

	1.	INTRODUCTION			
+	1.1.	DEFINITION OF A DISTRIBUTED SYSTEM		+	2.4.
				++	2.4.1.
				o	2.4.2.
+	1.2.	GOALS		+	2.4.3.
+	1.2.1.	Making Resources Accessible		o	2.4.4.
++	1.2.2.	Distribution Transparency			
++		Types of Transparency		o	2.5.
++		Degree of Transparency			SUMMARY
o	1.2.3.	Openness			
o		Separating Policy from Mechanism		+	3.
++	1.2.4.	Scalability			PROCESSES
++		Scalability Problems		+	3.1.
++		Scaling Techniques		+	3.1.
++	1.2.5.	Pitfalls		+	3.1.1.
				+	Introduction to Threads
				+	Thread Usage in Nondistributed Systems
				+	Thread Implementation
+	1.3.	TYPES OF DISTRIBUTED SYSTEMS		+	3.1.2.
+	1.3.1.	Distributed Computing Systems		+	Threads in Distributed Systems
+		Cluster Computing Systems		+	Multithreaded Clients
+		Grid Computing Systems		+	Multithreaded Servers
+	1.3.2.	Distributed Information Systems		+	3.2.
+		Transaction Processing Systems		+	VIRTUALIZATION
+		Enterprise Application Integration		+	3.2.1.
+	1.3.3.	Distributed Pervasive Systems		++	3.2.2.
+		Home Systems			Architectures of Virtual Machines
+		Electronic Health Care Systems		o	3.3.
+		Sensor Networks		o	3.3.1.
				o	Networked User Interfaces
				o	Example: The X Window System
o	1.4.	SUMMARY		o	Thin-Client Network Computing
				o	Compound Documents
				+	3.3.2.
					Client-Side Software for Distribution Transparency
	2.	ARCHITECTURES		+	3.4.
				+	SERVERS
++	2.1.	ARCHITECTURAL STYLES		+	3.4.1.
				+	General Design Issues
	2.2.	SYSTEM ARCHITECTURES		+	3.4.2.
++	2.2.1.	Centralized Architectures		+	Server Clusters
++		Application Layering		o	General Organization
++		Multitiered Architectures		+	Distributed Servers
+	2.2.2.	Decentralized Architectures		+	3.4.3.
++		Structured Peer-to-Peer Architectures		++	Managing Server Clusters
+		Unstructured Peer-to-Peer Architectures			Common Approaches
o		Topology Management of Overlay Networks		o	Example: PlanetLab
o		Superpeers		o	3.5.
+	2.2.3.	Hybrid Architectures		o	CODE MIGRATION
+		Edge-Server Systems		o	3.5.1.
+		Collaborative Distributed Systems		o	Approaches to Code Migration
				o	Reasons for Migrating Code
				o	Models for Code Migration
+	2.3.	ARCHITECTURES VERSUS MIDDLEWARE		o	3.5.2.
+	2.3.1.	Interceptors		o	3.5.3.
+	2.3.2.	General Approaches to Adaptive Software			Migration and Local Resources
+	2.3.3.	Discussion		o	3.6.
					SUMMARY
				+	4.
