Broadcast and aggregation in BBC

Hans Hüttel
Department of Computer Science, Aalborg University, Denmark
hans@cs.aau.dk

Nuno Pratas
Department of Electronic Systems, Aalborg University, Denmark
nup@es.aau.dk

In distributed systems, where multi-party communication is essential, two communication paradigms are ever present: (1) one-to-many, commonly denoted as *broadcast*; and (2) many-to-one denoted as *aggregation* or *collection*.

In this paper we present the BBC process calculus, which inherently models the *broadcast* and *aggregation* communication modes. We then apply this process calculus to reason on hierarchical network structure and provide examples on its expressive power.

1 Introduction

In the setting of distributed systems, interprocess communication will often be in the form of multi-party communication. Here there are two dual paradigms: that of one-to-many communication, commonly denoted as *broadcast*, and that of many-to-one, denoted as *aggregation* or *collection*.

Many-to-one communication is a commonly occurring phenomenon and often occurs in settings with intricate network topologies. As an example consider the case of a smart grid with thousands of meters (such as electricity, water, heating, etc ...) that report their current measurements. A central server somewhere will collect these data; however, the server will usually not be directly reachable from the meters – there will be several intermediate hops, each one facilitated by a relay. If the underlying protocol of the network is correct, we would expect this complicated network to be semantically equivalent to a simple planar network, where each node is at one-hop distance from the server.

There has already been considerable interest in understanding the semantic foundations of one-to-many communication in the process calculus community, starting with the work by Prasad [7] on a broadcast version of CCS. Later, Ene and Muntean extended the notion to a broadcast version of the π-calculus [1] and proved that this notion is strictly more expressive than standard π-calculus synchronization.

Other process calculi with a notion of broadcast arise in the search for behavioural models of protocols for wireless networks. Singh et al. describe a process calculus with localities [10], and Kouzapas and Philippou [4] introduce another process calculus with a notion of localities, whose configuration can evolve dynamically.

In this paper we introduce a process calculus BBC that has both forms of communication. For both many-to-one and one-to-many communication, it is often a natural assumption that communication is *bounded*; this reflects two distinct aspects of the limitations of a medium. In the case of broadcast, the bound limits the number of possible recipients of a message. In the case of collection, the bound limits the number of messages that can be received. For this reason, BBC uses a notion of bounded broadcast and collection. Moreover, the syntax of the calculus introduces an explicit notion of connectivity that

S. J. Gay and J. Alglave (Eds.): Programming Language Approaches to Concurrency- and Communication-Centric Software (PLACES 2015).
EPTCS 203, 2016, pp. 15–28, doi:10.4204/EPTCS.203.2

$$M ::= x \mid S \mid (M_1,\ldots,M_k) \mid g(E) \mid f(M)$$

$$P ::= a(\lambda\vec{x}M)P_1 \mid a[\lambda\vec{x}M.S]P_1 \mid \overline{a}\langle M\rangle.P_1 \mid (\nu x:\beta_x)P_1 \mid [M_1 = M_2]P_1 \mid [M_1 \neq M_2]P_1 \mid (P_1 \mid P_2)$$
$$\mid (P_1 + P_2) \mid \mathbf{0} \mid A\left(\vec{M}\right)$$

$$E ::= \{M_1,\ldots,M_k\} \mid S$$

$$N ::= l[P] \mid (N_1 \mid N_2) \mid (\nu x:b)N \mid l \triangleright m$$

Table 1: Formation rules for BBC

makes it possible to represent a communication topology directly. By using a proof technique introduced by Palamidessi [6] we show that even a version of BBC that only uses collection is more expressive than the π-calculus.

The remainder of our paper is organized as follows. In Section 2 we introduce the syntax of BBC, while Section 3 gives a reduction semantics. In Section 4 we introduce a notion of barbed bisimilarity, and in Section 5 we use this notion to prove the correctness of a protocol that uses both broadcast and collection. In Section 6 we outline a simple type system for BBC in which channels can be distinguished as being used for broadcast or for collection. Finally, in Section 7 we show that even a version of BBC that only uses collection is more expressive than the π-calculus.

2 The syntax of BBC

2.1 The syntactic categories

A central notion in BBC is that of *names*. As in the distributed π-calculus [8], processes reside at named sites, called *locations*, and use named *channels* for communication. We assume that names are taken from a countably infinite set **Names**. In general, we denote names of channels by $a,b,c\ldots$, names of locations by $l,m,n\ldots$ and if nothing is assumed about the usage of the name we denote them by $x,y,z\ldots$.

We let $M \in$ **Msg** range over the set of messages, let P range over the set of processes and let N range over the set of networks. Since a collecting input (defined below) can receive a multiset of messages, each coming from a distinct sender, we also consider *multiset expressions* E and multiset variables S that can be instantiated to multiset expressions. The formation rules defining the syntactic categories of BBC are given below.

2.2 Messages and patterns

For ordinary expressions we assume a collection of term constructors ranged over by f, that build messages out of other messages. Moreover, we assume the existence of a collection of multiset selectors ranged over by g; these can be used to build messages out of multisets.

If a channel is to be chosen among a collection of candidate channels, we can use a multiset selector to describe this.

Example 1. *An example of a multiset selector is the function find-a that intuitively returns the name a if this name occurs as the first component in a multiset of pairs of names and the name $k \neq a$ and is defined by find-$a(S) = a$ if $(a,x) \in S$ for some x and k otherwise.*

Practical examples of interest are the election of a common channel in a ad-hoc network; selection of a channel for cooperative sensing or for communication within an ad-hoc cluster.

An important notion is that of an *input pattern* which is of the form $(\lambda \vec{x} M)$, where the variable names in $\vec{x}$ are distinct and occur free in M. A message O matches this pattern, if it can be obtained from it through substitution.

More formally, a *term substitution* is a finite function $\theta : \mathbf{Names} \to \mathbf{Msg}$. The substitution can also be written as a list of bindings $\theta = [x_1 \mapsto M_1, \ldots, x_k \mapsto M_k]$. The action of θ on an arbitrary message or multiset expression is defined in the expected way. M' is said to match $(\lambda \vec{x} M)$ with θ if for a substitution θ with $\mathrm{dom}(\theta) = \vec{x}\ M' = M\theta$ is true.

