Business Process Management

Theory: The Pi-Calculus

Frank Puhlmann
Business Process Technology Group
Hasso Plattner Institut
Potsdam, Germany

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

What happens here?

- We discuss the application of a general theory for the description of mobile systems into the area of BPM and its wider parts

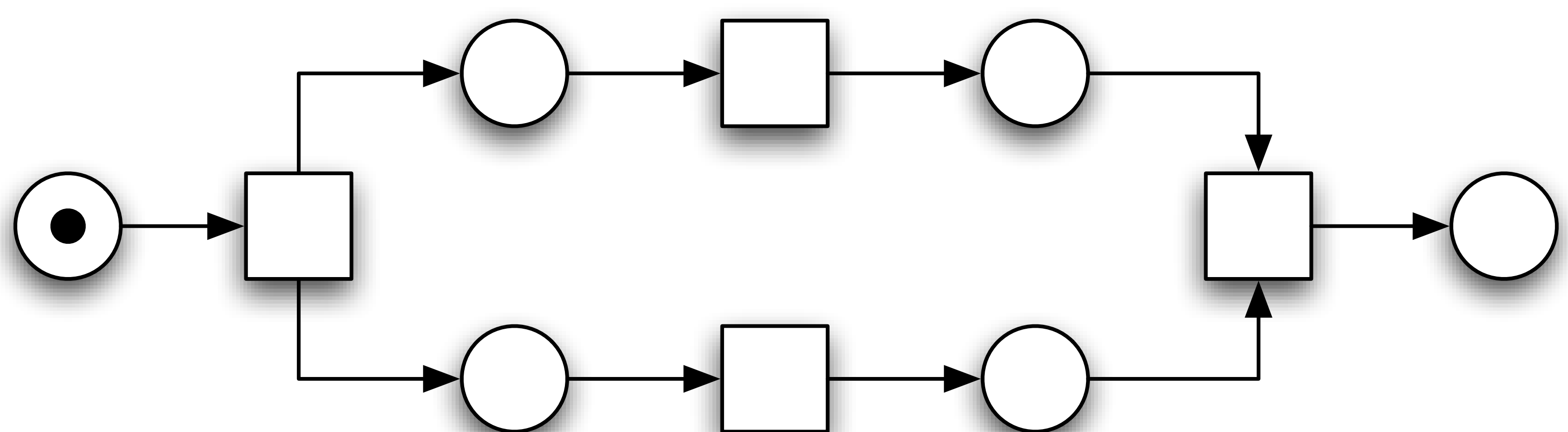
What are mobile systems?

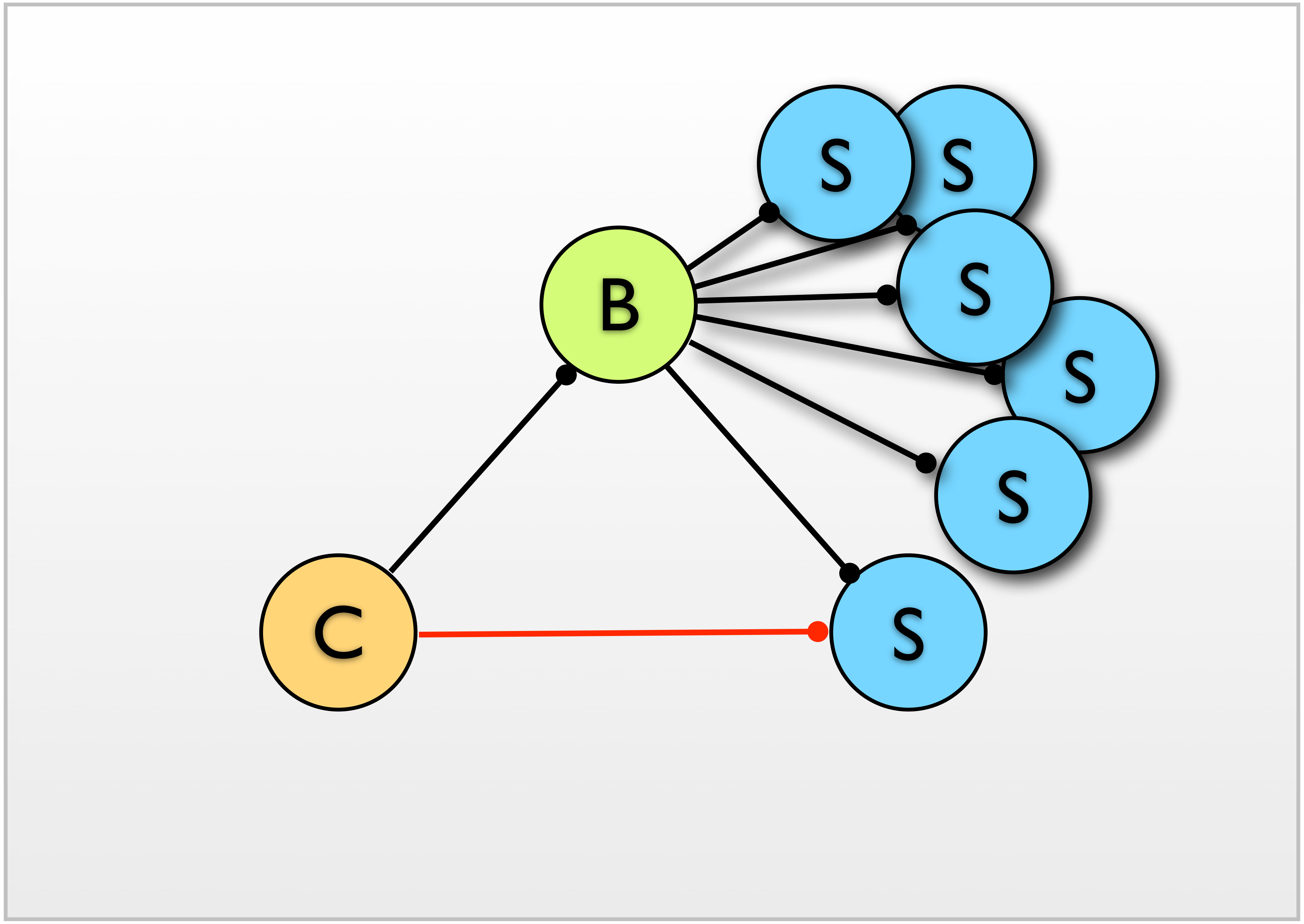
- Mobile systems are made of entities that move in a certain space
- Different kinds of mobility:

1. Links that move in an abstract space of linked processes
2. Processes that move in an abstract space of linked processes

Dynamic Topologies

- Mobile systems describe behavior with dynamic topologies, i.e. changing structures
- This is contrary to static structures for the description of behavior, i.e. Petri nets:





Link Passing Mobility

Outline Pi-Calculus Part

- Motivation
- The Theory of the Pi-Calculus
- Workflow and Data Patterns
- Application of the Pi-Calculus to BPM
- Verification

