OPAX - An Open Peer-to-Peer Architecture for XML Message Exchange

Bernhard Schandl, University of Vienna
bernhard.schandl@univie.ac.at

Users wishing to find multimedia material about interesting events from all over the world are confronted with the problem of efficiently locating and retrieving media that meets their personal interests. One possible solution for this problem are peer-to-peer based networks to publish and broadcast descriptions of such events; however, such networks require an efficient and reliable infrastructure. Because of its properties, the *hypercube* concept allows efficient broadcast of event notifications and thus may serve well for such networks. In this paper we explore and discuss the quality of the hypercube concept as peer-to-peer topology. We provide a fully distributed reference implementation and analyze its strengths, weaknesses, and robustness, and compare it to one possible alternative, a *hybrid hypercube* network. Our analysis indicates that maintenance of a fully distributed, peer-to-peer based hypercube is questionable: In an environment where participants may unexpectedly enter and leave the network at any time (e.g. a cell phone leaving the area covered by the radio transmitting station), inconsistencies may cause the system to run into undefined states.

1 Motivation and Contributions

Events of user interest happen all over the world, at any time. More and more, multimedia material about these events becomes available in digital form, published by different providers through various channels. Also, more and more users become able to receive and consume such multimedia material through various devices, including PCs, PDAs, mobile phones, and DVB set-top boxes. The evolution of technology makes it easy to publish multimedia content, but the amount of available material makes it hard to find material which meets the users' requirements.

One way to overcome this problem could be a middleware which allows for automated filtering of multimedia events, based on data provided by the event publishers, and incorporating user-defined selection criteria. MediÆther [1] is a multimedia event space wherein content providers place descriptions of interesting events along with metadata

(topic, time, location), describing the digital representation of the event as well as the event itself. Based on these descriptions, users are informed about published events if they meet their interest profiles.

MediÆther was designed as a peer-to-peer network, allowing each peer to publish, search for, and register notifications for multimedia events. Because high amounts of event packages have to be broadcast in order to notify every participating peer, a network topology which allows the implementation of an efficient broadcast algorithm is required. Additionally, the network must be able to cope with high member fluctuation and heterogeneous peer types: mobile devices, like cell phones or PDAs, may join or leave the network at any time.

The diploma thesis "*OPAX - An Open Peer-to-Peer Architecture for XML Message Exchange*" analyzes the adequacy and practicability of the *hypercube* topology concept, as presented in [2], for constructing, operating and maintaining peer-to-peer networks, especially such ones designed for broadcast of messages (like MediÆther is). The hypercube – as a way to arrange participants of a network – is well-known in the field of multiprocessor systems: As it allows message transportation between any two participants in at most $\log N$ steps, it is an infrastructure to implement efficient (in terms of messages sent) broadcasting, thus fulfilling a crucial requirement for appplications like MediÆther.

In the course of the diploma thesis, the following efforts were made to evaluate the suitability of the hypercube concept in the context of heterogeneous media networking:

1. Detailed discussion of the hypercube topology concept
2. Design and implementation of a topology- and platform-independent peer-to-peer messaging infrastructure (*OPAX*)
3. Design and implementation of a fully distributed hypercube topology within OPAX
4. Detailed analysis of the suitability of the fully distributed hypercube topology for a multimedia event space

Special attention has been paid to analyzing the robustness of the network: Sudden exit of a node during a topology modification or a message broadcast may lead to inconsistent topology information and may cause the network to be fragile, exhibit gaps, or – in the worst case – to collapse. Two directions for solving these problems were explored:

5. Implementation and evaluation of improvements and modifications of the fully distributed hypercube
6. Design and implementation of a hybrid hypercube network

Both of these approaches do not lead to the desired result: While some of the problems can be solved by extending or modifying the hypercube algorithms (work package 5), it is always possible to produce situations that can not be captured adequately. Enriching

the P2P-based messaging with a centralized topology management (work package 6) empowers the manager to satisfactorily handle such situations. However, it violates the basic ideas of P2P networking (no peer being in a more prominent situation than others), and introduces the disadvantages of server-based concepts (single point of failure, performance overload, scalability problems, etc.).

