21. Hybrid Peer-to-Peer Systems

Vasilios Darlagiannis (Technische Universität Darmstadt)

21.1 Introduction

Peer-to-Peer systems have been receiving considerable attention from the networking research community recently. Several approaches have been proposed as communication schemes in order to supply efficient and scalable inter-peer communication. These schemes are designed on top of the physical networking infrastructure as *overlay networks* taking advantage of the rich flexibility, which is accomplished at low cost. A number of important design approaches has been already presented in previous chapters. Their topologies and operation mechanisms influence greatly the performance of routing and topology maintenance algorithms and hence, the efficiency of the corresponding Peer-to-Peer system.

However, Peer-to-Peer systems are distributed systems with a large number of non-functional requirements such as scalability, dependability (including fault-tolerance, security, integrity, consistency), fairness, etc. These requirements should be met in order to design systems, which are easily deployed on top of the Internet while making use of available resources in an optimal way. Most approaches have been designed to deal with a subset of these requirements. Nevertheless, they have intrinsic limitations fulfilling the complete set of the aforementioned requirements. In most cases, trade-offs in meeting these requirements exist, thus, raising severe constraints.

To elaborate further the aforementioned trade-off issue, we consider the design of a Peer-to-Peer system where fault-tolerance should be supported in the presence of heterogeneous environments (peers may have different physical capabilities and behavioral patterns). In the context of Peer-to-Peer systems where peers represent unreliable components, fault-tolerance is achieved mostly by the employment of redundancy and replication mechanisms. *Pure* DHT-based approaches such as Chord [576] or Pastry [527] suggest a large number of neighbors that usually increases logarithmically with respect to the size of the system. While it has been shown that such approaches provide high fault-tolerance [385][1], they ignore practical limitations raised by peers of low physical capabilities that may not fulfill the continuously increasing requirements as system's size expands. In addition, by ignoring het-

[1] That study assumes a Peer-to-Peer system where both peers' inter-arrival and service (lifespan) time distributions follow the Poisson model. However, as it has been empirically observed in many studies (e.g. cf. [98]) that peer lifespan follows a different distribution.

R. Steinmetz and K. Wehrle (Eds.): P2P Systems and Applications, LNCS 3485, pp. 353-366, 2005.

erogeneity and dealing equally with each peer, the system maintenance cost increases significantly, while the least reliable peers contribute minimally in the fault-tolerance of the system. Further, similar requirement trade-offs (i.e. anonymity versus efficiency, heterogeneity versus load-balance, etc.) appear when pure design approaches are selected.

In this chapter we investigate Peer-to-Peer systems, which follow a hybrid design approach. The word *hybrid* is used in many disciplines such as in biology, in sociology or in linguistics. In general, it is used to characterize *"something derived from heterogeneous sources or composed of incongruous elements"* (Oxford Dictionary). Though initially the term *"hybrid Peer-to-Peer system"* was used in the context of Peer-to-Peer systems to describe approaches that combined both Peer-to-Peer and Client/Server aspects, its usage was broadened to cover further combinations of heterogeneous approaches.

In general, hybrid systems are claimed to be intrinsically better than pure approaches, mostly because of the great heterogeneity observed in deployed systems. They allow for the synergistic combination of two techniques with more strengths and less weaknesses than either technique alone.

In the remaining of this chapter we investigate and define a coarse-grained classification scheme for the observed topologies of the most important, state-of-the-art, hybrid overlay networks, their underlying mechanisms and the algorithms employed to operate on them. Then, we discuss their benefits and drawbacks in a general system-unaware way that does not consider specific Peer-to-Peer systems, where hybrid approaches are compared with non-hybrid approaches.

21.2 Overlay Network Design Dimensions

In order to meet the critical set of the aforementioned (and possibly additional) non-functional requirements for the operation of the Peer-to-Peer overlay networks, a great variety of approaches have been proposed. Analyzing the design mechanisms that characterize the Peer-to-Peer overlay networks, three major design dimensions can be identified to classify the proposed systems (cf. Figure 21.1). An alternative three dimensional approach is presented in [154].

