System Support for Mobile Wireless Computing

Evaggelia Pitoura

Computer Science Department,
University of Ioannina,
Ioannina, Greece
http://www.cs.uoi.gr/~pitoura

Summer School, Jyvaskyla, August 1998

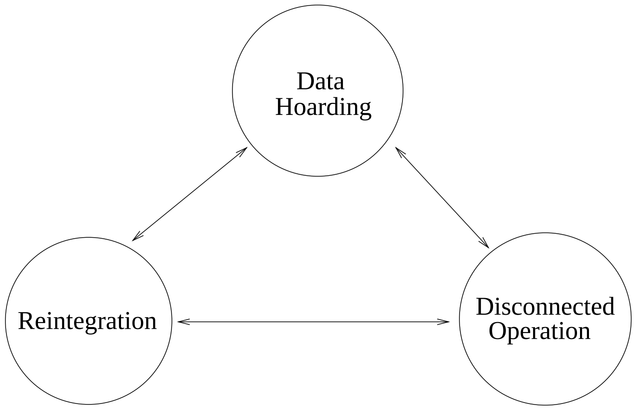
Introduction

Support for:

- *Disconnections:* disconnected operation - autonomous operation of a mobile host during disconnection.
- *Weak connectivity:* Operation should be tuned for communication environments characterized by low bandwidth, high latency, and expensive prices.
- *Mobility:* Basic support such as as establishing new communication links as well as advanced support such as migrating executing processes and database transactions in progress.
- *Failure recovery:* Since mobile elements are prone to hard failures, methods for failure handling and recovery are important.

Disconnected Operation

Can be described as a transition between three states [42]



Data Hoarding State

Data items needed for operation are preloaded into the mobile unit.

Simply relocate data: inaccessible to other sites
Replicate or cache data: consistency control

Type of data objects: depends on the application and the underlying data model.

For foreseeable disconnections, performed just before the disconnection. Else, on a regular basis, e.g., periodically.

How to anticipate future needs

– users explicitly specify which data

– use past history of data accesses

– based on the application for which the system will be used

Disconnected Operation

Disconnected State

Use only locally available data

Requests for other data may be inserted in an appropriate queue to be serviced upon reconnection.

Applications with unsatisfied data requests can either suspend execution or continue working on some other job.

Regarding updates:

– Pessimistic approach: updates only at one site (lock or check-in/check-out)

– Optimistic approach: updates at more than one site but conflicts.

– Updates at the mobile unit are logged in client's stable storage.

What information to keep in the log for effective reintegration and log optimizations

Keep the size of the log small to: (a) save memory, and (b) reduce the time for update propagation and reintegration at reconnection.

Optimization either (a) during disconnected operation, incrementally each time a new operation is inserted in the log, or (b) as a preprocessing step before propagating or applying the log upon reconnection.

Disconnected Operation

Reintegration State

Updates at the mobile host are reintegrated with updates at other sites by re-executing the log at the fixed host.

Summary

	Issue	Approach
Hoarding	Unit of hoarding	Depends on the system (e.g,. a file or a database fragment)
	Which items to hoard	Specified explicitly by the user Induced implicitly from the history of past operations Depends on the application for which the system is used
	When to perform hoarding	Prior to disconnection On a regular basis
Disconnection	Request for data not locally available	Raise an exception/error Requests are queued for future service
	What to log	Data Values Timestamps Operations
	When to optimize the log	Incrementally Prior to integration
	How to optimize the log	Depends on the system
Reintegration	How to integrate	Re-execute an operational log
	How to resolve conflicts	Use application-semantics Automatic resolution Provide tools to assist the user

DO in File Systems

Extend caching to handle disconnections

Hoarding vs Prefetching

– prefetching is an ongoing process that transfers to the cache soon-to-be-needed files during periods of low network traffic. thus, keeping its overhead low is important.

– hoarding is more critical than prefetching. Thus, hoarding tends to overestimate the client's need for data. (although, excessive estimations cannot be satisfied)

Unit of Hoarding:

ranges from a disk block, to a file, to groups of files or directories

When to Hoard

The Coda file system [42] runs a process called *hoard walk* periodically to ensure that critical files are in the cache.

Which Files to Hoard

– In Coda [42], data are prefetched using priorities based on a combination of recent reference history and user-defined hoard files.

– A tree-based method [74]: processes the history of file references to build an execution tree. The nodes of the tree represent the programs and data files referenced. An edge exists from parent node A to child node B, when either program A calls program B, or program A uses file B.

DO in File Systems

– Seer [46] predictive caching scheme: files are automatically prefetched based on a measure called semantic distance that quantifies how closely related they are. The measure is the local reference distance from a file A to a file B, defined as the number of file references separating two adjacent references to A and B in the history.

Cache Misses During Disconnection

– treated as errors and raise exceptions

Log Optimizations

– In Coda, a replay log that records all corresponding system call arguments as well as the version state of all objects referenced by the call [42]. Two optimizations before a new record is appended: (1) any operation which overwrites the effect of earlier operations may cancel the corresponding records (2) an inverse operation (e.g., rmdir) cancels both the inverting and inverted (e.g., mkdir) log records.

– The Little Work project [31] suggests applying rule-based techniques used in compiler peephole optimizers. Use such an off-the-shelf optimizer. Optimization at a preprocessing step before reintegrating at reconnection. Two types of rules: (1) replacement rules remove adjacent redundant operations, e.g., a create followed by a move. (2) ordering rules reorder adjacent operations to apply further replacement rules.

DO in File Systems

Conflicting Updates

– Cache updates are considered tentative.

– In Coda, the replay log is executed as a single transaction. All objects referenced in the log are locked.

