

# Object Monitor

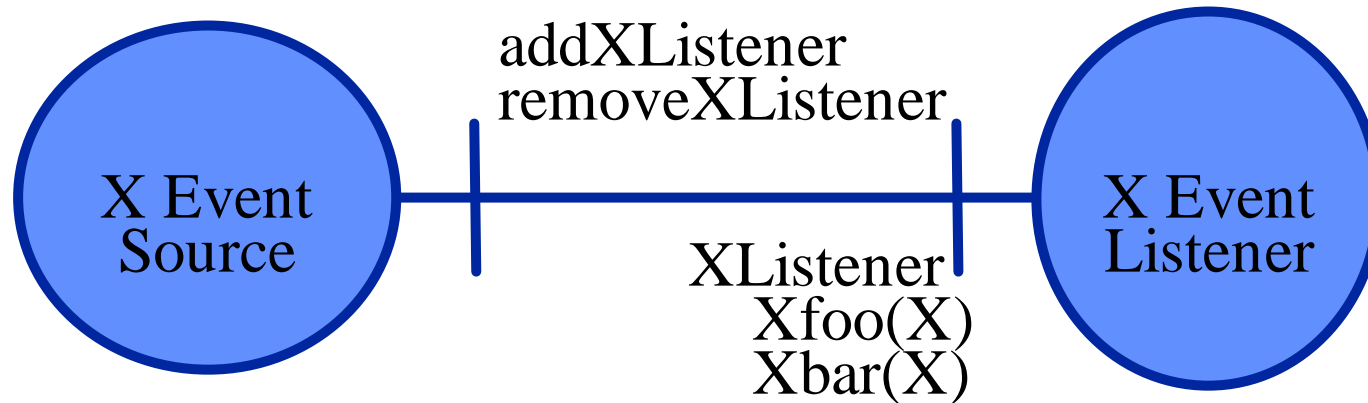
*An Open Extensible Flexible Framework  
For Building  
Distributed Event Monitoring Systems*

David Franklin

APM Ltd



# *An Extension of the Java Beans Event Model*



- Typed Events → Generic Events
- Decouple Sources and Sinks
  - View
  - Location
- Primitive Events → Composite Events

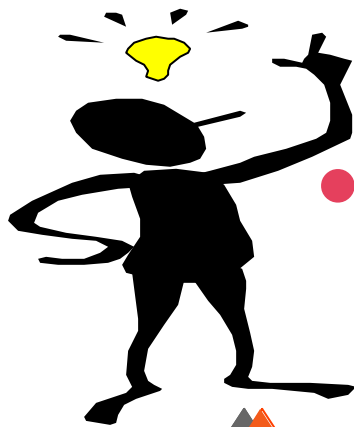


# *Events*



- Advantages ...
  - Loose coupling
  - Simple programming model

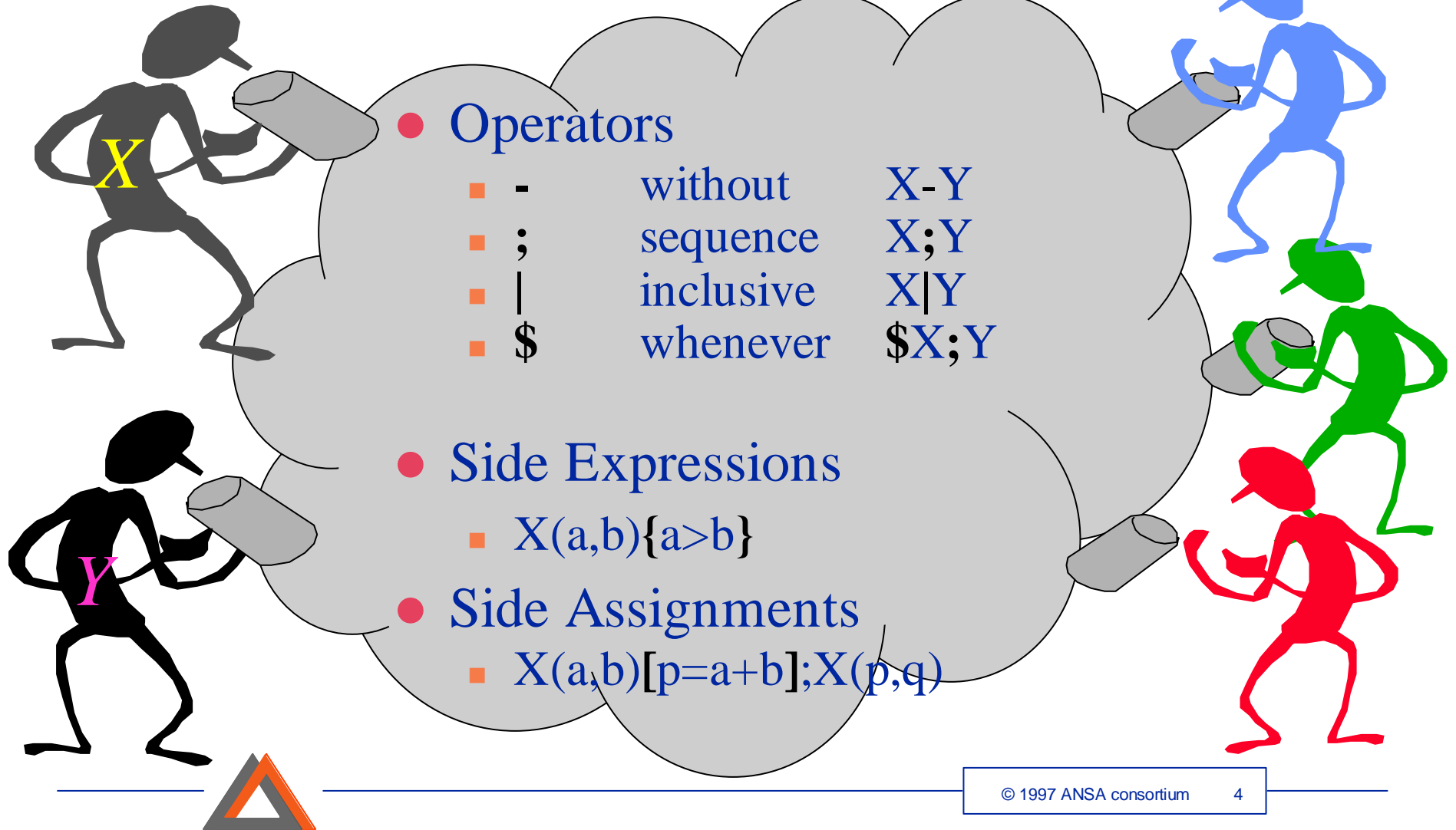
- But ...
  - Low level of abstraction
  - Understanding implications of low-level events



