Design and Analysis of Highly Available and Scalable Coherence Protocols for Distributed Shared Memory Systems Using Stochastic Modeling*

Oliver E. Theel† Brett D. Fleisch

UCR-CS-95-1

*This research was partially sponsored by NSF CCR-9209405 and the DEC External Research Program.

†also with the Department of Computer Science, University of Darmstadt, Germany

Abstract

Larger size networks require DSM coherence protocols which scale well. Fault-tolerance in terms of high availability is required for data access and for uninterrupted DSM service since large-scale environments have a greater number of potentially malfunctioning components. We present a new class of coherence protocols for DSM systems whose instances offer highly available access to shared data at low operation costs. The protocols proposed scale well; an increase in the number of client sites does not increase the operation costs after a certain threshold has been reached. The results presented in this paper give strong guidelines for the overall design of DSM systems which offer highly available, uninterrupted services.

Contents

1 Introduction

DSM systems have been extensively investigated during the past several years. These systems are increasingly being used by application programmers due to the attractive communication paradigm provided. As a natural consequence, even critical applications like monitoring systems for high-risk patients or power plants could be written based on this paradigm, leading to the need for a highly reliable and highly available DSM service. Although a large number of approaches has been developed (see surveys [14, 15]) only a few of these solutions have taken aspects of scalability and fault-tolerance into consideration [11, 21, 22, 8, 19]. Earlier work emphasized finding a solution to the DSM coherence problem with good performance for a relatively small network consisting of only a few sites. But as networks become larger, the need for DSM coherence protocols which scale well and provide highly available services becomes increasingly important. *Scalability* of a DSM coherence protocol is required for a larger network with an increasing number of users (and therefore sites) that are expected to make use of DSM. *Fault-tolerance*, in terms of high availability, is required for uninterrupted DSM service even in a large-scale environments with a greater number of potentially malfunctioning components.

The remainder of the paper is structured as follows. In the next section, we give basic terminology and overview related work in the area of fault-tolerant and scalable DSM coherence protocol research. In section 3, we present our underlying model of computation. The new approach itself is presented in section 4. There we also briefly discuss basic solutions to the classic DSM coherence problem, i.e. solutions with very limited properties of fault-tolerance and scalability, which nonetheless serve as the basis for our approach. Section 4 is followed by an analysis of key properties including costs, degree of availability, and behavior in a large-scale environment. The last section summarizes the paper and states our ongoing and future work.

2 Basic Terminology and Related Work

A DSM server guarantees a particular *coherence semantics* which defines the notion of correctness and therefore *what* is guaranteed by a DSM system. Coherence semantics used in the past include release consistency [13], weak consistency [1], processor consistency [6], sequential consistency [12], and strict consistency [2]. Informally speaking, release consistency is the least restrictive coherence semantics of the list given above, allowing a high degree of parallism and superior performance characteristics by exploiting operation or application program semantics. Conversely, strict consistency may restrict parallism and thereby sacrificing some degree of performance. Nevertheless, its major advantage is the generality it provides: no additional semantic knowledge concerning the application program is needed by a DSM system. This is a significant reason why sequential consistency is used for many DSM systems.

The coherence semantics defines the notion of correctness and therefore *what* is guaranteed by a DSM system. In contrast, the *coherence protocol* implements the coherence semantics; it provides *how* this notion of correctness is achieved. Examples for coherence protocols are write-update, write-broadcast, and write-invalidation protocols [15]. Furthermore, for a specific DSM system, the *coherence unit* must be defined, i.e. what abstract memory object is guaranteed to be consistent: possible objects are general application objects, segments, or single pages of a segment. For the most part, pages are often chosen, due to a variety of reasons, e.g. the identical nature, and small size, their correspondence to the memory

chunks supported by the processors memory management unit. See [14, 17] for a broader discussion on this topic. In the functional model used here, we assume a DSM system guaranteeing strict consistency for the individual pages of a segment; we define a new class of fault-tolerant coherence protocols herein.

Recoverability and consistency in reliable DSM systems has been studied [11, 21, 22, 8, 19] including studies of specific systems like *Recoverable Virtual Memory* (RVM) [20, 4], *Munin* [3], *ickp* [18], and *DiSOM* [16]. Many of these studies have examined reliability from a reliable application perspective whereas our experimental work examines reliability from a memory-oriented perspective. Our experimental reliability work for RELIABLE MIRAGE+ appears in [9, 10].

3 Model

In this section, we give definitions which we use throughout the paper. Additionally, we describe the functional and formal model of the DSM system under consideration.

3.1 Definitions

We assume a networking environment consisting of m sites $R_1, \ldots, R_m$ with independent failure rates. The single sites do not behave in a byzantinian manner, i.e. they either function according to their specification or not at all. The sites are arbitrarily connected by communication links which may also fail independently from each other and independently from the sites. This may result in network partitions with several independently operating subnets. In addition, we assume that within the network, a distributed shared memory service is available which functions in the manner described in the next section. The DSM system maintains one or more *segments* $S_1, \ldots, S_s$. A segment S_i consists of one or more *pages* $P_1, \ldots, Pp_i$, and finally all of those pages consists of a certain number of basic memory units, e.g. bytes or words. Whereas the number of pages per segment might vary, the size of the basic memory units per page is the same for all segments. Let $\mathbf{C} \subset \{R_1, \ldots, R_m\}$ be the set consisting of all potential client sites of the DSM system, with $\mid \mathbf{C} \mid = n < m$ giving the cardinality of this set. $\mathbf{S} \subseteq \{R_1, \ldots, R_m\}$ is the set containing the DSM server sites. Only sites included in $\mathbf{C}$ are allowed to submit requests to the DSM servers given by $\mathbf{S}$. Each DSM server is assumed to use the identical coherence protocol CP per page, thus, they all guarantee the same coherence semantics and use identical sized coherence units.

3.2 Functional Model of a DSM System

Besides operations for creating and destroying segments, the most important operations that a (segmented) DSM system must support are *read* and *write operations* to a particular page of a segment. These operations are carried out using the client/server paradigm: the client site which desires to read (write) a specific page currently not available in the local memory submits a *read request (write request)* to an arbitrary *DSM server* which is in charge of maintaining the segment the page is associated with. DSM servers[1] are designated sites of the network owning data structures and algorithms which enable them to carry out the requested operations. In particular, the DSM server maintains a request queue and keeps directory information for the page copies distributed in the network. For the sake of generality, every DSM server can potentially also be a client, eliminating the need for using the communication

[1] For the scope of the subsequent discussion, we assume a single, non-replicated DSM server. In case of a server failure, a new server is selected using the standard election algorithms for distributed systems.

links for submitting the request (e.g. $\mathbf{S} \subseteq \mathbf{C}$). In the ordinary case, clients and servers are different sites, thus the requests must be transmitted from the clients to the servers using the communication infrastructure. Once a client has submitted its request (i.e. it received an acknowledgment from the DSM server), it blocks and waits for the *result* delivered from the server. Analogously, results are either submitted to the client with or without the use of the communication links, depending on whether the client and the DSM server reside on the same site or on different ones. The result of a request typically consists of the requested page's data and a *mode*. In both cases, the requested operation is regarded as being *successful*, since the client received or achieved what was wanted. Due to the state of the network or the state of the DSM system itself, it might not always be possible to carry out the requested operation. For example, the site where the requested page data is stored may not be reachable. In such a case, an error message is returned and the request is deemed *unsuccessful*. It is, among others, the aim of this work to construct DSM systems where the vast majority of requests are carried out successfully for all points in time. This leads to highly available DSM.

Once a client received the results of a successful request, which includes the data of a page, it maps the page data into its memory and uses it only according to the granted mode. The page is regarded as being *cached* in a particular mode with respect to a particular operation. The mode determines how subsequent read and write operations operate. Figure 1 gives an overview of the various modes used by the functional model. A *mode* is regarded as being a two-dimensional tuple consisting of a *read attribute*

Mode	**Description**
(local read, local write)	the page data can be read and modified locally, i.e. without the need to submit read or write requests to one of the DSM servers
(local read, global write)	the page data can be read locally, but write operations have to be submitted to the DSM server in order to execute a global write operation
(global read, global write)	neither a read nor a write operation can be executed locally. This mode is implicitly assumed for all pages which are not cached at a client site
(global read, local write)	not used

Figure 1: *Modes associated with DSM pages at the client sites*

in the first dimension and a *write attribute* in the second dimension. The mode instances are stored at the client sites. The read (write) attribute defines how a read (write) operation originating at a particular client site must be handled: If the client wants to read (write) the data of a page cached in a mode containg a local read (local write) attribute, then this results in a *local read operation* (*local write operation*) carried out locally on the client site. If a particular page is not cached at all or cached in a mode including a global read (global write) attribute, then a read (write) request must be submitted to the DSM server, triggering a global read operation (global write operation) performed by the server, as described earlier in this section. Based on the functional model described above, we present in the next section a new DSM coherence protocol which is suited for a large-scale and failure-prone environment.

4 New Approach

The key concept for achieving fault-tolerance in terms of high availability is *redundancy*. The higher the degree of redundancy the more malfunctioning system components can be tolerated. Applying this concept to DSM systems means increasing the number of page copies cached at sites in the network.

