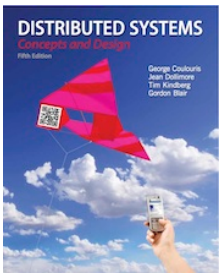
Slides for Chapter 10:
Peer-to-Peer Systems



From **Coulouris, Dollimore, Kindberg and Blair**

Distributed Systems:
Concepts and Design

Edition 5, © Addison-Wesley 2012

Edited and supplemented by Jonne Itkonen,
for TIE427 (2011)

Intro

- **The goal of peer-to-peer systems is to enable the sharing of data and recources on a very large scale by eliminating any requirements for separately managed servers and their associated infrastructure.**
- **Peer-to-peer systems aim to support useful distributed services and applications using data and computing resources available in the personal computers and workstations that are present in the network.**

Characteristics

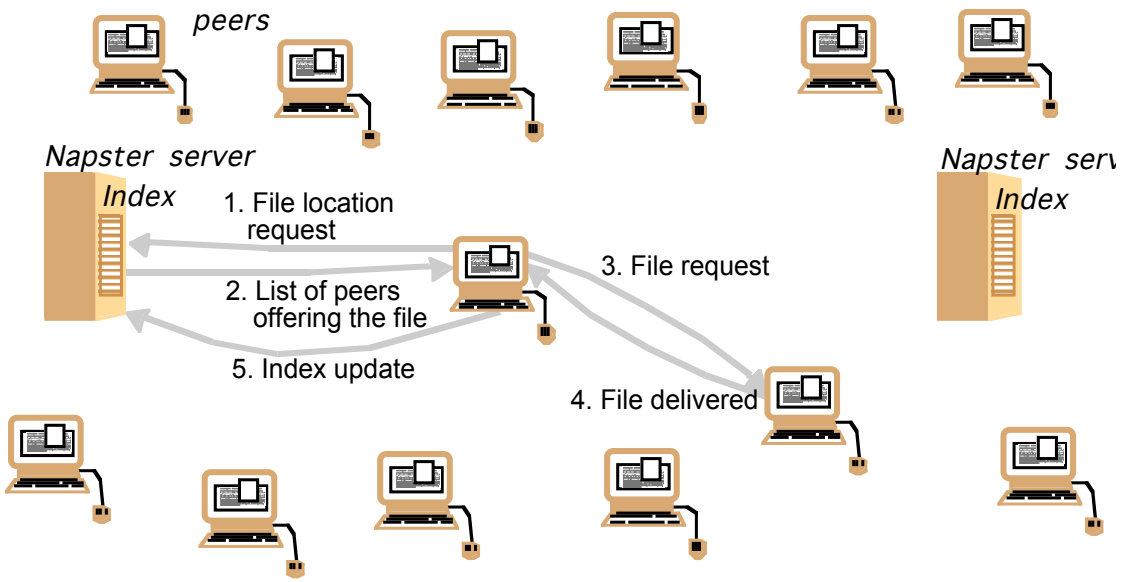
- **Their design ensures that each user contributes resources to the system.**
- **Although they may differ in the resources that they contribute, all the nodes in a peer-to-peer system have the same functional capabilities and responsibilities.**
- **Their correct operation does not depend on the existence of any centrally administrated systems.**
- **They can be designed to offer a limited degree of anonymity to the providers and users of resources.**
- **A key issue for their efficient operation is the choice of an algorithm for the placement of data across many hosts and subsequent access to it in a manner that balances the workload and ensures availability without adding undue overheads.**
- **Availability of the processes and computers participating in peer-to-peer systems is unpredictable.**

Figure 10.1: Distinctions between IP and overlay routing for peer-to-peer applications

	IP	*Application-level routing overlay*
Scale	IPv4 is limited to 232 addressable nodes. The IPv6 name space is much more generous (2128), but addresses in both versions are hierarchically structured and much of the space is pre-allocated according to administrative requirements.	Peer-to-peer systems can address more objects. The GUID name space is very large and flat (>2128), allowing it to be much more fully occupied.
Load balancing	Loads on routers are determined by network topology and associated traffic patterns.	Object locations can be randomized and hence traffic patterns are divorced from the network topology.
Network dynamics (addition/deletion of objects/nodes)	IP routing tables are updated asynchronously on a best-efforts basis with time constants on the order of 1 hour.	Routing tables can be updated synchronously or asynchronously with fractions of a second delays.
Fault tolerance	Redundancy is designed into the IP network by its managers, ensuring tolerance of a single router or network connectivity failure. *n*-fold replication is costly.	Routes and object references can be replicated *n*-fold, ensuring tolerance of *n* failures of nodes or connections.
Target identification	Each IP address maps to exactly one target node.	Messages can be routed to the nearest replica of a target object.
Security and anonymity	Addressing is only secure when all nodes are trusted. Anonymity for the owners of addresses is not achievable.	Security can be achieved even in environments with limited trust. A limited degree of anonymity can be provided.