- So ...
  - Composite event patterns



# Composite Event Language



- Operators

- - without  $X-Y$
- ; sequence  $X;Y$
- | inclusive  $X|Y$
- \$ whenever  $\$X;Y$

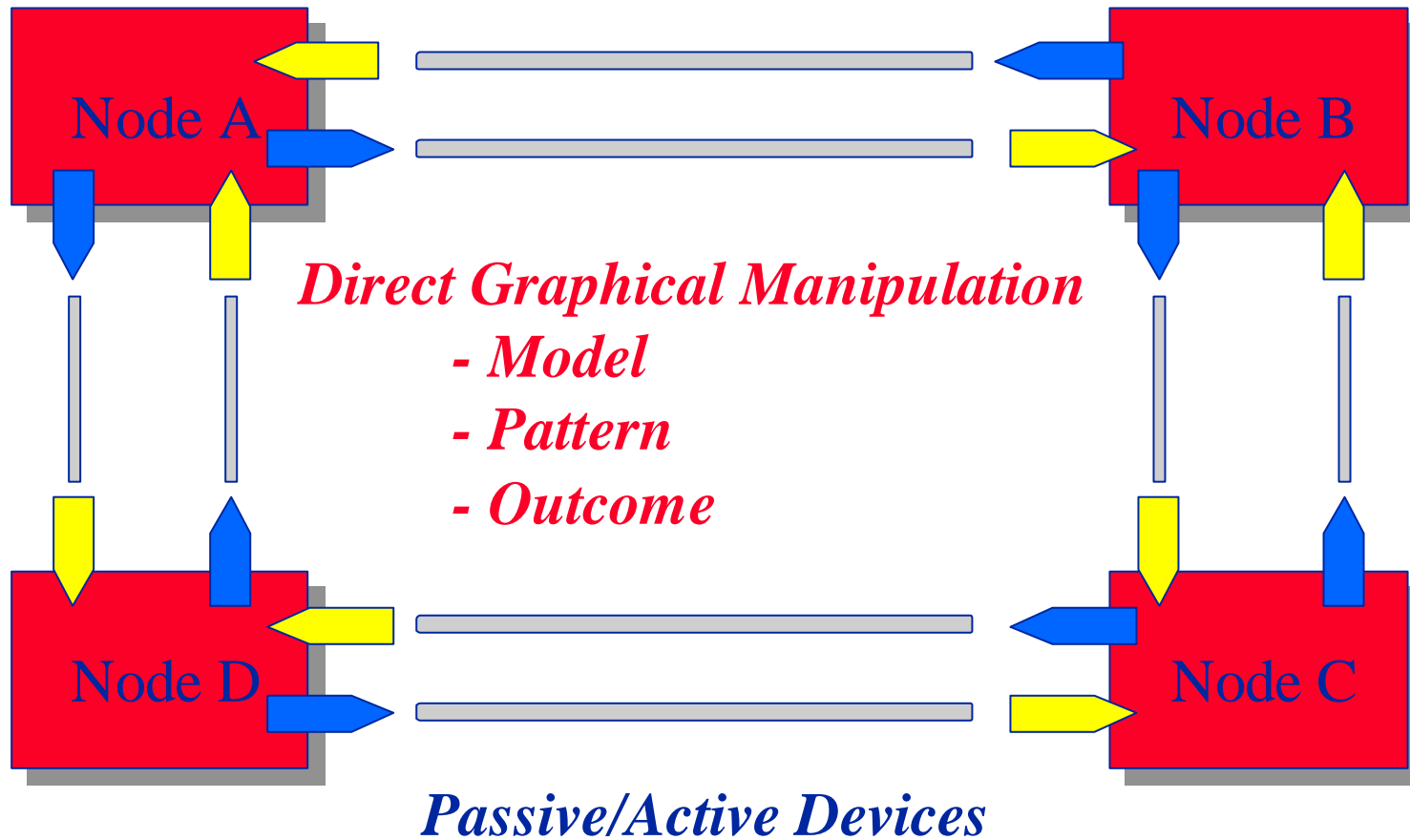
- Side Expressions

- $X(a,b)\{a>b\}$

- Side Assignments

- $X(a,b)[p=a+b];X(p,q)$

# *Network Management Demonstrator*



# *Event Patterns*

- Node Failure

- $N(\text{device}) = \$F(\text{device}, \text{code})\{\text{code} == 0\}$

- Security Violation

- $S() = \$F(\text{device}, \text{code})\{\text{device} == 1 \ \&\& \ \text{code} == 1\}$