Clearly, there is a price to be paid: the more copies stored, the more management overhead. Options for management include keeping the multiple copies either up-to-date, or preventing the client from reading copies when they have become out-of-date. Although these two approaches appear straightforward, it is not easy to decide which approach is best. The choice has a strong impact not only on the costs of subsequent operations but also on their availabilities. For instance, when updating *all* existing copies of a page as part of a write operation to a DSM page, then the number of up-to-date (or *actual*) copies of a page remains constant. Thus, the availabilities of subsequent operations (reads or writes) are not affected. The same holds true for the costs: since the number of copies and their storage locations has not changed, subsequent read and write operations may be carried out at the same costs. This is not the case when a write operation on a page alters the number of copies: deleting copies has a severe impact on the availability and the costs of subsequent operations. If, for example, a write operation decreased the number of copies from 5 to 1, and the only site caching the most recent copy subsequently fails, then the page data is not available any more. On the other hand, reducing the number of copies from 5 to 1 reduces the costs of the next write operation, since only one copy must be updated instead of five. This discussion shows that there is a strong correlation between operation costs and operation availabilities.

4.1 Design Goals

As suggested earlier, operation costs affect the operation availabilities and vice versa. It is important to understand factors that have an impact on those parameters. Clearly, the coherence protocol used by the DSM system strongly influences the operation costs. Two examples of this correlation are now examined: the Write-Broadcast and the Write-Invalidation coherence protocols.

4.1.1 Write-Broadcast Coherence Protocol

In the Write-Broadcast (WB) coherence protocol [15], once a site has cached a copy of a page as part of the results of a read or write request, this copy is never deleted. The cached copy has the mode (local read, global write). This has the following consequences: First, all future read operations are no-cost [2] local read operations. Furthermore, since a local read operation does not require any remote communication and cooperation with other sites, the probability of malfunctioning components successfully obstructing the execution is extremely low. This leads to a very high read operation availability. Second, since all existing copies of a page must be modified in the scope of a (global) write operation to a page, write operations are extremely expensive. If even a single site cannot be reached, then this write operation must be abandoned; consistency cannot be guaranteed since only a subset of the page copies is altered. The availability of the write operation is therefore affected by the maximum components involved. In the absence of perfect, i.e. failure-free, components, this causes an extremely low write operation availability.

Figure 2 shows the change of location and the change in the number of a single page's copies for a specific workload. The DSM server is assumed to guarantee consistency using the WB coherence protocol. The state of the five sites reflect the situation after a certain operation has been executed. The first situation is encountered after the execution of a read request issued by site R_2: only site R_2 caches a copy of the

[2] local reads are considered; no cost in this model

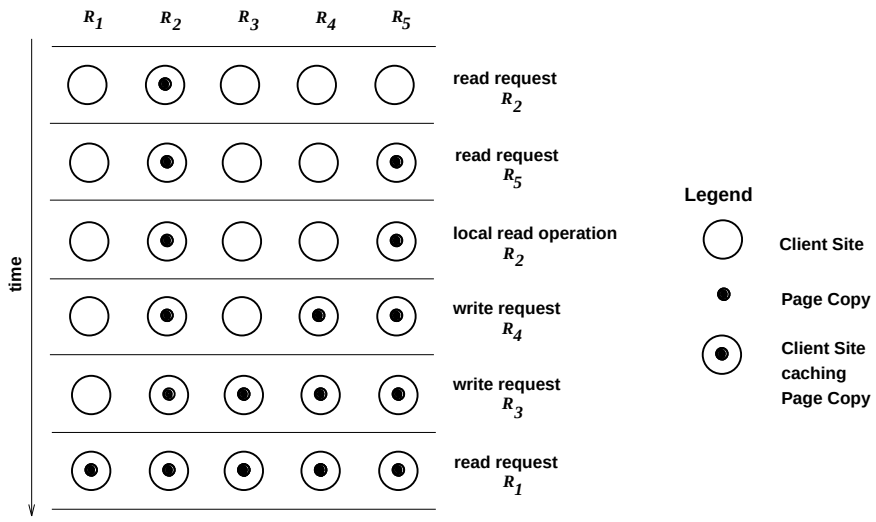

Figure 2: *Location and number of cached copies of a page in the Write-Broadcast coherence protocol*

page in (local read, global write)-mode. Subsequently, a read request by R_5 is executed, leading to an increase in the number of copies in the system. This is advantageous with respect to the availability of the page data. In a further step, R_2 performs a local read operation. The DSM system is not involved, since R_2 already caches a copy which is locally available. The next two requests are write requests issued by R_4 and R_3 respectively. The execution of the corresponding operations lead to an increase of one copy for each. This results in a state of the system where four copies are available, all of them in (local read, global write)-mode. Unfortunately, the execution of those write operations became increasingly expensive, since an increasing number of sites were involved. (Note, that the former copies must be updated every time a write operation is carried out.) Finally, site R_1 submits a read request to the DSM server. As the result, R_1 receives a copy in (local read, global write)-mode. The costs for executing this global read operation are, in contrast to executing a global write operation, independent of the number of copies currently present in the system; only one former copy has to be contacted in order to provide the new client site with a requested data. The execution of a global read operation has a positive impact on the data's availability and not on subsequent operation costs. Once every site caches a copy of the page, the system has reached a "stable state": regardless of the number and order of subsequent requests, the number and location of copies will not change (implying that copies of a page are not subject to preemption).

4.1.2 Write-Invalidate Coherence Protocol

A second widely used protocol is the *Write-Invalidate* (WI) coherence protocol [15]. This protocol guarantees strict consistency by freely increasing the number of copies granted to participating sites in (local read, global write)-mode. In case of a write operation to a page, all existing copies in (local read, global write)- or (local read, local write)-mode must be *invalidated*, i.e. destroyed. The remaining copy has (local read, local write)-mode. Thus, the number of copies of a page depends on the sequence of read and write requests submitted to the DSM server. There is no corresponding stable system state with respect to the number of copies as in the WB coherence protocol.

Figure 3 shows the distribution of page copies while processing the same workload as in the previous

example. This time, the DSM server uses the WI coherence protocol for guaranteeing consistency. The

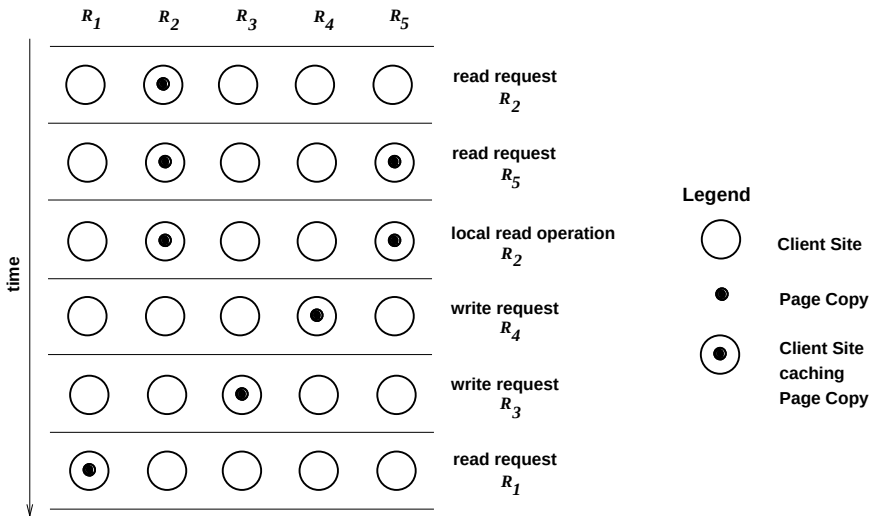

Figure 3: *Location and number of cached copies of a page in the Write-Invalidate coherence protocol*

scenario begins as in the former example: only site R_2 caches a copy of the page in (local read, global write)-mode. A read operation issued by R_5 increases the number of copies. As expected, the execution of a local read operation (here at R_2) has no impact on the number of cached copies. However, when write operations are executed, the differences between WB and WI coherence protocols in terms of the number of copies becomes apparent: write operations using the WI coherence protocol reduce the number of copies to one regardless of how many copies were present before the execution of the operation (see the write requests issued by R_4 and R_3). Although, the impact on the availability of the page is not good, there is a redeeming factor: executing a second and any further write operations in a row becomes extremely inexpensive, since only one former copy has to be destroyed. The costs for the first global write operation following an arbitrary number and sequence of global read operation is highly dependent on the number of cached copies present. After granting the read request of R_1, the system is again in a state where only one copy in (local read, global write)-mode[3] is present – a situation similar to the beginning of the scenario.

As a summary of observations from the discussion above, a suitable coherence protocol for a large, error-prone networking environment must have the following properties:

(P1) Limited Workload Dependability

the number of copies of a page should, only to a certain degree, be dependent on the workload, i.e. the mixture of read and write operations

(P2) Lower Bound in the Number of Cached Copies

situations where only a single copy of a page exists should be avoided. The availability of the page data is extremely vulnerable to even single component failures when there is only a single copy of the page

[3] an optimization is to grant a single reader a copy in (local read, local write)-mode

(P3) Upper Bound in the Number of Cached Copies

situations where all the client sites cache a copy of a page and all those pages must be updated as part of a global write operation must be avoided. This is required when there is a substantial number of global write operations inherent in the workload. Furthermore, situations where the number of caching sites increases must be avoided, since the probability that a write operation can successfully be executed decreases significantly

A coherence protocol which takes these observations into account is given in the following section.