© Pearson Education 2012

Figure 10.2: Napster: peer-to-peer file sharing with a centralized, replicated index

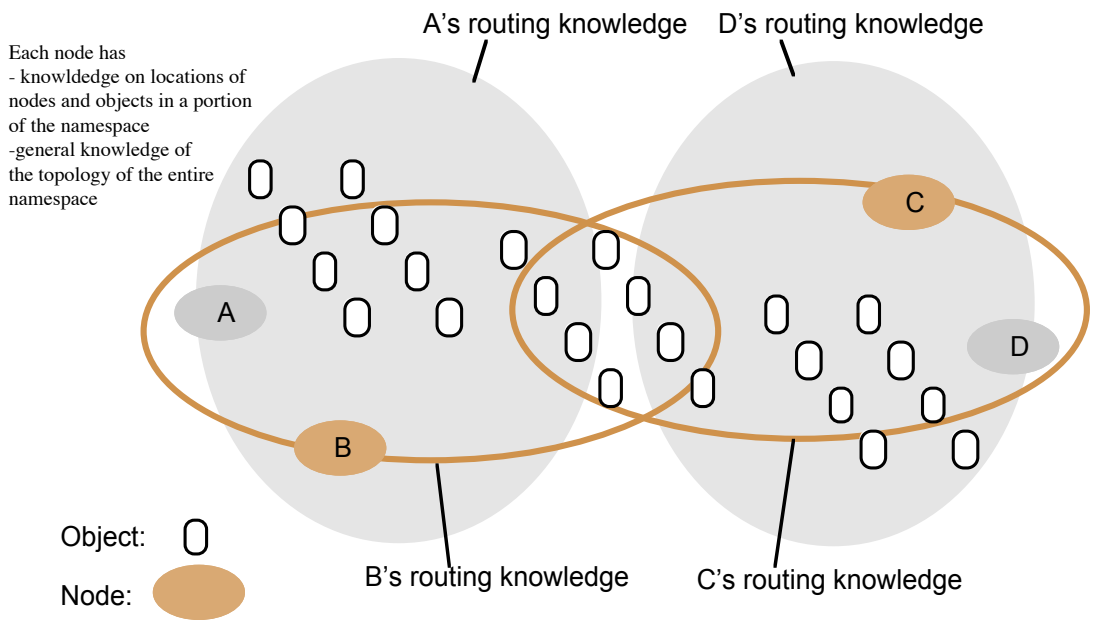


Instructor's Guide for Coulouris, Dollimore, Kindberg and Blair, Distributed Systems: Concepts and Design Edn. 5
© Pearson Education 2012

Non-functional requirements

- **Global scalability**
- **Load balancing**
- **Optimization for local interactions between neighbouring peers**
- **Accomodating to highly dymamic host availability**
- **Security of data in an environment with heterogenous trust**
- **Anonymity, deniability and resistance to censorship**

Figure 10.3: Distribution of information in a routing overlay



© Pearson Education 2012

Figure 10.4: Basic programming interface for a distributed hash table (DHT) as implemented by the PAST API over Pastry

put(GUID, data)

The *data* is stored in replicas at all nodes responsible for the object identified by *GUID*.

remove(GUID)

Deletes all references to *GUID* and the associated data.

value = get(GUID)

The data associated with *GUID* is retrieved from one of the nodes responsible it.

© Pearson Education 2012

Figure 10.5: Basic programming interface for distributed object location and routing (DOLR) as implemented by Tapestry

publish(GUID)

GUID can be computed from the object (or some part of it, e.g. its name). This function makes the node performing a *publish* operation the host for the object corresponding to *GUID*.

unpublish(GUID)

Makes the object corresponding to *GUID* inaccessible.