2.3 Processes

In a collecting communication setting, the receiver can make no assumption about the number of messages that will be received, nor on the order in which they are received. Moreover, we cannot assume that a message that has arrived will only occur once among the messages received during a single collecting communication. We shall therefore think of a collecting input as receiving a multiset of messages.

There are two kinds of input prefixes in BBC:

- The *broadcast input* $a(\lambda \vec{x} M)P_1$ in which a single term matching the pattern $(\lambda \vec{x} M)$ is received on the channel a. The pattern variables in $\vec{x}$ are bound in P_1 and get instantiated with the appropriate subterms that correspond to the pattern.
- The *collection input* $a[\lambda \vec{x} M.S]P_1$ in which a non-empty multiset of terms $\{M_1, \ldots, M_K\}$ each of which matches the pattern $[\lambda \vec{x} M]$ is received on the channel a. Note that in this case the scope of the pattern variables in $\vec{x}$ *does not extend* to P_1. Following the input, the multiset variable S is instantiated to the multiset $\{M_1, \ldots, M_K\}$.

In a *restriction* $(\nu x : \beta_x)P_1$ the notion of bounded communication is made explicit: we declare the name x to be private within P_1 and to have bound β_x, where $\beta_x : \mathbf{Names} \to \mathbb{N}$ is a function such that for any location name m we have that $\beta_x(m) = k$, if it is the case that for a process located at m there are at most k senders that are able to send a message to it using the channel x.

The remaining process constructs are standard. The *output* process $\overline{a}\langle M \rangle.P_1$ sends out the message M on the channel named a and then continues as P_1. *Match* $[M_1 = M_2]P_1$ and *mismatch* $[M_1 \neq M_2]P_1$ proceed as P_1 if M_1 and M_2 are equal, respectively distinct. *Parallel composition* $P_1 \mid P_2$ runs the components P_1 and P_2 in parallel. *Nondeterministic choice* $P_1 + P_2$ can proceed as either P_1 or P_2; *inaction* $\mathbf{0}$, has no behaviour.

Finally, we allow agent identifiers $A(\vec{M})$ parameterized by a sequence of messages; an identifier must be defined using an equation of the form $A(\vec{x}) \stackrel{\mathrm{def}}{=} P$. The only names free in P must be the parameters found in $\vec{x}$, that is $\mathrm{fn}(P) \subseteq \vec{x}$. Definitions of this form can be recursive, with occurrences of $A(\vec{x})$ (with names in $\vec{x}$ instantiated by concrete messages) occurring within P.

Restriction and broadcast input are name binders; for a process P, the sets of *free names* $\mathrm{fn}(P)$ and *bound names* $\mathrm{bn}(P)$ of P are defined as expected. For collection input we define

$$\mathrm{bn}(a[\lambda \vec{x} M\, S]P_1) = \mathrm{bn}(P_1)$$

Replication, denoted as $!\,P$, is a derived construct in BBC; a replicated process $!\,P$ is expressed by the agent identifier A_P whose defining equation is $A_P = P \mid A_P$ and should therefore be thought of as an unbounded supply of parallel copies of P.

2.4 Networks

As in the distributed π-calculus [8] a parallel composition of located processes is called a *network*. According to the formation rules, a *network* N can be a process P running at location l, which is denoted as $l[P]$. We also allow parallel composition $P_1 \mid P_2$ and restriction $(\nu x)N$ at network level. Moreover, the *neighbourhood predicate* $l \triangleright k$ denotes that location l is close to k. For the neighbourhood predicate, parallel composition is thought of as logical conjunction. So, if l and k are close to each other, then $l \bowtie m$ can be written instead of $l \triangleright m | m \triangleright l$.

For any term M, the set of free names $\mathrm{fn}(M)$ is defined in the standard way.

The usual notions of α-conversion also apply here. We write $P_1 \equiv_\alpha P_2$ if P_1 can be obtained from P_2 by renaming bound names in P_2, and likewise we write $N_1 \equiv_\alpha N_2$ if N_1 can be obtained from N_2 by renaming bound names in N_2.

3 The semantics of BBC

We now describe a reduction semantics of BBC.

3.1 Evaluation of message terms

In our semantics, we rely on an evaluation relation $\rightsquigarrow$ defined for both message terms and multiset expressions.

Example 2. *Assume that our set of function symbols contains the projection function which extracts the first coordinate of a pair of names and that this is the only function symbol that can lead to evaluation. The evaluation relation can then be defined by the axiom*

$$\mathit{first}(x,y) \rightsquigarrow x$$

and the rules

$$\frac{M \rightsquigarrow M'}{\{\!|M,\ldots|\!\} \rightsquigarrow \{\!|M',\ldots|\!\}} \qquad \frac{M \rightsquigarrow M'}{f(M) \rightsquigarrow f(M')} \qquad \frac{E \rightsquigarrow E'}{g(E) \rightsquigarrow g(E')}$$

A message term M is *normal* if $M \not\rightsquigarrow M_1$ for any M_1.

3.2 Structural congruence and normal form

Structural congruence $\equiv$ is defined for both processes and networks; two processes (or networks) are related, if they are identical up to simple structural modifications such as the ordering of parallel components. The relation is defined as the least equivalence relation satisfying the proof rules and axioms of Tables 2 and 3. In the rules defining structural congruence, the parallel composition is commutative and associative both at the level of processes (rules (P-COM) and (P-AS)) and of networks (rules (N-COM) and (N-AS)). This justifies the use of iterated parallel composition $\prod_{i\in I} P_i$ introduced earlier.

