



Typed Protocols for P2P Coordination

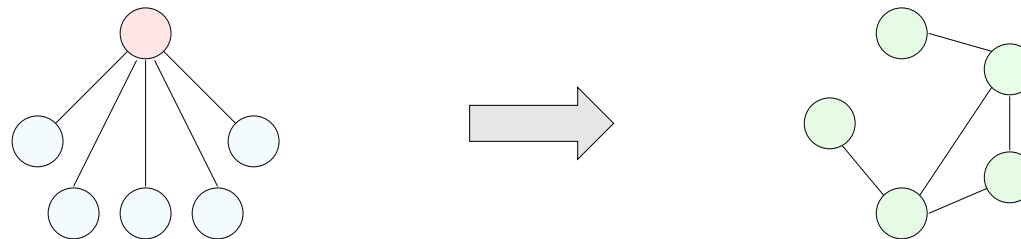
Dr Chris Walton (cdw@inf.ed.ac.uk)
Centre for Intelligent Systems and their Applications (CISA),
School of Informatics, University of Edinburgh, UK.



17 July 2005

P2P Knowledge Management

1. P2P: Dynamic Organisation without Central Coordination:

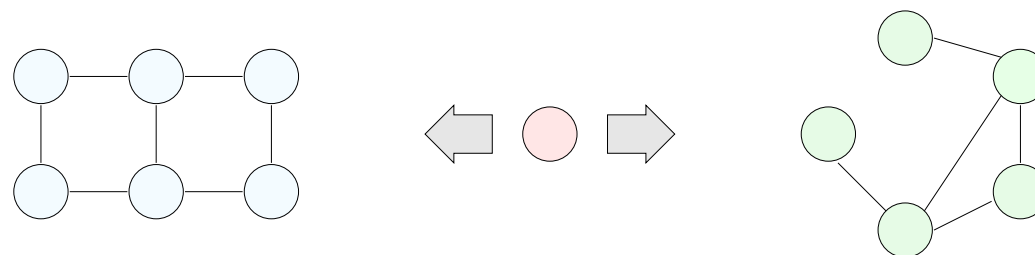


2. KM: Autonomous Peers Exchanging Decentralised Knowledge:

- Not restricted simply to file sharing (content is important).
- Applications in Semantic Web, Grid, DBS, mobile devices, etc.
- Closely Related To Web-Based Multi-Agent Systems.
- Key Issues are: **Scalability**, **Decentralisation**, and **Coordination**.

P2P Coordination

- **Problem:** There are many different P2P techniques and technologies.
 - Would like peers to be able to operate with different technologies.
 - Would like to coordinate peers of different types.



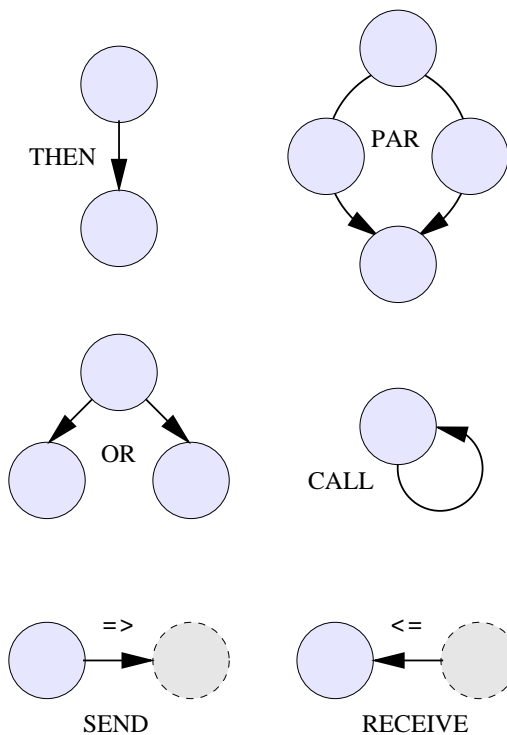
- **Goal:** To facilitate the rapid construction of ad-hoc P2P applications.
- **Our Solution:** Executable P2P Coordination Protocols.