- Broken Cable

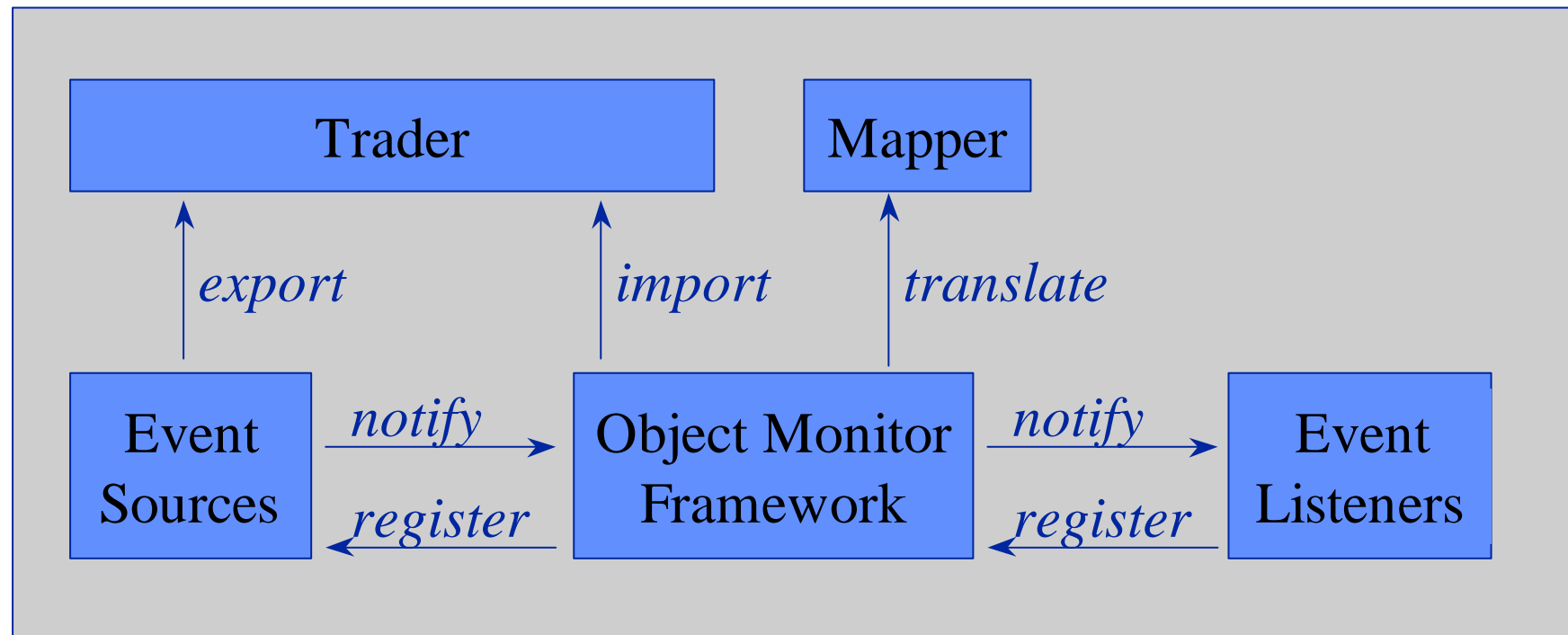
- $B(\text{devA}, \text{devB}) = \$ (F(\text{devA}, \text{code})\{\text{code} == 2\}; F(\text{devB}, \text{code})\{!(\text{devA} == \text{devB})\})$

