A Routing Calculus with Flooding Updates

Manish Gaur[1,2,⋆], Simon J. Gay[2], and Ian Mackie[3]

[1] Department of Computer Sc and Engg, IET Lucknow, India
[2] School of Computing Science, University of Glasgow, Glasgow, UK
[3] LIX, École Polytechnique, 91128 Palaiseau Cedex, France

Abstract. We propose a process calculus which explicitly models routing in a distributed computer network. We define a model which consists of a network of routers where the topology of routers is fixed. The calculus has three syntactic categories namely processes, nodes and systems. Processes reside in nodes which are connected to a specific routers which forms a system. Upon creation of new nodes, the routing tables are updated using flooding method. We show that the proposed routing calculi is reduction equivalent to its specification asynchronous distributed pi-calculus (ADpi). We believe that such modeling helps in prototyping the distributed routing algorithms.

Keywords: Routing, Process Calculi, Flooding, Specification, Computational Cost.

1 Introduction

In the last decade, we have witnessed the birth of many calculus intended to support programming of global distributed systems. These formalisms, in [7,12,6,4,11,1,2,10], in general provide constructs and mechanisms at different abstraction levels. These models don't consider the actual topology or routing of the process communication as the Internet connectivity is neither a clique nor a forest of trees. We present a name passing calculus, $\mathtt{DR}_\pi$, (an elaboration of ADpi calculus [7]) with a realistic topology of nodes with routers to act as functions in determining the path from a source node to the destination node. We characterise the cost of communication to prove certain properties, such as path determination, about the routers with an aim to show their impact on the quality of service of the network. We develop the calculus for describing distributed computations explicitly in the presence of routers in an Internet like network. The new concept is that of a site for computational activity. It consists of named routers which host computational entities called *nodes*. Each node is directly connected to specific router. These sites run in parallel to form a large distributed network called a *system*. The communication between processes of any two nodes in this network is possible only through their respective routers. In Fig.1 we present a very simple network for the purpose of illustration. This network consists of three *routers* R_1, R_2 and R_3.

⋆ We gratefully acknowledge the support received from commonwealth scholarship commission, UK (Ref: INCS-2005-145 and INCF-2012-252).

R. Natarajan et al. (Eds.): ICDCIT 2015, LNCS 8956, pp. 181–186, 2015.

The nodes l, m, n, o, p, q and r are connected to their respective *routers* R_1, R_2 and R_3 as shown. The routers are connected through a fixed topology. There connectivity is a directed graph and is defined as Γ_c. We shall view the routers as named functions which map set of node names to router names. Therefore the entries in the router function form a table called a *routing table*. The routing table at a router R is expressed as $\langle R \rangle$. The routing tables at each router are used to determine the path of communication between the communicating processes.

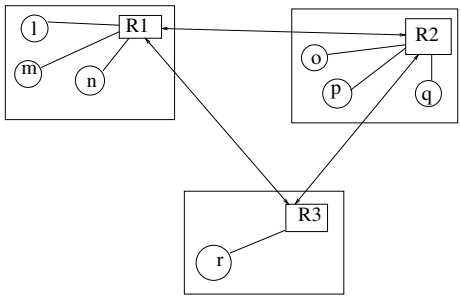

Fig. 1. A Simple Distributed Network with Routers

In $\mathtt{DR}_\pi$, we describe a method for routing table update where upon creation of a new node we update all the routing tables in the network. The fundamental approach followed is very similar to *breadth first traversal* [3] of a graph. In $\mathtt{DR}_\pi$, we use two types of messages. One called as control messages which are involved in propagating the update information and updating the routing tables about newly created nodes in the network. The other is a value propagating message which is used to propagate and deliver values to the waiting input processes. Whenever a new node is created a set of control messages propagate in a breadth first search manner across the network of routers to update all the routing tables. As soon as all the routing tables are updated the control messages are automatically discarded by the semantics of the language. This method of routing table update is known as Flooding[13]. We use two types of control messages; one for propagating the update information to all the connected routers and another for updating the router tables. The control messages don't participate in the communication of values between the processes.

