Grid Proof Programming Models

Jason Maassen
jason@cs.vu.nl

Before the break ...

- We looked at how Ibis makes Grids user friendly:
  - **JavaGAT** provides an easy-to-use API for the various flavours of Grid middleware
  - **Zorilla** provides a configuration free alternative to existing Grid middleware
  - **SmartSockets** provides an easy-to-use library that solves connectivity problems

Remaining problems

- However, grids are still:
  - **Heterogeneous**: many differences in hardware, performance, OS, libraries, etc.
  - **Faulty**: machines may be claimed by others, lose contact, or simply crash
  - **Malleable**: the set of available machines varies constantly
- Therefore we need ways to make applications **Grid Proof**

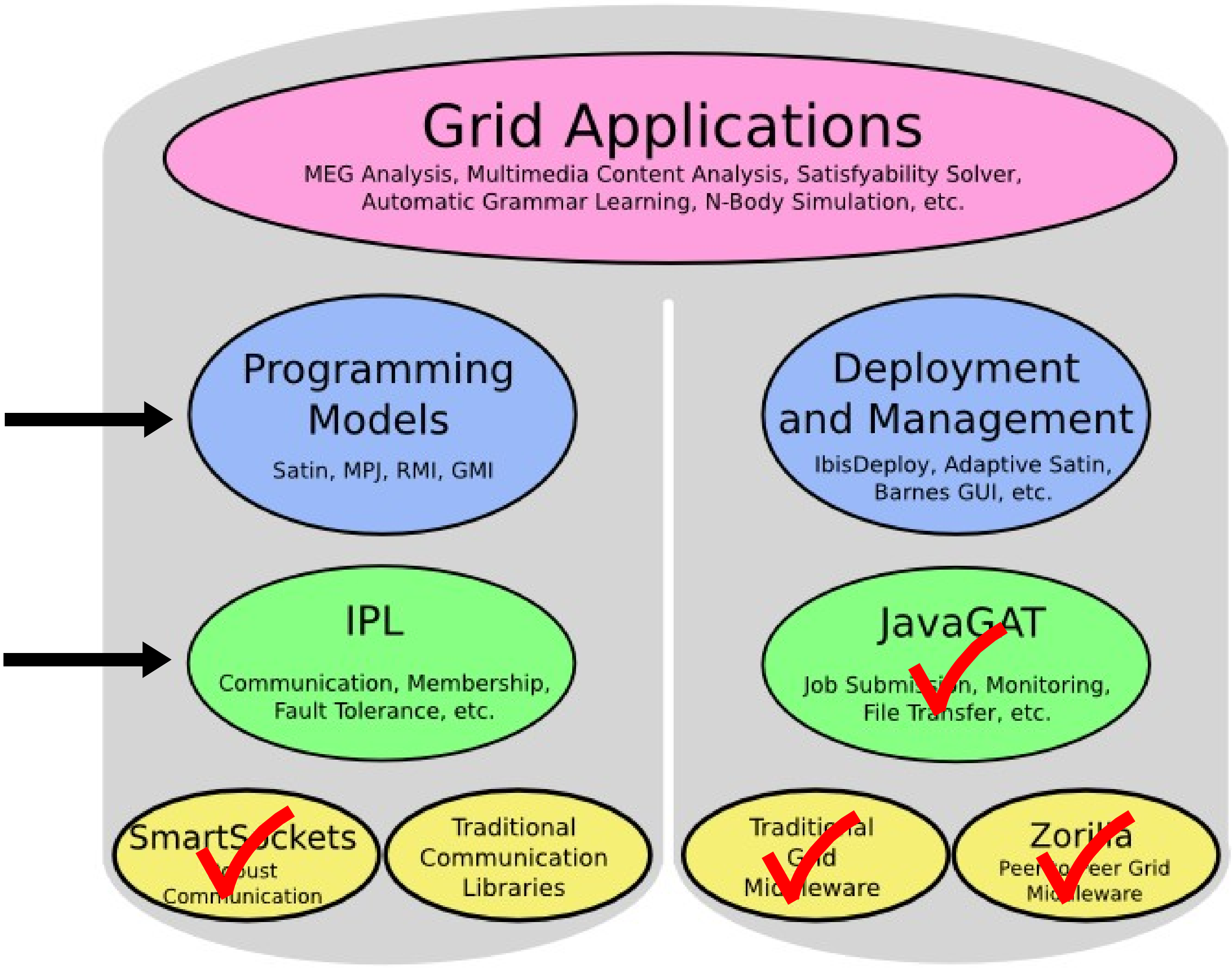
Grid Proof

- **Heterogeneity**:
  - Solved by using modern languages that run in a managed execution environment:
    - Safer and more portable
    - Easy to deploy and less dependencies
      - no compilation on grid sites
      - no libraries, headers, scripts, compilers, build tools...
  - **Java**, Python, Fortress, C#, ...
    - Language level virtualization
    - Alternative: system level virtualization

Grid Proof -- cont'd

- **Fault tolerance**:
  - We cannot prevent machines crashing
  - However, we can provide mechanisms to detects crashes ...
  - .. and use those to implement fault-tolerant programming models!
- **Malleability**:
  - Machines joining and leaving cleanly
  - Handled by same mechanism

Overview



Grid Applications

MEG Analysis, Multimedia Content Analysis, Satisfyability Solver, Automatic Grammar Learning, N-Body Simulation, etc.

Programming Models

Satin, MPJ, RMI, GMI

Deployment and Management

IbisDeploy, Adaptive Satin, Barnes GUI, etc.

IPL

Communication, Membership, Fault Tolerance, etc.

JavaGAT ✓

Job Submission, Monitoring, File Transfer, etc.