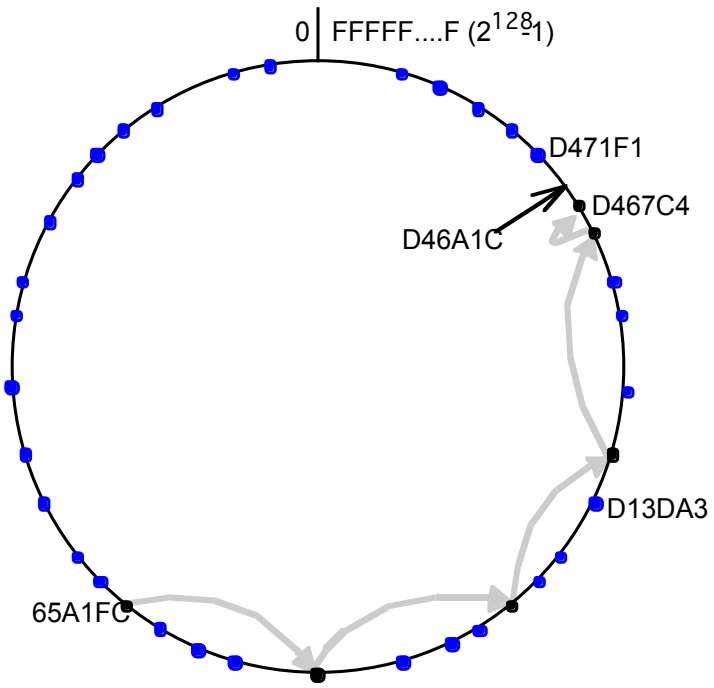
sendToObj(msg, GUID, [n])

Following the object-oriented paradigm, an invocation message is sent to an object in order to access it. This might be a request to open a TCP connection for data transfer or to return a message containing all or part of the object's state. The final optional parameter *[n]*, if present, requests the delivery of the same message to n replicas of the object.

© Pearson Education 2012

Figure 10.6: Circular routing alone is correct but inefficient

Based on Rowstron and Druschel [2001]



The dots depict live nodes. The space is considered as circular: node 0 is adjacent to node (2128-1). The diagram illustrates the routing of a message from node 65A1FC to D46A1C using leaf set information alone, assuming leaf sets of size 8 (l = 4). This is a degenerate type of routing that would scale very poorly; it is not used in practice.

Leaf set – a vector L (of size 2*l*) containing the GUIDs and IP addresses of the nodes whose GUIDs are numerically closest on either side of its own (*l* above and *l* below).

© Pearson Education 2012

Figure 10.7: First four rows of a Pastry routing table

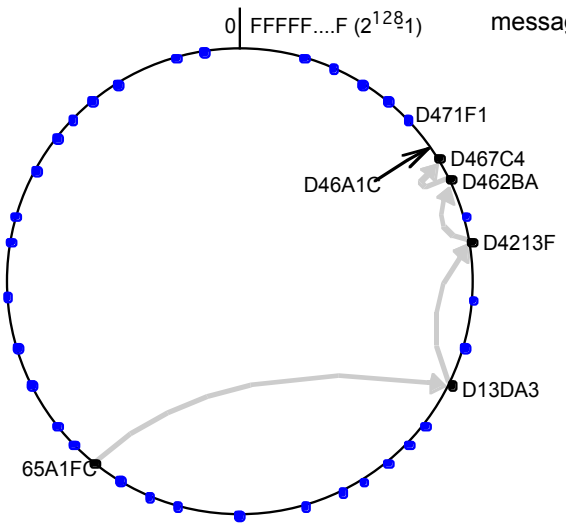
p =	*GUID prefixes and corresponding nodehandles n*															
0	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
	n	*n*	*n*	*n*	*n*	*n*		*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*
1	*60*	*61*	*62*	*63*	*64*	*65*	*66*	*67*	*68*	*69*	*6A*	*6B*	*6C*	*6F*	*6E*	*6F*
	n	*n*	*n*	*n*	*n*		*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*
2	*650*	*651*	*652*	*653*	*654*	*655*	*656*	*657*	*658*	*659*	*65A*	*65B*	*65C*	*65D*	*65E*	*65F*
	n	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*		*n*	*n*	*n*	*n*	*n*
3	*65A0*	*65A1*	*65A2*	*65A3*	*65A4*	*65A5*	*65A6*	*65A7*	*65A8*	*65A9*	*65AA*	*65AB*	*65AC*	*65AD*	*65AE*	*65AF*
	n		*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*	*n*

The routing table is located at a node whose GUID begins 65A1. Digits are in hexadecimal. The *n*'s represent [GUID, IP address] pairs specifying the next hop to be taken by messages addressed to GUIDs that match each given prefix. Grey- shaded entries indicate that the prefix matches the current GUID up to the given value of *p*: the next row down or the leaf set should be examined to find a route. Although there are a maximum of 128 rows in the table, only log16 *N* rows will be populated on average in a network with *N* active nodes.

