Monitoring Networks through Multiparty Session Types

Laura Bocchi[1], Tzu-Chun Chen[2], Romain Demangeon[1], Kohei Honda[2], Nobuko Yoshida[1]

[2]Queen Mary, University of London, [1]Imperial College, London

FORTE/FMOODS 2013, Firenze, 03/06/2013

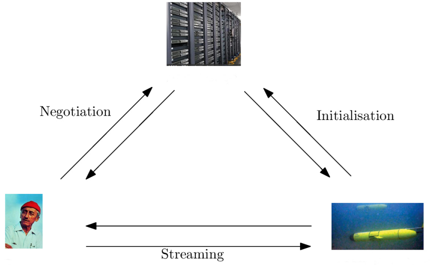
Background

Distributed programming

- Message-passing concurrency.
  - asynchronous networks.
  - examples: services, web applications, ...
- Distant interactions.
- Interoperability.

Distributed verification

- Desirable properties:
  - fidelity, lock-freedom, governance, security, ...
- Control is local[1] only.
- Typechecking impossible[2].



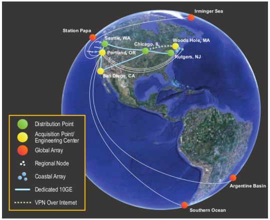
Communicating programs

- different languages, compilers, libraries.
- different hardware.
- different locations.

Solution: [2]Monitoring through [1]Multiparty Session-Types

Our collaboration: OOI

OCEAN OBSERVATORIES INITIATIVE



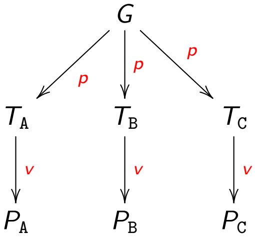
- international oceanography project.
  - vast, connected array of sensors, buoys, intruments.
    - using message-passing communications.
    - applications written in different languages, running on heterogenous hardware in an asynchronous network.
    - web-based user interface for oceanographs.
  - requires correct, safe interactions.

  ⇒ perfect framework for session type verification.
- integration into the CyberInfrastructure sub-project
  - design and maintain *Scribble*, a protocol language strongly based on MPST.

Session types

- Verification theory originating in typed π-calculi.
  - formal methods,
  - type systems naturally generate typecheckers.
- Principles:
  - design a specification for network interactions called session
    - session as atomic protocols (global types).
    - participants are abstracted in roles (Instrument, Buyer, ...).
    - specifies only the message layer.
  - project the session into local types
    - local behaviors for each endpoint.
  - ensure each endpoint conforms to its local type.
  - Fidelity: Local conformance implies global correction w.r.t. the specification.
- Multisession model: one participant can be engaged in several sessions.

Session types (II)



$G : \mathtt{A} \longrightarrow \mathtt{B}; \mathtt{B} \longrightarrow \mathtt{C}; \mathtt{B} \longrightarrow \mathtt{A}$

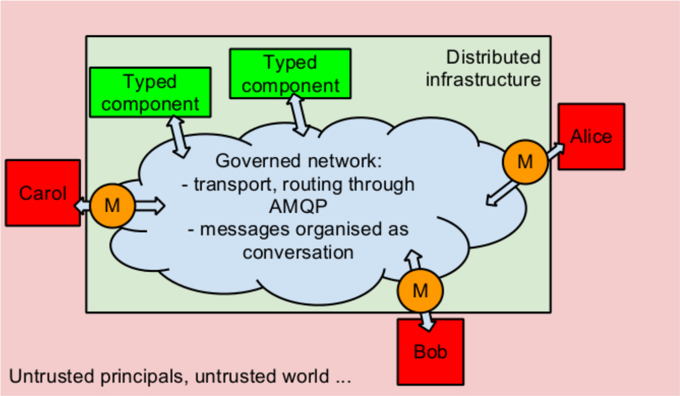
$T_{\mathtt{A}} : !\mathtt{B}.?\mathtt{A}$

$P_{\mathtt{A}} : \overline{s}[\mathtt{B}]!\langle 8\rangle.s[\mathtt{C}]?(y)\ldots$

- *p*: projection from global types to local types.
- *v*: verification of processes against local types.
  - type systems: ensure soundness and progress.
  - type-checkers not adapted to heterogenous networks.
  - suggests dynamic verification (monitors).