Coordination Protocols

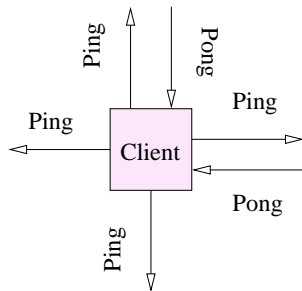
- Techniques inspired by agent-based coordination, e.g. Electronic Institutions, Conversation Policy (Societal view and Social norms).
- **Coordination is based on the definition of Protocols:**
 - Provide a “safe envelope” in which coordination can happen.
 - Define a clear role for each peer, and goals for the system.
 - Does not remove autonomy, not restricted to single type of peer.
- Our protocols are specified in a lightweight coordination calculus (MAP):
 - A sugared variant of the π calculus of mobile processes.
 - A formal semantics and type system have been defined.
 - Protocol specifications are directly executable by peers.

MAP Abstract Syntax

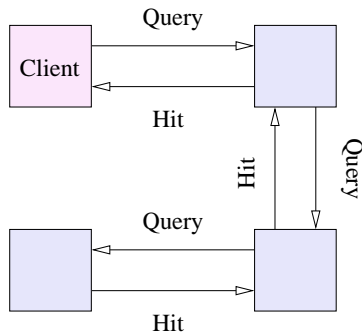
P	$::=$	$n(r\{\mathcal{M}\})^+$	(Protocol)
M	$::=$	$\text{method}(\phi^{(k)}) = op$	(Method)
op	$::=$	α	(Action)
		$op_1 \text{ then } op_2$	(Sequence)
		$op_1 \text{ or } op_2$	(Choice)
		$op_1 \text{ par } op_2$	(Parallel)
		$\text{waitfor } op_1 \text{ timeout } op_2$	(Iteration)
		$\text{call}(\phi^{(k)})$	(Replication)
α	$::=$	ϵ	(No Action)
		$v = p(\phi^{(k)})$	(Decision)
		$\rho(\phi^{(k)}) \Rightarrow \text{agent}(\phi^1, \phi^2)$	(Send)
		$\rho(\phi^{(k)}) \Leftarrow \text{agent}(\phi^1, \phi^2)$	(Receive)
ϕ	$::=$	$_ \mid a \mid r \mid c \mid v$	(Terms)
τ	$::=$	$utype \mid atype \mid rtype \mid tname$	(Types)



