SD Erlang Operational Semantics

Natalia Chechina, Huiqing Li,
Simon Thompson, and Phil Trinder

October 23, 2013

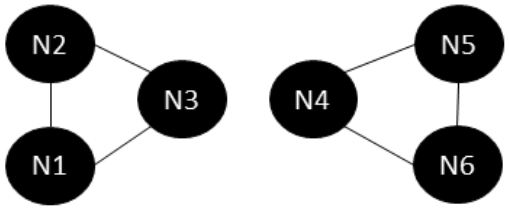
UPPSALA
UNIVERSITET

Outline

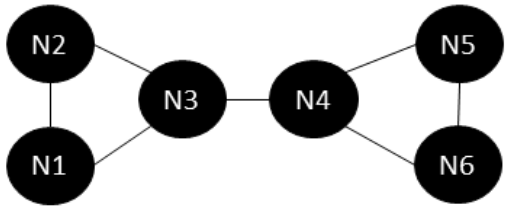
SD Erlang Overview

- SD Erlang is a small conservative extension of Distributed Erlang
- SD Erlang reduces the number of node connections and the number of nodes in a namespace
- Nodes in an s_group have transitive connections only with nodes from the same s_groups, but non-transitive connections with other nodes
- Each s_group has its own name space – names registered in an s_group are replicated only to the nodes from the same s_group
- Free and s_group nodes
  - A fee node (normal or hidden) belongs to no s_group
  - An s_group node belongs to at least one s_group

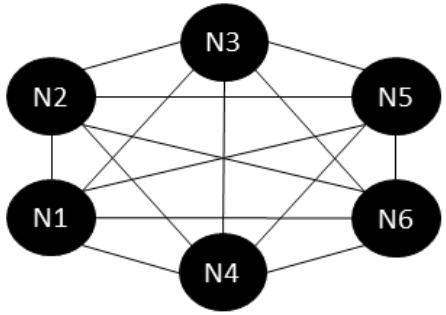
Free Node Connections vs. S_group Node Connections



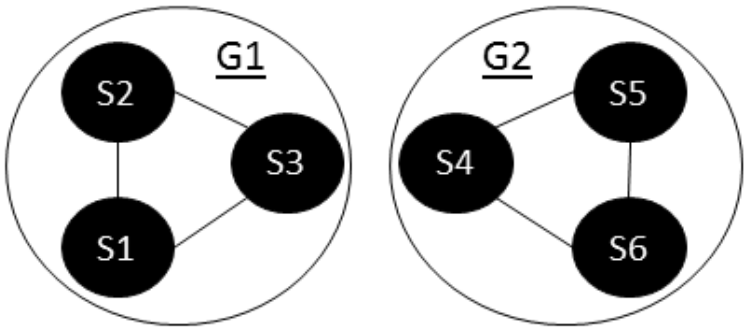
(a)



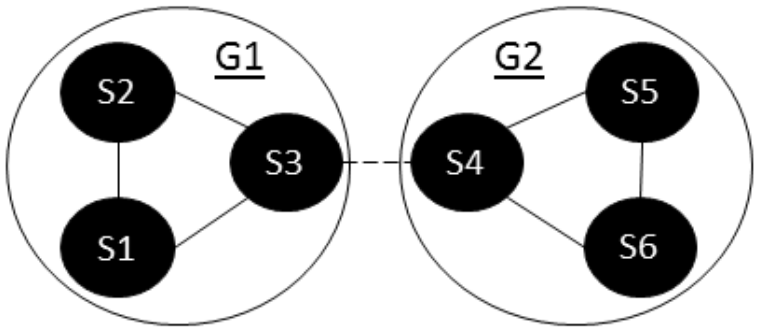
(b)



(c)