4.2 New Coherence Protocol

A DSM server's workload can be divided into two different phases. The first phase starts with the execution of the first global read operation after a global write operation and ends with the next global write operation. The second phase starts with the first global write operation following a global read operation and continues until the execution of the next global read operation. (Note that the local read and local write operations cannot be monitored by a DSM server since they are not resulting in a request submitted to the server.) We call the former phase *read phase* since it consists entirely of read operations and the latter phase analogously *write phase*. In order to control the number of page copies present in the critical situations (P1) – (P3) stated above, we establish the following constraints. We refer to these constrains as *protocol invariants* since the coherence protocol must ensure them at every point in time (analogously to the strict consistency coherence semantics, which must also always be guaranteed).

Protocol Invariants

The coherence protocol guarantees that during a read phase, copies of a page are exclusively cached in (local read, global write)-mode, and that the number of cached copies N_r during this phase always satisfies the constraint:

$$r_{min} \leq N_r \leq r_{max} \qquad \text{with} \qquad r_{min}, N_r, r_{max} \in \mathbf{N} \tag{1}$$

with r_{min} (r_{max}) representing the minimum (maximum) number of cached copies in the read phase. $\mathbf{N}$ is the set of positive integers excluding zero. For the number of copies present and modified as part of the execution of a local or global write operation N_w in the write phase, the following always holds:

$$w_{min} \leq N_w \leq w_{max} \qquad \text{with} \qquad w_{min}, N_w, w_{max} \in \mathbf{N} \tag{2}$$

With these invariants, Figure 4 shows three cases which correspond to the coherence protocols discussed earlier. Part a) of the figure shows the boundary setting for the WB coherence protocol, part b) for the WI coherence protocol, and part c) shows the boundary setting for a protocol named *Write-Invalidation with Downgrading* (WIwD) [5]. The WIwD coherence protocol is similar to the WI protocol, except when a global read occurs and the requesting site gets an actual copy, the copy of the former writer's page is *downgraded* from mode (local read, local write) to (local read, global write). Since the page is not deleted, it increases the *minimum* number of cached copies in the read phase to two. Unfortunately, even this approach suffers from too few copies present in the write phase. In order to address this serious drawback, we have studied a design modification that increases the minimal and maximum number of

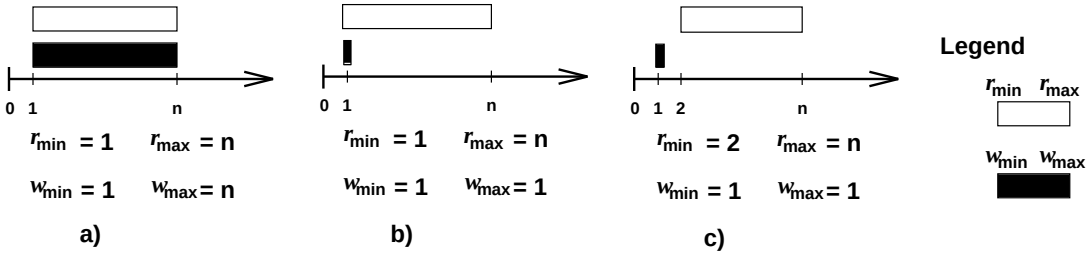

Figure 4: *Boundary Settings for three coherence protocols: a) Write-Broadcast, b) Write-Invalidate, and c) Write-Invalidate with Downgrading*

copies modified by a write operation (i.e. required in the write phase) together with the minimum number of copies cached in the read phase by $w-1$ with $w \in \{2, 3, \ldots\}$. This leads to boundary settings given in Figure 5. These constraints are enforced by the DSM server while serving a read or write

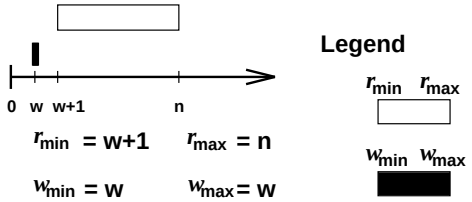

Figure 5: *Boundary Settings for the new coherence protocol*

request, and as part of the initial configuration when a segment is created. The values for r_{min}, r_{max}, w_{min}, and w_{max} can ideally be different for every page, leading to a different degree of availability for the pages' data and to different operation costs. Unless stated otherwise, we assume throughout the remainder of this paper that the boundaries are identical for all pages of a segment. Furthermore, we deal only with one segment. A generalization for different boundary values and multiple segments is nevertheless straightforward.

4.3 Protocol Operations

In this section, we describe the particular protocol operations.

Segment creation operation

Segment creation is performed at the DSM server after the client specifies the segment size in terms of pages, initial values for all the pages (eventually in an implicit manner, like initialize_with_zeroes), and values for the four boundaries r_{min}, r_{max}, w_{min}, and w_{max}. The segment creation operation can be regarded as a special write operation to all the pages of the new segment. Since page copies do not exist yet, the DSM server must create at least w_{min} copies of the page with the initial values on at least w_{min} client sites. Usually, a copy of every page is installed on the site where the segment creation request originated from. If the installation of the required number of copies succeeds, then the operation is successful and the DSM server informs the client. The mode of the installed page copies depends on the number of copies created: generally, the mode is (local read, global write) but in case there is only one copy to be installed, the (local read, local write)-mode is advantageous, since a single writer can locally modify the page data without violating the coherence semantics, thereby reducing write operation costs. If this is not possible, or if either w_{min} or r_{min} is greater then $\mid \mathbf{C} \mid$ (i.e. there are

less client sites than required for enforcing the lower boundaries of this particular coherence protocol), then the segment creation operation fails. In this case, the DSM server immediately notifies the client about the unsuccessful attempt.

Read operation

If a process at a client site wants to read a particular data item of a page, then the memory management unit checks whether this page is cached and therefore mapped into the local memory. Furthermore, the system verifies whether the local read attribute is part of the mode associated with the page. If both are true, then the read operation can be performed locally. Otherwise, a read request has to be submitted from the client site to an arbitrary DSM server of the segment where the page is stored. Once the read request is received at the DSM server and ready to be executed, the following steps are carried out by the server:

1. Since the DSM server is in a read phase, it checks whether granting a new copy to the client would increase the number of cached copies of this page beyond the threshold r_{max}. If this is true, then an arbitrary, already cached copy of this page has to be destroyed at a freely chosen site before granting a new copy to the requesting site, thereby preserving the maximum number of copies in read mode r_{max}.

2. Due to the fact that the assumed boundary $r_{min} = w_{max} + 1$ holds, there is no need – even at the beginning of a read phase – to install any more new copies except the copy at the site where the read request originated from. The number of cached copies in (local read, global write)-mode including the new one is – in this case – automatically equal to r_{min}. If, as in the more general case, $r_{min} = w_{max} + r + 1$ with $r \in \mathbf{N}$, then r *additional* copies in (local read, global write)-mode must be created on sites not already owing a copy of the particular page when entering a read phase in order to guarantee at least r_{min} copies while being in this phase.

Write operation

The client site first checks whether the write operation requested by a local process can be performed locally. In order to do so, a copy of the required page must have been cached and mapped into the site's memory. Furthermore, a local write attribute belonging to this page must be present. If one or both of those requirements does not hold, then a write request must be transmitted from the client site to a DSM server for the particular page. Once received, the DSM server performs the following steps:

1. If the DSM server recognizes that the execution of the write operation would increase the number of copies beyond the threshold w_{max}, then a copy at an arbitrary site must be destroyed. This guarantees the preservation of w_{max}.

2. While processing a write request issued by a client site currently not caching a copy of the page, the DSM server checks whether granting a copy to the requesting site would violate the upper boundary w_{max}. If $w_{min} = w_{max} = r_{min} - 1$, as assumed for our proposed boundary settings, then w_{max} is always exceeded. Hence, depending on the current number of client sites caching a copy, between 1 and $n - w_{max} + 1$ copies must be destroyed as part of the write operation. In

the more general case with $r_{min} > w_{max}$, between $r_{min} - w_{max} + 1$ and $n - w_{max} + 1$ copies must be removed.

3. Analogously, if the write request issued by a site already caching a copy, it must be verified that w_{max} is not already violated. In this case, up to $n - w_{max}$ copies must be deleted (in the general case between $r_{min} - w_{max}$ and $n - w_{max}$).

4. If the execution of the write operation at the DSM server leads to only one client site having a copy of a particular page, then the (local read, local write)-mode can be used for this copy. Although, if there is more than one cached copy, the (local read, global write)-mode must be used for all the copies.

At the end of the write operation it is guaranteed that only current page data is cached at the designated sites.

We call the coherence protocol with boundary settings as specified in Figure 5 *Boundary-Restricted* (BR) coherence protocol, since it restricts the number of cached copies to lie between w_{min} and r_{max}. Furthermore, because $w_{min} = w_{max}$ and $r_{min} = w_{max} + 1$, it suffices to instantiate w_{min} and r_{max}. For the rest of the paper, we abbreviate w_{min} with w and set r_{max} to n, the number of potential client sites. As a convention, we refer to an instance of the BR coherence protocol with given values for w and n as BR(w, n) coherence protocol.

