Computational Paradigms and Protection

Simon N. Foley and John P. Morrison
Department of Computer Science,
University College,
Cork, Ireland.

NSPW, Cloudcroft, NM, USA, September 2001

Motivation

Distributed Computation

Mechanisms provide control over sequencing, scheduling and synchronization of tasks distributed across processors.

Paradigms

Imperative: explicit control over sequencing of tasks.

Availability, Coercive: implicit control over sequencing of tasks.

Protection

Mechanisms provide control over sequencing, scheduling (authorization) and synchronization of tasks across protection domains.

Paradigms

Imperative: mediation based on explicit sequencing of tasks.

Availability, Coercive: ???

Casting protection in the availability/coercive styles provide a basis for more flexible and distributed control over sequencing and mediation of operations.

Overview

1. Computational paradigms

2. Computational paradigms and protection

3. Condensed Graph based protection mechanisms:
   Fragile, Tenacious and Emergent Protection

4. Practical Implementation: Secure WebCom

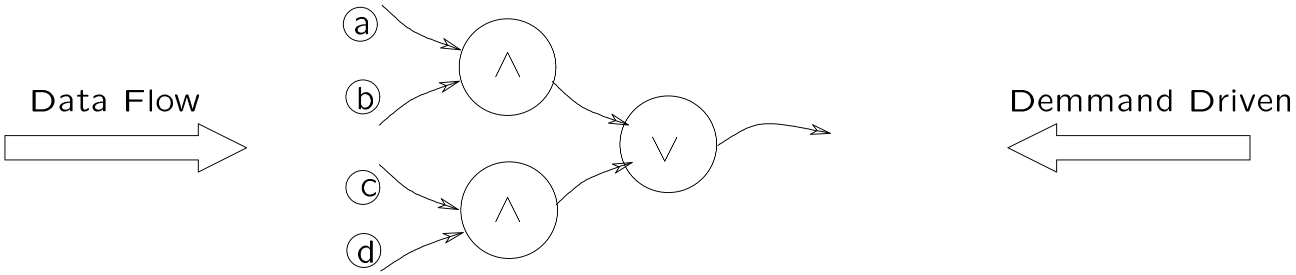
Computational Paradigms

Example: evaluate $(a \wedge b) \vee (c \wedge d)$:

Imperative: programmer explicitly specifies operation sequencing constraints.

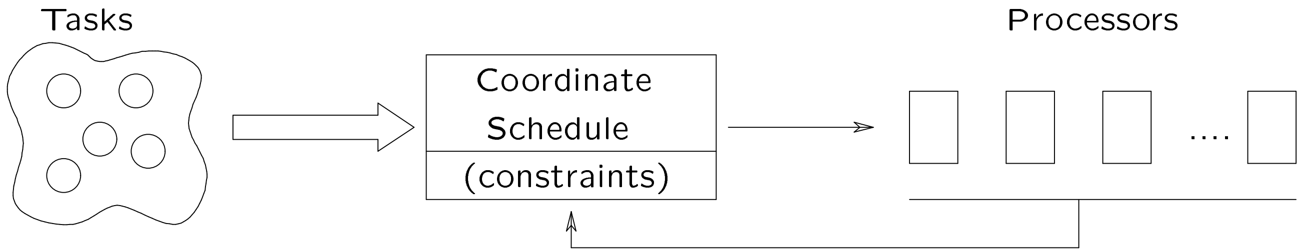
$\begin{array}{rcl} x & \leftarrow & a \wedge b \\ ;y & \leftarrow & c \wedge d \\ ;z & \leftarrow & x \vee y \end{array}$	$\begin{array}{rcl} z & \leftarrow & \text{if } (a \wedge b) \\ & & \text{then } \mathit{true} \\ & & \text{else } (c \wedge d) \end{array}$	$\begin{array}{rcl} (x & \leftarrow & a \wedge b \\ \Vert y & \leftarrow & c \wedge d) \\ ;z & \leftarrow & x \vee y \end{array}$

Availability: operation sequencing depends only on operand data availability.



Coercion: operations executed when and only when their results are needed.

Computational Models

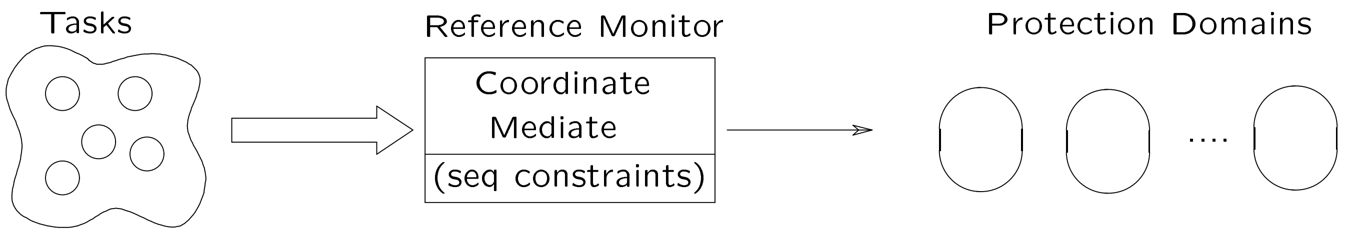


Imperative: coordination constraints explicit in tasks to be scheduled. Example: traditional von-Neumann architectures.

Availability and Coercion: coordination constraints implicit in data dependencies. Example: data flow architectures (availability); reduction architectures (coercion).

Condensed Graphs: unifies the three paradigms within a single model.

