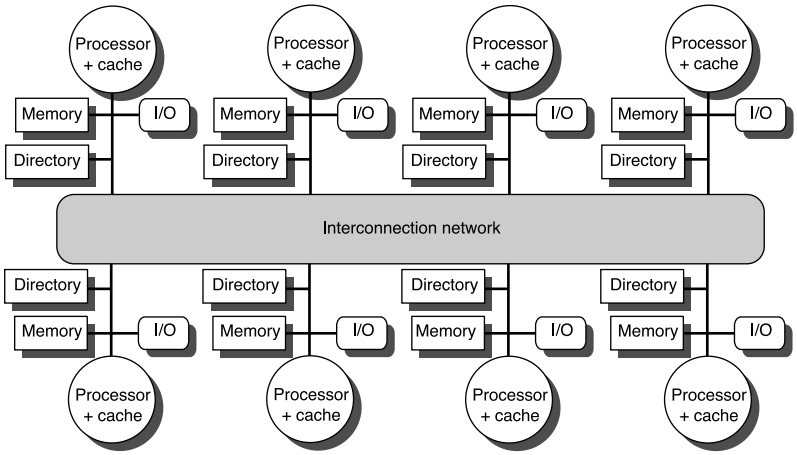
Cache Coherence Tutorial – Part 2 Directory Based

The snooping bus protocol tutorial was explained using a lot of detail and I'm not going to use that approach again here. If a student is still unsure of their grasp of the whole topic you could try the same approach for the directory-based protocol. The directory-based protocol is similar in some ways to the snooping bus protocol but there are some significant differences. The main difference is in terms of scale in respect to the number of processors involved in the multiprocessor system. In the snooping bus protocol each memory address was sent out on a common bus and each processor and its cache would receive and check out the address to determine if any local actions needed to be done. So all the memory operations, read/write/write back, would occur on the single bus for all the processors in the system even if the one particular processor has nothing to do with the memory location. When there are only a few processors involved the bus bandwidth can handle the quantity, even if only a small part of the memory addressing concerns any one processor. But when there are massive numbers of processors like hundreds or thousands, the bus will become very busy, often processors will be checking out addresses that have nothing to do with them. So this means that some kind of directory memory has to be included to keep track of which processors are concerned with which areas of memory so that they can only talk to those caches and memory that they are concerned with. With this approach the memory bus can be interconnected in some way so that only the processors concerned with the same area of memory are affected by the communication of the data, with possibly a small number of other processors that are simply along the path. The actual physical layout of the memory becomes a factor too, because rather than a global memory in one location the memories can be distributed in various locales. The snooping bus protocol could work in a distributed environment also but in the directory-based protocol the addresses can be directed along a path that provides the best communication between the two points and does not affect other paths that it has no interest in.

So the block diagram of figure 6.27 shows the processors with cache combined together connected to a line that contains a memory area and a directory area that attach to some kind of interconnection network. The cache will contain the cache coherence bits that we saw in the snooping bus protocol that indicated the state of the cache lines for that processor. The directory area will contain state bits for the memory blocks for the memory belonging to this processor and it will also keep track of which processors are presently using a copy of any particular memory block in this processor's memory.



© 2003 Elsevier Science (USA). All rights reserved.

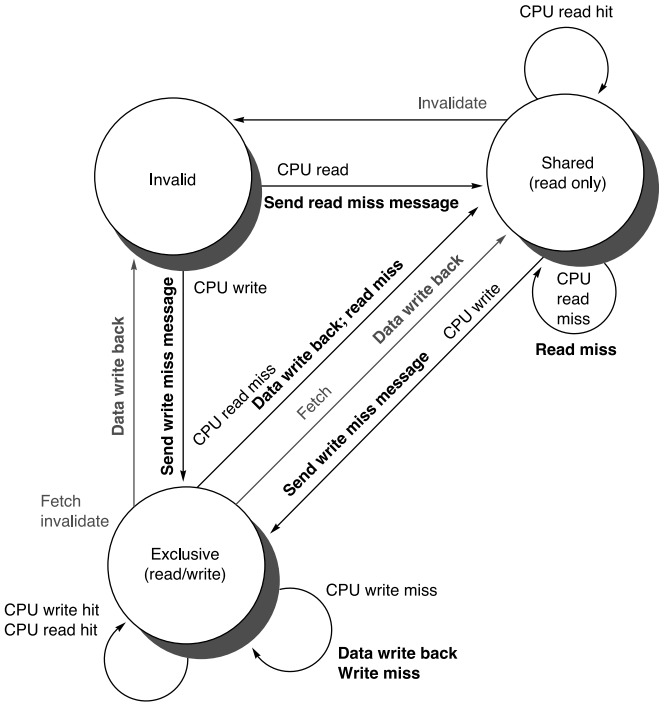
The most significant difference with the directory-based protocol is that the real memory has a directory block added to it that refers to the data in the real memory distributed in every processor. This directory block keeps track of the number of processors that are using the particular line of real memory. It also keeps track of whether a line has been written to in some processor and therefore the correct data now resides in that processor's cache line. Prof. Li showed the diagram for a transaction in this way:

Local processor

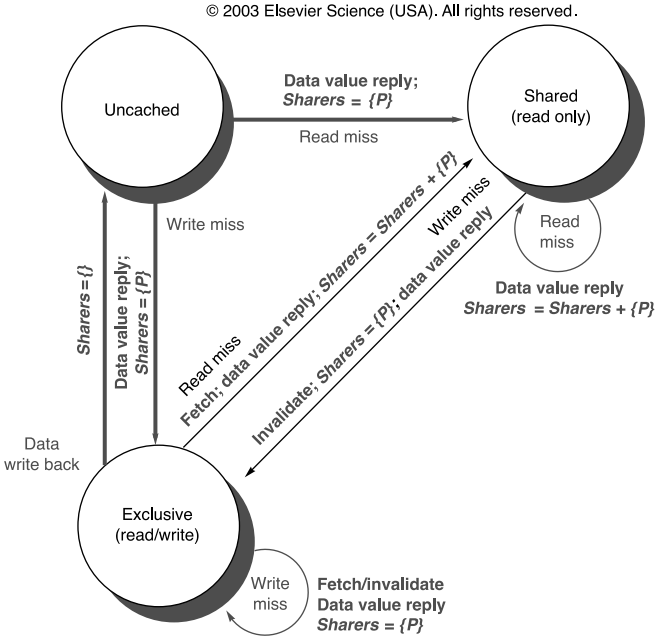
Home memory

Remote processor

Using the diagrams of figures 6.29 and 6.30 to follow the actions for the cache lines and also the actions in the directory for the memory location, we can follow a couple of situations.

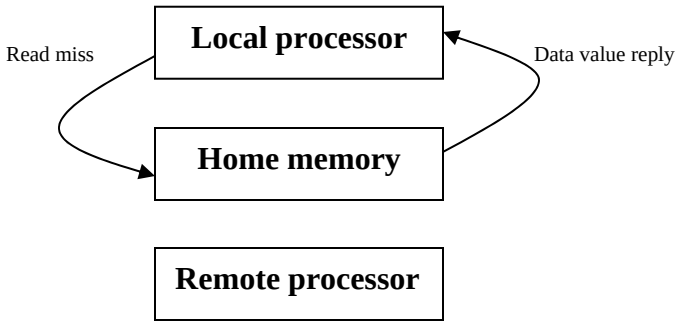


© 2003 Elsevier Science (USA). All rights reserved.



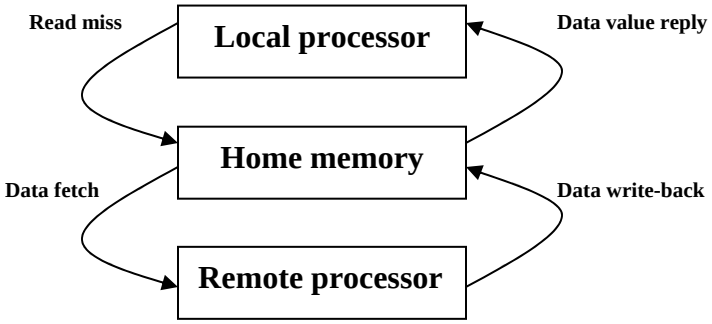
© 2003 Elsevier Science (USA). All rights reserved.