- Loose Connector

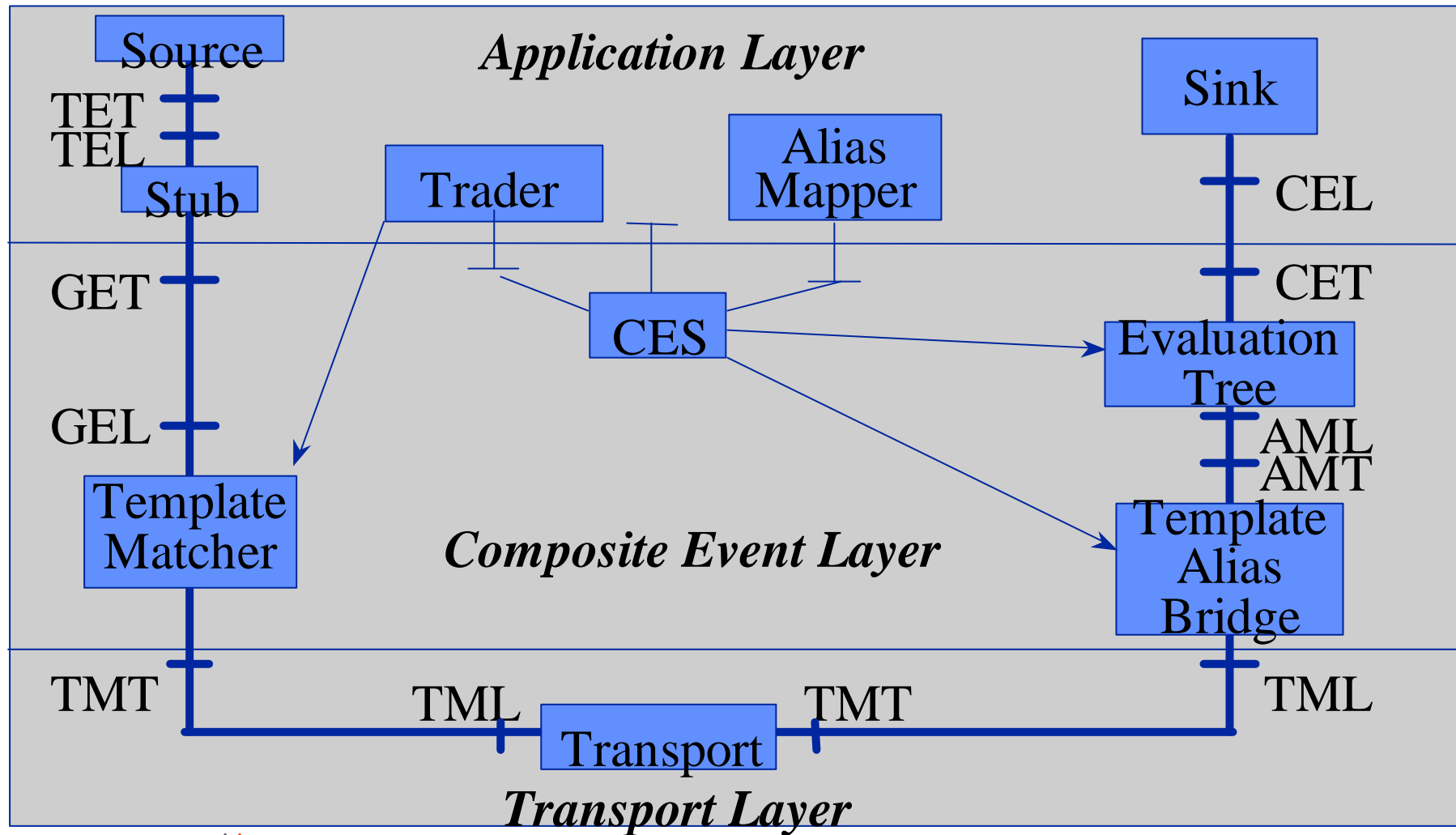
- $L(d) = \$F(d)[t = @]; F(d); F(d); F(d); F(d); F(d)\{ @ < t + 10\}$