The disadvantages of the proposed solutions lead to the conclusion that the hypercube paradigma in its current form is not applicable for an environment like MediÆther, where participants may fail randomly.

The remainder of this outline is organized as follows: In section 2, the OPAX framework and a Java-based prototype implementation are presented. Section 3 analyzes the distributed hypercube as a network topology, discusses its strengths and weaknesses, and introduces possible extensions to make it more resistant against failures. A hybrid network using a centralized topology manager is presented in section 4.

2 OPAX - A P2P Platform

In the course of this diploma thesis, an open P2P platform was designed to serve as an environment for exploring, implementing, and testing P2P network topology concepts. Its design principles include:

- *Openness.* To allow the integration of and communication between heterogeneous systems and networks, the framework should be open in a way that allows an implementation on various platforms.
- *Extensibility.* As requirements to a P2P network may change over time, it is important to allow the framework to grow with its requirements, without the need to reject previous efforts.
- *Modularity.* To satisfy different applications' requirements, the main components of the framework – esp. the topology manager component – should be replaceable or switchable by configuration without the need to restart a running instance.
- *Convenience.* The framework should provide an easy-to-use API to be used by the application, providing the following operations: *peer creation*, *opening/joining* of spaces, *message broadcast*, *message receipt*, *departure* of spaces, and *peer destruction*.

The basic architecture concept chosen for the framework is depicted in Fig. 1.

An application may create multiple peers, called *local peers*, running in parallel. Using a local peer, an application may open or join spaces. Each instance incorporates components working independently from the corresponding components of other local peers. These components include *configuration management*, *logging*, *synchronization*, and *topology managers*. Through the *network communication* component, peers may communicate with other peers.

