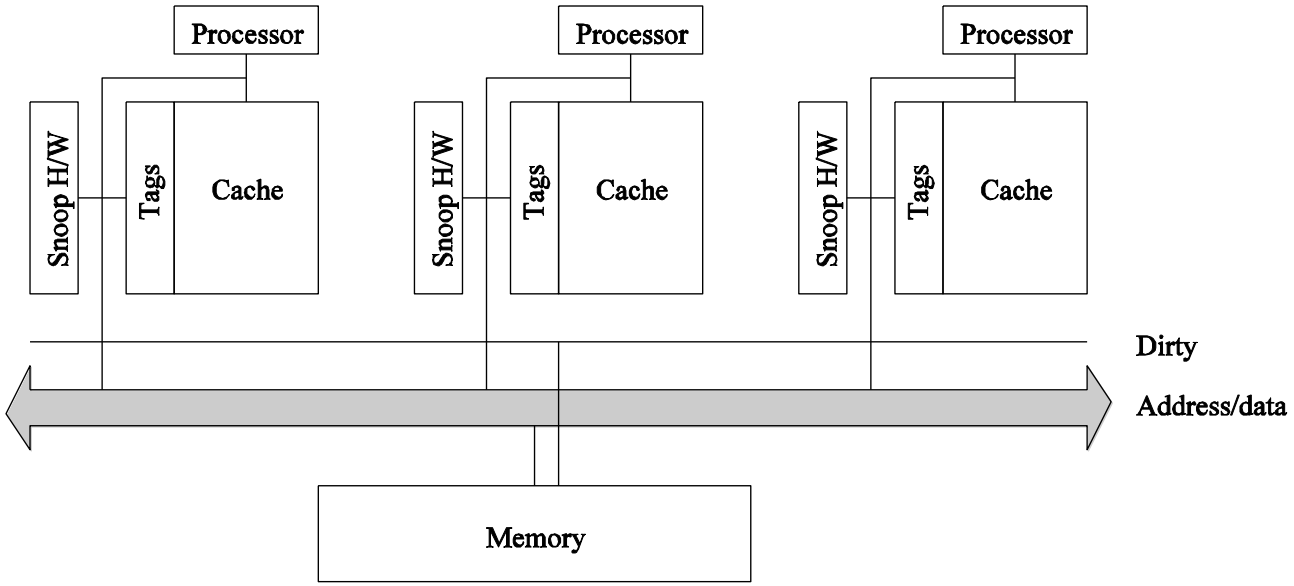
SNOOPY CACHE SYSTEMS

How are invalidates sent to the right processors?

In snoopy caches, there is a broadcast media that listens to all invalidates and read requests and performs appropriate coherence operations locally.



A simple snoopy bus based cache coherence system.

PURDUE NExT

PERFORMANCE OF SNOOPY CACHES

Once copies of data are tagged dirty, all subsequent operations can be performed locally on the cache without generating external traffic.

If a data item is read by a number of processors, it transitions to the shared state in the cache and all subsequent read operations become local.

If processors read and update data at the same time, they generate coherence requests on the bus - which is ultimately bandwidth limited.

