Software and System Modeling manuscript No.
(will be inserted by the editor)

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]**, Joachim Parrow**[1]

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

July 11, 2013

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 earlier papers we have shown how psi-calculi can capture the same phenomena as other proposed extensions of the pi-calculus such as the applied pi-calculus, the spi-calculus, the fusion calculus, the concurrent constraint pi-calculus, and calculi with polyadic communication channels or pattern matching. Psi-calculi can be even more general, for example by allowing structured channels, higher-order formalisms such as the lambda calculus for data structures, and predicate logic for assertions [5].

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 [30]. The communication paradigm in psi-calculi is binary: for each event there is one sender and one receiver, just as in the pi-calculus.

In several areas, e.g. wireless communications and hardware data buses, a natural paradigm is broadcast, where one transmission can be received by several processes. Broadcast communication cannot be uniformly encoded in the pi-calculus [7].

In this paper we extend the psi-calculi framework with primitives for synchronous unreliable broadcast. These require new operational actions and rules, and new connectivity predicates. In Section 3.1, 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. One of our main contributions is precise requirements that the connectivity predicates must satisfy, in order to model scoped broadcasts with dynamic connectivity, while still satisfying the meta-theoretical results of Section 3.1.

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 [28]. 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 [32,31] (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.

This paper is an extended version of [6] that adds clarifications, proofs, and elaborated examples of dynamic topology management.

2 Psi-calculi

This section is a brief recapitulation of psi-calculi; for an extensive treatment including more motivations and examples see [4,5], from which some examples and explanations below are taken.

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. As a general framework for terms and other data containing names, we work in the formalism of *nominal sets* [21,8]. A nominal set is an ordinary 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 finite set of names we write $A\#X$ to mean $\forall a \in A \;.\; a\#X$. We require all elements to have finite support, i.e., $\mathrm{n}(X)$ is finite for all 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 $\widetilde{ab}$. 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. For names, we write $(\widetilde{a}\ \widetilde{b})$ for the *name swapping* that swaps each element of $\widetilde{a}$ with the corresponding element of $\widetilde{b}$; here it is implicit that $\widetilde{a}$ and $\widetilde{b}$ have the same length, and that the names in $\widetilde{a}$ (resp. $\widetilde{b}$) are pair-wise distinct. A function f is *equivariant* if $(a\ b) \cdot f(X) = f((a\ b) \cdot X)$ holds for all X, and similarly for functions and relations of any arity. Intuitively, this means that all names are treated equally.

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, $\tilde{a}$ is a sequence of names without duplicates and $\tilde{Y}$ is an equally long sequence of elements of possibly another datatype, the *substitution* $X[\tilde{a} := \tilde{Y}]$ is an element of the same datatype as X. Substitution is required to satisfy a law akin to alpha-conversion: if $\widetilde{b}\#X, \widetilde{a}$ then $X[\widetilde{a} := \widetilde{T}] = ((\widetilde{b}\ \widetilde{a}) \cdot X)[\widetilde{b} := \widetilde{T}]$. Intuitively, this ensures that substitutions for bound names yield the same result no matter which alpha-equivalent version is used.

The main point of using nominal datatypes is that we obtain a general framework, allowing many different instantiations. Our only requirements are on the notions of support, name swapping, and substitution. Thus we can handle datatypes that are not inductively defined, such as equivalences classes and sets defined by comprehension or co-induction. Examples include higher-order datatypes such as the lambda calculus. As long as it satisfies the axioms of a nominal datatype it can be used in our framework. Similarly, the notions of conditions, i.e., the tests on data that agents can perform during their execution, and assertions, i.e. the facts that can be used to resolve conditions, are formulated as nominal datatypes. This means that logics with binders and even higher-order logics can be used. Moreover, alpha-variants of terms can be formally equated by taking the quotient of terms under alpha equality, thereby facilitating the formalism and proofs.

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 $\mathbf{A}$, where the substitution function on $\mathbf{T}$, in addition to the alpha-conversion-like law above, satisfies the following name preservation law: if $\widetilde{a} \subseteq \mathsf{n}(M)$ and $b \in \mathsf{n}(\widetilde{N})$ then $b \in \mathsf{n}(M[\widetilde{a} := \widetilde{N}])$.

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 $(\Psi, \varphi) \in \vdash$.

As an example, we can choose data terms inductively generated by some signature, assertions and conditions to be elements of a first-order logic with equality over these terms, entailment to be logical implication, $\otimes$ to be conjunction and $\mathbf{1}$ to be TRUE.

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 straightforward requisites on the sets and operators. In brief, channel equivalence must be symmetric and transitive (but not necessarily reflexive), $\otimes$ must be compositional with regard to $\simeq$, and the assertions with $(\otimes, \mathbf{1})$ form an abelian monoid modulo $\simeq$. For details see [5].

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 $\Psi \otimes F$ to mean $(\nu\epsilon)\Psi \otimes F$, and $(\nu c)((\nu\widetilde{b})\Psi)$ 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 $G \vdash \varphi$. Intuitively a condition is entailed by a frame if it is entailed by the assertion and does not contain any names bound by the frame, and two frames are equivalent if they entail the same conditions.

For an example of first-order logic with equality, assume that the term $\mathsf{enc}(M, k)$ represents the encoding of message M with key k, and let Ψ be the assertion $C = \mathsf{enc}(M, k)$, stating that the ciphertext C is the result of encoding M by k. If an agent contains this assertion, the environment of the agent will be able to use it to resolve tests on the data. In particular it may infer that $C = \mathsf{enc}(M, k)$, i.e., it can test if this C is the encryption of M. Access to the key k can be restricted by enclosing it in a scope: if the environment instead has access to the assertion $(\nu k)\Psi$, it can *not* infer that C is the encoding of M (assuming conditions only contain equivalence tests on terms). For more discussion see [5].

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
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.*

In the Output and Input forms M is called the subject and N the object. Output and Input are similar to those in the pi-calculus, but arbitrary terms can function as both subjects and objects. In the input $\underline{M}(\lambda \widetilde{x})N.P$ the intuition is that the pattern $(\lambda \widetilde{x})N$ can match any term obtained by instantiating $\widetilde{x}$, e.g., $\underline{M}(\lambda x, y) f(x, y).P$ can only communicate with an output $\overline{M} f(N_1, N_2)$ for some data terms N_1, N_2. This can be thought of as a generalisation of the polyadic pi-calculus where the patterns are just tuples of names. Another significant extension is that we allow arbitrary data terms also as communication channels. Thus it is possible to include functions that create channels.

