Interoperability is a key goal of the InfiniBand Trade Association. InfiniBand Architecture (IBA) subnet management is as interoperable as is technically feasible. Any IBA-compliant subnet manager is able to manage any IBA-compliant subnet, and support for migrating between different managers is provided. However, interoperability in the sense of failing over, on error, to a different vendor’s subnet manager is not supported. This is not technically feasible for reasons of architectural scope, packet routing, and management data integrity. This white paper explains the technical basis of this position.

IBTA Management Workgroup
Author: Gregory F. Pfister, Co-Chair
1 Introduction

Interoperability is a key goal of the InfiniBand Trade Association.

InfiniBand Architecture (IBA) subsystems were conceived from the outset as input/output and inter-host communication facilities that are simultaneously usable by multiple vendors’ products. The effort expended on this attests to this goal’s importance to the member companies. For example, the InfiniBand Trade Association (IBTA) constructed hundreds of detailed compliance statements, created very detailed test specifications, and sponsored numerous plugfests. In fact, the very first plugfest was well-attended even though it occurred prior to any vendor announcing the availability of any IBA product. That is a strong testament to the importance of interoperability for IBA product vendors.

Interoperability is no less an emphasis for management than for other parts of the InfiniBand Architecture. Management has been and remains a significant part of the activities mentioned above. By intent, design, and significant effort on the part of IBTA member companies, InfiniBand management is as interoperable as technically feasible. In particular, any IBA-compliant manager, from any vendor, is able to manage any IBA-compliant components it is designed to manage, no matter what vendor is the source of those components.\(^1\)

However, the InfiniBand Architecture Specification Volume 1, in all releases through version 1.1, states the following in Chapter 13 (“Management Model”):

“… compatibility and interoperability among SMs [Subnet Managers] from different vendors is not supported. Migration from one vendor’s SM to another’s by way of system reinitialization, i.e., through a planned outage, is supported.”

Additionally, the last sentence implies by omission that there is no support for migration from one vendor’s SM to another’s as result of an unplanned outage. That is, there is no support for one vendor’s SM taking over the network when another fails, an operation commonly referred to as failover.

Against the backdrop of strong emphasis on interoperability, this position stands out strongly. That it is deliberately documented seems particularly strange; limits like this are usually discovered in practice, becoming lore rather than specification. Lack of interoperability is clearly undesirable, and the implicitly rejected failover between vendors’ products seems to eliminate an especially valuable option, since errors that cause an outage by one product may well not be present in an independently-developed product. (This is actually a design principle deliberately used in the triply-redundant guidance computers used on the Space Shuttle. Each of the three is from a different vendor, and runs independently-developed programming.)

Did the IBTA Management Workgroup, the group chartered with defining IBA management facilities, simply give up on a hard problem? Will this limitation be removed in some future version of the specification?

The answers to those questions are “no,” and “perhaps, but probably not in a manner one would expect.”

In fact, that statement in the specification is the result of a significant amount of analysis performed by the IBTA Management Workgroup. This analysis is not included in the specification, since by IBTA policy the specification is primarily normative rather than informative. That is appropriate; the “informative” reasons why something was done a particular way may elicit

---

1. Should a situation occur in which this isn’t true, please tell us about it so we can correct it.
very different answers from different member companies, and in any event are not totally relevant to designing products that adhere to the standard.²

Some explanation is required in this case, however, since a perhaps less than obvious position was deliberately taken in an area that isn’t as generally well understood as most other areas involved in the standard. The purpose of this white paper is to explain why this was done.

Here is a short summary of that explanation:

- Routing in subnets is vendor-specific in IBA. So if a “foreign” subnet manager takes over, will in general not understand the routing; as a result, performing normal routing operations will be difficult, inefficient, or both.
- Even if routing was standardized (and there are good reasons not to do that), failing over correctly in error situations requires transactional semantics that would be difficult to require as part of the standard.
- Even if those semantics were brought into the standard, the intrinsic limitations on interoperability owing to the necessary boundaries of the IBTA itself imply that full interoperability lies outside of the IBTA.

