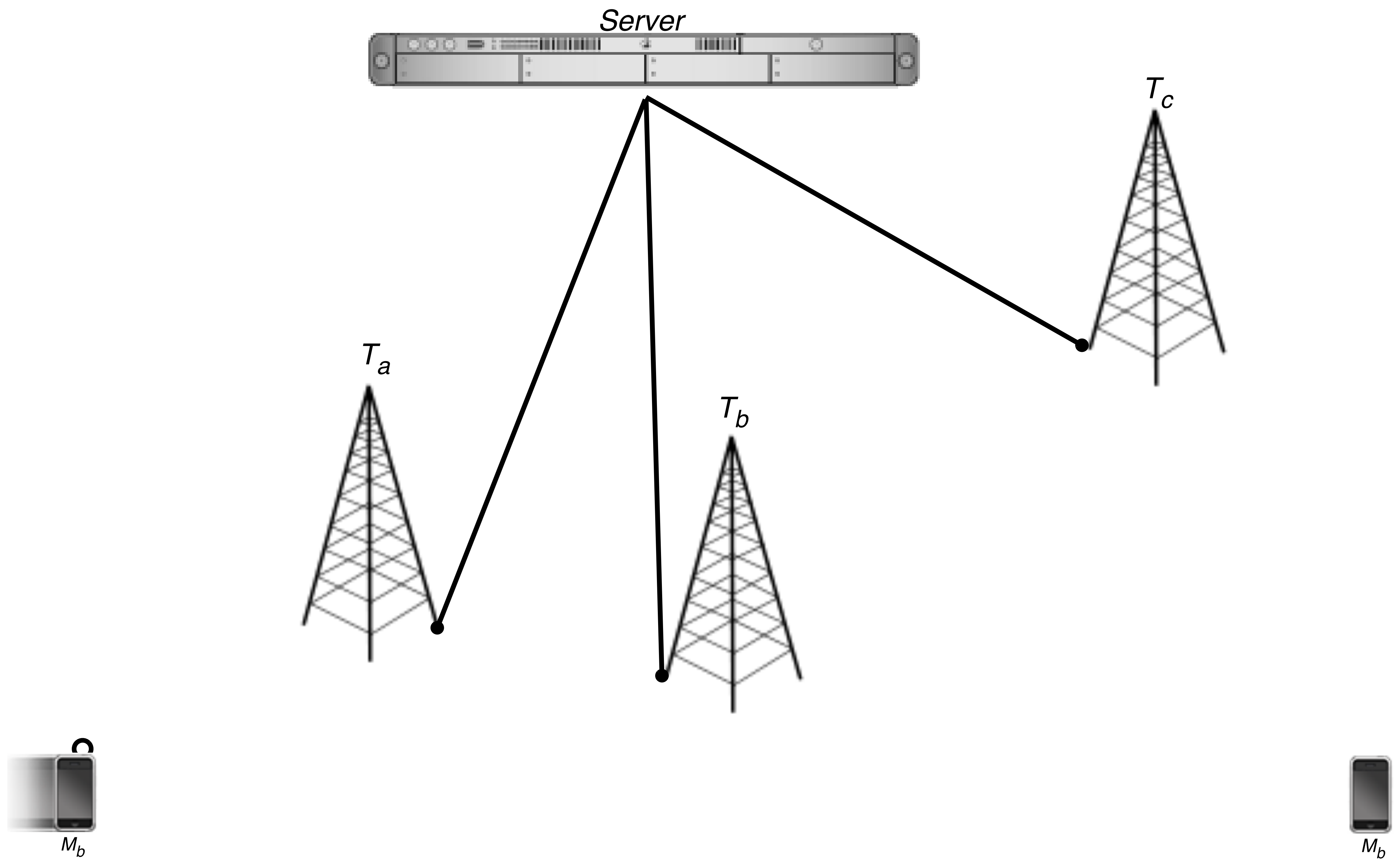
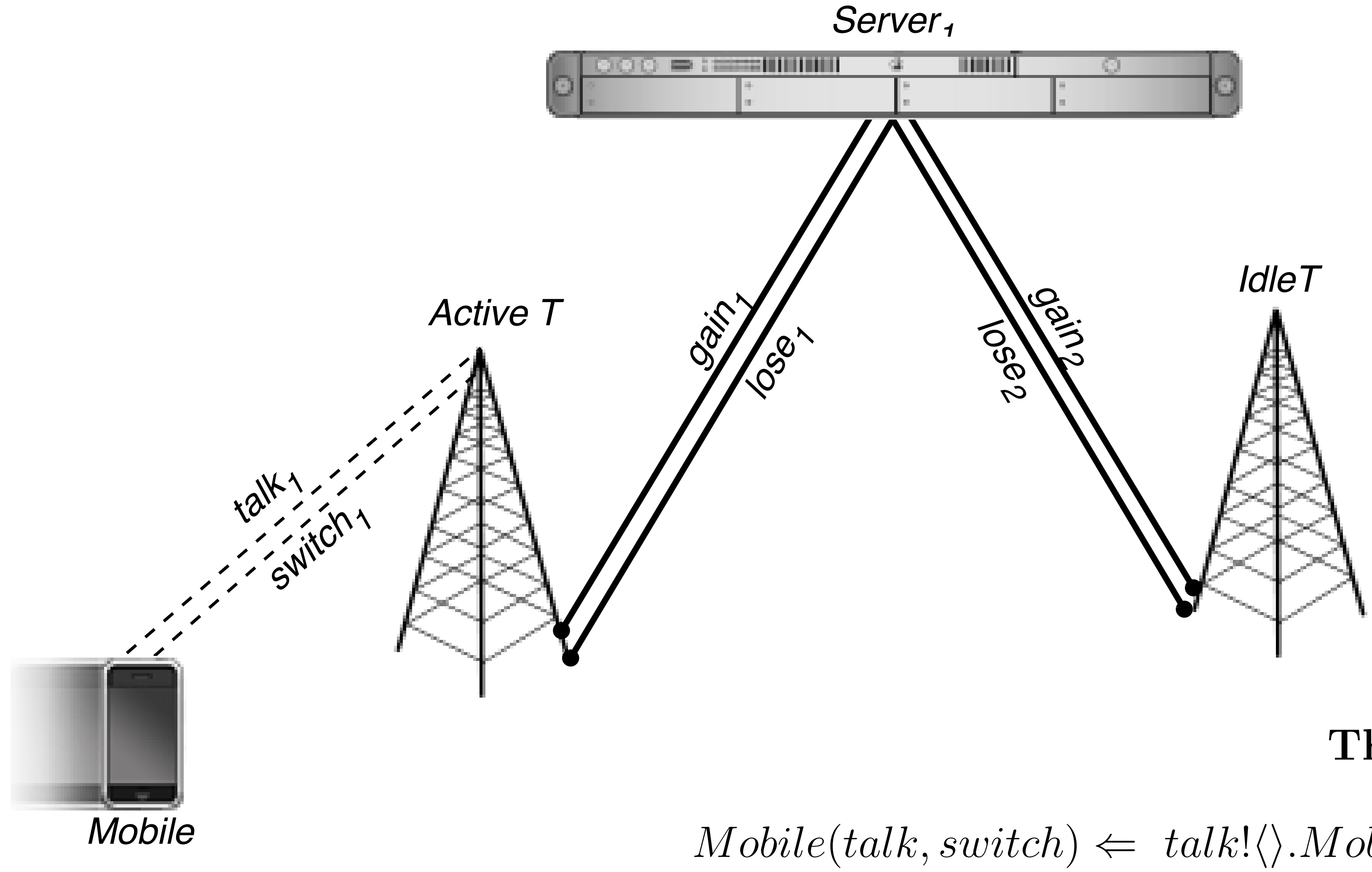
Distributed and Mobile Systems in the π calculus