The **case** construct behaves as one of the P_i for which the corresponding φ_i is true. 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. Input subjects are underlined to facilitate parsing of complicated expressions; in simple cases we often omit the underline. We sometimes write $\underline{M}(x).P$ for $\underline{M}(\lambda x)x.P$.

One of the simplest examples of a psi-calculus is the pi-calculus [**?**], which can be represented using names as the only data terms, $\mathbf{1}$ as the only assertion, and equality tests on names as conditions. Channel equivalence $\dot{\leftrightarrow}$ is also equality on names. Substitution is the standard capture-avoiding syntactic replacement of names for names. Choice in the pi-calculus can be represented using the **case** statement: $P + Q$ corresponds to $(\nu a)($**case** $a = a \;:\; P \;[]\; a = a \;:\; Q)$, where $a \# P, Q$, and the pi-calculus match construct $[a = b]P$ corresponds to **if** $a = b$ **then** P. The formal correspondence between this psi-calculus instance and the original pi-calculus is proved in [5].

As indicated in the encryption example above, the conditions tested in a process are affected by the assertions of parallel processes. For example in $P \mid Q$, the assertions of P can affect the conditions tested in Q, and thereby its transitions. We introduce the *frame of an agent* as the combination of its top level assertions, retaining all the binders: this is precisely what can affect a parallel agent. The *frame* $\mathcal{F}(P)$ *of an agent* P is defined inductively as follows:

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

For a simple example, if $a\#\Psi_1$:

$$\mathcal{F}((\!|\Psi_1|\!) \mid (\nu a)((\!|\Psi_2|\!) \mid \overline{M}\,N.(\!|\Psi_3|\!)) \quad = \quad (\nu a)(\Psi_1 \otimes \Psi_2)$$

Here Ψ_3 occurs under a prefix and is therefore not included in the frame.

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. As in the pi-calculus, the output $\overline{M}(\nu\tilde{a})N$ represents an action sending N along M and opening the scopes of the names $\tilde{a}$. Note in particular that the support of this action includes $\tilde{a}$. Thus $\overline{M}(\nu a)a$ and $\overline{M}(\nu b)b$ are different 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.

The environmental assertions $\Psi \rhd \cdots$ in Table 1 express the effect that the environment has on the agent: enabling conditions in Case, giving rise to action subjects in In and Out and enabling interactions in Com. The environment Ψ increases towards the leafs of the derivation only in the rules for the parallel operator, where an agent is part of the environment for another agent. If all environmental assertions are erased and channel equivalence replaced by identity we get the standard laws of the pi-calculus enriched with data structures.

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

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

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

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

$$\textsc{Scope}\ \frac{\Psi \ \rhd\ P \ \xrightarrow{\alpha}\ P'}{\Psi \ \rhd\ (\nu b)P \ \xrightarrow{\alpha}\ (\nu b)P'}\ b\#\alpha, \Psi$$

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

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.

For a simple example of a transition, suppose for an assertion Ψ and condition φ that $\Psi \vdash \varphi$. Assume that

$$\forall \Psi'.\Psi' \ \rhd\ Q \ \xrightarrow{\alpha}\ Q'$$

i.e., Q has an action α regardless of the environment. Then by the Case rule we get

$$\Psi \ \rhd\ \mathbf{if}\ \varphi\ \mathbf{then}\ Q \ \xrightarrow{\alpha}\ Q'$$

i.e., **if** φ **then** Q has the same transition if the environment is Ψ. Since $\mathcal{F}((\!|\Psi|\!)) = \Psi$ and $\Psi \otimes \mathbf{1} = \Psi$, if $\mathrm{bn}(\alpha)\#\Psi$ we get by Par that

$$\mathbf{1} \ \rhd\ (\!|\Psi|\!) \mid \mathbf{if}\ \varphi\ \mathbf{then}\ Q \ \xrightarrow{\alpha}\ (\!|\Psi|\!) \mid Q'$$

The notion of strong bisimulation is used to formalise the intuition that two agents "behave in the same way".

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*

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

We define $P \mathrel{\dot{\sim}_{\Psi}} Q$ *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 a congruence in the usual sense: it is preserved by all operators except input prefix, and satisfies the expected algebraic laws such as scope extension $P \mid (\nu a)Q \mathrel{\dot{\sim}} (\nu a)(P \mid Q)$ if $a \# P$. For details see [4,5]. Note that these meta-theoretic results have been proven correct once and for all using the interactive theorem prover Isabelle/Nominal [30].

Psi-calculi can capture the same phenomena as a wide range of previously proposed individual extensions of the pi-calculus. Examples in [4,5] range from foundational calculi such as polyadic pi-calculus, polyadic synchronisation pi-calculus, fusion calculus, and concurrent constraint calculi, to applied calculi for cryptography and systems with frequency hopping communication protocols. Each of the previous pi-calculus extensions need new proofs of basic results such as scope extension and bisimulation congruence. Formulated as psi-calculus instances, all the meta-theory of psi-calculi is automatically inherited.

3 Broadcast psi-calculi

In this section we extend the unicast psi-calculi of the previous section with a communication paradigm for synchronous unreliable non-blocking broadcast (suitable for modelling wireless communication). We introduce the notion of a *broadcast channel* as an abstraction of relevant properties of the transmission, such as frequency, sender location and signal strength. Formally a broadcast channel is just a term. We assume so called *connectivity predicates* that regulate which prefix subjects can send on or receive from which broadcast channels. These predicates may depend on assertions and therefore change as an agent evolves.

As an example, assume that the connectivity information Ψ allows the sender M_0 to send on the broadcast channel K, and receivers M_1 and M_2 to listen on K. We would then have the following transition:

$$\Psi \rhd \overline{M_0}\, N.P \mid \underline{M_1}(x).Q \mid \underline{M_2}(y).R \xrightarrow{!\overline{K}\, N} P \mid Q[x := N] \mid R[y := N]$$

Here, in one action two processes both receive the N sent along K, and moreover the action label retains the broadcast output action $!\overline{K}N$, meaning that in a larger context even more processes could receive N.