Protection Models



Conventional protection mechanisms may be viewed as driven by an imperative paradigm, with mediation based on explicit sequencing of operations.

Example: Chinese Wall Policy. Record a principle's access history in a protection state. The mechanism enforces the policy by explicitly constraining the subsequent sequencing of tasks (access operations) based on the current protection state.

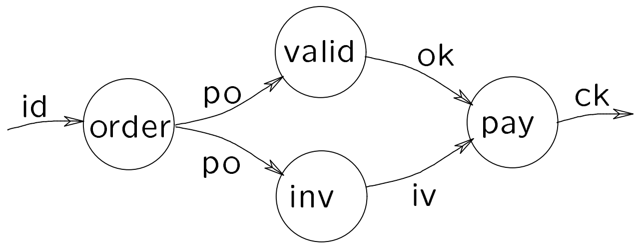
Exception: transaction-processing systems based on an availability paradigm.

Computational Paradigms for Protection

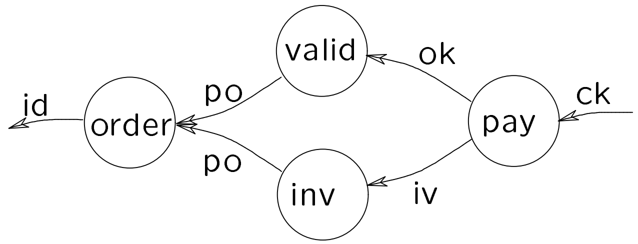
Imperative: a mechanism supporting purchase order processing requires explicit sequencing of tasks.

Sequential Control: [order; validate; invoice; payment]
Explicit Interleaving: [order; (validate||invoice); payment]

Availability: data flow determines sequencing.



Coercion: demand driven sequencing.

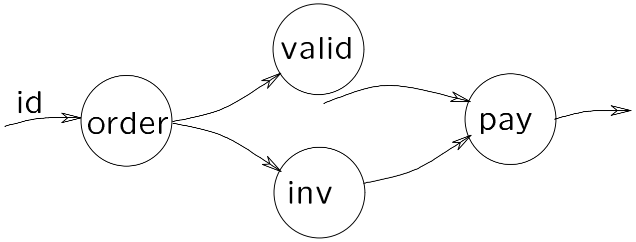


Condensed Graphs

Hierarchical graphs of nodes representing operations or further condensed graphs.

Firing Rule. A node is fireable when an operand is associated with every input port and a destination is associated with the output port.

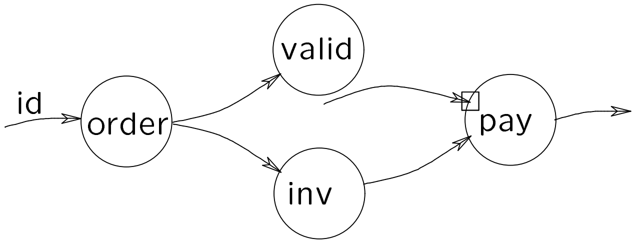
Example. Coercive Validation. Orders validated only when needed.



Sequencing order, inv is data-flow driven.
Sequencing valid, pay is demand driven.

Node valid is input to operand port of pay.
Strictness of the input ports of pay ensures that valid fires before pay.

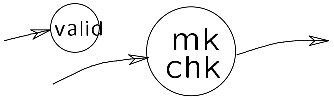
Example. Lazy Validation. Validation may be postponed.



Non-strictness of pay's input port (□) means that valid need not fire before being passed to pay.

pay decides not to fire value valid - output is chk

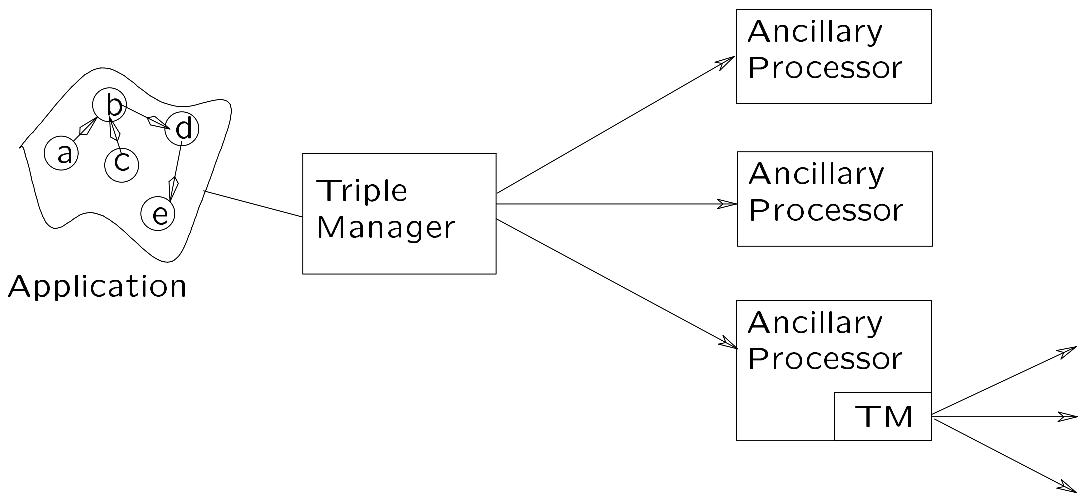
pay decides to postpone firing of valid - output is:



Condensed Graphs provide referential transparency in execution.

The Condensed Graphs Machine Model

Executing a Condensed Graph corresponds to *scheduling* its fireable nodes to run on ancillary processors.