Figure 6 gives an example that demonstrates how the BR(2,5) coherence protocol functions, under a sample workload. The set of client sites is given by $\mathbf{C} = \{R_1, \ldots, R_5\}$, thus all the sites can potentially cache a copy of the page. After the execution of a read request issued by R_2, this site together with

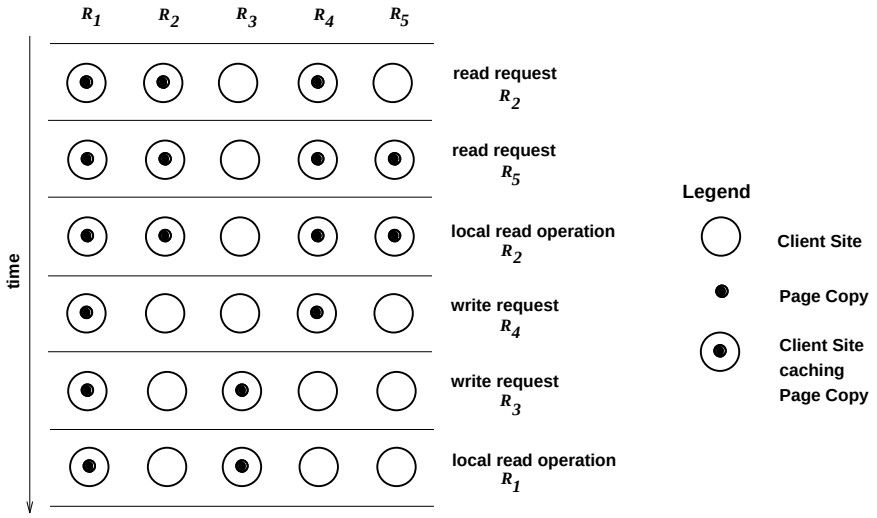

Figure 6: *Number and location of copies for the BR(2,5) coherence protocol*

R_1 and R_4 cache a copy of the page. (R_1 and R_4 were arbitrarily chosen). In a further step, site R_5 wants to read data of the page. Since it has no copy with a local read attribute cached yet, the execution of the corresponding global read operation performed by the DSM server increases the number of copies distributed in the network. Due to the fact that $\mid \mathbf{C} \mid = r_{max} = 5$, the upper boundary r_{max} is never in danger of being violated throughout this particular scenario. After R_2 locally reads its copy

of the page (the local read attribute is present), site R_4 submits a write request to the DSM server. Because $N_r = 4 > w_{max} = 2$ as defined by the coherence protocol, the DSM server must – prior to the completion of the write operation – destroy at least two cached copies. In this example, it chooses the copies located on sites R_2 and R_5.[4] Only R_1 and R_4 own a copy containing the most recent data, once the write operation has successfully been completed. In case of using the WB coherence protocol, all existing copies would have been updated at this point, whereas by using the WI or WIwD coherence protocols, all copies except the copy on site R_4 would have been invalidated and destroyed. Thus, the BR(2,5) coherence protocol mitigates the high costs for this write operation occurring while using the WB coherence protocol and mitigates the tremendous reduction in data availability as encountered by using the WI or WIwD coherence protocols. It does so by increasing costs against a gain of availability and decreasing availability against a saving in costs. A subsequent global write operation executed on behalf of R_3 leads to the destruction of the copy held by R_4. This was necessary, since the maximum number of copies present in a write phase would have exceeded w_{max} while executing the write operation. Finally, a subsequent read operation issued by site R_1 can be performed locally.

The example shows that the BR coherence protocol can be regarded as a kind of hybrid combination of the WB and WIwD coherence protocols. On one side, for $w = 1$ the BR coherence protocol is identical with the WIwD approach. On the other side, the protocol uses the WB approach on those sites which either want to modify the page or already cache a copy of the page and want to read it. It is therefore intuitively not surprising that the properties of the WB and WBwD coherence protocols strongly impact the properties of the resulting BR coherence protocol. The next section focuses on analyzing those properties in detail.

5 Analysis

The properties of DSM coherence protocols which are of primary interest in the current context are: the read and write *costs*, the *availability* of the user data managed by the DSM system (and therefore managed by the coherence protocol), and the impact of an increasing number of sites using a DSM system with a particular coherence protocol, in other words, the *scalability* of the coherence protocol. Throughout the rest of the paper, we adopt the following definitions.

Definition 5.1 (Costs) Let CP be the coherence protocol used by the DSM server of a DSM system. The number of sites $C_{R,\mathsf{CP}}$ ($C_{W,\mathsf{CP}}$) contacted by the DSM server using the communication infrastructure of the network in order to carry out a global read (write) operation are called *read (write) costs.* □

This definition of costs does not include the costs which arise for submitting the request to the DSM server. This additional consideration would lead to an increase of one copy in the case where the client and the server are located on different sites. If client and server reside on the same site then the additional costs would be zero. We chose the definition given above because they represent the part of the overall cost which is expected to be different when using different coherence protocols. The costs for submitting requests to the DSM servers occur regardless of the nature of the coherence protocol

[4]The decision which copies to delete in which situation can be based on various policies: analogously to load balancing algorithms they can be extremely simple or very sophisticated, e.g. by taking static or even dynamic network information into account. Since the particular policy used has no impact on the correctness of the coherence protocol presented here, we do not further explain them. We simply assume that a policy is in place.

chosen. The "submission costs" are only affected by the number of servers present in the networking environment.

Definition 5.2 (Data Availability) Let CP be the coherence protocol used by the DSM server of a DSM system. The probability that at an arbitrary point in time at least one functioning site caches an actual copy of a page is called *data availability (of this particular page by using* CP*).* □

In the following sections, we investigate the operation costs, the data availabilities, and discuss the scaling property of the Boundary-Restricted coherence protocol. Furthermore, we compare the BR coherence protocol with the previously mentioned WB, WI, and WIwD coherence protocols. Recall the WIwD coherence protocol is a special case of the BR coherence protocol, namely BR(1,n) protocol.

5.1 Stochastic Models

We base our protocol analyses on stochastic models, more precisely, on markov chains [7]. In the following, we give the markov chains for the three coherence protocols under investigation.

5.1.1 Stochastic Model of the Boundary-Restricted Coherence Protocol

The markov chain used for modeling the BR coherence protocol is given in Figure 7. The ovals present

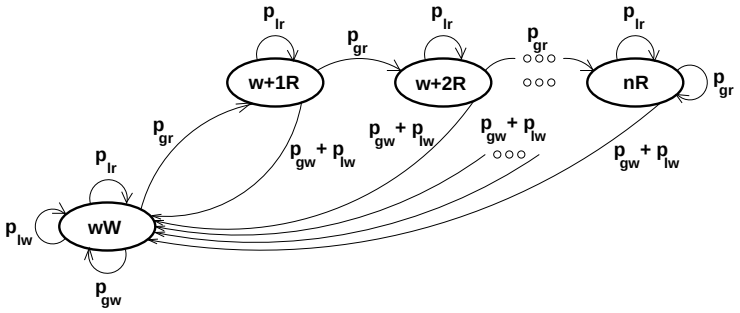

Figure 7: *Markov chain used for deriving the steady state distribution of the BR coherence protocol*

the states of the markov chain. The states ending with a "R" stand for those states one can encounter in a read phase, while those ending with a "W" represent states of a write phase. The preceeding figure (e.g. "$w+1$") indicates the number of copies cached on client sites. Thus, for example, the state labeled with "$w+1R$" represents the system state where $w+1$ copies are distributed to $w+1$ client sites while the DSM server is monitoring a read phase. "wW" represents the only state of the write phase. In this case, exactly w copies are cached at w different client sites. The arrows between the ovals represent the possible state transitions. The weight of an arrow indicates, the probability the system switches from the state the arrow originates at (tail of the arrow) to the state the arrow ends at (head of the arrow). For example, the probability that the system goes from state "wW" to state "$w+1R$" is given by p_{gr}. We refer to those weights as *transition probabilities.* Throughout the markov chain analyses, we use four different transition probabilities. They are given in Figure 8. After processing "a lot" of requests, for example while doing a parallel matrix calculation, the system reaches a *steady state.* For this steady state, the probability Π_i that an observer finds the DSM system in state i, can be computed [7]. This so-called *steady state distribution* $\vec{\Pi}_{\mathsf{CP}}$ of is of great assistantce in analyzing a particular DSM coherence protocol CP: once this distribution is known, operation costs and operation availabilities can be derived. The steady state distribution for the BR coherence protocol in terms of the four transition

Probability	Description
p_{lr}	Probability that a site already caching a page copy re-reads the data. It depends on the read attribute as to whether this can be done locally (local read attribute) or whether the site must issue a read request (global read attribute).
p_{gr}	Probability that a site currently not caching a copy of a page requests the data.
p_{lw}	Probability that a site already caching a copy of a page wants to modify it. In this case, if a local write attribute is present, this can be done locally. If the global write attribute is set, then the site must submit a write request to the DSM server.
p_{gw}	Probability that a site currently not caching a copy of a page wants to modify the page's data.

Figure 8: *Transition probabilities used for the markov chain analyses*

probabilities is given below. The steady state distribution is defined as

$$\vec{\Pi}_{\mathsf{BR}} := (\Pi_1, \Pi_2, \ldots, \Pi_{n-w+1}) \tag{3}$$

with Π_i as the probability of being in state "$w+iR$", $i = 1, \ldots, n-w$ and Π_{n-w+1} being the probability that the system is in state "wW". The state probabilities are

$$\Pi_i = \frac{1}{\mathcal{N}_{\mathsf{BR}}} \cdot \mathcal{P}^{i-1} \qquad \text{with } i = 1, \ldots, n-w-1 \tag{4}$$

$$\Pi_{n-w} = \frac{1}{\mathcal{N}_{\mathsf{BR}}} \cdot \mathcal{P}^{n-w-2} \cdot \frac{p_{gr}}{1 - p_{lr} - p_{gr}} \tag{5}$$

$$\Pi_{n-w+1} = \frac{1}{\mathcal{N}_{\mathsf{BR}}} \cdot \frac{1}{\mathcal{P}} \tag{6}$$

where

$$\mathcal{N}_{\mathsf{BR}} = \frac{1}{\mathcal{P}} + \mathcal{P}^{n-w-2} \cdot \frac{p_{gr}}{1 - p_{lr} - p_{gr}} + \frac{1 - \mathcal{P}^{n-w-1}}{1 - \mathcal{P}} \tag{7}$$

and

$$\mathcal{P} = \frac{p_{gr}}{1 - p_{lr}}. \tag{8}$$

Next, we give the markov chain model of the WI coherence protocol together with its steady state distribution.