Overlay networks vary in their *structural* design from *tightly structured* networks such as Chord [576] or Pastry [527] to *loosely structured* ones such as Freenet [124] or Gnutella [251]. This design dimension is graphically depicted in the projected axis of the design space in Figure 21.1. Tightly structured (or simply *structured*) overlays continuously maintain their topology, targeting to a "perfect" structure (e.g., a hypercube or a butterfly topology). Structured topologies may require high maintenance cost especially in the presence of high churn rate. Also, they deal uniformly with the shared

objects and services provided by the system and they are unaware of their query distribution, a fact that might cause a significant mismatch. Moreover, *Distributed Hash Table* (DHT) based approaches (which is the most common mechanism to build structured overlay networks) cannot support easily range queries[2]. Alternative investigations include several mappings of local data structures to distributed network topologies, such as tries [230] or modifications of traditionally used topologies such as hypercubes [541], butterflies [399] and multi-butterflies [155].

On the other hand, loosely structured (or simply *unstructured*) overlays do not aim to reach a predefined targeted topology, but rather they have a more "random" structure. However, it has been observed that certain connectivity policies (i.e., preferential attachment) may emerge their topology to power-law networks or networks with small-world characteristics. Unstructured topologies are typically inefficient in finding published, rare items and the embedded searching operations are in general considerably costly in terms of network overhead (most approaches use flooding or at best, selective dissemination mechanisms[398]). The observed power-law topology, though it provides a graph with a small diameter[3], it distributes unevenly the communication effort and introduces potential hot spots at these peers with high degree playing the role of a "hub". However, in scenarios where the query distribution is non-uniform (i.e., lognormal, Zipf) unstructured networks may be designed to operate efficiently.

Further, overlay networks may vary on the *dependency* of the peers on each other, as it is shown in the vertical axis of Figure 21.1. Approaches such as Chord or Freenet treat all of the participants equally and they are referred as *pure* or *flat* Peer-to-Peer networks. On the other hand, *hierarchical* approaches such as Napster [436] or eDonkey [185] separate the common overlay related responsibilities and assign the majority (or all) of the tasks to a small subset of (usually) more powerful nodes only (e.g. for resource indexing). These subset of peers is usually named as *"servers"*, *"super-peers"* or *"ultra-peers"*. The fault-tolerance of flat approaches is considerably higher than the hierarchical ones since failures or attacks to any single peer do not have significant consequences. However, such approaches do not deal well with the heterogeneity of the participating peers both in terms of physical capabilities and user behavior. The complexity of flat approaches is usually higher compared to the hierarchical counterparts. On the other hand, hierarchical solutions require a certain infrastructure to operate and may be controlled by third parties easier than the non-hierarchical alternatives. The operational load is unequally balanced among the networked entities and high dependency exists among them.

[2] Range queries are queries searching not for a single item that matches a specific key but rather for a set of items, which are "close" to a description based on e.g. metadata.

[3] Small diameter is a desirable feature for a network topology in order to reduce the maximum number of hops required to reach any destination in the overlay.

Finally, overlay networks can be designed either following *deterministic* or *probabilistic* (e.g., based on Bloom filters [77]) approaches. The selection of the former or the latter approach may improve the accuracy or the efficiency of the Peer-to-Peer systems, respectively. Also, a mixture of these mechanisms may provide improved results. A characteristic example demonstrating both probabilistic and deterministic mechanisms is OceanStore [368]. Deterministic approaches provide repeatedly similarly consistent results (as long as there are no critical intermediate changes in the system) and the provided operations can be well predicted and their cost is upper bounded. On the other hand, probabilistic approaches tolerate some unpredictability on the provided results, aiming to operate at a much lower cost than their deterministic alternatives. Such variation is shown in the horizontal axis of the overlay network design space in Figure 21.1.

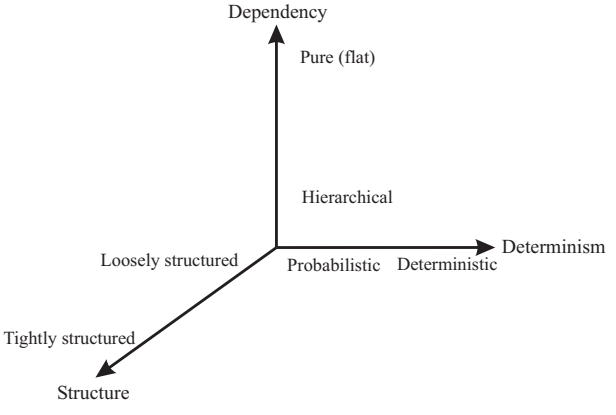

Fig. 21.1: Overlay network design dimensions

In this chapter we focus on systems that lie in the middle of at least one of the axes shown in Figure 21.1, though many of the proposed systems follow hybrid mechanisms in more than one dimensions. By doing so, hybrid designs aim to deal with the limitations of the pure approaches.