Server
T_a
T_b
T_c
M_b
M_b



The Phone:

$$Mobile(talk, switch) \Leftarrow \; talk!\langle\rangle.Mobile\langle talk, switch\rangle + switch?(t,s).Mobile\langle t,s\rangle$$

The Tower:

$$\begin{aligned} ActiveT(talk, switch, gain, lose) &\Leftarrow talk?().ActiveT\langle talk, switch, gain, lose\rangle \\ &\quad + lose?(t,s).switch!\langle t,s\rangle.IdleT\langle gain, lose\rangle \\ IdleT(gain, lose) &\Leftarrow gain?(t,s).ActiveT\langle t,s,gain,lose\rangle \end{aligned}$$

The Server:

$$\begin{aligned} Server_1 &\Leftarrow \; lose_1!\langle talk_2, switch_2\rangle.gain_2!\langle talk_2, switch_2\rangle.Server_2 \\ Server_2 &\Leftarrow \; lose_2!\langle talk_1, switch_1\rangle.gain_1!\langle talk_1, switch_1\rangle.Server_1 \end{aligned}$$

Process terms

$R :=$	$R_1 \mid R_2$	Composition
	$n!\langle \overline{V} \rangle$	Send
	$n?(\overline{X}).R$	Receive
	$\mathrm{new}(n).R$	Restriction
	if $v_1 = v_2$ then R_1 else R_2	Matching
	rec $p.R$	Recursion
	stop	Termination

System

$\mathrm{new}(c_1, ..., c_n).R_1 \mid ... \mid R_m$	$n, m >= 0$

Figure 0.0.2: *Terms in the asynchronous* π*-calculus*

$$P'(a) \Leftarrow\ a?(x).(\mathrm{new}(n').(n'!\langle\rangle \mid c!\langle\rangle))$$

is α-equivalent to

$$P(a) \Leftarrow\ a?(x).(\mathrm{new}(n).(n!\langle\rangle \mid x!\langle\rangle))$$

(which terms are bound?)

Definition 0.0.3 (Structural Equivalence) Structural Equivalence, denoted $\equiv$ is the smallest contextual equivalence relation that satisfies the following axioms:

$$\begin{aligned}
P \mid Q &\equiv Q \mid P && \textsc{(s-comp-comm)}\\
(P \mid Q) \mid R &\equiv P \mid (Q \mid R) && \textsc{(s-comp-assoc)}\\
P \mid \mathit{stop} &\equiv P && \textsc{(s-comp-id)}\\
\mathrm{new}(c).\mathit{stop} &\equiv \mathit{stop} && \textsc{(s-rest-id)}\\
\mathrm{new}(c).\mathrm{new}(d).P &\equiv \mathrm{new}(d).\mathrm{new}(c).P && \textsc{(s-rest-comm)}\\
\mathrm{new}(c).(P \mid Q) &\equiv P \mid \mathrm{new}(c).Q, \text{ if } c \notin \mathrm{fi}(P) && \textsc{(s-rest-comp)}
\end{aligned}$$

Definition 0.0.4 (Reduction) The *reduction relation* $\longrightarrow$ is the smallest contextual relation that satisfies the following rules:

$$
\begin{array}{rcll}
c!\langle\overline{V}\rangle \mid c?(\overline{X}).R & \longrightarrow & R[\![^{\overline{V}}/_{\overline{X}}]\!] & \text{(R-COMM)} \\
\text{rec } p.R & \longrightarrow & R[\![^{\text{rec } p.R}/_{p}]\!] & \text{(R-REP)} \\
\text{if } v = v \text{ then } P \text{ else } Q & \longrightarrow & P & \text{(R-EQ)} \\
\text{if } v_1 = v_2 \text{ then } P \text{ else } Q & \longrightarrow & Q \quad (\text{where } v_1 \neq v_2) & \text{(R-NEQ)} \\
\multicolumn{3}{c}{\dfrac{P \equiv P', P \longrightarrow Q, Q \equiv Q'}{P' \longrightarrow Q'}} & \text{(R-STRUC)}
\end{array}
$$

We use the notation $P \cdots\!\longrightarrow Q$ when an arbitrary number of these rules have been applied in reducing P to Q.