5.1.2 Stochastic Model of the Write-Invalidate Coherence Protocol

Figure 9 gives the markov chain used for modeling the WI coherence protocol. Compared to the markov

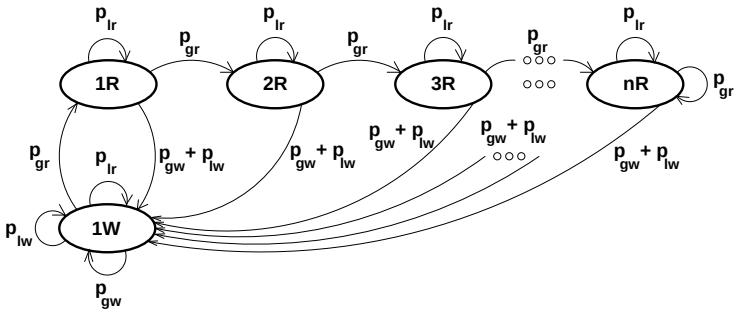

Figure 9: *Markov chain used for deriving the steady state distribution of the WI coherence protocol*

chain used for modeling the BR$(1,n)$ coherence protocol (identical to the WIwD coherence protocol), the markov chain for the WI coherence protocol only has an additional state "$1R$". The basic structure of both markov chains is identical. Therefore, the steady state distribution of the WI coherence protocol can be derived from the former one. Let

$$\vec{\Pi}_{\mathsf{WI}} := (\Pi_1, \Pi_2, \ldots, \Pi_{n-w+2}) \tag{9}$$

be the steady state distribution of the WI coherence protocol with Π_i the probability of being in state "iR", $i = 1, \ldots, n$ and Π_{n+1} as the systems' probability of being in state "$1W$". The state probabilities are

$$\Pi_i = \frac{1}{\mathcal{N}_{\mathsf{WI}}} \cdot \mathcal{P}^{i-1} \qquad \text{with } i = 1, \ldots, n-1 \tag{10}$$

$$\Pi_n = \frac{1}{\mathcal{N}_{\mathsf{WI}}} \cdot \mathcal{P}^{n-2} \cdot \frac{p_{gr}}{1 - p_{lr} - p_{gr}} \tag{11}$$

$$\Pi_{n+1} = \frac{1}{\mathcal{N}_{\mathsf{WI}}} \cdot \frac{1}{\mathcal{P}} \tag{12}$$

and

$$\mathcal{N}_{\mathsf{WI}} = \frac{1}{\mathcal{P}} + \mathcal{P}^{n-2} \cdot \frac{p_{gr}}{1 - p_{lr} - p_{gr}} + \frac{1 - \mathcal{P}^{n-1}}{1 - \mathcal{P}}. \tag{13}$$

5.1.3 Stochastic Model of the Write-Broadcast Coherence Protocol

Figure 10 shows the associated markov chain. For completeness, we give the markov chain of the

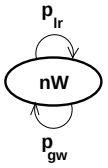

Figure 10: *Markov chain used for deriving the steady state distribution of the WB coherence protocol*

WB protocol consisting of all non-transient states, since we concentrate on steady state analysis. The resulting markov chain consists only of one state "nW" since by definition the DSM system is always in write phase (For switching from the write phase to the read phase, the DSM server had to process a read request. Due to the fact that in the WB coherence protocol all read operations are local, this can never happen.) As a consequence of having only one state, the steady state distribution $\vec{\Pi}_{\mathsf{WB}}$ is trivial:

$$\vec{\Pi}_{\mathsf{WB}} := (\Pi_1) \qquad \text{with} \qquad \Pi_1 = 1. \tag{14}$$

Once the steady state distributions are known, they can be used to obtain the average operation costs and the data availabilities, as shown in the next sections.

5.2 Costs

When coherence protocols are modeled in the proposed way, then only the state transitions raise costs in the terms given by Definition 5.1. This is because state transitions in the markov chain correspond to the execution of either a read or a write operation (and furthermore whether these operations are local

or global ones). By taking the current state of the system together with the next operation into account, it is straightforward to determine the costs for this particular operation and the resulting system state. If, for example, a DSM system using the BR coherence protocol given in Figure 7 is in state "$w + 2R$" and a client site currently not caching a copy of this page wants to perform a write operation, then the costs for this operation are $w + 3$ and the system state switches to "wW". The cost of $w + 3$ arise because all of the former $w + 2$ sites caching a copy must be contacted (three sites must destroy their copy and at $w - 1$ sites the copies must be updated) and the requesting site gets its own page copy.

Using these "static" costs together with the steady state distribution, it is possible to quantify the "dynamic" costs of the operations of a coherence protocol, i.e. the costs arising per operation when assuming a steady state of the system and a given workload in terms of read and write probabilities. Next, we define more precisely what we mean with "dynamic" costs:

Definition 5.3 (Cost matrices) Let $\Phi_1, \ldots, \Phi_N$ be the states of a markov chain modeling a coherence protocol CP. The matrix $\mathcal{C}^{lr}_{\mathsf{CP}} := \left(c^{lr}_{i,j}\right)$ with $c^{lr}_{i,j} \in \mathbf{R}^+$ and $i, j = 1, \ldots, N$ is called *local read cost matrix* of CP. $c^{lr}_{i,j}$ represents the read costs caused by a read operation issued at a site already caching a page copy when the system switches from state Φ_i to Φ_j. If such a transition does not exist, then the costs $c^{lr}_{i,j}$ are 0. $\mathcal{C}^{gr}_{\mathsf{CP}}$, $\mathcal{C}^{lw}_{\mathsf{CP}}$, and $\mathcal{C}^{gw}_{\mathsf{CP}}$ are defined analogously as the *global read cost matrix*, the *local write cost matrix*, and the *global write cost matrix* of CP respectively.

Note, that the prefixes "local" and "global" do not indicate whether the operations be definitely be executed loacally or globally, but whether a local copy is available ("local") or must be created ("global").

Definition 5.4 (Mean Operation Costs) Let CP be the coherence protocol used by the DSM server of a DSM system, and let $\vec{\Pi}_{\mathsf{CP}} = (\Pi_1, \ldots, \Pi_N)$ be the steady state distribution of a markov chain modeling the coherence protocol. $\mathcal{C}^{lr}_{\mathsf{CP}} := \left(c^{lr}_{i,j}\right)$ is the associated local read cost matrix. Then, the *mean local read operation costs* $\overline{E}^{lr}_{\mathsf{CP}}$ are given by

$$\overline{E}^{lr}_{\mathsf{CP}} := \sum_{i=1}^{N} \left(\Pi_i \cdot \sum_{j=1}^{N} c^{lr}_{i,j} \right). \tag{15}$$

$\overline{E}^{gr}_{\mathsf{CP}}$, $\overline{E}^{lw}_{\mathsf{CP}}$, and $\overline{E}^{gw}_{\mathsf{CP}}$ are defined analogously, being called the *mean global read operation costs*, *mean local* and *mean global write operation costs*. Let p_{lr}, p_{gr}, p_{lw}, and p_{gw} be the transition probabilities according to the definitions given in Figure 8. $\overline{E}^{read}_{\mathsf{CP}}$ with

$$\overline{E}^{read}_{\mathsf{CP}} := \frac{p_{lr}}{p_{lr} + p_{gr}} \cdot \overline{E}^{lr}_{\mathsf{CP}} + \frac{p_{gr}}{p_{lr} + p_{gr}} \cdot \overline{E}^{gr}_{\mathsf{CP}} \tag{16}$$

is called the *mean read operation costs* of CP. $\overline{E}^{write}_{\mathsf{CP}}$ with

$$\overline{E}^{write}_{\mathsf{CP}} := \frac{p_{lw}}{p_{lw} + p_{gw}} \cdot \overline{E}^{lw}_{\mathsf{CP}} + \frac{p_{gw}}{p_{lw} + p_{gw}} \cdot \overline{E}^{gw}_{\mathsf{CP}} \tag{17}$$

is called the *mean write operation costs* of CP. Finally, we call $\overline{E}^{op}_{\mathsf{CP}}$ the *mean operation costs* of CP with

$$\overline{E}^{op}_{\mathsf{CP}} := p_{lr} \cdot \overline{E}^{lr}_{\mathsf{CP}} + p_{gr} \cdot \overline{E}^{gr}_{\mathsf{CP}} + p_{lw} \cdot \overline{E}^{lw}_{\mathsf{CP}} + p_{gw} \cdot \overline{E}^{gw}_{\mathsf{CP}}. \tag{18}$$

Definition 5.5 (Mean Number of Cached Copies) Let CP be the coherence protocol used by the DSM server of a DSM system, and let $\Phi_1, \ldots, \Phi_N$ be the states of a markov chain modeling the coherence

protocol. Furthermore, let $\vec{\Pi}_{\mathsf{CP}}$ be the steady state distribution. n_i, $i = 1, \ldots, N$ are the numbers of copies cached on client sites when the system is in state Φ_i. Then, the *mean number of cached copies* is

$$\overline{E}_{\mathsf{CP}}^{nbr} := \sum_{i=1}^{N} n_i \cdot \Pi_i. \tag{19}$$

Next, we apply these metrics to the BR coherence protocol.