Example: Gnutella Protocol



Ping/Pong Protocol



Query/Hit Protocol

```

%node{
  method main() =
    $id:a = getId() then startSharing($id:a) then $nodes:alist = getNodes()
    then (call sendping($nodes:alist) or call mainloop($id:a))
  method mainloop($id:a) =
    waitfor
      ((ping() <= agent($n:a, %node) then pong() => agent($n:a, %node))
      or ((pong() <= agent($n:a, $role:r) then addActive($n:a, $role:r))
      or ((query($f:string) <= agent($n:a, $r:r) then
          ($fl:file = getFile($f:string) fault nofile then
            hit($f:string, $id:a) => agent($n:a, $r:r))
          or (setQuery($f:string, $n:a, $r:r) then
              $nodes:alist = getActiveNodes() then
                call sendquery($f:string, $nodes:alist)))
          or ((hit($f:string, $hid:a) <= agent($n:a, %node) then
              $nodes:alist = getQueryList($f:string) then
                call sendhits($f:string, $hid:a, $nodes:alist))
          or (download($f:string) <= agent($client:a, %client) then
              $fl:file = getFile($f:string) fault nofile then
                file($fl:file) => agent($client:a, %client))))))
    then call mainloop($id:a)}

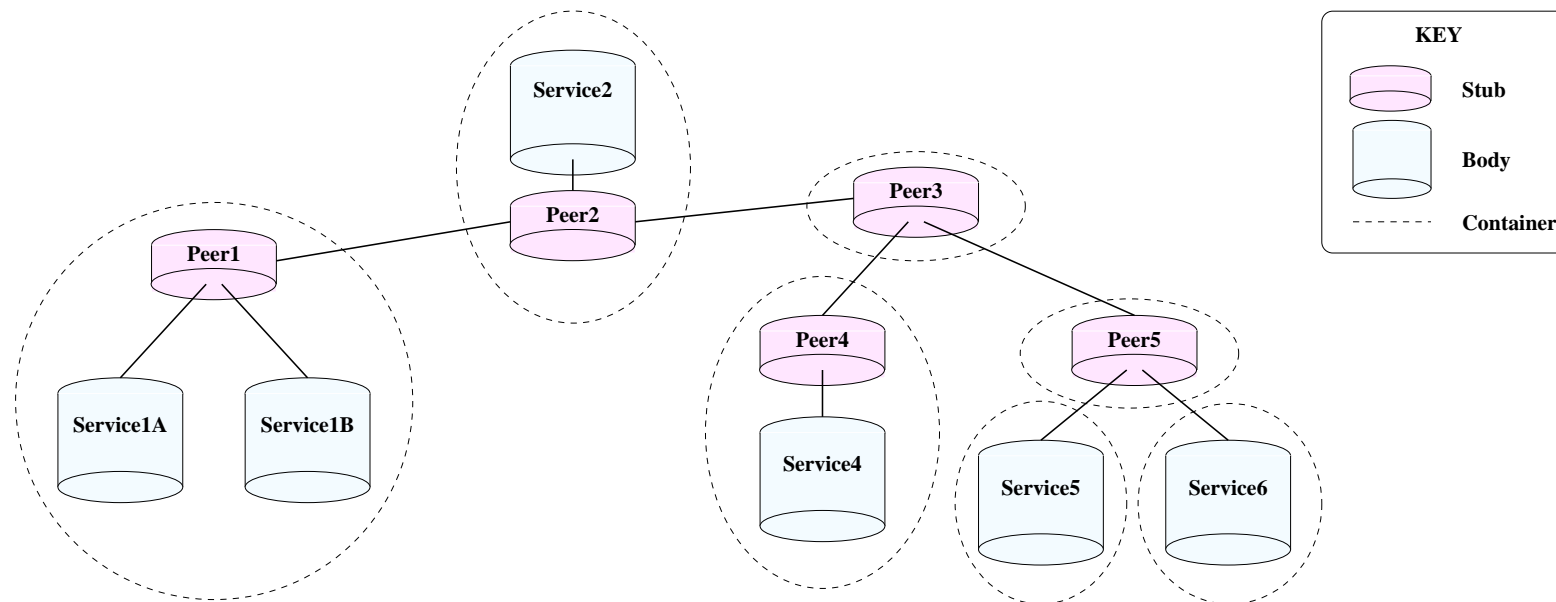
```

Protocol Verification

- A separate specification allows for protocol verification before deployment.
- **Structural Verification:**
 - Defined formally by a static type system, and type checker.
 - Ensures that a protocol is internally consistent.
- **Behavioural Verification:**
 - Exhaustive verification by model checking (SPIN).
 - Ensures that a protocol is free of undesirable external behaviour (e.g. deadlocks, starvation).
 - Can also check specific properties of protocols (local checking).

P2P Service Composition

- An architecture for performing service composition using MAP protocols:
 1. Separate the Services and Protocol Execution (Body and Stub).
 2. Permit a range of different composition strategies:



P2P Knowledge Management

- **Protocols as an anchor for knowledge technologies:**
 1. Allow us to compose knowledge services into P2P applications.
 2. Permit the composition of external services without modification.
 3. Provide a means to verify the composition before deployment.
- Intended to operate in concert with Semantic Web technologies (e.g. OWL-S, WSMO) for discovery and brokering.
- Future Directions:
 - Direct discovery of services using ontological knowledge techniques.
 - Automated protocol generation (using planning and simulation techniques).

Summary

- **Initial Problem:**

Would like to adopt multiple protocols and services in the construction of ad-hoc P2P applications.

- **Proposed Solution:**

Executable specifications of P2P coordination protocols (MAP).

Key Advantages:

1. Separation of protocol from actual services.
2. Verification of protocols before deployment.
3. Rapid construction of P2P applications.