(session type theory includes choices, recursion, predicates, . . .)

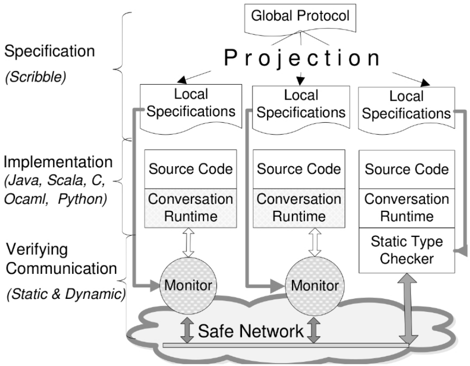
Monitors



- run in parallel (can be embedded or external).
- act as membranes between the trusted network and applications.
- ensure interoperability (no access to source code).
- unmonitored trusted components can be introduced.

Session types for monitoring

- Adapting MPST theory to monitoring.
- Allowing mixed networks.
- Principles:
  - developers design protocols for the whole network in a dedicated language,
  - well-formedness is checked,
  - protocols are projected into local types,
  - local types generate monitors,
    - or are statically typechecked.



Our Contribution

A theory for MPST-monitored networks:

- Formalise MPST-monitoring and asynchronous networks.
- Introduce monitors as first-class objects in the theory and make explicit routing information propagation.
- Compare different networks through equivalences.
- Justify monitoring by soundness theorems.
  - safety: monitors enforces specification conformance.
  - transparency: monitors does not affect correct behaviors.
  - fidelity: correspondence to global types is maintained.
- Ensure that theory interacts with implementation.

Formalism: MPST syntax

$$G \quad ::= \quad \mathtt{r}_1 \to \mathtt{r}_2 : \{l_i(x_i : S_i)\{A_i\}.G_i\}_{i \in I} \;\Big|\; G_1 \mid G_2 \;\Big|\; G_1 ; G_2 \;\Big|\; \mu \mathtt{t}.G \;\Big|\; \mathtt{t} \;\Big|\; \epsilon \;\Big|\; \mathsf{end}$$

$$T \quad ::= \quad \mathtt{r}!\{l_i(x_i : S_i)\{A_i\}.T_i\}_{i \in I} \;\Big|\; \mathtt{r}?\{l_i(x_i : S_i)\{A_i\}.T_i\}_{i \in I} \;\Big|\; T_1 \mid T_2 \;\Big|\; T_1 ; T_2 \;\Big|\; \mu \mathtt{t}.T \;\Big|\; \mathtt{t} \;\Big|\; \epsilon \;\Big|\; \mathsf{end}$$

- G: global types:
  - interaction from role $\mathtt{r}^1$ to role $\mathtt{r}^2$ (with choice),
  - parallel and sequential composition,
  - recursion and end.
- T: local types.
- A: predicates and expressions used to validate properties over messages inside types.

Formalism: Example

$$
\begin{aligned}
G_{\mathtt{ATM}} = \quad & \mathtt{C} \to \mathtt{A} : \{ \quad \mathtt{Login}(x_i : \mathsf{string})\{\mathtt{tt}\}. \\
& \mathtt{A} \to \mathtt{S} : \{ \quad \mathtt{LoginOK}()\{\mathtt{tt}\}.\ \mathtt{A} \to \mathtt{C} : \{\mathtt{LoginOK}()\{\mathtt{tt}\}.\ G_{\mathrm{Loop}}\}, \\
& \qquad\qquad\ \mathtt{LoginFail}()\{\mathtt{tt}\}.\ \mathtt{A} \to \mathtt{C} : \{\mathtt{LoginFail}()\{\mathtt{tt}\}.\ \mathsf{end}\}\}\} \\
G_{\mathrm{Loop}} = \quad & \mu\, \mathtt{LOOP}. \\
& \mathtt{S} \to \mathtt{C} : \{ \quad \mathtt{Account}(x_b : \mathsf{int})\{x_b \geq 0\}. \\
& \mathtt{C} \to \mathtt{S} : \{ \quad \mathtt{Withdraw}(x_p : \mathsf{int})\{x_p > 0 \land x_b - x_p \geq 0\}.\ \mathtt{LOOP}, \\
& \qquad\qquad\ \mathtt{Deposit}(x_d : \mathsf{int})\{x_d > 0\}.\ \mathtt{LOOP}, \\
& \qquad\qquad\ \mathtt{Quit}()\{\mathtt{tt}\}.\mathsf{end}\}\}
\end{aligned}
$$

- Protocol for interaction with ATM with three commands.
- Three roles are involved: Client, ATM and bank Server.
- Contains choices, nested loops and predicate checks.

