A Formal Model of Provenance in Distributed Systems

Issam Souilah\textsuperscript{2}  Adrian Francalanza\textsuperscript{1}  Vladimiro Sassone\textsuperscript{2}

\textsuperscript{1}Department of Computer Science
ICT, University of Malta

\textsuperscript{2}DSSE, ECS
Southampton University

San Francisco, USA. February 2009
Outline

Motivation

Proposed solution

Provenance Correctness

Conclusion
Motivation

Trust In a Distributed System
Motivation

Trust In a Distributed System

- Distribution ⇒ inherent parallelism.
Motivation

Trust In a Distributed System

- Distribution ⇒ inherent parallelism.
- Distribution ⇒ no shared memory i.e., message passing.
Motivation

Trust In a Distributed System

- Distribution ⇒ inherent **parallelism**.
- Distribution ⇒ no shared memory *i.e.*, **message passing**.
- Distribution ⇒ lack of centralised coordination *i.e.*, **non-determinism**.
Motivation

Trust In a Distributed System

- Distribution ⇒ inherent parallelism.
- Distribution ⇒ no shared memory i.e., message passing.
- Distribution ⇒ lack of centralised coordination i.e., non-determinism.

Appropriate Calculus?

The piCalculus:
Motivation

Trust In a Distributed System

- Distribution ⇒ inherent parallelism.
- Distribution ⇒ no shared memory i.e., message passing.
- Distribution ⇒ lack of centralised coordination i.e., non-determinism.

Appropriate Calculus?
The piCalculus:
- Captures the characteristic features of our domain of study.
Motivation

Trust In a Distributed System

- Distribution ⇒ inherent parallelism.
- Distribution ⇒ no shared memory i.e., message passing.
- Distribution ⇒ lack of centralised coordination i.e., non-determinism.

Appropriate Calculus?

The piCalculus:

- Captures the characteristic features of our domain of study.
- Well-studied.
Motivation

Trust In a Distributed System

- Distribution $\Rightarrow$ inherent **parallelism**.
- Distribution $\Rightarrow$ no shared memory *i.e.*, **message passing**.
- Distribution $\Rightarrow$ lack of centralised coordination *i.e.*, **non-determinism**.

Appropriate Calculus?

The **piCalculus**:

- Captures the characteristic features of our domain of study.
- Well-studied.
- Close connections to linear logic and resources
- ...
Names

\[ a, b, \ldots \in \text{NAMES} \]
piCalculus Primer

Names

\[ a, b, \ldots \in \text{Names} \]

- denote points of interaction (rendez-vous \textit{channels}).

...
Names

\[ a, b, \ldots \in \text{Names} \]

- denote points of interaction (rendez-vous \textbf{channels})
- \ldots and \textbf{values} which are transmitted during communication.
Processes

\[ P \parallel Q \parallel R \]
Processes

\[ a!b \parallel Q \parallel R \]
Processes

\[ a!b \parallel a?x.c!x \parallel R \]
Processes

\[ a!b \parallel a?x.c!x \parallel R \]
Processes

\[ a!b \parallel a?x.x!c \parallel R \]
Processes

\[ a!b \parallel a?x.x!c \parallel R \]

react on \( a \)
piCalculus Primer

Processes

\[ b!c \parallel R \]
Processes

\[ b!c \parallel b?y.R' \]
piCalculus Extension: Distribution!

From Processes to Systems

\[ P \parallel Q \parallel R \]
piCalculus Extension: Distribution!

From Processes to Systems

\[ p, q, r, \ldots \in \text{TRUSTPRINCIPALS} \]

\[ p[P] \parallel q[Q] \parallel p[R] \]
piCalculus Extension: Distribution!

From Processes to Systems

\[ p[a?x.P] \parallel q[a!v_1] \parallel r[R] \]
piCalculus Extension: Distribution!

From Processes to Systems

\[ p[a?x.P] \parallel q[a!v_1] \parallel r[R] \]

\textbf{across units of trust}
piCalculus Extension: Distribution!

From Processes to Systems

\[ p[a?x.P] \parallel q[a!v_1] \parallel r[a!v_2] \]

\underline{Market of values!}
piCalculus Extension: Distribution!

From Processes to Systems

\[
\text{communication}
\]

\[
p[a?x.P] \parallel q[a!v_1] \parallel r[a!v_2]
\]

\textbf{Market} of values!
piCalculus Extension: Distribution!

