High performance Peer to Peer Distributed Computing with Application to Obstacle Problem

D. EL BAZ (LAAS-CNRS, Toulouse France)

Coauthors : T. T. NGUYEN, P. SPITERI, G. JOURJON, M. CHAU

funded by ANR

HOTP2P 2010, April 23, 2010.

Outline

1. **Goal**
2. **Self-adaptive protocol**
3. **Environment**
4. **Experiments**
5. **Conclusions**

1. Goal

- Great development of peer to peer applications
  - File sharing, video, ...
  - Recent advances in microprocessor architecture and high bandwith network → new applications like distributed HPC computing/computing on the Internet.
- Great challenges
  - Scalability,
  - Heterogeneity,
  - Volatility,
  - Existing protocols, TCP, UDP not well suited to HPC.

1. Goal (cont'd)

- High performance peer to peer computing:
  - Task parallel model, distributed iterative methods.
  - Direct communications between peers.
  - Applications: numerical simulation & optimization.

- Self-adaptive protocol:
  - based on Cactus framework
  - uses micro-protocols
  - chooses dynamicaly the most appropriate communica-tion mode in function of elements of context from network level and choices at application level.

2. Self-adaptive protocol

- Micro-protocols
  - Introduced in x-kernel
  - Approach to design self-adaptive communication protocols
- Micro-protocols implement a functionnality (sample)
  - Communication: Synchronous, Asynchronous.
  - Fragmentation: FixeSize, Resize.
  - Reliability: Retransmission, PositiveAck, NegativeAck, DuplicateAck.
  - Order : LossyFifo, ReliableFifo.
  - Congestion control: NewReno TCP Congestion Control.
- Composition of micro-protocols → protocol
  - Reuse code, facilitate design, configure dynamically.

2. Self-adaptive protocol (cont'd)

- Protocol composition framework → deployment of architecture
  - Hierarchical model (stack of protocols), **x-kernel, APPIA** frameworks.
  - Nonhierarchical model (no order), **Coyote** and **ADAPTIVE** frame'ks.
  - Hybrid model (combo), **XQoS** and **<u>Cactus</u>** frameworks → CTP.
- Cactus framework
  - flexible, efficient.
  - Two grain levels:
    - Composite protocols : individual protocol made of micro-protocols.
    - Protocol stack : composite protocols layered on the top of each others.
  - Protocols can reconfigure by substituting protocols or micro-protocols.

2. Self-adaptive protocol (cont'd)

- Cactus is an event based framework:
  - Events: state changes, e.g. arrival of messages.
- Micro-protocols structured as a collection of event handlers:
  - Event handler : procedure like segments of codes bound to events.
  - When an event occurs all handlers bound to that event are executed.
- Our modifications to Cactus → improve protocol performance/facilitate reconfiguration:
  - Concurrent handler execution (multicore machines).
  - Eliminate unnecessary copies between layers (use pointers)
  - Operation for micro-protocol removing.

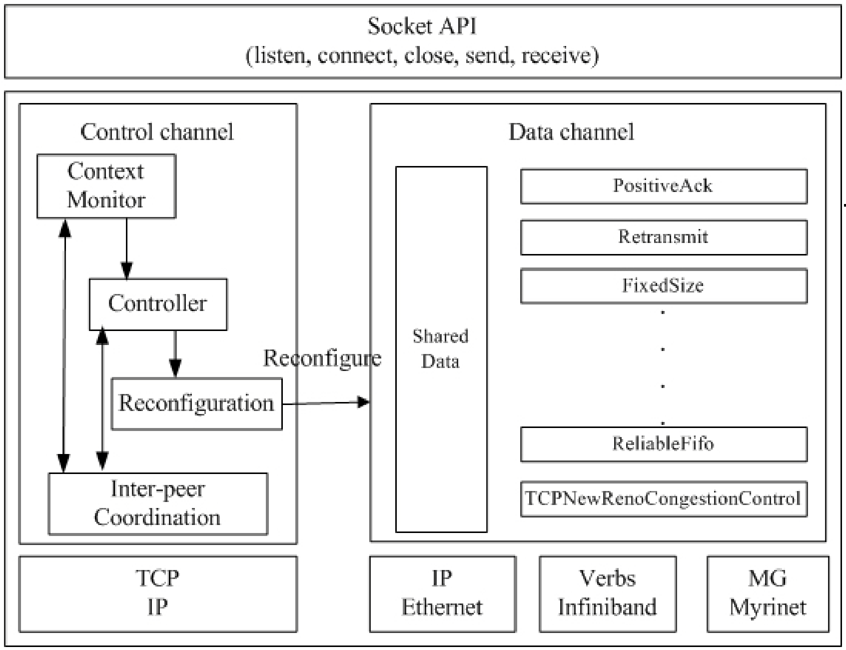
2. Self-adaptive protocol (cont'd)