					COMMUNICATION

o	4.1.	FUNDAMENTALS	++		Removing Data
o	4.1.1.	Layered Protocols	++		Applications
o		Lower-Level Protocols			
o		Transport Protocols			
o		Higher-Level Protocols		o	4.6. SUMMARY
o		Middleware Protocols			
++	4.1.2.	Types of Communication	+	5.	NAMING
++	4.2.	REMOTE PROCEDURE CALL	++	5.1.	NAMES, IDENTIFIERS, AND ADDRESSES
++	4.2.1.	Basic RPC Operation			
++		Conventional Procedure Call	+	5.2.	FLAT NAMING
++		Client and Server Stubs	+	5.2.1.	Simple Solutions
++	4.2.2.	Parameter Passing	+		Broadcasting and Multicasting
++		Passing Value Parameters	+		Forwarding Pointers
++		Passing Reference Parameters	+	5.2.2.	Home-Based Approaches
++		Parameter Specification and Stub Generation	++	5.2.3.	Distributed Hash Tables
+	4.2.3.	Asynchronous RPC	++		General Mechanism
+	4.2.4.	Example: DCE RPC	+		Exploiting Network Proximity
+		Introduction to DCE	o	5.2.4.	Hierarchical Approaches
+		Goals of DCE RPC			
+		Writing a Client and a Server	+	5.3.	STRUCTURED NAMING
+		Binding a Client to a Server	+	5.3.1.	Name Spaces
+		Performing an RPC	o	5.3.2.	Name Resolution
o	4.3.	MESSAGE-ORIENTED COMMUNICATION	o		Closure Mechanism
o	4.3.1.	Message-Oriented Transient Communication	o		Linking and Mounting
o		Berkeley Sockets	o	5.3.3.	The Implementation of a Name Space
o		The Message-Passing Interface (MPI)	o		Name Space Distribution
o	4.3.2.	Message-Oriented Persistent Communication	++	5.3.4.	Implementation of Name Resolution
o		Message-Queuing Model	++		Example: The Domain Name System
o		General Architecture of a Message-Queuing System	++		The DNS Name Space
+		Message Brokers	++		DNS Implementation
+		A Note on Message-Queuing Systems	+		Decentralized DNS Implementations
o	4.3.3.	Example: IBM's WebSphere Message-Queuing System		5.4.	ATTRIBUTE-BASED NAMING
o		Overview	+	5.4.1.	Directory Services
o		Channels	o	5.4.2.	Hierarchical Implementations: LDAP
o		Message Transfer	+	5.4.3.	Decentralized Implementations
o		Managing Overlay Networks	+		Mapping to Distributed Hash Tables
					Semantic Overlay Networks
+	4.4.	STREAM-ORIENTED COMMUNICATION		o	5.5. SUMMARY
+	4.4.1.	Support for Continuous Media			
+		Data Stream			
+	4.4.2.	Streams and Quality of Service			
+		Enforcing QoS			
++	4.4.3.	Stream Synchronization		6.	SYNCHRONIZATION
++		Synchronization Mechanisms	+	6.1.	CLOCK SYNCHRONIZATION
++	4.5.	MULTICAST COMMUNICATION	+	6.1.1.	Physical Clocks
++	4.5.1.	Application-Level Multicasting	++	6.1.2.	Global Positioning System
++		Overlay Construction	++	6.1.3.	Clock Synchronization Algorithms
++		Gossip-Based Data Dissemination	++		Network Time Protocol
++	4.5.2.	Information Dissemination Models	++		The Berkeley Algorithm
++			++		Clock Synchronization in Wireless Networks

++	6.2.	LOGICAL CLOCKS	++	7.4.2.	Content Replication and Placement
++	6.2.1.	Lamport's Logical Clocks	++		Permanent Replicas
++		Example: Totally Ordered Multicasting	++		Server-Initiated Replicas
+	6.2.2.	Vector Clocks	++		Client-Initiated Replicas
++		Enforcing Causal Communication	++	7.4.3.	Content Distribution
+		A Note on Ordered Message Delivery	++		State versus Operations
