Semantics for Nondeterministic Asynchronous Broadcast Networks

R.K.*Shyamasundar* §*, K.T.*Narayana* §, T.*Pitassi* †

1. Introduction

Distributed computing can be broadly characterized as a spectrum of activities varying in their degree of decentralization, with one extreme being remote computer networking and the other extreme being multiprocessing. Without loss of generality let us consider distributed programs to be a collection of processes cooperating to accomplish a common objective rather than processes with individual objectives but no common goal. There have been several *message passing* primitives for distributed programming. Broadly, we can characterize the primitives to (a) synchronous message passing (wherein both the sender and the receiver are delayed for the dual agent), and (b) asynchronous message passing (wherein the sender is not delayed). In this paper, we concentrate on varieties of asynchronous message passing primitives. The advent of VLSI hardware and various broadcast communication systems such as ETHERNET [MeB76] has made it possible to use *broadcast communication* as a linguistic feature for designing programming languages for distributed programming . The underlying principle of *broadcast* is that it is *one-to-many* in the sense that the message sent by a process is received by a set of processes. This is in contrast to the point-to-point asynchronous message passing. [Sch82] has articulated the use of *broadcast communication* as a paradigm for distributed programming and Chang [Cha84] has shown how the design of distributed database systems gets simplified by using a broadcast network. [Geh84] has performed studies on various examples to assess the use and convenience of broadcast programming in the context of distributed computing.

In this paper, we develop a language for asynchronous broadcast communication based variants of the CSP (cf. [Hoa78]). The language is then used for modeling and analysing the various categories of *broadcast communications*. Real-time models are essential for formalization and characterization of available broadcast architectures. The need and the notion of realistic models of real-time concurrency has been articulated in [KSD85, KSD86] by developing a spectrum of models ranging from interleaving to maximum parallelism models.

A formal semantics for asynchronous broadcast networks will permit an analysis of the structures suitable for broadcast and throw light on the design of languages with broadcast communication mechanisms. No such attempt at a formal treatment of broadcast communication has been made uptil now in the literature. Schlichting and Schneider[ScS84] have extended the interference freedom of CSP to asynchronous point-to- point message passing. The method is non-compositional mainly due to the underlying techniques. Jonnson[Jon85] describes a compositional specification of nondeterministic asynchronous networks (again point-to-point). Both these attempts are only in the context of *interleaving* models. As already said, the various broadcast categories cannot be distinguished or provided with realistic models unless one considers the maximum parallelism models. Our work is the first attempt at a classification and a formal treatment of broadcast. The contributions of the work are summarized below:

• We present a compositional denotational semantics for a spectrum of nondeterministic *broadcast networks* of processes using a simple semantic domain. Broadly, we consider two classes of broadcast communications, namely, *unbuffered broadcast* and *buffered broadcast* and use either *interleaving* or *maximum parallelism* models so as to provide a realistic model of the various variants. Broadly, we consider:

- unbuffered broadcast
- buffered broadcast
  - unbounded buffers
  - bounded buffers - with the condition that the transmission is immediate in the sense that the message need not be sent to over all the designated channels.
  - bounded buffers -- with the condition that the message is sent to all the designated channels as and when a buffer becomes available with reference to the corresponding channel.
  - atomic broadcast -- initiation is delayed till each channel has at least one buffer for the message.

§ Department of Computer Science, The Pennsylvania State University, University Park, PA 16802.
* Computer Science Group, Tata Institute of Fundamental Research, Bombay-400 005, India.
† AT&T Bell Labs, 6200 East Broad Street, Columbus, Ohio, Oh 43213

• The semantics provides a realistic model of the various broadcast categories. Thus, reliable and lossy transmission of order preserving asynchronous communication medium is captured through the *prefix property* and *the subsequence* property, respectively, of the projections of the histories of the processes.
• It leads to a specification-oriented semantics of broadcast networks -- thus, forming a basis for compositional verification of such systems.

2. Language Structures for Broadcast Networks

We investigate broadcast networks within which a process performs a single action at any time and has broadcast communication and choice reception as communication primitives. Each process in the broadcast network has a distinct processor and does not share variables with the others. Thus, cooperation among processes is achieved only through communication. With each process in the broadcast network, we associate two disjoint sets of channels. One set of channels are called *out* (broadcast) channels and the other set of channels are called *in* (receive) channels. A process names a subset of its *out* channels in the broadcast command. Similarly a process names a subset of its *in* channels in the receive command. Each channel connects exactly two processes. The syntax of the language is given below:

P ::= S | N

S ::= x :=e | skip | D??x (*receive*)| D!!e (*broadcast*) | $S_i;S_j$ | $[[]_{i=1}^{n} g_i \rightarrow S_i]$ | $*[[]_{i=1}^{n} g_i \rightarrow S_i]$ | [N]

g_i ::= b | $D??x$ | $D!!e$ | $b;D??x$ | $b;D!!e$

N ::= S_1 || S_2

Note: It may be observed that the network can have nested processes and further, the network is static. We have not included either the finite *delay* commands or the *delay* guards as we are interested in showing the applicability of *interleaving* or *real-time* models for the various broadcast schemes rather than proposing a language. In the following, we describe informally the broadcast and receive commands. The other commands have essentially the same meaning as in CSP. For the sake of convenience, we do not consider the *distributed termination* convention of CSP and we assume that the underlying boolean guards determine the *enablement* or otherwise of the communication guards.

2.1 Unbuffered Broadcast : In this scheme, there are no buffers associated with the channels. Thus, whenever a process executes the command $D!!x$, the message is immediately sent on all the channels. If the designated process is willing to receive the message at that point in time, it does so. Otherwise, the message sent on the channel is lost. Thus, unbuffered broadcast schemes model message loss with respect to a receiver's unwillingness or inability to receive a message at that time point at which the message is broadcast. Two simple examples of this scheme are (i) *radio announcements* and (ii) *broadcasting over a bus* -- as in a typical ETHERNET scheme.

2.2Buffered Broadcast : A spectrum of categories for broadcast schemes can be considered based on the waiting associated with the completion of a broadcast command. The extremes for such a waiting may be the complete waiting associated with synchronous communication and no waiting associated asynchronous communication. The provision of buffers leads to a compromise between the waiting and loss of messages in a broadcast command.

(a) *Broadcast with Unbounded Buffers*: In this scheme, we associate an unbounded number of buffers with each of the channels of the process. When a process executes the command $D!!x$, the message is placed in the buffers(queues) associated with each of its broadcast channels. Since the buffers are unbounded, there is no waiting associated with the broadcast command.