# *Event Sources , Event Listeners - the Object Monitor Framework*

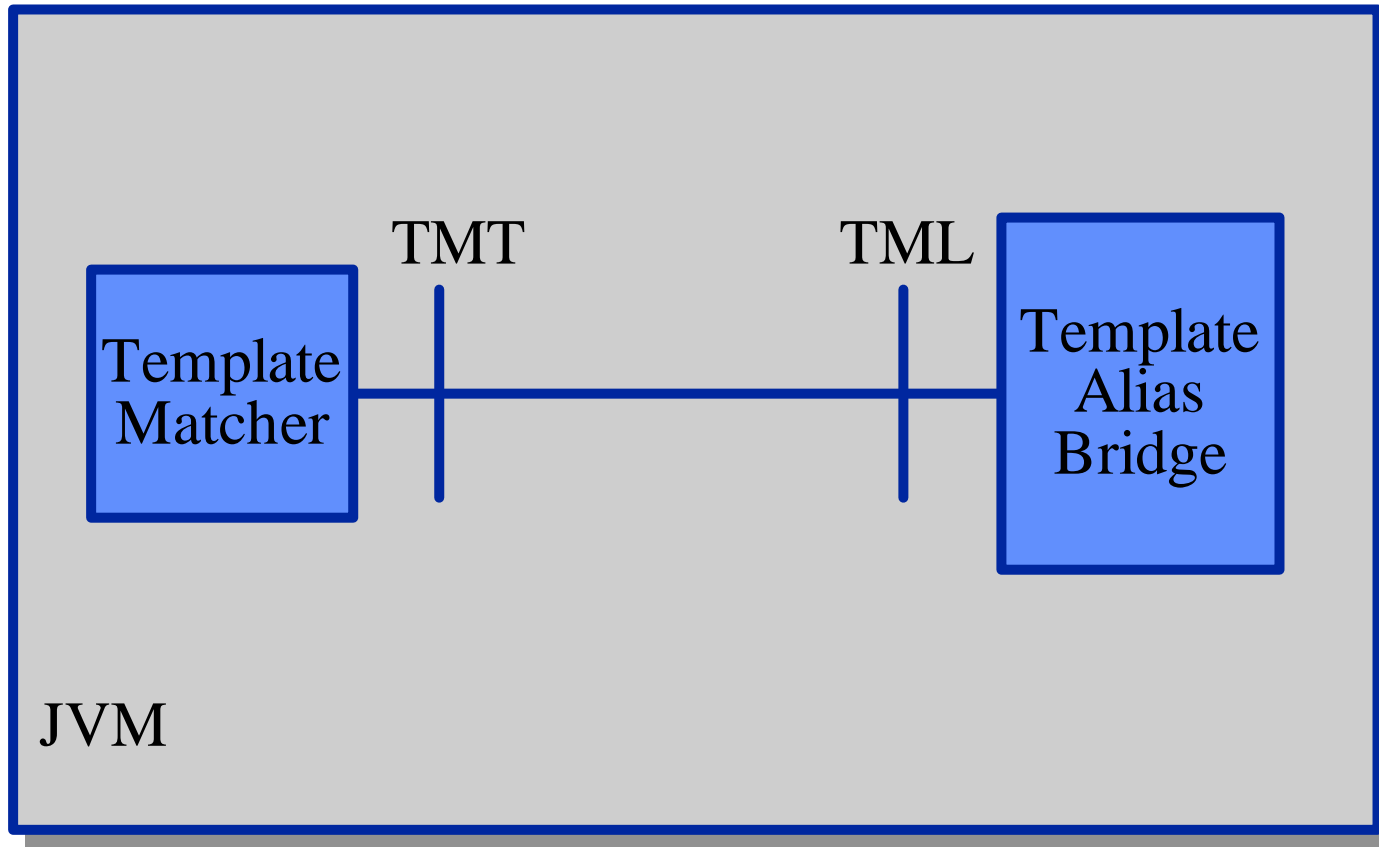


# Layers of Bridges