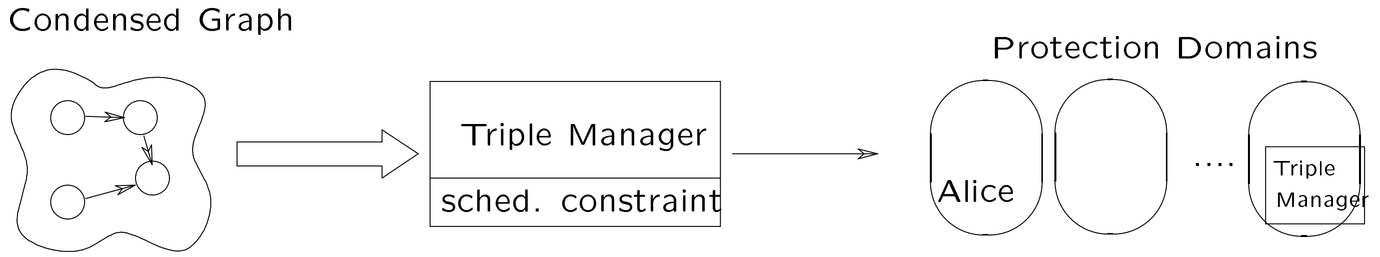
The Triple Manager manages the distribution and synchronization of nodes.

Condensed nodes (representing condensed graphs) may be scheduled to ancillary processors that run Triple Managers.

Condensed Graph Based Protection Mechanisms

Triple Manager schedules application tasks to protection domains, constrained by sequencing of application's graph and scheduling (security) constraint.

Permissions associated with each graph node and protection domain.



Scheduling Constraint: an operation may be scheduled to a protection domain only if the protection domain holds correct permissions that provides authorization to execute the operation.

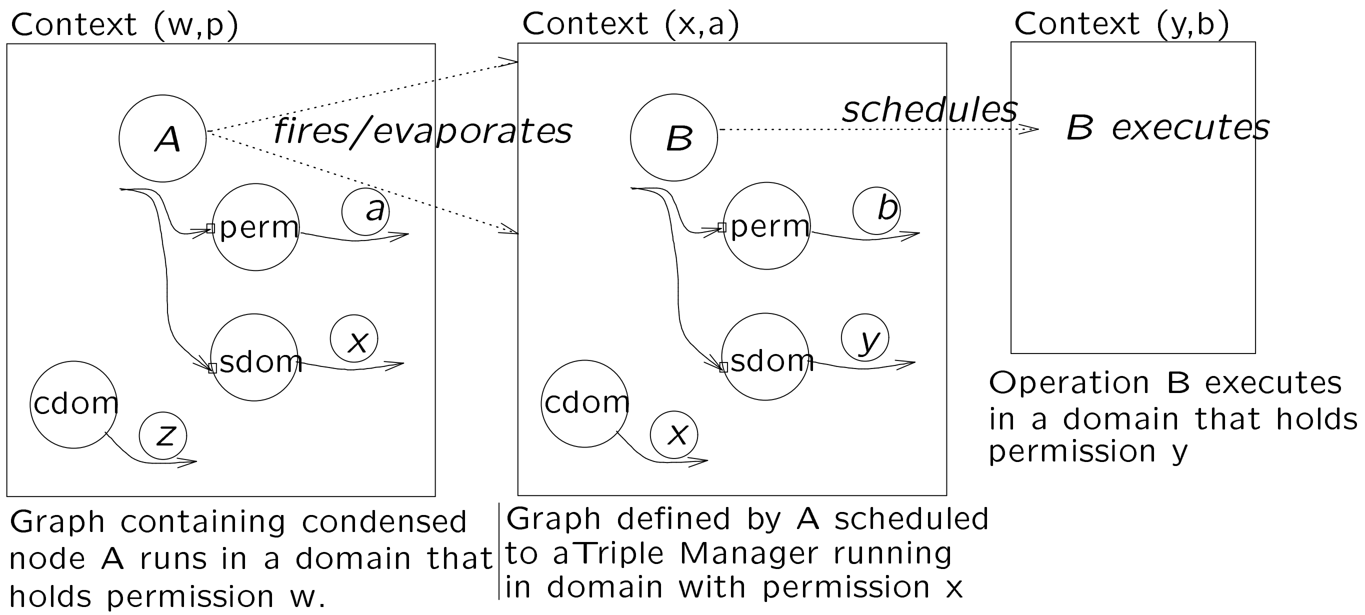
Example. May schedule *sensitive* operation to *sensitive* domain; may schedule *clerk* validate operation to *supervisor* domain (since (supervisor⇒clerk))

Paper describes a more sophisticated scheduling constraint that considers the permissions held by the domain that the triple manager runs in.

Protection Primitives

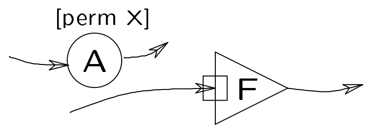
```
Perm perm(NonStrict Node n);
Perm sdom(NonStrict Node n);
Perm cdom();
```

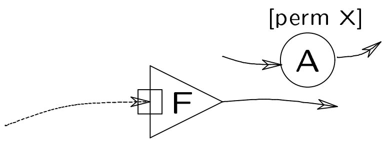
Protection Constraint



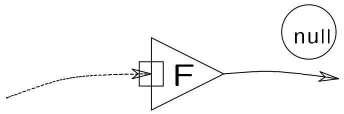
$$\text{implies}((x,a),(y,b)) \equiv (a \leq x) \wedge (y \leq x) \wedge (b \leq y)$$