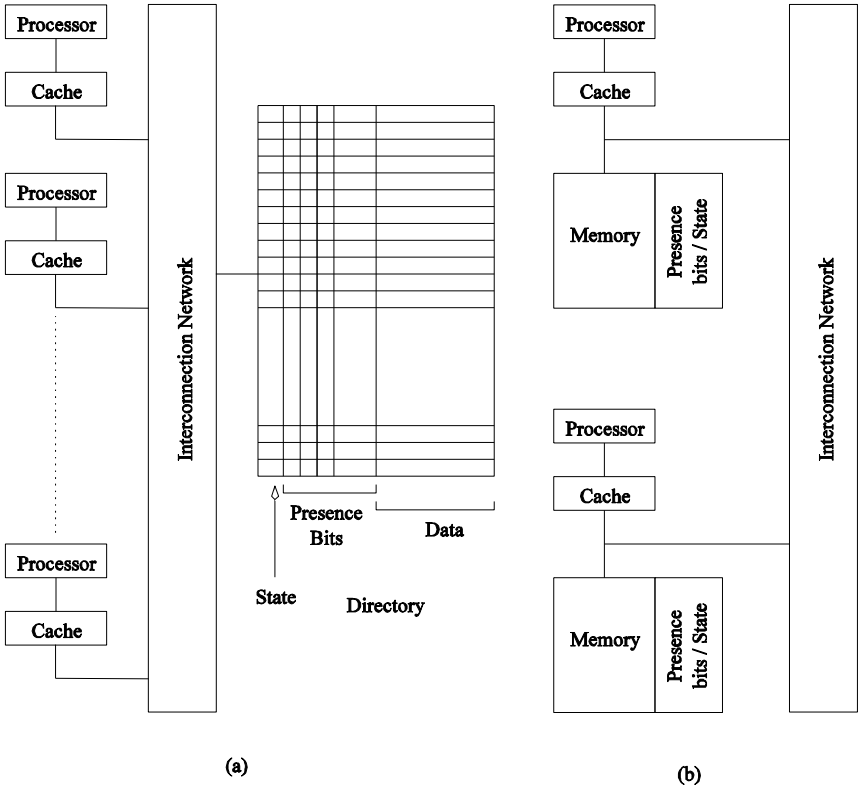
DIRECTORY BASED SYSTEMS

In snoopy caches, each coherence operation is sent to all processors. This is an inherent limitation.

Why not send coherence requests to only those processors that need to be notified?

This is done using a directory, which maintains a presence vector for each data item (cache line) along with its global state.

DIRECTORY BASED SYSTEMS



Architecture of typical directory based systems: (a) a centralized directory; and (b) a distributed directory.

PURDUE NExT

PERFORMANCE OF DIRECTORY BASED SCHEMES

The need for a broadcast media is replaced by the directory.

The additional bits to store the directory may add significant overhead.

The underlying network must be able to carry all the coherence requests.

The directory is a point of contention, therefore, distributed directory schemes must be used.

PURDUE NExT

COMMUNICATION COSTS IN PARALLEL MACHINES

Along with idling and contention, communication is a major overhead in parallel programs.

The cost of communication is dependent on a variety of features including the programming model semantics, the network topology, data handling and routing, and associated software protocols.

MESSAGE PASSING COSTS IN PARALLEL COMPUTERS

The total time to transfer a message over a network comprises of the following:

- *Startup time* (t_s): Time spent at sending and receiving nodes (executing the routing algorithm, programming routers, etc.).
- *Per-hop time* (t_h): This time is a function of number of hops and includes factors such as switch latencies, network delays, etc.
- *Per-word transfer time* (t_w): This time includes all overheads that are determined by the length of the message. This includes bandwidth of links, error checking and correction, etc.

STORE-AND-FORWARD ROUTING

A message traversing multiple hops is completely received at an intermediate hop before being forwarded to the next hop.