Formally, we assume a psi-calculus with the following extra predicates:

Definition 5 (Extra predicates for broadcast)

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

The first predicate, $M \dot{\prec} K$, 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, $K \dot{\succ} M$, is pronounced "M is in-connected to K" and means that an input prefix $\underline{M}(\lambda\tilde{x})N$ 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 lookup in a routing table: if the term *tab* is a list of pairs of identifiers and channels we can let $\Psi \vdash \texttt{lookup}(\mathit{tab}, \mathit{id}) \dot{\prec} \mathit{ch}$ be true iff $(\mathit{id}, \mathit{ch})$ appears in the routing table *tab*. We can also model connectivity: if Ψ contains connectivity information between channels *ch* and receivers n we may let $\Psi \vdash \mathit{ch} \dot{\succ} \texttt{rcv}(n, \mathit{ch})$ 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 $M \dot{\prec} K$ might not be equivalent to $K \dot{\succ} M$. Instead, for technical reasons related to scope extension (cf. Example 13), broadcast channels must have no greater support than the input and output prefixes that send and receive on them.

Definition 6 (Requirements for broadcast)

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

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* $\overline{!K}\,(\nu\tilde{a})N$ *for a broadcast of* N *on* K*, with names* $\tilde{a}$ *fresh in* K*. As before, we omit* $(\nu\tilde{a})$ *when* $\tilde{a}$ *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. The environmental assertion Ψ determines if a prefix is connected to a broadcast channel and thus gives rise to a broadcast in BrIn and BrOut. In the same way it determines if a prefix is channel equivalent to something else and thus gives rise to a unicast in In and Out. The same prefix could theoretically be used for both kinds of communication, although it may be unusual to find situations where that would be useful.

When two parallel processes both receive a broadcast on the same channel, the rule BrMerge combines the two actions. This rule is necessary

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

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

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

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

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

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.

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. The BrOpen rule allows broadcast communication of data containing scoped names. Rule BrClose states that a broadcast transmission does not reach beyond its scope. This allows for broadcasting on restricted channels. Dually, the Scope rule (of Table 1) ensures that broadcast receivers on restricted channels cannot proceed unless a message is sent. The Par rule allows for broadcasts to bypass a process, as in most other broadcast calculi for wireless systems.

3.1 Meta-theory

We have developed a meta-theory for broadcast psi-calculi. 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 of these results are mostly straightforward extensions of the corresponding proofs for standard (unicast) psi-calculi [13,3], where some technical lemmas can be simplified because of the requirement of syntactic equality of channels in rules BrCom and BrMerge. Most of the added complications are caused by the fact that the BrCom rule defers the closing of the communication to BrClose; cf. Lemma 12. The proofs [24] are formally verified in the interactive theorem prover Isabelle/Nominal. The full formalisation of broadcast psi-calculi amounts to ca 33 000 lines of Isabelle code, of which about 21 000 lines are re-used from our earlier work [5].

In the following we restrict attention to well-formed agents.

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

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

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 $P\sigma \stackrel{\cdot}{\sim}_{\Psi} Q\sigma$. 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 rules of structural equivalence are sound for bisimilarity congruence.

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

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

When proving Theorem 11 we encountered an unusual complication in the proof of the commutativity of restriction, due to the BrClose rule. Since this rule can insert binder sequences under name restrictions, the simulation proof needs to allow for permutations of sequences of top-level binders. This is the main difference in our meta-theoretical proofs as compared to the original psi-calculi. We write $\widetilde{a} \equiv \widetilde{b}$ to denote that the sequence $\widetilde{a}$ is a rearrangement of $\widetilde{b}$, preserving the number of occurrences of each name.

Lemma 12 *For all Ψ, P, x, y, we have $(\nu y)(\nu x)P \stackrel{\cdot}{\sim}_{\Psi} (\nu x)(\nu y)P$.*

Proof In standard psi-calculi, the proof of this result uses the candidate relation $\mathcal{S}_0 \overset{def}{=} \{(\Psi, (\nu y)(\nu x)P, (\nu x)(\nu y)P) \,:\, x, y\#\Psi\}$. Here we inductively close this relation under restriction, yielding $\mathcal{S}$:

$$\mathcal{S} \overset{def}{=} \mathcal{S}_0 \cup \{(\Psi, (\nu a)P, (\nu a)Q) : (\Psi, P, Q) \in \mathcal{S} \wedge a\#\Psi\}$$

We show that $\mathcal{S}$ is a bisimulation up to transitivity [25] (at every Ψ). That is, we only require the derivatives after a simulation step to be related by $\mathcal{S}^*$, inductively defined as

$$\mathcal{S}^* \overset{def}{=} \{(\Psi, P, P)\} \cup \{(\Psi, P, R) \;:\; \exists Q.\ (\Psi, P, Q) \in \mathcal{S}^* \wedge (\Psi, Q, R) \in \mathcal{S}\}.$$

We have proven "up to transitivity" to be sound, i.e., every bisimulation up to transitivity is a subset of some ordinary bisimulation.

The interesting part of the proof is in the simulation clause. We here consider only the base case of the definition of $\mathcal{S}$ (i.e. $\mathcal{S}_0$), where we need to prove that for all α, P' such that $\mathrm{bn}(\alpha)\#\Psi, Q$ and $\Psi \rhd (\nu y)(\nu x)P \xrightarrow{\alpha} P'$ there exists a Q' such that $\Psi \rhd (\nu x)(\nu y)P \xrightarrow{\alpha} Q'$ and $(\Psi, P', Q') \in \mathcal{S}^*$.

We first define a relation $\mathcal{R}$ that safely approximates $\mathcal{S}^*$ (i.e. $\mathcal{R} \subseteq \mathcal{S}^*$) and is easier to work with.

$$\mathcal{R} \overset{def}{=} \{(\Psi, (\nu\widetilde{a})P, (\nu\widetilde{b})P) : \widetilde{a}\#\Psi \wedge \widetilde{a} \equiv \widetilde{b}\}.$$

By induction on the length of $\widetilde{a}$, we get that for all $\widetilde{a}, \widetilde{b}, \Psi, P$ such that $\widetilde{a}\#\Psi$ and $\widetilde{a} \equiv \widetilde{b}$ we have $(\Psi, (\nu\widetilde{a})P, (\nu\widetilde{b})P) \in \mathcal{S}^*$. From this follows that the relation $\mathcal{R} \subseteq \mathcal{S}^*$; in order to show that the derivatives $(\Psi, P', Q') \in \mathcal{S}$ after a simulation step, we instead prove $(\Psi, P', Q') \in \mathcal{R}$.
The simulation proof is by case analysis on the derivations of transitions of $(\nu y)(\nu x)P$. We here focus on on the following derivation.

$$\textsc{Scope}\ \dfrac{\textsc{BrClose}\ \dfrac{\Psi \rhd P \xrightarrow{\overline{!M}\ (\nu\widetilde{a})N} P'}{\Psi \rhd (\nu x)P \xrightarrow{\tau} (\nu x)(\nu\widetilde{a})P'}\ x \in \mathsf{n}(M), x\#\Psi}{\Psi \rhd (\nu y)(\nu x)P \xrightarrow{\tau} (\nu y)(\nu x)(\nu\widetilde{a})P'}\ y\#\tau, \Psi$$

We assume that $\widetilde{a}\#(\Psi, P, M, x, y)$. There are three cases to consider.

