Broadcast Psi-calculi with an Application to Wireless Protocols

Johannes Borgström[1], Shuqin Huang[2], Magnus Johansson[1], Palle Raabjerg[1], Björn Victor[1], Johannes Åman Pohjola[1], and Joachim Parrow[1]

[1] Department of Information Technology, Uppsala University, Sweden
[2] Peking University, China

Abstract. Psi-calculi is a parametric framework for extensions of the pi-calculus, with arbitrary data structures and logical assertions for facts about data. In this paper we add primitives for broadcast communication in order to model wireless protocols. The additions preserve the purity of the psi-calculi semantics, and we formally prove the standard congruence and structural properties of bisimilarity. We demonstrate the expressive power of broadcast psi-calculi by modelling the wireless ad-hoc routing protocol LUNAR and verifying a basic reachability property.

1 Introduction

Psi-calculi is a parametric framework for extensions of the pi-calculus, with arbitrary data structures and logical assertions for facts about data. In psi-calculi (described in Section 2) the purity of the semantics is on par with the original pi-calculus, the generality and expressiveness exceeds many earlier extensions of the pi-calculus, and the meta-theory is proved correct once and for all using the interactive theorem prover Isabelle/Nominal [26].

In order to model wireless communication used in WSN (Wireless Sensor Network) and MANET (Mobile Ad-hoc Network) applications, the concept of broadcast communication is needed, where one transmission can be received by several processes. Broadcast communication cannot be encoded in the pi-calculus [5]; we extend the psi-calculi framework with broadcast primitives (Section 3). The broadcast primitives are added using new operational actions and rules, and new connectivity predicates. We formally prove the congruence properties of bisimilarity and the soundness of structural equivalence laws using the Isabelle/Nominal theorem prover.

The connectivity predicates allow us to model systems with limited reachability, for instance where a transmitter only reaches nodes within a certain range, and systems with changing reachability, for instance due to physical mobility of nodes. In Section 4, we present a technique for treating different generations of connectivity information. Broadcast channels can be globally visible or have limited scope. Scoped channels can be protected from externally imposed connectivity changes, while permitting connectivity changes by processes within the scope of the channel.

We demonstrate the expressive power of the resulting framework in Section 5, where we provide a model of the LUNAR protocol for routing in ad-hoc wireless networks [24]. The model follows the specification closely, and demonstrates several features of the psi-calculi framework: both unicast and broadcast communication, application-specific data structures and logics, classic unstructured channels as well as pairs corresponding to MAC address and port selector. Our model is significantly more succinct than earlier work [28,27] (ca 30 vs 250 lines). We show an expected basic reachability property of the model: if two network nodes, a sender and a receiver, are both in range of a third node, but not within range of each other, the LUNAR protocol can find a route and transparently handle the delivery of a packet from the sender to the receiver.

We discuss related work on process calculi for wireless broadcast in Section 6, and conclude and present ideas for future work in Section 7.

2 Psi-calculi

This section is a brief recapitulation of psi-calculi; for a more extensive treatment including motivations and examples see [3,4].

We assume a countably infinite set of atomic *names* $\mathcal{N}$ ranged over by $a, b, \ldots, z$. Intuitively, names will represent the symbols that can be scoped, and also represent symbols acting as variables in the sense that they can be subject to substitution. A *nominal set* [18,6] is a set equipped with a formal notion of what it means for a name a to occur in an element X of the set, written $a \in \mathrm{n}(X)$ (often pronounced as "a is in the support of X"). We write $a\#X$, pronounced "a is fresh for X", for $a \notin \mathrm{n}(X)$, and if A is a set of names we write $A\#X$ to mean $\forall a \in A \,.\, a\#X$. In the following $\tilde{a}$ means a finite sequence of names, $a_1, \ldots, a_n$. The empty sequence is written ϵ and the concatenation of $\tilde{a}$ and $\tilde{b}$ is written $\tilde{a}\tilde{b}$. When occurring as an operand of a set operator, $\tilde{a}$ means the corresponding set of names $\{a_1, \ldots, a_n\}$. We also use sequences of other nominal sets in the same way.

A *nominal datatype* is a nominal set together with a set of functions on it. In particular we shall consider substitution functions that substitute elements for names. If X is an element of a datatype, the *substitution* $X[\tilde{a} := \tilde{Y}]$ is an element of the same datatype as X. There is considerable freedom in the choice of functions and substitutions; see [3,4] for details.

A psi-calculus is defined by instantiating three nominal data types and four operators:

Definition 1 (Psi-calculus parameters). *A psi-calculus requires the three (not necessarily disjoint) nominal data types: the (data) terms* $\mathbf{T}$*, ranged over by* M, N*, the conditions* $\mathbf{C}$*, ranged over by* φ*, the assertions* $\mathbf{A}$*, ranged over by* Ψ*, and the four equivariant operators:*

$$\begin{array}{ll} \dot{\leftrightarrow} : \mathbf{T} \times \mathbf{T} \to \mathbf{C} & \textit{Channel Equivalence} \\ \otimes : \mathbf{A} \times \mathbf{A} \to \mathbf{A} & \textit{Composition} \\ \mathbf{1} : \mathbf{A} & \textit{Unit} \\ \vdash \subseteq \mathbf{A} \times \mathbf{C} & \textit{Entailment} \end{array}$$

and substitution functions $[\widetilde{a} := \widetilde{M}]$, substituting terms for names, on each of $\mathbf{T}$, $\mathbf{C}$ and $\mathbf{A}$.

The binary functions above will be written in infix. Thus, if M and N are terms then $M \dot{\leftrightarrow} N$ is a condition, pronounced "M and N are channel equivalent" and if Ψ and Ψ' are assertions then so is $\Psi \otimes \Psi'$. Also we write $\Psi \vdash \varphi$, "Ψ entails φ", for $(\Psi, \varphi) \in \vdash$.

We say that two assertions are equivalent, written $\Psi \simeq \Psi'$ if they entail the same conditions, i.e. for all φ we have that $\Psi \vdash \varphi \Leftrightarrow \Psi' \vdash \varphi$. We impose certain requisites on the sets and operators. In brief, channel equivalence must be symmetric and transitive, $\otimes$ must be compositional with regard to $\simeq$, and the assertions with $(\otimes, \mathbf{1})$ form an abelian monoid modulo $\simeq$. For details see [3].

A *frame* F can intuitively be thought of as an assertion with local names: it is of the form $(\nu\widetilde{b})\Psi$ where $\widetilde{b}$ is a sequence of names that bind into the assertion Ψ. We use F, G to range over frames. We overload Ψ to also mean the frame $(\nu\epsilon)\Psi$ and $\otimes$ to composition on frames defined by $(\nu\widetilde{b_1})\Psi_1 \otimes (\nu\widetilde{b_2})\Psi_2 = (\nu\widetilde{b_1}\widetilde{b_2})(\Psi_1 \otimes \Psi_2)$ where $\widetilde{b_1} \# \widetilde{b_2}, \Psi_2$ and vice versa. We write $(\nu c)((\nu\widetilde{b})\Psi)$ for $(\nu c\widetilde{b})\Psi$.