From Processes to Systems

communication

\[ \langle p[a?x.P] \parallel q[a!v_1] \parallel r[a!v_2] \rangle \]

Market of values!
Main Problem

What are the control mechanism need to assist consumers of data?

*How do we automate decisions based on trust?*
Main Problem

What are the control mechanism need to assist consumers of data?

How do we automate decisions based on trust?

▶ Static Analysis \textit{(not scalable)}
What are the control mechanism need to assist consumers of data?

How do we automate decisions based on trust?

- Static Analysis (*not scalable*)
- Dynamic Analysis:
  - decisions need to be computationally lightweight.
  - decision criteria produced in lightweight fashion.
What are the control mechanism need to assist consumers of data?

How do we automate decisions based on trust?

- Static Analysis (*not scalable*)
- Dynamic Analysis:
  - decisions need to be computationally lightweight. (*Full-blown verification methods do not cut it!*)
  - decision criteria produced in lightweight fashion.
Main Problem

What are the control mechanism need to assist consumers of data?

*How do we automate decisions based on trust?*

- Static Analysis *(not scalable)*
- Dynamic Analysis:
  - decisions need to be computationally lightweight. *(Full-blown verification methods do not cut it!)*
  - decision criteria produced in lightweight fashion. *(Proof-Carrying Code does not cut it!)*
Outline

Motivation

Proposed solution

Provenance Correctness

Conclusion
Annotated Values

\[ \nu : \]

\[ \nu ::= \epsilon \quad \| \quad \alpha; \nu \]

\[ \alpha ::= \text{rcv}(p, \nu) \quad \text{receive action} \]
\[ \quad \| \quad \text{snd}(p, \nu) \quad \text{send action} \]
Provenance

Annotated Values

\[ V : \kappa \]
Annotated Values

\[ V : \kappa \]

\[ \kappa ::= \epsilon \quad \text{empty provenance} \]
\[ \quad | \quad \alpha ; \kappa \quad \text{sequenced provenance} \]
\[ \alpha ::= \text{rcv}(p, \kappa) \quad \text{receive action} \]
\[ \quad | \quad \text{snd}(p, \kappa) \quad \text{send action} \]
Annotated Values

\[ p[a!v] \]
Annotated Values

\[ p[a : \kappa_a \; v : \kappa_v] \]
Provenance

Annotated Values

\[ p[a : \kappa_a ! v : \kappa_v] \parallel q[a : \kappa'_a ! v' : \kappa_{v'}.] \parallel p[a : \kappa''_a ! v'' : \kappa_{v''}.] \]

provenance is linear!
Automated:

1. orthogonal to programming (can be abstracted away)
2. ensures provenance annotation standardization.
3. avoids circular reasoning with respect to trust.

Two tiered architecture:

- **Computation Layer:** describes computation of values and processes.
- **Provenance Tracking Layer:** describes the aggregation of provenance information attached to data (typically assigned to a trusted middleware).
Automated:

1. orthogonal to programming (*can be abstracted away*)
Automated:

1. orthogonal to programming *(can be abstracted away)*
2. ensures provenance annotation standardization.
Provenance Tracking

Automated:

1. orthogonal to programming (*can be abstracted away*)
2. ensures provenance annotation standardization.
3. avoids circular reasoning with respect to trust.
Provenance Tracking

Automated:

1. orthogonal to programming *(can be abstracted away)*
2. ensures provenance annotation standardization.
3. avoids circular reasoning with respect to trust.

Two tiered architecture:
Automated:

1. orthogonal to programming (can be abstracted away)
2. ensures provenance annotation standardization.
3. avoids circular reasoning with respect to trust.

Two tiered architecture:

- **Computation Layer:** describes computation of values and processes.
Automated:

1. orthogonal to programming (*can be abstracted away*)
2. ensures provenance annotation standardization.
3. avoids circular reasoning with respect to trust.

Two tiered architecture:

- **Computation Layer**: describes computation of values and processes.
- **Provenance Tracking Layer**: describes the aggregation of provenance information attached to data (*typically assigned to a trusted middleware*)
Provenance Tracking Semantics

Operational Semantics

\[ p[a!v] \parallel Q \rightarrow a\langle v \rangle \parallel Q \]
Provenance Tracking Semantics

Operational Semantics

\[ p[a!v] \parallel Q \quad \rightarrow \quad a\langle v \rangle \parallel Q \]

loose immediate provenance information!
(Provenance Tracking) Operational Semantics

\[ p[a: \kappa_a ! v: \kappa_v] \parallel Q \rightarrow a\langle v: \text{snd}(p, \kappa_a); \kappa_v \rangle \parallel Q \]

provenance aggregation
Not-Automated!
Not-Automated!