1. $y\#(\overline{!M}\ (\nu\widetilde{a})N)$: We have the following transition.

$$\textsc{BrClose}\ \dfrac{\textsc{Scope}\ \dfrac{\Psi \rhd P \xrightarrow{\overline{!M}\ (\nu\widetilde{a})N} P'}{\Psi \rhd (\nu y)P \xrightarrow{\overline{!M}\ (\nu\widetilde{a})N} (\nu y)P'}\ y\#\overline{!M}\ (\nu\widetilde{a})N, \Psi}{\Psi \rhd (\nu x)(\nu y)P \xrightarrow{\tau} (\nu x)(\nu\widetilde{a})(\nu y)P'}\ x \in \mathsf{n}(M), x\#\Psi$$

Since $x, y, \widetilde{a}\#\Psi$ and $(x, \widetilde{a}, y) \equiv (y, x, \widetilde{a})$ we have
$(\Psi, (\nu y)(\nu x)(\nu\widetilde{a})P', (\nu x)(\nu\widetilde{a})(\nu y)P') \in \mathcal{R} \subseteq \mathcal{S}^*$.

2. $y \in \mathsf{n}(\overline{!M}\ (\nu\widetilde{a})N)$ and $y \in \mathsf{n}(M)$: We have the following transition.

$$\textsc{Scope}\ \dfrac{\textsc{BrClose}\ \dfrac{\Psi \rhd P \xrightarrow{\overline{!M}\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu y)P \xrightarrow{\tau} (\nu y)(\nu\widetilde{a})P'}\ y \in \mathsf{n}(M), y\#\Psi}{\Psi \rhd (\nu x)(\nu y)P \xrightarrow{\tau} (\nu x)(\nu y)(\nu\widetilde{a})P'}\ x\#\tau, \Psi$$

Since $x, y\#\Psi$ and $y, x \equiv x, y$ we have $(\Psi, (\nu y)(\nu x)(\nu\widetilde{a})P', (\nu x)(\nu y)(\nu\widetilde{a})P') \in \mathcal{R} \subseteq \mathcal{S}^*$

3. $y \in \mathsf{n}(\overline{!M}\ (\nu\widetilde{a})N)$ and $y\#M$: We then have $y \in \mathsf{n}(N)$, and derive

$$\textsc{BrClose}\ \dfrac{\textsc{BrOpen}\ \dfrac{\Psi \rhd P \xrightarrow{\overline{!M}\,(\nu\widetilde{a})N} P'}{\Psi \rhd (\nu y)P \xrightarrow{\overline{!M}\,(\nu y)(\nu\widetilde{a})N} P'}\ y\#\widetilde{a}, \Psi, M, y \in \mathsf{n}(N)}{\Psi \rhd (\nu x)(\nu y)P \xrightarrow{\tau} (\nu x)(\nu y)(\nu\widetilde{a})P'}\ x \in \mathsf{n}(M), x\#\Psi$$

Since $x, y\#\Psi$ and $y, x \equiv x, y$ we have $(\Psi, (\nu y)(\nu x)(\nu\widetilde{a})P', (\nu x)(\nu y)(\nu\widetilde{a})P') \in \mathcal{R} \subseteq \mathcal{S}^*$. □

The soundness proof for scope extension uses the same ideas as the proof of Lemma 12.

3.2 Motivating the Requisites

An apparently simpler way to define broadcast connectivity is to have just one binary connectivity predicate relating input and output prefixes, as $\dot{\leftrightarrow}$ does for unicast communication. However, such a predicate would need to be transitive and symmetric for Theorem 11 to hold, for the same reasons as in the original psi calculus (detailed in [5]). In wireless broadcast communication systems, symmetry and transitivity do not necessarily hold and the requirements would not be reasonable.

A weaker version of condition 2 (resp. 1) of Definition 6 would be to require $\mathsf{n}(K) \subseteq \mathsf{n}(M, \Psi)$ whenever $\Psi \vdash K \dot{\succ} M$ (resp. $\Psi \vdash M \dot{\prec} K$). However, this leads to structural equivalence not being sound for bisimulation: the scope extension case of Theorem 11 fails, as we see in the following example.

Example 13 We let $\mathbf{A} = \mathcal{P}_{\text{fin}}(\mathcal{N})$ with $\mathbf{1} = \emptyset$ and $\otimes = \cup$. We let $\mathbf{T} = \mathcal{N}$ and $\mathbf{C} = \{a \dot{\leftrightarrow} b, a \dot{\prec} b, a \dot{\succ} b : a, b \in \mathcal{N}\}$. We define $\vdash$ by $\forall \Psi, a, b,\ \Psi \vdash b \dot{\prec} b$ iff $b \in \Psi$ and $\Psi \vdash b \dot{\succ} a$ iff $b \in \Psi$. Note that this defintion of entailment does not satisfy Definition 6, since we may have $\Psi \vdash b \dot{\succ} a$ for some $b \neq a$.

We let $P := (\nu a)((\!|\{a\}|\!) \mid \overline{a}.\mathbf{0} \mid \underline{c}.\mathbf{0})$. Here $\mathbf{1} \rhd P \xrightarrow{\tau} (\nu a)((\!|\{a\}|\!) \mid \mathbf{0} \mid \mathbf{0})$. However, P results from scope extension from $Q := (\nu a)((\!|\{a\}|\!) \mid \overline{a}.\mathbf{0}) \mid \underline{c}.\mathbf{0}$, but Q does not have a corresponding transition under frame $\mathbf{1}$.

In contrast to unicast actions, the support of the subjects of broadcast actions is always included in the support of the process generating the action. This result is used in the proof of the scope extension case of Theorem 11, to show that a scope extension does not enable any additional broadcast communication.

Lemma 14 *If* $\Psi \rhd P \xrightarrow{\overline{!K}\,(\nu\tilde{a})N} P'$ *or* $\Psi \rhd P \xrightarrow{?K\;N} P'$ *then* $\mathsf{n}(K) \subseteq \mathsf{n}(P)$.

Proof By induction on the derivation, using Definition 6 at the base cases.