Alpha equivalent frames are identified. We define $F \vdash \varphi$ to mean that there exists an alpha variant $(\nu\widetilde{b})\Psi$ of F such that $\widetilde{b} \# \varphi$ and $\Psi \vdash \varphi$. We also define $F \simeq G$ to mean that for all φ it holds that $F \vdash \varphi$ iff $G \vdash \varphi$.

Definition 2 (Psi-calculus agents). *Given valid psi-calculus parameters as in Definition 1, the psi-calculus* agents*, ranged over by $P, Q, \ldots$, are of the following forms.*

$\mathbf{0}$	Nil
$\overline{M} N \,.\, P$	Output
$\underline{M}(\lambda\widetilde{x})N \,.\, P$	Input
$\mathbf{case}\ \varphi_1 : P_1 \ [\!]\ \cdots\ [\!]\ \varphi_n : P_n$	Case
$(\nu a)P$	Restriction
$P \mid Q$	Parallel
$!P$	Replication
$(\!\vert \Psi \vert\!)$	Assertion

Restriction binds a in P and Input binds $\widetilde{x}$ in both N and P. We identify alpha equivalent agents. An assertion is guarded *if it is a subterm of an Input or Output. An agent is* assertion guarded *if it contains no unguarded assertions. An agent is* well-formed *if in $\underline{M}(\lambda\widetilde{x})N.P$ it holds that $\widetilde{x} \subseteq \mathrm{n}(N)$ is a sequence without duplicates, that in a replication $!P$ the agent P is assertion guarded, and that in* **case** *$\varphi_1 : P_1 \ [\!]\ \cdots\ [\!]\ \varphi_n : P_n$ the agents P_i are assertion guarded.*

The agent **case** $\varphi_1 : P_1 \ [\!]\ \cdots\ [\!]\ \varphi_n : P_n$ is sometimes abbreviated as **case** $\widetilde{\varphi} : \widetilde{P}$, or if $n = 1$ as **if** φ_1 **then** P_1.

The *frame $\mathcal{F}(P)$ of an agent* P is defined inductively as follows:

$$\begin{aligned}
&\mathcal{F}(\underline{M}(\lambda\widetilde{x})N \,.\, P) = \mathcal{F}(\overline{M}\, N \,.\, P) = \mathcal{F}(\mathbf{0}) = \mathcal{F}(\mathbf{case}\ \widetilde{\varphi} : \widetilde{P}) = \mathcal{F}(!P) = \mathbf{1}\\
&\mathcal{F}((\!\vert \Psi \vert\!)) = (\nu\epsilon)\Psi\\
&\mathcal{F}(P \mid Q) = \mathcal{F}(P) \otimes \mathcal{F}(Q)\\
&\mathcal{F}((\nu b)P) = (\nu b)\mathcal{F}(P)
\end{aligned}$$

$$\textsc{In}\ \frac{\Psi \vdash K \dot{\leftrightarrow} M}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N\,.\,P \xrightarrow{\underline{K}\,N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y}:=\widetilde{L}]} \qquad \textsc{Out}\ \frac{\Psi \vdash M \dot{\leftrightarrow} K}{\Psi \rhd \overline{M}\,N\,.\,P \xrightarrow{\overline{K}N} P}$$

$$\textsc{Case}\ \frac{\Psi \rhd P_i \xrightarrow{\alpha} P' \qquad \Psi \vdash \varphi_i}{\Psi \rhd \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \xrightarrow{\alpha} P'}$$

$$\textsc{Com}\ \frac{\begin{array}{c}\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \dot{\leftrightarrow} K \\ \Psi_Q \otimes \Psi \rhd P \xrightarrow{\overline{M}(\nu\widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{\underline{K}\,N} Q'\end{array}}{\Psi \rhd P \mid Q \xrightarrow{\tau} (\nu\widetilde{a})(P' \mid Q')}\ \widetilde{a}\#Q$$

$$\textsc{Par}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd P \mid Q \xrightarrow{\alpha} P' \mid Q}\ \mathrm{bn}(\alpha)\#Q \qquad \textsc{Scope}\ \frac{\Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd (\nu b)P \xrightarrow{\alpha} (\nu b)P'}\ b\#\alpha,\Psi$$

$$\textsc{Open}\ \frac{\Psi \rhd P \xrightarrow{\overline{M}(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{\overline{M}(\nu\widetilde{a}\cup\{b\})N} P'}\ \begin{array}{l} b\#\widetilde{a},\Psi,M \\ b \in \mathrm{n}(N)\end{array} \qquad \textsc{Rep}\ \frac{\Psi \rhd P \mid !P \xrightarrow{\alpha} P'}{\Psi \rhd !P \xrightarrow{\alpha} P'}$$

Table 1. Structured operational semantics. Symmetric versions of COM and PAR are elided. In the rule COM we assume that $\mathcal{F}(P) = (\nu\widetilde{b_P})\Psi_P$ and $\mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q$ where $\widetilde{b_P}$ is fresh for all of $\Psi, \widetilde{b_Q}, Q, M$ and P, and that $\widetilde{b_Q}$ is similarly fresh. In the rule PAR we assume that $\mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q$ where $\widetilde{b_Q}$ is fresh for Ψ, P and α. In OPEN the expression $\tilde{a} \cup \{b\}$ means the sequence $\tilde{a}$ with b inserted anywhere.

The *actions* ranged over by α, β are of the following three kinds:
Output $\overline{M}(\nu\tilde{a})N$ where $\alpha \subseteq \mathrm{n}(N)$, Input $\underline{M}\,N$, and Silent τ. Here we refer to M as the *subject* and N as the *object*. We define $\mathrm{bn}(\overline{M}(\nu\tilde{a})N) = \tilde{a}$, and $\mathrm{bn}(\alpha) = \emptyset$ if α is an input or τ. We also define $\mathrm{n}(\tau) = \emptyset$ and $\mathrm{n}(\alpha) = \mathrm{n}(M) \cup \mathrm{n}(N)$ for the input and output actions.

Definition 3 (Transitions).
A transition *is written* $\Psi \rhd P \xrightarrow{\alpha} P'$*, meaning that in the environment* Ψ *the well-formed agent* P *can do an* α *to become* P'*. The transitions are defined inductively in Table 1. We write* $P \xrightarrow{\alpha} P'$ *without an assertion to mean* $\mathbf{1} \rhd P \xrightarrow{\alpha} P'$*.*

Agents, frames and transitions are identified by alpha equivalence. In a transition the names in $\mathrm{bn}(\alpha)$ bind into both the action object and the derivative, therefore $\mathrm{bn}(\alpha)$ is in the support of α but not in the support of the transition. This means that the bound names can be chosen fresh, substituting each occurrence in both the object and the derivative.

Definition 4 (Strong bisimulation). *A* strong bisimulation $\mathcal{R}$ *is a ternary relation on assertions and pairs of agents such that* $\mathcal{R}(\Psi, P, Q)$ *implies all of*

1. *Static equivalence:* $\Psi \otimes \mathcal{F}(P) \simeq \Psi \otimes \mathcal{F}(Q)$
2. *Symmetry:* $\mathcal{R}(\Psi, Q, P)$
3. *Extension of arbitrary assertion:* $\forall \Psi'.\ \mathcal{R}(\Psi \otimes \Psi', P, Q)$
4. *Simulation: for all* α, P' *such that* $\mathrm{bn}(\alpha)\#\Psi, Q$ *there exists a* Q' *such that*

$$\Psi \rhd P \xrightarrow{\alpha} P' \Longrightarrow \Psi \rhd Q \xrightarrow{\alpha} Q' \wedge \mathcal{R}(\Psi, P', Q')$$

We define $P \dot{\sim}_{\Psi} Q$ *to mean that there exists a bisimulation* $\mathcal{R}$ *such that* $\mathcal{R}(\Psi, P, Q)$, *and write* $\dot{\sim}$ *for* $\dot{\sim}_{\mathbf{1}}$.