5.2.1 Costs of the Boundary-Restricted Coherence Protocol

The cost matrices of the BR coherence protocol are given in Figure 11. The dashes in the matrices

$$C_{\mathsf{BR}}^{lr} = \begin{pmatrix} 0 & - & - & \cdots & - & - & - \\ - & 0 & - & \cdots & - & - & - \\ - & - & 0 & \cdots & - & - & - \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ - & - & - & \cdots & 0 & - & - \\ - & - & - & \cdots & - & 0 & - \\ - & - & - & \cdots & - & - & 0 \end{pmatrix} \qquad C_{\mathsf{BR}}^{gr} = \begin{pmatrix} - & 2 & - & \cdots & - & - & - \\ - & - & 2 & \cdots & - & - & - \\ - & - & - & \cdots & - & - & - \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ - & - & - & \cdots & - & 2 & - \\ - & - & - & \cdots & - & 2 & - \\ w+1 & - & - & \cdots & - & - & - \end{pmatrix}$$

$$C_{\mathsf{BR}}^{lw} = \begin{pmatrix} - & - & - & \cdots & - & - & w+1 \\ - & - & - & \cdots & - & - & w+2 \\ - & - & - & \cdots & - & - & w+3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ - & - & - & \cdots & - & - & n-1 \\ - & - & - & \cdots & - & - & n \\ - & - & - & \cdots & - & - & w \end{pmatrix} \qquad C_{\mathsf{BR}}^{gw} = \begin{pmatrix} - & - & - & \cdots & - & - & w+2 \\ - & - & - & \cdots & - & - & w+3 \\ - & - & - & \cdots & - & - & w+4 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ - & - & - & \cdots & - & - & n \\ - & - & - & \cdots & - & - & n+1 \\ - & - & - & \cdots & - & - & w+1 \end{pmatrix}$$

Figure 11: *Cost matrices of the BR coherence protocol*

represent costs which cannot occur since there is no corresponding state transitions in the markov chain. The costs in those cases are by definition zero. The local read costs matrix C_{BR}^{lr} equals the zero matrix, since reading an already cached copy of a page using the BR coherence protocol is always possible without submitting a read request to the DSM server and without the need to contact other sites. The costs of a global read operation, on the contrary, require one other site to send a recent copy of the page to the requesting site. Thus, without counting the DSM server itself, two sites are involved in this operation. For example, if the system is in state "$w + 2R$" (line 2 in the corresponding matrix) and a global read operation occurs, then the system transits into state "$w + 3R$" (intersection with column 3), thereby causing costs of two. In case the system is in state "wW" (bottom line) and a global read operation is to be executed, then a site currently caching the page together with $w - 1$ other sites and the requesting site is contacted for performing the operation. All of those sites receive a copy of the page in (local read, global write)-mode. Thus, costs of $w + 1$ occur (column 1). For local and global write operations, the costs associated with either a local write or a global write operation depend on the number of currently cached copies of a page: when $N > w$ copies are cached and a local write operation occurs, then all sites owning a copy have to be contacted. w of these copies including the requesting site get the new data of the page and the remaining $N - w$ are told to destroy their out-dated copy. The costs are N since N sites were involved in this operation. A global write operation is slightly more expensive than a local one: since the requesting site does not own a copy of the page at the beginning of the operation but gets a copy granted as the operation proceeds; one additional site is contacted while executing the request. This leads to costs of $N + 1$ in the general scenario described above.

The matrices are used to compute the mean operation costs. Apparently, the mean local read costs are

zero, since the entire local read matrix consists only of zeros:

$$\overline{E}^{lr}_{\mathsf{BR}} = 0 \tag{20}$$

The mean global read operation costs are

$$\overline{E}^{gr}_{\mathsf{BR}} = 2 \cdot (1 - \Pi_{n-w+1}) + (w+1) \cdot \Pi_{n-w+1} = (w-1) \cdot \Pi_{n-w+1} + 2. \tag{21}$$

This result states that the mean global read operation costs are governed by the lower boundary w and the systems' probability of being in write phase. The costs are in particular independent of the upper boundary n, i.e. of the maximum number of cached copies, once n is beyond a certain threshold. According to equation (16), the mean read operation costs were calculated as

$$\overline{E}^{read}_{\mathsf{BR}} = \frac{p_{gr}}{p_{lr} + p_{gr}} \cdot \left[(w-1) \cdot \Pi_{n-w+1} + 2\right]. \tag{22}$$

The mean write operation costs are

$$\overline{E}^{write}_{\mathsf{BR}} = \frac{p_{lw}}{p_{lw} + p_{gw}} \cdot \left[\sum_{i=1}^{n-w} i \cdot \Pi_i + w\right] + \frac{p_{gw}}{p_{lw} + p_{gw}} \cdot \left[\sum_{i=1}^{n-w} i \cdot \Pi_i + (w+1)\right] \tag{23}$$

The mean operation costs can be given:

$$\overline{E}^{op}_{\mathsf{BR}} = p_{lr} \cdot 0 + p_{gr} \cdot \left[(w-1) \cdot \Pi_{n-w+1} + 2\right] + p_{lw} \cdot \left[\sum_{i=1}^{n-w} i \cdot \Pi_i + w\right] + p_{gw} \cdot \left[\sum_{i=1}^{n-w} i \cdot \Pi_i + (w+1)\right] \tag{24}$$

Finally, the mean number of cached copies[5] is derived according to equation (19):

$$\overline{E}^{nbr}_{\mathsf{BR}} = \sum_{i=1}^{n-w} (w+i) \cdot \Pi_i + w \cdot \Pi_{n-w+1} \tag{25}$$

Figure 12 shows a characteristic behavior of the mean read, mean write, and mean operation costs when the upper boundary n is fixed to a certain value (here 100) and the lower boundary w is varied. Additionally, the mean number of cached copies is given. Obviously, an increase of the lower boundary w results in an increase of the mean costs and of the mean number of copies cached. This corresponds to our expectation: an increase of the minimum number of copies cached must result in an increase of the mean number of cached copies. The mean write costs must increase, since for every write operation initiating a write phase, a higher number of sites must be contacted when w grows. The mean read costs grow because at the beginning of every read phase, an increasing number of copies is cached at the client sites. Finally, because the mean read and write costs increase with a larger w, the mean operation costs must also grow. A remarkable fact is, that the behavior of all those cost measures is almost linear. Except in those cases, where the difference between the lower and the upper boundary, i.e. $n - w$, is relatively small, the mean number of cached copies and the mean write costs behave non-linearly. (The same holds for the mean operation costs, since it is directly impacted by the mean write costs.) The mean read costs always behaves linearly. They are not affected by the size of $n - w$.

[5] We do not present $\overline{E}^{write}_{\mathsf{BR}}$, $\overline{E}^{op}_{\mathsf{BR}}$, and $\overline{E}^{nbr}_{\mathsf{BR}}$ in their closed forms in order to show how they are principally computed. However, they do have closed forms, since $\sum_{i=1}^{n-w} i \cdot \Pi_i$ is similar to the first derivative of the geometric sum.

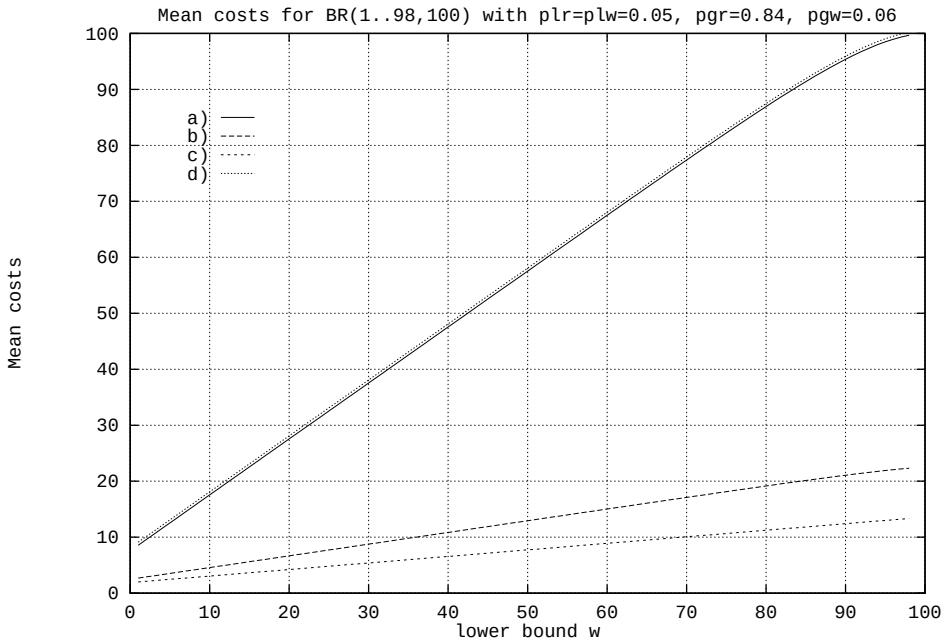

Figure 12: *Mean costs and mean number of cached copies of the BR(w,100) coherence protocol with w varying from $w = 1, \ldots, 98$. Graph a) represents the mean number of cached copies, b) the mean operation costs, and graphs c) and d) the mean read costs and the mean write costs respectively*

The reason for this behavior can quite easily be identified: the formulas for calculating the steady state distribution contain several geometric sums of the form $\sum_{i=0}^{n-w-2} \mathcal{P}^i$ and $\sum_{i=0}^{n-w-2} i \cdot \mathcal{P}^i$ with $\mathcal{P}$ given by equation (8). Since $\mathcal{P} < 1$, these sums have a known limit and reach it very rapidly when $n - w$ grows, or n grows and w is fixed. As a consequence, any further increase of these values does not have an impact on the steady state distribution; therefore, it has no impact (or only a very limited one) on the mean costs and mean number of copies. These properties have significant consequences with respect to availability and scalability issues.