(P-OUT)	$\dfrac{M \rightsquigarrow M'}{\overline{a}\langle M\rangle.P \equiv \overline{a}\langle M'\rangle.P}$	(P-COM)	$P_1 \mid P_2 \equiv P_2 \mid P_1$
(P-AS)	$(P_1 \mid P_2) \mid P_3 \equiv P_1 \mid (P_2 \mid P_3)$	(P-COM-PLUS)	$P_1 + P_2 \equiv P_2 + P_1$
(P-AS-PLUS)	$(P_1 + P_2) + P_3 \equiv P_1 + (P_2 + P_3)$	(P-NIL)	$P \mid \mathbf{0} \equiv P$
(P-EXT)	$(\nu x)(P_1 \mid P_2) \equiv (\nu x)P_1 \mid P_2$ if $x \notin \mathrm{fn}(P_2)$	(P-NEW)	$(\nu x)(\nu y)P \equiv (\nu y)(\nu x)P$
(P-EQ-1)	$[M = M]P \equiv P$	(P-EQ-2)	$[M \neq N]P \equiv P$
(P-EQ-3)	$\dfrac{M \rightsquigarrow M'}{[M \neq N]P \equiv [M' = N]P}$	(P-EQ-4)	$\dfrac{M \rightsquigarrow M'}{[M = N]P \equiv [M' = N]P}$
(P-EQ-5)	$[M \neq N]P \equiv [N \neq M]P$	(P-EQ-6)	$[M = N]P \equiv [N = M]P$
(P-ALPHA)	$\dfrac{P_1 \equiv_\alpha P_2}{P_1 \equiv P_2}$	(P-AG)	$\dfrac{A(\vec{x}) \stackrel{\mathrm{def}}{=} P}{A(\vec{M}) \equiv P\theta}$
(P-PAR)	$\dfrac{P_1 \equiv P_2}{P_1 \mid Q \equiv P_2 \mid Q}$	(P-RES)	$\dfrac{P_1 \equiv P_2}{(\nu x)P_1 \equiv (\nu x)P_2}$

Table 2: Structural congruence for processes

The restriction axioms describe the scope rules of name restrictions. The scope extension axioms (P-EXT) and (N-EXT) express that the scope of an extension can be safely extended to cover another parallel component if the restricted name does not appear free in this component. The exchange axioms (P-NEW) and (N-NEW) allow us to exchange the order of restrictions.

The agent axiom (P-AG) tells us that an instantiated agent identifier $A(\vec{n})$ should be seen as the same as the right-hand side of its defining equation $A(\vec{x}) = P$ instantiated by the substitution θ that maps the names in $\vec{x}$ component-wise to those of $\vec{n}$.

Finally, the rules (P-PAR) and (P-RES) express that structural congruence is indeed a congruence relation for parallel composition and restriction.

By using the laws of structural congruence, any network can be rewritten to normal form. Informally, a network is in normal form if it consists of the total neighbourhood information as one parallel component and the location information as the other. In the following, the notation $\prod_{i\in I} P_i$ is used to denote iterated parallel composition, i.e. the parallel composition of $P_1, \ldots, P_k$, where $I = \{1, \ldots, k\}$.

Definition 1. *A network N is on* normal form *if $N = (\nu\vec{m})(\mathscr{C} \mid \prod_{k\in K} l_k[P_k])$ where $\mathscr{C} = \prod_{i\in I}\prod_{j\in I\setminus\{i\}} l_i \triangleright m_j$.*

We write $l \triangleright m \in \mathscr{C}$ if $\mathscr{C} \equiv l \triangleright m \mid \mathscr{C}'$ for some $\mathscr{C}'$. It is easy do see that any network N can be rewritten into normal form.

Theorem 1. *For any network N, there exists a network N_1 such that $N_1 \equiv N$ and N_1 is on normal form.*

Proof. Induction in the structure of N. The proof is similar to that for the π-calculus [9]; the idea is to use the scope extension axioms to push out restrictions while α-converting bound names whenever needed. □

(N-CONG)	$\dfrac{P_1 \equiv P_2}{l[P_1] \equiv l[P_2]}$	(N-COM)	$N_1 \mid N_2 \equiv N_2 \mid N_1$
(N-AS)	$(N_1 \mid N_2) \mid N_3 \equiv N_1 \mid (N_2 \mid N_3)$	(N-NIL)	$N \mid \mathbf{0} \equiv N$
(N-EXT)	$(\nu x)(N_1 \mid N_2) \equiv (\nu x)N_1 \mid N_2$ if $x \notin \mathrm{fn}(P_2)$	(N-NEW)	$(\nu x)(\nu y)N \equiv (\nu y)(\nu x)N$
(N-LOC)	$l[P \mid Q] \equiv l[P] \mid l[Q]$	(N-EQ)	$(\nu n)l[P] \equiv l[(\nu n)P]$ if $l \neq n$

$$\text{(N-ALPHA)}\quad \frac{N_1 \equiv_\alpha N_2}{P_1 \equiv N_2} \qquad \text{(N-PAR)}\quad \frac{N_1 \equiv N_2}{P_1 \mid N_3 \equiv N_2 \equiv N_3}$$

$$\text{(N-RES)}\quad \frac{N_1 \equiv N_2}{(\nu x)N_1 \equiv (\nu x)N_2}$$

Table 3: Structural congruence for networks

3.3 The reduction relation

In our reduction rules, we assume that network terms are on normal form, as defined above. The rules are given in Table 4.

We call a location that contains an available input a *receiving location* for channel a and a location which contains an availabke output a *sending location*. For broadcast, we require that if the number of receiving locations for a channel a exceeds the bound $b(a,l)$ for a (or $\beta_a(m)$, if a is a bound name), then at most $b(a,l)$ receiving locations can receive the message. All other receiving locations do not receive anything and will still be waiting for an input. Each of the receiving locations must be connected to the sending location l. This is captured by the rule (R-BROADCAST).

For collection, the number of locations that can simultaneously send messages on a channel a cannot exceeds the bound $b(a,l)$ for a (or $\beta_a(m)$, if a is a bound name). Moreover, each sending location must be connected to the receiving location m. This is captured by the rule (R-COLLECT). Finally, the reduction rule (R-LOCAL) describes local communication within the confines of a single location.