Fragile Protection



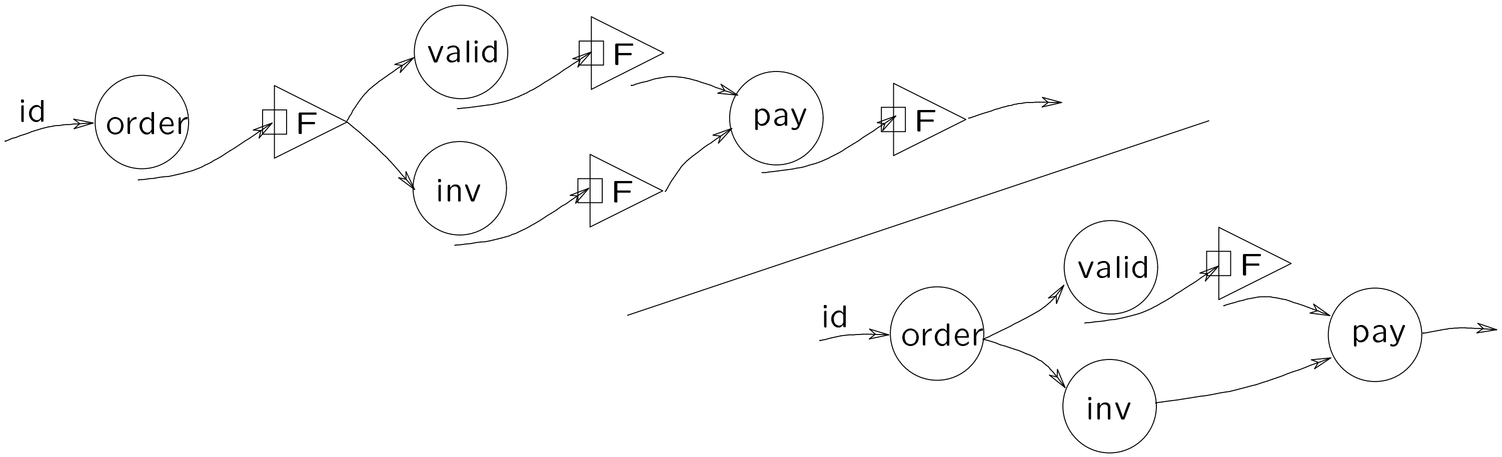


A is scheduled to a domain with authorization $Y \Rightarrow X$

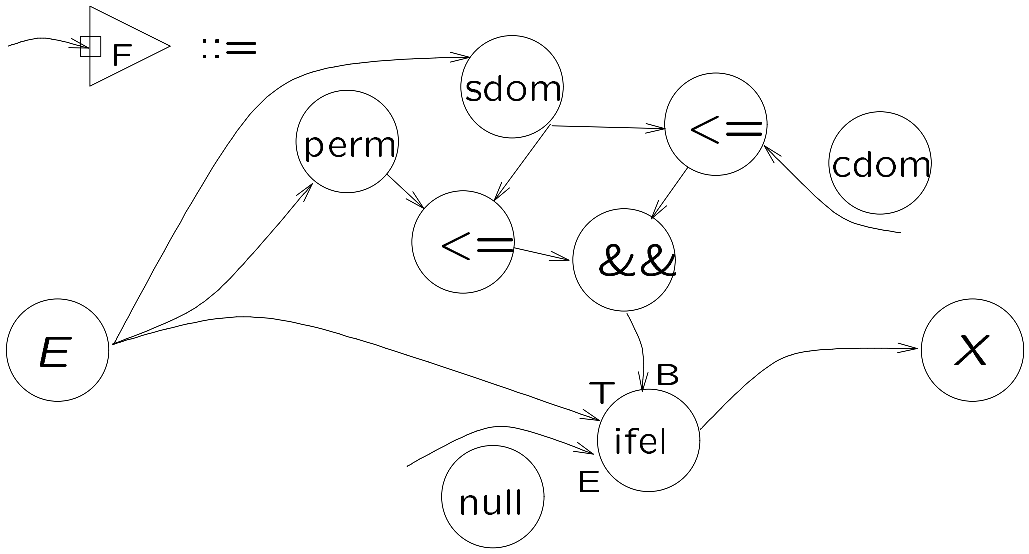


A is not scheduled to domain with authorization $Y \Rightarrow X$

Example

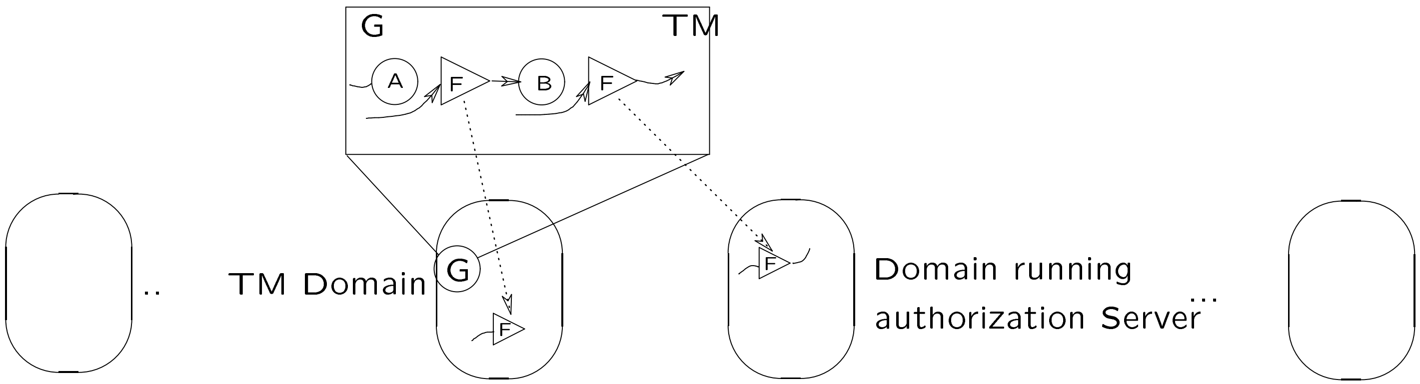


Fragile Protection



Where and When Should a Protection Operator Fire?