Strong bisimulation is preserved by all operators except input prefix and satisfies the expected algebraic laws such as scope extension, for details see [3,4].

3 Broadcast semantics

In this section we extend the unicast psi-calculi of the previous section with a broadcast semantics that models wireless (i.e., synchronous and unreliable) broadcast. As an example, assume that the connectivity information Ψ allows receivers M_1 and M_2 to listen to channel K. We would then expect the following transition: $\Psi \rhd \overline{K}\,N.P \mid \underline{M_2}(x).Q \mid \underline{M_3}(y).R \xrightarrow{\overline{K}\,N} P \mid Q[x := N] \mid R[y := N]$.

To allow connectivity to depend on assertions, and to permit broadcast channels to be computed at run-time, we assume a psi-calculus with the following extra predicates:

Definition 5 (Extra predicates for broadcast).

$$\begin{array}{ll} \dot{\prec} :\ \mathbf{T} \times \mathbf{T} \rightarrow \mathbf{C} & \textit{Output Connectivity} \\ \dot{\succ} :\ \mathbf{T} \times \mathbf{T} \rightarrow \mathbf{C} & \textit{Input Connectivity} \end{array}$$

The first predicate, $M \mathrel{\dot{\prec}} K$, is pronounced "$M$ is out-connected to K" and means that an output prefix $\overline{M}\,N$ can result in a broadcast on channel K. The second, $K \mathrel{\dot{\succ}} M$, is pronounced "$M$ is in-connected to K" and means that an input prefix $\underline{M}(\lambda\widetilde{x})N$ can receive broadcast messages from channel K. As usual in broadcast calculi, the receivers need to be using the same broadcast channel as the sender in order to receive a message.

As an example, we can model routing table lookup: if *tab* is a term corresponding to a routing table we can let $\Psi \vdash \mathtt{lookup}(tab, id) \mathrel{\dot{\prec}} ch$ be true if (id, ch) appears in *tab*. We can also model connectivity: if Ψ contains connectivity information between receivers n and channels ch we may let $\Psi \vdash ch \mathrel{\dot{\succ}} \mathtt{rcv}(n, ch)$ be true if n is connected to ch according to Ψ.

In contrast to unicast connectivity, we do not require broadcast connectedness to be symmetric or transitive, so in particular $M \mathrel{\dot{\prec}} K$ might not be equivalent to $K \mathrel{\dot{\succ}} M$. Instead, for technical reasons related to scope extension, broadcast channels must have no greater support than the input and output prefixes that can make use of them.

$$\textsc{BrOut}\ \frac{\Psi \vdash M \mathrel{\dot{\prec}} K}{\Psi \rhd \overline{M}\,N\,.\,P \xrightarrow{!\overline{K}\,N} P} \qquad \textsc{BrIn}\ \frac{\Psi \vdash K \mathrel{\dot{\succ}} M}{\Psi \rhd \underline{M}(\lambda\widetilde{y})N\,.\,P \xrightarrow{?\underline{K}\,N[\widetilde{y}:=\widetilde{L}]} P[\widetilde{y}:=\widetilde{L}]}$$

$$\textsc{BrMerge}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{?\underline{K}\,N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{?\underline{K}\,N} Q'}{\Psi \rhd P \mid Q \xrightarrow{?\underline{K}\,N} P' \mid Q'}$$

$$\textsc{BrCom}\ \frac{\Psi_Q \otimes \Psi \rhd P \xrightarrow{!\overline{K}\,(\nu\widetilde{a})N} P' \qquad \Psi_P \otimes \Psi \rhd Q \xrightarrow{?\underline{K}\,N} Q'}{\Psi \rhd P \mid Q \xrightarrow{!\overline{K}\,(\nu\widetilde{a})N} P' \mid Q'}\ \widetilde{a}\#Q$$

$$\textsc{BrClose}\ \frac{\Psi \rhd P \xrightarrow{!\overline{K}\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu b)P \xrightarrow{\tau} (\nu b)(\nu\widetilde{a})P'}\ \begin{array}{c} b \in \mathsf{n}(K) \\ b\#\Psi \end{array}$$

Table 2. Operational broadcast semantics. A symmetric version of BrCom is elided. In rules BrCom and BrMerge we assume that $\mathcal{F}(P) = (\nu\widetilde{b_P})\Psi_P$ and $\mathcal{F}(Q) = (\nu\widetilde{b_Q})\Psi_Q$ where $\widetilde{b_P}$ is fresh for $P, \widetilde{b_Q}, Q, K$ and Ψ, and that $\widetilde{b_Q}$ is fresh for $Q, \widetilde{b_P}, P, K$ and Ψ.

Definition 6 (Requirements for broadcast).

1. $\Psi \vdash M \mathrel{\dot{\prec}} K \implies \mathsf{n}(M) \supseteq \mathsf{n}(K)$
2. $\Psi \vdash K \mathrel{\dot{\succ}} M \implies \mathsf{n}(K) \subseteq \mathsf{n}(M)$

Definition 7 (Transitions of Broadcast Psi). *To the actions of psi-calculi we add broadcast input, written $?\underline{K}\,N$ for a reception of N on K, and broadcast output, written $!\overline{K}\,(\nu\widetilde{a})N$ for a broadcast of N on K, with names $\widetilde{a}$ fresh in K. As before, we omit $(\nu\widetilde{a})$ when $\widetilde{a}$ is empty, and in examples we omit N when it is not relevant. The transitions of well-formed agents are defined inductively in Tables 2 and 1, where we let α range over both unicast and broadcast actions.*

The rule BrOut, allows transmission on a broadcast channel K that the subject M of an output prefix is out-connected to. Similarly, the rule BrIn allows input from a broadcast channel K that the subject M of an input prefix is in-connected to. When two parallel processes both receive a broadcast on the same channel, the rule BrMerge combines the two actions. This rule is necessary to ensure the associativity of parallel composition. After a broadcast communication using BrCom, the resulting action is the original transmission. This is different from the unicast Com rule, where a communication yields an internal action τ. Finally, rule BrClose states that a broadcast transmission does not reach beyond its scope. This allows for broadcasting on restricted channels. Dually, the Res rule (of Table 1) ensures that broadcast receivers on restricted channels cannot proceed unless a message is sent. We allow the Open rule to also apply to broadcast output actions, in order to communicate scoped data. The Par rule allows for broadcasts to bypass a process, as in most other broadcast calculi for wireless systems.