A typical system in $\mathtt{DR}_\pi$ looks like $\langle R \rangle [\![n[P]]\!]$ where a process is P is located at node n. The node n is directly connected to the router R. The systems reduce with respect to the router connectivity Γ_c and therefore the reductions are defined on configurations. A configuration $\Gamma_c \rhd S$ consists of router connectivity Γ_c and system S.

In Sections 2 and 3 we describe the syntax and semantics of $\mathtt{DR}_\pi$. Section 4 discusses the equivalence between $\mathtt{DR}_\pi$ and its specification. Section 5 is the conclusion.

2 Syntax

We will use $v, u, \ldots$ to describe values which may be a name or a variable. We use variables $a, b, c, \ldots$ to range over channel names $\mathcal{C}$ or node names $\mathcal{N}$. $n, m, \ldots$ are used to range over node names $\mathcal{N}$ and we use $R, R_1, R_2, \ldots$ to range over set

of router names $\mathcal{R}$. The variables $k, l, \ldots$ range over integers to represent the cost of communication.

Further, we assume that sets of channel names, node names and router names are disjoint from each other. We also assume that router and node names are unique. There are three syntactic categories in the calculus: Systems, Nodes and Processes. The syntax of this calculus is described in Fig. 2.

We describe a system as $\langle R \rangle [\![M]\!]$ where R is a router and M is a nodes. All components in M are directly connected to the router R. Concurrency between two systems is expressed as $S \mid T$. $[R]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c)$ is a message at router R. This message propagates a value v sent by a process at node n to another process located at destination node m. Value v is supposed to be delivered at channel c of a waiting process at node m. k is an integer representing the number of hops the message has crossed on its way to the destination.

S, T ::=		**Systems**
	$\langle R \rangle [\![M]\!]$	Router
	$S \mid T$	Concurrency
	$[R]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c)$	Value Messages
	$[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R')$	Update message
	$[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{prop}, m)$	Propagate message
	$(\mathsf{new}\ d)\ S$	New name
	ε	Identity
M, N ::=		**Nodes**
	$m[T]$	Named processes
	$M \mid N$	Concurrency
	$(\mathsf{new}\ d)\ M$	New name
	0	Identity
T, U ::=		**Process terms**
	$c?(x)\,T$	Input
	$m!\langle v@c \rangle$	Output
	$\mathsf{if}\ u = v\ \mathsf{then}\ T\ \mathsf{else}\ U$	Matching
	$(\mathsf{new}\ b)\ T$	Channel name creation
	$\mathsf{newnode}\ m\ \mathsf{with}\ P\ \mathsf{in}\ Q$	New node creation
	$T \mid U$	Concurrency
	$* T$	Repetition
	stop	Identity

Fig. 2. Syntax of $\mathtt{DR}_\pi$

The control messages are categorised as update and propagate messages which we denote as $[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R')$ and $[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{prop}, m)$ respectively. The update message $[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R')$ is used to update the routing table $\langle R \rangle$ with an entry $\{m \rightarrow R'\}$ about the newly created node m. For example the message $[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R')$ will update the routing table at R with an entry $\{m \rightarrow R'\}$ provided the router R does not know about the node name m. If R already knows about m then this message at R is discarded. In this calculi we use a notation $\langle R \rangle(m) \downarrow$ to denote that node m is defined at the router R. Similarly we use $\langle R \rangle(m) \uparrow$ to denote that node m is undefined at router R.

The propagate message $[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{prop}, m)$ is used to propagate the new node name m across the network. The propagate message $[R]\mathsf{M}_{\mathsf{sg}}(\mathtt{prop}, m)$ generates set of update messages at all the routers which are directly connected to R with an update entry $\{m \rightarrow R\}$.