Motivation

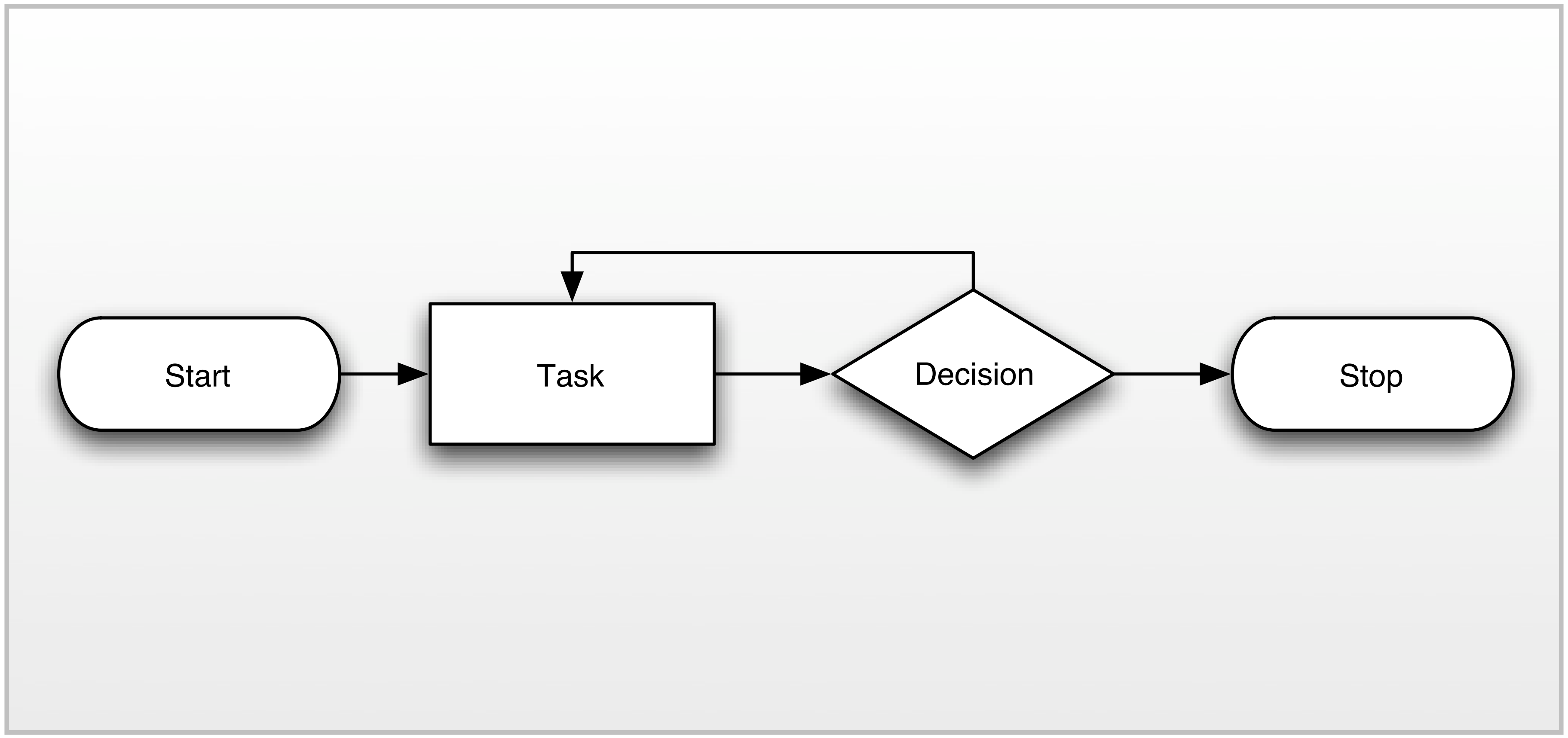
The Shifting Focus

A Shift in Theoretical Foundations

- From: Sequential systems
  - Lambda-Calculus (Church, Kleene, ≈1930)
- Over: Parallel systems
  - Petri nets (Petri, ≈1960)
- To: Mobile systems
  - Pi-Calculus (Milner, Parrow, Walker ≈1990)

The Lambda-Calculus

- Defined to investigate the definition of functions which are used for sequential computing
  - Precise definition of a computable function
  - Recursion
- Algebra: Compositional Structure
- Smallest universal programming language

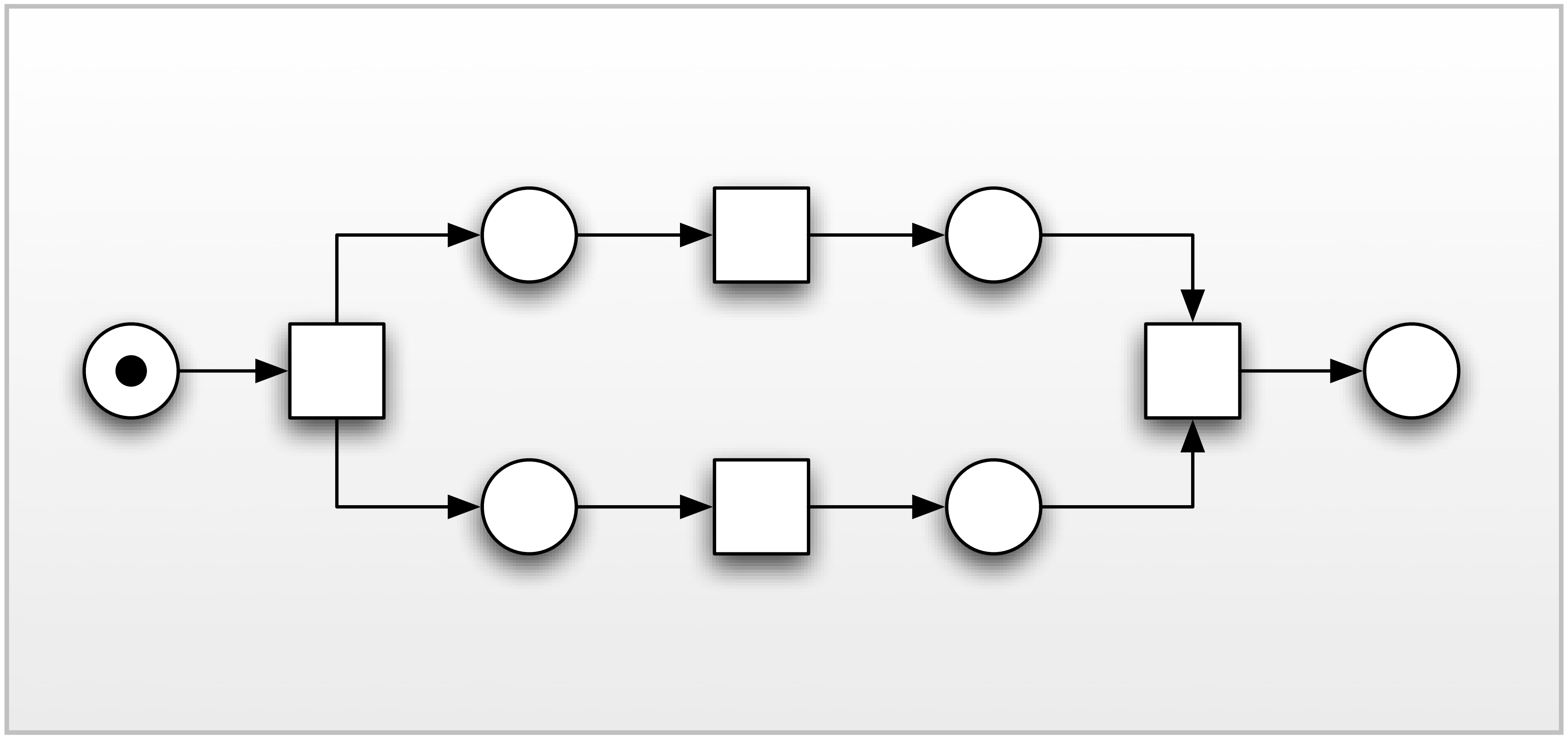


Sequential System

Petri nets

- Business processes require parallelism
  - Split, Joins
  - Dependencies
- Petri nets build a foundation for BPM
  - Explicit states and structure
  - Strong visualization