Formalism: Example (projection)

$$T_{\mathtt{C}} = \quad \mathtt{A}!\{\mathtt{Login}(x_i : \mathsf{string})\{\mathsf{tt}\}.$$
$$\mathtt{A}?\{\mathtt{LoginOK}()\{\mathsf{tt}\}.\ T_{\mathtt{Loop}}$$
$$\mathtt{LoginFail}()\{\mathsf{tt}\}.\ \mathsf{end}\}\}$$

$$T_{\mathtt{Loop}} = \quad \mu\,\mathtt{LOOP}.$$
$$\mathtt{S}?\{\mathtt{Account}(x_b : \mathsf{int})\{x_b \geq 0\}.$$
$$\mathtt{S}!\{\mathtt{Withdraw}(x_p : \mathsf{int})\{x_p > 0 \wedge x_b - x_p \geq 0\}.$$
$$\mathtt{LOOP},$$
$$\mathtt{Deposit}(x_d : \mathsf{int})\{x_d > 0\}.\mathtt{LOOP},$$
$$\mathtt{Quit}()\{\mathsf{tt}\}.\mathsf{end}\}\}$$

- Projection of ATM example onto role Client.
- Session seen from the point of view of Client.
- Type used for monitoring (local enforcement).

Formalism: Networks

$$
\begin{array}{rcl}
P & ::= & \overline{a}\langle s[\mathtt{r}] : T\rangle \;\Big|\; a(y[\mathtt{r}] : T).P \;\Big|\; k[\mathtt{r}_1, \mathtt{r}_2]!l\langle e\rangle \;\Big|\; k[\mathtt{r}_1, \mathtt{r}_2]?\{l_i(x_i).P_i\}_{i\in I} \;\Big| \\
& & \text{if } e \text{ then } P \text{ else } Q \;\Big|\; P \mid Q \;\Big|\; \mathbf{0} \;\Big|\; \mu X.P \;\Big|\; X \;\Big|\; P; Q \;\Big|\; (\nu a)\ P \;\Big|\; (\nu s)P \\
N & ::= & [P]_\alpha \;\Big|\; N_1|N_2 \;\Big|\; \mathbf{0} \;\Big|\; (\nu a)N \;\Big|\; (\nu s)N \;\Big|\; \langle r\ ;\ h\rangle \\
r & ::= & a \mapsto \alpha \;\Big|\; s[\mathtt{r}] \mapsto \alpha \qquad h ::= m \cdot h \;\Big|\; \emptyset \qquad m ::= \overline{a}\langle s[\mathtt{r}] : T\rangle \;\Big|\; s\langle \mathtt{r}_1, \mathtt{r}_2, l\langle v\rangle\rangle
\end{array}
$$

- π-based calculus.
- Asynchronous networks composed of:
  - processes P located at principals α,
    - abstracts local applications.
  - router r,
    - abstracts network routing information, updated on-the-fly.
  - global queue h.
    - abstracts messages in transit.

Formalism: Semantics

$$
\begin{array}{lcl}
[\overline{a}\langle s[\mathtt{r}] : T\rangle]_\alpha \mid \langle r \;;\; h\rangle & \longrightarrow & [\mathbf{0}]_\alpha \mid \langle r \;;\; h \cdot \overline{a}\langle s[\mathtt{r}] : T\rangle\rangle \\
[a(y[\mathtt{r}] : T).P]_\alpha \mid \langle r \;;\; \overline{a}\langle s[\mathtt{r}] : T\rangle \cdot h\rangle & \longrightarrow & [P[s/y]]_\alpha \mid \langle r \cdot s[\mathtt{r}] \mapsto \alpha \;;\; h\rangle \;^{\dagger} \\
[s[\mathtt{r}_1,\mathtt{r}_2]!l_j\langle v\rangle]_\alpha \mid \langle r \;;\; h\rangle & \longrightarrow & [\mathbf{0}]_\alpha \mid \langle r \;;\; h \cdot s\langle \mathtt{r}_1,\mathtt{r}_2, l_j\langle v\rangle\rangle\rangle \;^{\dagger\dagger} \\
[s[\mathtt{r}_1,\mathtt{r}_2]?\{l_i(x_i).P_i\}_i]_\alpha \mid \langle r \;;\; s\langle \mathtt{r}_1,\mathtt{r}_2, l_j\langle v\rangle\rangle \cdot h\rangle & \longrightarrow & [P_j[v/x_j]]_\alpha \mid \langle r \;;\; h\rangle \;^{\dagger\dagger\dagger}
\end{array}
$$