SmartSockets ✓

Robust Communication

Traditional Communication Libraries

Traditional Grid Middleware ✓

Zorilla ✓

Peer-to-Peer Grid Middleware

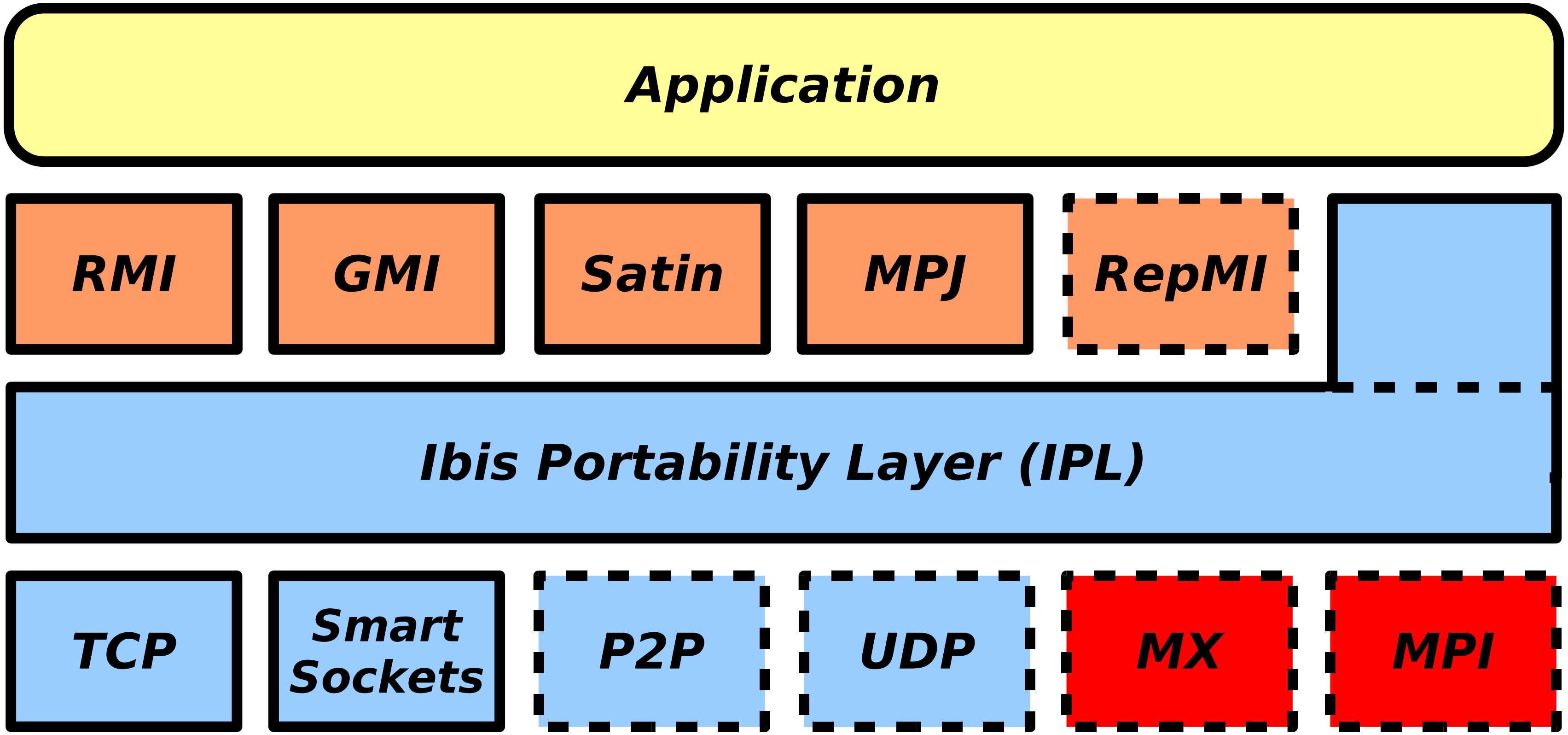
Ibis Portability Layer (IPL)

- Simple API for Grid Communication
  - Flexible communication model
    - connection oriented messaging
    - abstract addressing scheme
  - Malleability/Fault Tolerance
    - notifications when machines join/leave
    - open & closed world (not just SMPD)
  - Serialization
    - send bytes, doubles, objects, etc.