(C) 2007 Frank Puhlmann



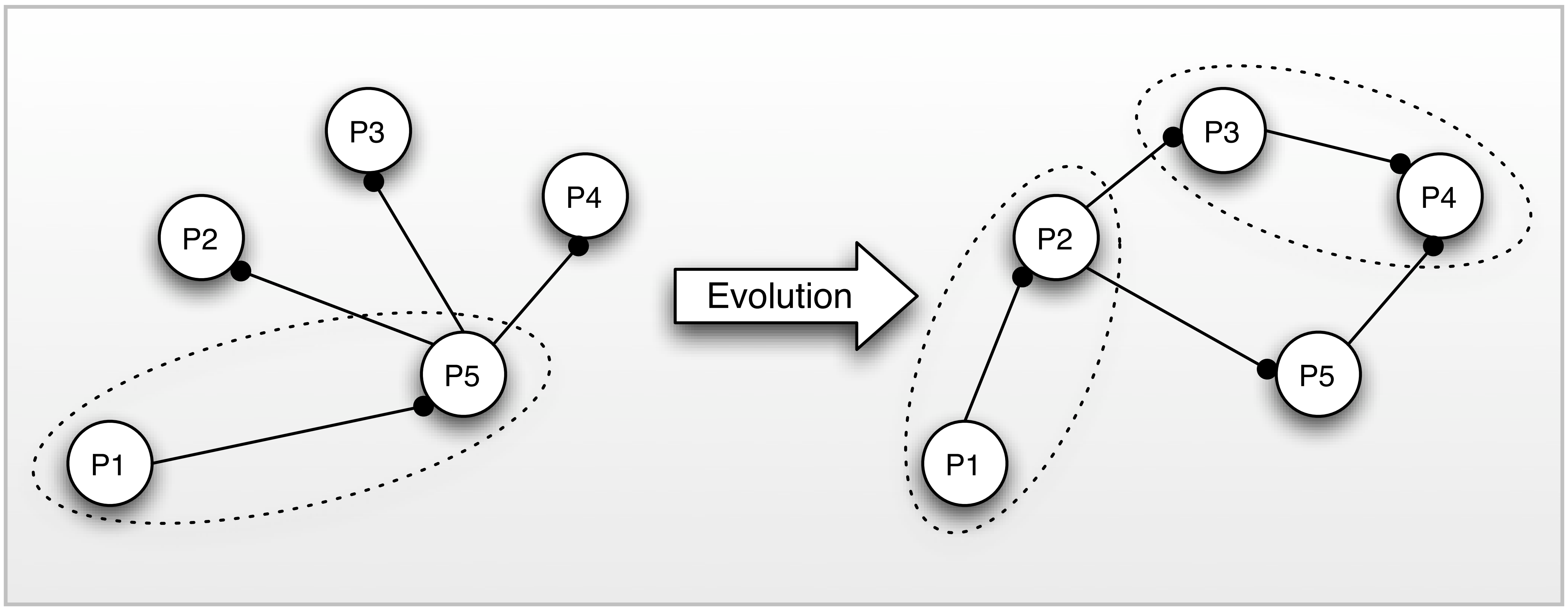
Parallel System

Petri net drawbacks

- Good and Bad: Static structure
- No advanced composition
- Regarding behavioral workflow patterns:
  - Excellent support for basic tasks
  - Poor support for advanced tasks

The Pi-Calculus

- Describes mobile systems
  - Agents (processes) interacting by
  - Names with agile scopes
- Is an algebra



Mobile System

The Pi-Calculus Advantage

- Overcomes the limitations of static structures
- Has the pros and cons of an algebra
- Supports all behavioral workflow patterns

Why mobile systems?

- What's wrong with BPM and Petri nets?
- Why do we need mobile instead of parallel systems?
  - Strong discussion between academics and practitioners

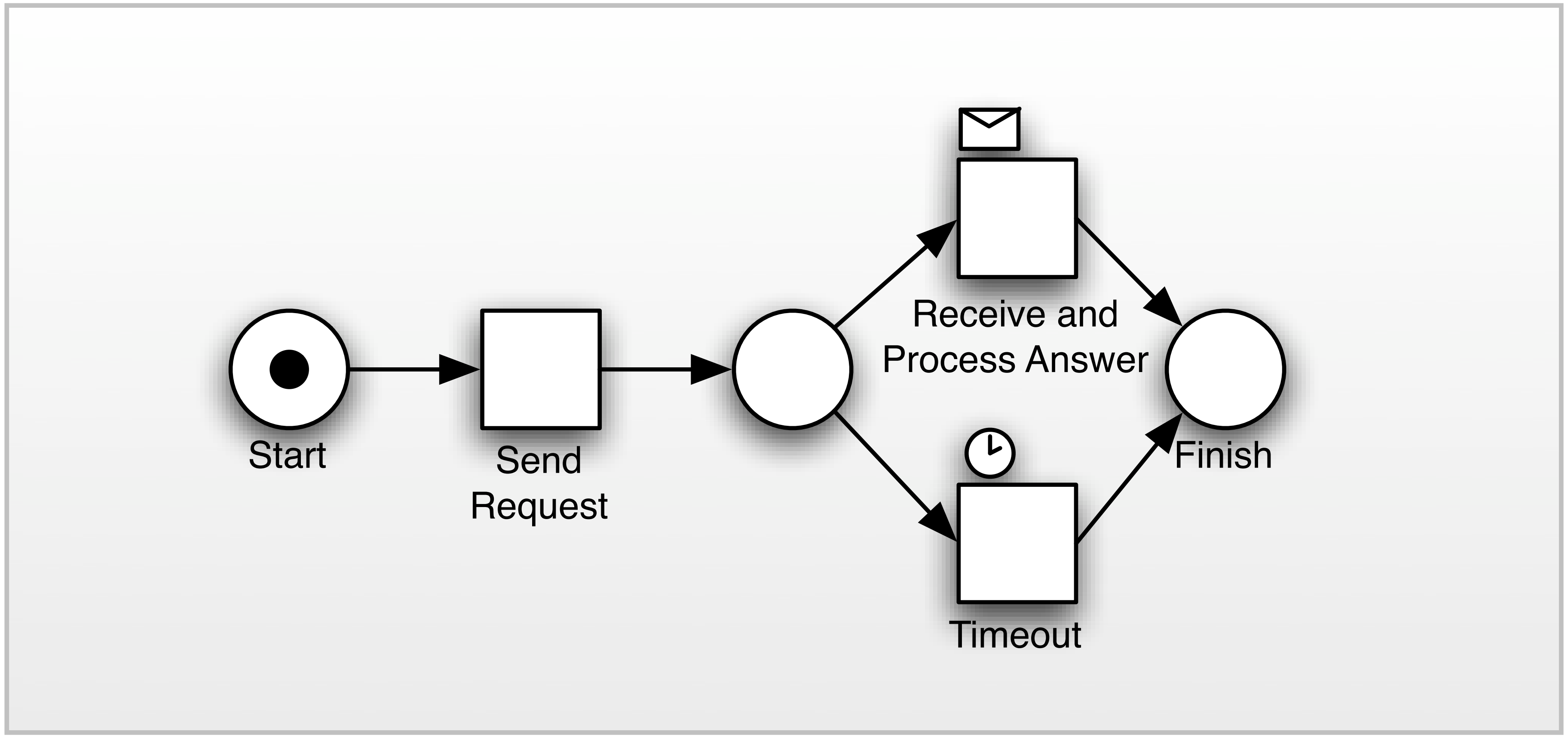
Why mobile systems?