$$
\dagger : r(a) = \alpha \qquad \dagger\dagger : r(s[\mathtt{r}_2]) \neq \alpha \qquad \dagger\dagger\dagger : r(s[\mathtt{r}_2]) = \alpha
$$

(reductions can happen inside contexts)

- Two rules for session invitations.
- Two rules for session interactions.
- Asynchrony is handled through global queue.
- Routing information is used and updated at runtime.

Specifications

- $\Sigma \ ::= \ \emptyset \ \big| \ \Sigma, \alpha : \langle \Gamma ; \Delta \rangle,$
  $\Gamma ::= \emptyset \ \big| \ \Gamma, a :?(T[\mathtt{r}]) \mid \Gamma, a :!(T[\mathtt{r}]) \quad \Delta ::= \emptyset \ \big| \ \Delta, s[\mathtt{r}] : T,$
  - Σ: spec., Δ: session env, Γ: shared env.
- Specifications have a semantics (used for satisfaction).

Monitored Networks

- Monitors $\mathrm{M} = \alpha : \langle \Gamma ; \Delta \rangle$ are introduced as component of monitored networks.
- Reduction rules for monitored networks (send rules):

$$\frac{\mathrm{M} \xrightarrow{s[\mathtt{r}_1,\mathtt{r}_2]!l\langle v \rangle} \mathrm{M}' \quad r(s[\mathtt{r}_2]) \neq \alpha}{[s[\mathtt{r}_1,\mathtt{r}_2]!l\langle v \rangle]_\alpha \mid \mathrm{M} | \langle r \ ; \ h \rangle \longrightarrow [\mathbf{0}]_\alpha \mid \mathrm{M}' | \langle r \ ; \ h \cdot s\langle \mathtt{r}_1, \mathtt{r}_2, l\langle v \rangle \rangle \rangle}$$

$$\frac{\mathrm{M} \overset{s[\mathtt{r}_1,\mathtt{r}_2]!l\langle v \rangle}{\not\longrightarrow}}{[s[\mathtt{r}_1,\mathtt{r}_2]!l\langle v \rangle]_\alpha \mid \mathrm{M} \mid \langle r \ ; \ h \rangle \longrightarrow [\mathbf{0}]_\alpha \mid \mathrm{M} \mid \langle r \ ; \ h \rangle}$$

Equivalences

- To compare networks, we use:
  - weak bisimulation $\approx$ over partial networks (i.e. without transport)
  - reduction-closed barbed congruence $\cong$ over networks.
- barbed congruence allows us to model interfaces:
  - 2 structurally different networks implementing the same services are equated,
  - structure is hidden through routing.

Interface: example

$$
\begin{array}{rl}
G^2_{\text{Loop}} = & \mu\,\mathtt{LOOP}. \\
& \mathtt{S} \to \mathtt{T} : \{\ \mathtt{Query}()\{\mathsf{true}\}. \\
& \mathtt{T} \to \mathtt{S} : \{\ \mathtt{Answer}(x_t : \mathsf{int})\{\mathsf{true}\}. \\
& \mathtt{S} \to \mathtt{C} : \{\ \mathtt{Account}(x_b : \mathsf{int})\{x_b \geq 0\}. \\
& \mathtt{C} \to \mathtt{S} : \{\ \mathtt{Withdraw}(x_p : \mathsf{int})\{x_p \geq 0 \wedge x_b - x_p \geq 0\}.\ \mathtt{LOOP}, \\
& \qquad\qquad \mathtt{Deposit}(x_d : \mathsf{int})\{x_d > 0\}.\ \mathtt{LOOP}, \\
& \qquad\qquad \mathtt{Quit}()\{\mathsf{true}\}.\mathsf{end} \qquad \}\}\}\}
\end{array}
$$