21.3 Hybrid Architectures

In this section we focus on Peer-to-Peer systems that mainly follow hybrid approaches in the architectural design of their overlay topologies.

21.3.1 JXTA

JXTA[4] [597] defines a common set of protocols for building Peer-to-Peer applications to address the recurrent problem with existing Peer-to-Peer systems of creating incompatible protocols. The main goal of JXTA is to define a generic Peer-to-Peer network overlay, which may be used to implement a wide variety of Peer-to-Peer applications and services. While JXTA offers the means to developers to design any kind of overlay network that suits to the needs of their applications, JXTA itself develops a hybrid overlay network to orchestrate the deployed applications and services. Peers in the JXTA network are self-organized into *peergroups*. A peergroup represents an ad hoc set of peers that have a common set of interests, and have agreed upon a common set of policies (membership, routing, searching, etc). However, there is a global peergroup as a bootstrap point where all the specialized peergroups can be advertised.

The JXTA specifications define the Resolver Service Protocol as a universal resource binding service. The Resolver Service is used to perform resolution operations found in traditional distributed systems, such as resolving a peer name into an IP address (DNS) or binding an IP socket to a port.

The global JXTA overlay network provides a default resolver service based on *rendezvous* peers. Rendezvous peers are peers that have agreed to index other peer advertisements to facilitate the discovery of resources in a peergroup. A peergroup can have as many rendezvous peers as required to support the size of the peergroup. Rendezvous peers are defined in the scope of peergroups to reduce the communication complexity. Any peer can potentially become a rendezvous peer, unless there are security restrictions.

Rendezvous maintain an index of advertisements published by edge peers via the *Shared Resource Distributed Index (SRDI)* service. Edge peers use SRDI to push advertisement indices to their rendezvous when new advertisements are published. The rendezvous/edge peer hierarchy allows resolver queries to be propagated between rendezvous only, significantly reducing the amount of peers that need to be searched when looking for an advertisement. Such a structure is illustrated in Figure 21.2.

Rendezvous Peers are organized into a loosely-coupled network to reduce the high maintenance cost that occurs in Peer-to-Peer systems with high churn. The JXTA approach separates the cost of a DHT solution into index maintenance, and data access. Project JXTA utilizes a hybrid approach that combines the use of a loosely-consistent DHT with a limited-range rendezvous walker. Rendezvous peers are not required to maintain a consistent view of the distributed hash index leading to the term loosely-consistent DHT. Each rendezvous maintains its own *Rendezvous Peer View* (RPV), which is an ordered list of known rendezvous in the peergroup. Inconsistency among the

[4] The name of JXTA come from the verb *juxtapose*, which means place things side by side to suggest a link together or emphasize the contrast between them.

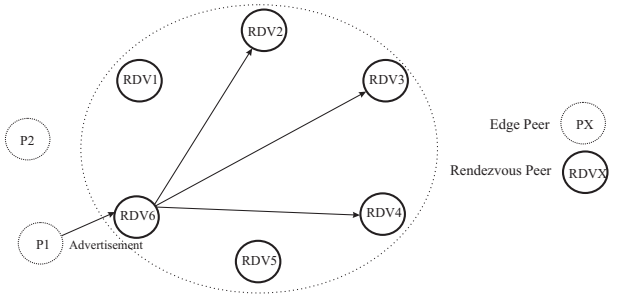

Fig. 21.2: JXTA overlay network

RPVs of different rendezvous peers might occur. A loosely-coupled algorithm is used for converging local RPV. A rumor-based technique is employed to disseminate information about the rendezvous peers. Seeding rendezvous are special rendezvous peers to accelerate the RPV convergence, as all rendezvous should know about all seeding rendezvous of a peergroup.