- We argue: Three major shifts in BPM will lead to mobile systems as a theoretical foundation

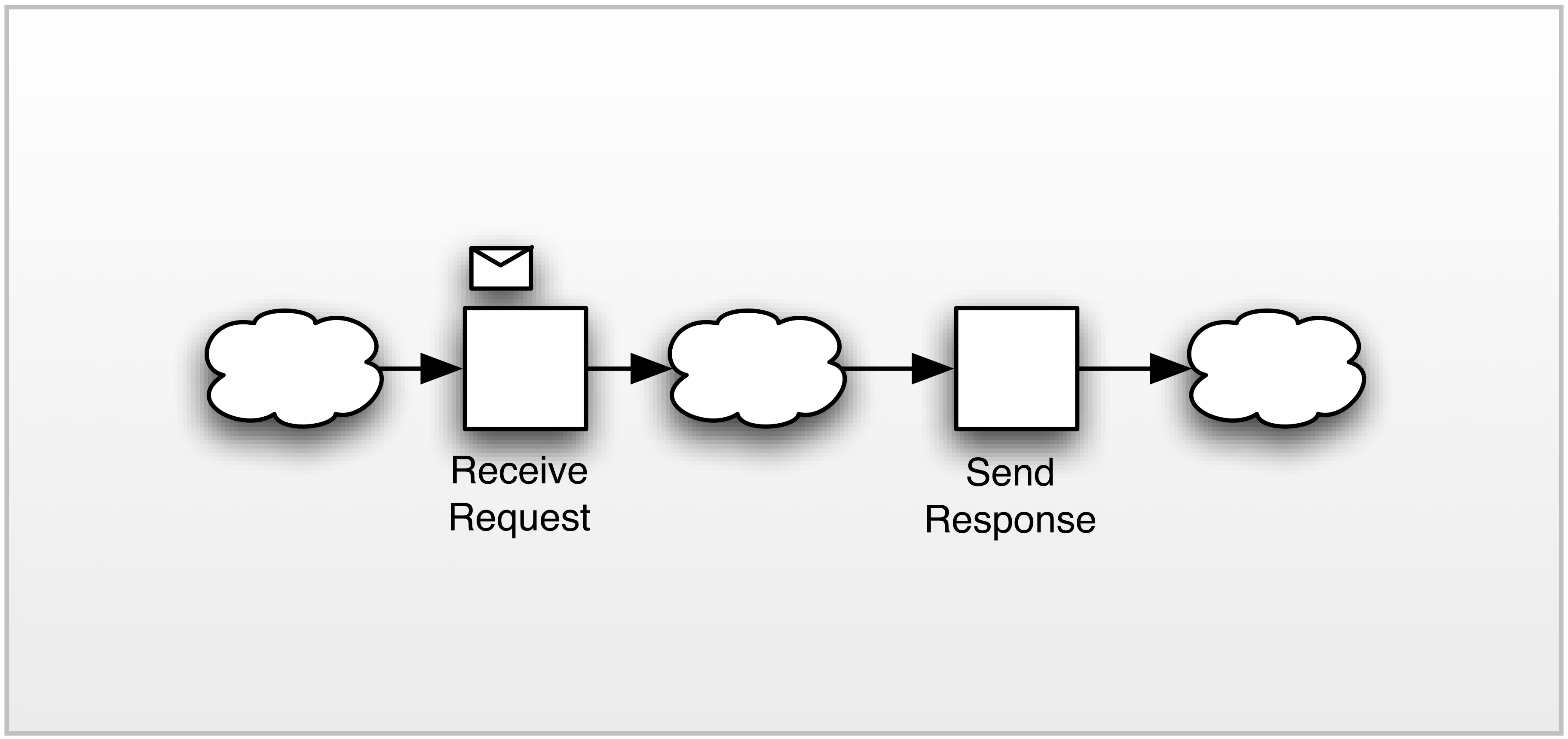
BPM Shift 1: From Static to Dynamic Systems

- Traditional: Static, state-based systems
  - e.g. Workflow nets, Activity Diagrams, BPMN (Token-Place concept)
- Today: Inter-organizational business processes
- “Hard to change”