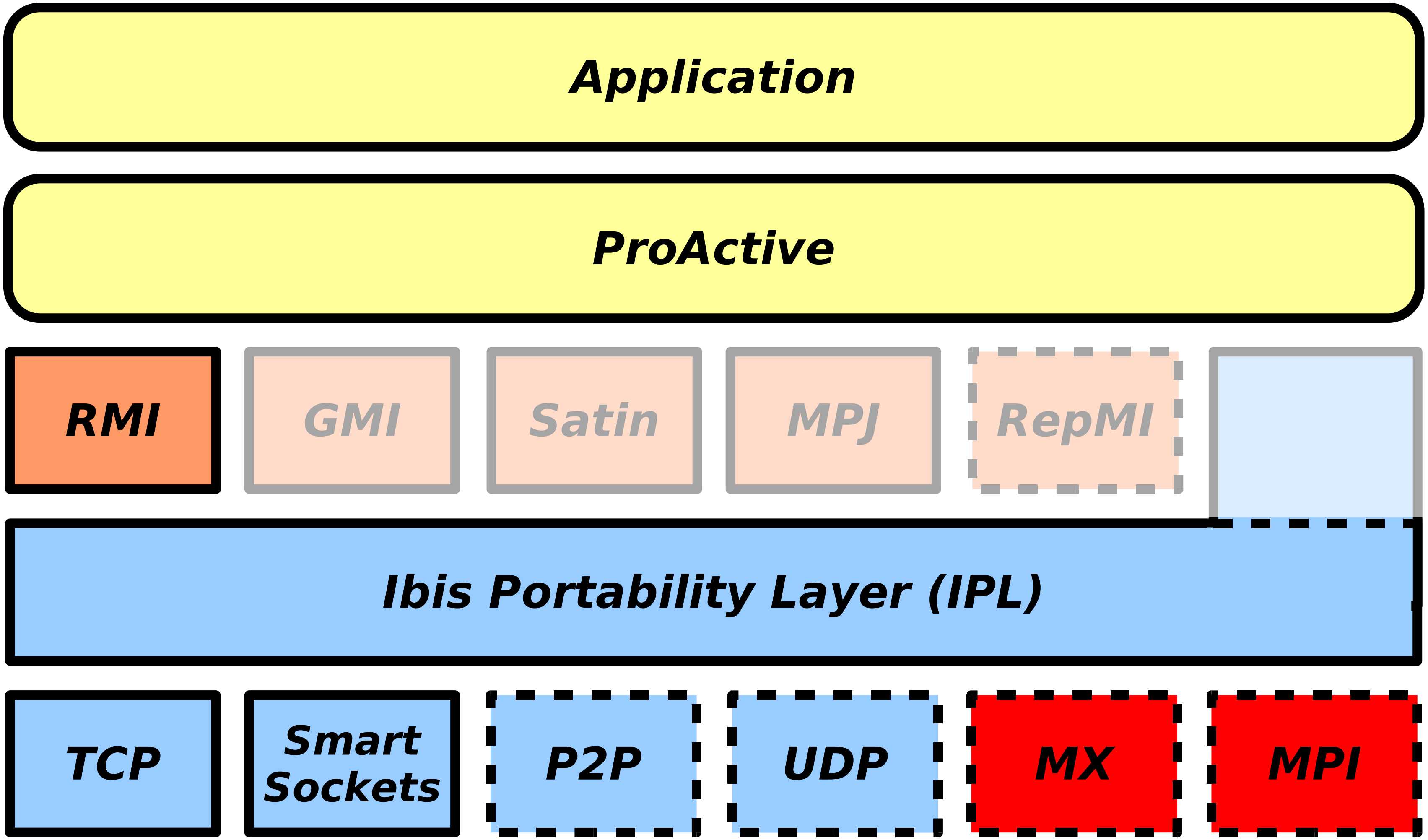
IPL

- Clean & abstract API
  - hides network specific details
    - hostnames, IP addresses, MPI ranks, etc.
  - easy to implement on TCP, UDP, MPI, MX...
- Hides network peculiarities
  - Results in more portable applications
  - Suitable for Grids