(b) *Broadcast with Bounded Buffers*: In the class of bounded buffer broadcast schemes, a broadcast may not succeed immediately (or forever) as the process may not be able to send messages over all the channels due to lack of empty buffers. Thus, depending on lossy or reliable transmission of messages and immediate or delayed initiation of the broadcast, we classify into the following categories.

(i) *Bounded Buffer Broadcast - immediate initiation*: In this scheme, a finite number of buffers (FIFO) is associated with each of the channels. Execution of a broadcast command $D!!e$ corresponds to sending the message e on all those channels of D having at least one empty buffer at that time and ignoring all other channels having nonempty buffers at that time. Note that the broadcast command is a deterministic one in the sense that there is no waiting associated with the command. Thus the command models message loss. Contention networks such as Ethernet[MeB76] and ring networks such as DCS [FFe73] implement such schemes with finite buffers associated with each processor in the network.

(ii) *Bounded Buffer Broadcast - delayed (possibly) initiation*: In this scheme, the execution of $D!!e$ corresponds to a repeated application of (i) till the message is sent on all of the channels.

(iii) *Atomic Broadcast*: It is essentially the same as (ii) except that the *initiation* of sending is delayed till the message can be sent over all the channels. Thus, the message is sent over all the designated channels at the same time.

In the following, we provide real-time models for all the categories except the unbounded buffer category. For the latter, we provide an interleaving model.

2.3 Receive Command D??x

An execution of the receive command *D??x* nondeterministically selects a message from the nonempty FIFO buffers of some channel in D and assigns the message to variable x. If the buffers of all the channels in D are empty, then the process waits for some message to arrive on some channel in D. For purposes of convenience of programming, one can make the command selective in the sense that the process receives messages of some *type* from some *specified channels* only. Note that we are assuming the buffers to be associated with channels rather than with the process or the node.

3. Semantic Domains

The domain for defining the meaning functions for the commands consist of prefix-closed sets of state-history pairs with subset ordering, i.e., $D = P(\Sigma \times H)$ where $\Sigma = S \cup \{\perp\}$, S being the *set of proper states* (i.e. partial functions from Id(set of identifiers) to V (set of expression values), and $\perp$ denotes *incomplete* computation (We are not distinguishing *incomplete* and *failure* states), and H = set of sequences of sets of communication assumption records (CAR's); in the history, we record claims of a process receiving and broadcasting messages over specified channels and claims for the impossibility of sending and receiving messages.

A set $X \in P(\Sigma \times H)$ is *prefix-closed* iff for all $<\sigma, h> \in X$, if $h' \leq h$ then $<\perp, h'> \in X$.

The *prefix-closure* of X, is defined by $PFC(X) \,\underline{\Delta}\, X \cup \{<\perp,\lambda>\} \cup \{<\perp,h'> \mid \exists\sigma\exists h \; <\sigma,h> \in X \;\wedge\; h' \leq h\}$.

Note that the domain forms a complete lattice with set-inclusion($\subseteq$); the lub is obtained by $\cup$ (set-union) and the least element is $\{<\perp,\lambda>\}$.

3.1 Structure of Communication Assumption Records

A message has the structure *<msg ,sender>* where *sender* designates the name of the process sending the value *msg*. The component msg has the structure <value, destination>. However, we will not explicitly indicate the destination in our CARs. The CARs used in describing the semantics are given below:

- $\Box$ denotes an internal action of a process.
- *<!,D ,msg>* - claims a successful broadcast of *msg* over the set of channels *D*.
- *<? ,d ,msg>* - claims a successful reception of a message *msg* on channel *d*.
- *<w !,D>* - claims the impossibility of broadcast on the channels in D. It may be noted that this assumption becomes meaningful only in cases of *buffered broadcast and the buffers are bounded a priori* ; i.e. the broadcast command waits for sending a message over the channels in D.
- *<w? ,D>* - claims that the buffers of channels in D are empty; thus, it denotes waiting for a message to arrive.

As the guards can be composed of either broadcast or *receive* commands, there is a need to distinguish the waiting on output(broadcast) and input(receipt). Note that though it is possible for a process to be waiting for a buffer to be filled in all the categories described earlier, the waiting for output (i.e. broadcast) can be claimed in a subset of categories. For this purpose, we have the following message structure corresponding to claims for the input command and broadcast command. In the case of *broadcast*, it is possible that the transmission takes place on a subset of channels as buffers for other channels are full; note that this situation is possible only for the category of *bounded buffers.*

- $<!,D, msg, w!,D'>$ - claiming a successful broadcast of *msg* on the set of channels D and an impossibility of sending messages over any of the channels in D' as the buffers for the channels in D' are full.

Note that a similar message structure is not necessary for the receive command due to the asymmetry of the broadcast and receive commands. We need one more CAR structure claiming the waiting for input and broadcast simultaneously. The structure of this CAR is given below.

- $<w!,D, w?, D'>$ - claiming an impossibility of sending a message on any of the channels in D and impossibility of receiving a message on any of the channels in D'.

3.2 Time Domain

We introduce a clock at a meta-level of reasoning for partially ordering the events in the distributed broadcast system and for achieving *maximum parallelism.* For purposes of convenience, we consider the time domain to be the domain of natural numbers. Also, we assume that *every action* takes *one unit of time*. For the *interleaving model* , we consider all possible interleaving of the commands treating a communication from one point to another point as *atomic*. For the *real-time semantics* , we use *Maximum Parallelism Model*[SaM81] as the underlying execution model. We refer to this model as the MAXPAR model. As argued in [KSD85], there is a need for a separate processor for each logically distinct process. Thus, we assume that each distinct logical process is associated with a designated physical processor. With this operational model, *maximum parallelism* can be understood as maximizing the number of instructions that can be executed simultaneously in a set of concurrently executing processes at any point of time without violating synchronization requirements .