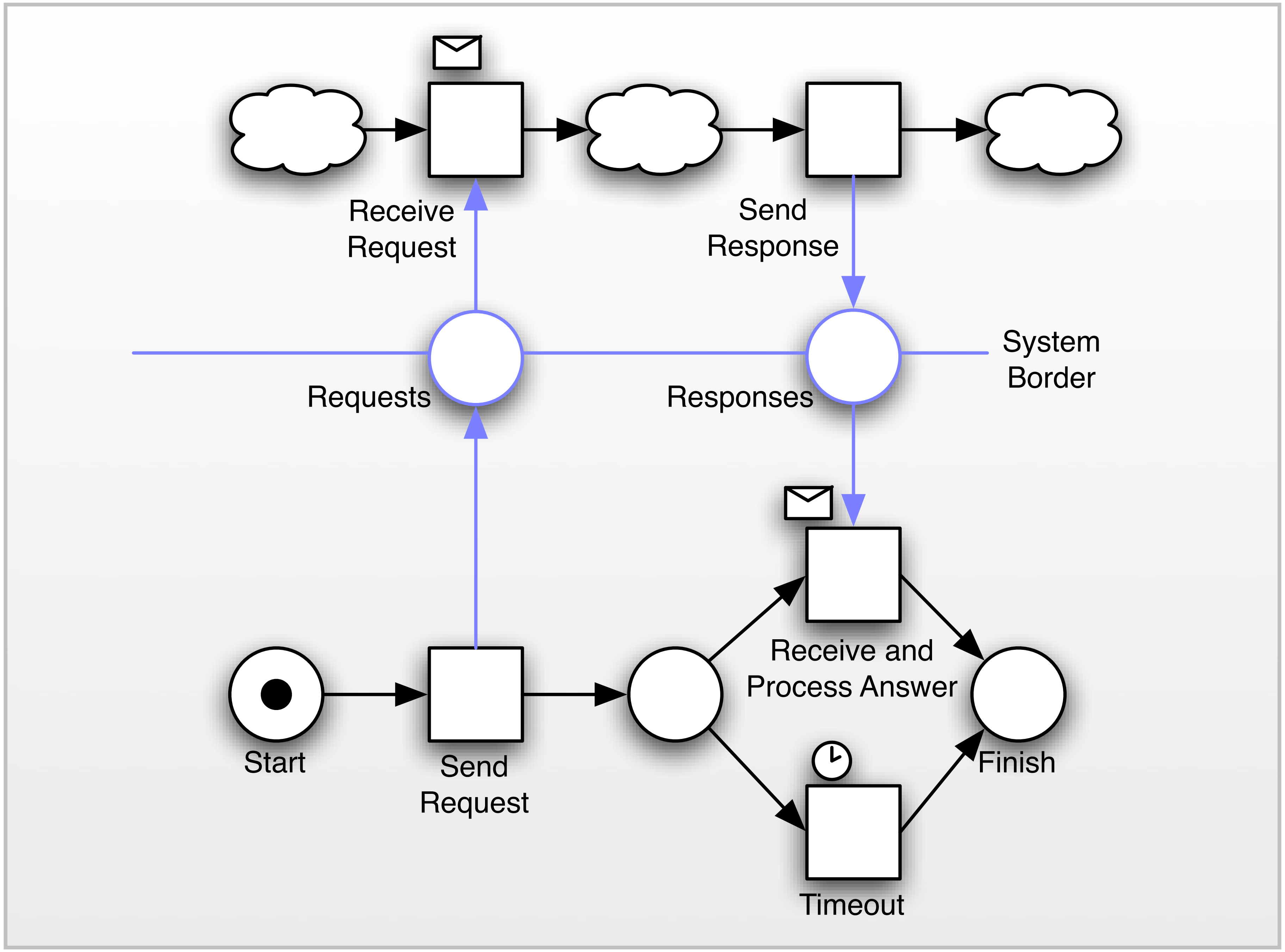
(C) 2007 Frank Puhlmann



Sample Process



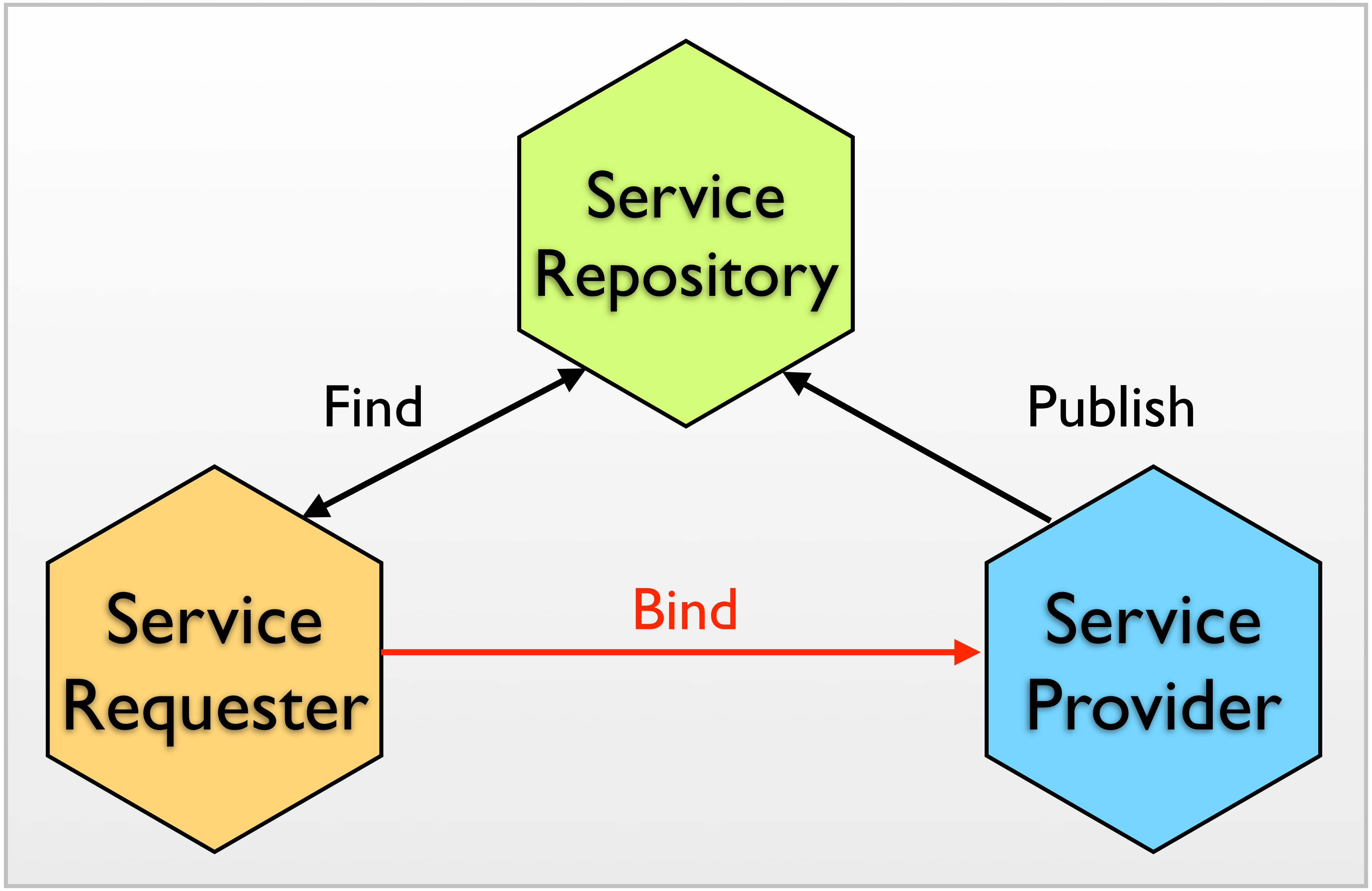
Corresponding Process



Static Interaction

Dynamic Systems

- No explicit state description
- Each task is mapped to a service:
  - Each task has pre- and postconditions (i.e. in- and outgoing messages)
  - All tasks are “swimming” inside a service-oriented environment



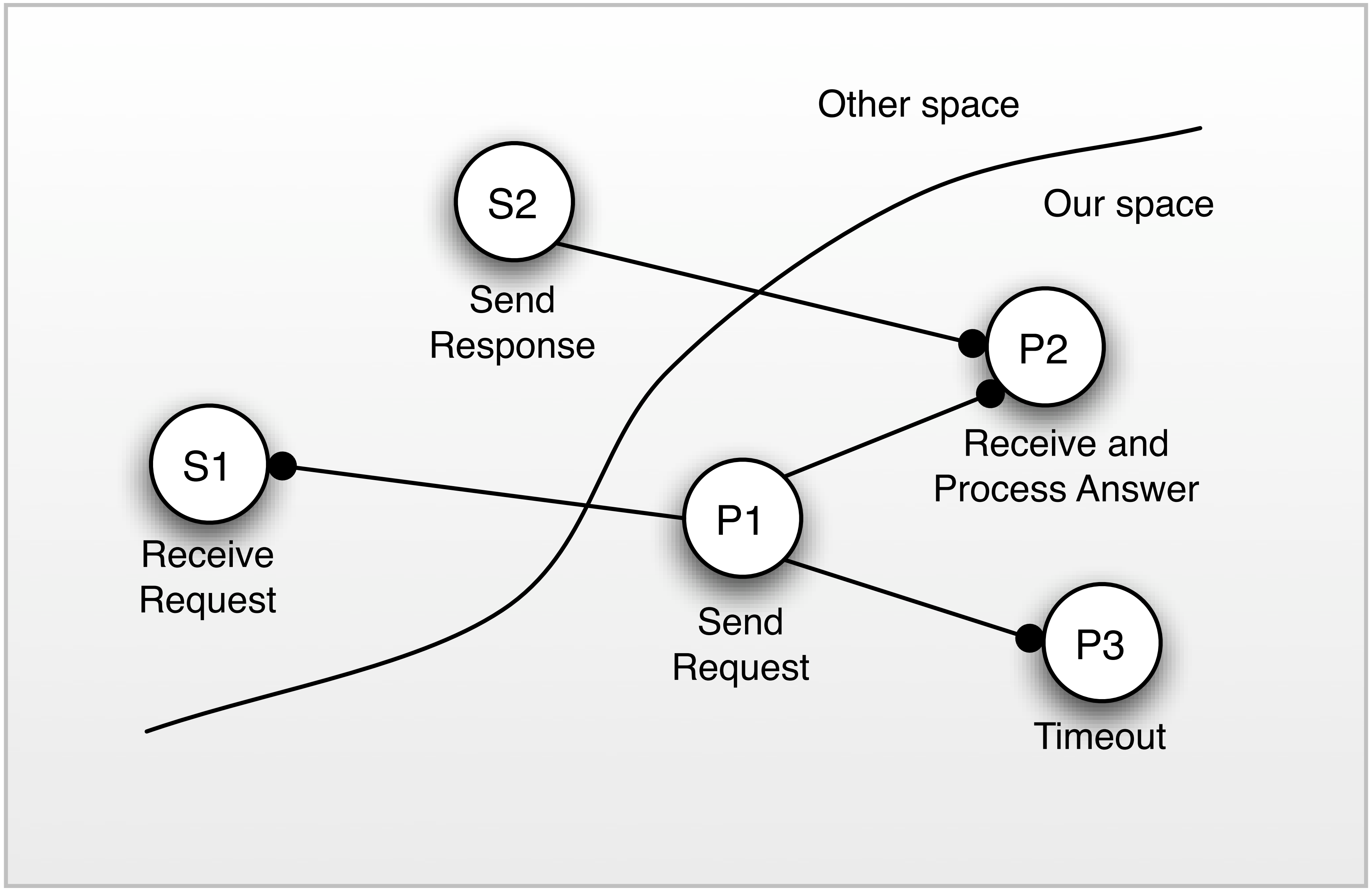
Service-oriented Architecture

Reason 1:

- Mobile systems are based on the idea of interaction by messages/events instead of state transitions
- Support for dynamic binding

BPM Shift II: From Central Engines to Distributed Services

- Follows direct from the last shift:
  - No more centralized engine as for intra-organizational “workflow”
  - Instead distributed services of different granularity



Distributed Services

Reason II:

- Mobile systems support advanced composition and visibility of their parts
- Support distribution and the service-oriented idea for BPM

BPM Shift III: From Closed to Open Environments

- The environment where processes are executed is shifting strongly from closed to open, which means:
  - Less accessibility
  - More uncertainty
  - Constant change regardless of us
  - Number of possible interaction partners rises fast

Issues regarding Open Environments

- Constant change requires dynamic adaption
- Flexible discovery and integration
- More agile interaction