Different strategies for handling concurrent updates on files and on directories [47, 32]

– *Directory resolution* fails only if a newly created name collides with an existing name, if an object updated at the client or the server has been deleted by the other, or if directory attributes have been modified at the server and the client [47].

– *File resolution* is based on application-specific resolvers (ASRs) per file [48]: programs that encapsulate the knowledge needed for file resolution, invoked at the client when divergence among copies is detected. Rules are used to select the appropriate ASR. The ASR's mutations are performed locally on the client's cache and written back to the server atomically after the ASR completes. The execution of an ASR is guaranteed transaction semantics. If no ASR is found or the ASR execution fails, an error code indicating a conflict is returned. For cases of unresolved conflicts, a manual repair tool is run on the client.

– Only write/write conflicts are considered

DO in Database Systems

Granularity of Hoarding

– In relational database systems, ranges from tuples, to set of tuples, to whole relations.

– In object-oriented database systems, at the object, set of objects or class (extension) level.

Hoard by Issuing Queries

– prefetch the data objects that constitute the answer to a given query.

– corresponds to loading on the mobile unit materialized views

– operation during disconnection by posing queries against these views

Which Items to Hoard

– or how to identify which views to materialize, or specify the hoarding queries

– users may explicitly issue hoarding queries

– use the history of past references to deduce dependencies among database items is hard

– issues related to integrity and completeness

– In [24]: (a) users use an object-oriented query to describe hoarding profiles, and (b) a history of references is maintained by using a tracing tool that records queries as well as objects.

DO in Database Systems

Hoard Keys [4]

An extended database organization: a set of *hoard keys* along with the primary and secondary key for each relation. Hoard keys capture typical access patterns of mobile clients. Each hoard key partitions the relation into a set of disjoint logical horizontal fragments. Hoard fragments constitute the hoard granularity. The hoard keys can either be logical or physical.

Network Partition vs DO

Network partition: a network failure partitions the sites of a distributed database system into disconnected clusters.

Various approaches [14] wrt the trade-off between consistency and availability. and the level of semantic knowledge

– In network partition, transactions executed at any partition of equal importance while, in mobile computing transactions at the mobile host most often consider second-class.

– frequency of disconnections, network partitions correspond to failure behavior,

– some disconnections in mobile computing can be considered foreseeable.

DO in Database Systems

Tentative Commitment

Tentatively commit transactions at a disconnected unit and make their results visible to subsequent transactions in that unit.

Upon reconnection, a certification process, during which the execution of any tentatively committed transaction is validated against an application or system defined correctness criterion.

If the criterion is met, the transaction is committed. Otherwise, it must be aborted, reconciled or compensated.

Such actions may have cascaded effects on other tentatively committed transactions.

Isolation-only transactions (IOTs) [52, 53]

An IOT is a sequence of file access operations. A transaction T is called a *first-class transaction* if it does not have any partitioned file access, i.e., the client machine maintains a connection for every file it has accessed. Otherwise, T is called a *second-class transaction.*

The result of a first-class transaction is immediately committed to the servers.

A second-class transaction remains in the pending state till connectivity is restored. Its result is held within the client's local cache and visible only to subsequent accesses on the same client. They are guaranteed to be locally serializable among themselves.

DO in Database Systems

A first-class transaction is guaranteed to be serializable with all transactions previously resolved or committed at the server.

Upon reconnection, a second-class transaction T is validated against one of two proposed serialization constraints: global serializability, or global certifiability. Transactions that cannot meet these criteria are resolved incrementally, i.e., one by one, according to their local serialization order. An invalidated transaction is aborted, reexecuted or an ASR is automatically invoked. Manual resolution or repair is left as a last resort.

Two-Tier Replication [22]

Replicated data have two versions at mobile nodes: master and tentative versions. A master version records the most recent value received while the site was connected. A tentative version records local updates.

Two types of transactions. A *tentative transaction* works on local tentative data and produces tentative data. A *base transaction* works only on master data and produce master data. Base transactions involve only connected sites.

Upon reconnection, tentative transactions are reprocessed as base transactions. If they fail to meet some application-specific acceptance criteria, they are aborted and a message is returned to the mobile node.

DO in Workflow Systems

Workflow processes: long-running tasks involving the coordinated execution of multiple activities performed by different processing entities.

Relevant terms: activities, data containers, rules or transition conditions

Workflow Management System: provides a (a) *modeling tool* to represent business processes via workflows and (b) a *runtime system* to coordinate and execute them in a heterogeneous and distributed environment.

Workflow execution and disconnected operation have contradictory goals

Note: The processing entities where the activities are executed are distributed, but the workflow system that coordinates their execution may be centralized or distributed.

Disconnections among the processing entities also called clients.

What functionality should be placed on disconnected clients. Replicating part of the functionality of the workflow server on disconnected clients complicates operation significantly especially since some applications may involve thousands of clients [2].

DO in Workflow Systems

– *The Exotica workflow* system [2]

assigns no additional functionality to disconnected clients.

Pessimistic approach: prior to disconnection, a client reserves by locking the activities it plans to work on and hoards the relative to the activities data.

Implementation based on worklists.

A *workitem* is an activity that belongs to a workflow process being executed.

A *worklist* is a list of workitems associated with a user.

Every time an activity is eligible for execution, it is broadcasted to the worklist of the users associated with this activity. When a user selects an activity from its worklist, the activity is deleted from all other worklists and is associated with that user.

Before disconnection, a user selects from its worklist and locks the activities to work with plus the input data container.

During disconnection, it works only on the activities it has locked. When the execution of an activity terminates, its results are stored locally in client's stable storage.

Upon reconnection, the results of all terminated activities are reported back to the server to be stored in the central database.