4 Modelling network topology changes

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

The main idea is to allow for different generations of assertions by tagging assertions with a time. Only the most recent generation is used; a generation is made obsolete by composition with 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 terms $\mathbf{B} \subseteq \mathbf{T}$ used as broadcast channels and in prefixes; we let B, B' range over elements of $\mathbf{B}$.

4.1 Simple topology

Here assertions are finite sets of connectivity information ($M \mathrel{\dot{\prec}} K$ resp. $K \mathrel{\dot{\succ}} M$), labelled with a time, with the empty set at time 0 as the unit assertion. Assertion composition intuitively computes the union of all connectivity information labelled with the most recent generation. The sets $\mathbf{C}$ and $\mathbf{A}$ are defined using constructors operating on terms. We define substitution on $\mathbf{C}$ and $\mathbf{A}$ homomorphically on their structure. For simplicity, we assume that no rewriting happens in broadcast output, i.e., that $\dot{\prec}$ is the equality relation of $\mathbf{B}$.

Formally,

$$\begin{aligned}
\mathbf{C} &\triangleq \{\bot\} \cup \{\mathsf{currentGeneration}(g) : g \in \mathbb{N}\} \cup \\
&\quad\;\; \{K \mathrel{\dot{\succ}} M : K, M \in \mathbf{T}\} \cup \{M \mathrel{\dot{\prec}} K : K, M \in \mathbf{T}\} \\
\mathbf{A} &\triangleq \mathbb{N} \times \mathcal{P}_{\text{fin}}(\{\langle K \mathrel{\dot{\succ}} M\rangle : K, M \in \mathbf{T}\}) \\
\mathbf{1} &\triangleq \langle 0, \emptyset\rangle
\end{aligned}$$

$$\langle g, S\rangle \otimes \langle g', T\rangle \triangleq \begin{cases} \langle g, S\rangle & \text{if } g > g' \\ \langle g', T\rangle & \text{if } g < g' \\ \langle g, S \cup T\rangle & \text{if } g = g' \end{cases}$$

$$\begin{aligned}
&\langle g, S\rangle \vdash \mathsf{currentGeneration}(g') \text{ iff } g = g' \\
&\langle g, S\rangle \vdash B \mathrel{\dot{\prec}} B' \text{ if } B = B' \\
&\langle g, S\rangle \vdash B \mathrel{\dot{\succ}} B' \text{ if } B \mathrel{\dot{\succ}} B' \in S \text{ and } \mathsf{n}(B) \subseteq \mathsf{n}(B')
\end{aligned}$$

Proposition 15 *Given* $\mathbf{T}$ *with a substitution function satisfying the requirements of Section 2, the definitions of* $\mathbf{C}$, $\mathbf{A}$, $\otimes$, $\mathbf{1}$ *and* $\vdash$ *as above and* $(M \dot{\leftrightarrow} N) \triangleq \bot$ *satisfy the requirements of a broadcast psi-calculus.*

The assertion $\langle g, \{B \dot{\succ} B'\}\rangle$ states that B' is in-connected to B in generation g if $\mathsf{n}(B) \subseteq \mathsf{n}(B')$. The condition $\mathsf{currentGeneration}(g)$ is used to test if g is the most recent generation. It is needed for assertion equivalence to be compositional: without this condition we would have $\langle 0, \{M \dot{\succ} K\}\rangle \simeq \langle 1, \{M \dot{\succ} K\rangle\}$ and $\langle 0, \{M \dot{\succ} K\}\rangle \otimes \langle 1, \{K \dot{\succ} M\}\rangle \not\simeq \langle 1, \{M \dot{\succ} K\}\rangle \otimes \langle 1, \{K \dot{\succ} M\}\rangle$, contradicting compositionality.

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

$$T = \llparenthesis \langle 1, \emptyset\rangle \rrparenthesis \mid \tau\,.\,\big(\llparenthesis \langle 2, \{K \dot{\succ} M, K \dot{\succ} N\}\rangle \rrparenthesis \mid \tau\,.\,(\llparenthesis \langle 3, \{K \dot{\succ} M\}\rangle \rrparenthesis)\big)$$

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

Such a connectivity controller can also implement standard mobility models [**?**] over a discretized finite space. More fine-grained mobility models can be implemented by associating a generation with each possible connection, together with a flag for whether the connection is possible or not. In such a model, assertion $\{\langle 0, M \dot{\succ} K, \text{true}\rangle\}$ states that the link $M \dot{\succ} K$ is enabled in its generation 0.

4.2 Scoped topology

As a variation of the example above we define a model where every name d corresponds to a broadcast channel with dynamic topology. The use of a name in the broadcast channel allows to restrict its scope.

$$\begin{aligned}
\mathbf{B} &\triangleq \{\mathsf{Bs}(d) : d \in \mathcal{N}\} \cup \{\mathsf{Br}(M, d) : M \in \mathbf{T},\ d \in \mathcal{N}\} \cup \mathcal{N} \\
\mathbf{C} &\triangleq \{\bot\} \cup \{\mathsf{currentGeneration}(g, K) : g \in \mathbb{N},\ K \in \mathbf{T}\} \cup \\
&\quad\ \{\mathsf{Bs}(M) \dot{\prec} K) : M, K \in \mathbf{T}\} \cup \{K \dot{\succ} \mathsf{Br}(M, N) : M, N, K \in \mathbf{T}\} \\
\mathbf{A} &\triangleq \mathbf{T} \rightarrow_{\text{fin}} \mathbb{N} \times \mathcal{P}_{\text{fin}}(\{\mathsf{Conn}(M, N) : M, N \in \mathbf{T}\}) \\
\mathbf{1} &\triangleq \emptyset
\end{aligned}$$

$$(\Psi \otimes \Psi')(M) \triangleq \begin{cases} \langle g, S\rangle & \text{if } \Psi(M) = \langle g, S\rangle \wedge \ \ (M \notin \text{dom}(\Psi') \vee \\ & \qquad\qquad\qquad\qquad\quad (\Psi'(M) = \langle j, T\rangle \wedge g > g')) \\ \langle g', T\rangle & \text{if } \Psi'(M) = \langle g', T\rangle \wedge (M \notin \text{dom}(\Psi) \vee \\ & \qquad\qquad\qquad\qquad\quad (\Psi(M) = \langle g, S\rangle \wedge g < g')) \\ \langle g, S \cup T\rangle & \text{if } \Psi(M) = \langle g, S\rangle \wedge \ \ \Psi'(M) = \langle g, T\rangle \end{cases}$$