The Triple Manager must decide where the protection operation should fire.



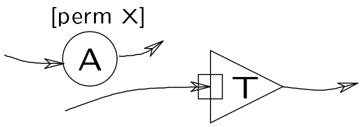
The Triple Manager must decide when the protection node should fire.

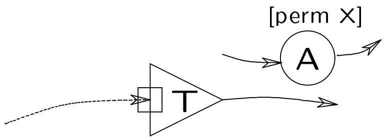
Conservative Protection. Defer the firing of the protection node until the node it mediates has all of its input ports bound.

Speculative Protection. Fire protection operator as soon as it is ready, reserving/deciding upon an authorized domain before the node it mediates is ready to fire.

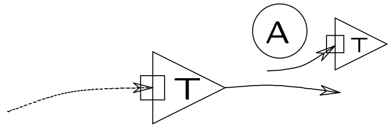
TM Protocols to support remote node invocation are investigated elsewhere.

Tenacious Protection



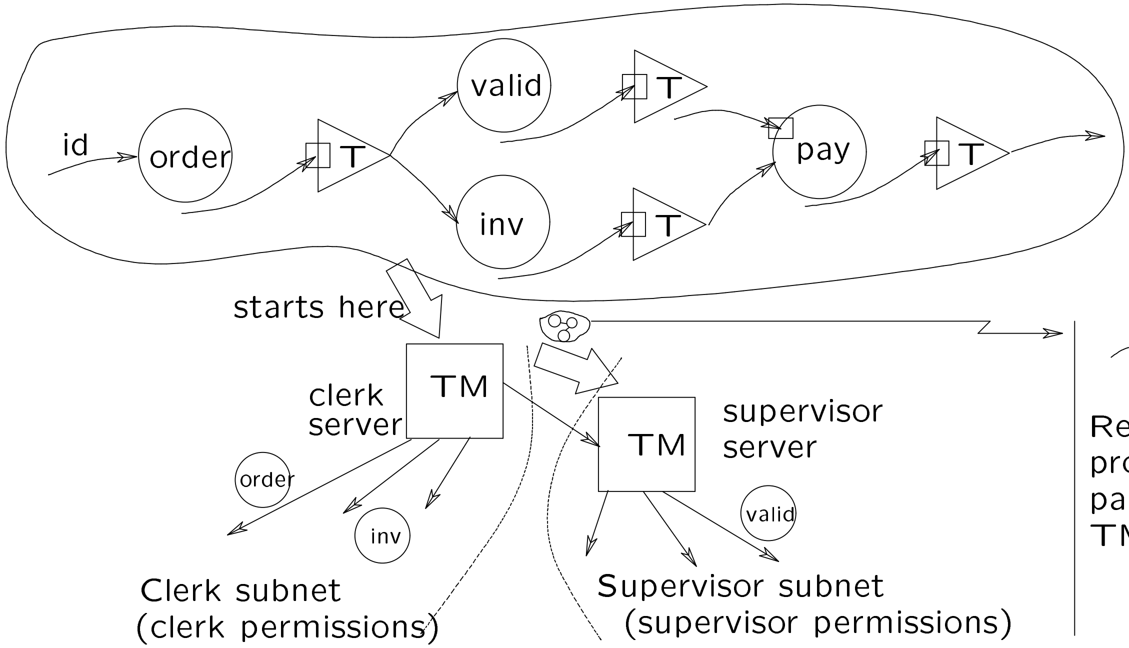


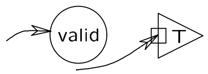
A is scheduled to domain with authorization $Y \Rightarrow X$



A is not scheduled to domain with authorization $Y \Rightarrow X$

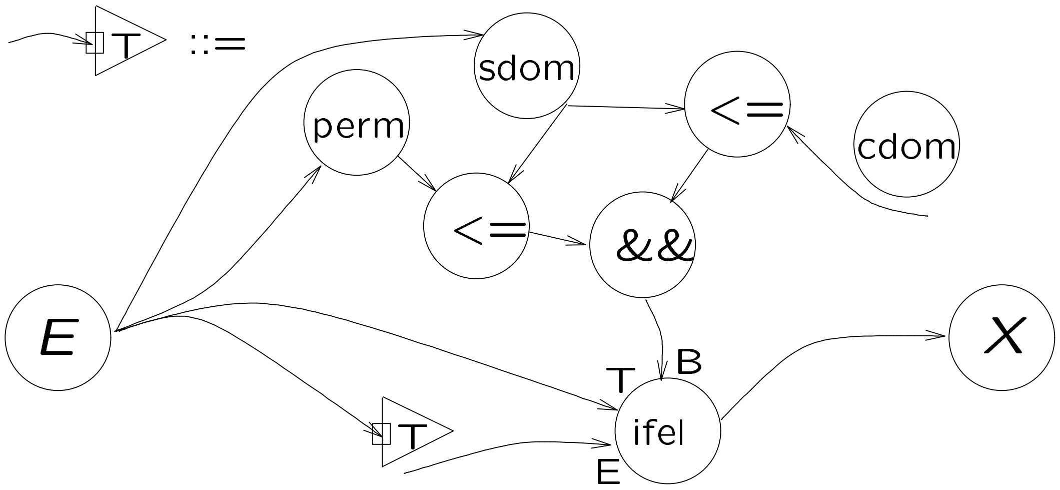
Example. Clerks propose, invoice and pay orders; supervisors validate.