❑ P2PSAP protocol architecture

Manages session opening an closure

Captures context information

Reconfigures data channel/coordinates peers



Transfers data between peers

API

~~~~~~~~~~~~

Transport layer

~~~~~~~~~~~~

Physical layer

1 Goal — 2 Protocol — 3 Environment — 4 Experiments — 5 Conclusions

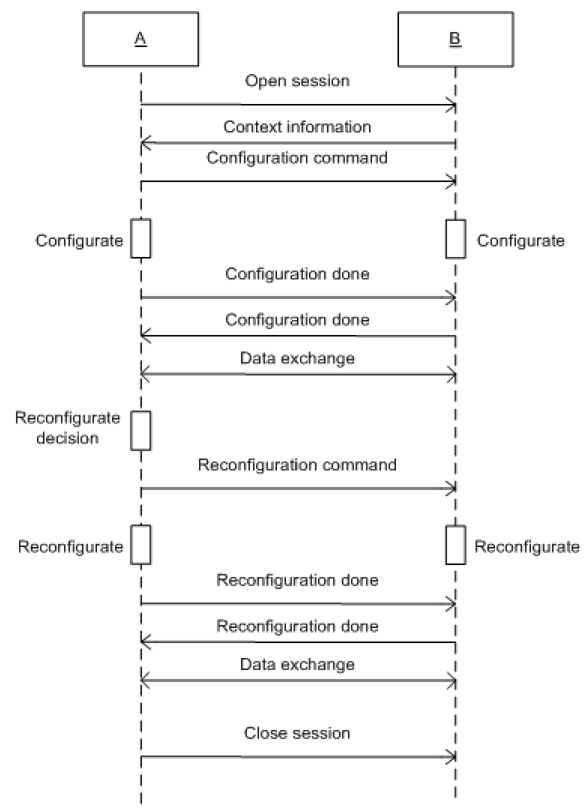
2. Self-adaptive protocol (cont'd)

- Communication adaptation rules

Scheme / Link	Synchronous	Asynchronous	Hybrid
Intra-cluster	Synchronous Reliable Com.	Asynchronous Reliable Com.	Synchronous Reliable Com.
Inter-cluster	Synchronous Reliable Com.	Asynchronous Unreliable Com.	Asynchronous Unreliable Com.

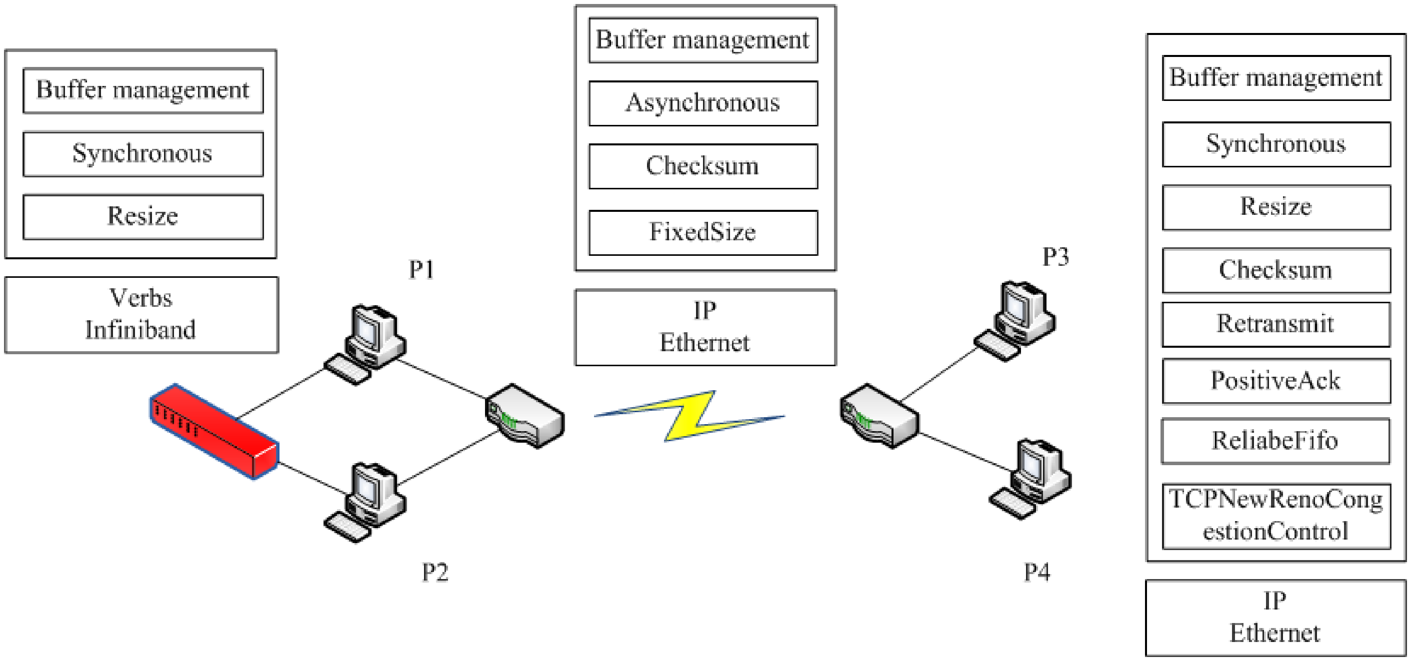
2. Self-adaptive protocol (cont'd)

