Published on The O'Reilly Network (http://www.oreillynet.com/)
http://www.oreillynet.com/pub/a/p2p/2002/01/08/p2p_topologies_pt2.html
See this if you're having trouble printing code examples

Distributed Systems Topologies: Part 2

by Nelson Minar
01/08/2002

In the first part of this two-part series, I presented a 10,000-foot view of how a framework for comparing distributed systems, based on system topology, is developed. In this second part, I introduce seven criteria for evaluating a system design and discuss their relative merits. Systems with hybrid toplogies often seem to demonstrate the advantages of the various constituent designs that comprise their makeup.

Evaluating System Topologies

In the first part of this series I described distributed systems in terms of their core topologies: centralized, decentralized, rings, hierarchies, and hybrids. Now I take advantage of this description by using it to evaluate system designs.

In this second part, I describe seven characteristics of distributed systems that are commonly used when talking about system design and then analyze each characteristic for each of the topologies. As with the topology descriptions, the same caution about the high-level nature of this analysis applies to this article. The observations made here are generalizations and may not apply to any specific system. The intent is to develop a broad framework for considering system design that can then be applied to specific domains.

Seven Evaluation Properties

For this article, I boil down all possible ways to evaluate distributed systems into seven properties. While not exhaustive, this set is chosen because these properties are often used when talking about the advantages or disadvantages of decentralized systems. The resulting framework is a useful shorthand for thinking about system design.

Manageability

How hard is it to keep the system working? Complex systems require management: updating, repairing, and logging.

Information coherence

How authoritative is information in the system? If a bit of data is found in the system, is that data correct? Non-repudiation, auditability, and consistency are particular aspects of information coherence.

Extensibility

How easy is it to grow the system, to add new resources to it? The Web is the ultimate extensible system; anyone can create a new Web server or Web page and immediately have that contribution be part of the Web.

Fault Tolerance

How well can the system handle failures? Fault tolerance is a necessity in large distributed systems.

Security

How hard is it to subvert the system? Security covers a variety of topics, such as preventing people from taking over the system, injecting bad information, or using the system for a purpose other than which the owners intend.

Resistance to lawsuits and politics

How hard is it for an authority to shut down the system? The designers of Gnutella or Freenet consider their resistance to lawsuits to be one of their best features. Other parties consider this property to be a danger.

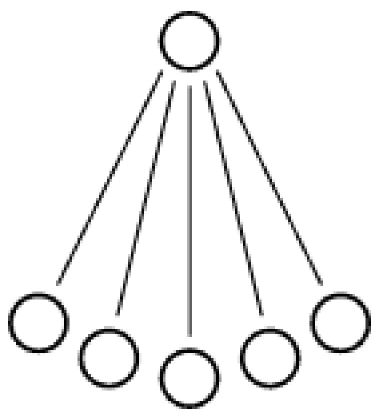
Scalability

How large can the system grow? Scalability is often promoted as a key advantage of decentralized systems over centralized, although the reality is more complex.

Evaluating Simple Topologies

With these seven concepts in mind, we can look at each of the basic system topologies and evaluate their effectiveness.

Centralized

Manageable	Yes	
Coherent	Yes	
Extensible	No	
Fault-Tolerant	No	
Secure	Yes	
Lawsuit-Proof	No	
Scalable	?	

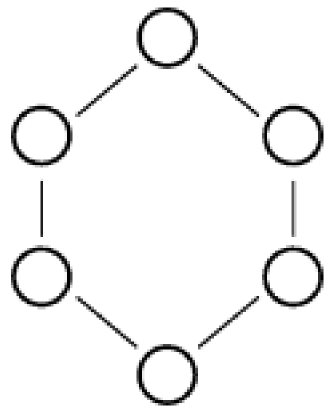
The primary advantage of centralized systems is their simplicity. Because all data is concentrated in one place, centralized systems are easily managed and have no questions of data consistency or coherence.

Centralized systems are also relatively easy to secure: there is only one host that needs to be protected. The drawback of centralization is that everything *is in only one place*. If the central server goes down, everything does. There is no fault tolerance, and the system is easy to shut down with a lawsuit. Centralized systems are also often hard to extend -- resources can only be added to the central system.

