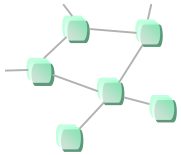
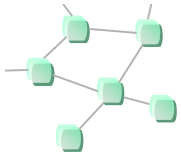


P2PS: a Peer-to-Peer Development Platform

Valentin Mesaros[1], Bruno Carton[2], and Peter Van Roy[1]

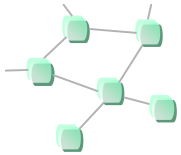
[1] *Université catholique de Louvain, Belgium*
{valentin, pvr}@info.ucl.ac.be

[2] *CETIC, Belgium*
bruno.carton@cetic.be



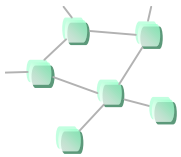
Peer-to-peer computing/systems

- **Equality** between the entities involved in a common action
  - no centralized control → no single point of failure
  - identical responsibility and capabilities (e.g, client, server, and router)
  - symmetric communication.
- Form **overlay** (logical) networks at the application level.
- Location-independent naming.
- High (fully) decentralization, robustness, scalability, high-availability.
- Appropriate for systems with dynamic topologies: *self-organization*.



"Structured" peer-to-peer systems

- Emphasize on lookup efficiency and provided guaranties
  - find any item, out of *N*, in at most H_{max} hops, given max R_{max} links per node
  - usually $H_{max} = O(\log N)$ and $R_{max} = \log N$
- Every node is ''imposed'' a well defined set of neighbors.
- Provide a **distributed hash table** - like API
  - data items are associated keys, and each node is responsible for a subset
  - insert(key, data) , lookup(key) , remove(key)
- Build the routing table adaptively.



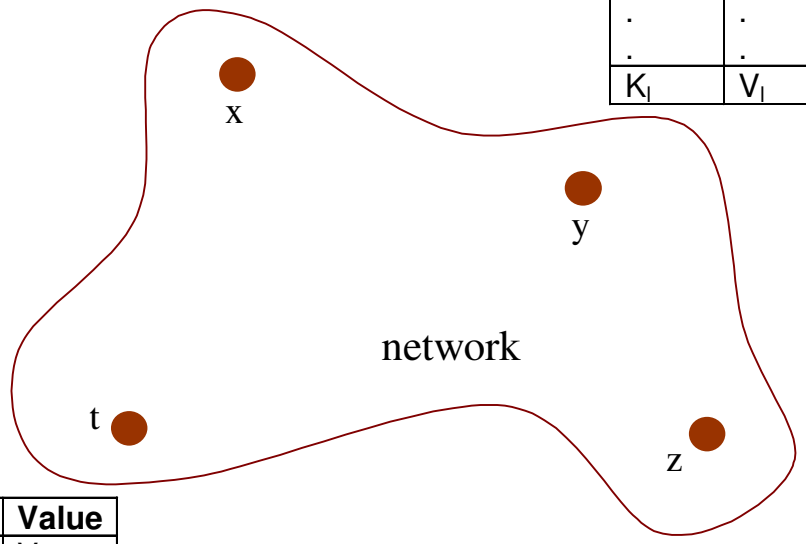
Distributed Hash Table (DHT)

Key	Value
K_0	V_0
K_1	V_1
K_2	V_2
.	.
.	.
K_{N-1}	V_{N-1}

local view

Key	Value
K_0	V_0
.	.
.	.
K_i	V_i

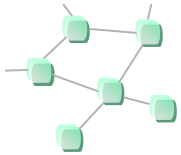
Key	Value
K_j	V_j
.	.
.	.
K_l	V_l



Key	Value
K_p	V_p
.	.
.	.
K_{N-1}	V_{N-1}

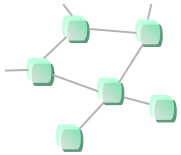
Key	Value
K_m	V_m
.	.
.	.
K_n	V_n

distributed view



What is P2PS?

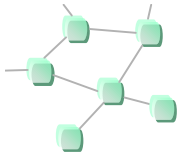
- P2PS is a library/platform for developing peer-to-peer (P2P) applications.
- P2PS provides the programmer with the ability to **easily** write and work with P2P applications.
- It offers different P2P primitives and services such as, *efficient data lookup*, *group communication,* and *fault-resilience*.
- It implements the Tango algorithm. Like DKS, Tango is a generalization of Chord but it scales better.
- P2PS is written in Mozart/Oz .