Notation

1. *EXT* and *INT* represent the set of external channels and the set of internal channels of a process respectively. In our operational model the input and output channels of each process are distinct and a channel connects at most two processes. Hence, if d is a channel, it would be the *input* channel for one process and the *output* channel for another process. We use d? to represent the *in* side of channel d and d! to represent the *out* side of channel d.
2. $h[k]$ to denote the k^{th} element of sequence h.
3. $h \le h'$ denotes that h is a prefix of h'.
4. h^k denotes the sequence obtained from h by chopping the suffix h[k+1]... from h; we will refer to this as the k^{th} prefix of h. λ denotes the *empty* sequence and ϕ denotes the *null* sequence.
5. h is said to be a *subsequence* of h' if h can be obtained from h' by deleting some symbols of h'. $h \underset{sub}{\le} h'$ denotes that h is a subsequence of h'.
6. The projection h $\uparrow$ d? is defined as the sequence of values received on channel d. Note that the projection is *order preserving* and the extension of the projection operation to sets of channels or sets of traces (histories) follows naturally. Similarly $h \uparrow d!$ is the sequence of values sent over channel d. Because of the asymmetry in the broadcast and receive, we need to distinguish the *in*put and the *out*put sides of the channels.
7. Further, we define a projection operator $h \uparrow_t D$ as the sequence h obtained from h by projecting onto the channels of D ; in other words, all records not having any names from D are dropped. Note that empty set is denoted by $\Box$.
8. We say $d \in chan(h_1)$ if either d? or d! is in h_1.
9. We can interpret X $\in \sigma \times H$ as the set of all possible computations of a program.
   - $<s,h> \in X$ with $s \in S$ models a computation of P that terminates in s with history h.
   - $<\perp,h> \in X$ models: either an incomplete computation of $<s,h'>$, $h \le h'$,
     or it is an element in the chain of approximations modeling an *infinite* computation;
   - $(\perp,\lambda)$ denotes failure.

Though, we need to distinguish a separate state for nondeterministic failure, we have ignored that for the sake of simplicity. In our interpretation, the only observable behaviours are: termination, and divergence (We do not distinguish deadlock from divergence).

4. A Priori Denotational Semantics of Sequential Constructs

We assume the existence of strict semantic functions $V[\![e]\!](\sigma)$ for an expression e from the states, Σ, to the domain of values $VAL \cup \{\perp\}$; similarly, $W[\![b]\!](\sigma)$ maps boolean expressions to { true, false } $\cup$ $\{\perp\}$.

The meaning function M is defined as $M[\![S]\!]: \Sigma \rightarrow P(\Sigma \times H)$

Strict Function: $M[\![S]\!]\perp = \{ <\perp,\lambda> \}$ for any S.

Skip : $M[\![skip]\!]\sigma = PFC\{(\sigma, \{ \Box \})\}$
In *interleaving semantics*, $\Box$ can be treated as representing some local action (used for identifying divergence properties). In *real-time semantics*, it can be interpreted as representing one unit of time for the action.

Assignment : $M[\![x:=e]\!]\sigma = PFC\{(\sigma[V[\![e]\!]\sigma/x], \{ \Box \})\}$

Sequential Composition: Firstly, we extend the meaning function from $P(\Sigma \times H)$ to $P(\Sigma \times H)$.
For any $X \subseteq P(\Sigma \times H)$, $M^*[\![S]\!]X = \{<\sigma', h.h'> \mid <\sigma,h> \in X \ \wedge \ <\sigma',h'> \in M[\![S]\!]\sigma\}$ where '.' denotes the concatenation operator. Note that from this definition, it is clear that the extension of the histories is independent of their contents-- an important property of the meaning function for the purpose of compositionality. Also, it can be shown that M^* is strict and continuous.

$M[\![S_1;S_2]\!]\sigma = M^*[\![S_2]\!](M[\![S_1]\!]\sigma)$

The definitions of the alternative and iterative commands are given in the respective categories.

5. Unbounded Buffer Broadcast

We describe an *interleaved semantics* for this class assuming that no messages are lost.

Broadcast : $M[\![D!!e]\!]\sigma = PFC\{(\sigma, \{<!, D, V[\![e]\!]\sigma>\})\}$
The effect of the broadcast command is to place the message (the value of e) in the channel queue for each channel in the broadcast command.

Receive : $M[\![D??x]\!]\sigma = PFC\{(\sigma[v/x], \{<?, d, v>\}) \mid v \in VAL, d \in D\}$
The nondeterminism arises due to two reasons: the first is that a priori the value that will be sent by some other process cannot be determined; the second is due to the nondeterministic nature of the *receive* command.

Test: $M[\![b]\!]\sigma =$ if $W[\![b]\!]\sigma$ *then* $PFC\{(\sigma,\lambda)\}$ *else* $\{(\perp,\lambda)\}$
Note that the *test* does not append the history and *Abortion* is modeled by $<\perp,\lambda>$.

Selection: $M[\![A]\!]\sigma =$ if $\neg \bigvee_{i=1}^{n} W[\![b_i]\!]\sigma$ *then* $\{(\perp,\lambda)\}$ *else* $\bigcup_{i=n}^{n} M[\![g_i;S_i]\!]\sigma$

Iteration: $M[\![*A]\!]\sigma = \mu\psi.(\lambda\sigma.\text{if } \bigvee_{i=1}^{n} W[\![b_i]\!]\sigma$ *then* $\psi^*(M[\![A]\!]\sigma)$ *else* $PFC\{(\sigma,\lambda)\}$

5.1 Parallel Composition of Processes

Two processes S_1 and S_2 constituting a parallel composition $S_1 \| S_2$ will have its meaning function derived from the a priori semantics of S_1 and S_2. Such a function must ensure that for every pair of histories h_1 and h_2 of the computation possibilities of S_1 and S_2, respectively, every communication assumption made by a process about the reception of a message on a channel connecting S_1 and S_2 is realizable as a broadcast communication assumption in the history of the other. Such internally consistent histories can be merged by dropping internal communications so as to obtain the externally visible communication histories of the composed processes. Let *cset* be the set of common channels of S_1 and S_2.
$Merge(h_1,h_2) = \{h \mid h_1 \leq h \uparrow Chan(h_1), h_2 \leq h \uparrow Chan(h_2), h = h \uparrow chan(h_1,h_2),$
$d \in cset \rightarrow h_1 \uparrow d? \leq h_2 \uparrow d! \ \vee \ h_2 \uparrow d? \leq h_1 \uparrow d!\}$

Essentially, the conditions imply that all possible interleavings of h_1 and h_2 are taken such that the sequences h_1 and h_2 can be obtained by taking the respective projections . The merged sequence can have only elements from h_1 and h_2, and the values received on any input channel must be a *prefix* of those of the corresponding output channel. The last condition corresponds to the fact that the sequence of values received is a prefix of those sent.