Nevertheless, further interoperability may conceivably be investigated by the IBTA, involving switching SM vendors (or an SM vendor’s releases) without downtime. Exactly what this means, and what is entailed, requires the more detailed explanation which follows.

The more detailed explanation begins with the third item above: What, exactly, is defined by the IBA standard in the management area?

### 2 What Is Standardized?

As illustrated in Figure 1, subnet managers have a wide variety of interfaces to many different kinds of entities. Not all of those interfaces should be standardized, and of the ones that should be standard, not all of them should be standardized by the IBTA. For example:

- GUIs and stored data formats are not be standardized by the IBTA. These are regarded as value an SM vendor can add to its implementation, affecting both usability and efficiency at various tasks.
- The API area is being addressed by the Interconnect Software Consortium, under the aegis of the Open Group (see http://www.opengroup.org/icsc).

2. And besides, to extend Otto von Bismark’s observation: One who knows where laws, sausages, and standards come from can no longer sleep calmly at night.
Levels of Interoperability

- data formats and descriptive formats interfacing with other management standards are now being pursued in their respective standards organizations: DMTF (CIM), and IETF (SNMP MIBs). See http://dmtf.org and http://ietf.org, respectively, for more information.

It is appropriate to leave the standardization of those areas to other organizations. Of course, IBTA member companies can, and some do, participate in those other standardization efforts.

On-the-wire messages, formats, and protocols within an IBA fabric are, on the other hand, appropriately part of the IBA standard and are the exclusive subject of the management portions of IBA. Nothing in the IBA specification is said about the implementations generating and receiving those messages or about how, and whether, those implementations make use of other interfaces.

So the interoperability specifiable by IBTA is limited to on-the-wire message formats and protocols. This means that if failover should occur, the customer may well be, on an error, presented with GUIs and other interfaces not matching those that have been deployed. This alone reduces the utility of fault-induced failover.

Within the bounds of message formats and protocols, however, the goal should be to provide as much interoperability as possible. How much, exactly, is that? To answer that question, it was first necessary to examine exactly what “interoperability” means in this context.

3 Levels of Interoperability

When the Management Workgroup examined the meaning of “interoperability of subnet managers” in detail, a spectrum of possibilities was discovered. These possibilities were tabulated by the workgroup as shown in Table 1 on page 5.

Levels 1, 1.5, and 2 document a straightforward increase in the interoperability between SMs and non-SM elements of IBA, both hardware and host software. Anything less than level 2 was considered insufficient for the goals of the IBTA: The specification should be such that any SM is able to manage any subnet with components from any vendor, including differing operating systems on attached hosts.

From level 3 up, the table shifts to describing a different type of interoperability: Ways that one vendor’s SM can take over operation from another’s. In all those cases, some explanation is needed about the multiple SMs shown in the illustrations.

In no case are the different vendors’ SMs actively managing the same subnet at the same time. This was not even considered; one of the base assumptions of IBA management architecture is that there is exactly one Master SM on a subnet (only the master is shown in the illustrations). There is an algorithm that is part of the architecture (Volume 1, Section 14.4.1) which is used to determine which of several operational SMs becomes that master. While there may and often will be standby SMs, at any given time there is exactly one master.

Given this single-master-SM situation, the focus of interoperability between different vendor’s SMs becomes whether one vendor’s SM can be a standby if the master is another vendor’s—i.e., whether one vendor’s SM can take over from another vendor’s on failure, which is what being a “standby SM” means. This is only achieved in the highest level of interoperability shown: level 5; only in that case are the master and standby SMs’ vendors different.