DO in Workflow Systems

The pessimistic approach taken by Exotica is justified by the high degree of data sharing in workflows and cooperative work.

Else need for a sophisticated activity merging mechanism, e.g., history merging [43] of the TransCoop project.

INCAS [8].

A new workflow model based on mobile agents called Information Carriers (INCAs).

INCAs are mobile agents that model a workflow process. There is no centralized workflow server. Instead, each INCA controls its own execution.

In particular, each INCA encapsulates the private data of the workflow, a set of rules that control the flow between the activities of the INCA computation, and the history (log) of its execution.

Each INCA is initially submitted to a processing entity, and roams among processing entities to achieve its goal. Being mobile agents, INCAs support operation among partly connected or disconnected processing entities. The execution of an INCA is guarantee transactional semantics.

DO in Web-Based Systems

Cache management to meet the needs of wired users where ad hoc browsing is common and cache methods can trade inexpensive network bandwidth for reducing storage [9, 54, 20].

Mostly used for browsing, no updates (vs files)

Which Pages to Prefetch/Hoard

Prefetch based on the user's surfing history list. Studies [41] indicate 2% usability of the prefetched objects.

To support disconnected operation, this policy can be extended by fetching all documents of pages visited previously or of pages anticipated to be visited based on the user's profile.

In [41], the HTTP server monitors access patterns. Then, it periodically augments objects with information about what objects a client should prefetch, given that the client requested a particular document.

Browser Sessions

Cache methods that either (a) purge the cache at the end of a browser session, or (b) let cached objects persist across sessions with updates occurring once on first reference per session.
Also, advanced options that cause cache objects to be updated on every reference or never to be updated.

Cross-session persistence of cached objects is critical [41, 28] during disconnections

DO in Web-based Systems

Cache Updates

Optimistic techniques [41, 28] that look for changes to stored objects based on elapsed time

coherency interval to measure the elapsed time that permits web users to adjust the update frequency of cached web-objects

Cache Misses

In [28], an "asynchronous-disconnected" mode that permits requests to be automatically queued when connectivity is lost and resumed when connectivity is re-established. Users can issue multiple web requests without having to wait for their replies. Arriving responses are queued.

Similarly, Rover [38] uses queued RPC to develop a non-blocking operation for browsers to allow users to click ahead. Requests are stored in the client's "operation log" that is executed at reconnection. The client is notified, when its request is satisfied.

Cache Validation at Reconnection

A single asynchronous coherency check for all cached objects that are older than the coherency interval, or

by initiating coherency checks only on object references.

Weak Connectivity

Connectivity provided by slow or expensive networks.

Intermittent connectivity: often lost for short periods of time.

In addition, in wireless computing, connectivity varies in cost, provided bandwidth and reliability.

Support for operation that *adapts* to the current degree of connectivity. In such systems, disconnected operation is just the form of operation in the extreme case of total lack of connectivity.

The aim of most proposals for weak connectivity is prudent use of bandwidth. They are willing to trade off “fidelity” against a reduction in communication cost.

WC in File Systems

- handling of cache misses,
- frequency of propagation to the server of updates performed at the client's cache, and
- the validity of the value of cached items

Cache Miss

Selective service of cache misses based on how critical the required item is or on the current connectivity.

Cache Updates

Early reintegration:

- reduces the effectiveness of log optimizations (records have less opportunity to be eliminated at the client)
- affects the response times of other traffic
+ achieves consistent cache management, timely propagation of updates and reduces the probability of conflicting operations.
+ keeps the size of the log in the client's memory short,
+ avoids the possibility of a client's cache overflow

WC in File Systems

Update of Cached Objects

Notify the client each time an item is updated at the server OR on demand, each time a client issues a read operation

Cache Validation

When connectivity is intermittent, the client must validate its cache.

Increase the granularity of cache coherency [55]: each server maintains version stamps for volumes, i.e., sets of files, in addition to stamps for individual objects

Fetch-Only Operation [27]

No continuous network connectivity

Attractive when the network has an associated charge for connect time, e.g., over a cellular phone or ISDN.

In this mode, cache updates are deferred and no cache consistency protocol is used. The network is used only to satisfy cache misses.

	Connected	**Disconnected**	**Weakly Connected**	**Fetch-only**
Cache Miss	Serviced	Raise an exception	Serviced, or Selectively serviced	Serviced
Propagation of Local Updates	Immediate	Upon reconnection	Differed propagation (e.g, periodic, or when the network traffic is low)	As when disconnected
Value Read	Value in cache (most recent)	Value in cache (possibly stale)	Value in cache (recent or stale), or Contact the server	As when disconnected

WC in File Systems

– *Coda* [56]

Cache misses are serviced selectively: a file is fetched only if the service time for the cache miss which depends among others on bandwidth is below the user's patience threshold for this file

Trickle reintegration: an ongoing background process that propagates updates to servers asynchronously.

Aging: a record is not eligible for reintegration until it spends a minimal amount of time, called aging window, in the log.

Adaptive reintegration chunk size

– *Little Work project* [33]

Background update propagation.

Three levels of queueing priority in the network drivers: interactive traffic, other network traffic, and replay traffic. A number of tickets assigned to each queue.

Immediate propagation of file updates to the client through callbacks. Directory updates are tricky, thus clients use only the locally updated directory.

Cache misses are always serviced.

WC in File Systems

– *The Variable-Consistency Approach* [75, 76]

A client/server architecture with replicated servers. The client communicates with the primary server only.

The primary makes periodic pickups from the clients and propagates updates back to the secondaries asynchronously. Once some number N of secondaries have acknowledged receipt of an update, the primary informs the client that the associated cached update has been successfully propagated and can be discarded.