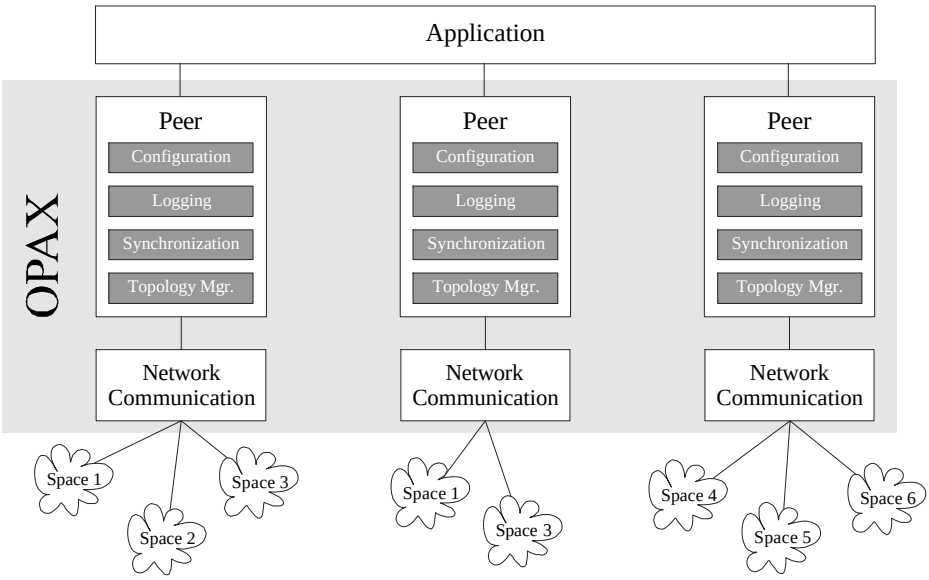

Figure 1: OPAX architecture

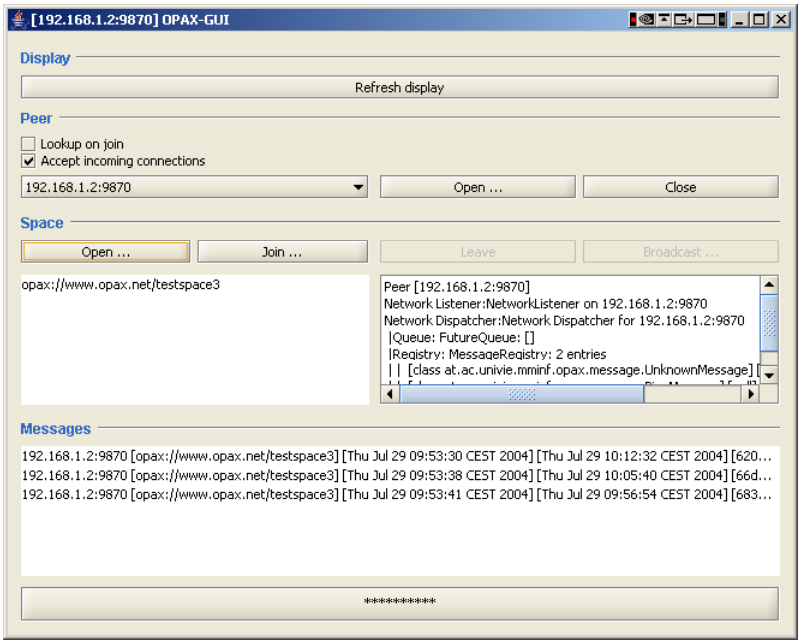

Figure 2: OPAX demo application

OPAX defines a simple message transmission protocol, as well as a set of message types, whereof the most important ones are `Application` and `Topology`. While the former is used to broadcast application data through the network, the latter is used to manage the network topology structure. Both of them are designed to be fully adaptable to the application's requirements: `Application` messages are able to incorporate complete XML documents, which allows for the transfer of any application data; `Topology` messages may be sub-typed and enriched with arbitrary properties.

A prototype of OPAX has been implemented in Java. Its classes provide API calls and callback entry points for the application, hiding the implementation details of the network layer. The application may adapt the OPAX subsystem using an open configuration interface. Different topology implementations, which can be realized by implementing an interface `TopologyManager`, may be registered to the OPAX subsystem and then used as neccessary.

As one part of the prototype implementation, a demo application allows to open, join, or leave networks, and to broadcast and receive application messages. The user may control the internal details of the OPAX components in order to perform testing and debugging. A screenshot of this application is depicted in Fig. 2.

3 The Hypercube as a Network Topology

In the hypercube network topology, peers are located at the nodes of a hypercube graph, and peers are linked directly along the edges of the graph. Figure 3 shows hypercubes of dimensions $d = 0 \ldots 4$, with two possible depictions of a 4-dimensional hypercube.

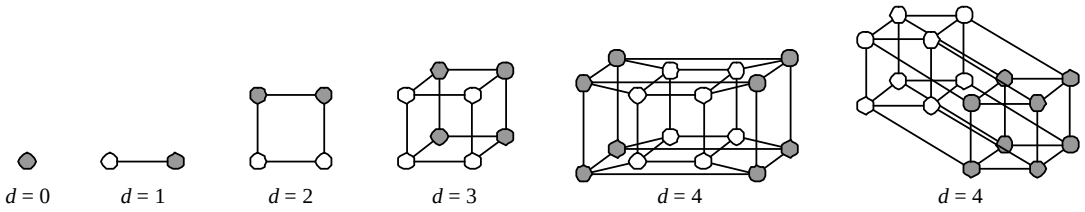

Figure 3: Hypercubes with 0 to 4 dimensions

The hypercube may represent an adequate topology for a symmetric peer-to-peer network, as it has the following properties:

- *The maximum Hamming distance between any two nodes (the network diameter) in the hypercube is d, with $d = \log N$, and N being the maximum node capacity of the network.* This implies that every message does, presuming an appropriate routing algorithm, in no case require more than d hops to reach its final destination node.