P2PS' functionality

- Create a peer-to-peer network
  - create the first node of the P2P network
  - set up the P2P network characteristics (e.g., max net size, fault-tolerance)
  - create an *access point* for this node.

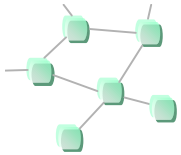
- Join and leave a peer-to-peer network
  - given an access point, join a node to the P2P network
  - at join, create an access point for this node
  - a node can leave P2PS gracefully or ungracefully (i.e., it just fails)
  - after a join or a leave the P2P network will *self-organize* to remain efficient.



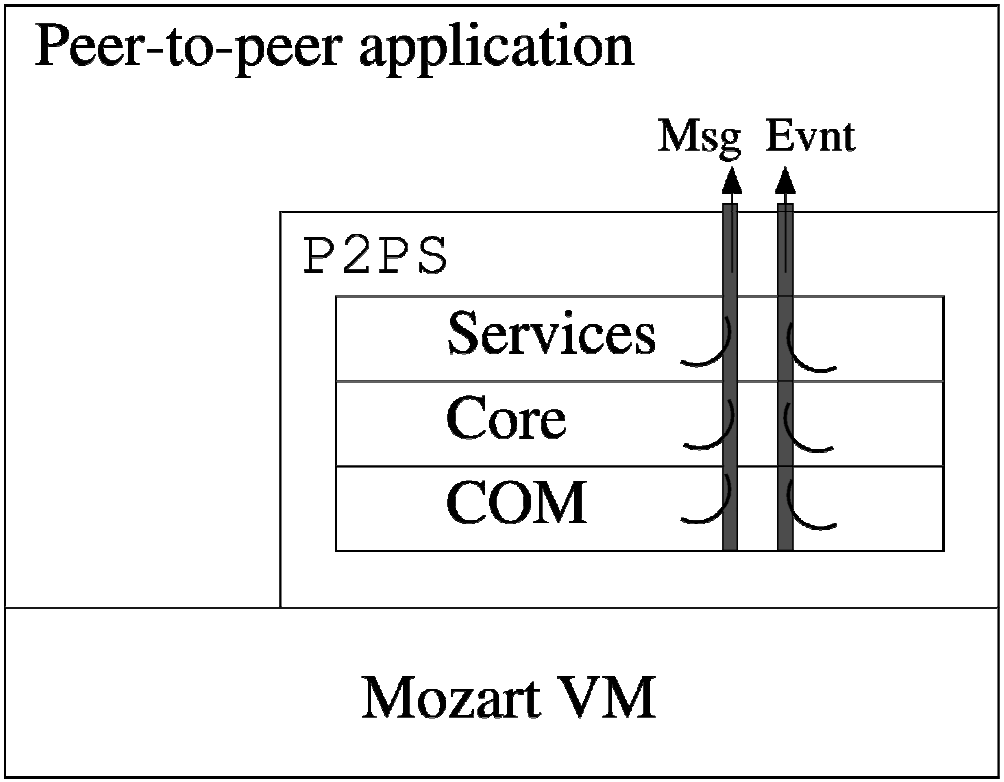
P2PS' functionality *(cont)*

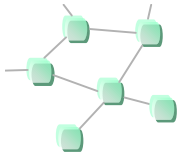
- Message sending and receiving
  - the communication is done over the overlay network and it is based on the *efficient key-based routing* algorithm of Tango
  - P2PS offers one-to-one and one-to-many (i.e., explicit multicast and broadcast) communication primitives.

- Monitoring
  - an application can be notified when the node or the network status has changed
  - one has access to statistics at a node (e.g., nr. of sent or forwarded messages).



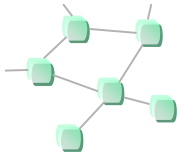
P2PS: internals of a node





P2PS: internals *(cont)*

- COM layer
  - access point (AP) creation
  - connection establishment
  - offer basic communication primitives
  - fault (detection and) handling
- Core layer
  - implement the Tango protocol
  - joining/leaving the P2P network
  - message routing
  - topology maintenance
- Services layer
  - act like a wrapper, building up the raw primitives offered by Core
  - offer more specialized services such as reliable communication.