Example 3. *Consider the network*

$$\begin{aligned} N = {} & l_1 \triangleright l_3 \mid l_2 \mid l_3 \mid l_1[\overline{a}\langle(a,b)\rangle] \\ & \mid l_2[\overline{a}\langle(c,b)\rangle] \mid l_3[a\,[\lambda x(x,b).S]\,.\overline{d}\langle \mathit{find}\text{-}a(S)\rangle] \end{aligned}$$

Assume that $b(a) = 2$. This network consists of three locations. Location l_1 offers an output on the a-channel of the pair (a,b), and location l_2 offers an output on the a-channel of the pair (c,b). At location l_3 we have a process that on the channel a will receive a set of messages, all of which are of the form (x,b) for some x, and subsequently output a if one of the pairs received contained a.

We have $N \to l_1 \triangleright l_3 \mid l_2 \mid l_3 \mid l_1[\mathbf{0}] \mid l_2[\mathbf{0}] \mid l_3[\overline{d}\langle a\rangle]$.

$$
\begin{array}{ll}
 & (\nu\vec{n} : \vec{\beta})(\mathscr{C} \mid l[\overline{a}\langle M\rangle.P_k] \mid \prod_{i=1}^{k} m_i[a\,(\lambda\vec{x}M_i')\,Q_i] \mid N_1) \\
\text{(BROAD)} & \longrightarrow \\
 & (\nu\vec{n} : \vec{\beta})(\mathscr{C} \mid l[P_k] \mid \prod_{i=k'+1}^{k} m_i[a\,(\lambda\vec{x}M_i')\,Q_i] \mid \prod_{i=1}^{k'} m_\ell[Q_i\theta_i] \mid N_1) \\
 & \text{where } l \triangleright m_i \in \mathscr{C} \text{ for all } 1 \leq i \leq m \text{ and either} \\
 & \qquad (1)\ a \notin \vec{n},\ k' = |\{m_\ell \mid l_k \triangleright m_\ell \in \mathscr{C}\}| \leq b(a,l) \text{ for all } \ell \in L, \text{ or} \\
 & \qquad (2)\ a \in \vec{n},\ k' \leq |\{m_\ell \mid l_k \triangleright m_\ell \in \mathscr{C}\}| \leq \beta_a(l) \text{ for all } \ell \in L, \\
 & \text{and for all } 1 \leq i \leq m' \text{ we have } M = M_i'\theta_i \text{ for some } \theta_i, \\
 & (\nu\vec{n} : \vec{\beta})(\mathscr{C} \mid \mathscr{D} \mid l[\overline{a}\langle M\rangle.P \mid a\,(\lambda\vec{x}M')\,Q \mid P'] \mid N_1) \\
\text{(LOCAL)} & \longrightarrow \\
 & (\nu\vec{n} : \vec{\beta})(\mathscr{C}_1 \mid \mathscr{D}_1 \mid l_k[P \mid Q\theta \mid P'] \mid N_1) \\
 & \text{where } M = M'\theta \text{ for some } \theta
\end{array}
$$

$$
\begin{array}{ll}
 & (\nu\vec{n} : \vec{\beta})(\mathscr{C} \mid \prod_{k\in K} l_k[\overline{a}\langle M_k\rangle.P_k \mid P_k'] \mid m[a\,[\lambda\vec{x}M.S]Q_\ell] \mid N_1) \\
\text{(COLL)} & \longrightarrow \\
 & (\nu\vec{n} : \vec{\beta})(\mathscr{C} \mid \prod_{k\in K} l_k[P_k \mid P_k'] \mid m[Q_\ell[S \mapsto \{\mid M_1,\ldots,M_{|K|}\mid\}] \mid N_1) \\
 & \text{where } l_j \triangleright m \in \mathscr{C} \text{ for all } j \in K \text{ and either } \begin{array}{l} a \notin \vec{n}, \text{ and } 1 \leq |K| \leq b(a,m) \text{ or} \\ a \in \vec{n} \text{ and } 1 \leq |K| \leq \beta_a(m) \end{array} \\
 & \text{and for all } j \in K \text{ we have } M_j = M\theta_\ell \text{ for some } \theta_\ell
\end{array}
$$

Table 4: Reduction rules for communication in BBC networks on normal form, assuming connectivity $b(a,m)$

4 Bisimilarity in BBC

4.1 Barbs

In our treatment of bisimilarity, we define an observability predicate (aka barbs) We write $P \downarrow_{a,\mathsf{B}}$ if P admits a broadcast observation on channel a, and $\vdash P \downarrow_{a,\mathsf{C}}$ if P admits a collection observation on channel a. We can define a similar predicate for networks. We write $N \downarrow_{a,d} @l$ if N allows a barb $\downarrow_{a,d}$ at location l. Table 5 contains the most interesting rules.

Definition 2. *A weak barbed bisimulation R is a symmetric binary relation on networks which satisfies that whenever $N_1 R N_2$ we have*

1. *If $N_1 \to N_1'$ then for some N_2' we have $N_2 \to^* N_2'$ where $N_1' R N_2'$*
2. *For every location l, if $N_1 \downarrow_{a,d} @l$ then also $N_2 \downarrow_{a,d} @l$*

We write $N_1 \dot{\approx} N_2$ if $N_1 R N_2$ for some weak barbed bisimulation R.

Theorem 2. *Weak barbed bisimilarity is an equivalence.*

As in other process calculi, the notion of barbed bisimilarity is unfortunately not a congruence, as it is not preserved under parallel composition.

Two networks are weak barbed congruent if they are barbed bisimilar in every parallel context, i.e. no surrounding network can tell them apart.