So lets do a simple read of a memory location where all the caches are initially empty. The local processor will get a read miss because the cache line will be in the invalid state. This will cause a read miss message to go out on the bus with the address on the bus address lines. Depending on the capabilities of the interconnect network, this address will be interpreted and the read request will go to the appropriate processor where the real memory exists. The directory will indicate that the state of that memory location is presently uncached. The data will be sent back to the requesting processor and the directory will be updated to put the state of this memory line to shared and it will also send the signal data value reply as an acknowledgement back to the requesting processor. The id of the requesting processor will be recorded in a sharers slack list in the directory block. The local processor's state for that memory location will be set to shared.

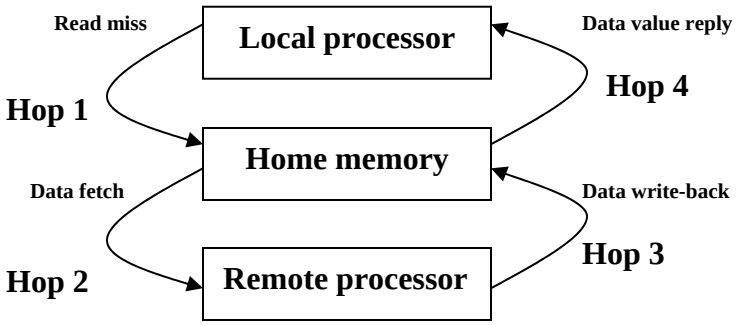


Let's get a little bit more complicated now. Let's assume that one of the processors has written to a location previously that our local processor now wants to read. The local cache will be invalid or occupied by another memory address. This time we will assume that it is invalid. So the read miss signal goes out on the bus with the address and the processor id. This will be directed to the home processor. At the home processor the state will indicate that this location is exclusive to a remote processor. So a data fetch signal is sent out on the bus through the interconnect network to the processor that was specified in the home directory sharers stack list. The remote processor will have the data in its cache and it will be in the exclusive state. The data will be put on the bus to go back to the home processor and the signal data write-back will be sent to the home processor. The remote processor will finally change its state from exclusive to shared and it is finished. The home processor will receive the data, put it in memory and send the data on to the requestor (our local processor). It will send the data value reply signal to the requesting processor and will finally change its directory state from exclusive to shared. The requesting

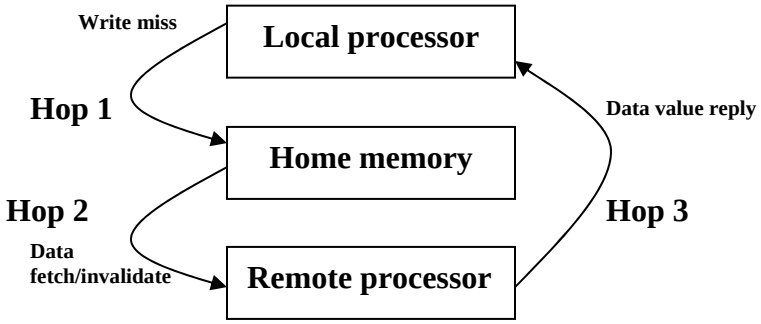
processor (our local processor) will receive the data, put it into the cache line and update the state that was invalid to make it shared.



Note here that we now see the 4-hop protocol that Prof. Li discussed in class.



Now it is interesting to note that with the atomic way that we deal with the cache line, that is that we only have one exclusive version in any cache or the version in memory is correct, means that in certain circumstances a request from the local processor can bypass the rewrite into memory on the return pass. Let's say we are doing a write in the local processor and another remote processor has an exclusive copy of the address of concern. Hop 1 will occur as above except it will be a write miss, and Hop 2 will likewise be the same. But for the return path it is unnecessary to go back to the real memory because the local processor will end up with the exclusive copy anyway and that data was not sent out with the original request. So we can skip the extra hop in the return path by sending the data directly to the original requesting processor. There are some details that have to be attended to, though, such as updating the home directory sharers stack list to indicate the new processor id now has the exclusive copy.



Remember that the 3-hop protocol is an experiment and not the standard. There are things that would be different in the figures for the state transitions. For exam questions follow the instructions specified and if nothing is stated then the assumption would be to follow the 4-hop protocol of the textbook. And also, just as everything else that has been covered in this course there are no hard rules that define all computers, these are just guidelines that introduce the students to generalities in order to develop an understanding of the many topics.

Assignment #5 has a cache coherence application mixed together with an interconnect protocol. It looks like a good exam type question, I suggest you investigate it and check the answers.