- program with it to control non-derminism...
Not-Automated!

- program with it to control non-derminism...
- ...using intuitive programming idioms/constructs
Not-Automated!

- program with it to control non-derminism...
- ...using intuitive programming idioms/constructs

Operational Semantics

\[ a\langle v \rangle \parallel q[a?(x).Q] \rightarrow q[Q\{v/x\}] \]
Provenance Usage

Not-Automated!

- program with it to control non-determinism...
- ...using intuitive programming idioms/constructs

Operational Semantics

\[ a\langle v : \kappa_v \rangle \parallel q[a : \kappa_a ?(x \textbf{from } \pi).Q] \longrightarrow q[Q\{v/x\}] \quad \text{if } \kappa_v \models \pi \]

provenance pattern matching
Provenance Usage

Not-Automated!

- program with it to control non-derminism...
- ...using intuitive programming idioms/constructs

Operational Semantics

\[ a\langle v:\kappa_v \rangle \parallel q[a:\kappa_a ?(x\ from\ \pi).Q] \rightarrow q[Q\{v:rcv(q,\kappa_a);\kappa_v/x\}] \quad \text{if } \kappa_v \models \pi \]

- provenance aggregation
Provenance Usage Example

Client/Server

$\textit{srv}$ Name of server

$\textit{ret}$ Name of return channel on which server returns answer

Client = $p[\textit{srv}!\langle\textit{ret}\rangle] \parallel p[\textit{ret}? (x \textbf{from} \pi).P]$

Server = $q[\textit{srv}? (y \textbf{from} \ast).y!\langle v\rangle]$
Provenance Usage Example

Client/Server

\( srv \)  Name of server

\( ret \)  Name of return channel on which server returns answer

\begin{align*}
\text{Client} & = \ p[ \text{srv!}\langle ret\rangle ] \parallel p[ ret?(x \text{ from } \pi).P ] \\
\text{Server} & = \ q[ \text{srv?}(y \text{ from } \ast).y!\langle v\rangle ]
\end{align*}
Provenance Usage Example

Client/Server

\[\text{srv} \quad \text{Name of server}\]
\[\text{ret} \quad \text{Name of return channel on which server returns answer}\]

\[
\begin{align*}
\text{Client} & = \ p[\text{srv}!(\text{ret})] \ || \ p[\text{ret}?(x \ \text{from } \pi).P] \\
\text{Server} & = \ q[\text{srv}?(y \ \text{from } *).y!(v)]
\end{align*}
\]
Provenance Usage Example

Client/Server

\textit{srv} Name of server
\textit{ret} Name of return channel on which server returns answer

\textbf{Client} = \textit{p[srv!\langle ret\rangle]} \parallel \textit{p[ret?(x \textbf{from } \pi).P]}

\textbf{Server} = \textit{q[srv?(y \textbf{from } \ast).y!\langle v\rangle]}
Provenance Usage Example

Client/Server

\( \text{srv} \) Name of server

\( \text{ret} \) Name of return channel on which server returns answer

\[
\text{Client} = p[ \text{srv}: \kappa^{1}_{\text{srv}} \langle \text{ret}: \epsilon \rangle ] \parallel p[ \text{ret}: \epsilon ?(x \text{ from } \pi) . P ]
\]

\[
\text{Server} = q[ \text{srv}: \kappa^{2}_{\text{srv}} ?(y \text{ from } \ast) . y! \langle v: \kappa_{v} \rangle ]
\]
Provenance Usage Example

Client/Server

\(\text{srv}\) Name of server
\(\text{ret}\) Name of return channel on which server returns answer

\[\begin{align*}
\text{Client} & = p[\text{srv}:\kappa_{\text{srv}}^1 \langle \text{ret}:\epsilon \rangle] \parallel p[\text{ret}:\epsilon ?(x \textbf{from} \pi).P] \\
\text{Server} & = q[\text{srv}:\kappa_{\text{srv}}^2 ?(y \textbf{from} \ast).y!\langle v:\kappa_v \rangle]
\end{align*}\]
Provenance Usage Example