We have developed a meta-theory for broadcast psi-calculi. In the following we restrict attention to well-formed agents. The expected compositionality properties of strong bisimilarity hold:

Theorem 8 (Congruence properties of strong bisimulation). *For all Ψ:*

$$\begin{array}{rcl}
P \mathrel{\dot{\sim}_\Psi} Q & \Longrightarrow & P \mid R \mathrel{\dot{\sim}_\Psi} Q \mid R \\
P \mathrel{\dot{\sim}_\Psi} Q & \Longrightarrow & (\nu a)P \mathrel{\dot{\sim}_\Psi} (\nu a)Q \qquad \textit{if } a \# \Psi \\
P \mathrel{\dot{\sim}_\Psi} Q & \Longrightarrow & !P \mathrel{\dot{\sim}_\Psi} !Q \qquad \textit{if } P, Q \textit{ assertion guarded} \\
\forall i. P_i \mathrel{\dot{\sim}_\Psi} Q_i & \Longrightarrow & \mathbf{case}\ \widetilde{\varphi} : \widetilde{P} \mathrel{\dot{\sim}_\Psi} \mathbf{case}\ \widetilde{\varphi} : \widetilde{Q} \\
P \mathrel{\dot{\sim}_\Psi} Q & \Longrightarrow & \overline{M}\, N \,.\, P \mathrel{\dot{\sim}_\Psi} \overline{M}\, N \,.\, Q \\
(\forall \widetilde{L}.\ P[\widetilde{x} := \widetilde{L}] \mathrel{\dot{\sim}_\Psi} Q[\widetilde{x} := \widetilde{L}]) & \Longrightarrow & \underline{M}(\lambda \widetilde{x})N \,.\, P \mathrel{\dot{\sim}_\Psi} \underline{M}(\lambda \widetilde{x})N \,.\, Q
\end{array}$$

As usual in channel-passing calculi, bisimulation is not a congruence for input prefix. We can characterise strong bisimulation congruence in the usual way.

Definition 9 (Strong Congruence). *$P \sim_\Psi Q$ iff for all sequences σ of substitutions it holds that $P\sigma \mathrel{\dot{\sim}_\Psi} Q\sigma$. We write $P \sim Q$ for $P \sim_{\mathbf{1}} Q$.*

Theorem 10. *Strong congruence $\sim_\Psi$ is a congruence for all Ψ.*

The standard structural laws hold for strong congruence.

Theorem 11 (Structural equivalence). *Assume that $a \# Q, \widetilde{x}, M, N, \widetilde{\varphi}$. Then*

$$\begin{array}{cc}
\mathbf{case}\ \widetilde{\varphi} : \widetilde{(\nu a)P} \sim (\nu a)\mathbf{case}\ \widetilde{\varphi} : \widetilde{P} & (\nu a)\mathbf{0} \sim \mathbf{0} \\
\underline{M}(\lambda \widetilde{x})N \,.\, (\nu a)P \sim (\nu a)\underline{M}(\lambda \widetilde{x})(N) \,.\, P & Q \mid (\nu a)P \sim (\nu a)(Q \mid P) \\
\overline{M}\, N \,.\, (\nu a)P \sim (\nu a)\overline{M}\, N \,.\, P & (\nu b)(\nu a)P \sim (\nu a)(\nu b)P \\
P \mid (Q \mid R) \sim (P \mid Q) \mid R & !P \sim P \mid !P \\
P \mid Q \sim Q \mid P & P \sim P \mid \mathbf{0}
\end{array}$$

Theorems 8, 10 and 11 give us assurance that any broadcast psi-calculus has a compositional labelled bisimilarity that respects important structural laws. The proofs [21] are formally verified in the interactive theorem prover Isabelle/Nominal. The full formalisation of broadcast psi-calculi amounts to ca 33000 lines of Isabelle code, of which about 21000 lines are re-used from our earlier work [4]. The fact that the BrComm rule defers the closing of the communication to BrClose causes most of the added complications.

4 Modelling network topology changes

When modelling wireless protocols, one important concern is dealing with connectivity changes. We here give a general description of a method of modelling different connectivity configurations using assertions.

The idea is to allow for different generations of assertions by tagging each part of an assertion with a generation number. Only the most recent generation is used; a generation is made obsolete by adding an assertion from a later generation. We here consider broadcast connectivity, but this technique can also be used in other scenarios where there is a need to retract assertions.

In the following we assume a set of broadcast terms $\mathbf{B} \subseteq \mathbf{T}$; we let B, B' range over elements of $\mathbf{B}$. For simplicity, we assume that no rewriting happens in broadcast output, i.e., that $\dot{\prec}$ is the equality relation of $\mathbf{B}$. Assertions are finite sets of connectivity information, labelled with a generation, with set union as assertion composition $\otimes$ and the empty set as the unit assertion. Formally,

$$\begin{aligned}\mathbf{C} &\triangleq \{\mathsf{currentGeneration}(i) : i \in \mathbb{N}\} \cup \\ & \{K \dot{\succ} M : K, M \in \mathbf{T}\} \cup \{M \dot{\prec} K : K, M \in \mathbf{T}\} \\ \mathbf{A} &\triangleq \mathcal{P}_{\text{fin}}(\{\langle i, K \dot{\succ} M\rangle : i \in \mathbb{N},\ K, M \in \mathbf{T}\} \cup \{\langle i, 0\rangle : i \in \mathbb{N}\})\end{aligned}$$

$\Psi \vdash \mathsf{currentGeneration}(i)$ if $\forall \langle j, *\rangle \in \Psi \,.\, j \leq i$ and $\exists \langle j, *\rangle \in \Psi \,.\, i = j$
where $*$ is $B \dot{\succ} B'$ or 0
$\Psi \vdash B \dot{\prec} B'$ if $B = B'$
$\Psi \vdash B \dot{\succ} B'$ if $\langle i, B \dot{\succ} B'\rangle \in \Psi$ and $\mathsf{n}(B) \subseteq \mathsf{n}(B')$ and $\Psi \vdash \mathsf{currentGeneration}(i)$

The condition $\mathsf{currentGeneration}(i)$ is used to test if i is the most recent generation. The assertion $\{\langle i, B \dot{\succ} B'\rangle\}$ states that B' is in-connected to B in generation i if $\mathsf{n}(B) \subseteq \mathsf{n}(B')$, while the assertion $\{\langle i, 0\rangle\}$ states that nothing is connected in generation i.

As an example, we can define a topology controller (assuming a suitable encoding of the τ prefix):

$$T = (\!|\{\langle 1, 0\rangle\}|\!) \mid \tau \,.\, \left((\!|\{\langle 2, K \dot{\succ} M\rangle, \langle 2, K \dot{\succ} N\rangle\}|\!) \mid \tau \,.\, ((\!|\{\langle 3, K \dot{\succ} M\rangle\}|\!))\right)$$

In the process $P \mid T$, P can broadcast on K while T manages the topology. Initially $\mathcal{F}(T) = \{\langle 1, 0\rangle\}$ and the broadcast is disconnected; after $T \xrightarrow{\tau} T'$ then $\mathcal{F}(T') = \{\langle 1, 0\rangle, \langle 2, K \dot{\succ} M\rangle, \langle 2, K \dot{\succ} N\rangle\}$ and a broadcast on K can be received on both M and N, and after $T' \xrightarrow{\tau} T''$ then a broadcast can be received only on M, since $\mathcal{F}(T'') = \{\langle 1, 0\rangle, \langle 2, K \dot{\succ} M\rangle, \langle 2, K \dot{\succ} N\rangle, \langle 3, K \dot{\succ} M\rangle\}$.