Let π_i be projection functions from the complete state on to the variables of process S_i. Then,
$M[\![P_i \| P_j]\!]\sigma = \{(\sigma_i \times \sigma_j, h) \mid h \in merge(h_1,h_2) <\sigma_i,h_1> \in M[\![P_i]\!]\pi_i(\sigma), <\sigma_j,h_2> \in M[\![P_j]\!]\pi_j(\sigma)\}$
where $\sigma_i \times \sigma_j(x) = \sigma_i(x) \text{ if } \sigma_i(x) = \sigma(x) \ \wedge\ \sigma_i,\sigma_j \neq \bot,$
$\sigma_j(x) \text{ if } \sigma_j(x) = \sigma(x) \ \wedge\ \sigma_i,\sigma_j \neq \bot,$
$\bot$ otherwise

Network Abstraction: $M[\![[N]]\!]\sigma = \{(\sigma', h \uparrow EXT) \mid (\sigma',h) \in M[\![N]\!]\sigma \ \wedge\ EXT =$ external channels of N$\}$

6. Unbuffered Broadcast Communication

Broadcast: $M[\![D\,!!e]\!]\sigma = PFC\{(\sigma,\{<!,D,V[\![e]\!]\sigma>\})\}$
The message , e, is placed on the channels(or the bus) specified in D. Whether the message has been received by the receiver or not can be determined only when composing the processes. There is no waiting associated with this command.

Receive: $M[\![D??x]\!]\sigma = PFC\{(\sigma[v/x],\{<w?,D>^t<?,d,v>\}) \mid d \in D, t \geq 0, v \in VAL\}$
There is a waiting associated with this command as the process waits indefinitely for a message to arrive on one of the channels in D.

Meaning of Guards in an Environment of other Guards: In defining the a priori semantics for guarded commands, we need to ascertain whether or not the meaning of a *communication command* changes according to the composition of the guards. This is done by defining an auxiliary function $E : G \times \Sigma \rightarrow C? \times C!$ where G is a set of guards and C? and C! are a set of *in* and *out* channels respectively. In other words, the function ,E, yields the possible set of channels that a process could be waiting for input and output in a given state.

$$E[\![G]\!]\sigma = \begin{cases} (\phi,\phi) \textbf{ if } (\exists b;D!!e \in G \ \wedge\ W[\![b]\!]\sigma) \vee (\exists b \in G \ \wedge\ W[\![b]\!]\sigma) \\ (\overline{C}?,\phi) \textbf{ otherwise} \text{ where } \overline{C}? = \cup_i D_i \text{ such that } b_i;D_i??x_i \in G \ \wedge\ W[\![b_i]\!]\sigma \end{cases}$$

That is there cannot be any waiting, if there is an *open pure boolean guard or an open broadcast guard* ; this is reflected in the first part of the equation (i.e. the output of E is (ϕ,ϕ)). As there is no waiting associated with *broadcast command,* the second component is always ϕ. In the sequel, we use $E^?$ and $E^!$ to denote the first and the second component of E respectively.

$M[\![(b,G)]\!]\sigma =$ **if** $W[\![b]\!]\sigma$ **then** $PFC\{(\sigma,\lambda)\}$ **else** $\{(\bot,\lambda)\}$

$$M[\![(D??x,G)]\!]\sigma = \begin{cases} PFC\{(\sigma[v/x],\{<w?,E^?>^t<?,d,v>\})\} \mid d \in D, v \in VAL, t \geq 0\} \\ \qquad\qquad \textbf{if } E[\![\{D??x\} \cup G]\!]\sigma \neq (\phi,\phi) \\ PFC\{(\sigma[v/x],\{<?,d,v>\}) \mid d \in D, v \in VAL\} \textbf{ otherwise} \end{cases}$$

In the case of receive command, an indeterminate waiting is poss if and only if there are no open pure boolean guards and open broadcast guards.

$M[\![(D\,!!e,G)]\!]\sigma = PFC\{(\sigma,\{<!,D,[\![e]\!]\sigma>\})\}$
Note that the semantics of broadcast command is unaffected by the environment.

$M[\![(b;g,G)]\!]\sigma = M^*[\![(g,G)]\!](M[\![(b,G)]\!]\sigma)$

$$M[\![A]\!]\sigma = \textbf{if } \neg \bigvee_{i=1}^{n} W[\![b_i]\!]\sigma \textbf{ then } \{(\bot,\lambda)\} \textbf{ else } \bigcup_{i \in \{1..n\} \wedge G = \cup_{j \neq i} g_j} M[\![(g_i,G);S_i]\!]\sigma$$

The meaning function of the iterative command follows on the same lines as given earlier.

Parallel Composition

Definition 1: Let h_1, h_2 be sequences as defined earlier in processesP_1 and P_2 respectively. Let *cset* be the set of common channels in P_1 and P_2. Then h_1 and h_2 are said to be *consistent* iff the following conditions are satisfied:
a) For every claim of receiving a message $<?,d,v>$ in h_1 at time k there must be a broadcast claim of $<!,D,e>$ at time k in h_2 such that $d \in D \ \wedge \ v=e$. That is,
$<?,D,v> \in h_1[k] \ \wedge \ d \in cset \rightarrow <!,D,v> \in h_2[k] \ \wedge \ d \in D$
Note that we are only looking for a matching output value for an input claim. This means that *the number of values received* on any channel is only a *subsequence* of those values that are sent. In other words, the *subsequence* property captures the order-preserving lossy asynchronous communication medium. Note that the modeling of reliable transmission of messages corresponds to the *prefix* property.

b) If there is a claim of waiting on channel d in h_1 at time k, then there cannot be a broadcast claim on channel d at time k in h_2. That is, $(<w?,D> \in h_1[k] \ \wedge \ <!,D',e> \in h_2[k]) \rightarrow D \cap D' \cap cset = \phi$
This condition ensures the satisfaction of the MAXPAR condition.

c) Symmetric counterparts of (a)-(b).

Definition 2: A pointwise merge of h_1 and h_2, denoted $h_1 \# h_2$, is defined as follows:
$h_1 \# h_2 = h[1] \cdots h[r]$ *where* $r = max(|h_1|, |h_2|)$ and
for $1 \leq k \leq Min(|h_1|, |h_2|)$, $\quad h[k] = h_1[k] \cup h_2[k]$
for $Min(|h_1|, |h_2|) < k \leq r$, $\quad h[k] =$ **if** $|h_1| > |h_2|$ **then** $h_1[k]$ **else** $h_2[k]$

Definition 3: The $merge(h_1, h_2) = h_1 \# h_2$ if $consistent(h_1, h_2)$.