The scalability of centralized systems is subtle. Scale is clearly limited by the capacity of the server, and so centralized systems are often thought of as unscalable. But computers are very fast and a single computer can often support all the demands of its users.

For example, a modest computer running a Web server can easily handle hundreds of thousands of visitors a day. And unlike more complex topologies, the scalability of a centralized system is very easy to measure. So while, theoretically, centralized systems are not scalable, in practice they often suffice.

Ring

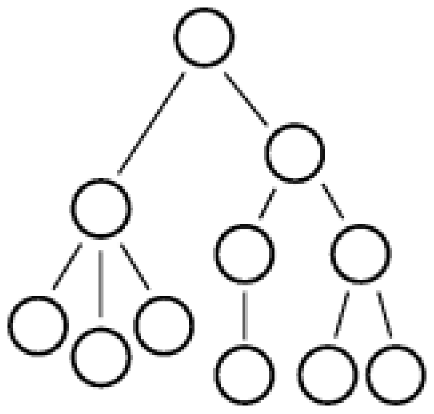
Manageable	Yes	
Coherent	Yes	
Extensible	No	
Fault-Tolerant	Yes	
Secure	Yes	
Lawsuit-Proof	No	
Scalable	Yes	

Ring systems typically have a single owner. This concentration gives them many of the same advantages of centralized systems: they are manageable, coherent, and relatively secure from tampering.

The added complexity of the ring is mitigated by fairly simple rules for propagating state between the nodes in a ring. But the single-owner restriction means rings are also not extensible: a user still needs the owner's permission to add a resource like a music file or a Web page into the ring. Similarly, a lawsuit only needs to shut down the owner to shut down the whole ring.

The advantages of rings over centralized systems are fault tolerance and simple scalability. If a host goes down in a ring, failover logic makes it a simple matter to have another host cover the problem. And well-designed rings are scalable -- one can simply add more hosts to the ring and expand the capacity nearly linearly.

Hierarchical

Manageable	Partially	
Coherent	Partially	
Extensible	Partially	
Fault-Tolerant	Partially	
Secure	No	
Lawsuit-Proof	No	
Scalable	Yes	

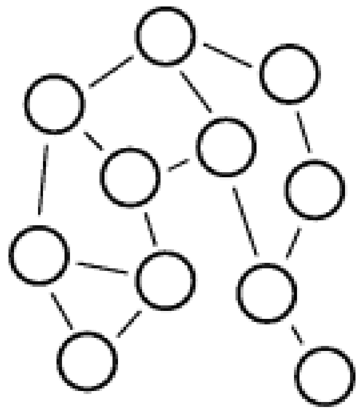
Hierarchical systems have a completely different set of advantages from that of rings. Hierarchical systems are somewhat manageable in that they have a clear chain of action. But because these systems have such a broad scope, it can be hard to correct a host with a problem. Coherence is usually achieved with a cache consistency type of strategy; effective, but not complete.

Hierarchical systems are extensible in that any host in the system can add data, but the rules of data management may limit what information can be added. (For example, the oreilly.com DNS server can add hosts for oreilly.com, but for no one else.)

Hierarchical systems are more fault-tolerant and lawsuit-proof than centralized systems, but the root is still a single point of failure. They tend to be harder to secure than centralized systems. If a node high in the hierarchy is subverted or spoofed, the whole system suffers. And it's not just the root that is a risk: if data travels up the branches to the root, then leaf nodes may be able to inject bad information to the system.

The primary advantage of hierarchical systems is their incredible scalability -- new nodes can be added at any level to cover for too much load. This scalability is best demonstrated in DNS, which has scaled over the last 15 years from a few thousand hosts to hundreds of millions. The relative simplicity and openness of hierarchical systems, in addition to their scalability, make them a desirable option for large Internet systems.