We call these messages, update and propagate, *control messages*. Since control messages don't deliver any values therefore they are not used in determining the quality of services in delivery of values. For this reason the superscript representing the number of hops a message has already crossed in a routers network has been left blank in control messages whereas in the value

carrying message $[R]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c)$, which is different than control messages, it has been represented as k. The term $(\mathsf{new}\, d)\, S$ is a scoping mechanism for names as usual [9,7]. The syntax for nodes are described in [7]. Similarly the process terms are very similar to the terms in [7,9]. The definitions of bound variables and names in [7,9] are extended in a similar way in this calculus as well. Since names of routers and nodes are fixed and disjoint therefore they can't be renamed in this calculi. However α-conversion may be applied to channel names. We use a formal relation between the systems called *structural equivalence*, intuitively to represent the systems as same computational entities. This is defined in a conventional way [7,9]. We use the notion $\equiv$ to represent this relation. Structural equivalence is defined for each syntactic categories. However, the process equivalence is inherited by the node equivalence and the node equivalence is inherited by the system equivalence. For example $\langle R\rangle[\![N]\!] \equiv \langle R\rangle[\![S]\!]$ is true provided $N \equiv S$ where N and S are nodes at router R. There are certain axioms which are standard and applicable to all syntactic categories. These standard axioms are similar to standard pi-calculus structural equivalence axioms [7,9]. The standard axioms for structural equivalence, which is applicable at all syntactic categories, is same as described in the conventional manner at [7]. We use two additional notations; one as $Adj(R_1)$ to represent $Adj(R_1) = \{R_2 \mid (R_1, R_2) \in \Gamma_c\}$ and the other $\Gamma_c \rhd \prod_{Adj(R_1)} [R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, k, R_1)$ to mean $[R_2]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, k, R_1) \mid [R_3]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, k, R_1) \mid \ldots \mid [R_n]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, k, R_1)$ where $Adj(R_1) = \{R_2, \ldots, R_n\}$.

3 Reduction Semantics

The reduction semantics of $\mathtt{DR}_\pi$ are defined on configurations $\Gamma_c \rhd S$. A typical configuration reduction step is described as $\Gamma_c \rhd S \longrightarrow^k \Gamma_c \rhd S'$ where the cost of this reduction is k and a system S reduces to S' with respect to the router connectivity Γ_c. The reduction rules for $\mathtt{DR}_\pi$ are given in Figures 3 and 4. The reduction rules (R-CONTX), (R-STRUCT) in Figure 4 are about compositional reductions and reductions that are defined upto structural equivalence. These rules are directly inherited from [7]. The rules (R-OUT) and (R-IN) in Figure 3 are about message creation and delivery. We have only one rule for message forwarding for those messages which carry a value for delivery to a waiting input process at some node. For example in a configuration $\Gamma_c \rhd [R_1]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c) \mid \langle R_2\rangle[\![N]\!] \mid S$, suppose $\langle R_1\rangle(m) = R_2$ then this message is hopped to router R_2. This means that the configuration $\Gamma_c \rhd [R_1]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c) \mid \langle R_2\rangle[\![N]\!] \mid S$ is reduced to $\Gamma_c \rhd [R_2]\mathsf{M}^{k+1}_{\mathsf{sg}}(n, m, v@c) \mid \langle R_2\rangle[\![N]\!] \mid S$. Note that the superscript k is incremented by one to record the number of hops the message has travelled so far. This reduction is irrespective of the knowledge of the routing table $\langle R_2\rangle$ about the $\mathsf{n}(v)$ if $v \in \mathcal{NN}$. This is because, in case $\langle R_2\rangle(v) \uparrow$, there are separate control messages which will eventually update the router table of R_2 about $\mathsf{n}(v)$.

The reduction rules about new node creation and routing table updates are significant and they follow flooding mechanism (in a breadth first traversal

mechanism [3]) to update the routing tables about the newly created nodes. These rules, (R-NEWNODE − CREATION), (R-UPDATE − I), (R-UPDATE − II) and (R-PROPAGATE), are self explanatory in Figure 3.