- *The network structure is symmetric.* In terms of the network's topology, no node incorporates a more prominent position than others, which is crucial for load balancing in the network: Every node can become the source of a broadcast (i.e. the

root of a spanning tree of the network), yet the load of a broadcast will always be shared equally.

- *The topology allows to create an optimal spanning tree in terms of messages sent.* Crucial for an efficient broadcasting algorithm, it is always possible to create a spanning tree, starting with any node of the hypercube, so that the maximum number of message transmissions (the edges in the spanning tree) is $N - 1$, and every node receives the message exactly once.

- *The topology provides redundancy.* The *connectivity* (the minimum number of nodes to be removed in order to partition the graph) is optimal, i.e. equal to $\log N$.

- *The network is scalable.* The number of dimensions of the hypercube can be chosen to adjust the network diameter according to the requirements of the application. Additionally, if the network is designed so that peers may cover multiple positions simultaneously, the topology allows the creation of networks with the number of participants ranging from 1 to 2^d.

However, the highly organized hypercube topology requires participating peers to minutely follow a well-defined protocol, which manages data transfer in the course of integration and departure of peers. Peers must communicate repeatedly during this protocol to exchange the neccessary data about neighbour peers and their positions. When a fully-distributed hypercube network is implemented, significant drawbacks of this high communication effort become apparent. Two major issues have been discussed in the report in detail:

- *Algorithmic complexity.* Particular calculations that network nodes have to perform during topology modification (i.e. integration or departure of nodes) are of complexity $O(N)$ resp. $O(2^d)$, with N being the maximum number of nodes in the network and d being the dimensionality of the hypercube.

- *Peer and link failures.* A sudden failure of a peer or a link between two peers may lead to the loss of messages. An abortive delivery of messages may cause peers to remain in a state where they are no more able to continue their algorithmic computations.

To resolve these issues, we present improvements and modifications to the original algorithms and protocols. Some of them have been implemented to prove their functionality, others have only been discussed in theory.

- *Dimension increase.* To address the problem of algorithmic complexity, the hypercube dimensionality is selected dynamically in a way that it is always high enough for the hypercube to accommodate the actual number of nodes, but as low as possible so that no dimensions remain "unused".

- *Topology modification rollback.* To prevent the emergence of deadlocks caused by peers or links failing during an topology modification, peers being in a transitional state may fall back to the "normal" state after a certain period of waiting for an acknowledgement message (timeout). However, this mechanism causes inconsistent peer state combinations because peers may fall back independently and delayed.
- *Stale peer ignoring.* Albeit not actually being a solution to the problem of peer failures, ignoring peers in this context means that a failing peer is ignored for a certain time and is then dropped from the topology. However, this causes major changes in the algorithm for message broadcast since the stale peer must be bypassed in order to ensure that all other peers receive broadcast messages, introducing another level of complexity into the broadcast algorithm.
- *Peer watch.* To immediately detect peer failures, a monitor peer is assigned to every member of the network. In the case of a peer failing, the monitor peer performs the departure protocol on behalf of the lost peer. Thus, it is possible to keep the topology in a consistent state. However, the peer watch concept is ineffectual when a peer and its monitor fail simultaneously.
- *Global shutdown and topology reconstruction.* In this approach, whenever a certain number of peers have failed, the topology is discarded and then subsequently recreated "from scratch". With this approach, problems arise when multiple peers in different parts of the hypercube fail.
- *Implicit coordinates.* In the hypercube, a node joining a hypercube network is assigned a position (identified by coordinates) depending on the current arrangement of nodes in the network. "Implicit coordinates" means that a node can determine its position by itself, independent of the current situation or the moment of its joining. This would eliminate the need for complex joining and departure protocols and ease the recovery of peer failures.