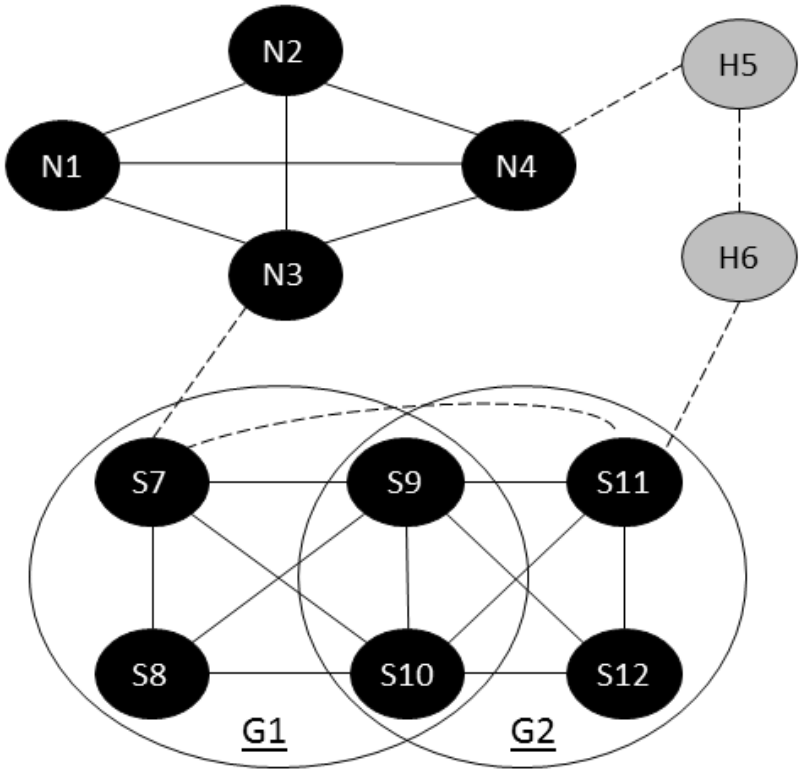


(d)



(e)

Types of Connections between Different Types of Nodes



Transitive connection ——————

Non-transitive connection - - - - - - - -

State

$$(grs, fgs, fhs, nds) \in \{state\} \equiv$$
$$\equiv \{(\{s\_group\}, \{free\_group\}, \{free\_hidden\_group\}, \{node\})\}$$

$$gr \in grs \equiv \{s\_group\} \equiv \{(s\_group\_name, \{node\_id\}, namespace)\}$$
$$fg \in fgs \equiv \{free\_group\} \equiv \{(\{node\_id\}, namespace)\}$$
$$fh \in fhs \equiv \{free\_hidden\_group\} \equiv \{(node\_id, namespace)\}$$
$$nd \in nds \equiv \{node\} \equiv \{(node\_id, node\_type, connections, gr\_names)\}$$

Property. Every node in an SD Erlang state is a member of one of the three classes of groups: *s_group*, *free_group*, or *free_hidden_group*. The three classes of groups partition the set of nodes.

State Components

$$gs \in \{gr\_names\} \equiv \{NoGroup, \{s\_group\_name\}\}$$
$$ns \in \{namespace\} \equiv \{\{(name, pid)\}\}$$
$$cs \in \{connections\} \equiv \{\{node\_id\}\}$$
$$nt \in \{node\_type\} \equiv \{Normal, Hidden\}$$
$$s \in \{NoGroup, s\_group\_name\}$$
$$n \in \{name\}$$
$$p \in \{pid\}$$
$$ni \in \{node\_id\}$$
$$nis \in \{\{node\_id\}\}$$
$$m \in \{message\}$$

Assumptions

1. No two s_groups have the same name, that is all *s_group_name*s are unique.
2. All *node_id*s identify some node. More formally, for all *node_id*s occurring in some state (*grs*, *fgs*, *fhs*, *nds*), there exists some node in *nds* with that *node_id*.

Transitions

$$(state, \mathrm{command}, ni) \longrightarrow (state', value)$$

Executing command on node *ni* in *state* returns *value* and transitions to $state'$.

register_name

SD Erlang function

s_group:register_name(SGroupName, Name, Pid) $\longrightarrow$ yes | no

$$((grs, fgs, fhs, nds), \text{register\_name}(s, n, p), ni)$$
$$\longrightarrow ((\{(s, \{ni\} \oplus nis, \{(n, p)\} \oplus ns)\} \oplus grs', fgs, fhs, nds), \text{True})$$
$$\text{If } (n, \_) \notin ns \wedge (\_, p) \notin ns$$
$$\longrightarrow ((grs, fgs, fhs, nds), \text{False})$$
$$\text{Otherwise}$$

where

$$\{(s, \{ni\} \oplus nis, ns)\} \oplus grs' \equiv grs$$

new_s_group

SD Erlang function

s_group:new_s_group(SGroupName, [Node]) $\longrightarrow$ ok | error

$$((grs, fgs, fhs, nds), \text{new\_s\_group}(s, nis), ni)$$

$$\longrightarrow ((grs', fgs', fhs', nds''), \text{Ok}) \qquad \text{If } ni \in nis$$

$$\longrightarrow ((grs, fgs, fhs, nds), \text{Error}) \qquad \text{Otherwise}$$

where

$$nds' \equiv \text{InterConnectNodes}(nis, nds)$$

$$nds'' \equiv \text{AddSGroup}(s, nis, nds')$$

$$grs' \equiv grs \oplus \{(s, nis, \{\})\}$$

$$(fgs', fhs') \equiv \text{RemoveNodes}(nis, fgs, fhs)$$

new_s_group – Auxiliary Functions (1)

$$\text{InterConnectNodes}(nis, nds)$$
$$= nds \ \cup \ \{(ni, nt, (cs \oplus nis) - \{ni\}, gs) \mid (ni, nt, cs, gs) \in nds, ni \in nis\}$$