Reasons III:

- Mobile systems describe dynamic process structures
- Based on a prototypical viewpoint
- Support “flexibility” regarding discovery and interaction for BPM

(C) 2007 Frank Puhlmann

Motivation in a Nutshell

- Mobile systems support advanced key concepts of BPM:
  - Dynamic Binding
  - Composition and Visibility
  - Change
- The Pi-Calculus is a process algebra for mobile systems

The Theory of the Pi-Calculus

Syntax & Semantics

Informal Introduction

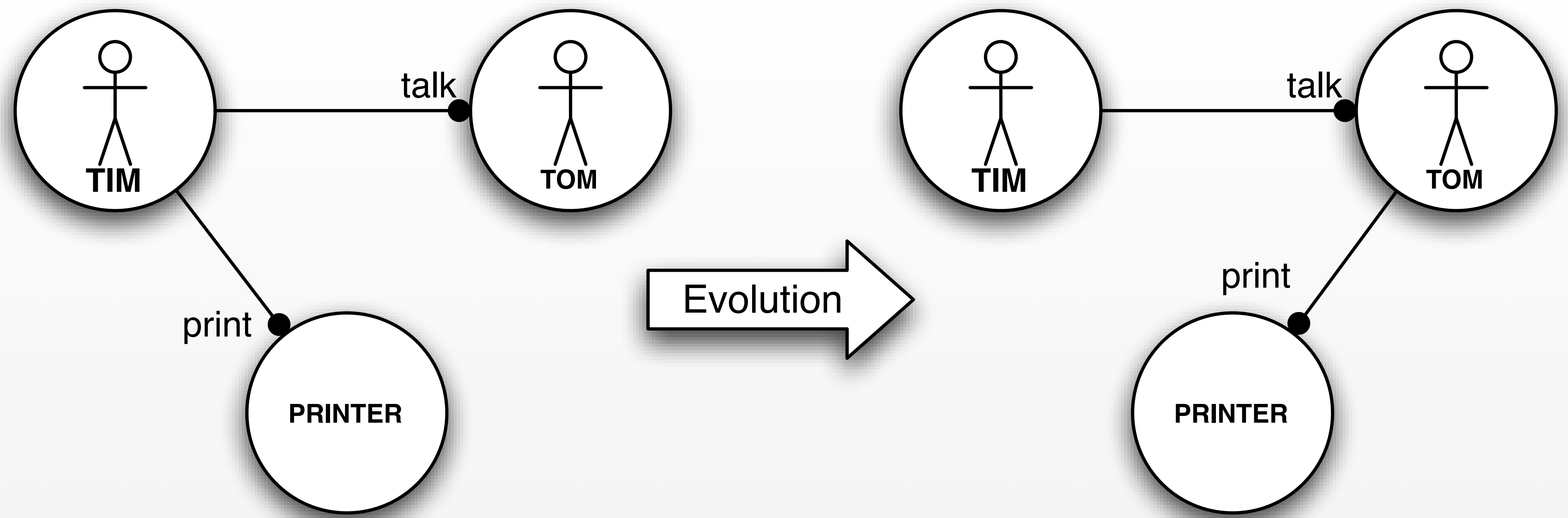
- The Pi-Calculus is based on few concepts:
  - Agents (Processes)
  - Names
  - Synchronized Interactions

$$TIM \stackrel{def}{=} \overline{talk}\langle message\rangle.\mathbf{0}$$

$$TOM \stackrel{def}{=} talk(message).\tau_{TOM}.\mathbf{0}$$

$$SYSTEM \stackrel{def}{=} TIM \mid TOM$$

Basic Interaction



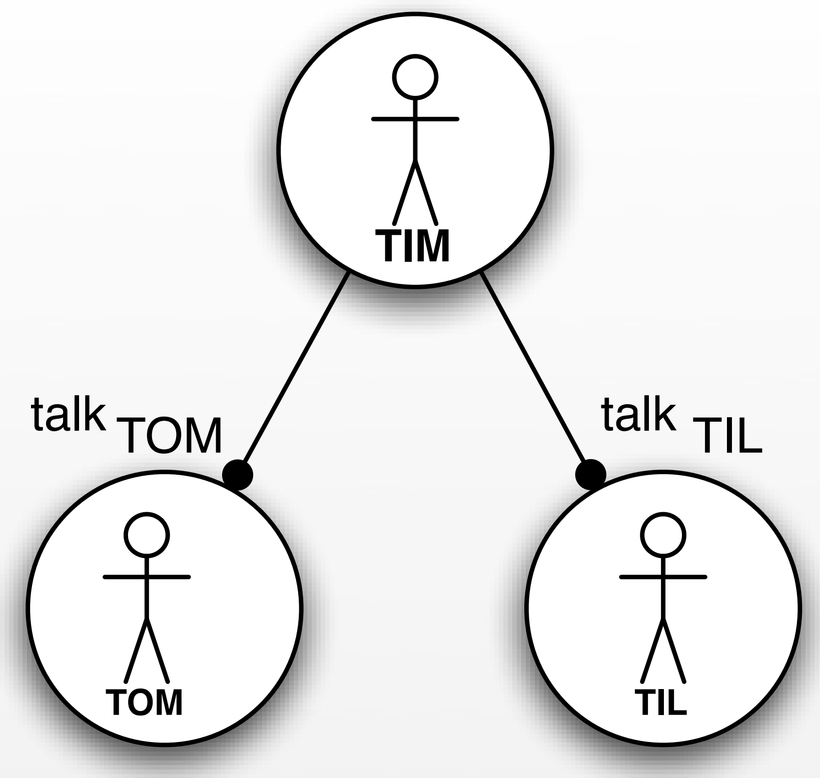
$$TIM \stackrel{def}{=} \overline{talk}\langle print \rangle.\mathbf{0}$$

$$TOM \stackrel{def}{=} talk(print).\overline{print}\langle file \rangle.\mathbf{0}$$

$$PRINTER \stackrel{def}{=} !print(file).\tau_{PRINT}.\mathbf{0}$$

$$SYSTEM \stackrel{def}{=} TIM \mid TOM \mid PRINTER$$

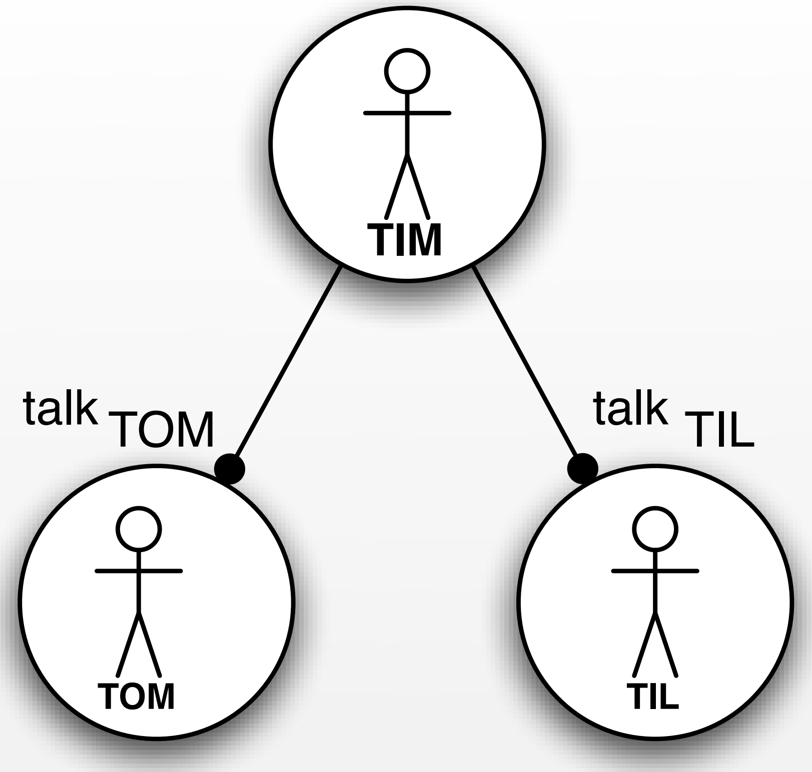
Advanced Interaction



$$TIM \stackrel{def}{=} \overline{talk_{TOM}}\langle message\rangle.\mathbf{0} + \overline{talk_{TIL}}\langle message\rangle.\mathbf{0}$$