Client/Server

\(srv\)  Name of server
\(ret\)  Name of return channel on which server returns answer

\[
\begin{align*}
\text{Client} &= p[srv: \kappa^1_{srv} !\langle ret: \epsilon \rangle] \parallel p[ret: \epsilon ?(x \textbf{ from } \pi).P] \\
\text{Server} &= q[srv: \kappa^2_{srv} ?(y \textbf{ from } \ast).y!\langle v : \kappa_v \rangle]
\end{align*}
\]

\[\pi = \text{snd}(q, \text{rcv}(q, \text{snd}(p, \epsilon))) ; \ast\]
Provenance Correctness

Correctness Intuition

- provenance attached to values records *history related* to that value.
- provenance of a value is correct if it describes a *partial history* which corresponds to the *total history* of events.
Provenance Correctness

History represented by Logs

\[
\begin{align*}
\phi &::= \emptyset \mid \rho ; \phi & \text{ logs} \\
\rho &::= \text{rcv}(\rho) \mid \text{snd}(\rho) & \text{ log actions}
\end{align*}
\]
Provenance Correctness

History represented by Logs

\[ \phi ::= \emptyset \mid \rho; \phi \]

\[ \rho ::= \text{rcv}(\rho) \mid \text{snd}(\rho) \]

Sub-Log Comparison

\begin{align*}
\text{cmp1} & : \emptyset \leq \phi \\
\text{cmp2} & : \phi \leq \phi' \\
\text{cmp3} & : \phi \leq \rho; \phi' \leq \rho; \phi'
\end{align*}
Provenance Correctness

Monitored Systems

$$\phi \triangleright p[a : \kappa_a ! v : \kappa_v] \parallel Q \quad \xrightarrow{\text{mon}} \quad \text{snd}(p); \phi \triangleright a \langle v : \text{snd}(p, \kappa_a); \kappa_v \rangle \parallel Q$$
Provenance Correctness

Monitored Systems

\[ \phi \triangleright p[a : \kappa_a ! v : \kappa_v] \parallel Q \quad \xrightarrow{\text{mon}} \quad \text{snd}(p); \phi \triangleright a \langle v : \text{snd}(p, \kappa_a); \kappa_v \rangle \parallel Q \]

Erasure Function:

\[ | - | : \text{MONITOREDSYS} \rightarrow \text{SYS} \]
Provenance Correctness

Monitored Systems

\[ \phi \triangleright p[a : \kappa_a ! v : \kappa_v] \parallel Q \quad \longrightarrow_{\text{mon}} \quad \text{snd}(p); \phi \triangleright a(v : \text{snd}(p, \kappa_a); \kappa_v) \parallel Q \]

Erasure Function:

\[ | - | : \text{MONITOREDSYS} \rightarrow \text{SYS} \]

Lemma

\[ M \longrightarrow_{\text{mon}} M' \quad \text{implies} \quad |M| \longrightarrow |M'| \]
Provenance Correctness

Partial Log Extraction

\[ pLog : \kappa \rightarrow \mathbb{P}(\phi) \]

\[ pLog(\epsilon) = \emptyset \]
\[ pLog(rcv(p, \kappa); \kappa') = rcv(p); pLogV(\kappa') \cup pLog(\kappa) \]
\[ pLog(snd(p, \kappa); \kappa') = snd(p); pLogV(\kappa') \cup pLog(\kappa) \]

\[ pLogV(\epsilon) = \emptyset \]
\[ pLogV(rcv(p, \kappa); \kappa') = rcv(p); pLogV(\kappa') \]
\[ pLogV(snd(p, \kappa); \kappa') = snd(p); pLogV(\kappa') \]
Definition

$M$ has correct provenance iff $\forall \phi \in \mathbf{pLog} (\mathbf{prov}(M))$ we have $\phi \leq \log(M)$.

Theorem

$M$ has correct provenance and $M \xrightarrow{\text{mon}} M'$ implies $M'$ has correct provenance.
Outline

Motivation

Proposed solution

Provenance Correctness

Conclusion
Conclusions

- Designed a provenance based calculus for distributed computing.
- Proposed a two-tier system for provenance tracking and usage.
- Defined provenance correctness
- Proved provenance correctness for our provenance tracking semantics.
Conclusions

Thank You... Questions?