$\Psi \vdash \mathsf{currentGeneration}(g, d)$ if $\Psi(d) = \langle g, S \rangle$
$\Psi \vdash \mathsf{Bs}(c) \dot{\prec} d$ if $c = d$
$\Psi \vdash c \dot{\succ} \mathsf{Br}(N, d)$ if $c = d$ and $\Psi(c) = \langle g, S \rangle$ with $\mathsf{Conn}(N, d) \in S$

Proposition 16 *Given* $\mathbf{T}$ *with a substitution function satisfying the requirements of Section 2, the definitions of* $\mathbf{C}$, $\mathbf{A}$, $\otimes$, $\mathbf{1}$ *and* $\vdash$ *as above and* $(M \dot{\leftrightarrow} N) \triangleq \bot$ *satisfy the requirements of a broadcast psi-calculus.*

We can then define a topology controller which gradually changes the topology from fully disconnected to "a listens on d and b listens on d":

$$\begin{aligned} T = {} & (\!|d \mapsto \langle 1, \emptyset \rangle|\!) \mid \tau \, . \, ((\!|d \mapsto \langle 2, \{\mathsf{Conn}(a, d)\}\rangle|\!) \\ & \mid \tau \, . \, ((\!|d \mapsto \langle 3, \{\mathsf{Conn}(a, d), \mathsf{Conn}(b, d)\}\rangle|\!))) \end{aligned}$$

We now put a process P inside the scope of d in parallel with the topology controller as $(\nu d)(P \mid T)$. This ensures that P can communicate using broadcast on channel d while letting T, but not the environment, influence the topology.

5 The LUNAR protocol in Psi

In this section we present a model of the LUNAR routing protocol for mobile ad-hoc networks [28,29]. 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 [32,31] using SPIN and UPPAAL; our model is significantly more succinct 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 time-to-live (TTL) fields in messages, optional protocol fields, globally unique host identifiers, etc. These abstractions are similar to those made in [32,31]. We do not deal with time explicitly. In the SPIN verification, time is handled at an abstract level by using the Promela `timeout` predicate which is true when no other statement is executable, and checking that in this case, the protocol has succeeded in delivering a message (cf. Theorem 18).

5.1 The LUNAR broadcast psi-calculus

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 4.1, and unicast channels which are pairs $\langle sel, mac\rangle$ where sel is a selector name and mac is a MAC address name. The sel part can also be a $\mathsf{RouteOf}(node, ip)$ 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 are used to record requests originated at the local node with $\mathsf{Redirected}(node, sel)$, and with $\mathsf{HaveRoute}(node, destip, hops, sel)$ to specify found routes. The conditions contain predicates for testing if a route has been found ($\mathsf{HaveRoute}(node, ip)$), if a selector has been used for a request originating at the local node ($\mathsf{Redirected}(node, sel)$), and to extract the forwarder of a route ($\langle \mathsf{RouteOf}(node, ip), x\rangle \dot{\leftrightarrow} \langle sel, x\rangle$).

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

The parameters of the LUNAR broadcast psi-calculus extend the simple topology calculus in Section 4.1. We define substitution in the standard way, as the syntactic replacement of names by terms. The sets $\mathbf{T}, \mathbf{C}$ and $\mathbf{A}$ are defined recursively using constructors operating on terms in order to be closed under substitution.

$$
\begin{aligned}
\mathbf{T} &\triangleq \mathcal{N} \cup \{\mathsf{node}_i : i \in \mathbb{N}\} \cup \{\mathsf{delivered}\} \cup \\
&\quad \{\mathsf{RREQ}(Ser, TargIp, Rep) : Ser,\ TargIp,\ Rep \in \mathbf{T}\} \cup \\
&\quad \{\mathsf{RREP}(i, Fwd) : i,\ Fwd \in \mathbf{T}\} \cup \\
&\quad \{\mathsf{RouteOf}(Node, Ip) : Node,\ Ip \in \mathbf{T}\} \cup \\
&\quad \{\langle Sel, N\rangle : Sel, N \in \mathbf{T}\} \cup \{N+1 : N \in \mathbf{T}\} \cup \{0\} \\
\mathbf{C} &\triangleq \{M = N, M \dot{\leftrightarrow} N, \mathsf{HaveRoute}(M, N), \mathsf{Redirected}(M, N) : M, N \in \mathbf{T}\} \cup \\
&\quad \{K \dot{\succ} M : K, M \in \mathbf{T}\} \cup \{M \dot{\prec} K : K, M \in \mathbf{T}\} \cup \\
&\quad \{\mathsf{currentGeneration}(g) : g \in \mathbb{N}\} \cup \{\neg\phi : \phi \in \mathbf{C}\} \\
\mathbf{A} &\triangleq \mathbb{N} \times \mathcal{P}_{\text{fin}}(\{\langle K \dot{\succ} M\rangle : K, M \in \mathbf{T}\}) \times \\
&\quad \mathcal{P}_{\text{fin}}(\{\mathsf{HaveRoute}(M, N_1, i, N_2) : i,\ M, N_1, N_2 \in \mathbf{T}\} \cup \\
&\qquad \{\mathsf{Redirected}(M, N) : M, N \in \mathbf{T}\}) \\
\mathbf{1} &\triangleq \langle 0, \emptyset, \emptyset\rangle
\end{aligned}
$$

$$
\langle g, S, A\rangle \otimes \langle g', T, B\rangle \triangleq \begin{cases} \langle g, S, A \cup B\rangle & \text{if } g > g' \\ \langle g', T, A \cup B\rangle & \text{if } g < g' \\ \langle g, S \cup T, A \cup B\rangle & \text{if } g = g' \end{cases}
$$

Given $\Psi = \langle g, S, A\rangle$, we let $\mathcal{R}_\Psi$ be the symmetric and transitive closure of the relation

$$\{(\langle a,b\rangle,\langle a,b\rangle) : a,b \in \mathcal{N}\} \cup \{(\langle \mathsf{delivered}, \mathsf{node}_i\rangle, \langle \mathsf{delivered}, \mathsf{node}_i\rangle) : i \in \mathbb{N}\} \cup$$
$$\{(\langle \mathsf{RouteOf}(\mathsf{node}_i, a), x\rangle, \langle b, x\rangle) : i \in \mathbb{N}, j \in \mathbf{T}, \mathsf{HaveRoute}(\mathsf{node}_i, a, j, b) \in A\}$$

Entailment is then defined as follows.