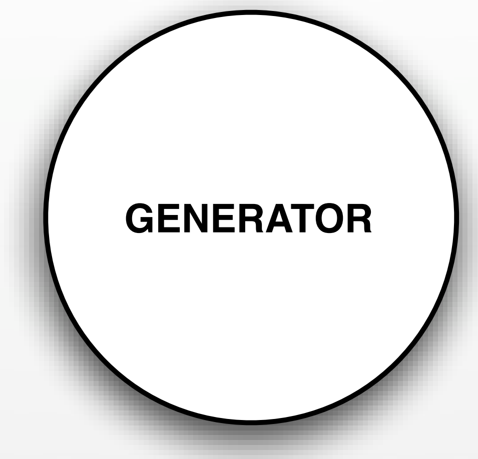
$$TIM \stackrel{def}{=} [x = \top]\overline{talk_{TOM}}\langle message\rangle.\mathbf{0} +$$
$$[x = \bot]\overline{talk_{TIL}}\langle message\rangle.\mathbf{0}$$

Choice



$$TIM \stackrel{def}{=} \overline{talk_{TOM}}\langle message\rangle.\mathbf{0} \mid \overline{talk_{TIL}}\langle message\rangle.\mathbf{0}$$

Concurrency



$$GENERATOR \stackrel{def}{=} (\nu x)\overline{get}\langle x \rangle.\mathbf{0}$$

Name Creation

The Pi-Calculus BNF

$$P ::= M \mid P|P \mid \nu z\ P \mid !P$$

$$M ::= \mathbf{0} \mid \pi.P \mid M + M$$

$$\pi \quad ::= \quad \overline{x}\langle y\rangle \mid x(z) \mid \tau \mid [x = y]\pi$$

Abbreviations

$$\text{Composition: } \prod_{1}^{3}(P) = P|P|P$$

$$\text{Summation: } \sum_{1}^{3}(P) = P + P + P$$

$$\text{with index: } \sum_{i=1}^{3}(d_i.\mathbf{0}) = d_1.\mathbf{0} + d_2.\mathbf{0} + d_3\mathbf{0}$$

$$\text{Sequence:} \{\pi\}_1^3 = \pi.\pi.\pi$$

Bound and free names

- In each of
  $x(z).P \text{ and } \nu z\ P$
  the displayed occurrence of *z* is *binding* with *scope P*
- An occurrence of a name in an agent is *bound* if it is, or it lies within the scope of, a binding occurrence of the name
- An occurrence of a name in an agent is *free* if it is not bound

Substitution

- We write

$$P\{^{y_1}/_{x_1}, \cdots, ^{y_n}/_{x_n}\}$$

- for the simultaneous substitution of y_i for all free occurrences of x_i in P, with the change of bound names if necessary to prevent any of the new names y_i from becoming bound in P

Defined Agent Identifiers

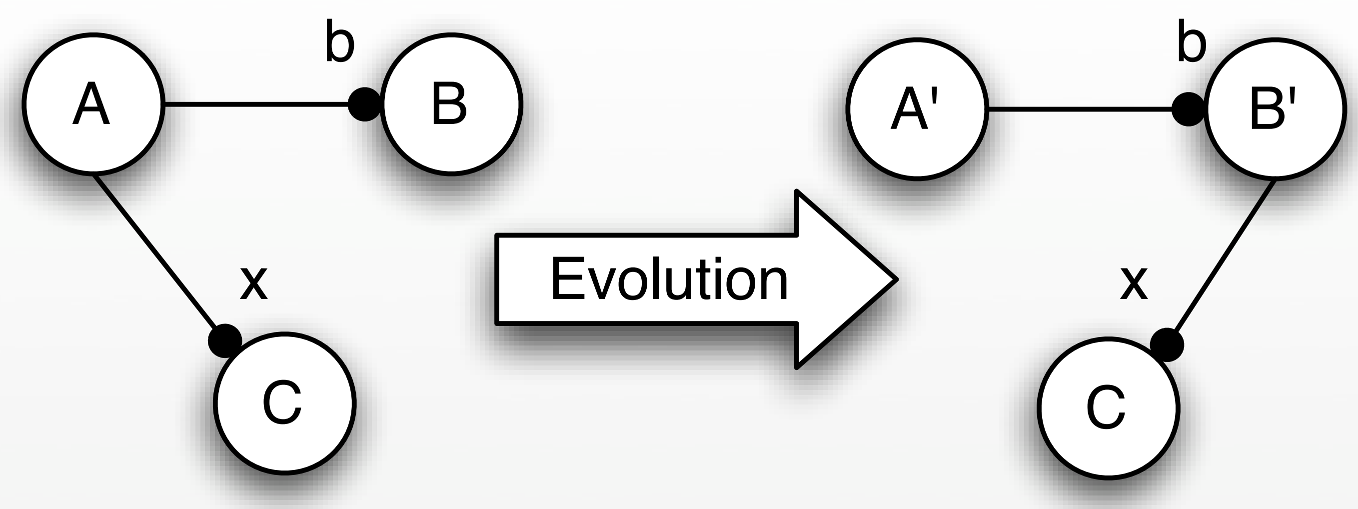
- A defined agent identifier is given by:

$$A(x_1, \cdots, x_n) \overset{def}{=} P$$

- Then

$$A(y_1, \cdots, y_n) \text{ behaves as } P\{^{y_1}/_{x_1}, \cdots, ^{y_n}/_{x_n}\}$$

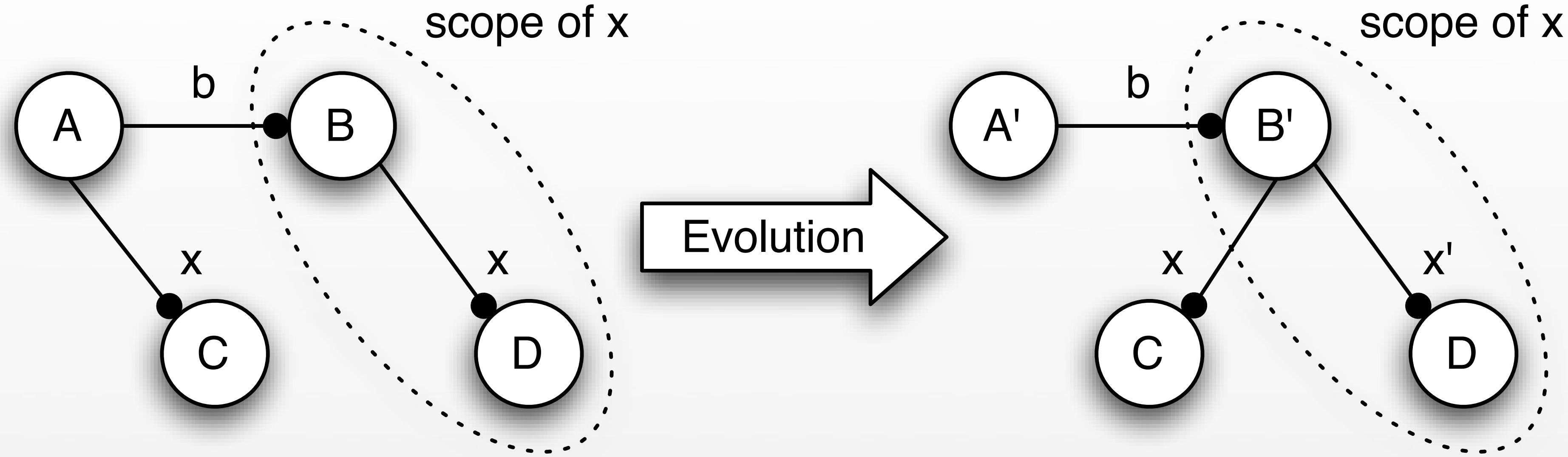
- if x_i are free names in P
- the definition can be thought of as an agent declaration with $x_1, ..., x_n$ as formal parameters, and the identifier $A(y_1, ..., y_n)$ as an invocation with actual parameters $y_1,..., y_n$