Referential transparancy:
protected "valid" operation
passed to another
TM for scheduling

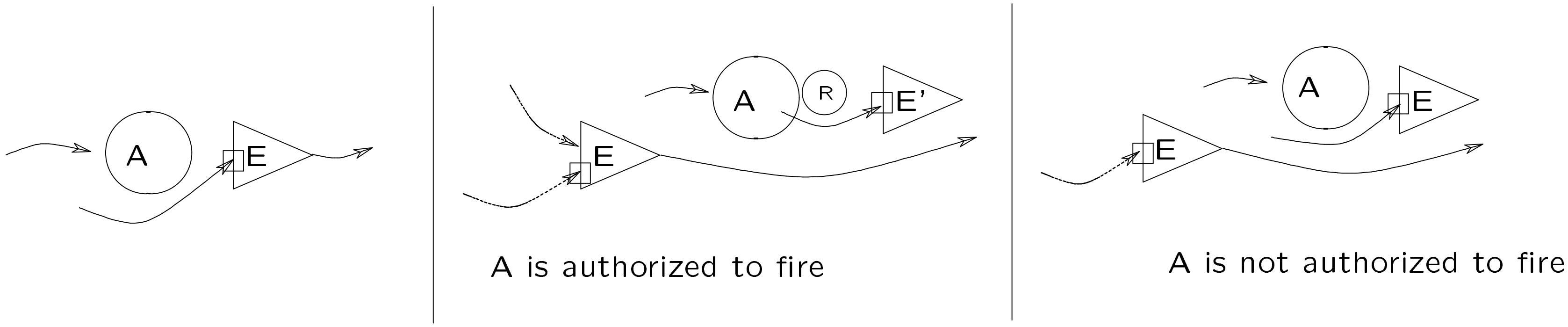
Tenacious Protection



Emergent Protection

Tenacious protection is analogous to a ‘wrapper’ operation.

Extend operator so that wrapper’s behavior changes as computation evolves.



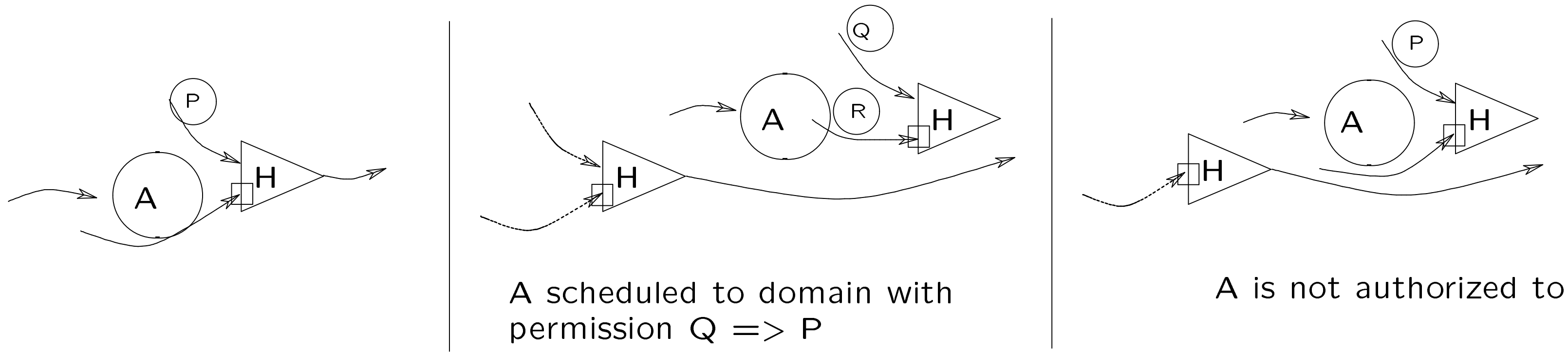
New protection operation emerges protecting result generated by node A.

Emergent mechanism is used to effectively encode protection state, providing a basis for decentralized protection mechanisms (no centralized state).

Emergent Protection: High Water Marks

Operator H(A,P) takes input node A and permits it to fire in a domain $Q \Rightarrow P$. The result R from firing A is then wrapped as H(R,$Q \oplus P$).

The second parameter of H continues a local state (for this node) by acting as a high water mark of the permissions of past domains.