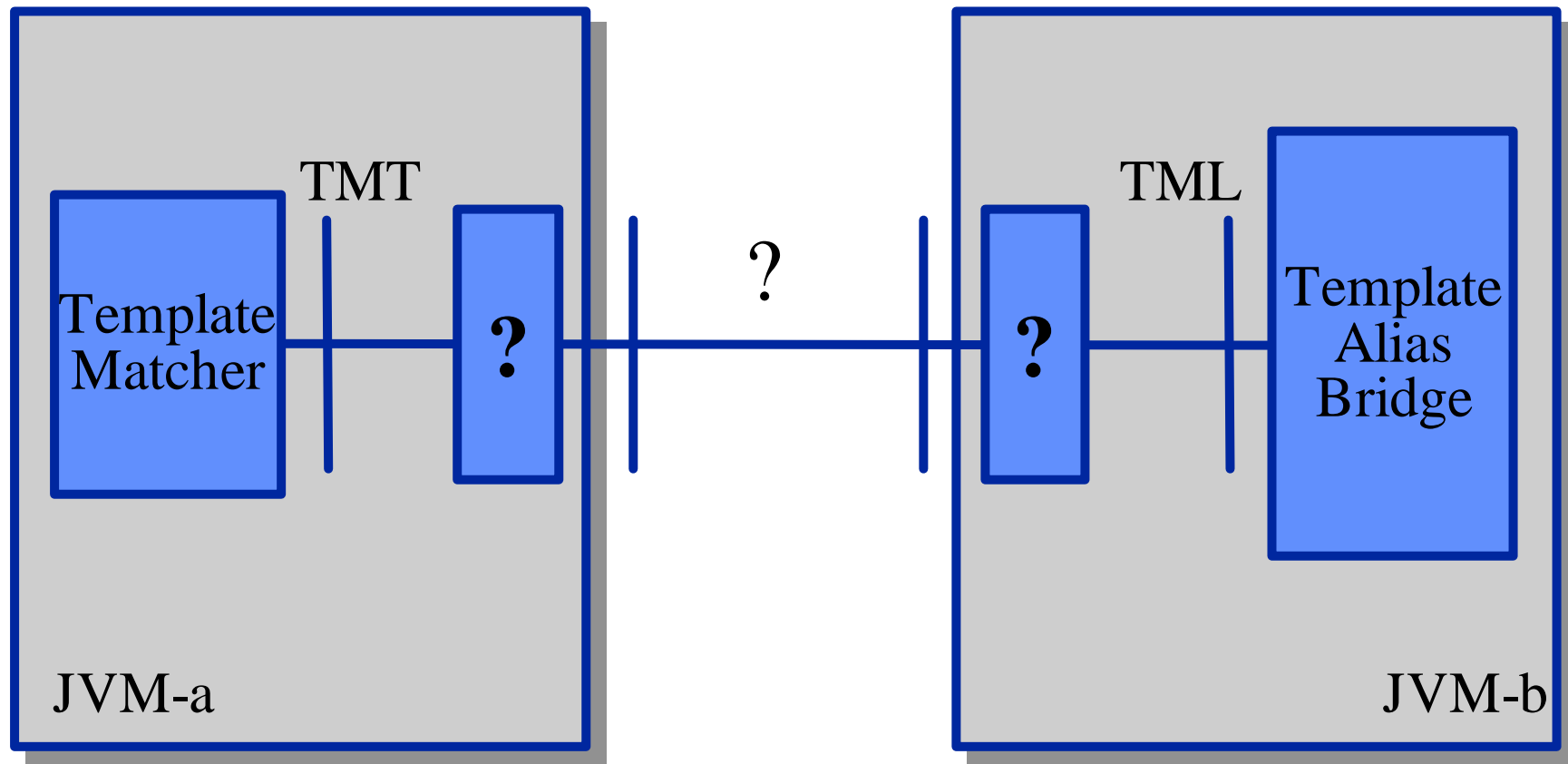




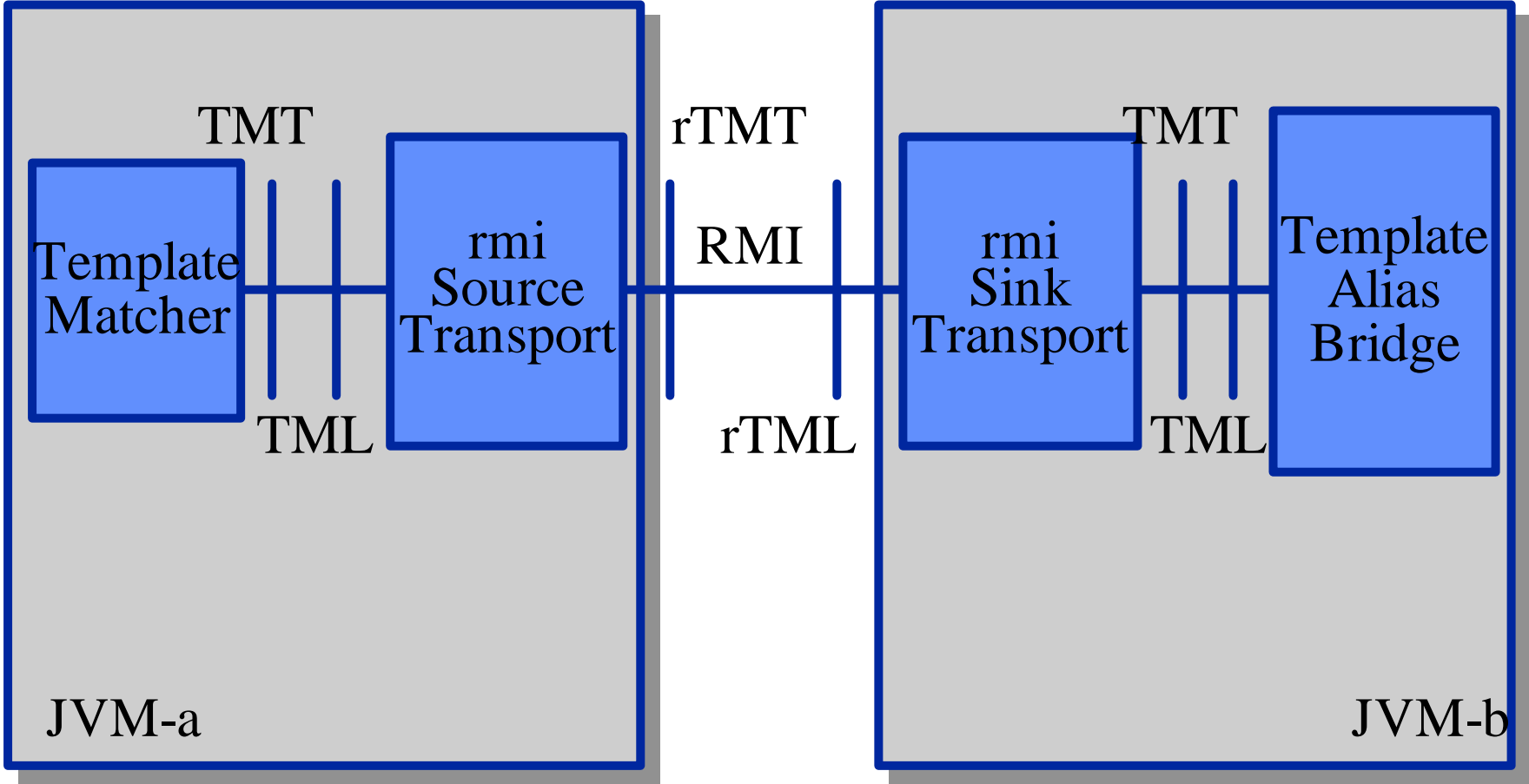
# *Non-Distributed Transport*



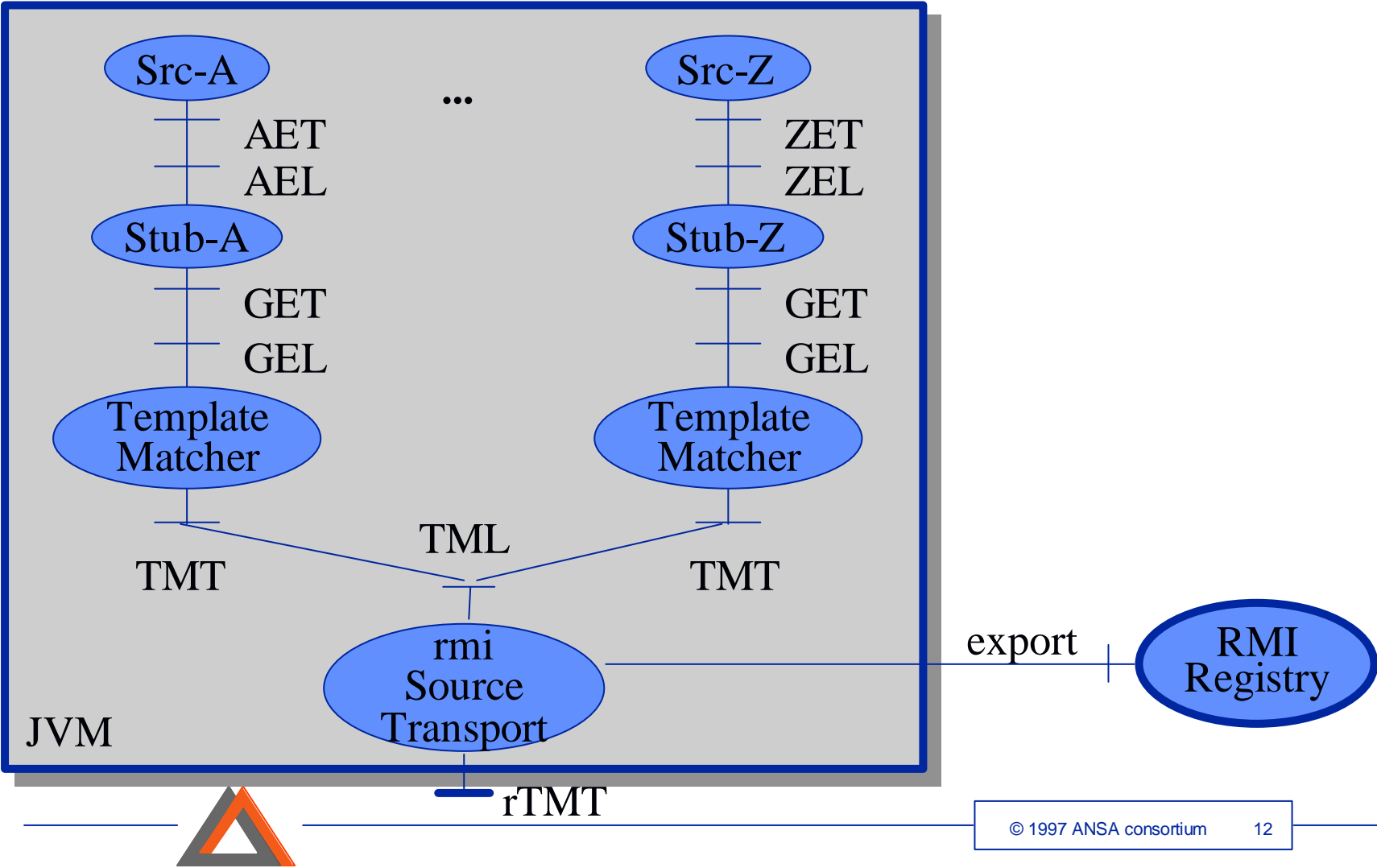