Loose read returns the value of the cache copy, if such a copy exists; otherwise, the value of any copy. *Strict read* returns the most recent value by contacting the necessary number of servers and clients.

– *Ficus and its descendant Rumor* [25, 26, 68]

Peer-to-peer architecture

Organized as a directed acyclic graph of volumes. A *volume* is a logical collection of files that are managed collectively and typically share replication characteristics.

Pair-wise reconciliation algorithm executed periodically and concurrently with normal file activity: the state of the local replicated volume is compared to that of a single remote replica of the volume. The procedure continues till updates are propagated to all sites storing replicas of the volume.

WC in Database Systems

Approaches

– the mobile host simply submits operations to be executed on a server or an agent at the fixed network [36, 84, 16].

either operations of a transaction sequentially one at a time [36] or the complete transaction as one atomic unit [84]

– database processing locally at the mobile host

raises physical database design issues, e.g., how to appropriately fragment the database and allocate fragments at fixed and mobile hosts

Mobile Transactions

distributed transactions that involve both mobile and fixed hosts

– traditional approaches (locks or timestamps)

– open-nested transaction models [13]: a mobile transaction is modeled as a set of relatively independent component transactions some of which run solely at the mobile host
Component transactions can commit without waiting for the commitment of other component transactions.

WC in Database Systems

Weak and Strict Transactions [63, 62, 64]

Two types of copies: *weak copies* and *strict copies.* Weak transactions local to a mobile host and update weak copies, strict transactions access strict copies. Weak copies are integrated with strict when connectivity improves or to enforce an application-defined limit to the allowable deviation among copies.

Strict transactions are slower than weak transactions but guarantee permanence of updates and currency of reads. *Adaptability* by adjusting the number of strict transactions or the degree of divergence among copies. During disconnection, only weak transactions.

Bayou [77, 78, 15]

Peer-to-peer architecture with a number of replicated servers weakly connected to each other. Read-any and write-any available copy Writes are propagated to other servers during pair-wise contracts called anti-entropy sessions.

Sessions instead of transactions: a session is a sequence of read and write operation. *Session guarantees* o avoid inconsistencies when accessing copies at different servers; e.g.,, read operations reflect previous writes.

Adaptivity by individually selectable session guarantees, choices of committed or tentative data, age parameters on reads.

Arbitrary disconnections

WC in Database Systems

Complex Data Items

Split large or complex objects into smaller fragments, so that operation at each of the fragments can proceed relatively independently [45, 79].

Site escrow methods [57, 44, 45]: the total number of instances of a given item is partitioned across a number of sites.

The *fragmentation approach* [79]: a master copy of a large object at the fixed network split into smaller physical fragments logically removed from the copy and loaded at the mobile host. Type-specific merge procedure to re-assemble the fragments.

Distributed Object Repositories

Organize data items as a collection of objects. Objects become the unit of caching and replication. Very flexible, e.g, objects can encapsulate conflict resolution procedures. Can be built on top of an existing system.

In Rover [39], relocatable dynamic objects (RDO). Each RDO has a home server that maintains the primary canonical copy. Clients import secondary copies of RDOs in their local caches.

In the *Pro-motion infrastructure* [80], the unit of caching and replication is a compact. A *compact*: an object that encapsulates the cached data, operations for accessing them, state information (e.g., number of accesses to the object), consistency rules to guarantee consistency, and obligations (such as deadlines).

WC in Web-Based Systems

– *Server Cache*

A cache at the fixed network in addition to the client's cache.

Prefetching at this cache hides some latency without affecting the wireless link.

Can facilitate user-specified data reductions. E.g.,, a filtering script that selects documents that match some criteria written by the user at the palmtop [82] and executed remotely by examining cached objects at the server's cache.

– *Distillation and Refinement*

Inline graphics add to the latency of loading web pages [67].

Distillation [19, 18]: a highly lossy, real-time, datatype-specific compression that preserves most of the semantic content of a document. If the client needs to see the full or part of the original image, *refinement* allows the user to make such a request.

An agent proxy, called Pythia distills and forwards an image to the client, and then caches a copy of the original image locally in case the client requests refinement. Users can register their capabilities with the Pythia proxy, thus distillation tuned towards their configuration or current connectivity.

WC in Web-Based Systems

– *Connection overhead*

Each request requires the browser to open a TCP/IP socket, this increases user response time and network traffic [67].

Thus, web prefetching or pipelining might cause more problems than benefits, especially if usability of the prefetched objects is low.

A single, long-lived TCP/IP connection between the mobile browser and the fixed network [50, 28] is required.

– *Differencing* [28]

Takes advantage of the fact that only small parts of cached objects are modified

When an application requests an object, the object is cached at both the client and the server. Both cache a common base object.

When a client requests an outdated cached object (an object whose coherency interval has expired), the request is directed to the server's cache. If this also outdated, then to the web server for a fresh copy.

The server computes the difference between its local copy and the response and sends the difference to the client. The client merges the difference with its cached object to create the browser's response.

WC in Web-Based Systems

– *Prefetching and Pipelining.*

Pipelining: a form of prefetching [67].

A modification of the GET method, called GETALL. GETALL: request from the web server to send not only the requested document, but also all of its inline images residing on that server.

GETLIST: selectively request a set of documents or images from the server (i.e., request those documents that are not in cache)

An HTTP batch get [50] by having the proxy prefetching the inline documents of a requested HTML page that are not present in its cache.

Dynamic sets [72]: explicit grouping of file accesses and communication of this grouping by the application to the operating system.