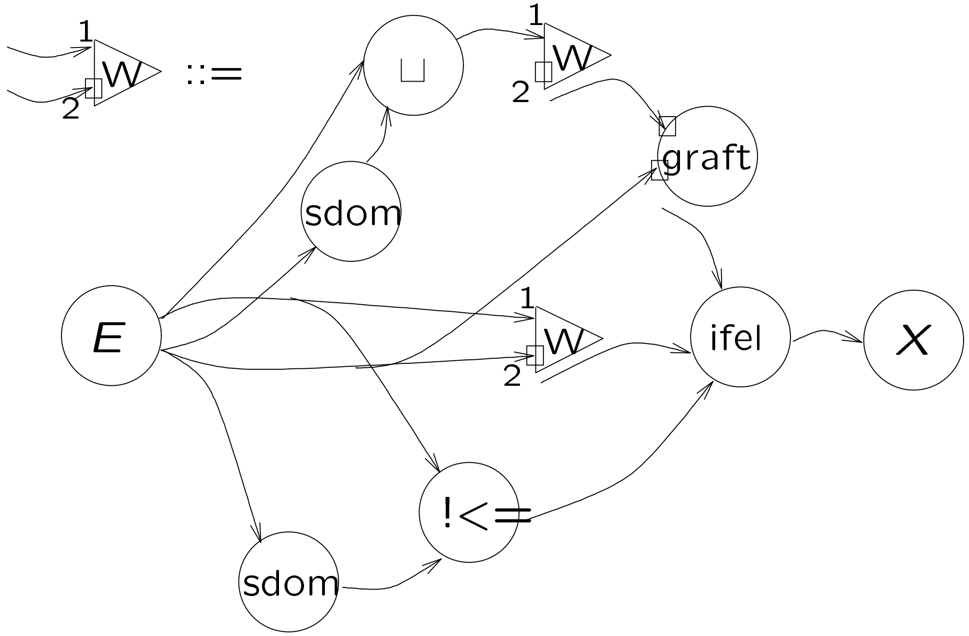


A scheduled to domain with
permission Q => P

A is not authorized to fire

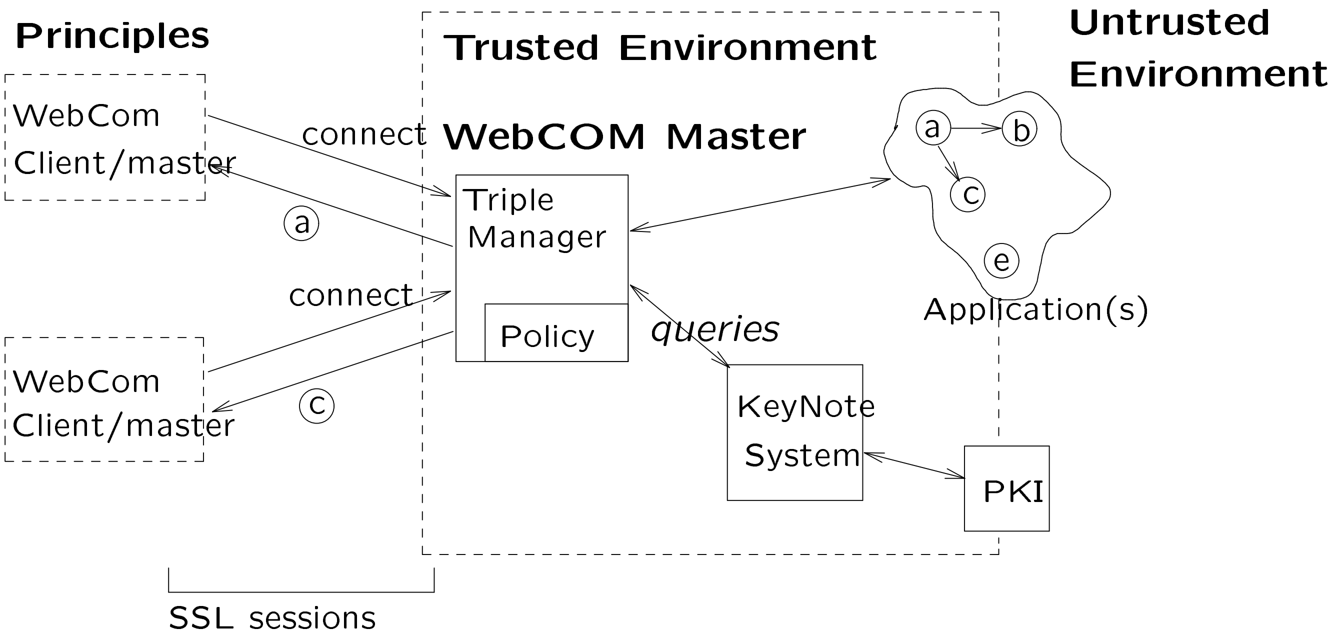
Paper defines a similar operator W that provides de-centralized support for dynamic separation of duties.

Emergent Protection



Secure WebCom: Implementing Fragile Protection

WebCom Masters schedule applications over remote WebCom Clients.