Definition 3. *We write $N_1 \approx N_2$ if for all networks N we have that $N_1 \mid N \dot{\approx} N_2 \mid N$*

(INP-B) $a\,(\lambda\vec{x}M)P_1 \downarrow_{a,\mathsf{B}}$ (INP-C) $a\,[\lambda\vec{x}M.S]P_1 \downarrow_{a,\mathsf{C}}$

(OUTP) $\overline{a}\langle M\rangle.P \downarrow_{a,d}$ (PAR) $\dfrac{P_1 \downarrow_{a,d}}{P_1 \mid P_2 \downarrow_{a,d}}$

(NEW) $\dfrac{P_1 \downarrow_{a,d}}{(\nu x:\beta_n)P_1 \downarrow_{a,d}} \quad x \neq a$ (LOCATE) $\dfrac{P \downarrow_{a,d}}{l[P] \downarrow_{a,d} @\ell}$

(CONG) $\dfrac{N_1 \downarrow_{a,d} @l \quad N_1 \equiv N}{N \downarrow_{a,d} @l}$

Table 5: Selected rules for barbs

5 A hierarchical protocol

In this section we outline how one can reason about a distributed protocol that involves both collection and broadcast. The protocol itself is generic but representative of the current trend in distributed communication systems with a large number of devices and very few controlling/central entities. In essence, this protocol is composed by traffic collection in the upstream direction, i.e. from the leaves to the central entity, and traffic broadcast in the downstream direction, i.e. from the central entity to the leaf nodes.

Figure 1 shows the multi-level network topology assumed for this protocol; the topology is that of a tree. Each level of the network is connected to the one above through a local collection and broadcast node, which we simply denote as local central node. This node is then a leaf node of the above level, e.g. the nodes denoted as C^1 are leaf nodes of the level l_0, while being the local central nodes at each of the locations of l_1^i, which for ease of notation we denote solely as l_1. We assume that there can be at most β nodes connected to every central node, meaning that the bound of every channel should be β.

The goal of the distributed protocol is to collect information from all leaf nodes in the network; and then communicate it to a central entity, denoted as C^0. The central entity C^0, then reaches a global decision, which is then communicated to the leaf nodes. For simplicity, we assume that this decision only depends on the data collected from the leaves.

5.1 The n-level protocol in BBC

In what follows, we define the components of the network and protocol inductively. We assume the network to have n levels or depth n, where level 0 corresponds to the central entity, while level n denotes the level where all the leaf nodes are.

One of the sub-networks of the n-level protocol at level k , as depicted in Figure 1, is composed by m nodes and a local central node C_k. We denote this sub-network by the agent identifier $D^{k,n}\left(a'_k, a''_k, b'_k, b''_k, \mathscr{L}, \ell_{k-1}\right)$ where $\mathscr{L}$ is a set of locations and ℓ_{k-1} is the local central location at $k-1$ level. The other name parameters are

- a'_k, a''_k - names of the channels used for communications at location l_k, i.e. between the leaf nodes and the local central entity; the names transmitted are collected by the central node at level $k+1$.
- b'_k, b''_k - names of the channels that the local central entity uses to communicate to the local central entity for which it is a lead, i.e. the channel over which the local central entity C_k communicates

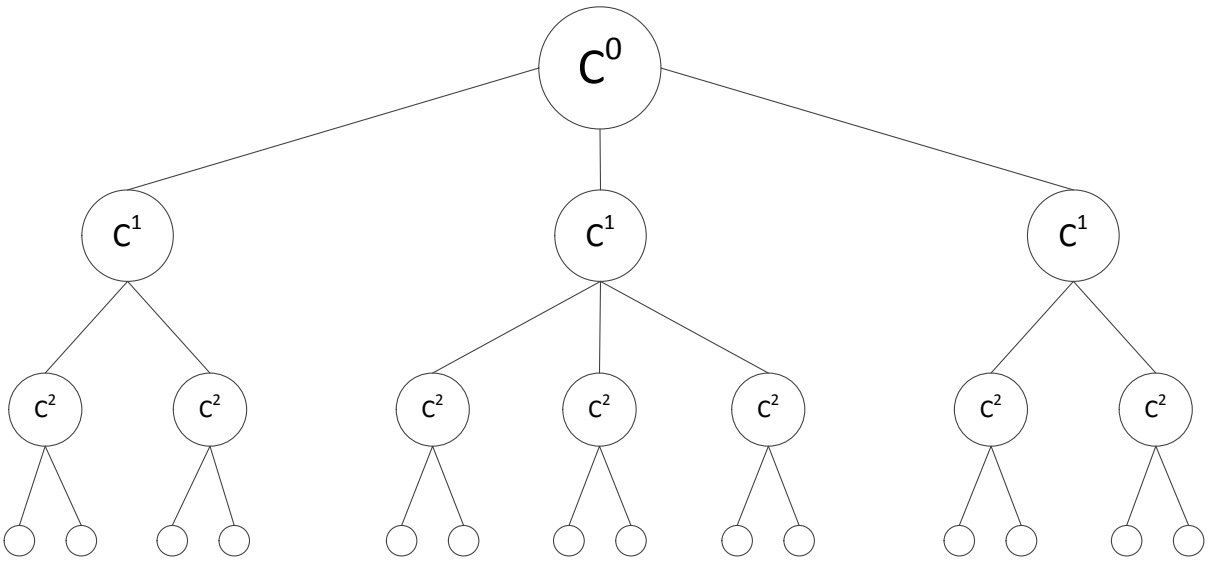

Figure 1: A multi-level network.

with the local central entity C_{k-1}. The names transmitted are broadcast to the child nodes of the local central (at level $k-1$).

- $\mathscr{L}$ is the set of nodes at this level.

At the leaf level, $k = 0$ we have,

$$D^{0,n}(a'_0, a''_0, b'_0, b''_0 \mathscr{L}, \ell_0) \stackrel{\text{def}}{=} \prod_{\ell_i \in \mathscr{L}} \ell_i[P_i(a'_0, a''_0)] \mid \prod_{\ell_i \in \mathscr{L}} \ell_i \triangleright \ell_1 \mid \ell_0[C^0(a'_0, a''_0, b'_0, b''_0)]$$

The subprocess P_i where $\ell_i \in \mathscr{L}$ is defined by

$$P_i(a'_0, a''_0) \stackrel{\text{def}}{=} (\nu \omega_i)\overline{a'_0}\langle \omega_i \rangle.a''_0(z).(Q_i(z) \mid P_i(a'_0, a''_0))$$

where $Q_i(z)$ depicts the computation that occurs at the leaf ℓ_i; we shall not describe this here and ω_i is the local name that is the contribution made by the process.