			++		Pull versus Push Protocols
			++		Unicasting versus Multicasting
o	6.3.	MUTUAL EXCLUSION	++	7.5.	CONSISTENCY PROTOCOLS
o	6.3.1.	Overview	+	7.5.1.	Continuous Consistency
o	6.3.2.	A Centralized Algorithm	++		Bounding Numerical Deviation
o	6.3.3.	A Decentralized Algorithm	+		Bounding Staleness Deviations
o	6.3.4.	A Distributed Algorithm	o		Bounding Ordering Deviations
o	6.3.5.	A Token Ring Algorithm	++	7.5.2.	Primary-Based Protocols
o	6.3.6.	A Comparison of the Four Algorithms	++		Remote-Write Protocols
+	6.4.	GLOBAL POSITIONING OF NODES	++		Local-Write Protocols
o	6.5.	ELECTION ALGORITHMS	++	7.5.3.	Replicated-Write Protocols
o	6.5.1.	Traditional Election Algorithms	++		Active Replication
o		The Bully Algorithm	++		Quorum-Based Protocols
o		A Ring Algorithm	+	7.5.4.	Cache-Coherence Protocols
o	6.5.2.	Elections in Wireless Environments	o	7.5.5.	Implementing Client-Centric Consistency
+	6.5.3.	Elections in Large-Scale Systems	o		A Naive Implementation
			o		Improving Efficiency
o	6.6.	SUMMARY	o	7.6.	SUMMARY
++	7.	CONSISTENCY AND REPLICATION	+	8.	FAULT TOLERANCE
++	7.1.	INTRODUCTION	+	8.1.	INTRODUCTION TO FAULT TOLERANCE
++	7.1.1.	Reasons for Replication	+	8.1.1.	Basic Concepts
++	7.1.2.	Replication as Scaling Technique	+	8.1.2.	Failure Models
++	7.2.	DATA-CENTRIC CONSISTENCY MODELS	+	8.1.3.	Failure Masking by Redundancy
++	7.2.1.	Continuous Consistency	+	8.2.	PROCESS RESILIENCE
++		The Notion of a Conit	+	8.2.1.	Design Issues
++	7.2.2.	Consistent Ordering of Operations	+		Flat Groups versus Hierarchical Groups
++		Sequential Consistency	+		Group Membership
++		Causal Consistency	+	8.2.2.	Failure Masking and Replication
++		Grouping Operations	o	8.2.3.	Agreement in Faulty Systems
++		Consistency versus Coherence	+	8.2.4.	Failure Detection
++	7.3.	CLIENT-CENTRIC CONSISTENCY MODELS	+	8.3.	RELIABLE CLIENT-SERVER COMMUNICATION
++	7.3.1.	Eventual Consistency	+	8.3.1.	Point-to-Point Communication
++	7.3.2.	Monotonic Reads	++	8.3.2.	RPC Semantics in the Presence of Failures
++	7.3.3.	Monotonic Writes	++		Client Cannot Locate the Server
++	7.3.4.	Read Your Writes	++		Lost Request Messages
++	7.3.5.	Writes Follow Reads	++		Server Crashes
++	7.4.	REPLICA MANAGEMENT	++		Lost Reply Messages
++	7.4.1.	Replica-Server Placement	++		Client Crashes

+	8.4.	RELIABLE GROUP COMMUNICATION	skip		Secure Replicated Servers
+	8.4.1.	Basic Reliable-Multicasting Schemes	skip	9.2.4.	Example: Kerberos
+	8.4.2.	Scalability in Reliable Multicasting			
+		Nonhierarchical Feedback Control	skip	9.3.	ACCESS CONTROL
+		Hierarchical Feedback Control	skip	9.3.1.	General Issues in Access Control
++	8.4.3.	Atomic Multicast	skip		Access Control Matrix
++		Virtual Synchrony	skip		Protection Domains
o		Message Ordering	skip	9.3.2.	Firewalls
skip		Implementing Virtual Synchrony	skip	9.3.3.	Secure Mobile Code
			skip		Protecting an Agent
++	8.5.	DISTRIBUTED COMMIT	skip		Protecting the Target
++	8.5.1.	Two-Phase Commit	skip	9.3.4.	Denial of Service
++	8.5.2.	Three-Phase Commit			
			skip	9.4.	SECURITY MANAGEMENT