$$\begin{array}{rl}
\Psi \vdash & a = a,\ a \in \mathcal{N} \\
\Psi \vdash & M \dot{\leftrightarrow} N \text{ iff } (M,N) \in \mathcal{R}_\Psi \\
\langle g,S,A\rangle \vdash & \mathsf{currentGeneration}(g) \\
\Psi \vdash & M \dot{\prec} N \text{ iff } M = N \\
\langle g,S,A\rangle \vdash & M \dot{\succ} N \text{ iff } M \dot{\succ} N \in S \\
 & \qquad\qquad \text{and } \mathsf{n}(M) \subseteq \mathsf{n}(N) \\
\langle g,S,A \cup \{\mathsf{HaveRoute}(\mathsf{node}_i,a,j,b)\}\rangle \vdash & \mathsf{HaveRoute}(\mathsf{node}_i,a) \\
\langle g,S,A \cup \{\mathsf{Redirected}(\mathsf{node}_i,s)\}\rangle \vdash & \mathsf{Redirected}(\mathsf{node}_i,s) \\
\Psi \vdash & \neg\varphi \text{ if not } \Psi \vdash \varphi
\end{array}$$

Theorem 17 *The LUNAR psi-calculus defined above satisfies all the requisites of a broadcast psi-calculus.*

This theorem has been formally proved in Isabelle/Nominal [**?**]. A sketch outlining the main ideas of the proof follows:

Proof (sketch) The requisites on the support of the broadcast channels are immediate from the definition. It is straight-forward to show the Abelian monoid laws for $\otimes, \mathbf{1}$. Transitivity and symmetry of channel equivalence holds by definition. The only nontrivial property is compositionality: We establish that $\Psi \otimes \Psi_1 \vdash \varphi$ and $\Psi_1 \simeq \Psi_2$ implies $\Psi \otimes \Psi_2 \vdash \varphi$ by induction on the structure of the condition φ. The only inductive step is for negation and this follows by symmetry of $\simeq$. If φ is a broadcast connectivity condition or $\mathsf{currentGeneration(g)}$, the proof is by case distinction on the relative generations of Ψ_1, Ψ_2 and Ψ. If φ is a channel equivalence an inner induction on the length of the chain of the involved $\mathsf{HaveRoute}$ elements in $\Psi \otimes \Psi_1$ is necessary. Each such element is either in Ψ and therefore also in $\Psi \otimes \Psi_2$, or in Ψ_1. In the latter case Ψ_1 entails a channel equivalence from this element alone and therefore Ψ_2 entails the same. Thus Ψ_2 must contain a suitable sequence of $\mathsf{HaveRoute}$ elements to derive this channel equivalence; this sequence is then in $\Psi \otimes \Psi_2$.

5.2 Representing process identifiers

We use process identifiers to improve the readability of the LUNAR protocol model. However, an astute reader will note that broadcast psi-calculi do not feature process identifiers - rather, replication is used as the mechanism for expressing infinite behaviour. In many other process calculi, process identifiers and recursion can be encoded in a standard fashion using replication, see e.g. [26]. Unfortunately, there is currently no proof that the same encodability results apply to broadcast psi-calculi.

To introduce process identifiers on a more sound theoretical foundation, we combine broadcast psi-calculi with higher-order psi-calculi [**?**], an orthogonal extension of psi-calculi which allows terms to act as handles to invoke the behaviour of processes. In this setting, process identifiers are simply terms.

Briefly, higher-order psi-calculi introduce the notion of a *clause* $M \Leftarrow P$, meaning that the term M is a handle for invoking P. We extend the entailment relation $\vdash$ so that assertions can entail clauses in addition to conditions. Agents are extended with *invocations* $\mathbf{run}\ M$, and a single new rule is added to the semantics:

$$\textsc{Invocation}\ \frac{\Psi \vdash M \Leftarrow P \qquad \Psi \rhd P \xrightarrow{\alpha} P'}{\Psi \rhd \mathbf{run}\ M \xrightarrow{\alpha} P'}$$

The calculi that result from adding the above-mentioned extensions to broadcast psi-calculi will be referred to as *higher-order broadcast psi-calculi.* We use Isabelle/Nominal to formally prove that all the meta-theoretic results presented in Section 3.1 apply not only to broadcast psi-calculi, but also to higher-order broadcast psi-calculi - hence we feel justified in claiming that broadcast and higher-order are orthogonal extensions. The proof scripts are available online [**?**].

Further, higher-order psi-calculi feature a lifting technique whereby an arbitrary first-order psi-calculus can be lifted to a corresponding canonical higher-order psi-calculus, extending it with parametrised clauses. In a canonical higher-order psi-calculus, sets of parametrised clauses on the form $M(N) \Leftarrow P$ are added to the assertions, such that $\{M(N) \Leftarrow P\} \vdash M(N[\widetilde{x} := \widetilde{T}]) \Leftarrow P[\widetilde{x} := \widetilde{T}]$.

In the following, we will implicitly be representing clauses using this feature of the canonical higher-order calculus corresponding to the LUNAR broadcast psi-calculus of Section 5.1.

5.3 The psi-calculus model of the LUNAR protocol

Figures 1-7 describe our psi-calculus model of the LUNAR protocol. Process declarations are 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 $P[\widetilde{x} := \widetilde{L}]$. For our purposes, lists can be adequately represented using the pairing construct included in the term language. We write $\mathbf{if}\ \varphi\ \mathbf{then}\ P\ \mathbf{else}\ Q$ for $\mathbf{case}\ \varphi : P\ [\!]\ \neg\varphi : Q$, and assume a suitable encoding of the τ prefix.

Our model of the protocol closely follows the informal protocol description in [29, Section 4]. Each figure in our model corresponds 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. An example of this can be seen in the LunARP process declaration in Fig. 1. The description in [29, Section 4, step 0.a] says "`Allocate an unused "receiver chosen" selector S and bind it to a transient "source RREP packet handler"`", which in our process declaration corresponds to the binding of *rchosen* and the subprocess $!\,\underline{\langle rchosen, mymac\rangle}(x)\,.\,\mathsf{SRepHandler}(mynode, mymac, destip, x)$.

In the informal protocol description [29], the FIB is "abused" (in steps 0.b and 1.b) 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 and a matching condition. An example can be seen is the subprocess $(\!|\mathsf{Redirected}(mynode, schosen)|\!)$ of the LunARP process declaration, and the test on the first line of the RreqHandler process declaration (Fig. 2) using the $\mathsf{Redirected}(mynode, schosen)$ condition.

The routing table is modelled using assertions, which illustrates how these can be used as a global data structure. Additions to the routing table are done in the SRrepHandler process definition (Fig. 4), which adds $(\!|\mathsf{HaveRoute}(mynode, destip, hops, rchosen)|\!)$ to the environment. Such assertions together form the routing table, which is tested in the IPtransmit process definition (Fig. 7) using the $\mathsf{HaveRoute}(mynode, destip)$ condition.