As already mentioned, the mean costs and the mean number of cached copies behave in a linear manner, assuming the difference between n and w is sufficiently large. In the following, we present linear equations for those cost functions. These are approximations that are sufficiently precise, once $n-w$ is larger than 10. The linear approximations simplifies the analysis of the BR protocol family since linear equations in contrast to equations of higher-order polynomials can be investigated more easily. In particular, they ease the identification of major dependencies among the values involved, thereby simplifying e.g. the design of a BR coherence protocol with partially given characteristics. The approximations are given below.

$$\lim_{n\to\infty} \overline{E}^{read}_{\mathsf{BR}}(w) = \frac{p_{gr}}{p_{lr}+p_{gr}} \cdot \left[\underbrace{\frac{1-p_{lr}-p_{gr}}{1-p_{lr}}}_{a_{gr}} \cdot w + \underbrace{\frac{1-p_{lr}+p_{gr}}{1-p_{lr}}}_{b_{gr}} \right] \tag{26}$$

$$\lim_{n\to\infty} \overline{E}^{nbr}_{\mathsf{BR}}(w) = \underbrace{1}_{a_{nbr}} \cdot w + \underbrace{\frac{p_{gr}}{p_{lw}+p_{gw}}}_{b_{nbr}} \tag{27}$$

$$\lim_{n\to\infty} \overline{E}^{write}_{\mathsf{BR}}(w) = \underbrace{1}_{a_{write}} \cdot w + \underbrace{b_{nbr} + \frac{p_{gw}}{p_{lw}+p_{gw}}}_{b_{write}} \quad (28)$$

$$\lim_{n\to\infty} \overline{E}^{op}_{\mathsf{BR}}(w) = (p_{gr}\cdot a_{gr} + (p_{lw}+p_{gw})\cdot a_{write})\cdot w + (p_{gr}\cdot b_{gr} + (p_{lw}+p_{gw})\cdot b_{write}) \quad (29)$$

We performed analyses with different values for p_{lr}, p_{gr}, p_{lw}, p_{gw}, n, and w. They show that the graphs of the type given in Figure 12 are characteristic for the BR coherence protocol. Furthermore, it turned out that the higher the ratio of $(p_{lr}+p_{gr})/(p_{lw}+p_{gw})$ (refered to as *read/write ratio or rwr* hereafter), the more elliptical are the graphs for the mean write costs and the mean number of cached copies when w is almost as large as n. In other words, those are the cases where the approximations are more imprecise for small values of $n-w$. Additionally, the higher rwr, the higher is the offset of the mean write costs and the mean number of cached copies. That is because the system returns often to the "wW" state, causing heavy costs in this case. The mean read costs vary insignificantly with respect to a variation of the ratio of local to global operation, e.g. $(p_{lr}+p_{lw})/(p_{gr}+p_{gw})$ (referred to as *local/global ratio* hereafter). On the other side, their steepness is severely governed by rwr: the smaller rwr, the higher is the increase of the mean read costs when w approaches n.

5.2.2 Costs of the WI and the WB Coherence Protocols

For comparison, we give the corresponding values of the WI and the WB coherence protocols.

The mean read, mean write, and mean operation costs of the WI coherence protocol can be directly derived from equations (20) – (24) by substituting n with $n+1$ and by setting $w=1$. Accordingly, the markov chain of BR$(1,n+1)$ is isomorphic to the WI markov chain with n read states. Since the mean costs for even a relatively small value of $n-w$ become independent of any further increase of n once w remains constant, the mean costs of the WI coherence protocol are basically equal to those of the a BR$(1,n)$ coherence protocol. The mean number of cached copies of the WI coherence protocol can be easily derived using equation (19).

For the WB coherence protocol, it is straightforward to provide the corresponding mean costs due to the trivial steady state distribution. The mean read costs are always zero, once the DSM system reached a steady state:

$$\overline{E}^{read}_{\mathsf{WB}} = 0 \quad (30)$$

The mean write operation costs are equal to the mean number of cached copies:

$$\overline{E}^{write}_{\mathsf{WB}} = \overline{E}^{nbr}_{\mathsf{WB}} = n. \quad (31)$$

Finally, the mean operation costs are

$$\overline{E}^{op}_{\mathsf{WB}} = p_{lw}\cdot n. \quad (32)$$

Note, that for the WB coherence protocol, p_{gr} and p_{lw} are always zero, because all reads are local and all write operations are executed globally. Thus, in order to compare the mean operation costs of the WB protocol with those of the WI and BR protocols, p_{lr} (p_{gw}) used for the former should be equal to $p_{lr}+p_{gr}$ $(p_{lw}+p_{gw})$ used by the latter two protocols. We assume such an assignment from now on throughout the rest of the paper.

Although it is difficult to formulate a general rule due to the large number of variables involved, our investigations indicate the following relationships as rules-of-thumb:

$$\overline{E}^{write}_{\mathsf{WI}} \leq \overline{E}^{write}_{\mathsf{BR}} \leq \overline{E}^{write}_{\mathsf{WB}} \tag{33}$$

$$\overline{E}^{read}_{\mathsf{WB}} \leq \overline{E}^{read}_{\mathsf{WI}} \leq \overline{E}^{read}_{\mathsf{BR}} \tag{34}$$

$$\overline{E}^{op}_{\mathsf{WI}} \leq \overline{E}^{op}_{\mathsf{BR}} \leq \overline{E}^{op}_{\mathsf{WB}} \tag{35}$$

These relationships are stronger, the larger n and $n - w$ gets, and the more rwr decreases.

5.3 Availability

In this section, we compare the *data availability* provided by the protocols under investigation. The mean number of cached copies is used to determine with what probability the DSM system is able to deliver the most recent value of a particular page. As already pointed out, the higher the average number of copies present in the networking environment, the higher is the probability that such a copy can be accessed by the DSM server. In Figure 13 selected mean numbers of cached copies are given for all three coherence protocols discussed so far. The local/global ratio is 1/9. Obviously, the WB protocol caches

Mean Number of Cached Copies											
n	rwr	p_{lr}	p_{gr}	p_{lw}	p_{gw}	BR$(1,n)$	BR$(2,n)$	BR$(3,n)$	BR$(4,n)$	WI	WB
10	8	0.05	0.84	0.05	0.06	6.1	6.8	7.4	8.0	5.5	10
10	4	0.05	0.75	0.05	0.15	4.3	5.2	6.0	6.8	3.6	10
10	2	0.05	0.62	0.05	0.28	2.8	3.8	4.8	5.7	2.2	10
10	1	0.05	0.45	0.05	0.45	1.9	2.9	3.9	4.9	1.4	10
10	0.5	0.05	0.28	0.05	0.62	1.4	2.4	3.4	4.4	1.1	10
25	8	0.05	0.84	0.05	0.06	8.1	9.1	10.1	11.0	7.3	25
25	4	0.05	0.75	0.05	0.15	4.7	5.7	6.7	7.7	3.9	25
25	2	0.05	0.62	0.05	0.28	2.8	3.8	4.8	5.8	2.2	25
25	1	0.05	0.45	0.05	0.45	1.9	2.9	3.9	4.9	1.4	25
25	0.5	0.05	0.28	0.05	0.62	1.4	2.4	3.4	4.4	1.1	25
50	8	0.05	0.84	0.05	0.06	8.5	9.5	10.5	11.5	7.7	50
50	4	0.05	0.75	0.05	0.15	4.7	5.8	6.8	7.8	3.9	50
50	2	0.05	0.62	0.05	0.28	2.8	3.9	4.9	5.9	2.2	50
50	1	0.05	0.45	0.05	0.45	1.9	2.9	3.9	4.9	1.4	50
50	0.5	0.05	0.28	0.05	0.62	1.4	2.4	3.4	4.4	1.1	50

Figure 13: *Mean number of cached copies in the WI, the WB, and the BR coherence protocol*

the maximum number of copies in all those cases. This is generally true, but unfortunately, the WB coherence protocol achieves this advantageous behavior at a high price: the mean operation costs for any reasonable value of p_{lw} are extremely high. The a price for achieving this degree of data availability is not only too high but also not necessary, as we will show later in this section. As expected, the mean number of cached copies of the BR(w,n) protocol is essentially equal to the corresponding value of the BR$(w+1,n)$ protocol plus one for $w = 1,2,3$. Only for small values of n is the difference between BR(w,n) and BR$(w+1,n)$ is not exactly one. With the same constraints stated for equations (33) – (35) the following can be said about the mean number of cached copies:

$$\overline{E}^{nbr}_{\mathsf{WI}} \leq \overline{E}^{nbr}_{\mathsf{BR}} \leq \overline{E}^{nbr}_{\mathsf{WB}} \tag{36}$$

In order to derive how many cached copies are actually necessary to achieve a certain degree of data availability in a particular network with failure-bound components, we present Figure 14. We assume