(R-OUT)
$$\Gamma_c \rhd \langle R\rangle[\![n[m!\langle v@c\rangle \mid P] \mid N]\!] \longrightarrow \Gamma_c \rhd [R]\mathsf{M}^0_{\mathsf{sg}}(n, m, v@c) \mid \langle R\rangle[\![n[P] \mid N]\!]$$

(R-MSG−FWD)
$$\frac{(R_1, R_2) \in \Gamma_c \quad \langle R_1\rangle(m) = R_2}{\Gamma_c \rhd [R_1]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c) \mid \langle R_2\rangle[\![N]\!] \mid S \longrightarrow \Gamma_c \rhd [R_2]\mathsf{M}^{k+1}_{\mathsf{sg}}(n, m, v@c) \mid \langle R_2\rangle[\![N]\!] \mid S}$$

(R-IN)
$$\frac{\langle R\rangle(m) = R}{\Gamma_c \rhd [R]\mathsf{M}^k_{\mathsf{sg}}(n, m, v@c) \mid \langle R\rangle[\![m[c?(x)\, P] \mid N]\!] \longrightarrow^k \Gamma_c \rhd \langle R\rangle[\![m[P\{^v/_x\}] \mid N]\!]}$$

(R-NEWNODE−CREATION)
$$\Gamma_c \rhd \langle R\rangle[\![n[\mathsf{newnode}\ m\ \mathsf{with}\ P\ \mathsf{in}\ Q]]\!] \longrightarrow \Gamma_c \rhd (\mathsf{new}\, m)\langle R\rangle[\![n[Q] \mid m[P]]\!] \mid [R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R)$$

(R-MATCH)
$$\Gamma_c \rhd \langle R\rangle[\![n[\mathsf{if}\ v = v\ \mathsf{then}\ P\ \mathsf{else}\ Q]]\!] \longrightarrow \Gamma_c \rhd \langle R\rangle[\![n[P]]\!]$$

(R-MISMATCH)
$$\Gamma_c \rhd \langle R\rangle[\![n[\mathsf{if}\ v_1 = v_2\ \mathsf{then}\ P\ \mathsf{else}\ Q]]\!] \longrightarrow \Gamma_c \rhd \langle R\rangle[\![n[Q]]\!]\ \ v_1 \neq v_2$$

(R-UPDATE−I)
$$\frac{\langle R_1\rangle(m) \uparrow}{\Gamma_c \rhd \langle R_1\rangle[\![N]\!] \mid [R_1]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R_2) \longrightarrow \Gamma_c \rhd \langle R_1\{m \to R_2\}\rangle[\![N]\!] \mid [R_1]\mathsf{M}_{\mathsf{sg}}(\mathtt{prop}, m)}$$

(R-UPDATE−II)
$$\frac{\langle R_1\rangle(m) \downarrow}{\Gamma_c \rhd \langle R_1\rangle[\![N]\!] \mid [R_1]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R_2) \longrightarrow \Gamma_c \rhd \langle R_1\rangle[\![N]\!]}$$

(R-PROPAGATE)
$$\Gamma_c \rhd [R_1]\mathsf{M}_{\mathsf{sg}}(\mathtt{prop}, m) \mid S \longrightarrow \Gamma_c \rhd \prod_{Adj(R_1)} [R]\mathsf{M}_{\mathsf{sg}}(\mathtt{update}, m, R_1) \mid S \qquad Adj(R_1) \in S$$

Fig. 3. Reduction semantics for $\mathtt{DR}_\pi$

How do we know that the reduction semantics is reasonable and does not introduce inconsistencies in the system? We know because we can prove that if a configuration is "coherent" before we apply a reduction it remains "coherent". "Coherence" will mean a series of properties, which summed together gets us the notion of a well-formed configuration. As an example all the routing tables must have knowledge of the propagating node names. Therefore this should be a condition on a well formed configurations. The notion of well formed configurations in $\mathtt{DR}_\pi$ is adapted from the established theory of typed behavioural equivalence [8,5]. It's easy to show that conditions on well-formed configurations are preserved by semantic reductions.