IPL and Friends

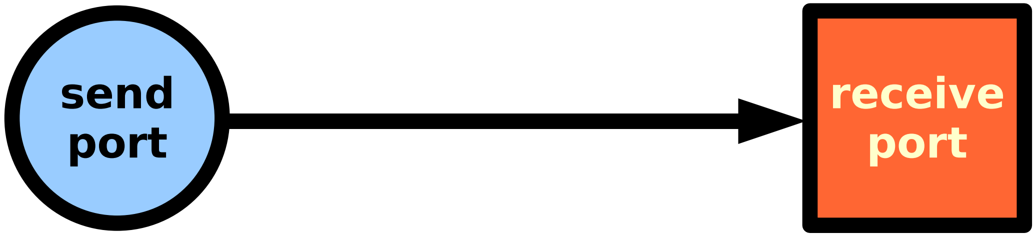


IPL and Friends



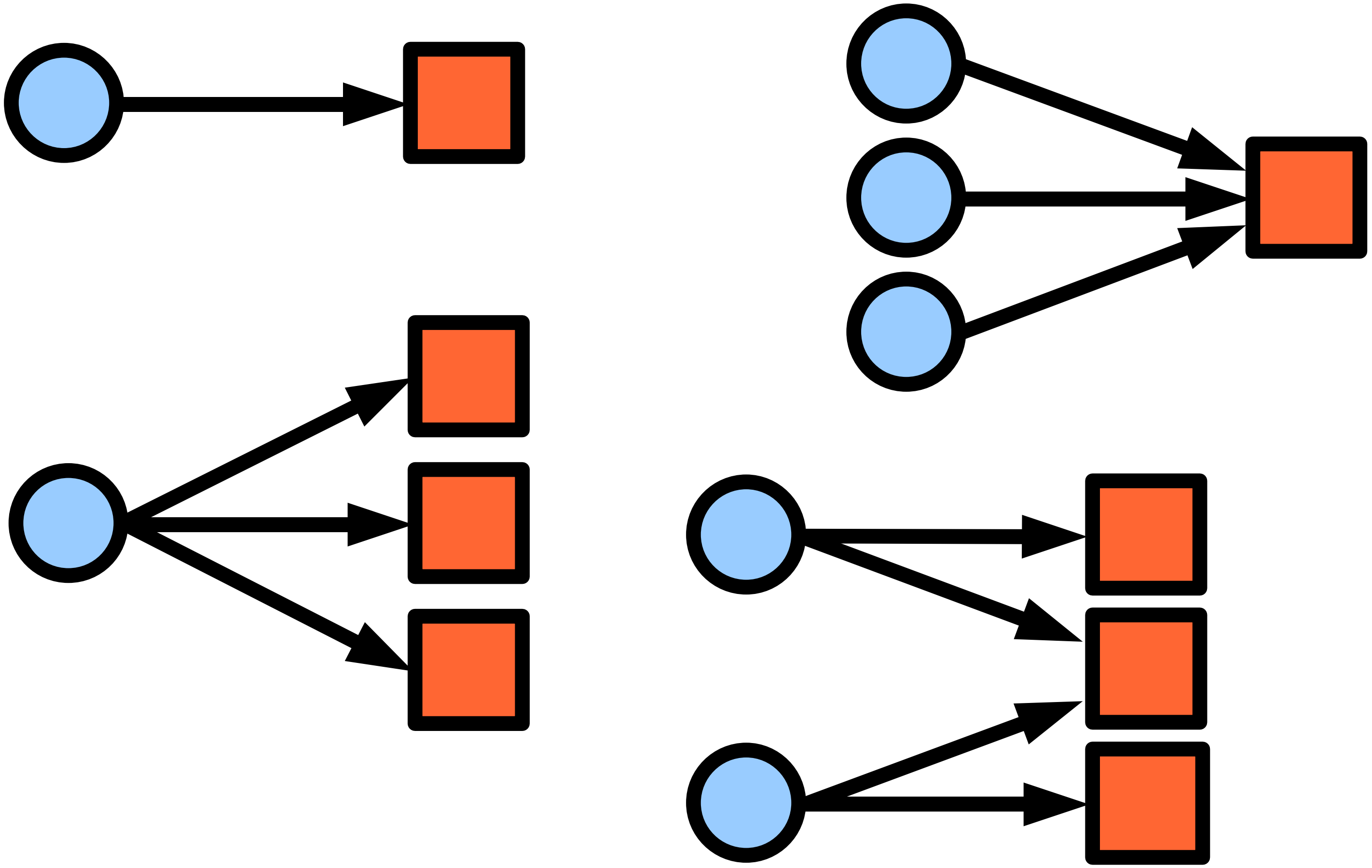
Communication

- 'Low-level' communication model
- Unidirectional pipes
- Two end points
- Connection oriented (allows streaming)



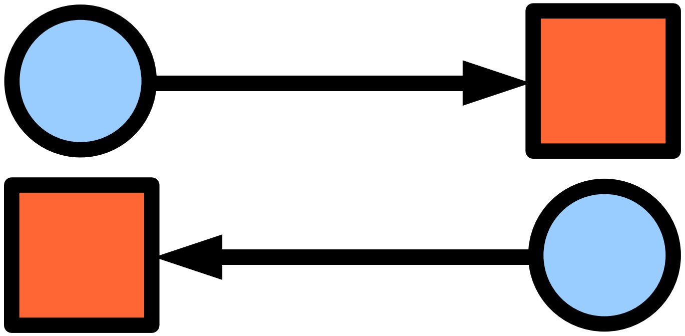
Send & receive ports

- Can be connected in arbitrary ways



Send & receive ports

- Simplicity may cause some overhead...
  - Example: need two pairs for RPC / RMI

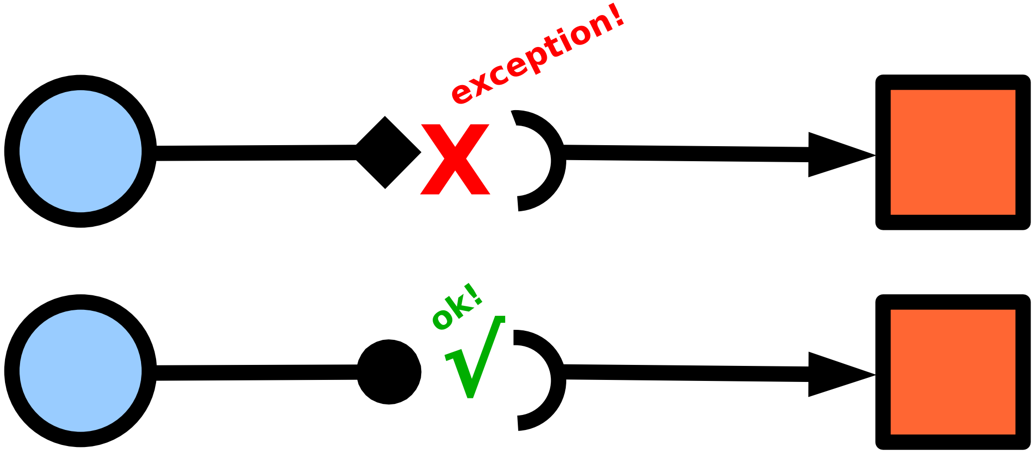


Port Types

- All ports have a **type**
  - Consists of a set of required capabilities:
    - Connection patterns
      - Unicast, many-to-one, one-to-many, many-to-many.
    - Communication properties:
      - Fifo ordering, numbering, reliability.
    - Serialization properties:
      - bytes, data, object
    - Message delivery:
      - Explicit receipt, automatic upcalls, polling

Port Types

- Defined at runtime
  - Specify set of capabilities
- Types must match when connecting!



Port Types

- Forces programmer to specify how each communication channel is used
    - Prevents bugs
        - Exception when contract is breached
    - Allows efficient impl. to be selected
        - Unicast only ?
        - Bytes only ?
        - Can save a lot complexity!

IbisIdentifiers

- In a parallel/distributed application
  - Each process has an Ibis instance
  - Each instance has an **IbisIdentifier**

- IbisIdentifier:
  - Uniquely identifies an Ibis instance
  - Abstracts away from the implementation
    - e.g. hostnames, IP addresses, MPI-ranks, etc.
  - Makes your application a bit more portable

Connection setup WEG ?

- Two options:
  - 1) Using a IbisIdentifier and a name
    - Name specifies the receiveport
      - Unique per Ibis instance
    - Human-readable (usually)
  - 2) Using a ReceivePortIdentifier
    - Uniquely identifies a receiveport
    - Created when ReceivePort is created
    - Can be passed around between Ibis instances.