5 The LUNAR protocol in Psi

In this section we present a model of the LUNAR routing protocol for mobile ad-hoc networks [24,25]. LUNAR is intended for small wireless networks, ca 15 nodes, with a network diameter of 3 hops. It does not handle route reparation, caching etc, and routes must be re-established every few seconds. It is reasonably simple in comparison to many other ad-hoc routing protocols, and allows us to focus on properties such as dynamic connectivity and broadcasting. It has previously been verified in [28,27] using SPIN and UPPAAL; our model is significantly shorter and at an abstraction level closer to the specification.

The LUNAR protocol is at "layer 2.5", between the link and network layers in the Internet protocol stack. Addressing is by pairs of MAC/Ethernet addresses and 64-bit selectors, similarly to the IP address and port number used in UDP/TCP. The selectors are used to find the appropriate packet handler through the FIB (Forwarding Information Base) table.

Below, we define a psi-calculus for modelling the LUNAR protocol. In an effort to keep our model simple we abstract from details such as TTL fields in messages, optional protocol fields, globally unique host identifiers, etc. We do not deal with time at all.

Channels are of two kinds: broadcast channels are terms node_i with (for simplicity) empty support, whose connectivity is given by the $\dot{\succ}$ and $\dot{\prec}$ predicates as defined in Section 4, and unicast channels which are pairs $\langle \mathit{sel}, \mathit{mac}\rangle$ where sel is a selector name and mac is a MAC address name. The mac part can also be a $\mathsf{RouteOf}(\mathit{node}, \mathit{ip})$ construction, which looks up the route of an IP address ip in the routing table of the node node. Special channels $\langle \mathsf{delivered}, \mathsf{node}_i\rangle$ are used to signal delivery of a packet to the IP layer. Assertions record requests originated at the local node using $\mathsf{Redirected}(\mathit{node}, \mathit{sel})$ and specify found routes using $\mathsf{HaveRoute}(\mathit{node}, \mathit{destip}, \mathit{hops}, \mathit{sel})$. The conditions contain predicates for testing if a route has been found ($\mathsf{HaveRoute}(\mathit{node}, \mathit{ip})$), if a selector has been used for a request originating at the local node ($\mathsf{Redirected}(\mathit{node}, \mathit{sel})$), and to extract the forwarder of a route ($\langle x, \mathsf{RouteOf}(\mathit{node}, \mathit{ip})\rangle \dot{\leftrightarrow} \langle x, \mathit{ip}\rangle$).

LUNAR protocol messages are of two types. The first is a route request message $\mathsf{RREQ}(\mathit{selector}, \mathit{targetIP}, \mathit{replyTo})$, where the $\mathit{selector}$ identifies the request, $\mathit{targetIP}$ is the IP address the route should reach, and $\mathit{replyTo}$ is the $\langle \mathit{sel}, \mathit{mac}\rangle$ channel the response should be sent to. The second is a route reply message, $\mathsf{RREP}(\mathit{hops}, \mathit{fwdptr})$), where hops is the number of hops to the destination, and fwdptr is a forwarding pointer, i.e. a $\langle \mathit{sel}, \mathit{mac}\rangle$ channel where packets can be sent.

The parameters of the psi-calculus for LUNAR extend the general topology psi-calculus in Section 4 as follows. The sets $\mathbf{T}, \mathbf{C}$ and $\mathbf{A}$ recursively include terms in order to be closed under substitution of terms for names.

$$\begin{aligned}
\mathbf{T} &\triangleq \mathcal{N} \cup \{\mathsf{node}_i : i \in \mathbb{N}\} \cup \{\mathsf{delivered}\} \cup \\
&\quad\ \{\mathsf{RREQ}(\mathit{Ser}, \mathit{TargIp}, \mathit{Rep}) : \mathit{Ser},\ \mathit{TargIp},\ \mathit{Rep} \in \mathbf{T}\} \cup \\
&\quad\ \{\mathsf{RREP}(i, \mathit{Fwd}) : i,\ \mathit{Fwd} \in \mathbf{T}\} \cup \\
&\quad\ \{\mathsf{RouteOf}(\mathit{Node}, \mathit{Ip}) : \mathit{Node},\ \mathit{Ip} \in \mathbf{T}\} \cup \\
&\quad\ \{\langle \mathit{Sel}, N\rangle : \mathit{Sel}, N \in \mathbf{T}\} \cup \{N+1 : N \in \mathbf{T}\} \cup \{0\} \\
\mathbf{C} &\triangleq \{M = N, \mathsf{HaveRoute}(M,N), \mathsf{Redirected}(M,N) : M, N \in \mathbf{T}\} \\
\mathbf{A} &\triangleq \mathcal{P}_{\mathrm{fin}}(\{\mathsf{HaveRoute}(M, N_1, i, N_2) : i,\ M, N_1, N_2 \in \mathbf{T}\} \cup \\
&\qquad\quad \{\mathsf{Redirected}(M,N) : M, N \in \mathbf{T}\})
\end{aligned}$$

$$\begin{aligned}
\Psi &\vdash a = a,\ a \in \mathcal{N} \\
\Psi &\vdash \langle a, b\rangle \dot{\leftrightarrow} \langle a, b\rangle,\ a, b \in \mathcal{N} \\
\Psi &\vdash \langle \mathsf{delivered}, \mathsf{node}_i\rangle \dot{\leftrightarrow} \langle \mathsf{delivered}, \mathsf{node}_i\rangle,\ i \in \mathbb{N} \\
\Psi \cup \{\mathsf{HaveRoute}(\mathsf{node}_i, a, j, b)\} &\vdash \langle \mathsf{RouteOf}(\mathsf{node}_i, a), x\rangle \dot{\leftrightarrow} \langle b, x\rangle \\
\Psi \cup \{\mathsf{HaveRoute}(\mathsf{node}_i, a, j, b)\} &\vdash \mathsf{HaveRoute}(\mathsf{node}_i, a) \\
\Psi \cup \{\mathsf{Redirected}(\mathsf{node}_i, s)\} &\vdash \mathsf{Redirected}(\mathsf{node}_i, s) \\
\Psi &\vdash \neg\varphi \text{ if } \neg(\Psi \vdash \varphi)
\end{aligned}$$

Figures 1-7 describe our psi-calculus model of the LUNAR protocol. We use process identifiers to improve the readability of the model. Process identifiers and recursion can be encoded in a standard fashion using replication, see e.g. [22]. In this section we use process declarations of the form $M(\widetilde{N}) \Leftarrow P$, where M is a process identifier (and also a term, implicitly included in $\mathbf{T}$), $\widetilde{N}$ a list of terms where occurrences of names are binding, and P is a process s.t. $\mathsf{n}(P) \subseteq \mathsf{n}(\widetilde{N})$. In a process, we write $M(\widetilde{N})$ for invoking a process declaration $M(\widetilde{K}) \Leftarrow P$ such that $\widetilde{N} = \widetilde{K}[\widetilde{x} := \widetilde{L}]$ with $\widetilde{x} = \mathsf{n}(\widetilde{K})$, resulting in the process $P[\widetilde{x} := \widetilde{L}]$. We write **if** φ **then** P **else** Q for **case** $\varphi : P \ [] \ \neg\varphi : Q$, and assume a suitable encoding of the τ prefix.

