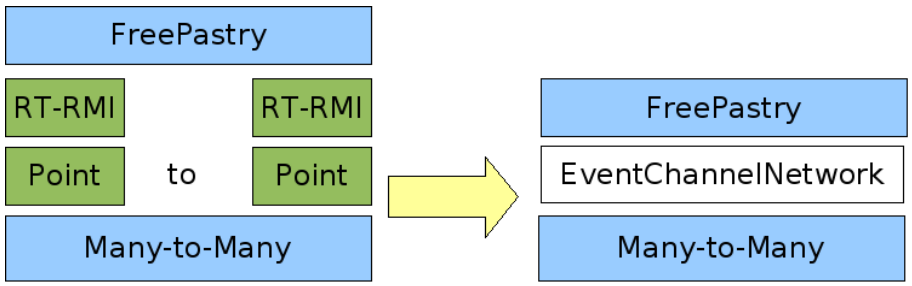

Figure 1: Comparison of a RMI- and an ECN-based Pastry communication

RMI uses a point-to-point oriented transport protocol, which is resource wasting and complex in its implementation, regarding the fact, that within embedded systems broadcast oriented networks with many-to-many communication are used. P2P protocols like Pastry take place upon this communication layer and build again many-to-many communication to offer the features of dynamics, scalability and durability.

1.3 Benefits of the Real-Time-Event-Service as backend for a P2P-API

The real-time publish/subscribe event-service (EventChannelNetwork, ECN) in HIJA[3] uses anonymous many-to-many communication based on logical or virtual event channels. Due to the broadcast transmission of messages, a complex routing among nodes is not necessary. Using asynchronous communication, a sending node is not blocked and thus can handle other tasks in the meantime. A event channel builds on top of as many as desired physical networks. These channels can be created and administrated dynamically and handle events with publish/subscribe logic. The ECN exists in two versions, one with hard real time and one with flexible real time abilities. The flexible ECN offers error handling with Java exception handling injury of real-time warranties or communication failures. The ECN provides a priority based scheduling that controls the transport of messages, as well as their processing in time. Necessary extensions for the implementation of a Pastry API based upon the event-services node or system-ID there is the need for a protocol to discover and control of other nodes. To interconnect two different physical networks in our logic channel a forwarding functionalilty of the EventChannelNetwork is used. But a Forwarding gateway between the logical event channels has to be realized. As prototype for the commonly in the area of embedded systems used broadcast networks or fieldbuses, the UDP/IP protocol will used as a communication backend for the flexible ECN. Due to its platform independence, the Event-Service can be easily adapted to another physical network and offers a wide range of utilization of the Pastry-API. Furthermore it avoids disadvantages of RMI-based protocol stack.

2 Pastry-Protocol-Conformity

Due to the aspect of reusability of applications there is the need to smoothely integrate the EventChannelNetwork interface into the existing program-library (API). As an Open-Source-Implementation FreePastry was used in its Version 1.4.2. The Library already contains a network backend for the Internet based Socket communication, as in earlier version, there was also an RMI backend implementation. To implement an EventChannelNetwork network, the same integration schemes were used, to take advantage of the existing infrastructure. Figure 2 shows the location of the new network backend.

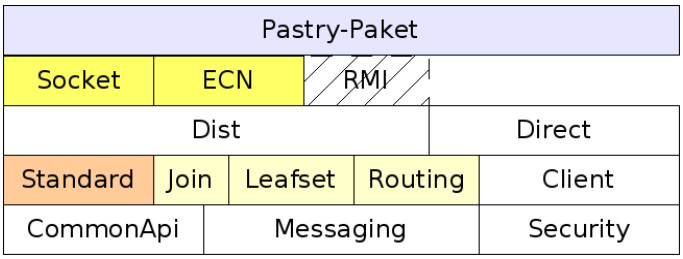

Figure 2: The packages of the FreePastry transport middleware with ECN interface

2.1 Location properties

A point-to-point based communication results in a star-like topology, regarding the subjective view of a pastry node. Switching to a broadcast oriented structure there exist differences concerning the need of a routing function, which is even distinct in two possibilities. One is a Pastry network within only one logical EventChannel or to have multiple EventChannels interconnected to a large network with an additional hierarchie of distance. The following figures 3 and 4 illustrate the resulting structures for a Point-to-Point- and an ECN-based Pastry network.