+	8.6.	RECOVERY	skip	9.4.1.	Key Management
+	8.6.1.	Introduction	skip		Key Establishment
o		Stable Storage	skip		Key Distribution
+	8.6.2.	Checkpointing	skip		Lifetime of Certificates
+		Independent Checkpointing	skip	9.4.2.	Secure Group Management
+		Coordinated Checkpointing	skip	9.4.3.	Authorization Management
+	8.6.3.	Message Logging	skip		Capabilities and Attribute Certificates
+		Characterizing Message-Logging Schemes	skip		Delegation
+	8.6.4.	Recovery-Oriented Computing			
			skip	9.5.	SUMMARY
o	8.7.	SUMMARY			
			+	10.	DISTRIBUTED OBJECT-BASED DISTRIBUTED SYSTEMS
skip	9.	SECURITY	o	10.1.	ARCHITECTURE
skip	9.1.	INTRODUCTION TO SECURITY	o	10.1.1.	Distributed Objects
skip	9.1.1.	Security Threats, Policies, and Mechanisms	o		Compile-time versus Runtime Objects
skip		Example: The Globus Security Architecture	o		Persistent and Transient Objects
skip	9.1.2.	Design Issues	o	10.1.2.	Example: Enterprise Java Beans
skip		Focus of Control	skip	10.1.3.	Example: Globe Distributed Shared Objects
skip		Layering of Security Mechanisms	skip		Object Model
skip		Distribution of Security Mechanisms			
skip		Simplicity	++	10.2.	PROCESSES
skip	9.1.3.	Cryptography	++	10.2.1.	Object Servers
skip		Symmetric Cryptosystems: DES	++		Alternatives for Invoking Objects
skip		Public-Key Cryptosystems: RSA	++		Object Adapter
skip		Hash Functions: MD5	++	10.2.2.	Example: The Ice Runtime System
skip	9.2.	SECURE CHANNELS	o	10.3.	COMMUNICATION
skip	9.2.1.	Authentication	o	10.3.1.	Binding a Client to an Object
skip		Authentication Based on a Shared Secret Key	o		Implementation of Object References
skip		Authentication Using a Key Distribution Center	o	10.3.2.	Static versus Dynamic Remote Method Invocations
skip		Authentication Using Public-Key Cryptography	o	10.3.3.	Parameter Passing
skip	9.2.2.	Message Integrity and Confidentiality	+	10.3.4.	Example: Java RMI
skip		Digital Signatures	+		The Java Distributed-Object Model
skip		Session Keys	+		Java Remote Object Invocation
skip	9.2.3.	Secure Group Communication	skip	10.3.5.	Object-based Messaging
skip		Confidential Group Communication			

skip	10.4.	NAMING	skip	11.6.1.	Client-side Caching
skip	10.4.1.	CORBA Object References	skip		Caching in NFS
skip	10.4.2.	Globe Object References	skip		Client-side caching in Coda
skip	10.5.	SYNCHRONIZATION	skip		Client-side Caching for Portable Devices
+	10.6.	CONSISTENCY AND REPLICATION	skip	11.6.2.	Server-side Replication
+	10.6.1.	Entry Consistency	skip		Server Replication in Coda
skip		Replication Frameworks	skip	11.6.3.	Replication in Peer-to-Peer File Systems
+	10.6.2.	Replicated Invocations	skip		Unstructured Peer-to-Peer Systems
skip	10.7.	FAULT TOLERANCE	skip		Structured Peer-to-Peer Systems
skip	10.7.1.	Example: Fault-Tolerant CORBA	skip	11.6.4.	File Replication in Grid Systems
skip		An Example Architecture	skip	11.7.	FAULT TOLERANCE
skip	10.7.2.	Example: Fault-Tolerant Java	skip	11.7.1.	Handling Byzantine Failures
skip	10.8.	SECURITY	skip	11.7.2.	High Availability in Peer-to-Peer Systems
skip	10.8.1.	Example: Globe	skip	11.8.	SECURITY