$M[\![S_1 \| S_2]\!]\sigma = \{(\sigma_i \times \sigma_j, h) \mid h \in merge(h_1, h_2), <\sigma_i, h_1> \in M[\![S_i]\!]\pi_i(\sigma), <\sigma_j, h_2> \in M[\![S_j]\!]\pi_j(\sigma)\}$

Network Abstraction: $M[\![[N]]\!]\sigma = \{(\sigma', h \uparrow_t EXT) \mid (\sigma', h) \in M[\![N]\!]\sigma\}$

7. Bounded Buffer Broadcast - immediate initiation

In this scheme, whenever a process executes the broadcast command, it transmits messages only on those channels that contain at least one empty buffer. It does not send or wait to send the message on any of the channels that have no empty buffer at that point. Thus, if the buffers of all the named channels in the broadcast command are full, then the broadcast command is equivalent to a *skip* command.

$M[\![D!!e]\!]\sigma = PFC\{(\sigma, \{<!, D', V[\![e]\!]\sigma, w!, D-D'>\}) \mid D' \subseteq D\}$
That is, the message is sent on those channels that have an empty buffer; that is the reason for taking all the subsets of D (including ϕ). Note that the structure of the a priori semantics is essentially the same as defined for the broadcast command in section 6 except for the structure of the CARs used.

$M[\![D??x]\!]\sigma = PFC\{(\sigma[v/x], \{<w?, D>^t <?, d, v>\}) \mid d \in D, v \in VAL, t \geq 0\}$
Indeterminate waiting can take place iff all the buffers of channels named in D are empty.

In defining the a priori semantics for guarded commands, we need to obtain the meaning of a guard in the presence of other guards in a given state. The definition remains the same as given earlier since even in this case the meaning of a broadcast in the presence of other guards remains unaffected. The difference between the meaning of the broadcast command just discussed and the one discussed in section 6 is that in the former the message is sent only on those channels that have at least one empty buffer whereas in the latter the broadcasting process sends the message on all the channels.

$$E[\![G]\!]\sigma = \begin{cases} (\phi,\phi) \ \textbf{if} \ (\exists b; D!!e \in G \ \wedge \ W[\![b]\!]\sigma) \ \vee \ (\exists b \in G \ \wedge \ W[\![b]\!]\sigma) \\ (\overline{C}?, \phi) \ \textbf{otherwise} \ \text{where} \ \overline{C}? = \cup D_i \ \text{such that} \ b_i; D_i??x_i \varepsilon G \ \wedge \ W[\![b_i]\!]\sigma \end{cases}$$

$M[\![(b,G)]\!]\sigma =$ **if** $W[\![b]\!]\sigma$ **then** $PFC\{(\sigma, \lambda)\}$ **else** $\{(\perp, \lambda)\}$

$$M[\![(D??x,G)]\!]\sigma = \begin{cases} PFC\{(\sigma[v/x],\{<w?,E^?>^t<?,d,v>\}) \mid d \in D, v \in VAL, t \geq 0\} \\ \qquad \textbf{if } E[\![\{D??x\} \cup G]\!]\sigma \neq (\phi,\phi) \\ PFC\{(\sigma[v/x],\{<?,d,v>\}) \mid d \in D, v \in VAL\} \textbf{ otherwise} \end{cases}$$

In the case of receive command an indeterminate waiting is possible if and only if there are no open pure boolean guards and open broadcast guards.

$$M[\![(D!!e,G)]\!]\sigma = PFC\{(\sigma,\{<!,D',[\![e]\!]\sigma,w!,D-D'>\}) \mid D' \subseteq D\}$$

The meaning function for other constructs follows on the same lines as given earlier.

Parallel Composition

Definition 4: Let h_1 and h_2 be any two histories in processes P_1 and P_2 respectively. Let *cset* be the set of common channels in P_1 and P_2. h_1 and h_2 are said to be *consistent* iff:
a) The sequence of messages received on $d \in cset$ in h_1 is a subsequence of the sequence of messages broadcast on d in h_2. That is, $d \in cset \rightarrow h_1 \uparrow d? \underset{sub}{\leq} h_2 \uparrow d!$

b) A waiting to receive claim made at time k in h_1 on channel $d \in cset$, is valid only if the buffer associated with that channel d is completely empty at time k. That is,
$<w?,D> \in h_1[k] \wedge d \in D \wedge d \in cset \rightarrow |h_2^k \uparrow d!| - |h_1^k \uparrow d?| = 0$

c) For every receive claim made at time k in h_1 on channel d, there must be at least one nonempty buffer associated with that channel . That is, $<?,d,v> \in h_1[k] \wedge d \in cset \rightarrow |h_2{}^k \uparrow d!| - |h_1{}^k \uparrow d?| \geq 0$
As there is a possibility of receive and a broadcast claim at time k in h_1 and h_2 the condition reduces to ≥ 0 rather than ≥ 1.

d) A priori a process cannot determine whether the buffer associated with the channel is empty (for sending a message). Hence, in the a priori semantics, we consider all subsets of D where D is the set of channels over which it is required to broadcast a value. While composing the processes, the claim needs to be verified. The condition of the availability of at least one empty buffer on channel d? corresponds to saying that the number of messages sent on d minus the number of messages received on that channel is greater than the size (denoted by |*buffer(d)*|) of the buffer for that channel. That is,
$<!,*,*,w!,D> \in h_2[k] \wedge d \in D \wedge d \in cset \rightarrow |h_2{}^k \uparrow d!>| - |h_1{}^k \uparrow d?| = |buffer(d)|$
where * is used denote any arbitrary value.

e) The above condition checks for the waiting claims. The following condition corresponds to checking for the broadcasting claim at time k. That is, a broadcast claim made at time k in h_2 on d, is valid provided that the buffer associated with that channel is not completely full at time k.
$<!,d,v> \in h_2[k] \wedge d \in cset \rightarrow 0 \leq |h_2{}^k \uparrow d!| - |h_1{}^k \uparrow d?| \leq |buffer(d)|$

f) The symmetric counterparts of (a)-(e).

$$M[\![S_i \| S_j]\!]\sigma = \{(\sigma_i \times \sigma_j, h) \mid h \in merge(h_1,h_2) <\sigma_i,h_1> \in M[\![S_i]\!]\pi_i(\sigma), <\sigma_j,h_2> \in M[\![S_j]\!]\pi_j(\sigma)\}$$

8. Bounded Buffer Broadcast- delayed (possibly) initiation

In this scheme, whenever a process executes a broadcast command, it transmits the message on a subset of channels(including empty) that have at least one free buffer at that point of time (i.e. immediately), and then waits for the other channels to become empty. This is repeated till the message is broadcast over all the channels of D. In order to capture the notion of waiting associated with the transmission of a message when the buffer on the channel is full and with the reception of a message when the buffer is empty, we introduce the waiting assumption $<w!,D>$ for broadcast and the waiting assumption $<w?,D>$ for the receive. Thus, the a priori semantics for broadcast and receive commands may be formulated as follows.