Connection setup (1)

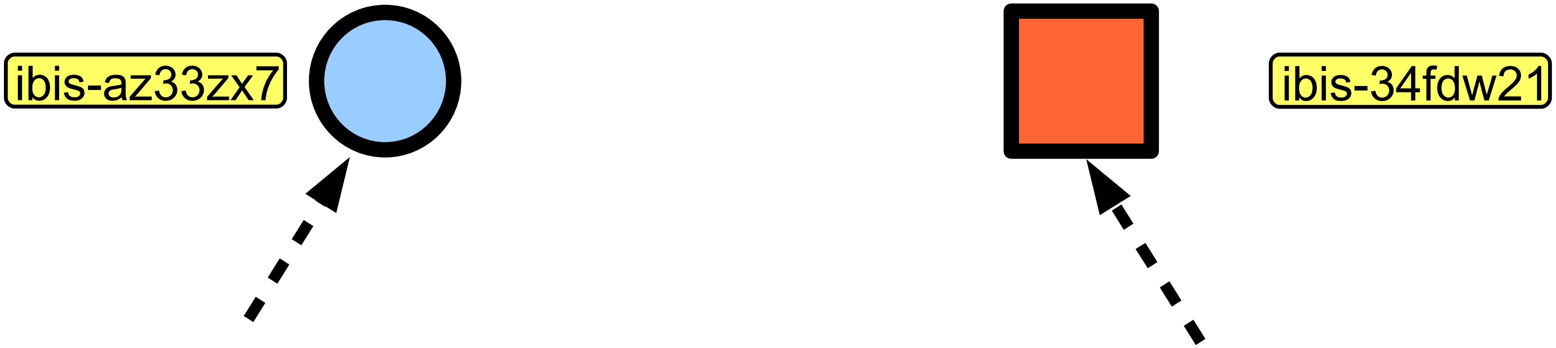
ibis-az33zx7

ibis-34fdw21

Create Ibis

Create Ibis

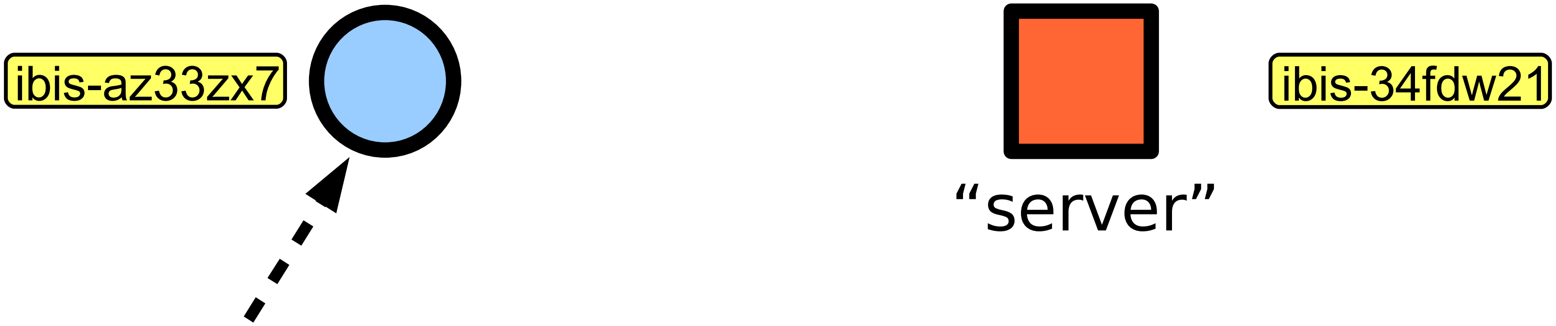
Connection setup (1)



Create SendPort

Create ReceivePort
“server”

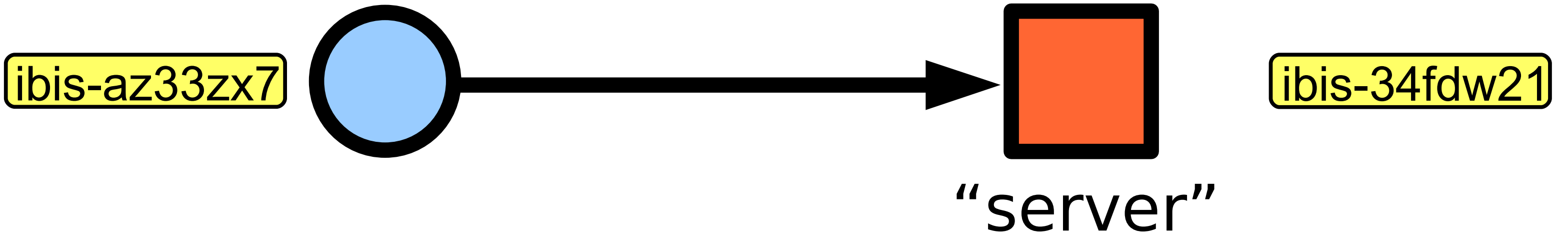
Connection setup (1)