skip		Overview	skip	11.8.1.	Security in NFS
skip		Secure Method Invocation	skip		Secure RPCs
skip	10.8.2.	Security for Remote Objects	skip		Access Control
skip	10.9.	SUMMARY	skip	11.8.2.	Decentralized Authentication
skip	11.	DISTRIBUTED FILE SYSTEMS	skip	11.8.3.	Secure Peer-to-Peer File-Sharing Systems
skip	11.1.	ARCHITECTURE	skip		Secure Lookups in DHT-based Systems
skip	11.1.1.	Client-Server Architectures	skip		Secure Collaborative Storage
skip		File System Model	o	11.9.	SUMMARY
skip	11.1.2.	Cluster-Based Distributed File Systems	+	12.	DISTRIBUTED WEB-BASED SYSTEMS
skip	11.1.3.	Symmetric Architectures	+	12.1.	ARCHITECTURE
skip	11.2.	PROCESSES	+	12.1.1.	Traditional Web-based Systems
skip	11.3.	COMMUNICATION	+		Web Documents
skip		RPCs in NFS	++		Multitiered Architectures
skip		The RPC2 Subsystem	o	12.1.2.	Web Services
skip		File-oriented Communication in Plan 9	o		Web Services Fundamentals
skip	11.4.	NAMING	o		Web Services Composition and Coordination
skip	11.4.1.	Naming in NFS	o	12.2.	PROCESSES
skip		File Handles	skip	12.2.1.	Clients
skip		Automounting	+	12.2.2.	The Apache Web Server
skip	11.4.2.	Constructing a Global Name space	+	12.2.3.	Web Server Clusters
skip	11.5.	SYNCHRONIZATION	skip	12.3.	COMMUNICATION
skip		Semantics of File Sharing	skip	12.3.1.	Hypertext Transfer Protocol
skip		File Locking	skip		HTTP Connections
skip		Sharing Files in Coda	skip		HTTP Methods
skip	11.6.	CONSISTENCY AND REPLICATION	skip		HTTP Messages
			skip	12.3.2.	Simple Object Access Protocol
			skip	12.4.	NAMING
			skip	12.5.	SYNCHRONIZATION

++	12.6.	CONSISTENCY AND REPLICATION	skip		Example: Fault Tolerance in TIB/Rendezvous
++	12.6.1.	Web Proxy Caching	skip	13.8.2.	Fault Tolerance in Shared Dataspaces
++	12.6.2.	Replication for Web Hosting Systems	skip	13.9.	SECURITY
++		Metric Estimation	skip	13.9.1.	Confidentiality
++		Adaptation Triggering	skip		Decoupling Publishers from Subscribers
++		Adjustment Measures	skip	13.9.2.	Secure Shared Dataspaces
++	12.6.3.	Replication of Web Applications			
skip	12.7.	FAULT TOLERANCE	o	13.10.	SUMMARY
skip	12.8.	SECURITY			
o	12.9.	SUMMARY			
+	13.	DISTRIBUTED COORDINATION-BASED SYSTEMS			
+	13.1.	INTRODUCTION TO COORDINATION MODELS			
+	13.2.	ARCHITECTURES			
+	13.2.1.	Overall Approach			
+	13.2.2.	Traditional Architectures			
+		Example: Jini and JavaSpaces			
+		Example: TIB/Rendezvous			
skip	13.2.3.	Peer-to-Peer Architectures			
skip		Example: A Gossip-based Publish/Subscribe System			
skip		Discussion			
skip	13.2.4.	Mobility and Coordination			
skip		Example: Lime			
skip	13.3.	PROCESSES			
++	13.4.	COMMUNICATION			
++		Content-based Routing			
skip		Supporting Composite Subscriptions			
skip	13.5.	NAMING			
skip		Describing Composite Events			
skip		Matching Events and Subscriptions			
skip	13.6.	SYNCHRONIZATION			
++	13.7.	CONSISTENCY AND REPLICATION			
++	13.7.1.	Static Approaches			
++		General Considerations			
++	13.7.2.	Dynamic Replication			
+		GSpace Overview			
+		Adaptive Replication			
skip	13.8.	FAULT TOLERANCE			
skip	13.8.1.	Reliable Publish-Subscribe Communication			