Broadcast: The effect of the command is to send the message on those channels that have at least one empty buffer. Thus, the process would wait if there is no channel with an empty buffer. Let us assume the broadcast is completed in m stages, i.e. the process sends the message to $D_i \subseteq D$ at each stage. As the message is not duplicated on any channel, it should be clear that $D_i \cap D_j = \phi$ for $i \neq j$. If the process is trying to send at time k, it can wait without sending any message over any channel provided all the channels are full ,i.e. $D_0 = \phi$.

The sets D_is have property *P1* if they satisfy the following properties:

$D_0 \subseteq D$, $D_{r+1} = D - \bigcup_{i=1}^{r} D_i$ (subsets of D over which the communication is not yet successful), such that $D - \bigcup_{i=1}^{m} D_i = \phi$ (i.e., the message is sent over all the channels). .... **(P1)**

Construct the sequence $\prod_{m=0}^{r} \tau_m \eta_m$ where $\tau_m = \langle w!, D - \bigcup_{i=0}^{m} D_i \rangle$ and $\eta_m = \langle !, D_{m+1}, [\![e]\!]\sigma, w!, D - \bigcup_{i=0}^{m+1} D_i \rangle$, such that $\tau_0 = \lambda$ if $D_0 \neq \phi$; i.e. there can be waiting only if the process has to wait on all of them- i.e. initially waiting is possible only if $D_0 = \phi$. In the above expression, the product denoted by Π denotes the product of the terms - the implicit operator between the terms denoting concatenation. In other words , τ_m and η_m represent the waiting and the successful transmission on at least one channel at stage m. Obviously, the sets $D_0, \ldots, D_m$ cannot be determined a priori. Hence, we have to consider for all $D_i \subseteq D$ satisfying the above property (note that there can be at most n stages where n = |D| corresponding to sending a message to one process(channel) at a time).

$$M[\![D!!e]\!]\sigma = PFC\{(\sigma, \{\prod_{m=0}^{r} \langle\tau_m\rangle^t \eta_m\}) \mid \forall\, t \geq 0, \forall D_i \subseteq D \textit{ satisfying } \mathbf{P1}\}$$

Receive: $M[\![D??x]\!]\sigma = PFC\{(\sigma[v/x], \{\langle w?, D\rangle^t \langle ?, d, v\rangle\}) \mid d \in D, v \in VAL \wedge t \geq 0\}$
The semantics is essentially the same as given earlier.

In defining the a priori semantics for guarded commands, we need to obtain the meaning of a guard in the presence of other guards in a given state. In the category under consideration, there is a waiting associated with the broadcast command. Thus, the meaning of the broadcast and the receive commands change in the environment of other guards. Thus, before any of the guards is chosen there is a possibility of waiting on *receive* guards and *broadcast* guards. This necessitates the use of the CAR structure <w!,D,w?,D'> described earlier(where D is the set of channels on which the process is waiting for broadcast and D' is the set of channels on which the process is waiting for receiving a message). Thus, at an alternative command before any guard starts executing, there is a possibility of waiting for input and output. But, once the communication command starts then it essentially behaves as the corresponding communication command.

$E[\![G]\!]\sigma = (E^?, E^!)$ where $E^? =$ **if** $\exists b \in G \wedge W[\![b]\!]\sigma$ **then** ϕ **else** $\{D_i \mid b_i; D_i??x_i \in G \wedge W[\![b_i]\!]\sigma\}$
$E^! =$ **if** $\exists b \in G : W[\![b]\!]\sigma$ **then** ϕ **else** $\{D_i \mid b_i; D_i!!e_i \in G \wedge W[\![b_i]\!]\sigma\}$
In other words, $E^?$ and $E^!$ will be empty only if there is a pure open boolean guard. As both the components could be nonempty, it follows that the process can be simultaneously waiting for a buffer to be filled and waiting for sending a message.

$M[\![(b,G)]\!]\sigma =$ **if** $V[\![b]\!]\sigma$ **then** $PFC\{(\sigma,\lambda)\}$ **else** $\{(\perp,\lambda)\}$

$$M[\![(D??x,G)]\!]\sigma = \begin{cases} PFC\{(\sigma[v/x], \{\langle w!, E^!, w?, E^?\rangle^t \langle ?, d, v\rangle\}) \mid d \in D, v \in VAL, t \geq 0\} \\ \qquad \textbf{if } E[\![\{D??x\} \cap G]\!]\sigma \neq (\phi,\phi) \\ PFC\{(\sigma[v/x], \{\langle ?, d, v\rangle\}) \mid d \in D, v \in VAL\} \textbf{ otherwise} \end{cases}$$

Note that whenever $E[\![\{g\} \cup G]\!] = (\phi,\phi)$ the semantics is the same as the pure receive command; in the other case there is an extra prefix, namely $\langle w!, E^!, w?, E^?\rangle^t$ which reflects the initial waiting.

$$M[\![(D!!e,G)]\!]\sigma = \textbf{if } E[\![\{D!!e\} \cup G]\!] = (\phi,\phi) \textbf{ then } PFC\{(\sigma, \{\prod_{m=0}^{r} \langle\tau_m\rangle^t \eta_m\}) \mid \forall\, t \geq 0, \forall D_i \subseteq D \textit{ satisfying } \mathbf{P1}\}$$
$$\textbf{else } PFC\{(\sigma, \langle w!, E^!, w?, E^?\rangle^t \prod_{m=0}^{r} \langle\tau_m\rangle^t \eta_m) \mid \forall\, t \geq 0, \forall D_i \subseteq D \textit{ satisfying } \mathbf{P1}\}$$

Again, note that the difference comes in the prefix sequence of CARs that reflects the possible waiting before any communication guard is selected. The treatment of the other constructs follows as described earlier.