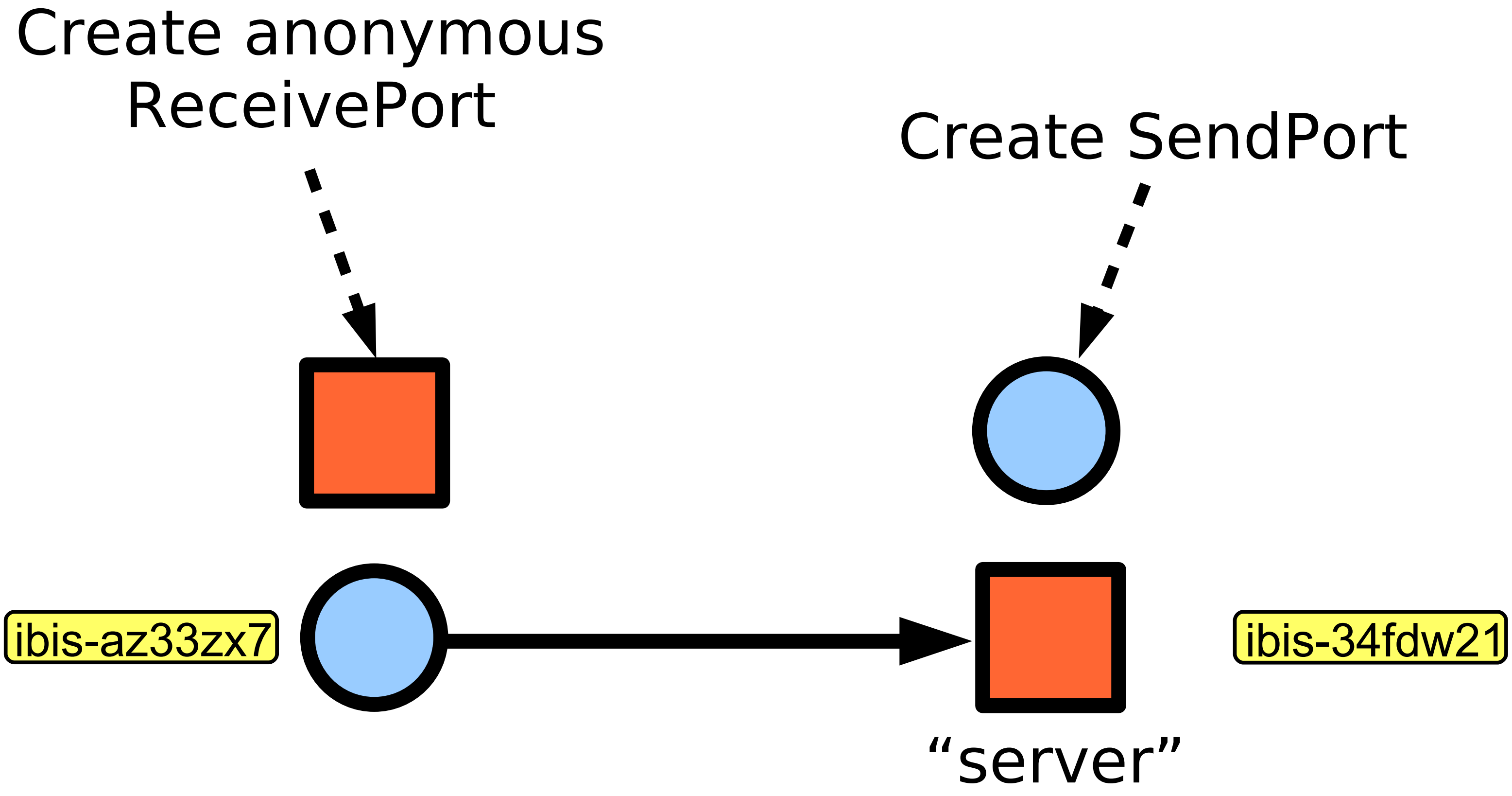
connect(ibis-34fdw21 , “server”)

(How do you get this ? Explained later!)

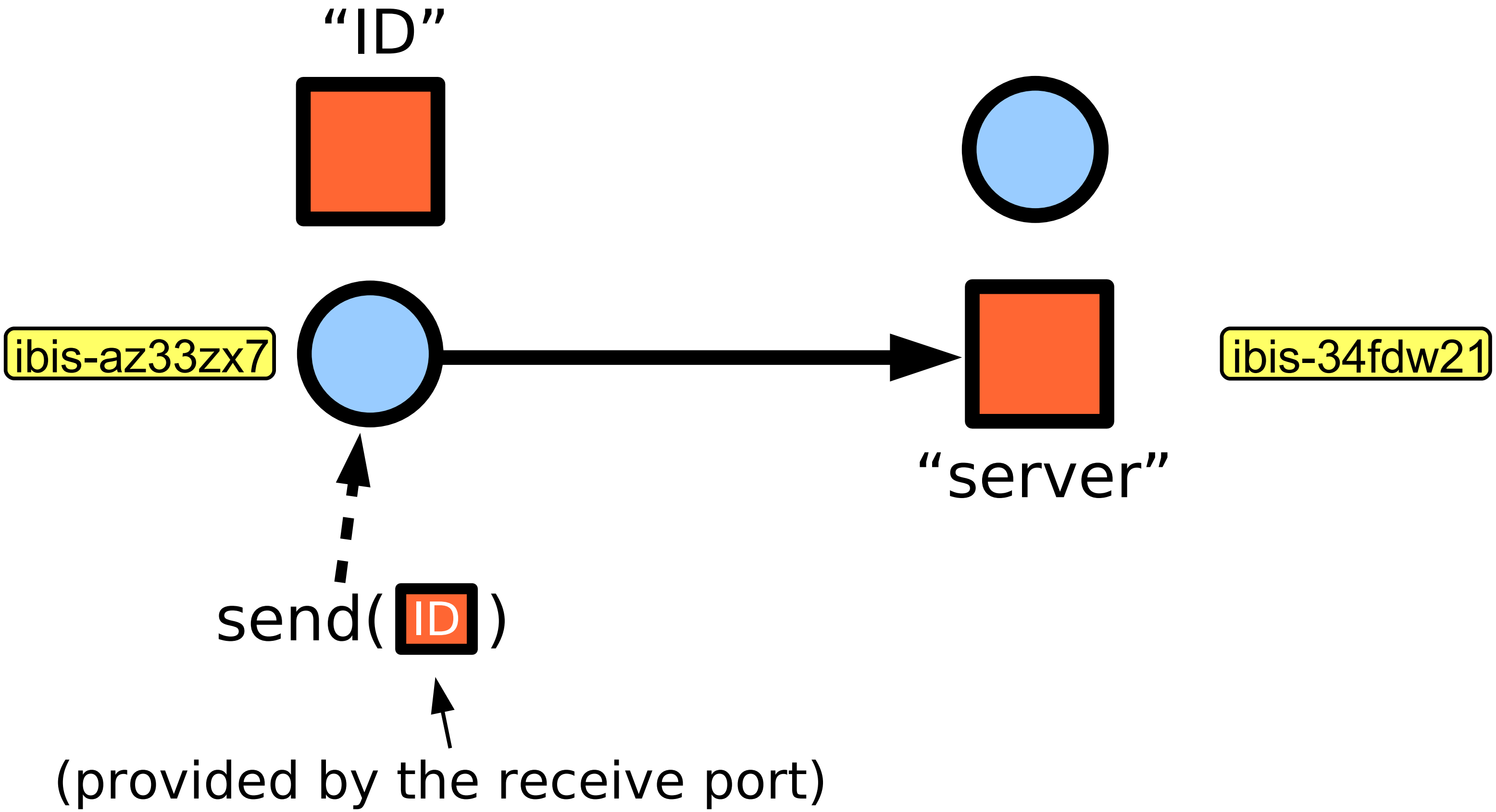
Connection setup (1)