The hybrid approach of a loosely-consistent DHT combined with a limited range walker to search for advertisements has the advantages of not requiring a strong-consistency DHT maintenance, and is well adapted to ad hoc unstructured Peer-to-Peer networks. However, when very large peergroups are constructed requiring several hundreds of rendezvous the system may suffer considerably from the resulting inconsistency, which becomes a boomerang to the performance of the system.

21.3.2 Brocade

The majority of DHTs assume that most nodes in the system are uniform in resources such as network bandwidth and storage. As a result, messages are often routed across multiple *autonomous systems (AS)* and administrative domains before reaching their destinations.

Brocade is a hybrid overlay network proposal, where a secondary overlay is layered on top of a primary DHT. The secondary overlay exploits knowledge of underlying network characteristics and builds a location-aware layer between "supernodes", which are placed in critical locations in each AS of the Internet. Supernodes are expected to be endpoints with high bandwidth and fast access to the wide-area network and act as landmarks for each network domain. Sent messages across different ASs can be delivered much faster if normal peers are associated with their nearby supernodes that can operate as "shortcuts" to tunnel the messages towards their final destination, thus,

greatly improving endpoint-to-endpoint routing distance and reducing network bandwidth usage.

The critical aspects in designing an effective Brocade overlay are the appropriate selection of supernodes and the mappings between supernodes and normal DHT nodes. A straightforward solution is to exploit the hierarchical structure of network domains. Each network gateway may act as a brocade routing landmark for all nodes in its subdomain. An example of this mapping is shown in Figure 21.3. Supernodes are organized in a different DHT (i.e., Tapestry).

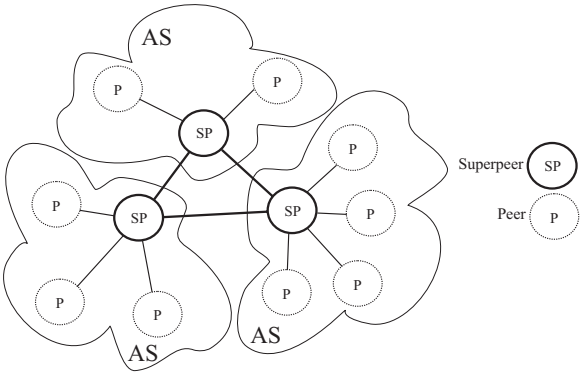

Fig. 21.3: Brocade overlay network

The routing operation works as follows. When a message reaches a supernode, the supernode may do a lookup to determine whether the message is destined for a local node, or whether brocade routing may be useful. In the brocade overlay, each supernode advertises the IDs on this list as IDs of objects it stores. When a supernode tries to route an outgoing message which should be delivered out of the local AS, it uses the supernode DHT to search for destination supernode. By finding the object on the brocade layer, the source supernode forwards the message directly to the destination supernode, which resumes normal overlay routing to the final destination.

Summarizing, Brocade is a hybrid system aiming merely to exploit the underlying network topology to supply more efficient routing services. However, the load balance of the network may be unevenly distributed among the peers. Moreover, superpeers may either act maliciously or become targets of attacks.

21.3.3 SHARK

Pure DHT-based solutions rely on hash functions, which may map advertised items to certain locations in the overlay structure (by assigning hash-generated identifiers both to each item and overlay location). Such mechanisms (while they are very efficient) are limited to single lookup queries of these identifiers. *Range* (or *rich*) search queries based on keywords remain challenging features for such systems. However, usually users prefer to specify what they are looking for in terms of keywords. For instance, a user of a file sharing application could look for a certain genre of music and not for a particular song. Additionally, multiple dimensions of meta-data are also highly desirable, for instance, looking for a document released at a certain period and related with a specific topic.

SHARK (Symmetric Hierarchy Adaption for Routing of Keywords) [421] employs a hybrid DHT solution for rich keyword searching. Its hybrid overlay structure is composed of two parts: a structured one that considers the Group of Interest (GoI) concept of AGILE [420] and several unstructured subnetworks grouping peers with similar interests. Queries are initially being forwarded in the structured part of the network to reach the targeted unstructured subnetwork. Then, they are broadcasted to the set of interesting peers that provide the matched items. SHARK is described in deeper detail in Section 17.4.