The local central node is defined by

$$C^{0,n}(a', a'', b', b'') \stackrel{\text{def}}{=} a'(S).\overline{b'}\langle f(S) \rangle.b''(z).\overline{a''}\langle z \rangle.C^{0,n}(a', a'', b', b'')$$

This process will receive names that are collected to the set S and then pass the processed information $f(S)$ upwards on the b channel. After this, the local central node waits for a response to the information sent and passes on the received name to its children.

The *selection function* $f : \mathscr{P}(\mathscr{N}) \to \mathscr{N}$ processes the messages received from the leaf nodes and selects a name; this could be e.g. a channel estimate. We require that selection is *idempotent* in the sense that for any family of multisets $S_1, \ldots, S_k$ we have

$$f(\{f(S_1), \ldots, f(S_k)\}) = f(\cup_{i=1}^{k} S_i)$$

If we think of names as natural numbers, the function $f(S) = \min_{x \in S} x$ is an example of an idempotent selection function.

For intermediate levels, $0 < k < n$, $D^k(a_k, b_k, \ell_k, \ell_{k-1})$ is defined as follows,

$$D^{k+1,n}(a'_{k+1}, a''_{k+1}, b'_{k+1}, b''_{k+1}, \mathscr{L}, \ell_{k+1}) \stackrel{\text{def}}{=} (\nu\vec{\ell}, a'_k : \beta, a''_k : \beta, b'_k : \beta, b''_k : \beta)$$
$$\left(\prod_{\mathscr{L}_i \in \mathsf{Locs}(\mathscr{L},k)} D^{k,n}(a'_k, a''_k, b'_k, b''_k, \mathscr{L}_i, \ell_i) \mid \prod_{i=1}^{m} \ell_i \triangleright \ell_k \mid \ell_{k+1}[C^{k+1}(a'_k, a''_k, b'_k, b''_k)] \right)$$

where the set of locations $\mathsf{Locs}(\mathscr{L},k)$ can be partitioned as $\mathsf{Locs}(\mathscr{L},k) = \bigcup_i i \in I \mathscr{L}_i$ where I is some index set and $\mathscr{L}_i \cap \mathscr{L}_j = \emptyset$ for all $i \neq j, i, j \in I$. m'_i is the number of neighbours in the i'th sub-network found at level $k+1$ and $\ell_i \in \vec{\ell}$.

The definition of local central nodes is (for $k < n$)

$$C^{k,n}((a_k, b_k) \stackrel{\text{def}}{=} a_k(S).\overline{b_k}\langle f(S)\rangle.b_k(z).\overline{a_k}\langle z\rangle.(Q_c(z) \mid C^k(a_k, b_k))$$

Finally, at level n the information collected from all the leaf nodes reaches the central entity, where the final processing of the received information is performed. The description of the entire network D^n is $D^{n,n}_m(a_0, \ell_c)$, defined as

$$D^{n,n}(a'_n, a''_n, b'_n, b''_n, \ell_n) \stackrel{\text{def}}{=} (\nu\vec{\ell}, a'_{n-1} : \beta, a''_{n-1} : \beta, b'_{n-1} : \beta, b''_{n-1} : \beta)$$
$$\left(\prod_{\mathscr{L}_i \in \mathsf{Locs}(\mathscr{L},n)} D^{1,n}_{m'_i}(a'_{n-1}, a''_{n-1}, b'_{n-1}, b''_{n-1}, \mathscr{L}_i, \ell_n) \mid \prod_{i=1}^{m} \ell_i \triangleright \ell_0 \mid \ell_0[C^n(a'_n, a''_n, b'_n, b''_n)] \right)$$

Here the central entity will collect the estimates from its children and then broadcast back the value of the selection.

$$C^{n,n}(a'_n, a''_n, b'_n, b''_n) \stackrel{\text{def}}{=} a'_n(S).\overline{b'_n}\langle f(S)\rangle.C^{n,n}(a'_n, a''_n, b'_n, b''_n)) \tag{1}$$

5.2 Correctness

We would like the multi-level network that serves K leaf nodes to be equivalent to a single-level network containing the same K leaf nodes but now connected to a single central entity.

To this end we define the flattened version of a hierarchical network N, denoted $\mathsf{flat}(N, \ell)$. This is precisely a network in which all nodes are connected to the same central node at location ℓ.

This operation is defined inductively as follows.

$$\mathsf{flat}(D^{0,n}(a'_0, a''_0, b'_0, b''_0, \mathscr{L}, \ell_0), \ell) \stackrel{\text{def}}{=} D^{0,n}(a'_0, a''_0, b'_0, b''_0, \mathscr{L}, \ell_0)$$
$$\mathsf{flat}(D^{k+1,n}(a'_{k+1}, a''_{k+1}, b'_{k+1}, b''_{k+1}, \mathscr{L}, \ell_{k+1}), \ell) \stackrel{\text{def}}{=}$$
$$(\nu\vec{\ell}, a'_k, a''_k, b'_k, b''_k) \left(\prod_{\mathscr{L}_i \in \mathsf{Locs}(\mathscr{L},k)}^{m} \mathsf{flat}(D^{k,n}(a'_k, a''_k, b'_k, b''_k, \mathscr{L}_i, \ell_i)) \mid \prod_{i=1}^{m} \ell_i \triangleright \ell \right) \text{where } 0 < k < n$$

Theorem 3. *For every $n \geq 0$, for every set of distinct names $\{a'_0, a''_0, b'_0, b''_0\}$ and set of locations $\mathscr{L}$ we have*

$$D^{0,n}(a'_0, a''_0.b'_0, b''_0, \mathscr{L}, \ell) \dot{\approx} \mathsf{flat}(D^{0,n}(a'_0, a''_0, b'_0, b''_0, \mathscr{L}, \ell), \ell)$$

6 A type system

We can formulate a type discipline that distinguishes between broadcast and collection and assigns bounds to these.

Types are given by the formation rules

$$T ::= \mathsf{Ch_C}(T) \mid \mathsf{Ch_B}(T) \mid T^k \mid T_1 \times T_2 \cdots \times T_k \mid T_1 \to T_2 \mid \mathsf{Loc}$$

We now explain the intent behind these formation rules.