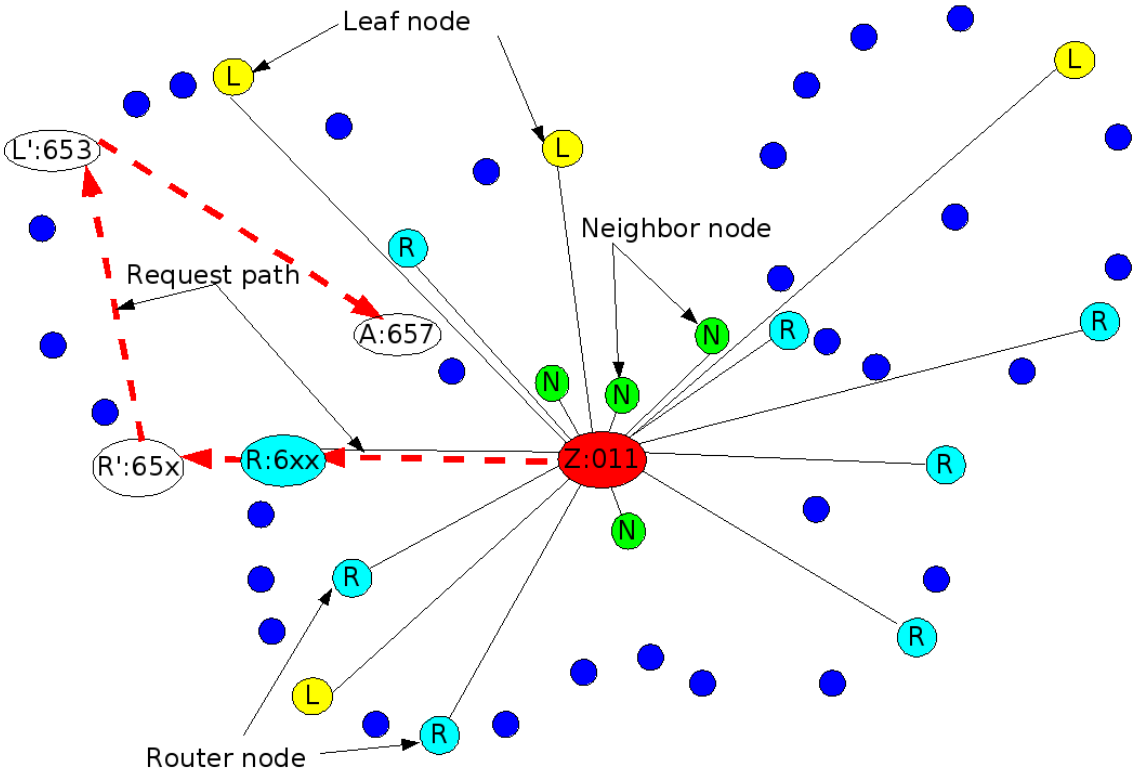

Figure 3: The Pastry network topologie with point-to-point-connections

If examining the EventChannelNetwork based case, the most significant fact is the vanished necessacity of routing within a single logical channel. To come up the locality differences and the routing to another event channel, the nodes ID is generated on the EventChannels unique name or ID number. Computing the Hash function on the EventChannels name as the node IDs prefix and filling the rest of the 128 bit node key with a random number builds the unique identifiacation key for a Pastry node. This approach is done analogous to the standardly used IP and Port based