Our model of the protocol closely follows the informal protocol description in [25, Section 4]. Each figure in our model corresponds quite directly to one or more of part 0-5 of the protocol description. To allocate a selector, we simply bind a name; to associate (or bind) a selector to a packet handler we use a replicated process which receives on the unicast channel described by the pair of the selector and our MAC address (see e.g. the second line of the LunARP process declaration in Figure 1). In the informal protocol description [25], the FIB is "abused" by installing a null packet handler for the selector created when sending a route request. This FIB entry is only used to detect and avoid circular forwarding of route requests. We model this by an explicit assertion $\mathsf{Redirected}$ and a matching condition. The routing table is modelled using assertions, to show how these can be used as a global data structure. For simplicity we do not model route timeouts and the deletion of routes, but this could be done using the mechanism in Section 4.

The LUNAR procedure for route discovery starts when a node wants to send a message to a node it does not already have a route to (Figure 7, **else** branch). It then (Figure 1) associates a fresh selector with a response packet handler, and broadcasts a Route Request (RREQ) message to its neighbours. A node which receives a RREQ message (Figure 2) for its own IP address sets up a packet handler to deliver IP packets, and includes the corresponding selector in a response Route Reply (RREP) message to the reply channel found in the RREQ message. If the RREQ message was not for its own IP address, the message is re-broadcast after replacing the reply channel with a freshly allocated reply selector and its own MAC address. When such an intermediary node receives a RREP message (Figure 3), it increments the hop counter and forwards the RREP message to the source of the original RREQ message. When the originator of a RREQ message eventually receives the matching RREP (Figure 4), it installs a route and informs the IP layer about it. The message can then be resent (Figure 7, **then** branch) and delivered (Figure 5) by unicast messages through the chain of intermediary forwarding nodes.

We show the basic correctness of the model by the following theorem, which in essence corresponds to the correct operation of an ad-hoc routing protocol [28, Definition 1]: if there is a path between two nodes, the protocol finds it, and it is possible to send packets along the path to the destination node.

$$\begin{array}{l}
\mathsf{LunARP}(\mathit{mynode},\mathit{mymac},\mathit{destip}) \Leftarrow\\
\quad (\nu \mathit{rchosen},\mathit{schosen})\\
\quad\left(\begin{array}{l}
\;!\,\underline{\langle \mathit{rchosen},\mathit{mymac}\rangle}(x)\,.\,\mathsf{SRrepHandler}(\mathit{mynode},\mathit{mymac},\mathit{destip},x)\\
\mid (\!|\mathsf{Redirected}(\mathit{mynode},\mathit{schosen})|\!)\\
\mid \overline{\mathit{mynode}}\langle \mathsf{RREQ}(\mathit{schosen},\mathit{destip},\langle \mathit{rchosen},\mathit{mymac}\rangle)\rangle\,.\,\mathbf{0}
\end{array}\right)
\end{array}$$

Fig. 1. Part 0: the initialisation step at the node that wishes to discover a route

$$\begin{array}{l}
\mathsf{RreqHandler}(\mathit{mynode},\mathit{mymac},\mathit{myip},\mathsf{RREQ}(\mathit{schosen},\mathit{destip},\mathit{repchn})) \Leftarrow\\
\quad \mathbf{if}\ \mathsf{Redirected}(\mathit{mynode},\mathit{schosen})\ \mathbf{then}\ \mathbf{0}\\
\quad \mathbf{else}\ \tau\,.\left(\begin{array}{l}
(\!|\mathsf{Redirected}(\mathit{mynode},\mathit{schosen})|\!)\mid\\
\mathbf{if}\ \mathit{destip}=\mathit{myip}\ \mathbf{then} \qquad\qquad\qquad \text{/* Part 2: Target found */}\\
\quad (\nu \mathit{rchosen})\\
\quad\left(\begin{array}{l}
\;!\,\underline{\langle \mathit{rchosen},\mathit{mymac}\rangle}(x)\,.\,\mathsf{IPdeliver}(x,\mathit{mynode})\\
\mid \overline{\mathit{repchn}}\langle \mathsf{RREP}(0,\langle \mathit{rchosen},\mathit{mymac}\rangle)\rangle\,.\,\mathbf{0}
\end{array}\right)\\
\mathbf{else}\\
\quad (\nu \mathit{rchosen})\\
\quad\left(\begin{array}{l}
\;!\,\underline{\langle \mathit{rchosen},\mathit{mymac}\rangle}(x)\,.\,\mathsf{IRrepHandler}(\mathit{mymac},\mathit{repchn},x)\\
\mid \overline{\mathit{mynode}}\langle \mathsf{RREQ}(\mathit{schosen},\mathit{destip},\langle \mathit{rchosen},\mathit{mymac}\rangle)\rangle\,.\,\mathbf{0}
\end{array}\right)
\end{array}\right)
\end{array}$$

Fig. 2. Part 1: RREQ packet handler, and Part 2: Target found branch

$$\begin{array}{l}
\mathsf{IRrepHandler}(\mathit{mymac},\mathit{repchn},\mathsf{RREP}(\mathit{hops},\mathit{fwdptr})) \Leftarrow\\
\quad (\nu \mathit{rchosen})\\
\quad\left(\begin{array}{l}
\;!\,\underline{\langle \mathit{rchosen},\mathit{mymac}\rangle}(x)\,.\,\overline{\mathit{fwdptr}}\,x\,.\,\mathbf{0}\\
\mid \overline{\mathit{repchn}}\langle \mathsf{RREP}(\mathit{hops}+\mathit{1},\langle \mathit{rchosen},\mathit{mymac}\rangle)\rangle\,.\,\mathbf{0}
\end{array}\right)
\end{array}$$

Fig. 3. Part 3: Intermediate RREP packet handler

$$\begin{array}{l}
\mathsf{SRrepHandler}(\mathit{mynode},\mathit{mymac},\mathit{destip},\mathsf{RREP}(\mathit{hops},\mathit{fwdptr})) \Leftarrow\\
\quad (\nu \mathit{rchosen})\\
\quad\left(\begin{array}{l}
\;!\,\underline{\langle \mathit{rchosen},\mathit{mymac}\rangle}(x)\,.\,\overline{\mathit{fwdptr}}\,x\,.\,\mathbf{0}\\
\mid (\!|\mathsf{HaveRoute}(\mathit{mynode},\mathit{destip},\mathit{hops},\mathit{rchosen})|\!)
\end{array}\right)
\end{array}$$

Fig. 4. Part 4: Source RREP packet handler

$$\mathsf{IPdeliver}(x,\mathit{node}) \Leftarrow \overline{\langle \mathsf{delivered},\mathit{node}\rangle}\,x\,.\,\mathbf{0}$$

Fig. 5. Part 5: IP delivery

$$\begin{array}{l}
\mathsf{BrdHandler}(\mathit{mynode},\mathit{mac},\mathit{ip}) \Leftarrow\\
\quad \underline{\mathit{mynode}}(\lambda s,t,r)\mathsf{RREQ}(s,t,r)\,.\left(\begin{array}{l}
\;\mathsf{RreqHandler}(\mathit{mynode},\mathit{mac},\mathit{ip},\mathsf{RREQ}(s,t,r))\\
\mid \mathsf{BrdHandler}(\mathit{mynode},\mathit{mac},\mathit{ip})
\end{array}\right)
\end{array}$$