	Connected	**Disconnected**	**Weakly Connected**
Cache Miss	Serviced	Queued	Queued Serviced
Value Read	Object in cache (If outdated or not in cache, fetch from server)	Object in cache (ignore coherency interval)	Object in cache, or Delayed fetch from server, or Fetch difference from server, or Background prefetching, or Fetch distilled/refined object
Validation of cached objects	Immediate, or Based on coherency intervals	Upon reconnection, Validate : all cached objects or on a per-object basis when an object is referenced	Based on coherency intervals

Mobility

Algorithms for dynamic data and task distribution

Placing Data

Variety of Approaches:

In [30], a distinction between the replication cost in stationary and mobile computing, assuming that in mobile computing the I/0 cost becomes insignificant, because of wireless communication charges. A dynamic data allocation algorithm: whenever a processor reads a copy, it saves the copy in its local database.

Whether to maintain or not a (read-only) local copy of an item [29]. Two different cost models: a connection-based, where the user is charged per minute and a message-based, where the user is charged per message.

Three alternative locations for placing copies: the server, the mobile client, and the location servers for the client [3]. Main consideration: the search cost to locate copies at mobile hosts.

Dynamic algorithms for replicated data placement simply by letting transactions update the directory [7] To locate copies, transactions first obtain read locks on the directory. A change in the placement of copies is an update transaction on the directory.

Mobility

Moving Computations

Issues

– transparently from the associated application or be application-aware

– location-dependent or location-independent (this case includes moving computation on and off a mobile client)

– after a unit of computation has been completed (e.g., after the transmission of a particular URL request or a transaction) OR while still in progress

– transmit also *context*: enough information so that the computation can continue correctly

– How often does a computation move?

Involves various costs, including the cost of transferring the context and of informing other sites about the movement.

Mobility in the Various Models

Service hand-off. Surrogate vs Service-specific.

In mobile agents, the programming language or the operating system provides primitives to create and move agents. Tcl [61] and Java [21] move code; Telescript agents [83] + the state of execution and are re-animated at each location; Obliq [10] + active network connections with remote agents.

Failure Recovery

Emphasis on recording consistent global checkpoints [59, 65, 1]

Main issues under consideration:

- the mobility of the mobile host from cell to cell: find the best place to store the next local checkpoint
- the availability of stable storage at the mobile host that determines the degree of participation of the mobile host in the checkpointing process
- bandwidth, which in conjunction with the storage availability refines the participation of the mobile host in checkpoints and affects the frequency of taking such checkpoints

Checkpoint protocols [12]: periodically store the state of the application in stable storage. After failure, the application uses these checkpoints to roll back to the last saved stable state and restart execution.

A *global state* includes the state of each process participating in the distributed application and possibly some messages.

A global checkpoint Gl_{Ckpt} is defined [12, 1] as consisting of a local checkpoint for each one of the mobile hosts/processes participating in the application.

Failure Recovery

For correct recovery, the protocol must save a recoverable consistent global state.

Consistency : A global checkpoint is "consistent" if the following condition holds: for any message m, if $rcv(m)$ is included in the checkpoint Gl_{Chkt}, then send(m) is also included in the checkpoint.

Recoverability : To avoid loss of in-transit messages, that is messages that were sent but not received by any other process, if the Gl_{Chkt} contains the $send(m)$ event but not the $rcv(m)$ event then the checkpoint protocol must save message m as well.

Protocols classified into:

Coordinated protocols [12, 58]: require the participants to coordinate their local checkpoints to ensure a consistent and recoverable global checkpoint

Uncoordinated protocols [73, 37, 40, 17, 81]: allow the participants to independently checkpoint their local state. During recovery a coordinated effort is required to select one checkpoint from each participant to create a consistent global checkpoint.

Failure Recovery

Coordinated Protocols and Mobility

- require sending control messages to different mobile hosts (MHs) to synchronize the checkpointing process

  search cost to locate the MH.

  the MH mobile host might move to another cell before the checkpointing process is completed
- during disconnection, the MH's local checkpoint is inaccessible to the synchronous checkpointing algorithm

Uncoordinated Protocols and Mobility

- allow mobile participants to checkpoint their state without exchanging messages
- more information needs to be stored by the protocol to guarantee deterministic execution
- additional messages need to be exchanged during recovery to find the global checkpoint [37], to gather information about other participants [17], or to garbage-collect stored information [81].

Soft vs Hard Failures

Soft failures: do not permanently damage the mobile host, such as battery discharge or operating system crashes

Failure Recovery

Hard failures are handled by *hard checkpoints* that are stored on the fixed network, while soft failures are dealt by *soft checkpoints* that are stored locally on the mobile host.

Locally stored checkpoints do not consume bandwidth, are easily created and allow the mobile host to continue functioning during disconnections.

	Connected	**Disconnected**	**Weak Connectivity**
Logging Type	Immediate or periodic	Periodic	Periodic
Ratio of Checkpoints	#hard ~= #soft	Only soft	# soft > # hard
Coordination Type	Coordinated or uncoordinated	Uncoordinated	Uncoordinated
Light-weight Client (stable storage unsafe)	Balance or minimize client's use	Maximize client's use (only client's memory)	Maximize client's use
Heavy-weight Client (stable storage safe)	Balance or maximize client's use	Maximize client's use (only client's memory)	Maximize client's use

Failure Recovery

[1, 65] present three uncoordinated protocols that consider the MH's disk storage as unstable and thus inappropriate for storage of a participant's state.

[59, 66], assumes that MH has a relatively safe stable storage and thus can participate in the checkpointing process like static hosts.

[59] describes a time coordinated recovery protocol which adapts its processing by tuning the MH's stable storage participation in the checkpointing process based on the current network characteristics.

[66] treats all participants equally and provides special treatment only for MH's handoffs.