The other two cases, levels 3 and 4, are compromises that do not provide unplanned failover between vendors; instead, they provide a switchover planned in advance. At level 3, the switch of SM vendors is accomplished off-line: The entire subnet is deactivated, and then brought up again with a different SM in control, with all the standbys and the master from the same ven-
At level 4, a similar switch occurs, but without bringing down the subnet; after the switch has occurred, all the master and standby SMs are homogeneous (from the same vendor).

The decision was made to support only level 3 interoperability: Offline migration.

This is clearly the least desirable of the three possibilities, and was picked only after significant discussion revealed that, in fact, there was little if any choice. There are two reasons behind this decision. One involves routing, and the other involves maintaining consistent internal SM data. The sections following explain each of these.

---

**TABLE 1. Levels of Interoperability**

<table>
<thead>
<tr>
<th>Type</th>
<th>Level</th>
<th>Interoperability</th>
<th>Description</th>
<th>Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM with other types of elements</td>
<td>1</td>
<td>None</td>
<td>An SM from a given vendor can only manage IBA hardware from that vendor.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.5a</td>
<td>Heterogeneous Hardware</td>
<td>An SM from any vendor can manage a heterogeneous collection of IBA-compliant hardware from any vendors the OSs on nodes may be coupled to a particular SM implementation; inter-vendor SM compatibility not provided.</td>
<td></td>
</tr>
<tr>
<td>SM taking over from another SM</td>
<td>2</td>
<td>Heterogeneous Nodes</td>
<td>An SM from any vendor can manage a heterogeneous collection of IB-compliant nodes, including their operating systems, from any vendors; inter-vendor SM compatibility not provided.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Offline Migration</td>
<td>Level 2, plus the ability to migrate from one vendor’s SM to another’s with system initialization, i.e., a planned outage. (Involves prior step of transferring data between vendors’ SMs.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Planned Takeover (Online Migration)</td>
<td>Level 2, possibly plus Level 3, with in addition the ability to do planned takeover from one vendor’s SM to another’s without a system outage.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Online Failover</td>
<td>Level 2, possibly plus Levels 3 &amp; 4, with in addition the ability to do unplanned failover from one vendor’s SM to another’s without a system outage.</td>
<td></td>
</tr>
</tbody>
</table>

a. The use of “1.5” as a level number is a historical artifact of the Management Workgroup discussions. It is retained here to enhance our institutional memory.
4 Routing

Early in the process of developing the IBA specification, it was decided that the techniques used to define data routing within a subnet would be vendor-specific. No one routing technique was deemed adequate because IBA was conceived as spanning a very broad range, from single-host systems to very large collections of hosts and I/O facilities.

Since routing within a subnet is vendor-specific, it cannot be assumed that one vendor’s SM has any understanding whatsoever of the routing algorithms used by another vendor’s SM. The SM taking over may not even be licensed to use the routing performed by the original SM; there are IBTA member companies which own patents in this area.

Lack of knowledge by the failed-to SM of the routing algorithm is an extremely severe restriction. To understand why, consider that a primary task of the SM during operation is the creation and removal of paths through the subnet. This is not just performed at power-on. It must be done when equipment is added and removed from the subnet (hot plug and unplug), and also must be done when multicast groups are created and destroyed. Routing activities continue throughout the operational life of the subnet.

Now imagine the situation of some poor benighted SM that takes over a subnet routed in an unknown manner. The SM can examine the routing tables in the switches; and also examine the use of virtual lanes, deducing in some way which are used for quality-of-service and which to break cycles in routing; and thereby figure out how packets are transferred from point to point. But how does it alter the routes or create new ones? In the Turing sense that anything can be programmed, this may be possible. However, the SM will be reduced to using what amount to maze-running techniques, none of which scale well to subnets of the size feasible in IBA. Furthermore, exactly how this is done efficiently without accidentally creating cycles in the routing, and therefore deadlock situations, is unclear; cycle (deadlock) detection would most likely have to be done separately, an operation that does not scale well. Furthermore, since the SM already has its own native routing protocol, this process would have to be bolted on in addition to its native capabilities.