$$A \mid B \equiv \overline{b}\langle x \rangle.A' \mid b(y).B'$$

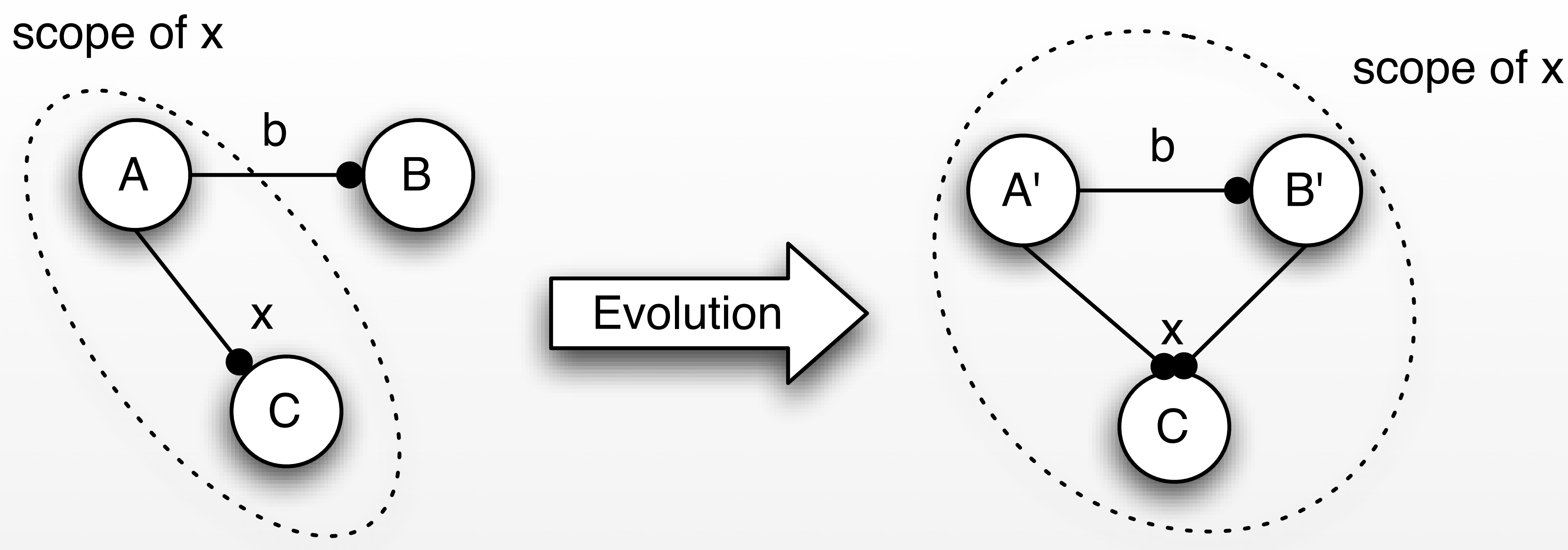
$$\overline{b}\langle x \rangle.A' \mid b(y).B' \mid C \longrightarrow A' \mid B'\{^x/_y\} \mid C$$

Example: Communication



$$\bar{b}\langle x \rangle.A' \mid C \mid (\nu x)(b(z).B' \mid D)$$
$$\longrightarrow$$
$$A' \mid C \mid (\nu x')(B'\{^{x'}/_{x}\}\{^{x}/_{z}\} \mid D\{^{x'}/_{x}\})$$

Example: Scope Intrusion



$$(\nu x)(\overline{b}\langle x\rangle.A' \mid C) \mid b(z).B'$$

$$\longrightarrow$$

$$(\nu x)(A' \mid C \mid B'\{^x/_z\})$$

Example: Scope Extrusion

$$\frac{A \stackrel{def}{=} \overline{b}\langle x \rangle.A + A'}{M(x) \stackrel{def}{=} write(x).M(x) + \overline{read}\langle x \rangle.M(x)}$$

$$(\nu write, read)(M(z) \mid A)$$

Example: Recursion

The Polyadic Pi-Calculus

- How can we send messages consisting of multiple names?

The Polyadic Pi-Calculus

- Syntactical enhancement:
  - $\overline{x}\langle y_1, \ldots, y_n \rangle.P \longmapsto (\nu w)(\overline{x}\langle w \rangle.\overline{w}\langle y_1 \rangle.\ldots.\overline{w}\langle y_n \rangle.P)$
  - $x(z_1, \ldots, z_n).P \longmapsto x(w).w(z_1).\ldots.w(z_n).P$
- Sequences:
  - $x_1, \ldots, x_n \longmapsto \tilde{x}$
- Empty messages:
  - $\overline{x}\langle \widetilde{y} \rangle \longmapsto \overline{x}$ iff $\widetilde{y} = \emptyset, x(\widetilde{z}) \longmapsto x$ iff $\widetilde{z} = \emptyset$

Reduction

- Evolution is formally defined as reduction
- The essence of reduction is captured in two axioms:
  - $(\overline{x}\langle y\rangle.P_1 + M_1) \mid (x(z).P_2 + M_2) \longrightarrow P_1 \mid P_2\{^y/_z\}$
  - $\tau.P + M \longrightarrow P$
- and three rules:
  - $\text{from } P_1 \longrightarrow P_1' \text{ infer } P_1|P_2 \longrightarrow P_1'|P_2$
  - $\text{from } P \longrightarrow P' \text{ infer } \nu z\ P \longrightarrow \nu z\ P'$
  - $\text{from } P \longrightarrow P' \text{ and } P \equiv Q \text{ and } P' \equiv Q' \text{ infer } Q \longrightarrow Q'$

Structural Congruence

- The axioms of structural congruence (Part 1):
  - SC-MAT: $[x = x]\pi.P \equiv \pi.P$
  - SC-SUM-ASSOC: $M_1 + (M_2 + M_3) \equiv (M_1 + M_2) + M_3$
  - SC-SUM-COMM: $M_1 + M_2 \equiv M_2 + M_1$
  - SC-SUM-INACT: $M + \mathbf{0} \equiv M$
  - SC-COMP-ASSOC: $P_1|(P_2|P_3) \equiv (P_1|P_2)|P_3$
  - SC-COMP-COMM: $P_1|P_2 \equiv P_2|P_1$
  - SC-COMP-INACT: $P|\mathbf{0} \equiv P$

Structural Congruence

- The axioms of structural congruence (Part 2):
  - SC-RES: $\nu z \nu w\ P \equiv \nu w \nu z\ P$
  - SC-RES-INACT: $\nu z\ \mathbf{0} \equiv \mathbf{0}$
  - SC-RES-COMP: $\nu z\ (P_1 | P_2) \equiv P_1 | \nu z\ P_2,\ \text{if } z \notin fn(P_1)$
  - SC-REP $!P \ \equiv\ P|!P$
  - UNFOLDING: $A(\tilde{y}) \equiv P\{^{\tilde{y}}/_{\tilde{x}}\} \text{ if } A(\tilde{x}) \stackrel{def}{=} P$

$$A \stackrel{def}{=} (\nu z)a(x, y).\overline{x}\langle z\rangle.\overline{y}.\mathbf{0}$$

$$B \stackrel{def}{=} \overline{a}\langle c, b\rangle.b.\mathbf{0}$$

$$C \stackrel{def}{=} c(m).\mathbf{0}$$

$$P \stackrel{def}{=} A \mid B \mid C$$

Example: Reduction