21.3.4 Omicron

Omicron (Organized Maintenance, Indexing, Caching and Routing for Overlay Networks) [153] is a Peer-to-Peer overlay network aiming to address issues of heterogeneous, large-scale and dynamic Peer-to-Peer environments. Its hybrid, DHT-based topology makes it highly adaptable to a large range of applications. Omicron deals with a number of conflicting requirements, such as scalability, efficiency, robustness, heterogeneity and load balance.

The rational in Omicron's approach is to reduce the high maintenance cost by having a small, constant number of connections and routing table sizes per peer (at least for the majority of them), while still performing lookup operations at low costs. For this reason the usage of appropriate graph structures (such as de Bruijn graphs) is suggested. However, while the small number of connections reduces the operational cost, it causes robustness problems. To address this issue, clusters of peers are formed with certain requirements on their stability over time. In order to maintain their stability, new joins are directed to the least stable clusters. When the stability of a cluster gets below a certain threshold, a merging operation takes place between the unstable cluster and a neighbor cluster.

Before we present the way de Bruijn graphs are deployed in Omicron, the topology of these graphs and the way the routing algorithm works is

described. The upper part of Figure 21.4 shows a $(2, 3)$ directed de Bruijn graph denoting a graph with a maximum out-degree of 2, a diameter of 3 and order 8. Each node is represented by k-length (three in this example) strings. Every character of the string can take d different values (two in this example). In the general case each node is represented by a string such as $u_1u_2...u_k$. The connections between the nodes follow a simple left shift operation from node $u_1(u_2...u_k)$ to node $(u_2...u_k)u_x$, where u_x can take one of the possible values of the characters $(0, d-1)$. For example, we can move from node (010) to either node (100) or (101).

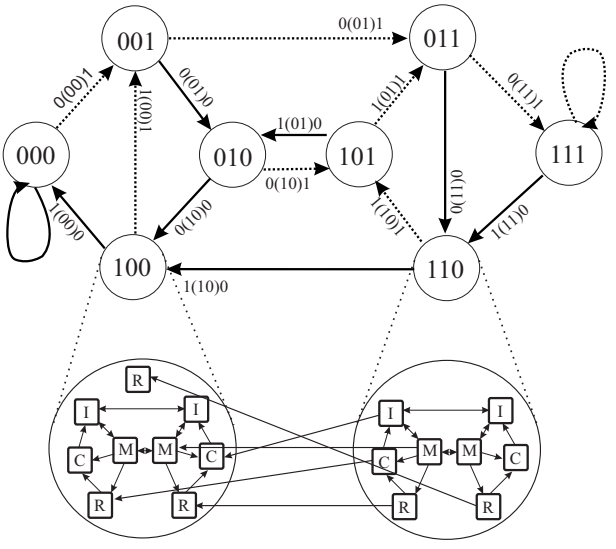

Fig. 21.4: Omicron overlay network

The most attractive feature of de Bruijn digraphs is the constant degree requirement for every node. However, this is also their "Achilles' heel" since robustness is difficult to achieve. As an alternative approach Omicron suggests the construction of clusters of peers and where application of a de Bruijn topology for the inter-cluster communication is proposed. This way the nodes (clusters of peers) of the digraph will be much more stable than single peers.

A dual identification scheme has been introduced for Omicron with a number of advantages. Clusters are assigned a Globally Unique IDentifier (GUID) that is used to route requests over the network. Advertised items are assigned a GUID and are located at the clusters whose GUID matches best. Moreover, peers are assigned their own GUID to trace their actions in the system.

Inter-cluster routing of messages is based on shift operations performed on the cluster GUID to select the neighbor cluster whose GUID matches best to the requested key. The operation is repeated until the final destination is reached.

Going a step further, a role-based scheme is introduced to deal with the heterogeneity of the peer capabilities and user behavior. This scheme fits the contribution of each node to its resource capabilities and aims at the maximization of the cluster efficiency by providing appropriate incentives to peers to take a certain role. The identified roles are based on the core overlay operations:

- **Overlay maintenance.** *Maintainers* perform the most demanding operation since peers are required to maintain complete routing tables, indexing information and cluster organization.
- **Indexing.** *Indexers* are required to handle the indexing responsibilities of the whole cluster in a balanced and co-operative way. Information redundancy is an additional requirement in order to shield the system against the single peer (mis-)behavior. Indexers can provide the final reply to queries when they reach the destination cluster.
- **Routing.** Routing is the most simple operation where *Routers* should forward messages to the neighbor clusters that are closer to the final destination. Combined with the low requirements raised by the de Bruijn digraphs it is a role suitable for any peer, even those that have very low capabilities and an unstable behavior. They participate only in the inter-cluster message forwarding service but they do not have the required information to provide the final reply as Indexers have.
- **Caching.** Although caching is not a basic operation (it is rather considered as advanced operation) it is included in the basic scheme because of the low requirements it poses and the fact that it closes the design gap between the Routers and the other more demanding roles. *Cachers* are expected to perform caching of the indexing information for the most popular items in order to reduce the effort of the Indexers. Studies (i.e. [379]) indicate that there is a Zipf-law distribution of the queries in popular content-related Peer-to-Peer systems.

An illustration of Omicron and a typical intra-cluster structure is pictured in the lower part of Figure 21.4.

It is clear that different requirements are related with each role (apparently, this is the design goal). Peers are "promoted" to adopt roles with higher requirements as they prove their stability and they fulfill the physical capability needs. Incentive mechanisms are present to motivate the promotion procedure.

Finally, an incrementally expandable algorithm has been designed to adapt the exponentially growing de Bruijn graphs to the incrementally expandable Peer-to-Peer systems.

21.4 Hybrid Routing

In this section, we describe hybrid Peer-to-Peer systems that focus on improving the performance of the routing mechanism. The constructed overlay network might also be hybrid, though in general, an additional mechanism such as caching might be required to enable the hybrid routing algorithm.

21.4.1 OceanStore

OceanStore [368] is a Peer-to-Peer storage system built on top of Tapestry [642] to take advantage of its scalable characteristics. However, OceanStore, employs an additional probabilistic mechanism based on attenuated Bloom Filters, resulting to a hybrid solution to improve even further Tapestry's routing performance.

The Bloom Filters algorithm was initially proposed by Bloom in the early '70s [77] to help word processors perform capitalization and/or hyphenation on a document. Bloom filters exploit efficiently the usually present non-uniform distribution of requests, where a small set of items is requested much more often than the rest of the stored items. In general, Bloom filters are capable of answering questions of the type: "Is this item member of that group"? The algorithm uses hash functions, though it requires less space and is faster than a conventional one to one hash-based mapping algorithm. However, it allows errors to happen. While negative replies to the aforementioned question are always correct (the mechanism is capable of replying correctly that an item does not belong to a group), it might provide false positive replies (an item that does not exist in a group might be falsely reported as a member). The probability of false positive replies can be configured with a number of parameters (e.g., increasing the space required for the data structure) and reduced to obey certain predefined bounds.

OceanStore uses attenuated Bloom Filters to provide a fast probabilistic search algorithm, where attenuated Bloom Filters are arrays of such filters. In the context of the OceanStore algorithm, the first Bloom filter (located at position '0') is a record of the objects contained locally on the current node. The ith Bloom filter is the union of all of the Bloom filters for all of the nodes a distance i through any path from the current node. An attenuated Bloom filter is stored for each directed edge in the network. A query is routed along the edge whose filter indicates the presence of the object at the smallest distance.

When the fast probabilistic algorithm fails to provide the requested results, OceanStore activates the Tapestry routing mechanism to forward the request to the final destination. As a result, OceanStore provides replies much faster for the very popular items than is using a Tapestry approach. However, the routing cost is increased for the cases where Bloom Filters provide false

replies. Moreover, the dissemination of Bloom Filters may consume considerable bandwidth.

21.4.2 Hybrid PIER

PIER [308] is a distributed query engine built on top of CAN. Similarly to the goal of OceanStore, in order to exploit the advantages of looking for popular items, a hybrid system has been proposed for PIER [394]. Hybrid PIER benefits both from DHTs and popularity-aware mechanisms, which are employed to get an improved overlay network. Hybrid PIER overlay is composed of two components, (i) an UltraPeer-based Gnutella network[5] and (ii), a structured CAN where UltraPeers may only participate. The hybrid search infrastructure utilizes selective publishing techniques that identify and publish only rare items into the DHT (a decision taken by the UltraPeers). The search algorithm uses flooding techniques for locating popular items, and structured (DHT) search techniques for locating rare items.

As long as the distribution of object replicas in the system follow a long tail distribution, such a hybrid system may perform better than a pure DHT alternative. However, the indexing and routing load is not evenly distributed.