In contrast, when the SM knows the algorithm used to create the routing, modifying those paths or determining which paths exist is usually an extremely efficient process. Incremental changes using data stored about the existing routes can be done very quickly, even for subnets that are extremely large, e.g., thousands of nodes. Furthermore, there is no additional question of avoiding deadlock; that is built into any correct routing algorithm.

So the IBA decision to leave routing vendor-dependent makes failover between different vendor’s SMs problematical. Of itself, this situation alone might be taken to be a good argument for defining some standard routing algorithm that, for example, was optional. Customers could choose to use it, or SM vendors using it, so that a different vendor’s SM could take over, if that were deemed of greater value than whatever benefits a proprietary routing technique might have.

5 Maintaining Consistent Data

Unfortunately, routing alone is not the whole issue.

The very concept of a standby subnet manager connotes failover and high availability in the face of failure. There is an axiom that must be used when designing for high availability: Fail-
ures always happen at the wrong time. Systems that maintain their availability only when the stars are appropriately aligned are not an appropriate design target for many situations.

Keeping that in mind, notice that Subnet Managers will usually keep on hand, for easy access, a variety of data that reflects the current state of the subnet they are managing. This is not mandated by the specification, but it is a highly likely implementation strategy that can enhance runtime responsiveness, reduce initialization time, be a basis for support of cloning configurations for multiple installations, etc. That data will be updated, of course, as operations are done on the subnet.

Now put that together with the axiom mentioned: Failures will happen just when the SM is updating that data. If care is not taken, the SM will cease operation just when an update is partially done, so the data is inconsistent.

This far from an unknown problem. A number of very reliable techniques have been devised to maintain data integrity under failure conditions, originally developed by the database and OLTP communities. These techniques are collectively known as maintaining transactional semantics, and discussions of how they work can be found in any intermediate or higher-level text or reference concerning databases. It's therefore valid to assume that any given SM vendor intent on producing a highly-available product will use such techniques.

The difficulty one runs into across vendors, however, is that transactional semantics must be provided between different vendors' copies of the data. Otherwise the SM which takes over will have an incomplete or scrambled view of what the subnet contains and how it is operating, and so may well throw the subnet into chaos.

Maintaining transactional semantics between separate instances of data, as in a standby and a master SM, is also not an unknown problem; it is closely associated with distributed transaction processing. There are two basic ways to go about it, depending on whether one assumes the data is kept on shared storage, and access switched from master to standby; or the data is instead replicated on separate storage units.3

The switchover (shared storage) case is illustrated in Figure 2. There, the master and the standby both have access to a single copy of the data (often on duplexed disk or RAID to tolerate storage failures). Both understand the data formats, and, furthermore, understand the techniques used to maintain data integrity on the disks: write-ahead logging, write-behind logging, or whatever is used. During normal operation, the master sends regular “heartbeat” messages to a standby, repeatedly saying “I'm OK!” to prove it's still operational; but otherwise there is no other overhead.

On failure in the switchover case, as illustrated, the standby cuts the failed system off from the data (in case it goes berserk); runs whatever log-scanning technique is needed to ensure data self-consistency; and proceeds to take over. Provision for multiple standby systems is simple,

---

3. This discussion is simplified. For more information, see, for example, In Search of Clusters by Gregory F. Pfister, published by Prentice-Hall.
and involves no additional overhead, but does require that all the standby systems have access to the data store.

The replication ("shared nothing") case, in contrast, is illustrated in Figure 3. The master and standby maintain separate copies of the data; each can have its own independent data format, and store the information using its own independent logging protocol (which is still needed). The storage media on each side can also be completely different; for example, in some circumstances data can be replicated in memory alone rather than using disk. During normal operation, in addition to “heartbeat” maintenance, the master must send every intended change to the standby, and ensure the two copies are accurately synchronized, even (especially!) if failure occurs while a change is being communicated. Doing so is again a well-understood technique called a "two-phase commit protocol"; it involves exchanging multiple messages for each alteration performed, and requires specific semantics for the logging operations on each side.