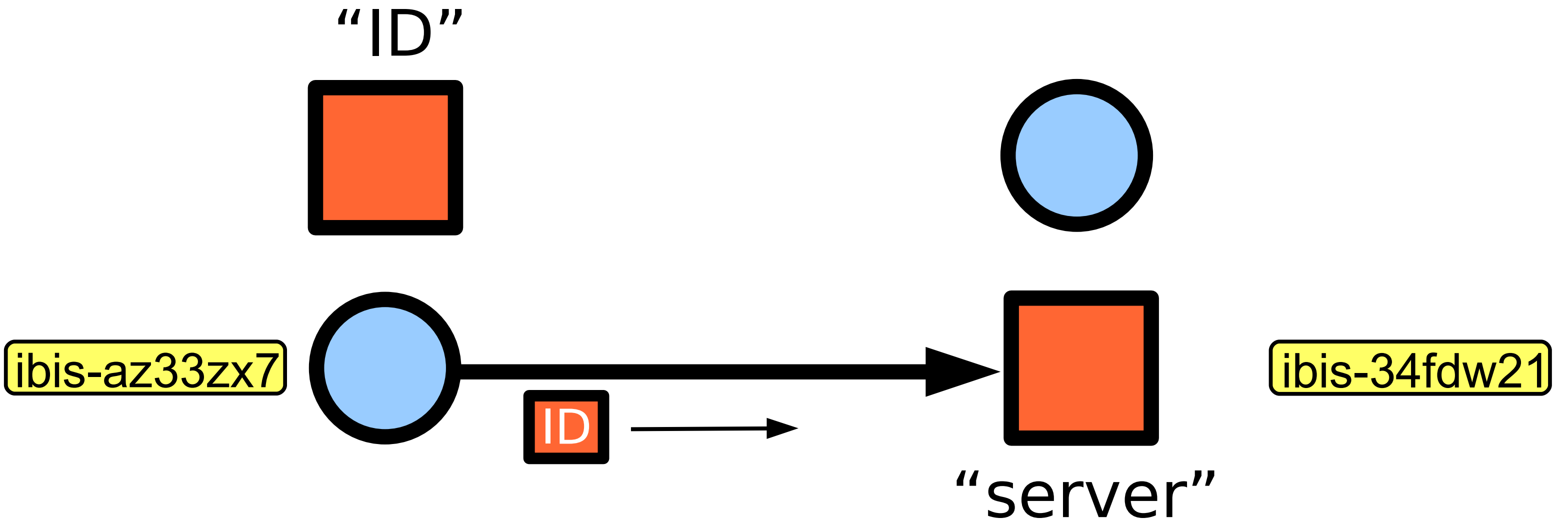
Connection setup (2)