Definition 0.0.5 (Action) The *action relation* $\rightarrow$ is the smallest relation between processes that satisfy the following rules:

$$c?(\overline{X}).R \quad \xrightarrow{c?\overline{V}} \quad \mathrm{R}[\![{}^{\overline{V}}/_{\overline{X}}]\!] \qquad \text{(A-IN)}$$

$$c!\langle\overline{V}\rangle \quad \xrightarrow{c!\overline{V}} \quad \mathit{stop} \qquad \text{(A-OUT)}$$

$$\text{rec } x.R \quad \xrightarrow{\tau} \quad R[\![{}^{\text{rec } x.R}/_{x}]\!] \qquad \text{(A-REP)}$$

$$\text{if } v = v \text{ then } P \text{ else } Q \quad \xrightarrow{\tau} \quad P \qquad \text{(A-EQ)}$$

$$\text{if } v_1 = v_2 \text{ then } P \text{ else } Q \quad \xrightarrow{\tau} \quad Q \qquad v_1 \neq v_2 \qquad \text{(A-NEQ)}$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad bn(\alpha) \cap fn(Q) = \emptyset \qquad \text{(A-COMP)}$$

$$\frac{P \xrightarrow{\alpha} P'}{\text{new}(b).P \xrightarrow{\alpha} \text{new}(b).P'} \qquad b \notin n(\alpha) \qquad \text{(A-REST)}$$

$$\frac{P \xrightarrow{(\overline{B})c!\overline{V}} P'}{\text{new}(n).P \xrightarrow{(n,\overline{B})c!\overline{V}} P'} \qquad n \neq c, n \text{ is in } \overline{V} \qquad \text{(A-OPEN)}$$

$$\frac{P \xrightarrow{c?\overline{V}} P',\ Q \xrightarrow{(\overline{B})c!\overline{V}} Q'}{P \mid Q \xrightarrow{\tau} \text{new}(\overline{B}).(P' \mid Q')} \qquad (\overline{B}) \cap fn(P) = \emptyset \qquad \text{(A-COMM)}$$

Action Prefixes

$\pi :=$	$n!\langle\overline{V}\rangle$	Send
	$n?(\overline{X})$	Receive

Process terms

$R :=$	$\sum_i \pi_i.R_i$	Summation
	$R_1 \mid R_2$	Composition
	$\mathrm{new}(n).R$	Restriction
	if $v_1 = v_2$ then R_1 else R_2	Matching
	rec $x.R$	Recursion
	stop	Termination

Figure 0.0.6: *Terms in the synchronous π-calculus*

Lemma 4.1 *Let P be a process of the asynchronous π-calculus. Assume that P can make two transitions $P \xrightarrow{\alpha_s} Q$ and $P \xrightarrow{\alpha_r} Q'$, where α_s is a send action while α_r is a receive action. Then there exists a process R such that $Q \xrightarrow{\alpha_r} R$ and $Q' \xrightarrow{\alpha_s} R$.*

Consider, for example:

$$P_0 \mid P_1 \Leftarrow\ c_0!\langle\rangle.o!\langle c_0\rangle + c_1?().o!\langle c_1\rangle \mid c_1!\langle\rangle.o!\langle c_1\rangle + c_0?().o!\langle c_0\rangle$$

Palamidessi uses the separation results to construct two criteria for encodings:

Uniformity:

$$[\![\sigma(P)]\!] = \sigma([\![P]\!])$$
$$[\![P \mid Q]\!] = [\![P]\!] \mid [\![Q]\!]$$

Reasonability:

the capability of a language to distinguish between two processes when their actions given channel differ.