Failover in the replication case is simple: The standby begins operation using its own replica of the data, possibly after backing out partial changes that happened to be in progress at failure. Provision for multiple standby systems requires that multiple copies of all changes be transmitted, and all the systems involved must participate in the two-phase commit protocol.

The area of computer science and software/hardware engineering covering this type of operation, transaction processing, is extremely well-researched. Its results have been deployed in uncountable numbers worldwide, dating back to the 1960s and 1970s; every commercial database system and transaction monitor uses it in some fashion. As a result it can be said, with a significant degree of certainty, that the two methods outlined above are the only ways to achieve failover without data corruption.

The problem is that neither was considered suitable for inclusion in the IBA specification.

Switchover has very low overhead and scales well with number of standby systems if the storage interconnect scales well (like InfiniBand) — but would require all vendors to standardize all on-disk data formats at all levels, down to and including the file system used; otherwise another vendor’s system cannot directly use the data on the disk. This was immediately rejected as infeasible.

Replication allows independent data storage formats. It has some seemingly less desirable properties in overhead and scaling the number of standby SMs, but the rate of updates in this application is likely low enough that this may be acceptable. Also, two-phase commit protocols have been successfully standardized, e.g., in distributed-object support systems like CORBA, Enterprise Java Beans, and COM.NET. However, the inclusion such a protocol was considered inappropriate for the IBA, for three reasons:

- First, there was significant question as to whether the level of operation of the management architecture was appropriate to high level operations such as two-phase commit; these capabilities are usually implemented at significantly higher levels of the software stack than is implied by the IBA management packets.
- Second, this is simply a complex, fairly subtle area, deemed too complex an area to work in for the initial releases of the IBA specification.
- Third, there is an issue of allowing for lower-end, less complex and less expensive subnet manager implementations. Nothing requires that an SM vendor actually maintain its inter-
nal data transactionally; there is complexity and overhead involved which for some market areas may be deemed unnecessary. Ruling out such implementations by requiring some form of transactional semantics was also considered inappropriate. At the same time, it seemed very inappropriate to include in the specification some form of failover facility was not transactional, since that would be standardizing and requiring implementation of something that is known to work incorrectly.

• Finally: Transactional semantics aren’t the only issue inhibiting failover between vendors; it’s just another speed bump, or, perhaps, nail in the coffin. Even if it were defined in the IBA specification, and it perhaps could be, failover between different vendors still wouldn’t work because of the routing issue discussed above.

For those reasons, it was decided that arranging for transactional semantics between SMs was not appropriate for the specification.

6 Support For Level 3

As described above, there are two problems which might in theory be solved to allow the highest possible degree of interoperability (failover): routing, and maintaining correct data. But doing so would, if possible at all (for routing), at least require substantial and complex additions to an already rather voluminous specification. Either one alone may or may not be considered a tipping point against inter-SM failover, but together they provide a significant argument against it. It can also be argued that the limits on interoperability in IBA—on-the-wire protocols only—further argue against this. It appears inappropriate, to say the least, that an unexpected (by definition) failure should present a customer with an unfamiliar GUI, as well as new collections of messages sent to other management systems.

That leaves offline and online migration, levels 3 and 4. Both of these are pre-planned activities, explicitly initiated by a customer, so the issues of unfamiliar GUI and unanticipated messages don’t apply. Online migration still runs head-on into the routing problem. Offline migration, in which a subnet is rerouted, completely avoids routing and data consistency issues; it therefore appeared least problematical, and was chosen as the level of interoperability supported.

It is important to note that support for offline migration—level 3—does not mean “no support for interoperability.”