© Pearson Education 2012

Figure 10.8: Pastry routing example Based on Rowstron and Druschel [2001]

Routing a message from node 65A1FC to D46A1C. With the aid of a well-populated routing table the message can be delivered in ~ $\log_{16}(N)$ hops.



© Pearson Education 2012

Figure 10.9: Pastry's routing algorithm

To handle a message M addressed to a node D (where $R[p,i]$ is the element at column i, row p of the routing table):

1. If ($L_{-l} < D < L_l$) { // the destination is within the leaf set or is the current node.
2. Forward M to the element L_i of the leaf set with GUID closest to D or the current node A.
3. } else { // use the routing table to despatch M to a node with a closer GUID
4. find p, the length of the longest common prefix of D and A. and i, the $(p+1)^{th}$ hexadecimal digit of D.
5. If ($R[p,i] \neq null$) forward M to $R[p,i]$ // route M to a node with a longer common prefix.
6. else { // there is no entry in the routing table
7. Forward M to any node in L or R with a common prefix of length i, but a GUID that is numerically closer.

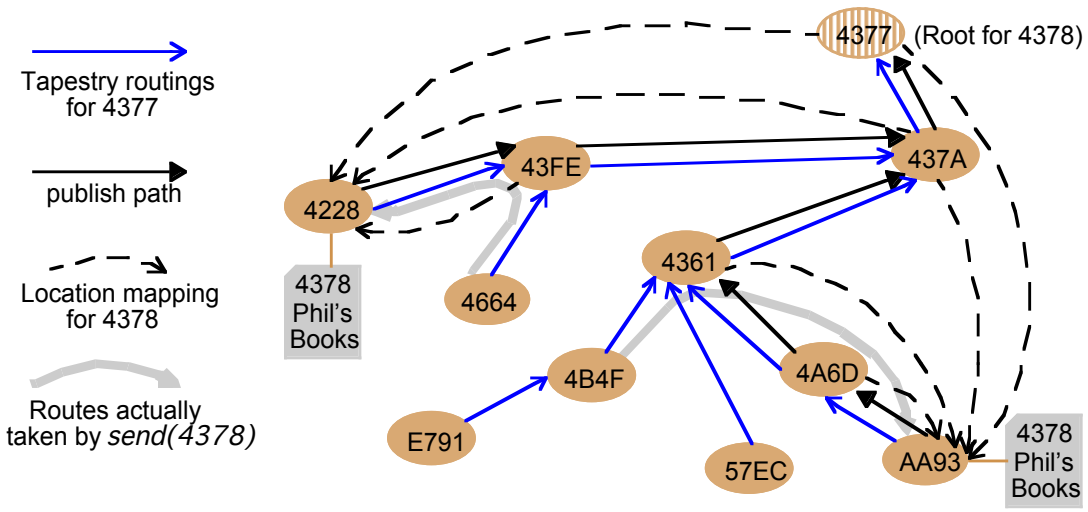
   }

}

© Pearson Education 2012

Figure 10.10: Tapestry routing

From [Zhao et al. 2004]



Replicas of the file *Phil's Books* (G=4378) are hosted at nodes 4228 and AA93. Node 4377 is the root node for object 4378. The Tapestry routings shown are some of the entries in routing tables. The publish paths show routes followed by the publish messages laying down cached location mappings for object 4378. The location mappings are subsequently used to route messages sent to 4378.