	[1]	**[139](a)**	**[139](b)**	**[126]**	**[140]**
Logging Type	Immediate	Immediate	Periodic	Periodic	Periodic
Ratio of Checkpoints	Only hard	Only hard	Only hard	Hard and soft	Only hard
Coordination Type	Uncoordinated	Uncoordinated	Uncoordinated	Coordinated	Coordinated
Coordination Method	n/a	n/a	n/a	Through timer	Message-based
Client's Storage	Unsafe	Unsafe	Unsafe	Safe	Safe
Adaptability	No	No	No	Vary the ratio of hard to soft checkpoints based on network QoS	No, mobile and fixed are peers

References

[1] A. Acharya and B. R. Badrinath. Checkpointing Distributed Applications on Mobile Computers. In *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems*, pages 73–80, Austin, Texas, September 1994.

[2] G. Alonso, G. Gunthor, M. Kanath, D. Agrawal, A. El. Abbadi, and C. Mohan. Exotica/FMDC: A Workflow Management System for Mobile and Disconnected Clients. *Distributed and Parallel Databases*, 4:27–45, 1996.

[3] B. R. Badrinath and T. Imielinski. Replication and Mobility. In *Proceedings of the 2nd IEEE Workshop on the Management of Replicated Data*, pages 9–12, November 1992.

[4] B. R. Badrinath and S. Phatak. Database Server Organization for Handling Mobile Clients. Technical Report DCS-342, Department of Computer Science, Rutgers University, 1997.

[5] Bandwidth Conservation Society. Bandwidth Conservation Society's Homepage. www.infohiway.com/faster/.

[6] D. Barbará. Certification Reports: Supporting Transactions in Wireless Systems. In *Proceedings of the IEEE International Conference on Distributed Computing Systems*, 1997.

[7] D. Barbará and H. Garcia-Molina. Replicated Data Management in Mobile Environments: Anything New Under the Sun? In *Proceedings of the IFIP Conference on Applications in Parallel and Distributed Computing*, April 1994.

[8] D. Barbarà, S. Mehrota, and M. Rusinkiewics. INCAs: Managing Dynamic Workflows in Distributed Environmnets. *Journal of Database Management*, 7(1):5–15, 1996.

[9] T. Berners-Lee, R. Caililiau, A. Luotonen, H.F. Nielsen, and A. Secret. The World-Wide Web. *Communications of the ACM*, 37(8):76–82, August 1994.

[10] L. Cardelli. A Language with Distributed Scope. *Computing Systems*, 8(1):27–59, 1995.

[11] L. Cardelli. Mobile Computation. In J. Vitek and C. Tschudin, editors, *Mobile Computation. In Mobile Object Systems - Towards the Programmable Internet*, pages 3–6. Spinger-Verlag, LNCS 122, 1997.

[12] K. M. Chandy and L. Lamport. Distributed Snaphots: Determining Global States of Distributed Systems. *ACM Transactions on Computer Systems*, 3(1):63–75, February 1985.

[13] P. K. Chrysanthis. Transaction Processing in Mobile Computing Environment. In *Proceedings of the IEEE Workshop on Advances in Parallel and Distributed Systems*, pages 77–83, Princeton, New Jersey, October 1993.

[14] S. B. Davidson, H. Garcia-Molina, and D. Skeen. Consistency in Partitioned Networks. *ACM Computing Surveys*, 17(3):341–370, September 1985.

[15] A. Demers, K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and B. Welch. The Bayou Architecture: Support for Data Sharing Among Mobile Users. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 2–7, Santa Cruz, CA, December 1994.

[16] M. Dunham, A. Helal, and S. Balakrishnan. A Mobile Transaction Model that Captures both the Data and Movement Behavior. *ACM/Baltzer Journal on Special Topics on Mobile Networks*, 1997. To appear.

[17] E. N. Elmozahy, W. Zwuenepoel, and W. Manetho. Transparent Rollback Recovery with Low Overhead, Limited Rollback and Fast Output Commit. *IEEE Transaction on Computers*, 41(5):526–531, May 1992.

[18] A. Fox and E. A. Brewer. Reducing WWW Latency and Bandwidth Requirements by Real-Time Distillation. In *Proceedings of the 5th International World Wide Web Conference*, Paris, France, May 1996.

[19] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. Adapting to Network and Client Variability via On-Demand Dynamic Distillation. In *Proceedings of the ASPLOS-VII*, Cambridge, MA, October 1996.

[20] S. Glassman. A Caching Relay for the World Wide Web. *Computer Networks and ISDN Systems*, 27(2), November 1994.

[21] J. Gosling, B. Joy, and G. Steele. *The Java Language Specification*. Addisson-Wesley, 1996.

[22] J. Gray, P. Helland, P. O' Neil, and D. Shasha. The Dangers of Replication and a Solution. In *Proceedings of the ACM SIGMOD Conference*, pages 173–182, Montreal, Canada, 1996.

[23] J. N. Gray. Notes on Data Base Operating Systems. In B. R. Graham and G. Seegmuller, editors, *Operating Systems - An Advanced Course*. Spinger-Verlag, LNCS 60, 1978.

[24] R. Gruber, F. Kaashoek, N. Liskov, and L. Shrira. Disconnected Operation in the Thor Object-Oriented Database System. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

[25] R. G. Guy, J. S. Heidemann, W. Mak, T. W. Jr. Page, G. J. Popek, and D. Rothmeier. Implementation of the Ficus Replicated File System. In *Proceedings of USENIX Conference*, pages 63–71, 1990.

[26] J.S. Heidemann, T. W. Page, R. G. Guy, and G. J. Popek. Primarily Disconnected Operation: Experience with Ficus. In *Proceedings of the 2nd Workshop on the Management of Replicated Data*, November 1992.