Although some of the mentioned approaches led to improvements concerning network stability, it was at all times possible to find a failure constellation which could not be absorbed in a way that allowed the network to continue to exist, and be both usable (in terms of application message broadcast) and administrable (in terms of topology modifications).

Therefore, we developed a new approach by designing a *hybrid network*: A central topology manager keeps all information about node arrangement, thus disburdening peers from the topology administration, and simplifying the topology manipulation protocols, while message broadcast is still processed in peer-to-peer manner. This approach is introduced in the following section.

4 A Hybrid Hypercube

To overcome the problem of distributedly managing a topology infrastructure, we introduced a *hybrid network*: Topology management (i.e. the management of integration and

departure of peers) is performed by a *topology server*, while messaging is processed in the same manner as in the pure P2P variant.

To fully represent the hypercube topology, a tree was selected as data structure for the topology manager. This structure (*i*) is formally equal to a hypercube, (*ii*) can adapt to an arbitrary number of dimensions, and (*iii*) can be easily divided, which allows to distribute the management among multiple servers.

The disadvantage of the tree structure is its memory consumption: To represent a d-dimensional hypercube, data about 2^d nodes and $2^{d-1} \cdot d$ edges has to be stored. To address this problem, we can construct a *reduced hypercube*, where the number of edges is limited to a fixed value for all dimensions $d_i \geq d_{red}$. Clearly, this data structure is not suitable to hold a full representation of the hypercube. The reduced capacity of higher-dimensional levels allows only the representation of a *degenerated hypercube*, i.e. a hypercube wherein not all edges are present. Two degenerated hypercubes are depicted in figure 4.

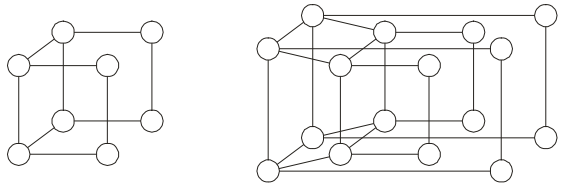

Figure 4: Two degenerated hypercubes

It is obvious that for such a hypercube, the broadcast algorithm must be reformulated. In the diploma thesis, a modification to the original broadcast algorithm is presented: The modified algorithm is no more optimal in terms of messages sent, as every missing edge of the cube must be bypassed.

5 Conclusion

In this paper, we defined the requirements for a multimedia event space network, where participants publish events and users get notified about these events based on their personal interest profiles. To allow for efficient broadcast of event notifications, a hypercube may be used as the topology of a network infrastructure: This particular topology's properties provide the basis for efficient search and broadcasting algorithms. To explore and analyze these properties, a peer-to-peer framework has been designed and implemented: OPAX provides a messaging infrastructure to build P2P networks, using any topology.

However, the hypercube approach has significant disadvantages in the field of reliability: the fully distributed implementation of the hypercube is not able to cope with sudden peer failures or exits, which are commonplace in an environment of heterogeneous mobile devices. The proposed extensions and modifications of the hypercube concept improve the stability of the network, but it is always possible to precipitate failure situations which cannot be covered by the topology management. A different approach, a

hybrid hypercube, may satisfactory handle these situations, but violates the basic ideas of P2P networking.

The results of this thesis can be seen as a starting point for further research in the areas of peer-to-peer topologies (esp. hypercube and related types) and as origin for further investigating methods to stabilize, yet optimize ad-hoc peer-to-peer networks.

References

[1] Boll, S. and Westermann, G. *MediÆther - an Event Space for Context-Aware Multimedia Experiences*. In: Proceedings of International ACM SIGMM Workshop on Experiential Telepresence (ETP'03), Berkeley, USA, 2003.

[2] Schlosser, M. *Semantic Web Services*. Technical Report, University of Hannover and Stanford University, 2002.

The full reference list can be found in the appendix of the diploma thesis.