Fig. 6. Broadcast handler

$$\begin{array}{l}
\mathsf{IPtransmit}(\mathit{mynode},\mathit{mymac},\mathit{destip},\mathit{pkt}) \Leftarrow\\
\quad \mathbf{if}\ \mathsf{HaveRoute}(\mathit{mynode},\mathit{destip})\ \mathbf{then}\ \overline{\langle \mathsf{RouteOf}(\mathit{mynode},\mathit{destip}),\mathit{mymac}\rangle}\,\mathit{pkt}\,.\,\mathbf{0}\\
\quad \mathbf{else}\ \mathsf{LunARP}(\mathit{mynode},\mathit{mymac},\mathit{destip})
\end{array}$$

Fig. 7. IP transmission: if have route, send it to local forwarder, else ask for route

The system to analyse consists of n nodes with their respective broadcast handler; node 0 attempts to transmit a packet to the IP address of node n.

$$\mathsf{Spec}_n(pkt, ip_0, \ldots, ip_n) \Leftarrow (\nu mac_0, \ldots, mac_n) \begin{pmatrix} \prod_{0 \le i \le n} \mathsf{BrdHandler}(\mathsf{node}_i, mac_i, ip_i) \\ \mid\, !\, \mathsf{IPtransmit}(\mathsf{node}_0, mac_0, ip_n, pkt) \end{pmatrix}$$

Theorem 12. *If Ψ connects node_0 and node_n via a node node_i (i.e. $\Psi \vdash \mathsf{node}_0 \dot{\succ} \mathsf{node}_i$ and $\Psi \vdash \mathsf{node}_i \dot{\succ} \mathsf{node}_n$), then*

$$\Psi \mid (\nu ip_0, \ldots, ip_n)\mathsf{Spec}_n(pkt, ip_0, \ldots, ip_n)$$
$$\Longrightarrow \xrightarrow{\overline{\langle \mathsf{delivered}, \mathsf{node}_n \rangle} pkt} \Psi \mid (\nu ip_0, \ldots, ip_n) S$$

and $\mathcal{F}(S) \vdash \mathsf{HaveRoute}(\mathsf{node}_0, ip_n)$, where $\Longrightarrow$ stands for an interleaving of τ and broadcast output transitions.

Proof. By following transitions.

Our analysis is limited to a two-hop configuration due to the labour of manually following transitions in a non-trivial specification. We anticipate this can be automated using a future extension of our symbolic semantics for psi-calculi [10,11].

The definition of BrdHandler illustrates a peculiarity of broadcast semantics: a reader well-versed in pi-calculus specifications with replication and recursion may consider a more concise variant of the definition using replication instead of recursion, e.g.

$$\mathsf{BrdHandler}'(mynode, mac, ip) \Leftarrow$$
$$!\, \underline{mynode}(\lambda s, t, r)\mathsf{RREQ}(s, t, r) \,.\, \mathsf{RreqHandler}(mynode, mac, ip, \mathsf{RREQ}(s, t, r))$$

When the input prefix is over a broadcast channel, as is the case here, the two are not equivalent since a single communication with BrdHandler′ may result in arbitrarily many RreqHandler processes, while BrdHandler only results in one.

6 Related work

Process calculi with broadcast communication go back to the early 1980's. Milner developed SCCS [16] as a generalisation of CCS [15] to include multiway communication, of which broadcast can be seen as a special case. At the same time Austry and Boudol presented MEIJE [2] as a semantic basis for high-level hardware definition languages.

The first process calculus to seriously consider broadcast with an asynchronous parallel composition was CBS [19,20]. Its development is recorded in a series of papers, examining it from many perspectives. The main focus is on employing broadcast as a high level programming paradigm. CBS was later extended to the pi-calculus in the bπ formalism [5]. Here the broadcast communication channels are names that can be scoped and transmitted between agents.

The main point of this work is to establish a separation result in expressiveness: in the pi-calculus, broadcast cannot be uniformly encoded by unicast.

Recent advances in wireless networks have created a renewed interest in the broadcast paradigm. The first process calculus with this in mind was probably CBS$^\sharp$ [17]. This is a development of CBS to include varying interconnection topologies. Input and output is performed on a universal ether and transitions are indexed with topologies which are sets of connectivity graphs; the connectivity graph matters for the input rule (reception is possible from any connected location). Main applications are on cryptography and routing protocols in mobile ad hoc wireless networks. CBS$^\sharp$ has been followed by several similar calculi. In CWS [14,12] the focus is on modelling low level interference. Communication actions have distinct beginnings and endings, and two actions may interfere if one begins before another has ended. The main result is an operational correspondence between a labelled semantics and a reduction semantics. CMAN [8] is a high level formalism extended with data types, just as the applied pi-calculus extends the original pi-calculus. Data can contain constructors and destructors. There are results on properties of weak bisimulation and an analysis of a cryptographic routing protocol. In the ω-calculus [23] emphasis is on expressing connectivity using sets of group names. An extension also includes separate unicast channels, making this formalism the first to accommodate both multicast and unicast. There are results about strong bisimulation and a verification of a mobile ad hoc network leader election protocol through weak bisimulation. RBPT [7] is similar and uses an alternative technique to represent topology changes, leading to smaller state spaces, and is also different in that it can accommodate an asymmetric neighbour relation (to model the fact that A can send to B but not the other way).

$bA\pi$ [9] is an extension of the applied pi-calculus [1] with broadcast, where connectivity information appears explicitly in the process terms and can change non-deterministically during execution. The claimed result of the paper is proving that a weak labelled bisimulation, for which connectivity is irrelevant, coincides with barbed equivalence. However, for the same reasons as in the applied pi-calculus (cf. [3]), labelled bisimilarity is not compositional in $bA\pi$, so the correspondence does not hold. A suggested fix is to remove unicast channel mobility from the calculus. We would finally mention CMN [13]. The claimed result is to compare two different kinds of semantics for a broadcast operation, but it is in error. The labelled transition semantics contains no rule for merging two inputs as in our BrMerge. As a consequence parallel composition fails to be associative. Consider the situation where P does an output and Q and R both do inputs. A broadcast communication involving all three agents can be derived from $(P|Q)\,|\,R$ but not from $P\,|\,(Q|R)$, since in the latter agent the component $Q|R$ cannot make an input involving both Q and R.

It is interesting to compare these formalisms and our broadcast psi from a few important perspectives. Firstly, the broadcast channels are explicitly represented in ω, bπ, CWS and CMN; they are mobile (in the sense that they can be transmitted) only in bπ. In ω, only unicast channels can be communicated.

In broadcast psi, channels are represented as arbitrary mobile data terms which may contain any number of names. Secondly, the data transmitted in CMAN and $bA\pi$ is akin to the applied pi-calculus where data are drawn from an inductively defined set and contain names which may be scoped. In ω and bπ data are single names which may be scoped; in the other calculi data cannot contain scoped names. In broadcast psi data are arbitrary terms, drawn from a nominal set, and may include higher order objects as well as bound names. Finally, node mobility is represented explicitly as particular semantic rules in CMAN, CMN, $bA\pi$ and ω, and implicitly in the requirements of bisimulation in CBS$^\sharp$ and RBPT. In this respect broadcast psi calculi are similar to the latter: connectivity is determined by the assertions in the environment, and in a bisimulation these may change after each transition.