Decentralized

Manageable	No	
Coherent	No	
Extensible	Yes	
Fault-Tolerant	Yes	
Secure	No	
Lawsuit-Proof	Yes	
Scalable	Maybe	

Decentralized systems such as Gnutella have almost the exact opposite characteristics as centralized systems. The far-flung nature of these networks means the systems tend to be difficult to manage and that data in the system is never fully authoritative. They also tend to be insecure, in the sense that it is easy for a node to join the network and start putting bad data into the system.

A primary virtue of decentralized systems is their extensibility. For example, in Gnutella any node can join the network and instantly make new files available to the whole network. Decentralized systems also tend to be fault-tolerant and harder to sue. The failure or shutdown of any particular node does not impact the rest of the system.

The scalability of decentralized systems is hard to evaluate. In theory, the more hosts you add, the more capable a decentralized network becomes. In practice, the algorithms required to keep a decentralized system coherent often carry a lot of overhead. If that overhead grows with the size of the system, then the system may not scale well. The Gnutella network suffered this problem in the early stages, and it remains to be seen if Gnutella can ever scale to the millions of active users that more centralized architectures enjoy. Scalability of decentralized systems remains an active research topic.

Evaluating Hybrid Topologies

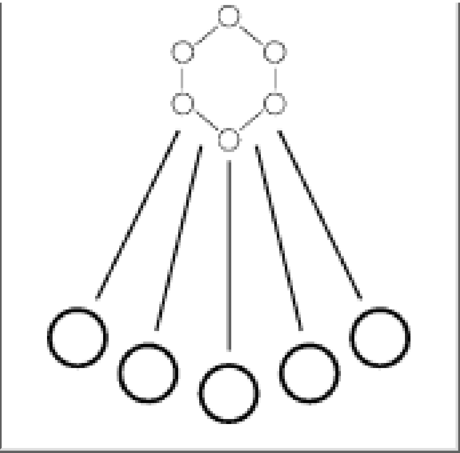
System topologies become even more interesting when you combine them into hybrid architectures. Often, different topologies are chosen for different parts of a system to get the best of the strengths without the weaknesses.

Centralized + Ring

Manageable	Yes	

Systems that have a ring as their central server often enjoy the best of the simplicity of centralization with the redundancy of a ring. The hybrid system is still easily managed, coherent, and secure; the ring does not add much complexity over a purely centralized system.

Coherent	Yes
Extensible	No
Fault-Tolerant	Yes
Secure	Yes
Lawsuit-Proof	No
Scalable	Yes

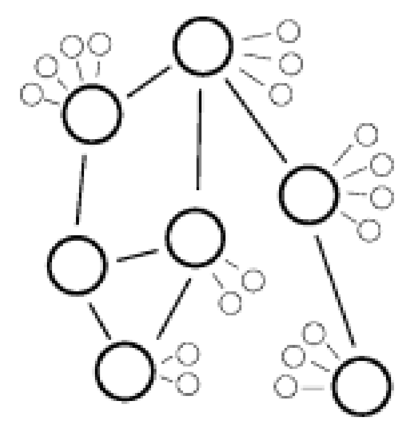


This combination still has a single owner and therefore is not particularly extensible or lawsuit-proof. The key advantage is that using a ring as the server adds fault-tolerance and scalability. The power and simplicity of the combination of rings and centralized systems explains why this architecture is so popular with serious server-based applications such as Web commerce and high-availability databases.

Centralized + Decentralized

A system combining centralized and decentralized systems enjoys some of the advantages of both. Decentralization contributes to the extensibility, fault-tolerance, and lawsuit-proofing of the system. The partial centralization makes the system more coherent than a purely decentralized system, as there are relatively fewer hosts that are holding authoritative data. Manageability is about as difficult as a decentralized system, and the system is no more secure than any other decentralized system.

Manageable	No
Coherent	Partially
Extensible	Yes
Fault-Tolerant	Yes
Secure	No
Lawsuit-Proof	Yes
Scalable	Apparently



The amazing story is the scalability of this hybrid. Internet email runs very well for hundreds of millions of users and has grown enormously since its initial design. FastTrack-based systems have grown very quickly with none of the slowdowns that plagued Napster or Gnutella in their growth. There is growing interest in this kind of hybrid topology as an excellent architecture for peer-to-peer systems.