Parallel Composition

Definition 5: Let *cset* be the set of common channels between processes S_1 and S_2. h_1 and h_2 belonging to two distinct processes S_1 and S_2 are said to be *consistent* iff:

a) The sequence of messages received on channel d in h_1 is a prefix of the sequence of the messages broadcast on that channel in h_2. Note that the prefix property models the order preserving reliable transmission of the communication medium. Thus, $d \in cset \rightarrow h_1 \uparrow d? \leq h_2 \uparrow d!$

b) A waiting assumption to receive on channel d in h_1 at time k is valid only if the pending messages at time k on d is empty. The buffer associated with channel d is completely empty at that time. Thus,
$\forall\, d \,\varepsilon\, D\, (<w?,D> \in h_1[k] \;\wedge\; d \in cset \rightarrow |h_2^k \uparrow d!| - |h_1^k \uparrow d?| = 0)$

c) For every value received assumption on a channel d at time k in h_1, the buffer associated with that channel in h_2 is not empty. That is, $<?,d,v> \in h_1[k] \;\wedge\; d \in cset \rightarrow |h_2^k \uparrow d!| - |h_1^k \uparrow d?>| \geq 0$
Note that the condition essentially implies that for a value to be received at time k on channel d, it must be the case that the buffer associated with d at that point should be nonempty (note we are considering the value received and sent at time k also).

d) A further condition that needs to be satisfied corresponds to an assumption made by each process when it attempts to broadcast a message. Since a process cannot locally determine which of the channels are empty, it makes the local assumption that every subset of the channels is full and waits on those channels. Thus, we need to reject computations involving unnecessary waiting assumptions when the channels are empty. That is, if there is a waiting to send claim on channel d in h_2, then the claim is valid only if the number of messages sent on that channel minus the number of messages received on that channel is greater than the size of the buffer for that channel. Thus,
$(<w!,D> \in h_2[k] \;\vee\; <!,*,*,w!,D> \in h_2[k]) \;\wedge\; d \in D \;\wedge\; d \in cset \rightarrow |h_2^k \uparrow d!| - |h_1^k \uparrow d?| = |buffer(d)|$

e) For every successful broadcast assumption made at a time k in h_2 on channel d, there is at least one nonempty buffer with respect to d.
$<!,d,v> \in h_2[k] \;\wedge\; d \in cset \rightarrow 0 \leq |h_2^k \uparrow d!| - |h_1^k \uparrow d?| \leq |buffer(d)|$

f) the symmetric counterparts of (a)-(e).

$M[\![S_i \,||\, S_j]\!]\sigma = \{(\sigma_i \times \sigma_j, h) \mid h \in merge(h_1,h_2) <\sigma_i,h_1> \in M[\![S_i]\!]\pi_i(\sigma), <\sigma_j,h_2> \in M[\![S_j]\!]\pi_j(\sigma)\}$

9. Atomic Broadcast

In this case, the message must be sent to all the destinations in only one stage. That is, the initiation is delayed till it is possible to send the message over all the designated channels; i.e., sending of the messages is delayed if the process is waiting on any one of the channels. For describing atomic broadcast, the message structure considered is insufficient for obtaining a compositional semantics. While merging a subset of process, we cannot determine the time associated with a waiting to broadcast assumption of the process on a channel as the time of sending depends on when all the channels named in the atomic broadcast become available. We can determine the time under the maximum parallelism requirement, if we merge all the histories of the processes together. Since associative merge of histories is central to compositional semantics, we have to record in the a priori semantics the following additional information: when a process makes the assumption of waiting on a subset of channels, then it does so with the knowledge that the buffers associated with rest of the channels (other than that subset) are free. Maintaining that information requires a change in the message structure. The new message structure is given below:

- $<w!,D',E!,D''$ - denoting a claim by a process that it is waiting to broadcast a message on the set of channels in D' and a simultaneous assumption that at that time the set of channels in D'' have free buffers associated with them.

Let **P1** be as defined in section 8 and construct the sequence $\prod_{m=0}^{r} \tau_m$, where $\tau_m = <w!, D - \bigcup_{i=0}^{m} D_i, E!, \bigcup_{i=0}^{m} D_i>$

Broadcast: $M[\![D\,!!e]\!]\sigma = PFC\,\{(\sigma, \{\prod_{m=0}^{r} \tau_m{}' <!,D,[\![e]\!]\sigma>\}) \mid \forall t \geq 0, \forall D_i \sqsubseteq D \text{ satisfying } P1\}$

Since the sets D_i (i=0..m) cannot be determined a priori, we have to consider all subsets satisfying **(P1)**.

Receive: $M[\![D??x]\!]\sigma = PFC\,\{(\sigma[v/x], \{<w?,D>^t<?,d,v>\}) \mid d \in D, v \in VAL \;\wedge\; t \geq 0\}$

Semantic equations for the other commands can be obtained on the same lines as given sections 7 and 8.

Parallel Composition: In this case, an additional check is needed to ensure the assumption of free buffers on a channel made at a given point in time is indeed correct.

Definition 6: h_1 and h_2 of two processes S_1 and S_2 with *cset* as the common set of channels are said to be consistent iff:
(i) conditions (a) - (e) are as in section 8.
(ii) A free buffer assumption (E!) on channel d in h_1 at the time point k can be made only if the buffer associated with d is free.
$\forall d \in D(<E!,D> \in h_1[k] \;\wedge\; d \in cset \rightarrow |h_1^k \uparrow d!| - |h_2^k \uparrow d?| < |buffer(d)|)$

(iii) symmetric counterparts of (i)-(ii).

Other details of parallel composition remain the same as in section 8.

10. A Simple Example

For lack of space, we illustrate the semantics of the class of *bounded buffer broadcast --immediate initiation* with the following program assuming unit buffer with channel D.

P ::= skip;skip ; $\{D\}$!! e ; skip; skip ; skip ;skip; C_1 ::= D ?? x ; skip; skip;

$M[\![P]\!]\sigma \equiv PFC\{\ (\sigma_p,\ \{\square^2<!,\{D\},e,w!,\varnothing>\square^4\}),\ (\sigma_p,\ \{\square^2<!,\varnothing,e,w!,\{D\}>\square^4\})\ \}$

$M[\![C_1]\!]\sigma \equiv PFC\ \{\ (\ \sigma_{c1}[v/x]\ ,\ \{\ <w?,D>^t<?,D,v>\square^2\}) \mid \forall\ v\ \in\ \text{Domain}(x)\ ,\ t \geq 0\ \}$

$M[\![P \parallel C_1]\!]\sigma \equiv PFC\{(\sigma_p + \sigma_{c1}[e/x], \{\{<w?,D>,\square\}\{<w?,D>,\square\}\{<!,\{D\},e,w,\varnothing>,<?,D,e>\}\{\square\}\{\square\}\{\square\}\{\square\}\})\}$

However, if we consider C_2 ::= skip; skip; skip; D ?? y ; skip; skip; then $M[\![P \parallel C_2]\!]\sigma$ will consist of prefix-closures of $(\perp, h)$ where $h \equiv \{\square\}\ \{\square\}\ \{\ \square\ ,\ <!,\ \varnothing\ ,\ e\ ,\ w!\ ,\ \{D\}>\ \}\ \{<w?,D>^t\}$ for any t -- reflecting, infinite waiting (deadlock).