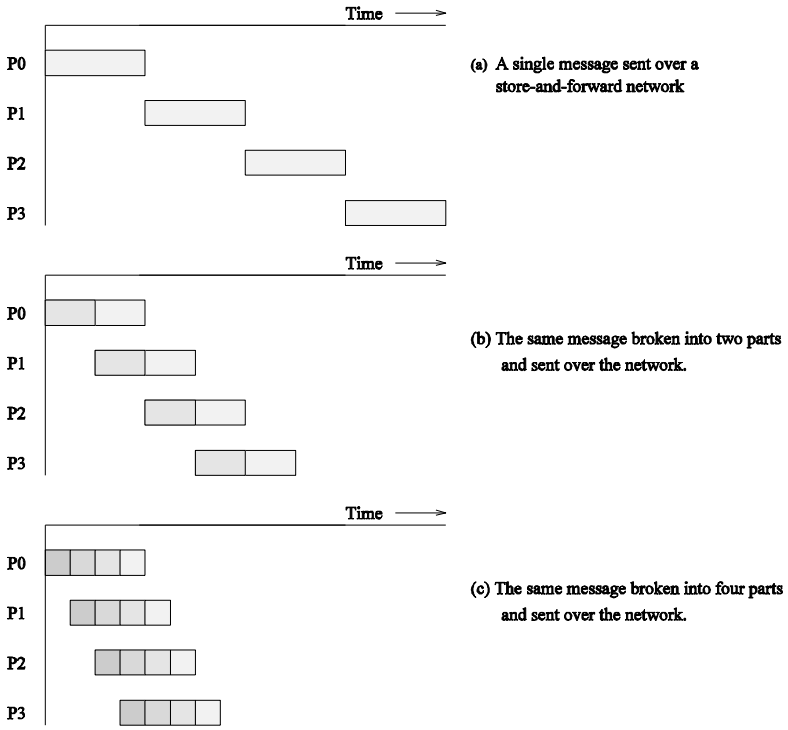
The total communication cost for a message of size *m* words to traverse *l* communication links is

$$t_{comm} = t_s + (mt_w + t_h)l.$$

In most platforms, t_h is small and the above expression can be approximated by

$$t_{comm} = t_s + mlt_w.$$

ROUTING TECHNIQUES



Passing a message from node P_0 to P_3 (a) through a store-and-forward communication network; (b) and (c) extending the concept to cut-through routing. The shaded regions represent the time that the message is in transit. The startup time associated with this message transfer is assumed to be zero.

PURDUE NExT

PACKET ROUTING

Store-and-forward makes poor use of communication resources.

Packet routing breaks messages into packets and pipelines them through the network.

Since packets may take different paths, each packet must carry routing information, error checking, sequencing, and other related header information.

The total communication time for packet routing is approximated by:

$$t_{comm} = t_s + t_h l + t_w m.$$

The factor t_w accounts for overheads in packet headers.

CUT-THROUGH ROUTING

Takes the concept of packet routing to an extreme by further dividing messages into basic units called flits.

Since flits are typically small, the header information must be minimized.

This is done by forcing all flits to take the same path, in sequence.

A tracer message first programs all intermediate routers. All flits then take the same route.

Error checks are performed on the entire message, as opposed to flits.

No sequence numbers are needed.

PURDUE
NExT

CUT-THROUGH ROUTING

The total communication time for cut-through routing is approximated by:

$$t_{comm} = t_s + t_h l + t_w m.$$

This is identical to packet routing, however, t_w is typically much smaller.

SIMPLIFIED COST MODEL FOR COMMUNICATING MESSAGES

The cost of communicating a message between two nodes *l* hops away using cut-through routing is given by

$$t_{comm} = t_s + lt_h + t_w m.$$

In this expression, t_h is typically smaller than t_s and t_w. For this reason, the second term in the RHS does not show, particularly, when m is large.

Furthermore, it is often not possible to control routing and placement of tasks.

For these reasons, we can approximate the cost of message transfer by

$$t_{comm} = t_s + t_w m.$$

SIMPLIFIED COST MODEL FOR COMMUNICATING MESSAGES

It is important to note that the original expression for communication time is valid for only uncongested networks.

If a link takes multiple messages, the corresponding t_w term must be scaled up by the number of messages.

Different communication patterns congest different networks to varying extents.

It is important to understand and account for this in the communication time accordingly.

COST MODELS FOR SHARED ADDRESS SPACE MACHINES

While the basic messaging cost applies to these machines as well, a number of other factors make accurate cost modeling more difficult.

Memory layout is typically determined by the system.

Finite cache sizes can result in cache thrashing.

Overheads associated with invalidate and update operations are difficult to quantify.

Spatial locality is difficult to model.

Prefetching can play a role in reducing the overhead associated with data access.

False sharing and contention are difficult to model.

PURDUE NExT