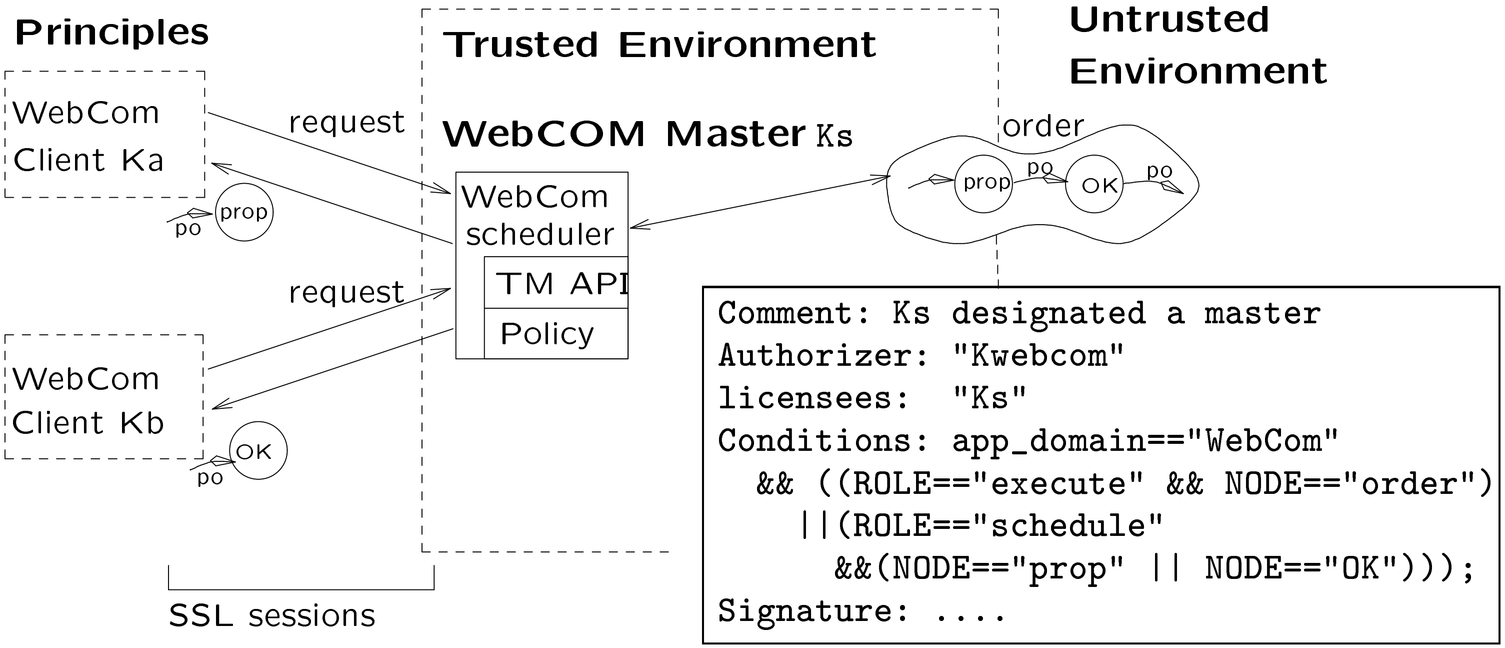
Public keys represent protection domains.

KeyNote credentials define permissions.

KeyNote authorization query corresponds to fragile mediation.

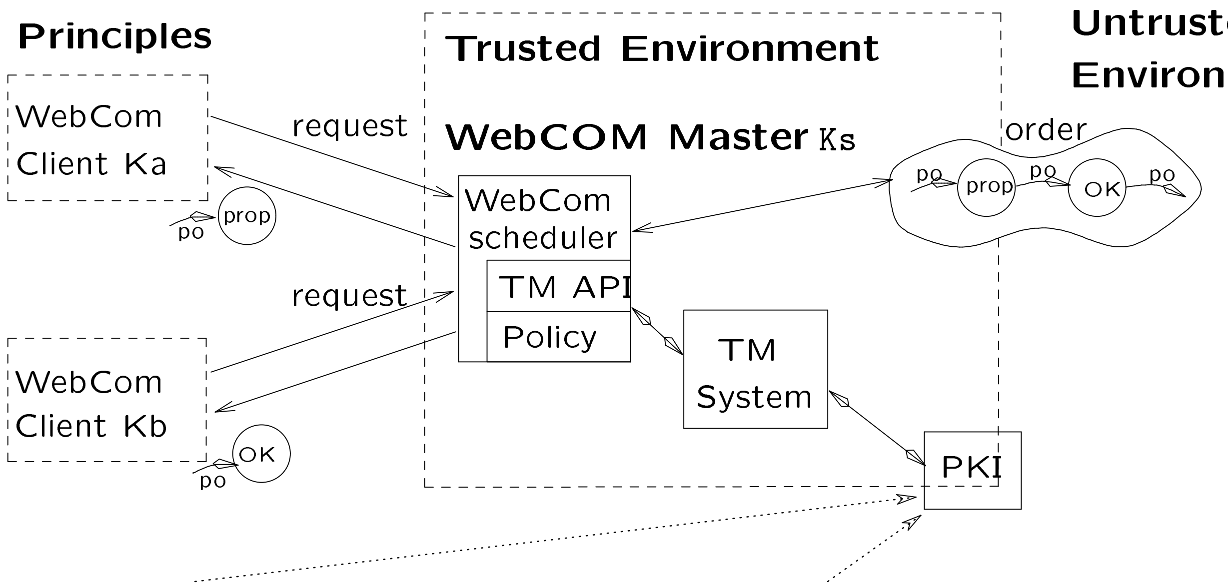
WebCom Master Authorization

WebCom clients use KeyNote to determine whether the credentials provided by the authenticated master prove its authority to schedule operations.



Client Authorization

WebCom Master must determine whether an authenticated client has the necessary authorization to execute certain operations.




```
Comment: Ka authorized to execute prop
Authorizer: "Kwebcom"
licensees:  "Ka"
Conditions: app_domain=="WebCom"
  && ROLE=="execute" && NODE=="prop";
Signature: ....
```

```
Comment: Kb authorized to execute OK
Authorizer: "Kwebcom"
licensees:  "Kb"
Conditions: app_domain=="WebCom"
  && ROLE=="execute" && NODE=="OK";
Signature: ....
```

Conclusion

Condensed Graphs model provides a single framework in which protection may be supported within the imperative, availability and coercion paradigms.

Some advantages of taking this approach include:

- Implementation model exists;
- Protection mechanisms support flexible operation sequencing;
- Referential Transparency of computation and protection;
- Potential for implementing de-centralized protection mechanisms;