[27] P. Honeyman and L. B. Huston. Communication and Consistency in Mobile File Systems. *IEEE Personal Communications*, 2(6), December 1995.

[28] B C. Housel, G. Samaras, and D. B. Lindquist. WebExpress: A Client/Intercept Based System for Optimizing Web Browsing in a Wireless Environment. *ACM/Baltzer Mobile Networking and Applications (MONET)*, 1997. Special Issue on Mobile Networking on the Internet. To appear. Also, University of Cyprus, CS-TR 96-18, December 1996.

[29] Y. Huang, P. Sistla, and O. Wolfson. Data Replication for Mobile Computers. In *Proceedings of the 1994 SIGMOD Conference*, pages 13–24, May 1994.

[30] Y. Huang and O. Wolfson. Object Allocation in Distributed Databases and Mobile Computing. In *Proceedings of the 10th International Conference on Data Engineering*, pages 20–29, February 1994.

[31] L. Huston and P. Honeyman. Peephole Log Optimization. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

[32] L. B. Huston and P. Honeyman. Disconnected Operation for AFS. In *Proceedings USENIX Symposium on Mobile and Location-Independent Computing*, pages 1–10, Cambridge, Massachusetts, August 1993.

[33] L. B. Huston and P. Honeyman. Partially Connected Operation. *Computing Systems*, 4(8), Fall 1995.

[34] T. Imielinksi and B. R. Badrinath. Data Management for Mobile Computing. *SIGMOD Record*, 22(1):34–39, March 1993.

[35] R. Jain and N. Krishnakumar. Network Support for Personal Information Services for PCS Users. In *Proceedings of the IEEE Conference on Networks for Personal Communications*, March 1994.

[36] J. Jing, O. Bukhres, and A. Elmagarmid. Distributed Lock Management for Mobile Transactions. In *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems*, May 1995.

[37] D. B. Johnson and W. Zwuenepoel. Recovery in Distributed Systems using Optimistic Message Logging and Checkpointing. *Journal of Algorithms*, 11(3):462–491, September 1990.

[38] A. D. Joseph, A. F. deLespinasse, J. A. Tauber, D. K. Gifford, and M. F. Kaashoek. Rover: A Toolkit for Mobile Information Access. In *Proceedings of the 15th Symposium on Operating Systems Principles*, December 1995.

[39] A. D. Joseph, J. A. Tauber, and M. F. Kaashoek. Mobile Computing with the Rover Toolkit. *IEEE Transactions on Computers*, February 1997.

[40] T. Juang and S. Venkatesan. Crash Recovery with Little Overhead. In *Proceedings of the 11th IEEE International Conference on Distributed Computing Systems*, 1991.

[41] M. Frans Kaashoek, Tom Pinckney, and Joshua A. Tauber. Dynamic Documents: Mobile Wireless Access to the WWW. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

[42] J. J. Kistler and M. Satyanarayanan. Disconnected Operation in the Coda File System. *ACM Transactions on Computer Systems*, 10(1):213–225, February 1992.

[43] J. Klingemann, T. Tesch, and J. Wasch. Enabling Cooperation among Disconnected Mobile Users. In *Proceedings of the 2nd IFCIS International Conference on Cooperative Information Systems (CoopIS'97)*, June 1997.

[44] N. Krishnakumar and A. Bernstein. High Throughput Escrow Algorithms for Replicated Databases. In *Proceedings of the 18th VLDB Conference*, pages 175–186, August 1992.

[45] N. Krishnakumar and R. Jain. Mobility Support for Sales and Inventory Applications. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 571–594. Kluwer Academic Publishers, 1995.

[46] G. H. Kuenning. The Design of the Seer Predictive Caching System. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, December 1994.

[47] P. Kumar and M. Satyanarayanan. Log-based Directory Resolution in the Coda FIle System. In *Proceedings of the 2nd International Conference on Parallel and Distributed Information Systems*, San Diego, CA, January 1993.

[48] P. Kumar and M. Satyanarayanan. Flexible and Safe Resolution of File Conflicts. In *Proceedings of the USENIX Winter 1995 Conference*, New Orleans, LA, January 1995.

[49] B. W. Lampson. Atomic Transactions. In B. W. Lampson, editor, *Distributed Systems: Architecture and Implemementation*, pages 246–285. Spinger-Verlag, LNCS 105, 1981.

[50] M. Liljberg, T. Alanko, M. Kojo, H. Laamanen, and K. Raatikainen. Optimizing World-Wide Web for Weakly Connected Mobile Workstations: An Indirect Apprach. In *Proceedings of the SDNE*, Whistler, Canada, June 1995.

[51] G. Y. Liu, A. Marlevi, and G. Q. Maguire Jr. A Mobile Virtual-Distributed System Architecture for Supporting Wireless Mobile Computing and Communications. *Wireless Networks*, 2:77–86, 1996.

[52] Q. Lu and M. Satyanarayanan. Isolation-Only Transactions for Mobile Computing. *Operating Systems Review*, pages 81–87, April 1994.

[53] Q. Lu and M. Satyanarayanan. Improving Data Consistency in Mobile Computing Using Isolation-Only Transactions. In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems*, Orcas Island, Washington, May 1995.

[54] R. Malpani, J. Lorch, and D. Berger. Making World Wide Web Caching Servers Cooperate. In *Proceedings of the 4th International World Wide Web Conference*, Boston, MA, December 1995.

[55] L. Mummert and M. Satyanarayanan. Large Granularity Cache Coherence for Intermittent Connectivity. In *Proceedings of the 1994 Summer USENIX Conference*, Boston, MA, June 1994.