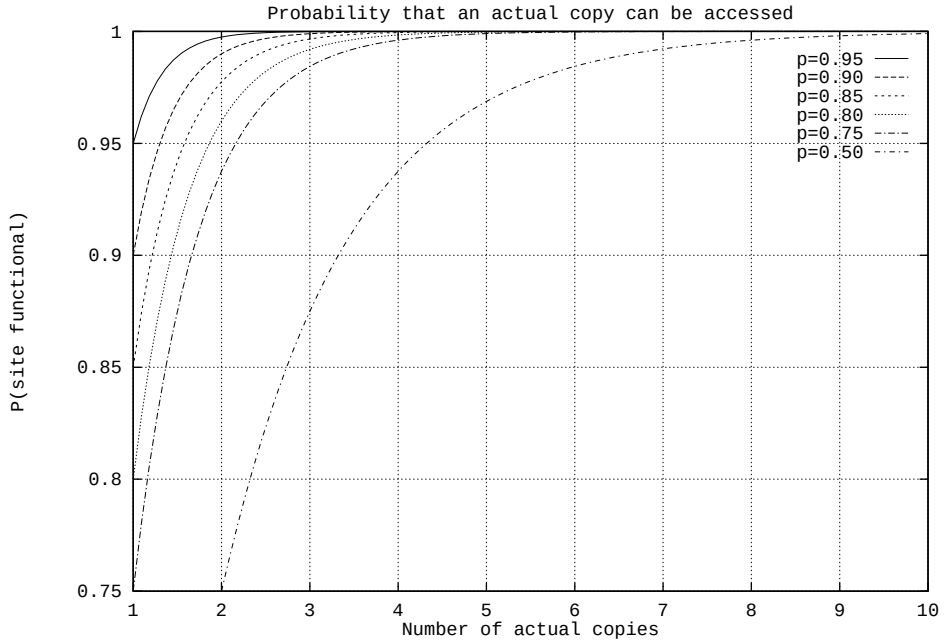

Figure 14: *Data availability depending on the number of cached copies and the probability that sites are functioning*

that all sites of the network – or at least those caching copies – are functioning with a uniform probability of p. This does not only imply that the individual sites are up and running but also that the software needed to manage the copy is in working order. Since a page data is available in the network as long as at least one of $a > 0$ copies is accessible, the data availability is computed according to a binomial distribution as $A_{data}(p) = 1 - (1 - p)^a$. Figure 14 shows the data availability graphs for six different values of p. From the graph of $A_{data}(0.75)$ for example, it can be derived that with only one actual copy present in the system, the data availability is, of course, 0.75. By increasing the number of copies to two, a data availability gain of 18 % is achieved. If three copies are guaranteed, then the data availability increases to approx. 98 %, and once more than three copies are cached, the data availability is basically 100 %. The major result of this analysis is, that for realistic values of $p > 0.75$, four cached copies are sufficient to provide an almost perfect degree of page data availability. Thus, those coherence protocols which offer a mean number of cached copies of at least four at the lowest cost possible are of particular interest. Using the approximation of the BR coherence protocol given in section 5.2.1, those protocols can easily be identified.

6 Conclusion and Future Work

We presented a new coherence protocol family for DSM systems, whose instances offer highly available access to the shared data at low operation costs. Results show that the protocols proposed scale well in the sense that an increasing number of client sites do not increase the operation costs beyond a certain boundary. The analysis with respect to data availability suggests that it is not sensible to maintain more than four copies in the network. This result must be interpreted carefully: the possibility of successfully

accessing a page with the most recent page data does not necessarily allow the DSM system to continue proper operation. Any read or write operations executed at the DSM server on behalf of a client may require the cooperation of additional sites. Those sites (and their software) may fail as well, leading to a lower availability of the operation. This is the case with all the coherence protocols discussed in this paper. Specifically, the WB protocol requires *all* sites caching a copy to be functional in case of a write operation. Thus, the availability of the write operation is p^a if a client sites are present. This value is much smaller than $1-(1-p)^a$ for the data availability. Unfortunately, applications are mainly interested in continuing operation in case of failures, i.e. issuing read and write operations. Consequently, those two operations, in addition, must be offered highly available in addition to the page data. Nevertheless, the availability analysis presented in this paper gives an upper bound of what can be achieved for read and write operations. The results provide, as we see it, valuable guidance for designing DSM systems with good fault-tolerance properties.

Subsequent work will include a more extensive analysis of the scaling properties of the proposed protocols and a derivation of the read and write operation availabilities. In a subsequent work, we also plan to investigate the adaptive behavior of the protocols, i.e. the dynamic adaption of the w boundary to a varying workload monitored by the DSM server.

References

[1] M. Dubois and C. Scheurich. Memory access dependencies in shared-memory multiprocessors. *IEEE Transactions on Computers*, 16(6):660–673, June 1990.

[2] M. Dubois, C. Scheurich, and F.A. Briggs. Synchronization, coherence, and event ordering in multiprocessors. *IEEE Computer*, 21(2):9–21, February 1988.

[3] Elmootazbellah Nabil Ehnozahy, David B. Johnson, and Willy Zwaenepoel. The performance of consistent checkpointing. In *Proceedings of the* 11^{th} *Symposium on Reliable Distributed Systems*, pages 39–47, Houston, Texas, USA, 1992. IEEE-CS, IEEE-CS Press.

[4] Michael J. Feeley, Jeffrey S. Chase, Vivek R. Narasayya, and Henry M. Levy. Integrating coherency and recoverability in distributed systems. In *Proceedings of the First Symposium on Operating Systems Design and Implementation (OSDI'94)*, Monterey, CA, USA, November 1994. USENIX, ACM SIGOPS, USENIX.

[5] Brett D. Fleisch and Gerald J. Popek. Mirage: A coherent distributed shared memory design. In Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, published in *Operating Systems Review* 23(5) Special Issue, pages 211–223, The Wigwam, Litchfield Park, Arizona, USA, December 1989. ACM SIGOPS, ACM Press.

[6] J.R. Goodman. Cache consistency and sequential consistency. Technical report 61, SCI Committee, March 1989.

[7] D. L. Isaacson and R. W. Madson. *Markov Chains: Theory and Applications.* Wiley series in probability and mathematical statistics. John Wiley & Sons, 1976. ISBN 0-471-42862-0.

[8] B. Janssens and W. K. Fuchs. Relaxing consistency in recoverable distributed shared memory. In *Proceedings of the Twenty-Third Annual International Symposium on Fault-Tolerant Computing: Digest of Papers*, pages 155–163, June 1994.

[9] N. C. Juul and B. D. Fleisch. A Memory Approach to Consistent, Reliable Distributed Shared Memory. In *Proceedings of the 5th Symposium on Hot Topics in Operation Systems*, May 1995. To be published.

[10] Niels Christian Juul, Brett D. Fleisch, and Cheryl DeMattis. Reliable Distributed Shared Memory. Mirage Research Note 7, Department of Computer Science, University of California, Riverside, CA, December 1994. Unpublished research note on the design of RELIABLE MIRAGE+ with multiple consistency levels.

[11] Richard Koo and Sam Toueg. Checkpointing and roolback-recovery for distributed systems. *IEEE Transactions on Software Engineering*, SE-13(1):23–31, January 1987.

[12] Leslie Lamport. How to make a multiprocessor computer that correctly executes multiprocess programs. *IEEE Transactions on Computers*, 28(9):690–691, September 1979.

[13] D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy. The directory-based cache coherence protocol for the DASH multiprocessor. In *Proceedings of the 17th Annual International Symposium on Computer Architecture*, pages 148–159, May 1990.

[14] Virginia Lo. Operating systems implementations of distributed shared memory. *Advances in Computers*, 39, 1994.

[15] Ajay Mohindra and Umakishore Ramachandran. A survey of distrbuted shared memory in loosely-coupled systems. Technical Report GIT-CC-91/01, Georgia Institute of Technology, Atlanta, GA, USA, January 1991.

[16] N. Neves, M. Castro, and P. Guedes. A checkpoint protocol for an entry consistent shared memory system. In *Proceedings of the 13th ACM Symposium on Principles of Distributed Computing (PODC'94)*. ACM, ACM Press, August 1994.

[17] B. Nitzberg and V. Lo. Distributed shared memory: A survey of issues and algorithms. *IEEE Computer*, 24(8):52–60, August 1991.

[18] James S. Plank and Kai Li. ickp: A consistent checkpointer for multicomputers. *IEEE Parallel & Distributed Technology*, 2(2):62–67, Summer 1994.

[19] Golden G. Richard, III and Mukesh Singhal. Using logging and asynchronous checkpointing to implement recoverable distributed shared memory. In *Proceedings of the* 12^{th} *Symposium on Reliable Distributed Systems*, pages 86–95, Princeton, New Jersey, USA, October 1993. IEEE-CS, IEEE-CS Press.

[20] M. Satyanarayanan, Henry H. Mashburn, Puneet Kumar, David C. Steere, and James J. Kistler. Lightweight recoverable virtual memory. *ACM Transactions on Computer Systems*, 12(1):33–57, February 1994.

[21] M. Stumm and S. Zhou. Fault tolerant distributed shared memory algorithms. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing*, pages 719–724. IEEE-CS, IEEE Press, December 1990.

[22] Kun-Lung Wu and W. Kent Fuchs. Recoverable Distributed Shared Virtual Memory. *IEEE Transactions on Computers*, 39(4):460–469, April 1990.