There are two separate areas of support explicitly provided in the IBA specification:

• Ensure that only one vendor’s SMs become master or standby SMs, even though multiple vendors’ SMs may be present in a subnet
• Provide everything necessary for the off-line transition between different vendor’s SMs, making it as simple as feasible for a customer.

Ensuring that only one vendor’s SMs become master or standby occurs at two different levels. The PortInfo:CapabilityMask.IsSMdisabled bit can be set to cause an SM on a port to not even attempt to become a master, i.e., to neither attempt to discover the network nor engage at all in the initial protocol among SMs to determine the master. Setting that bit “1” except where known desired SMs reside will keep all but those SMs off the subnet. Since mistakes can be made in the setting of such bits, there is additional protection available through the PortInfo:M_Key component. This can be used to prohibit access by an SM without the key to any data about nodes on the subnet. While it alone does not keep those SMs from becoming active, it can keep them from doing harm; and the intended master SM can report to the system administrator about “foreign” SMs which should have been disabled.
Conclusion, and Towards Level 4

The second area of support, enabling off-line transition, is provided by Subnet Administration attributes. There are 19 Subnet Administration attributes, total, whose support is required for an implementation to be compliant with the specification and its options. Of those, only six are required for normal operation: MultiPathRecord, TraceRecord, MCMemberRecord, PathRecord, ServiceRecord, and NodeRecord.

All of the other 13, which primarily duplicate information in Subnet Management attributes, exist solely to support this aspect of level 3 interoperability. Their only purpose is to allow one SM vendor (vendor A) to create a utility program, not an SM, which runs while another vendor’s SM (vendor B) is in charge. This utility can read all the architected information about the subnet and store it away in vendor B’s chosen format. In that way, a customer can switch to vendor B’s product without re-entering of “nonalgorithmic” data which must be provided by the customer, such as key values.

In addition, there is Subnet Administration’s facility for trusted requests: When the 64-bit SA_Key element of the SA header contains a valid SA_Key value, the request is considered trusted. Only trusted requests are allowed to return information that would allow access to information an indiscreet or buggy application could use to compromise subnet integrity, such as PortInfoRecord: M_Key, P_KeyTableRecords, ServiceRecords:ServiceKey, etc. (Volume 1, section 15.4.1.2). The “read it all out” utility mentioned above can be provided with the proper SA_Key to enable reading that data, while other applications are not given that key.

Conclusion, and Towards Level 4

Interoperability is a key goal of the InfiniBand Architecture, and the effort expended on attaining interoperability in management attests to this. Six possible levels of interoperability were analyzed, and the best that was implementable was chosen, namely:

- Any IBA-compliant subnet manager is able to manage any subnet formed from IBA-compliant entities. Heterogeneity in both IBA-compliant hardware and software is supported.
- Migration from one vendor’s subnet manager to another’s is supported, with facilities explicitly included to make this as simple as possible for a customer, insofar as this can be accomplished within the bounds of the area standardized by IBA.
- However, different vendors’ subnet manager implementations cannot simultaneously coexist on a single IBTA subnet, and, as a corollary, failover between subnet managers from different vendors is not supported. The primary reasons for this are that routing is vendor-dependent, and successful high-availability failover demands transactional semantics in the data exchange between subnet managers, which was deemed inappropriate to standardize.

In the future it may be possible to standardize level 4 interoperability: online migration between subnet management implementations, without an outage. Doing so would involve standardizing some form of interaction between the master subnet manager and other entities to achieve the swap of managers. This might include, for example, having the master enter some type of well-defined quiescent state in which no data updates are made; that would allow data to be transferred without transactional complexities. Routing will still be an issue; it is at present unclear how to deal with that. This direction may be pursued for future releases of the IBA specification if the member companies of the IBTA consider doing so a worthwhile endeavor.

Even without further work, however, IBA provides in its current specification the most management interoperability that is realistically possible.