21.5 Comparison with Non-hybrid Systems

In this section we compare hybrid solutions with non-hybrid in a general, abstract way, thus avoiding references to specific concrete systems in order to understand better the advantages and the shortcomings of following a hybrid design approach.

On the one hand, hybrid systems have increased complexity since they are combinations of more than one approaches and moreover, merged in a possibly constrained way that reveals the advantages of each sub-component. Hybrid systems naturally follow temporally the non-hybrid approaches that should be first well understood and both their benefits and drawbacks be identified.

On the other hand, hybrid systems may be better designed since their designers learn from the limitations and the mistakes of the pioneered pure approaches. Hybrid solutions show high adaptability to environmental conditions. Usually they are designed considering these conditions and they may reveal certain scenario-aware advantages or avoid related limitations. Performance may be greatly increased and new characteristics may be added to the constrained pure alternatives. Apparently, hybrid approaches may be the only viable way to address the large number of (usually) conflicting requirements raised in large-scale, dynamic and heterogeneous Peer-to-Peer systems.

[5] Based on Gnutella v0.6 protocol.

21.6 Summary and Conclusion

We have examined a large number of important hybrid Peer-to-Peer systems in order to reveal their advantages and the design purpose they serve. Though initially hybrid Peer-to-Peer systems were targeting in merging the Peer-to-Peer and Client/Server paradigms in different services they have been extended to explore a much wider range of combinations. Table 21.1 provides a summary of the main objectives and the key mechanisms to achieve them for each described hybrid systems.

In summary, with respect to the non-functional requirements, all the described systems address the scalability issue with a structured component. JXTA, Brocade and Omicron make heavy use of peers with special characteristics to deal with the heterogeneity of the peers. However, it is only Omicron that addresses heterogeneity in a balanced way where each peer contributes to the common overlay operations. Hybrid PIER and OceanStore take advantage of the popularity distributions for documents to increase their performance and reduce the network overhead. Brocade and Omicron deal in a certain degree with the creation of vicinity-aware overlay construction. Fault-tolerance is mainly achieved through redundancy mechanisms or the assignment of certain responsibilities to stable peers (or combination of both). SHARK and JXTA address the requirement for range queries support.

Hybrid Peer-to-Peer systems are interesting alternatives to pure system designs since they may overcome the limitations of the original approaches. Reality has shown that hybrid systems are usually the ones that are widely deployed and extensively used.

Hybrid Peer-to-Peer systems	Main objectives	Key mechanisms
Omicron	Efficient and stable large scale, heterogeneous, dynamic Peer-to-Peer systems	Clustering, Dynamic role assignment
SHARK	Scalable range queries	Hybrid structural Overlay
JXTA	Low cost overlay management	Role separation
Brocade	Efficient mapping of overlay to underlay network	Location-aware sub-network
OceanStore	Low cost search for popular queries	Bloom Filters based caching
Hybrid PIER	Low cost search for popular items	Hybrid structural overlay

Table 21.1: Objectives of Hybrid Peer-to-Peer systems

What is missing is a more systematic approach that may identify the requirements of the targeted systems, select and appropriately combine the identified components to meet these requirements. There are three main design dimensions to explore in the design of hybrid overlay networks, however, researchers may investigate further options defined in the context of the problem to be solved.

Further Reading

For the interested reader, a number of further citations are provided here to trigger further research areas. A hybrid topology inspired by Peer-to-Peer overlays and applied in mobile ad hoc networks can be found in [364]. A two-tier hierarchical Chord is explored in [238]. Measurement efforts on the KaZaA system can be found in [384]. A hybrid topology that extends Chord to increase the degree of user anonymity can be found in [562]. An early comparison of some pioneering hybrid approaches such as Napster and Pointera is provided in [634]. A hybrid protocol named Borg [641] aims in scalable Application-level Multicast. A generic mechanism for the construction and maintenance of superpeer-based overlay networks is proposed in [428]. A Peer-to-Peer overlay network simulator has been implemented especially to augment the evaluation of a wide range of hybrid designing methods [152]. A hybrid approach on deploying hybrid Content Delivery Networks (CDNs) based on an ad hoc Peer-to-Peer overlay and a centralized infrastructure is described in [633].