In [SNP87], we illustrate the semantics with realistic examples and formally establish for each of the categories that M is continuous, the merging of the histories is associative and the composition of the processes is compositional.

11. Discussion

In this paper, we presented a syntax-directed compositional denotational semantics for nondeterministic broadcast networks. The semantics uses a simple domain of prefix closed state history pairs. Broadcast networks have been characterized based on buffer availability, reliable/lossy transmission, and immediate/delayed initiation. One of the central aspects that can be inferred from the semantics is that the distinguishing features of the various broadcast schemes are observable only in real-time models. Conversely, it also reflects on the inherent suitability of broadcast schemes for real-time programming.

In the language considered in the paper, we have not explicitly included the delay commands or guards. It should be clear, from the development of the semantics of guards in the environment of other guards, that explicit real-time constructs such as delay commands or guards can be incorporated in a straight-forward manner.

Though we are not distinguishing deadlock and divergence as in [HGD87], our semantics is not fully abstract. Currently, we are investigating fully abstract models[HGD87] for broadcast networks. It may be noted that we come across two types of deadlock in broadcast networks:- the first is due to nonavailability of empty buffers for sending (depositing) messages and the second is due to the nonavailability of messages in the buffers. It is interesting to extend the semantics that distinguishes deadlock from divergence and compare the resulting one with the linear history semantics of [FLP84] from the point of view of deadlock behaviour. From the merging functions discussed, a compositional network specification can be obtained using predicate logic and the set of histories of the network on the lines of [Hoa85]. Using this method, invariant properties of the network can be specified. We are currently investigating the extension of this method to include liveness properties using the *quiescent* states as in [MiC81]. Further, we are working towards a proof system for the broadcast networks on the lines of [ZDB85]. We hope that such methods will enable us to provide methods for the specification and verification of broadcast protocols[SeA83, Wal80].

Based on the semantics presented in the paper, it is possible to obtain a nice formal model for Ethernet like bus structures. Some important aspects of such bus structures are:- i) The broadcast is delayed until bus is free, ii) The message broadcast on the bus may collide with another broadcast attempted simultaneously, iii) Collisions are detectable, and iv) A collision free broadcast of a message may be lost if no process is willing to receive. From the unbuffered broadcast semantics, the model satisfying only (ii) is immediate. For collision free broadcast, it is possible to treat the bus as a *restricted monitor*, or a process employing both synchronous and asynchronous communication. The formal characterization of Ethernet structures corresponding to the above four properties is currently under preparation.

12. References

[Cha84] Chang Jo-Mei, Simplifying Distributed Database Systems Design by Using a Broadcast Network, *ACM Conference on Data Bases*, , 1984, pp. 223-233.

[FFe73] Farber,D, Feldman,J and et. al, The Distributed Computing System, *IEEE Computer Society International Conference*, , Feb 1973, pp. 31-34.

[FLP84] Francez,N, Lehman,D and Pnueli,A, A Linear History Semantics for Languages for Distributed Programming, *TCS 32*, (1984), pp. 25-46.

[Geh84] Gehani,N, Broadcast Sequential Processes, *IEEE Trans. on Software Eng. 10*, 4 (1984), pp. 343-351.

[Hoa78] Hoare,C.A.R, Communicating Sequential Processes, *Comm. ACM 20*, 8 (1978), pp. 666-676.

[Hoa85] Hoare,C.A.R, Communicating Sequential Processes, *Prentice-Hall International*, , 1985.

[HGD87] Huizing,C, Gerth,R and De Roever,W.P, Full Abtraction of a Real-Time Denotational Semantics for an *Occam-* like Language, *POPLS*, , 1987.

[Jon85] Jonsson Bengt, A Model and Proof System for Asynchronous Networks, *PODC*, , 1985, pp. 49-58.

[KSD85] Koymans,R, Shyamasundar,R.K and DeRoever,W.P.,Gerth,R.,ArunKumar,S., Compositional Denotational Semantics for Real-Time Distributed Computing, *LNCS 193*, , 1985.

[KSD86] Koymans,R, Shyamasundar,R.K and DeRoever,W.P.,Gerth,R.,ArunKumar,S., Compositional Denotational Semantics for Real-Time Distributed Computing, *Information and Control (to appear)*, , 1986.

[MeB76] Metcalfe,R.M and Boggs,D.R, Ethernet: Distributed Packet Switching for Local Computer Networks, *Comm. ACM 19*, 7 (Jul 1976), pp. 395-404.

[MiC81] Misra,J and Chandy,K.M, Proofs of Networks of Processes, *IEEE Trans. on Software Eng. 7*, 4 (1981), pp. 417-426.

[SaM81] Salwicki,A and Muldner,T, On the Algorithmic Properties of Concurrent Programs, *LNCS 125*, (1981), , Springer-Verlag.

[ScS84] Schlichting,R.D and Schneider,F.B, Using Message Passing for Distributed Programming: Proof Rules and Disciplines, *ACM Trans. Prog. Lang. and Systems 6*, 3 (1984), pp. 402-431.

[Sch82] Schneider,F.B, Synchronization in Distributed Programs, *ACM Trans. Prog. Lang. and Systems 4*, 2 (Apr 1982), pp. 179-195.

[SeA83] Segall,A and Awerbuch,B, A reliable Broadcast Protocol, *IEEETCommn 31*, 7 (1983), pp. 896-901.

[SNP87] Shyamasundar,R.K., Narayana,K.T. and Pitassi,T, Semantics of Nondeterministic Asynchronous Broadcast Networks, *Technical Report, Pennysvania State University*, , March 1987.

[Wal80] Wall,D.W, in *Mechanisms for Broadcasts and Selective Broadcasts*, Computer Science Department, Stanford University, 1980. Ph.D Thesis.

[ZDB85] Zwiers,J, De Roever,W.P and Boas Peter Van Emde, Compositionality and Concurrent Networks Soundness and Completeness of a Proof System, *ICALP 1985 LNCS 194*, (1985), pp. 509-519, Springer Verlag.