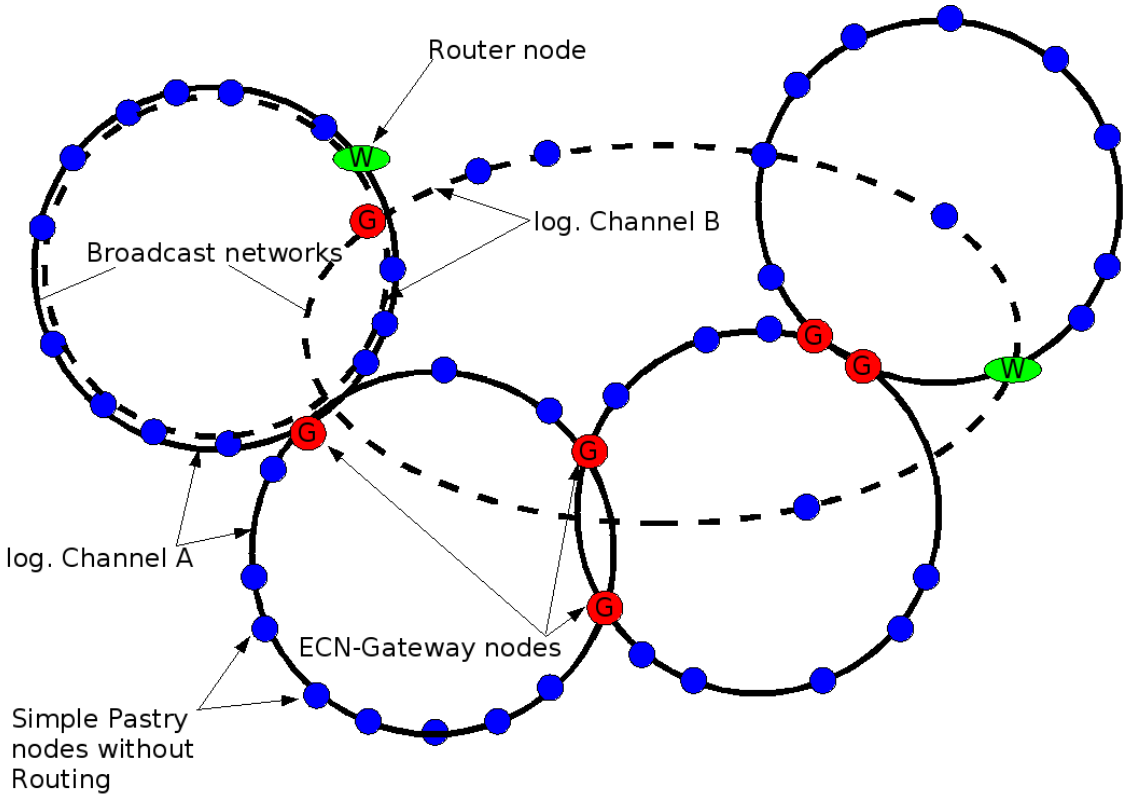

Figure 4: Possible structure for a Pastry network upon the EventChannelNetwork

ID generation.

Another difference points on hierarchisation of nodes. Only a small number of „super" nodes connect between different logical event channels. The other more significant part of „simple" nodes do not need a routing table and thus leave a smaller ressource demand for the underlying hardware.

2.2 API-Integration

The following section introduces the API conformity within the FreePastry library. The figure 5 shows the abstract interaction of a Pastry application (the right most part) through protocol layer, communication layer and the EventChannelNetwork interface layer.

As the interface between the communication and the protocol layer there are three important classes. The ECNPastryNode is the central logical point of the Pastry network. This class inherits from the class `rice.pastry.dist.DistPastryNode` and gives access to the contained data structures and the send/receive functions. The pre-existing software technical observer-observable-pattern (in `MessageDispatch` and `MessageReceiver`) is reused to organize the local message delivery. The EventChannelNetwork backend functions are totally encapsulated within `ECNManager`. Within the `ECBackend` class conversion between Pastry's `Messages` and the event service like `RemoteEvents` takes place.

2.3 Adaption to the many-to-many communication

Using many-to-many connection renders the routing unnessecary. But when switching from the point-to-point scheme to a broadcast based communication the problem of responsibility plays an important role. To avoid chain reactions an efficient message dropping on the request side is necessary. Due to the fact, that such a dropping is protocol or application specific, the three