Conclusions

A decentralized system is not always better or worse than a centralized system. The choice depends entirely on the needs of the application. The simplicity of centralized systems makes them easier to manage and control, while decentralized systems grow better and are more resistant to failures or shutdowns.

Have you seen these basic systems combined in any particularly exciting ways not mentioned in this article?

Post your comments

As for scalability, the story is not clear. Centralized systems have limited scale, but that limit is easy to understand. In contrast, decentralized systems offer the possibility of massive scalability, but in practice that can be very hard to achieve.

The second conclusion is the power of creating hybrid topologies. In centralized+ring systems, the ring covers many of the drawbacks of a purely centralized approach, providing easy scalability and fault tolerance. And centralized+decentralized systems are showing powerful scalability and extensibility while retaining some of the coherence of centralized systems.

System designers have to evaluate the requirements for their particular area and pick a topology that matches their needs. We are not limited to a few simple topologies; topologies can be combined to make hybrids. And while centralized systems are doing a lot of the work on the Internet, there is a lot of exciting potential in decentralized systems. In particular, combining decentralized topologies with other simpler topologies is a powerful approach.

Nelson Minar was co-founder of Popular Power.

Return to the OpenP2P.com.

Published on **OpenP2P.com** (http://www.openp2p.com/)
http://www.openp2p.com/pub/a/p2p/2001/12/14/topologies_one.html
See this if you're having trouble printing code examples

Distributed Systems Topologies: Part 1

by Nelson Minar
12/14/2001

The peer-to-peer explosion has reminded people of the power of decentralized systems. The promise of robustness, open-endedness, and infinite scalability have made many people excited about decentralization. But in reality, most systems we build on the Internet are largely centralized.

This two-part article develops a framework for comparing distributed system designs. In this first part, I boil down the design of many systems to their essential topologies and describe how hybrid topologies can be made by combining parts. In the second part, I will introduce seven criteria for evaluating a system design and discuss the relative merits of distributed system designs.

Looking at topology

The peer-to-peer trend has renewed interest in decentralized systems. The Internet itself is the largest decentralized computer system in the world. But ironically in the 1990s many systems built on the Internet were completely centralized. The growth of the Web meant most systems were single web servers running in fabulously expensive collocation facilities. Now with peer-to-peer, the pendulum has swung the other way to radically decentralized architectures such as Gnutella. In practice, extreme architectural choices in either direction are seldom the way to build a usable system.

With the Internet, we have 30 years of experience with distributed systems architectures. With all this experience, a high-level framework for understanding distributed systems is helpful for organizing what we have learned.

In this article, I focus on the *topology* of the distributed systems -- how the different computers in the system fit together. Four basic topologies are in use on the Internet: centralized and decentralized, but also hierarchical and ring systems. These topologies can be used by themselves, or combined into one system creating hybrid systems.

Reader beware: This article takes a very high-level view, trying to encompass 30 years of Internet development into a short article. By necessity, the following analysis is in terms of generalities and may not be accurate for any specific case. My hope is that this approach will help the reader look at system design from the top down, and thereby better evaluate choices, such as whether to use a centralized or decentralized approach for a specific application.

Basic distributed systems topologies

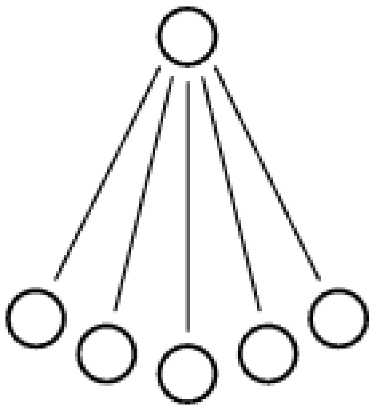
The debate between centralized and decentralized systems is fundamentally about topology -- in other words, how the nodes in the system are connected. Topology can be considered at many different levels: physical, logical, connection, or organizational.