- The type $\mathsf{Ch_C}(T)$ is the type of a channel than can collect messages of type T, and the type $\mathsf{Ch_B}(T)$ is the type of a channel that can broadcast a message of type T.
- A multiset with k elements of type T will have type T^k.
- A tuple with k components of types $T_1, \ldots, T_k$, respectively, will have type $T_1 \times \cdots \times T_k$
- A constructor f will have type $T_1 \to T_2$, and a multiset g will have type $T^k \to T$. If g has type $T^k \to T$, g produces an element of type T from a multiset of k terms, each having type T.
- Locations have type Loc.

A type environment Γ is a finite function $\Gamma : \mathbf{Names} \cup \mathbf{Sels} \cup \mathbf{Cons} \to \mathbf{Types}$ that assigns types to all names, term constructors and multiset selectors.

Type judgments are relative to a type environment and a set of agent variables Θ and are of the form

$$\Gamma \vdash M : T \text{ for terms}$$
$$\Gamma \vdash^{\Theta} P \text{ for processes}$$
$$\Gamma \vdash^{\Theta} N \text{ for networks}$$

When the set is the same in all the judgements of a type rule, we omit it throughout the rule.

(VAR) $\Gamma \vdash x : T$ if $\Gamma(x) = T$ (SETVAR) $\Gamma \vdash S : T^k$ if $\Gamma(S) = T^k$

(MULTISET) $$\frac{\Gamma \vdash M_i : T \quad 1 \leq i \leq k}{\Gamma \vdash \{M_1, \ldots, M_k\} : T^k}$$ (SELECT) $$\frac{E : T^k \quad \Gamma(g) = T^k \to T}{\Gamma \vdash g(E) : T}$$

(CONS) $$\frac{\Gamma \vdash M : T_1 \quad \Gamma(f) = T_1 \to T_2}{\Gamma \vdash f(M) : T_2}$$

Table 6: Type rules for message terms

The following result, which guarantees that a well-typed network stays well-typed under reductions, is easily established.

Theorem 4. *If* $\Gamma \vdash N$ *and* $N \to N'$ *then* $\Gamma \vdash N'$

Proof. Induction in the reduction rules. A crucial lemma needed in the proof is that whenever $N_1 \equiv N_2$, then $\Gamma \vdash N_1$ if and only if $\Gamma \vdash N_2$. □

$$\text{(B-INP)}\quad \frac{\Gamma,\vec{x}:\vec{T}\vdash M:T_1 \quad \Gamma(a)=\mathsf{Ch}_{\mathsf{B}}(T_1) \quad \Gamma,\vec{x}:\vec{T}\vdash P_1}{\Gamma\vdash a(\lambda\vec{x}M)P_1}$$

$$\text{(C-INP)}\quad \frac{\Gamma,\vec{x}:\vec{T}\vdash M:T_1 \quad \Gamma(a)=\mathsf{Ch}_{\mathsf{C}}(T_1) \quad \Gamma,\vec{x}:\vec{T}\vdash P_1}{\Gamma\vdash a[\lambda\vec{x}M.S]P_1}$$

$$\text{(OUTP)}\quad \frac{\Gamma(a)=\mathsf{Ch}_{\mathsf{d}}(T) \quad \mathsf{d}\in\{\mathsf{B},\mathsf{C}\} \quad \Gamma\vdash M:T_1}{\Gamma\vdash \overline{a}\langle M\rangle.P_1}$$

$$\text{(NIL)}\quad \Gamma\vdash \mathbf{0} \qquad \text{(PAR)}\quad \frac{\Gamma\vdash P_1 \quad \Gamma\vdash P_2}{\Gamma\vdash P_1\mid P_2}$$

$$\text{(SUM)}\quad \frac{\Gamma\vdash P_1 \quad \Gamma\vdash P_2}{\Gamma\vdash P_1+P_2} \qquad \text{(MATCH)}\quad \frac{\Gamma\vdash M_1:T \quad \Gamma\vdash M_2:T \quad \Gamma\vdash P_1}{\Gamma\vdash [M_1=M_2]P_1}$$

$$\text{(MISMATCH)}\quad \frac{\Gamma\vdash M_1:T \quad \Gamma\vdash M_2:T \quad \Gamma\vdash P_1}{\Gamma\vdash [M_1\neq M_2]P_1} \qquad \text{(NEW)}\quad \frac{\Gamma,x:T\vdash P_1}{\Gamma\vdash (\nu x:\beta_n)P_1}$$

$$\text{(AGENT)}\quad \frac{\Gamma\vdash M_i:T_i \quad 1\le i\le|\vec{M}| \quad A\in\Theta}{\Gamma\vdash^{\Theta} A(\vec{M})}$$

$$\text{(CALL)}\quad \frac{\Gamma,\vec{x}:\vec{T}\vdash^{\Theta\cup\{A\}} P \quad \Gamma\vdash M_i:T_i \quad 1\le i\le|\vec{M}| \quad A(\vec{x})\stackrel{\text{def}}{=}P}{\Gamma\vdash^{\Theta} A(M)}$$

Table 7: Type rules for processes

7 The expressive power of BBC

There is no compositional encoding of BBC into the π-calculus. In the presence of broadcast this follows from the negative result due to Ene and Muntean [1]. However, the result also holds if we only consider collection. The following criteria for compositionality are due to Palamidessi [6] and Ene and Muntean [1].

Definition 4. *An encoding* $[\![\quad]\!]$ *is* compositional *if*

1. $[\![N_1 \mid N_2]\!] = [\![N_1]\!] \mid [\![N_2]\!]$.
2. *For any substitution* σ *we have* $[\![N\sigma]\!] = [\![N]\!]\sigma$
3. *If* $N \to^* N'$ *then* $[\![N]\!] \to^* [\![N']\!]$.
4. *If* $[\![N]\!] \to^* M$ *then* $[\![N]\!] \to^* M \to^* [\![N']\!]$ *for some* N'.