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 (Fig. 7, **else** branch). It then (Fig. 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 (Fig. 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 (Fig. 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 (Fig. 4), it installs a route and informs the IP layer about it. The message can then be resent (Fig. 7, **then** branch) and delivered (Fig. 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

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

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

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

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

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

Fig. 3 Part 3: Intermediate RREP packet handler

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

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

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

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

protocol [32, 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.

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.

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

Theorem 18 *If Ψ connects node_0 and node_n via a node node_i (i.e. $\Psi \vdash \mathsf{node}_0 \mathrel{\dot{\succ}} \mathsf{node}_i$ and $\Psi \vdash \mathsf{node}_i \mathrel{\dot{\succ}} \mathsf{node}_n$), then*

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

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.*

The SPIN verification performed in [32] checks the same reachability property, for up to five nodes. Our analysis is valid for any n, but is limited to a configuration where the sender (node 0) and the receiver (node n) are only separated by a single node. This limitation is due to the labour of manually following transitions in a non-trivial specification. We are currently working on remedies for this: firstly by extending our symbolic semantics for psi-calculi [14], secondly by implementing the symbolic semantics in our tool for automatic verification [12], and thirdly and orthogonally, by implementing the LUNAR model in Isabelle/Nominal. These remedies are still work in progress. In the Isabelle approach, we hope to prove the following conjecture.

Conjecture 19 If Ψ connects node_0 and node_n via k proxy nodes $\mathsf{pn}_1, \ldots, \mathsf{pn}_k$, where $\{\mathsf{pn}_1, \ldots, \mathsf{pn}_k\} \subseteq \{\mathsf{node}_1, \ldots, \mathsf{node}_{n-1}\}$
(i.e. $\Psi \vdash \mathsf{node}_0 \mathrel{\dot{\succ}} \mathsf{pn}_1, \mathsf{pn}_1 \mathrel{\dot{\succ}} \mathsf{pn}_2, \ldots, \mathsf{pn}_{k-1} \mathrel{\dot{\succ}} \mathsf{pn}_k, \mathsf{pn}_k \mathrel{\dot{\succ}} \mathsf{node}_n$), then

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

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

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.

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

However, when the input prefix is over a broadcast channel, as is the case here, the two are not equivalent since a single communication with $\mathsf{BrdHandler}'$ may result in arbitrarily many $\mathsf{RreqHandler}$ processes, while $\mathsf{BrdHandler}$ only results in one.

6 Related work

Process calculi with broadcast communication go back to the early 1980's. Milner developed SCCS [19] as a generalisation of CCS [18] 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 [22,23]. 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 [7]. 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$^\sharp$ [20]. 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 [17,15] 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 [10] 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 [27] 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 in wireless networks. There are results about strong bisimulation and a verification of a mobile ad hoc network leader election protocol through weak bisimulation. RBPT [9] 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).

$bA\pi$ [11] 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. [4]), labelled bisimilarity is not compositional in $bA\pi$, so the correspondence does not hold. A suggested fix is to remove communication of unicast channels from the calculus. We would finally mention CMN [16]. 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.

Finally, broadcast psi is different from the other calculi for wireless broadcast in that there is no stratification of the syntax into processes and

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.

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 proofs of the meta-theoretic results in Section 3.1 [24] are formally verified in the interactive theorem prover Isabelle/Nominal. The full formalisation of broadcast psi-calculi amounts to ca 33 000 lines of Isabelle code, of which about 21 000 lines are re-used from our earlier work [5].

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. [32], although we do not model timeouts. Their model [31] 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 are currently working on extending the symbolic semantics for psi-calculi [14] with broadcast, and implementing the semantics in our tool for automatic verification, the Psi-calculi Workbench [12]. 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. *Formalising process calculi.* PhD thesis, Uppsala University, June 2010.
4. 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.
5. 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*, 7(1), 2011. This is an extended version of [4].
6. J. Borgström, S. Huang, M. Johansson, P. Raabjerg, B. Victor, J. Å. Pohjola, and J. Parrow. Broadcast psi-calculi with an application to wireless protocols. In G. Barthe, A. Pardo, and G. Schneider, editors, *Software Engineering and Formal Methods: SEFM 2011*, volume 7041 of *LNCS*, pages 74–89. Springer, Nov. 2011.
7. 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.
8. M. Gabbay and A. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.
9. 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.
10. 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.
11. J. C. Godskesen. Observables for mobile and wireless broadcasting systems. In *Proc. of COORDINATION 2010*, volume 6116 of *LNCS*, pages 1–15. Springer, 2010.
12. R. Gutkovas. *Exercising Psi-calculi: A Psi-calculi workbench.* M.Sc. thesis, Department of Information Technology, Uppsala University, June 2011.
13. M. Johansson. *Psi-calculi: a framework for mobile process calculi.* PhD thesis, Uppsala University, May 2010.
14. M. Johansson, B. Victor, and J. Parrow. Computing strong and weak bisimulations for psi-calculi. *Journal of Logic and Algebraic Programming*, 81(3):162–180, 2012.
15. I. Lanese and D. Sangiorgi. An operational semantics for a calculus for wireless systems. *Theor. Comp. Sci.*, 411(19):1928–1948, 2010.

16. M. Merro. An observational theory for mobile ad hoc networks (full version). *Inf. Comput.*, 207(2):194–208, 2009.
17. N. Mezzetti and D. Sangiorgi. Towards a calculus for wireless systems. *Electr. Notes Theor. Comput. Sci.*, 158:331–353, 2006.
18. R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer, 1980.
19. R. Milner. Calculi for synchrony and asynchrony. *Theor. Comput. Sci.*, 25:267–310, 1983.
20. S. Nanz and C. Hankin. A framework for security analysis of mobile wireless networks. *Theor. Comp. Sci.*, 367(1-2):203–227, 2006.
21. A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.
22. 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.
23. K. V. S. Prasad. A calculus of broadcasting systems. *Sci. Comput. Program.*, 25(2-3):285–327, 1995.
24. 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.
25. D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, 1998. An extended abstract appeared in the *Proceedings of MFCS '95*, LNCS 969: 479–488.
26. D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes.* Cambridge University Press, 2001.
27. 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.
28. 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.
29. C. F. Tschudin. Lightweight underlay network ad hoc routing (LUNAR) protocol. Internet Draft, Mobile Ad Hoc Networking Working Group, Mar. 2004.
30. 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.
31. 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 [32].
32. 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.