[56] L. B. Mummert, M. R. Ebling, and M. Satyanarayanan. Exploiting Weak Connectivity for Mobile File Access. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995.

[57] P. O' Neil. The Escrow Transactional Method. *ACM Transactions on Database Systems*, 11(4):405–430, 1986.

[58] N. Neves and W. K. Fuchs. Using Time to Improve the Performance of Coordinated Checkpointing. In *Proceedings of the IEEE International Computer Performance and Dependability Symposium*, pages 182–191, Urbana, Illinois, September 1996.

[59] N. Neves and W. K. Fuchs. Adaptive Recovery for Mobile Environments. *Communications of the ACM*, 40(1):69–74, January 1997.

[60] Oracle. Oracle Mobile Agents Technical Product Summary. www.oracle.com/products/networking/mobile_agents/html/, June 1997.

[61] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addisson-Wesley, 1994.

[62] E. Pitoura. A Replication Schema to Support Weak Connectivity in Mobile Information Systems. In *Proceedings of the 7th International Conference on Database and Expert Systems Applications (DEXA96)*, pages 510–520. LNCS 1134, Springer Verlag, September 1996.

[63] E. Pitoura and B. Bhargava. Maintaining Consistency of Data in Mobile Distributed Environments. In *Proceedings of the 15th IEEE International Conference on Distributed Computing Systems*, pages 404–413, May 1995.

[64] E. Pitoura, B. Bhargava, and O. Wolfson. Data Consistency in Intermittently Connected Distributed Systems. Technical Report DCS-96-10 (Revised 7/97), Department of Computer Science, University of Ioannina, 1997.

[65] D. K. Pradhan, P. P. Krishna, and N. H. Vaidya. Recovery in Mobile Wireless Environment: Design and Trade-off Analysis. In *Proceedings of the 26th IEEE International Symposium on Fault-Tolerance Computing*, pages 16–25, Sendai, Japan, June 1996.

[66] R. Prakash and M. Sihghal. Low-Cost Checkpointing and Failure Recovery in Mobile Computing Systems. *IEEE Transactions on Parallel and Distributed Systems*, 7(10), October 1996.

[67] V. N. Radmanabhan and J. C. Mogul. Improving HTTP Latency. *Computer Networks and ISDN Systems*, 28(1), December 1995.

[68] P. Reiher, J. Popek, M. Gunter, J. Salomone, and D. Ratner. Peer-to-Peer Reconiliation Based Replication for Mobile Computers. In *Proceedings of the European Conference on Object Oriented Programming 2nd Workshop on Mobility and Replication*, June 1996.

[69] G. Samaras, K. Britton, A. Citron, and C. Mohan. Two-Phase Commit Optimizations in a Commercial Distributed Environment. *Distributed and Parallel Databases*, 3(4):325–361, October 1995.

[70] G. Samaras and S. D. Nikolopoulos. Algorithmic Techniques Incorporating Heuristic Desicions in Commit Protocols. In *Proceedings of the 25th IEEE Euromicro Conference*, September 1995.

[71] D. Skeen. Nonblocking Commit Protocols. In *Proceedings of the ACM/SIGMOD International Conference on Management of Data*, pages 133–142, Ann Arbor. Michigan, 1981.

[72] D. Steere and M. Satyanarayanan. Using Dynamic Sets to Overcome High I/O Latencies During Search. In *Proceedings of the 5th IEEE HotOS Conference*, May 1995.

[73] R. E. Strom and S. Yemini. Optimistic Recovery in Distributed Systems. *ACM Transactions on Computer Systems*, 3(3), 1985.

[74] C. Tait, H. Lei, S. Acharya, and H. Chang. Intelligent File Hoarding for Mobile Computers. In *Proceedings of the 1st ACM International Conference on Mobile Cmputing and Networking (Mobicom'95)*, Berkeley, Ca, October 1995.

[75] C. D. Tait and D. Duchamp. Service Interface and Replica Management Algorithm for Mobile File System Clients. In *Proceedings of the First International Conference on Parallel and Distributed Information Systems*, pages 190–197, 1991.

[76] C. D. Tait and D. Duchamp. An Efficient Variable-Consistency Replicated File Service. In *Proceedings of the USENIX File Systems Workshop*, pages 111–126, May 1992.

[77] D. Terry, A. Demers, K. Petersen, M. Spreitzer, M. Theimer, and B. Welch. Session Guarantees for Weakly Consistent Replicated Data. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, pages 140–149, September 1994.

[78] D. B. Terry, M. M Theimer, K. Petersen, A. J. Demers, M. J Spreitzer, and C. H. Hauser. Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System. In *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, December 1995.

[79] G. Walborn and P. K. Chrysanthis. Supporting Semantics-Based Transaction Processing in Mobile Database Applications. In *Proceedings of the 14th Symposium on Reliable Distributed Systems*, September 1995.

[80] G. Walborn and P. K. Chrysanthis. PRO-MOTION: Support for Mobile Database Access. *Personal Technologies Journal*, 1(3), September 1997.

[81] Y. M. Wang and W. K. Fuchs. Optimistic Message Logging for Independent Checkpointing in Message-Passing Systems. In *Proceedings of the 11th IEEE Symposium on Reliable Distributed Systems*, pages 147–154, Houston, Texas, October 1992.

[82] T. Watson. Application Design for Wireless Computing. In *Proceedings of the First IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, California, December 1994.

[83] J. E. White. Mobile Agents. General Magic White Paper, www.genmagic.com/agents, 1996.

[84] L. H. Yeo and A. Zaslavsky. Submission of Transactions from Mobile Workstations in a Cooperative Multidatabase Processing Environment. In *Proceedings of the 14th International Conference on Distributed Computing Systems*, Poznan, Poland, June 1994.