Primary Constructs

Data Types

$\mathcal{B}$ = byte arrays $\quad$ $\mathcal{L}$ = listeners $\quad$ $\mathcal{P}_\mathcal{I}$ = input ports $\quad$ $\mathcal{P}_\mathcal{O}$ = output ports

$\mathcal{T}$ = transports $\quad$ $\mathcal{U}$ = URIs $\quad$ $\varnothing$ = void

A *reference* is an opaque token that identifies an external resource. Listeners and transports are references. A *port* is a reliable, buffered, ordered byte stream. Ports either consume or produce bytes. A *URI* is a universal resource identifier as defined in RFC 3986, restricted here to URL authorities as defined in the roadmap.

API Functions

$\text{listen} : \mathcal{U} \to \mathcal{L}$ $\quad$ $\text{accept} : \mathcal{L} \to \mathcal{T}$ $\quad$ $\text{connect} : \mathcal{U} \to \mathcal{T}$ $\quad$ $\text{release} : \mathcal{L} \cup \mathcal{T} \to \varnothing$

$\text{send} : \mathcal{T} \times \mathcal{B} \to \varnothing$ $\quad$ $\text{receive} : \mathcal{T} \to \mathcal{B}$

Internal Functions

$\text{listener} : \mathcal{U} \to \mathcal{L}$ $\quad$ $\text{accepter} : \mathcal{L} \to \mathcal{T} \times \mathcal{U} \times \mathcal{P}_\mathcal{I} \times \mathcal{P}_\mathcal{O}$ $\quad$ $\text{connector} : \mathcal{U} \to \mathcal{T} \times \mathcal{U} \times \mathcal{P}_\mathcal{I} \times \mathcal{P}_\mathcal{O}$

$\text{sender} : \mathcal{B} \times \mathcal{P}_\mathcal{O} \to \mathcal{P}_\mathcal{O}$ $\quad$ $\text{receiver} : \mathcal{P}_\mathcal{I} \to \mathcal{B} \times \mathcal{P}_\mathcal{I}$

An *internal function* is a placeholder for implementation details that do not affect the API.

Run-time Environment

$\Lambda : \mathcal{L} \to \mathcal{U}$ $\quad$ $\Gamma : \mathcal{T} \to \mathcal{U} \times \mathcal{U}$ $\quad$ $\Pi : \mathcal{T} \to \mathcal{P}_\mathcal{I} \times \mathcal{P}_\mathcal{O}$

The run-time state consists of three maps. Adding a binding to a map tells the run-time to perform a computation with visible effects until the binding is removed. In Λ, $\cdot \mapsto u_L$ listens for connections on u_L. In Γ, $\cdot \mapsto (u_L, u_R)$ establishes a connection between u_L and u_R. In Π, $\cdot \mapsto (p_I, p_O)$ opens ports p_I and p_O for reading and writing. When $\Pi(t) = (p_I, p_O)$, re-binding $t \mapsto (p'_I, p_O)$ or $t \mapsto (p_I, p'_O)$ tells the run-time to read or write bytes from or to the port.

Operational Semantics

$$\frac{\text{listener}(u_L) = \ell}{\Lambda, \Gamma, \Pi \vdash \text{listen}(u_L) \rightsquigarrow [\ell \mapsto u_L]\Lambda, \Gamma, \Pi \vdash \ell} \textsc{Lsn}$$

$$\frac{\Lambda(\ell) = u_L \qquad \text{accepter}(\ell) = (t, u_R, p_I, p_O)}{\Lambda, \Gamma, \Pi \vdash \text{accept}(\ell) \rightsquigarrow \Lambda, [t \mapsto (u_L, u_R)]\Gamma, [t \mapsto (p_I, p_O)]\Pi \vdash t} \textsc{Acc}$$

$$\frac{\text{connector}(u_R) = (t, u_L, p_I, p_O)}{\Lambda, \Gamma, \Pi \vdash \text{connect}(u_R) \rightsquigarrow \Lambda, [t \mapsto (u_L, u_R)]\Gamma, [t \mapsto (p_I, p_O)]\Pi \vdash t} \textsc{Con}$$

$$\frac{}{\Lambda, \Gamma, \Pi \vdash \text{release}(\ell) \rightsquigarrow \Lambda \setminus \{\ell \mapsto \cdot\}, \Gamma, \Pi \vdash \varnothing} \textsc{RlsL}$$

$$\frac{}{\Lambda, \Gamma, \Pi \vdash \text{release}(t) \rightsquigarrow \Lambda, \Gamma \setminus \{t \mapsto \cdot\}, \Pi \setminus \{t \mapsto \cdot\} \vdash \varnothing} \textsc{RlsT}$$

$$\frac{\Pi(t) = (p_I, p_O) \qquad \text{sender}(b, p_O) = p'_O}{\Lambda, \Gamma, \Pi \vdash \text{send}(t, b) \rightsquigarrow \Lambda, \Gamma, [t \mapsto (p_I, p'_O)]\Pi \vdash \varnothing} \textsc{Snd}$$

$$\frac{\Pi(t) = (p_I, p_O) \qquad \text{receiver}(p_I) = (b, p'_I)}{\Lambda, \Gamma, \Pi \vdash \text{receive}(t) \rightsquigarrow \Lambda, \Gamma, [t \mapsto (p'_I, p_O)]\Pi \vdash b} \textsc{Rcv}$$

Lsn says listen(u_L) produces a listener ℓ on URL authority u_L. Acc says accept(ℓ) produces a transport t that, when URL authority u_R connects to u_L, represents the connection between u_L and u_R. Con says connect(u_R) produces a transport t that, when u_R accepts the connection from a "chosen" URL authority u_L, represents the connection between u_L and u_R. Snd says send(t, b) writes the bytes in b to the output port bound to t. Rcv says receive(t) produces all available bytes b from the input port bound to t.