

P2P experimentations with SPLAY: *from idea to deployment results in 30 minutes*

- SPLAY, integrated system for **developing**, **deploying** and **controlling** distributed systems experiments
 - **Concise** code (similar to pseudo-code) & comprehensive **libraries**
 - **Sandboxing** and secure testbeds management
 - **Resource selection** facilities
 - Cutting edge features: advanced networking, logging, **churn management**, ...

Algorithm 1: A simple push-pull dissemination protocol, on node P .

```

Constants
  f: Fanout (push:  $P$  forwards a new message to  $f$  random peers).
  TTL: Time To Live (push: a message is forwarded to  $f$  peers at most TTL times).
  Δ: Pull period (pull:  $P$  asks for new messages every Δ seconds).

Variables
  R: set of received messages IDs
  N: set of node identifiers

/* Invoked when a message is pushed to node P by node Q */
function PUSH(Message m, hops)
  if m received for the first time then
    /* Store the message locally */
    R ← R ∪ m
    /* Propagate the message if required */
    if hops > 0 then
      invoke PUSH(msg, hops-1) on f random peers from N

/* Periodic pull */
thread PERIODICPULL()
  every Δ seconds do
    invoke PULL(R, P) on a random node from N

/* Invoked when a node Q requests messages from node P */
function PULL(RQ, Q)
  foreach m | m ∈ R ∧ m ∉ RQ do
    invoke PUSH(m, 0) on Q
    
```



```

pushPull(msg)
  m = createMessage()
  f = randomPeers(f)
  f.forEach {
    pushPull(msg, 1)
  }
}

pull()
  RQ = randomPeer()
  RQ = RQ - R
  RQ.forEach {
    push(msg, 0)
  }
}

start()
  R = {}
  N = {}
  f = 3
  TTL = 3
  Δ = 10
  pushPull(msg)
  start(PERIODICPULL)
  start(pull)
  start()
  
```