For this analysis, topology is considered in terms of the information flow. Nodes in the graph are individual computers or programs, links between nodes indicate that those nodes are sharing information regularly in the system. Typically, an edge implies that the two nodes are directly sharing bits across a network link. For simplicity, I do not consider the direction of information flow; edges are considered undirected.

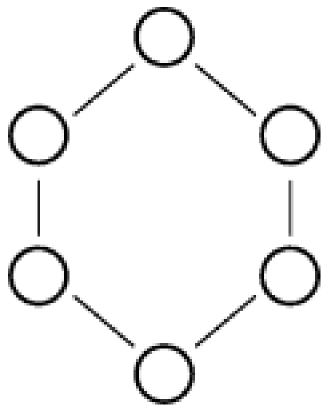
Four common topologies will be explained here: centralized, ring, hierarchical, and decentralized. (A fifth distributed system pattern, group communication, is not considered in this article.)

Centralized

Centralized systems are the most familiar form of topology, typically seen as the client/server pattern used by databases, web servers, and other simple distributed systems. All function and information is centralized into one server with many clients connecting directly to the server to send and receive information. Many applications called "peer-to-peer" also have a centralized component. SETI@Home is a fully centralized architecture with the job dispatcher as the server. And the original Napster's search architecture was centralized, although the file sharing was not.



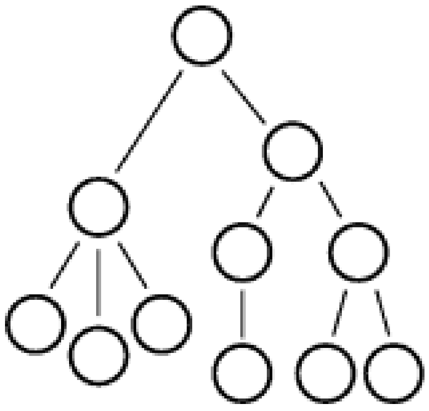
Ring



A single centralized server cannot handle high client load, so a common solution is to use a cluster of machines arranged in a ring to act as a distributed server. Communication between the nodes coordinates state-sharing, producing a group of nodes that provide identical function but have failover and load-balancing capabilities. Unlike the other topologies here, ring systems are generally built assuming the machines are all nearby on the network and owned by a single organization.

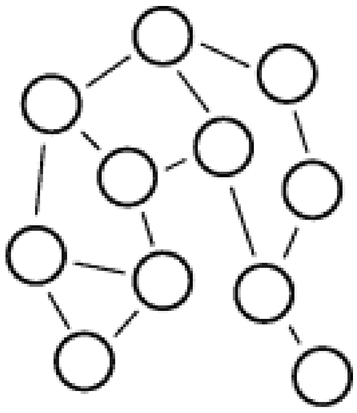
Hierarchical

Hierarchical systems have a long history on the Internet, but in practice are often overlooked as a distinct distributed systems topology. The best-known hierarchical system on the Internet is the Domain Name Service, where authority flows from the root name-servers to the server for the registered name and often down to third-level servers. NTP, the Network Time Protocol, creates another hierarchical system.



In NTP, there are root time servers that have authoritative clocks; other computers synchronize to root time servers in a self-organizing tree. NTP has over 175,000 hosts with most hosts being two or three links away from a root time source. Usenet is another large hierarchical system, using a tree-like structure to copy articles between servers. It is particularly interesting in that the underlying protocols are symmetric but in practice, articles propagate along tree-like paths with a relatively small set of hosts acting as the backbone.

Decentralized

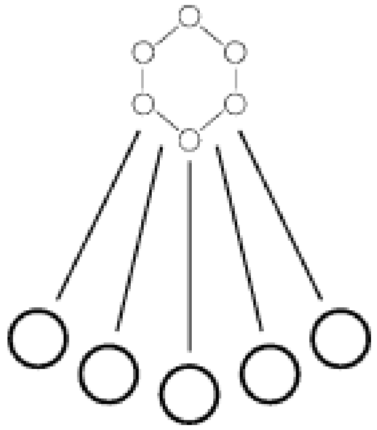


The final topology we consider here is decentralized systems, where all peers communicate symmetrically and have equal roles. Gnutella is probably the most "pure" decentralized system used in practice today, with only a small centralized function to bootstrap a new host. Many other file-sharing systems are also designed to be decentralized, such as Freenet or OceanStore. Decentralized systems are not new; the Internet routing architecture itself is largely decentralized, with the Border Gateway Protocol used to coordinate the peering links between various autonomous systems.