- Reconfiguration mechanism



2. Self-adaptive protocol (cont'd)

- Example of scenario



3. Environment

- Direct communication between peers
- Reduced set of communication operations:

  - only send and receive operations (P2P_send and P2P_receive).

  - facilitate programming, hide complexity.
- Communication mode can vary with context:

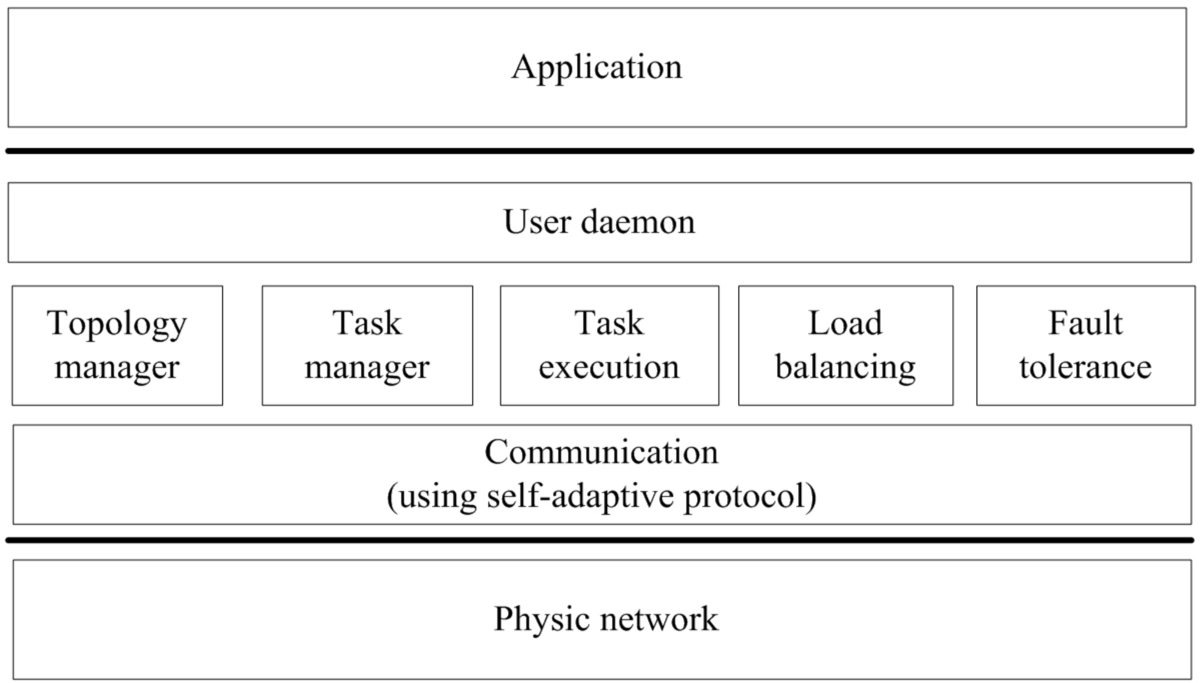
  - programmer does not select directly a communication mode (programmer can select a scheme of computation).

  - communication mode depends on the context and is determined by the protocol.

  - good efficiency.