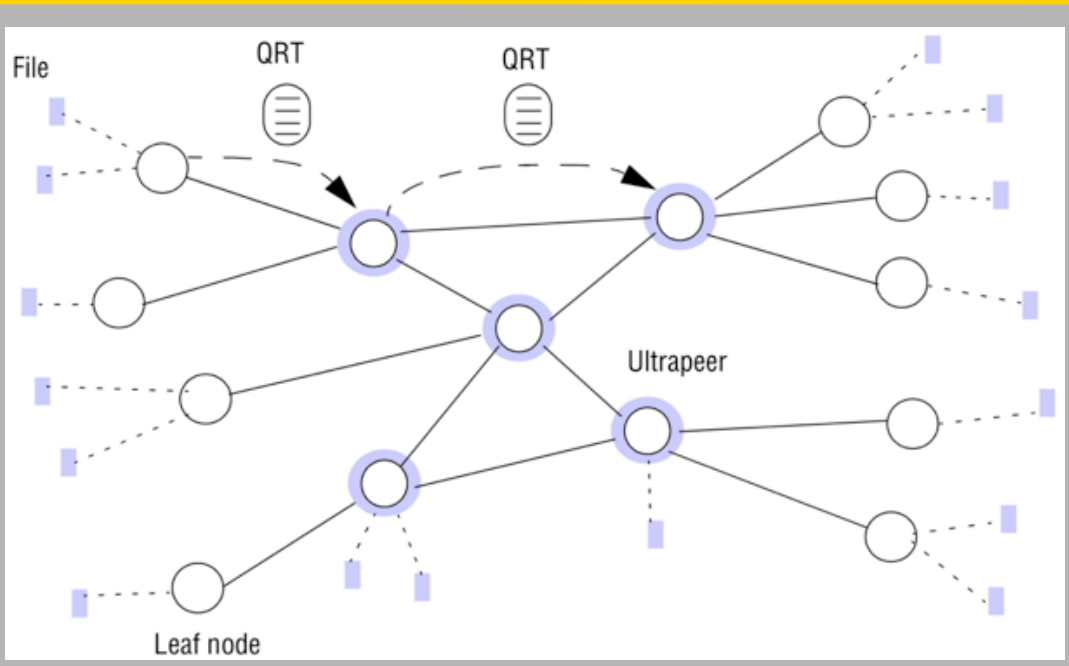
© Pearson Education 2012

Figure 10.11: Structured versus unstructured peer-to-peer systems

	Structured peer-to-peer	*Unstructured peer-to-peer*
Advantages	Guaranteed to locate objects (assuming they exist) and can offer time and complexity bounds on this operation; relatively low message overhead.	Self-organizing and naturally resilient to node failure.
Disadvantages	Need to maintain often complex overlay structures, which can be difficult and costly to achieve, especially in highly dynamic environments.	Probabilistic and hence cannot offer absolute guarantees on locating objects; prone to excessive messaging overhead which can affect scalability.