Hybrid topologies

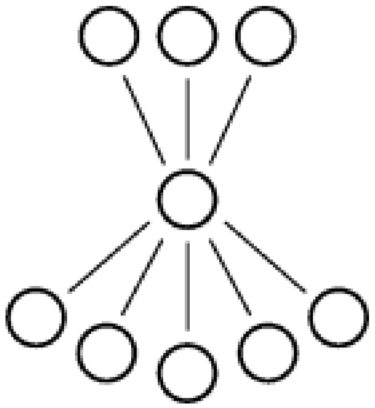
Distributed systems often have a more complex organization than any one simple topology. Real-world systems often combine several topologies into one system, making a *hybrid topology*. Nodes typically play multiple roles in such a system. For example, a node might have a centralized interaction with one part of the system, while being part of a hierarchy in another part.

Centralized + Ring



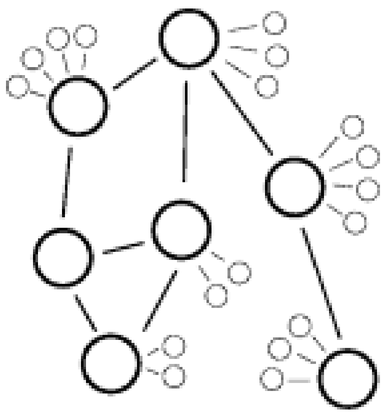
As mentioned above, serious web server applications often have a ring of servers for load balancing and failover. The server system itself is a ring, but the system as a whole (including the clients) is a hybrid: a centralized system where the server is itself a ring. The result is the simplicity of a centralized system (from the client's point of view) with the robustness of a ring.

Centralized + Centralized



The server in a centralized system is itself often a client of one or more other servers. Stacking multiple centralized systems is the core of *n*-tier application frameworks. For example, when a web browser contacts a server, the software on that server may just be formatting results into HTML for presentation and itself calling to servers hosting business logic or data. Web services intermediaries such as [Grand Central Networks] also create several layers of centralized system. Centralized systems are often stacked as a way to compose function.

Centralized + Decentralized



A new wave of peer-to-peer systems is advancing an architecture of centralized systems embedded in decentralized systems. This hybrid topology is realized with hundreds of thousands of peers in the [FastTrack] file-sharing system used in KaZaA and Morpheus. Most peers have a centralized relationship to a "supernode," forwarding all file queries to this server (much like a Napster client sends queries to the Napster server). But instead of supernodes being standalone servers, they band themselves together in a Gnutella-like decentralized network, propagating queries. Internet email also shows this kind of hybrid topology. Mail clients have a centralized relationship with a specific mail server, but mail servers themselves share email in a decentralized fashion.

Other topologies

There are limitless possibilities in combining various kinds of architectures. A centralized system could have a hierarchy of machines in the role of server. Decentralized systems could be built that span different rings or hierarchies. Systems could conceivably be built with three or more topologies combined, although the resulting complexity may be too difficult to manage. Topology is a useful simplifying tool in understanding the architecture of distributed systems.

Conclusion, and on to evaluation

In this article, I have introduced a way to think of distributed system design. By looking at systems in terms of their topology, it is possible to examine a wide spectrum of systems we have on the Internet today, from centralized to decentralized and with architectures in between. In the second part of this article, I will develop a framework for analyzing the strengths and weaknesses of distributed systems and apply it to the different topologies presented here.

Note: this article is based on a talk given by Nelson Minar at the O'Reilly Peer-to-Peer and Web Services Conference in November, 2001. The slides from that talk are also online.

Nelson Minar was co-founder of Popular Power.

Return to OpenP2P.com.