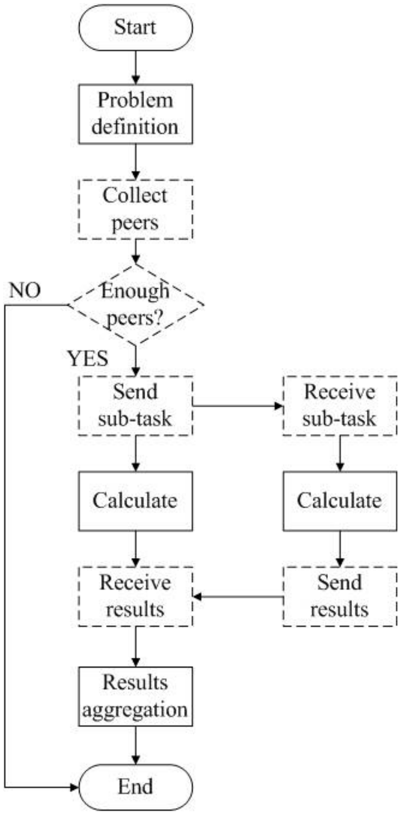
3. Environment (cont'd)

- P2PDC Environment architecture



3. Environment (cont'd)

- Application deployment



4. Experiments

- 3D Obstacle problem
  - numerical simulation problems (pde)
  - financial mathematics, e.g. option pricing
  - mechanics

4. Experiments (cont'd)

- Fixed point problem:

$$\begin{cases} Find\ u^* \in V\ such\ that \\ u^* = F(u^*), \end{cases}$$

Distributed asynchronous iterative scheme:

$$\begin{cases} u_i^{p+1} = F_{i,\delta}\left(u_1^{\rho_1(p)}, \dots, u_j^{\rho_j(p)}, \dots, u_\alpha^{\rho_\alpha(p)}\right)\ if\ i \in s(p), \\ u_i^{p+1} = u_i^p\ if\ i \notin s(p), \end{cases}$$

$$\begin{cases} s(p) \subset \{1, \dots, \alpha\}, s(p) \neq \emptyset, \forall p \in N, \\ \{p \epsilon N | i \in s(p)\}, is\ infinite, \quad \forall i \in \{1, \dots, \alpha\}, \end{cases}$$

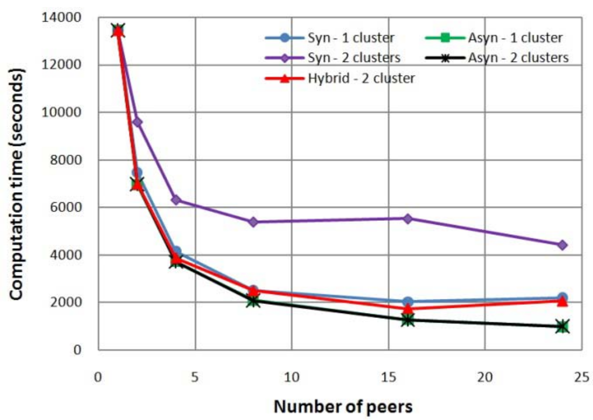
$$\begin{cases} \rho_j(p) \in N, 0 \leq \rho_j(p) \leq p, \forall j \in \{1, \dots, \alpha\}, \forall p \in N, \\ lim_{p \to \infty} \rho_j(p) = +\infty, \forall j \in \{1, \dots, \alpha\}. \end{cases}$$

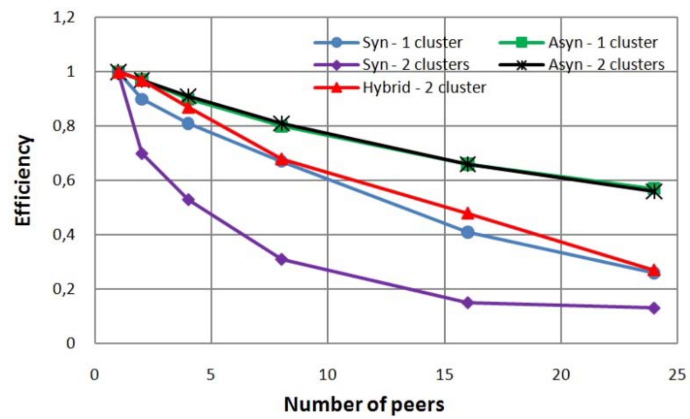
(5)

4. Experiments (cont'd)

❑ Results

3D obstacle problem, slice decomposition, 3,000,000 variables, NICTA testbed, Sidney.





5. Conclusions

- ❑ Self-adaptive protocol P2PSAP for P2P HPC
- ❑ Current version of environment P2Pdc
- ❑ Experiments on NICTA and Grid 5000 testbeds for obstacle problem.
- ➢ Decentralised functions of P2PDC.
- ➢ Improvements: code, protocol, environment.
- ➢ Applications: process engineering, logistics.
- ➢ Other testbeds PlanetLab (GENI).
- ➢ Self-organization → efficiency & everlastingness.