(R-STRUCT)
$$\frac{S \equiv S',\ \Gamma_c \rhd S' \longrightarrow^k \Gamma_c \rhd R',\ R' \equiv R}{\Gamma_c \rhd S \longrightarrow^k \Gamma_c \rhd R}$$

(R-CONTX)
$$\frac{\Gamma_c \rhd S_1 \longrightarrow^k \Gamma_c \rhd S_1'}{\begin{array}{l}\Gamma_c \rhd S_1 \mid S_2 \longrightarrow^k \Gamma_c \rhd S_1' \mid S_2 \\ \Gamma_c \rhd S_2 \mid S_1 \longrightarrow^k \Gamma_c \rhd S_2 \mid S_1' \\ \Gamma_c \rhd (\mathsf{new}\, d)\, S_1 \longrightarrow^k \Gamma_c \rhd (\mathsf{new}\, d)\, S_1'\end{array}}$$

Fig. 4. contd...Reduction semantics for $\mathtt{DR}_\pi$

4 Equivalence between $\mathtt{DR}_\pi$ and its Specification

We shall now try to establish an equivalence of $\mathtt{DR}_\pi$ with a specification of it. ADpi [7] like language where located processes are called nodes, and each pair of nodes are directly connected can be a specification for $\mathtt{DR}_\pi$. In [7], as all pair of nodes are directly connected they form a clique of the graph of connected nodes. Intuitively, $\mathtt{D}_\pi$ is a top level view of $\mathtt{DR}_\pi$.

We show that both languages, $\mathtt{DR}_\pi$ *and* $\mathtt{D}_\pi$*, are reduction equivalent after abstracting away the details of routers and paths from* $\mathtt{DR}_\pi$. For the purpose of abstraction of routers and paths from $\mathtt{DR}_\pi$ we define a function, $\mathfrak{I}$, over $\mathtt{DR}_\pi$ system to a $\mathtt{D}_\pi$ system. Function $\mathfrak{I}$ abstracts away the routers from a $\mathtt{DR}_\pi$ term.

5 Conclusion

We described the routing calculi $\mathtt{DR}_\pi$, where the crucial role of routers in determining the quality of communication services in a distributed network is demonstrated. We justified this model by showing that this is, in fact, implementation of ADpi [7]. The basic design of the model itself shows that it is closer to the real distributed networks.

References

1. Barbanera, F., Bugliesi, M., Dezani-Ciancaglini, M., Sassone, V.: A calculus of bounded capacities. In: Saraswat, V.A. (ed.) ASIAN 2003. LNCS, vol. 2896, pp. 205–223. Springer, Heidelberg (2003)
2. Cardelli, L., Gordon, A.D.: Mobile ambients. Theor. Comput. Sci. 240(1), 177–213 (2000)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press (2003)
4. Gaur, M., Hennessy, M.: Counting the cost in the picalculus (extended abstract). Electronic Notes in Theoretical Computer Science (ENTCS) 229(3), 117–129 (2009)
5. Gay, S.J., Hole, M.: Subtyping for session types in the pi-calculus. Acta Inf. 42(2-3), 191–225 (2005)
6. Griffin, T.G., Sobrinho, J.L.: Metarouting. In: SIGCOMM, pp. 1–12 (2005)
7. Hennessy, M.: A distributed Pi-Calculus. Cambridge University Press (2007)
8. Hennessy, M., Rathke, J.: Typed behavioural equivalences for processes in the presence of subtyping. Mathematical Structures in Computer Science 14(5), 651–684 (2004)
9. Milner, R.: Communicating and mobile systems: The π-Calculus. Cambridge University Press (1999)
10. Nicola, R.D., Gorla, D., Pugliese, R.: Basic observables for a calculus for global computing. Inf. Comput. 205(10), 1491–1525 (2007)
11. Orava, F., Parrow, J.: An algebraic verification of a mobile network. Formal Asp. Comput. 4(6), 497–543 (1992)
12. Sewell, P., Wojciechowski, P.T., Pierce, B.C.: Location-independent communication for mobile agents: A two-level architecture. In: Bal, H.E., Cardelli, L., Belkhouche, B. (eds.) ICCL 1998 Workshop. LNCS, vol. 1686, pp. 1–31. Springer, Heidelberg (1999)
13. Tanenbaum, A.S.: Computer Networks. Pearson Education, Inc., Upper Saddle River (2003)