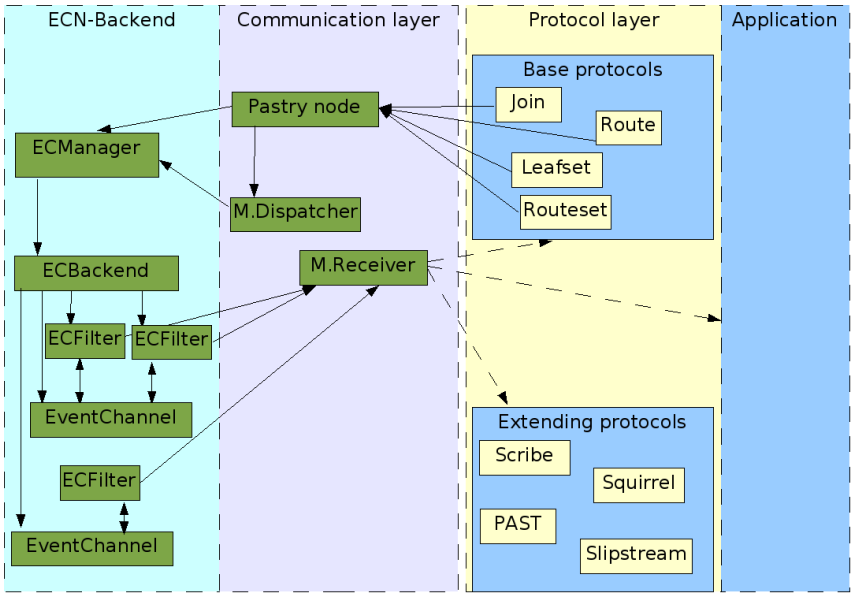

Figure 5: Interaction of FreePastry with the ECN-API

FreePastry protocols for the Leaf and Route set, and the Routing algorithms have been changed. Only the server side of the communication is adapted to the altered communication background.

3 Example

This part shows the easy reuse of existing applications. The example taken into account exists in `rice.pastry.testing.DistHelloWorld`. This examples shows the initialisation process of the Pastry-Network and is followed by a sending of random addressed messages to show the routing functionalities. The needed modifications to use an EventChannelNetwork based connection in contrast to the network protocol used as standard are the following parameters. First it is necessary to instruct the PastryNode-factory to use the EventChannelNetwork backend, which is specified by a `PROTOCOL_ECN` in the step of its initialisation. Due to the fact, that it is not possibly to use the IP-address for a random Pastry-Node-ID, an ID is generated using the EventChannel's name as input for the Hash-function. For a Pastry network taking place within only one EventChannel, also a complemete random-based node-ID is possible.

```
nidFactory = new ECNNodeIdFactory(environment);
factory = DistPastryNodeFactory.getFactory(nidFactory, PROTOCOL_ECN, port, environment);
```

In the above code snippet, the initialisation of the node-ID and the node factory is shown. The parameter `environment` specifies global parameters and also contains the EventChannelNetwork specific constants. The parameter `port` is not used in the case of the specified `PROTOCOL_ECN` due to the fact, that the EventChannelNetwork uses a variety of connection possibility and thus special parameters. This description takes place in a XML-file, read out in the initialisation phase of the EventChannelNetwork. The other parts of the example code remained untouched due to the realized API-comformity.

4 Evaluation

4.1 Complexity

To illustrate the reduced cost of messages, the node join is described for the case of an ECN-based communication. For simplicity, the case within only one EventChannel is described. The nodes A