- same protocol, except makes use of a transaction agent.
- $P_{\mathtt{S}}$: original server program, $P^2_{\mathtt{S}}$: new server program, $P_{\mathtt{T}}$: agent program.
- $$
\begin{array}{rl}
& ([P_{\mathtt{S}}]_\alpha \mid \langle \emptyset\ ;\ s[\mathtt{S}] \mapsto \alpha, s[\mathtt{C}] \mapsto \beta, s[\mathtt{A}] \mapsto \gamma \rangle) \\
\cong & ([P^2_{\mathtt{S}}]_\alpha \mid [P_{\mathtt{T}}]_\delta \mid \langle \emptyset\ ;\ s[\mathtt{S}] \mapsto \alpha, s[\mathtt{C}] \mapsto \beta, s[\mathtt{A}] \mapsto \gamma, s[\mathtt{T}] \mapsto \delta \rangle)
\end{array}
$$

Satisfaction

The satisfaction relation $\models N : \Sigma$ relates networks and specification:

- if Σ expects an input, N should be able to process it.
- if N performs an output, Σ should be expecting it.
- still holds after reduction (coinductive definition).

- Tailored for monitoring.
  - monitors do not enforce liveness.

Satisfaction equivalence

If $N_1 \cong N_2$ and $\models N_1 : \Sigma$ then $\models N_2 : \Sigma$.

Results (Safety)

Local Safety

$\models [P]_\alpha \mid \mathsf{M} : \alpha : \langle \Gamma; \Delta \rangle$ with $\mathsf{M} = \alpha{:}\langle \Gamma; \Delta \rangle$.

- A monitored process satisfies its specification.

Global Safety

If N is fully monitored w.r.t. Σ, then $\models \mathsf{N} : \Sigma$.

- monitored networks behave as expected.
- does not ensure liveness.

Results (Transparency)

Local Transparency

If $\models [P]_\alpha : \alpha : \langle \Gamma; \Delta \rangle$, then $[P]_\alpha \approx ([P]_\alpha \mid \mathsf{M})$ with $\mathsf{M} = \alpha : \langle \Gamma; \Delta \rangle$.

- unmonitored correct processes are undistinguishable from their monitored counterparts.
- allows one to mix monitored and typechecked processes.

Global Transparency

Assume N and N have the same global transport $\langle r\ ;\ h \rangle$.
Assume:

1. N is fully monitored w.r.t. Σ and
2. $N = M \mid \langle r\ ;\ h \rangle$ is unmonitored but $\models M : \Sigma$.

We have $\mathsf{N} \cong N$.

- monitors does not alterate behaviors of correct networks.
- monitor actions are not observable on correct components.

Results (Fidelity)

- a configuration is consistent: when it corresponds to a well-formed array of global types $(G_1, \ldots, G_n)$ through projection.
- conformance is satisfaction + receivability (queue can be emptied).

Session Fidelity

Assume:

1. configuration $\Sigma; \langle r \ ; \ h \rangle$ is consistent,
2. network $N \equiv M | \langle r \ ; \ h \rangle$ conforms to configuration $\Sigma; \langle r \ ; \ h \rangle$.

For any ℓ, whenever we have $N \xrightarrow{\ell}_{\mathrm{g}} N'$ s.t. $\Sigma; \langle r \ ; \ h \rangle \xrightarrow{\ell}_{\mathrm{g}} \Sigma'; \langle r' \ ; \ h' \rangle$, it holds that $\Sigma'; \langle r' \ ; \ h' \rangle$ is consistent and N' conforms to $\Sigma'; \langle r' \ ; \ h' \rangle$.

- consistence is preserved by reduction,
- at any time, the network correspond to a well-formed specification.

Conclusion

- A theory for monitoring through MPST inside asynchronous networks:
  - models monitor behaviors,
  - models dynamic routers,
  - monitoring ensures correction,
  - equate networks with the same interface.
- Implementation is done.
- Future works:
  - Ongoing partnership with OOI.
  - Express governance properties.
  - Handle interruptions and exceptional behaviors.