Summation:

$$[\![\sum_i \pi_i.R_i]\!] \overset{\text{def}}{=} \text{new}(l).(l!\langle \mathit{true}\rangle \mid \prod_i [\![\pi_i.R_i]\!]_l)$$

Sending:

$$\begin{aligned}[\![c!\langle \overline{V}\rangle.P]\!]_r \overset{\text{def}}{=} &\text{new}(\mathit{ack}).(c!\langle r, \mathit{ack}, \overline{V}\rangle \mid \mathit{ack}?(x).\text{if } x = \mathit{true} \text{ then } [\![P]\!] \\ &\text{else (if } x = \mathit{retry} \text{ then } c!\langle r, \mathit{ack}, \overline{V}\rangle \text{ else } \mathit{stop}))\end{aligned}$$

Receiving:

$$[\![c?(\overline{X}).P]\!]_l \overset{\text{def}}{=} \text{rec } q.c?(r, \mathit{ack}, \overline{X}).(l, r)?\mathit{d\text{-}lock}.[\![P]\!]$$

$(\mathbf{l}, \mathbf{r})?$d-lock.P means:

$$\begin{aligned}&l?(x).(\text{if } x = \mathit{true} \\ &\quad \text{then } r?(y).(\text{if } y = \mathit{true} \\ &\quad\quad \text{then } l!\langle \mathit{false}\rangle \mid r!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{true}\rangle \mid P \\ &\quad\quad \text{else } l!\langle \mathit{true}\rangle \mid r!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{false}\rangle \mid q) \\ &\quad \text{else } l!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{retry}\rangle)\end{aligned}$$

Summation:

$$[\![\sum_i \pi_i.R_i]\!]^d \overset{\text{def}}{=} d?(n).(d!\langle n+1\rangle \mid \text{new}(l).(l!\langle \mathit{true}\rangle \mid \prod_i [\![\pi_i.R_i]\!]^d_{n,l}))$$

Sending:

$$\begin{aligned}[\![c!\langle \overline{V}\rangle.P]\!]^d_{n,l} \overset{\text{def}}{=} &\text{new}(\mathit{ack}).(c!\langle n,l,\mathit{ack},\overline{V}\rangle \mid \mathit{ack}?(x).\text{if } x = \mathit{true} \text{ then } [\![P]\!]^d \\ &\quad \text{else (if } x = \mathit{retry} \text{ then } c!\langle n,l,\mathit{ack},\overline{V}\rangle \text{ else } \mathit{stop}))\end{aligned}$$

Receiving:

$$\begin{aligned}[\![c?(\overline{X}).P]\!]^d_{n,l} \overset{\text{def}}{=} &\text{rec } q.(c?(m,r,\mathit{ack},\overline{X}).(\\ &\text{if } n = m \text{ then } (\mathit{ack}!\langle \mathit{retry}\rangle \mid q) \text{ else (} \\ &\text{if } n < m \text{ then } (l,r)?\mathit{d\text{-}lock}.[\![P]\!]^d \text{ else } (r,l)?\mathit{rd\text{-}lock}.[\![P]\!]^d)))\end{aligned}$$

$(\mathbf{l},\mathbf{r})$?d-lock.P means:

$$\begin{aligned}&l?(x).(\text{if } x = \mathit{true} \\ &\qquad \text{then } r?(y).(\text{if } y = \mathit{true} \\ &\qquad\qquad \text{then } l!\langle \mathit{false}\rangle \mid r!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{true}\rangle \mid P \\ &\qquad\qquad \text{else } l!\langle \mathit{true}\rangle \mid r!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{false}\rangle \mid q) \\ &\qquad \text{else } l!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{retry}\rangle)\end{aligned}$$

$(\mathbf{r},\mathbf{l})$?rd-lock.P means:

$$\begin{aligned}&r?(x).(\text{if } x = \mathit{true} \\ &\qquad \text{then } l?(y).(\text{if } y = \mathit{true} \\ &\qquad\qquad \text{then } l!\langle \mathit{false}\rangle \mid r!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{true}\rangle \mid P \\ &\qquad\qquad \text{else } l!\langle \mathit{false}\rangle \mid r!\langle \mathit{true}\rangle \mid \mathit{ack}!\langle \mathit{retry}\rangle) \\ &\qquad \text{else } r!\langle \mathit{false}\rangle \mid \mathit{ack}!\langle \mathit{false}\rangle \mid q)\end{aligned}$$