$$\text{AddSGroup}(s, nis, nds) = nds \cup nds''$$
where
$$nds' \equiv \{(ni, nt, cs, gs) \mid (ni, nt, cs, gs) \in nds, ni \in nis\}$$
$$nds'' \equiv \{(ni, nt, cs, \text{AddSGroupS}(s, gs)) \mid (ni, nt, cs, gs) \in nds'\}$$

$$\text{AddSGroupS}(s, gs)$$
$$= \{s\} \qquad \text{If } gs \equiv NoGroup$$
$$= gs \oplus \{s\} \qquad \text{Otherwise}$$

new_s_group – Auxiliary Functions (2)

$$\text{RemoveNodes}(nis, fgs, fhs) = (fgs'', fhs')$$

where

$$fgs' \equiv \{(\{ni\} \oplus nis', ns') \mid (\{ni\} \oplus nis', ns') \in fgs, ni \in nis\}$$

$$fgs'' \equiv (fgs - fgs') \oplus \\ \oplus\{(nis', ns') \mid nis' \neq \{\},\ (\{ni\} \oplus nis', ns') \in fgs', ni \in nis\}$$

$$fhs' \equiv fhs - \{(ni, ns) \mid (ni, ns) \in fhs, ni \in nis\}$$

Validation of Semantics and Implementation

- Validate the consistency between the formal semantics and the SD Erlang implementation
- Use Erlang QuickCheck tool developed by Quivq
- Behaviour is specified by properties expressed in a logical form
- `eqc_statem` is a finite state machine in QuickCheck

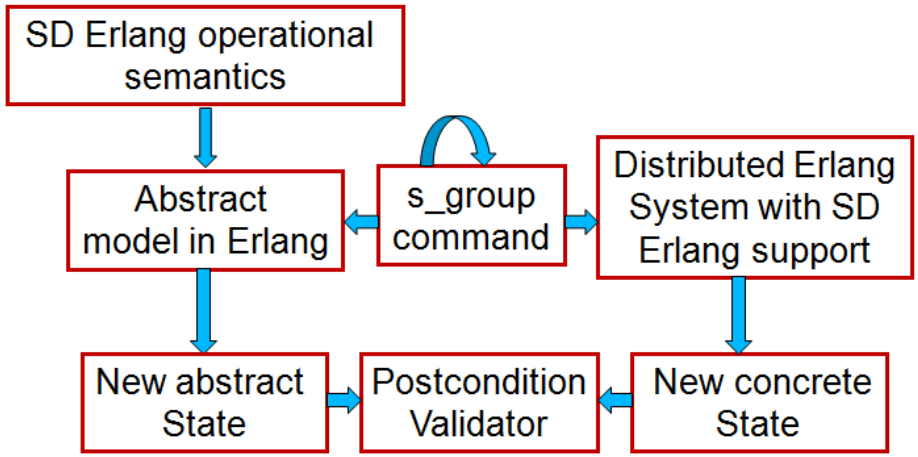


Figure: Testing SD Erlang Using QuickCheck `eqc_statem`

Precondition for `new_s_group` operation

$$
\begin{aligned}
&\mathit{precondition}(\_\mathit{State}, \{\mathit{call}, ?\mathit{MODULE}, \mathit{new\_s\_group}, \\
&\qquad [\{\_\mathit{SGroupName}, \mathit{NodeIds}, \_\mathit{CurNode}\}, \\
&\qquad \_\mathit{AllNodeIds}]\}) \rightarrow \\
&\mathit{NodeIds} /= [];
\end{aligned}
$$

Postcondition for `new_s_group` operation

- AbsRes means abstract result; AbsState means abstract state
- ActRes means actual result; ActState means actual state

$$
\begin{aligned}
&\mathit{postcondition}(\mathit{State}, \{\mathit{call}, ?\mathit{MODULE}, \mathit{new\_s\_group}, \\
&\qquad \{\mathit{SGroupName}, \mathit{NodeIds}, \mathit{CurNode}\}, \\
&\qquad \_\mathit{AllNodeIds}]\}, \\
&\qquad \{\mathit{ActResult}, \mathit{ActState}\}) \rightarrow \\
&\{\mathit{AbsRes}, \mathit{AbsState}\} = \\
&= \mathit{new\_s\_group\_next\_state}(\mathit{State}, \mathit{SGroupName}, \mathit{NodeIds}, \mathit{CurNode}), \\
&(\mathit{AbsResult} == \mathit{ActResult})\ \mathit{and}\ \mathit{is\_the\_same}(\mathit{ActState}, \mathit{AbsState});
\end{aligned}
$$

Conclusion and Future Work

Conclusion. To date...

- we have validated nine SD Erlang functions (out of sixteen)
- we found and fixed four bugs in the semantics
- we identified an issue in the SD Erlang implementation (due to node synchronisation?)

Future Work

- Continue to work on the semantics by running more QuickCheck tests
- Relaxing assumptions
- Fixing the outstanding SD Erlang implementation issue

Questions?

Thank you!