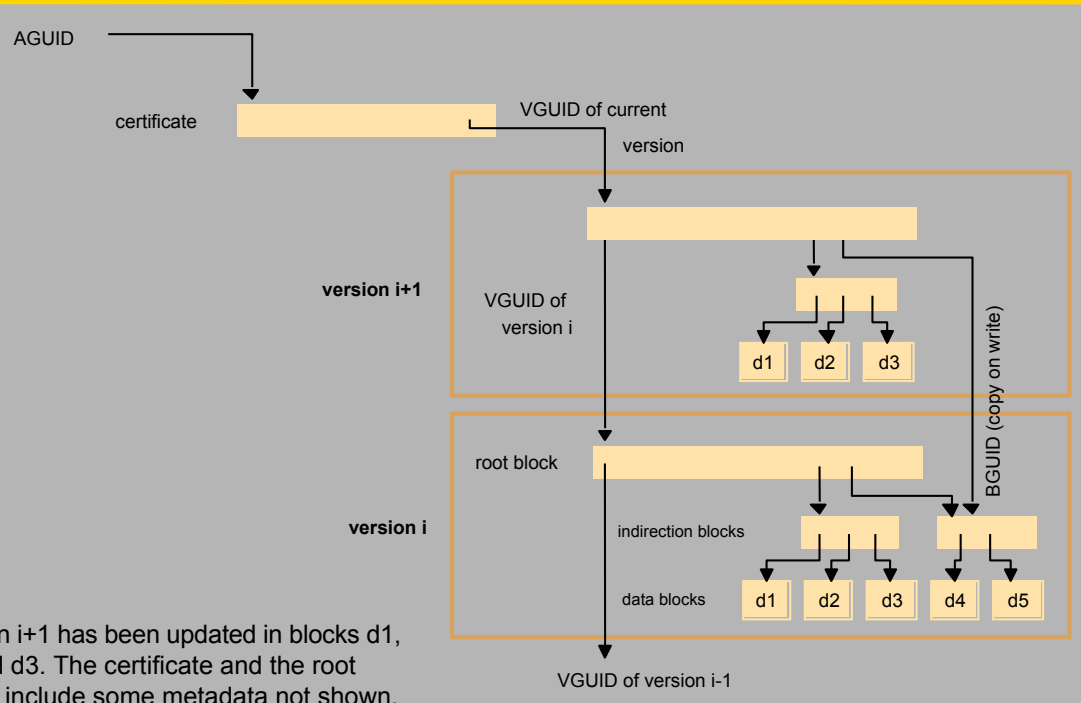
© Pearson Education 2012

Figure 10.12: Key elements in the Gnutella protocol



© Pearson Education 2012

Figure 10.13: Storage organization of OceanStore objects



Version i+1 has been updated in blocks d1, d2 and d3. The certificate and the root blocks include some metadata not shown. All unlabelled arrows are BGUIDs.

© Pearson Education 2012

Figure 10.14: Types of identifier used in OceanStore

Name	*Meaning*	*Description*
BGUID	block GUID	Secure hash of a data block
VGUID	version GUID	BGUID of the root block of a version
AGUID	active GUID	Uniquely identifies all the versions of an object

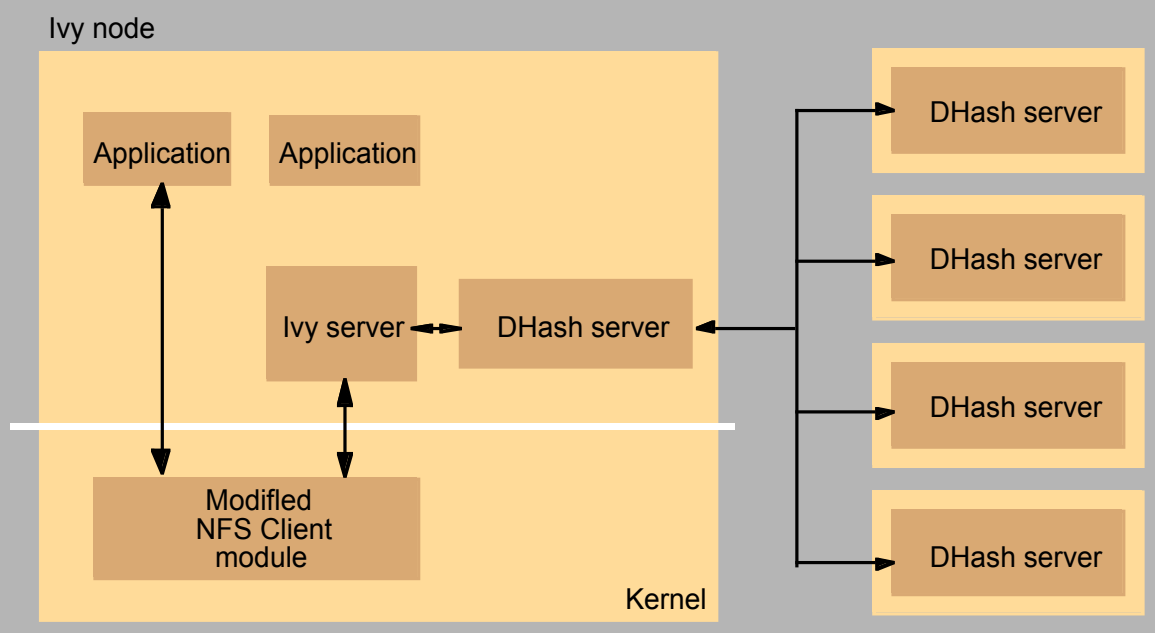
© Pearson Education 2012

Figure 10.15: Performance evaluation of the Pond prototype emulating NFS

	LAN		*WAN*		*Predominant operations in benchmark*
Phase	*Linux NFS*	*Pond*	*Linux NFS*	*Pond*	
1	0.0	1.9	0.9	2.8	Read and write
2	0.3	11.0	9.4	16.8	Read and write
3	1.1	1.8	8.3	1.8	Read
4	0.5	1.5	6.9	1.5	Read
5	2.6	21.0	21.5	32.0	Read and write
Total	4.5	37.2	47.0	54.9	

© Pearson Education 2012

Figure 10.16: Ivy system architecture



© Pearson Education 2012