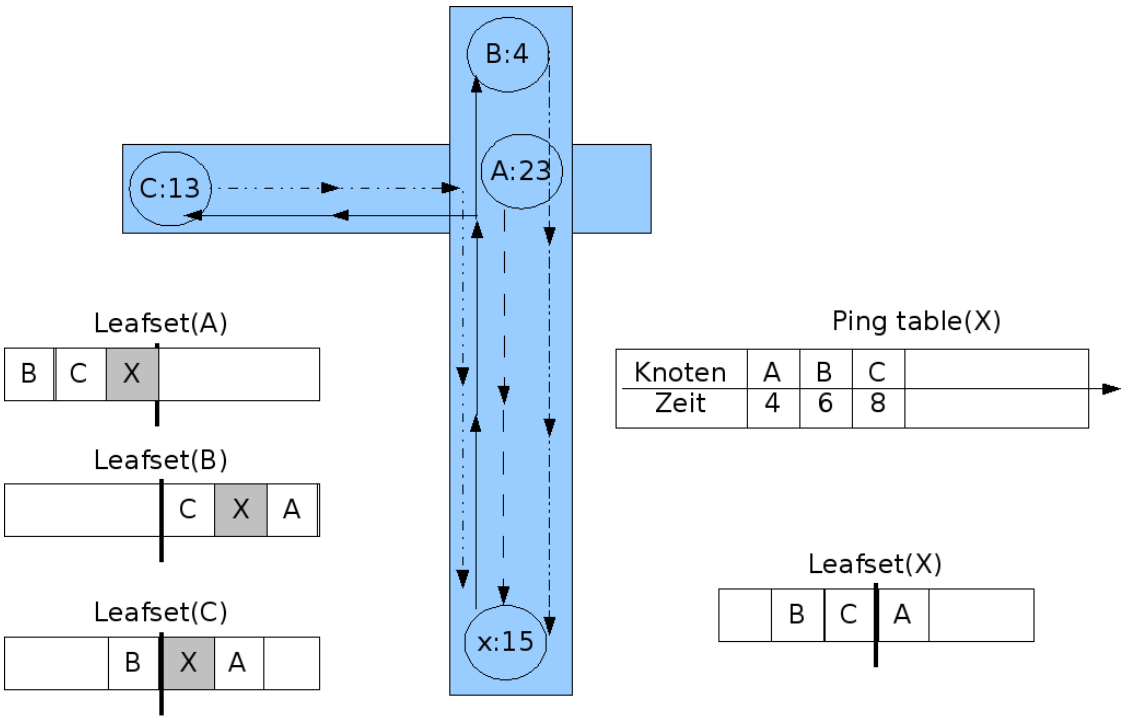

Figure 6: Example for a finished joining of Pastry node X

to C are already a part of the network and contain a valid Leaf set also shown in the figure. The node X has joined the Pastry network and thus also contains a Leaf set and a Ping-table, which is omitted for the nodes A to C for a clear arrangement. While the arrow symbolize the route of the requests, the dotted drawn ones mark the responses.

In contrast to point-to-point based „join" with $3 * log(n)$ messages, which depends on the count of nodes (n), only the table of the Leaf set has to be filled. This results in $L + 1$ messages, where L is the size of the new nodes Leaf set. The list of the nearest neighbors is computed from the local request-response-time difference and thus omitts extra messages.

Following dump shows the three steps to join the Pastry network shown in figure 6.

1. The node X connects with the EventChannelNetwork
2. A „Join"-request is started by sending a `NodeIDRequestMessage` in the EventChannel. Subsequently a row of messages is received by the joining node. These are listed in the following table:

Time	sent	received(sender node)
0	node-ID request(X)	-
4	-	node-ID response(A)
5	-	leaf nodes(A)
6	-	leaf nodes(B)
8	-	leaf nodes(C)
9	leaf nodes(X)	

Due to the sequence and the point of time the answers are received at a local node, it is easy to build the distance table for the nearest neighbors. The received Leafset nodes will only be included in the local one, if they are suitable.

3. The self-introduction finally makes the node a full partitipant of the network and allows a sending of messages. This introduction is done by sending the local Leafset into the EventChannel.

In the routed case with different logical channels, only for routed „Super-nodes" a exchange of the Routing table is necessary. This additional cost is bounded to the count of EventChannels in the network.

4.2 Protocol conformity vs. Optimization

In the previous section the node join was done in a optimized way, abstracted from the normal FreePastry „Join" cycle. As mentioned before, the neighbor proclamation, as for non-routed nodes also the routing information can be omitted, thus reducing the count of messages by a minimum of factor two. But to interconnect to non-ECN based Pastry network, these messages have to be sent. Due to the fact, that the FreePastry protocol moduls are registerable at run-time a Pastry point-to-point to a Pastry-ECN Gateway is possible. Whereas an efficient communication in many-to-many networks demands an adaption of the server functionalities of the used protocol.

5 Conclusion

This project shows, that it is possible to build a Peer-2-Peer network upon the EventChannelNetwork-Communication-API. Furthermore it was achieved to conserve the existing protocol specification, which allows a wide portability of the Pastry based application. Due to the possibility to reduce the ressource demand, it is possible to adopt it for very specialized applications and hardware plattforms. But to achieve these, the preceeding analysis of the planned deployment is essential. These take into account:

- length and generation of the node:
  Determines the maximum addressable participants and its location properties.
- size of the interconnection tables of nodes:
  A more detailed hierarchisation of Pastry nodes in „Super nodes" and „Simple nodes" is possible.
- size of the observation tables of node:
  These also address the hiercharchisation but within the durability of the service.

Analogous to the implemented Pastry-Middleware protocols, optimized realisations for extending modules like PAST or Scribe are possible. Especially Scribe could directly take advantage of the EventChannelNetwork's publish/subscribe logic to reduce the efforts of building this service on top of pastry. As the promising complexity reducings lead to the further work to test and to examine the resulting performance in real-world examples, as for real-world applications.

References

[1] Eventchannelnetwork, Dezember 2005. http://www.eventchannelnetwork.org.

[2] Freepastry internet ressourcen, November 2005. http://freepastry.rice.edu/.

[3] Hija - high integrity java, November 2005. http://www.hija.info/.

[4] Pastry Internet Ressourcen, Dezember 2005. http://research.microsoft.com/~antr/Pastry.