# *Distributed Transport*



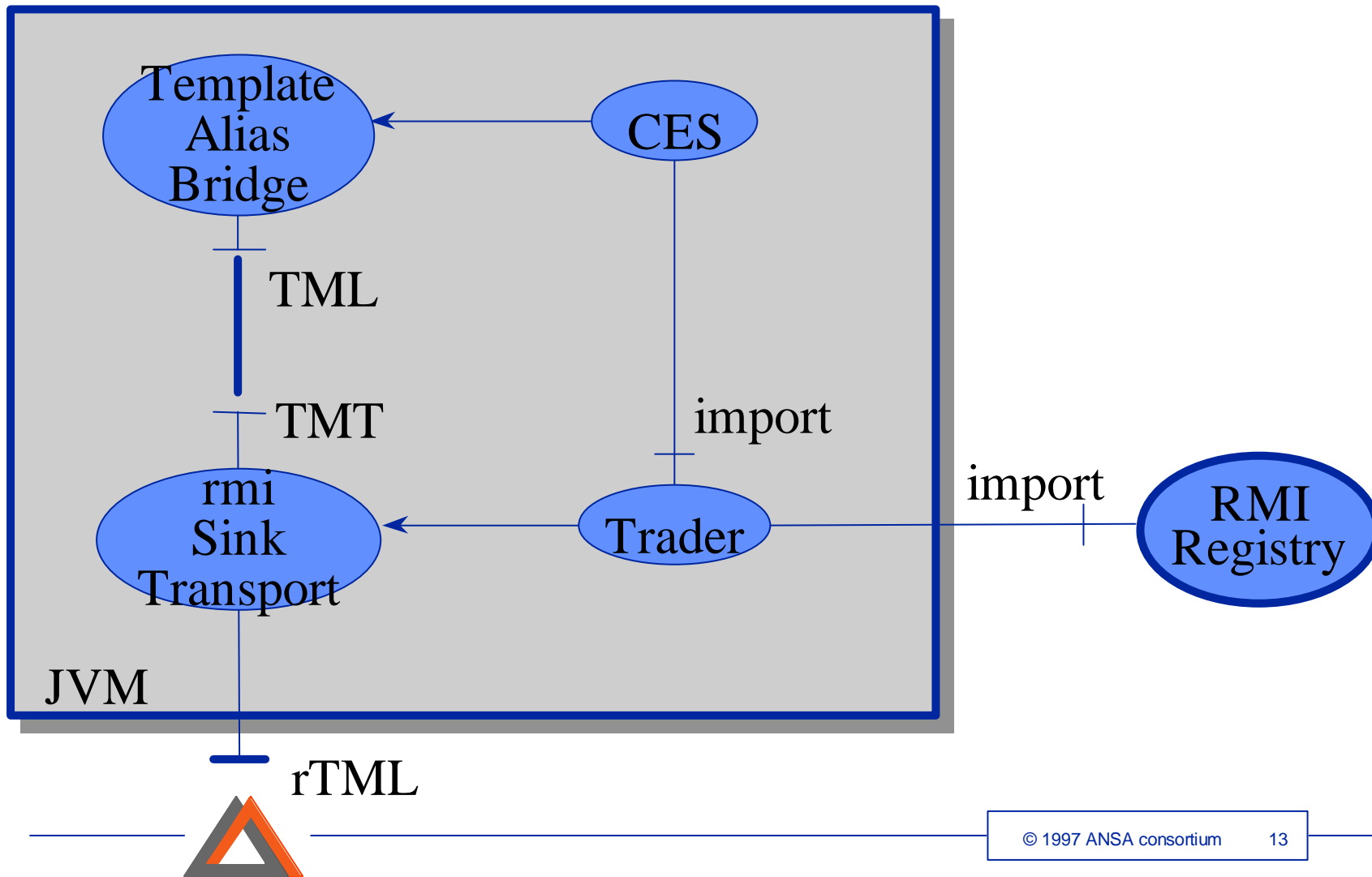
# Remote Proxy Classes



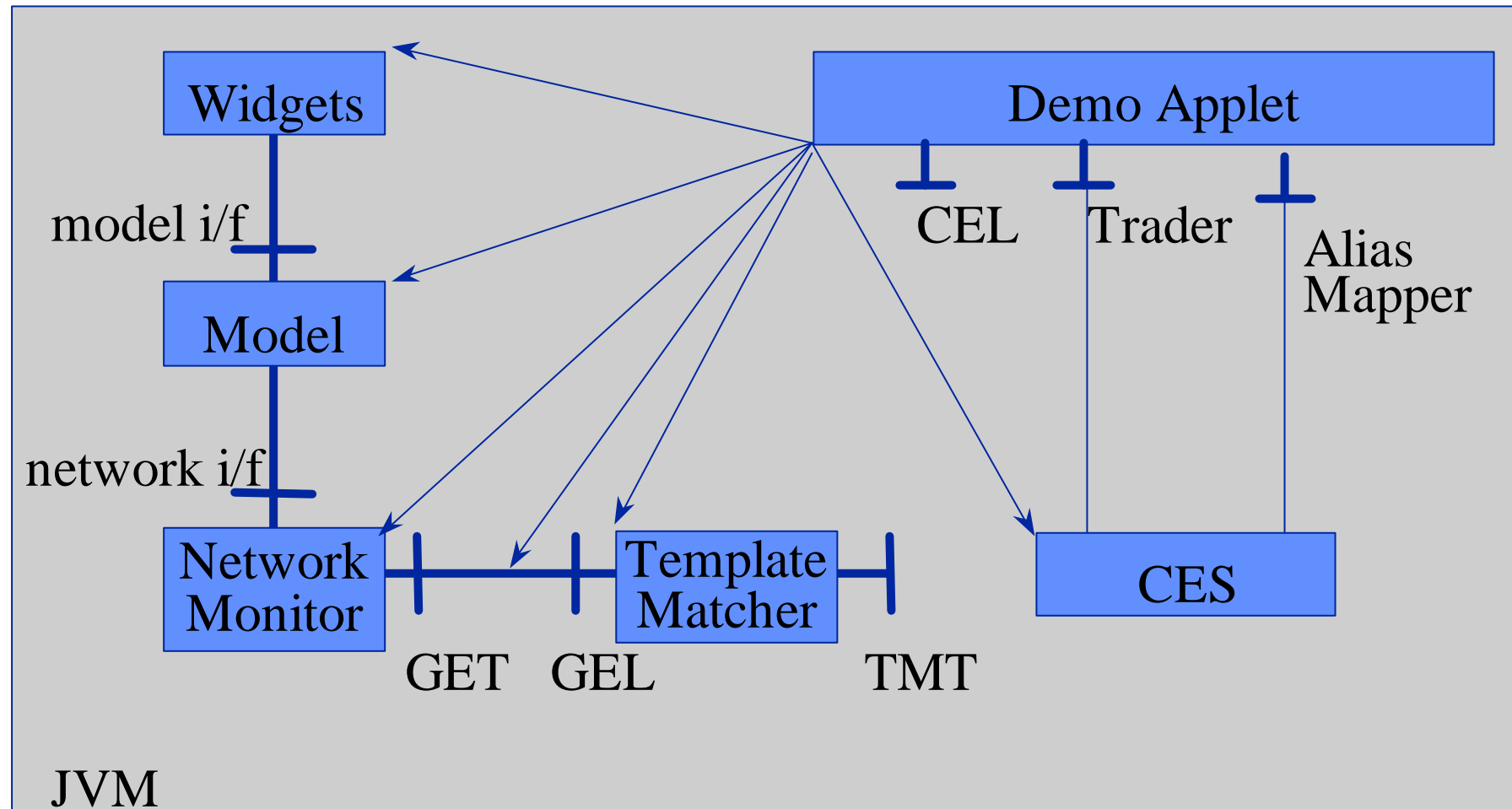
# Source-side Transport



# *Sink-side Transport*



# *Demonstrator - Single JVM*



# Demonstrator - Multiple JVMs