A central notion in the study of process calculi is that of an *electoral system*. This is a network in which the participants perform a computation that elects a unique leader. Following [6, 1], our definition

$$\text{(PAR-N)} \qquad \frac{\Gamma \vdash N_1 \quad \Gamma \vdash N_2}{\Gamma \vdash N_1 \mid N_2}$$

$$\text{(NEW)} \qquad \frac{\Gamma, x : T \vdash N_1}{\Gamma \vdash (\nu x : \beta_n) N_1}$$

$$\text{(LOC)} \qquad \frac{\Gamma \vdash P \quad \Gamma(\ell) = \mathsf{Loc}}{\Gamma \vdash l[P]}$$

$$\text{(NEAR)} \qquad \frac{\Gamma(l) = \mathsf{Loc} \quad \Gamma(m) = \mathsf{Loc}}{\Gamma \vdash l \triangleright m}$$

$$\text{(NOT-NEAR)} \qquad \frac{\Gamma(l) = \mathsf{Loc} \quad \Gamma(m) = \mathsf{Loc}}{\Gamma \vdash l \not\triangleright m}$$

Table 8: Type rules for networks

assumes a network in which the names used are the 'natural names' that represent the identity of the n processes in the network.

Definition 5 (Electoral system)**.** *A network $N = P_1 \mid \cdots \mid P_n$ is an* electoral system *if for every computation C, there exists an extension C' of C, and an index $k \in \{1,\ldots,n\}$ such that for every $i \in \{1,\ldots,n\}$ the projection C'_i contains exactly one output action of the form k and any trace of a P_i may contain at most one action of the form $\overline{l}$ with $l \in \{1,\ldots,n\}$.*

We now describe such a system in BBC. The system uses a common channel a whose bound is n, where n is the number of principals. The idea is to collect names until n sets of names have been collected. The first to do so sends out a success announcement to everyone in the form of a name $\mathsf{chosen}(m)$; the collection function $g(S)$ is defined to select the name with the minimal index among the names of S unless S contains one or more occurrences of a name of the form $\mathsf{chosen}(m')$ for some m. Thus, the first component to receive sufficiently many names from all other components chooses the leader.

Our network is of the form

$$N = \prod_{i=1}^{n} l[l_i]P_i \mid \prod_{i=1}^{n} \prod_{j=1, j\neq i}^{m} l_i \triangleright l_j$$

We define the n components as follows:

$$P_i \stackrel{\text{def}}{=} \overline{a}\langle i\rangle.a[\lambda x.S_1]\ldots a[\lambda x.S_k]\overline{a}\langle \mathsf{chosen}(g(\{\{g(S_1),\ldots g(S_k)\}\}))\rangle.P$$

The selection function is defined by

$$g(S) = \begin{cases} \min_{x\in S} x & \text{if } S \subseteq \{1,\ldots,n\} \\ \mathsf{chosen}(n) & \text{if } \mathsf{chosen}(n) \in S \end{cases}$$

As there is no such electoral system for the π-calculus [6], we can use the same proof as that given by Ene and Muntean to conclude that there can be no compositional encoding of BBC in the π-calculus.

8 Directions for further work

In this paper we have presented the BBC calculus, which is a distributed process calculus that generalizes the π-calculus with notions of channels with bounded forms of broadcast and collection and an explicit notion of connectivity.

The present work only considers barbed bisimulation; it is well-known that this relation is not preserved by parallel composition and that the notion of barbed congruence does not lend itself well to coinductive proof techniques. Further work includes finding a labelled characterization of barbed congruence; this requires a labelled transition semantics in the spirit of [1].

At present, we informally distinguish between channels for broadcast and collection. However, if we think of channels as radio frequencies, one would sometimes like the same frequency to be used for both broadcast and collection. An interesting direction of work is therefore to develop a type system that will allow the same channel to be used non-uniformly in both modes and according to a protocol. This is a variant of binary session types [2] that has been adapted to a setting with broadcast communication by [5]. A further step in this direction is to study multiparty session types [3] and how the projection to local session types can be defined in the setting of BBC.

References

[1] Cristian Ene and Traian Muntean. A broadcast-based calculus for communicating systems. In *Proceedings of IPDPS'1*. IEEE Computer Society, 2001. doi:10.1007/3-540-48321-7_21

[2] Kohei Honda, Vasco Thudichum Vasconcelos, and Makoto Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP*, volume 1381 of *LNCS*, pages 122–138. Springer, 1998. doi:10.1007/BFb0053567

[3] Kohei Honda, Nobuko Yoshida, and Marco Carbone. Multiparty asynchronous session types. In George C. Necula and Philip Wadler, editors, *POPL*, pages 273–284. ACM, 2008. doi:10.1145/1328438.1328472

[4] Dimitrios Kouzapas and Anna Philippou. A process calculus for dynamic networks. In Roberto Bruni and Juergen Dingel, editors, *Formal Techniques for Distributed Systems*, volume 6722 of *Lecture Notes in Computer Science*, pages 213–227. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-21461-5_14

[5] Dimitrios Kouzapas and Ramunas Gutkovas and Simon J. Gay. Session Types for Broadcasting In Alastair F. Donaldson and Vasco T. Vasconcelos. *Proceedings 7th Workshop on Programming Language Approaches to Concurrency and Communication-cEntric Software, PLACES 2014*, Grenoble, France, 12 April 2014, pages 25–31. doi:10.4204/EPTCS.155.4

[6] Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *MSCS*, 13(5):685–719, 2003. doi:10.1017/S0960129503004043

[7] K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Prog.*, 25(2-3):285–327, 1995. doi:10.1016/0167-6423(95)00017-8

[8] James Riely and Matthew Hennessy. A typed language for distributed mobile processes (extended abstract). In *POPL*, pages 378–390, 1998. doi:10.1145/268946.268978

[9] Davide Sangiorgi and David Walker. *The π-Calculus: A Theory of Mobile Processes*, Cambridge University Press, 2001. ISBN 978-0-521-78177-0.

[10] Anu Singh, C. R. Ramakrishnan, and Scott A. Smolka. A process calculus for mobile ad hoc networks. *Science of Computer Programming*, 75(6):440–469, 2010. doi:10.1016/j.scico.2009.07.008