All calculi presented here use a kind of labelled transition semantics (LTS). bπ, $bA\pi$, CBS$^\sharp$, CWS and ω use it in conjunction with a structural congruence (SC), the rest (including broadcast psi) do not use a SC. In our experience SC is efficient in that the definitions become more compact and easy to understand, but introduces severe difficulties in making fully rigorous proofs. $bA\pi$, CWS, CMAN and CMN additionally use a reduction semantics using structural congruence (RS) and prove its agreement with the labelled semantics. Table 3 summarises some of the distinguishing features of calculi for wireless networks.

Calculus	Broadcast Channels	Scoped Data	Mobility	Semantics
$bA\pi$	-	term	in semantics	LTS+SC and RS
CBS$^\sharp$	-	-	in bisimulation	LTS+SC
CWS	constant	-	-	LTS+SC and RS
CMAN	-	term	in semantics	LTS and RS
CMN	name	-	in semantics	LTS and RS
ω	groups	name	in semantics	LTS+SC
RBPT	-	-	in bisimulation	LTS
Broadcast psi	term	term	in bisimulation	LTS

Table 3. Comparison of some process algebras for wireless broadcast.

Finally, broadcast psi is different from the other calculi for wireless broadcast in that there is no stratification of the syntax into processes and networks. There is just the one kind of agent, suitable for expressing both processes operating in nodes and behaviours of entire networks. In contrast, the other calculi has one set of constructs to express processes and another to express networks, sometimes leading to duplication of effort (for example, there can be a parallel composition operator both at the process and network level). Our conclusion is that broadcast psi is conceptually simpler and more efficient for rigorous proofs, and yet more expressive.

7 Conclusion

We have extended the psi-calculi framework with broadcast communication, and formally proved using Isabelle/Nominal that the standard congruence and structural properties of bisimilarity hold also after the addition. We have shown how node mobility and network topology changes can be modelled using assertions. Since bisimilarity is closed under all assertions, two bisimilar processes are equivalent in all initial topologies and for all node mobility patterns. We demonstrated expressive power by modelling the LUNAR protocol for route discovery in wireless ad-hoc networks, and verified a basic correctness property of the protocol.

The model of LUNAR is simplified for clarity and to make manual analysis more manageable. The simplifications are similar to those in the SPIN model by Wibling et al. [28], although we do not model timeouts. Their model [27] is ca 250 lines of SPIN code (excluding comments) while ours is approximately 30 lines. Our model could be improved at the cost of added complexity. For example, allowing broadcast channels to have non-empty support would let us hide broadcast actions, routing tables could be made local by including a scoped name per node, and route deletions could be modelled using generational mechanisms similar to Section 4.

We intend to extend the symbolic semantics for psi-calculi [10,11] with broadcast, and implement the semantics in a tool for automatic verification. We also plan to study weak bisimulation for the broadcast semantics. In order to model more aspects of wireless protocols, we would like to add general resource awareness (e.g. energy or time) to psi-calculi.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Proceedings of POPL '01*, pages 104–115. ACM, 2001.
2. D. Austry and G. Boudol. Algèbre de processus et synchronisation. *Theor. Comput. Sci.*, 30:91–131, 1984.
3. J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *Proceedings of LICS 2009*, pages 39–48. IEEE, 2009.
4. J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: A framework for mobile processes with nominal data and logic. *Logical Methods in Computer Science*, 2011. Accepted for publication. This is an extended version of [3].
5. C. Ene and T. Muntean. Expressiveness of point-to-point versus broadcast communications. In G. Ciobanu and G. Paun, editors, *FCT*, volume 1684 of *LNCS*, pages 258–268. Springer, 1999.
6. M. Gabbay and A. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.
7. F. Ghassemi, W. Fokkink, and A. Movaghar. Restricted broadcast process theory. In A. Cerone and S. Gruner, editors, *SEFM*, pages 345–354. IEEE Computer Society, 2008.
8. J. C. Godskesen. A calculus for mobile ad hoc networks. In A. L. Murphy and J. Vitek, editors, *COORDINATION*, volume 4467 of *LNCS*, pages 132–150. Springer, 2007.

9. J. C. Godskesen. Observables for mobile and wireless broadcasting systems. In *Proc. of COORDINATION 2010*, volume 6116 of *LNCS*, pages 1–15. Springer, 2010.
10. M. Johansson, B. Victor, and J. Parrow. A fully abstract symbolic semantics for psi-calculi. In *Proceedings of SOS 2009*, volume 18 of *EPTCS*, pages 17–31, 2010.
11. M. Johansson, B. Victor, and J. Parrow. Computing strong and weak bisimulations for psi-calculi. Submitted for publication, 2011.
12. I. Lanese and D. Sangiorgi. An operational semantics for a calculus for wireless systems. *Theor. Comp. Sci.*, 411(19):1928–1948, 2010.
13. M. Merro. An observational theory for mobile ad hoc networks (full version). *Inf. Comput.*, 207(2):194–208, 2009.
14. N. Mezzetti and D. Sangiorgi. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.
15. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.
16. R. Milner. Calculi for synchrony and asynchrony. *Theor. Comput. Sci.*, 25:267–310, 1983.
17. S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theor. Comp. Sci.*, 367(1-2):203–227, 2006.
18. A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.
19. K. V. S. Prasad. A calculus of broadcasting systems. In S. Abramsky and T. S. E. Maibaum, editors, *TAPSOFT, Vol.1*, volume 493 of *LNCS*, pages 338–358. Springer, 1991.
20. K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.
21. P. Raabjerg and J. Åman Pohjola. Broadcast psi-calculus formalisation. `http://www.it.uu.se/research/group/mobility/theorem/broadcastpsi`, July 2011. Isabelle/HOL-Nominal formalisation of the definitions, theorems and proofs.
22. D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
23. A. Singh, C. R. Ramakrishnan, and S. A. Smolka. A process calculus for mobile ad hoc networks. *Sci. Comput. Program.*, 75(6):440–469, 2010.
24. C. Tschudin, R. Gold, O. Rensfelt, and O. Wibling. LUNAR: a lightweight underlay network ad-hoc routing protocol and implementation. In *Proc of NEW2AN'04*, St. Petersburg, Feb. 2004.
25. C. F. Tschudin. Lightweight underlay network ad hoc routing (LUNAR) protocol. Internet Draft, Mobile Ad Hoc Networking Working Group, Mar. 2004.
26. C. Urban and C. Tasson. Nominal techniques in Isabelle/HOL. In R. Nieuwenhuis, editor, *Proceedings of CADE 2005*, volume 3632 of *LNCS*, pages 38–53. Springer, 2005.
27. O. Wibling. SPIN and UPPAAL ad hoc routing protocol models. `http://www.it.uu.se/research/group/mobility/adhoc/gbt/other_examples`, 2004. Models of LUNAR scenarios used in [28].
28. O. Wibling, J. Parrow, and A. Pears. Automatized verification of ad hoc routing protocols. In D. de Frutos-Escrig and M. Núñez, editors, *FORTE 2004*, volume 3235 of *LNCS*, pages 343–358. Springer, 2004.