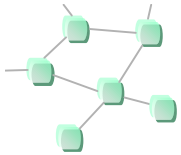
P2PS' simple API

- A node in P2PS can be initiated :
  - either as the **first node** in the P2P network,

```
createNet (netConfig:   +NetC    <= NetConfig
           nodeConfig: +NodeC  <= NodeConfig
           apConfig:    +APC     <= APConfig
           msgStrm:     ?MS      <= _
           evntStrm:    ?ES      <= _          )
```

  - or as a **regular node joining** a P2P network

```
joinNet (remoteAP:   +RemAP
         nodeConfig: +NodeC  <= NodeConfig
         apConfig:    +APC     <= APConfig
         msgStrm:     ?MS      <= _
         evntStrm:    ?ES      <= _          )
```



P2PS' simple API *(cont)*

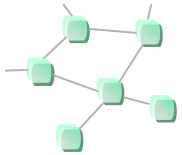
- Sending and receiving are asynchronous
  - nodes and keys are identified by positive integers
  - any Oz value can be a message
  - a message can be sent *to a particular node* or *to the responsible* of a key
  - incoming messages are available on *the message stream*.

send (dst: +*NodeId*
msg: +*Msg*
toResp: +TR <= false)

multicast (dst: +*LNodeId*
msg: +*Msg*
toResp: +TR <= false)

broadcast (msg: +*Msg*
toMyself: +TM <= false)

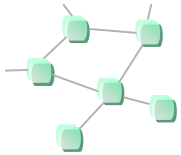
sendToSucc (msg: +*Msg*
nrSucc: +*N*)



P2PS' simple API *(cont)*

- Changes in the status of the node and network are available on *the event stream*. Some examples of possible events:
  - connected
  - alone
  - newpred
  - newsucc

- Get statistics: **getStatistics** (?*Stat*)
  - number of data messages sent and forwarded
  - number of control messages sent and forwarded
  - the current number of in/out links
  - ...

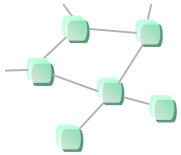


An Example using P2PS : *node1*

```
declare
[P2PS] = {Module.link ['x-ozlib://cetic_ucl/p2ps/P2PS.ozf']}
MS

% Create first node (with id 1) in a P2PS network. Set the port# to 3001.
OP2PS = {New P2PS.p2pServices
                createNet(nodeConfig: nodeConfig(nodeId:1)
                                apConfig: apConfig(pn:3001)
                                msgStrm: MS)}

% Display each message received on the message stream.
for M in MS do
   {Show M}
end
```

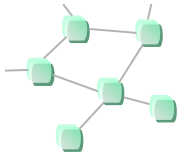


An Example using P2PS : *node2*

```
declare
% Construct the access point token of a node (here, node 1) in the P2PS network.
RemAP = {P2PS.address2ap "206.102.118.78" 3001}

% Create a node with id 16 and join the network, using the token RemAP.
OP2PS = {New P2PS.p2pServices
            joinNet(remoteAP: RemAP
                    nodeConfig: nodeConfig(nodeId:16)
                    apConfig: apConfig(pn:3002))}

% Get the message stream and display each received messages.
for M in {OP2PS getMsgStrm($)} do
   {Show M}
end
```



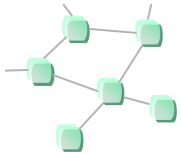
An Example using P2PS : *node3*

```
declare
% Construct the access point token of a node (here, node 2).
RemAP = {P2PS.address2ap "192.227.140.1" 3002}

% Create a node and join the network, using the token RemAP.
% The choice of the node id and local port # is left to the system.
OP2PS = {New P2PS.p2pServices joinNet(remoteAP: RemAP)}

% Send a message to the responsible of key 42.
{OP2PS send(dst:42 msg:anOzValue toResp:true)}

% Multicast a message to nodes with id 1 and 16.
{OP2PS multicast(dst:[1 16] msg:hello)}
```



P2PS is ready to be deployed

- P2PS offers means for developing peer-to-peer applications:
  - scalable and fault-tolerant, with efficient communication
  - P2PS is easy to understand and straightforward to use.
- There is ongoing work to enhance its functionality (e.g., object transactions).
- P2PS library was released last fall on MOGUL
  http://www.mozart-oz.org/mogul/info/cetic_ucl/p2ps.html
- P2PS is used to build different applications
  - *PostIt*: a decentralized collaborative application for group message exchange
  - *CommunityPanel*: a decentralized tool for real-time collaboration software edition
  - Some graduation thesis are using P2PS as distribution middleware.
- We're working at deploying and testing P2PS on PlanetLab.

Just use it! … and let us know about your opinions.