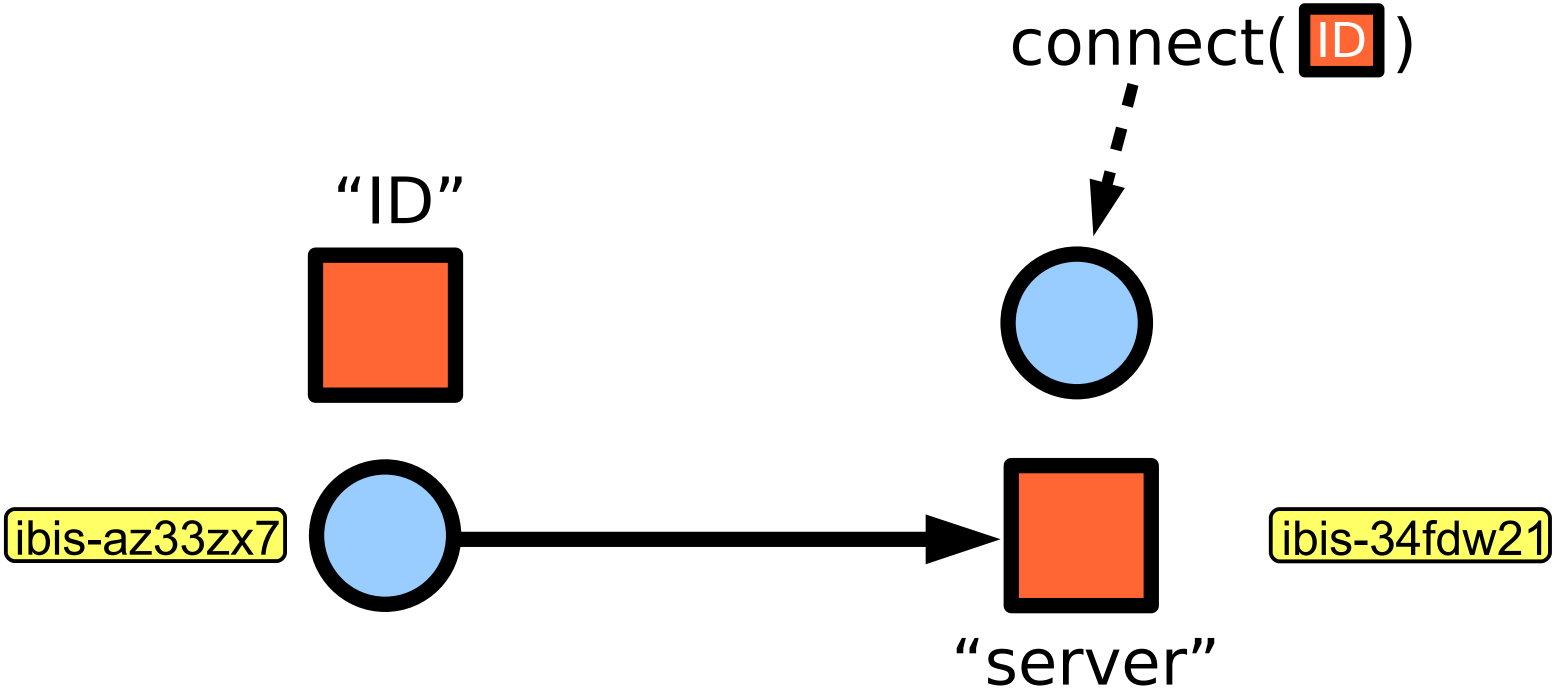
Connection setup (2)



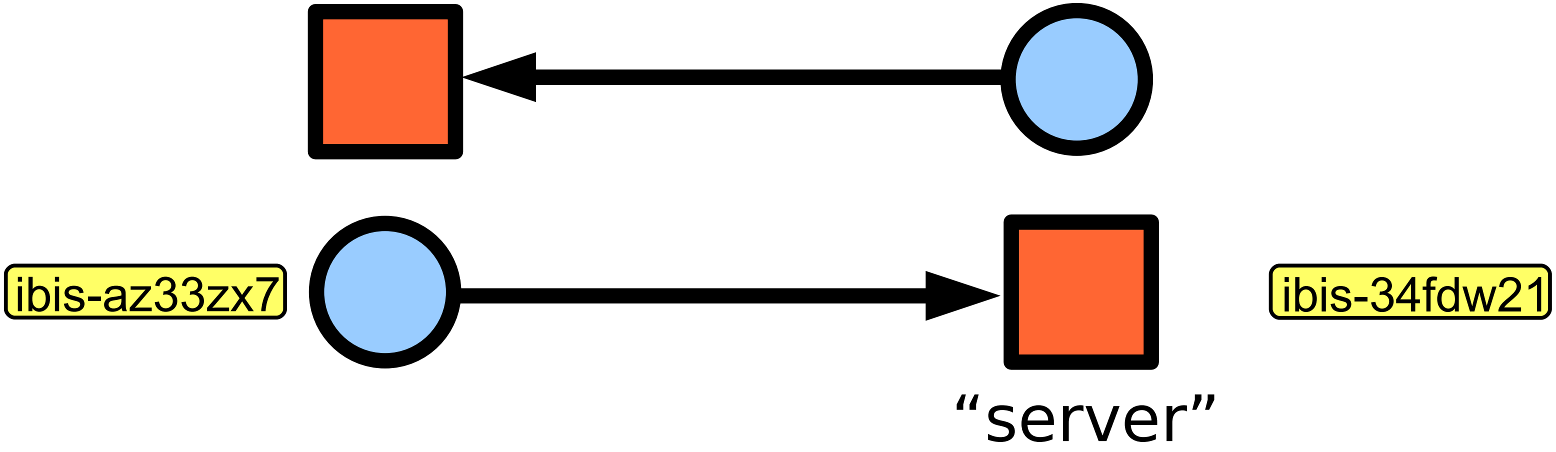
Connection setup (2)



Connection setup (2)



Connection setup (2)



Messages

- Ports communicate using 'messages'

- Contain read or write methods for
  - Primitive types (byte, int, ...)
  - Object
  - Arrays slices (partial write / read in place)

- Unlimited message size

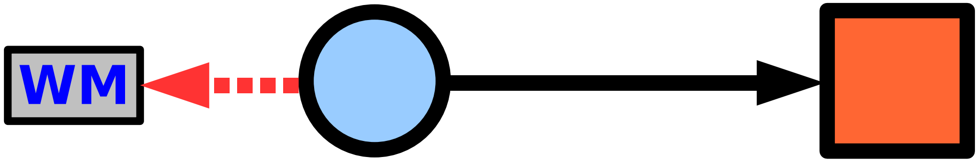
Ibis Serialization

- Based on bytecode-rewriting
  - Adds serialization and deserialization code to serializable types
  - Prevents reflection overhead during (de-)serialization
  - Has fallback mechanism for non-rewritten classes

- Experimented with runtime rewriting

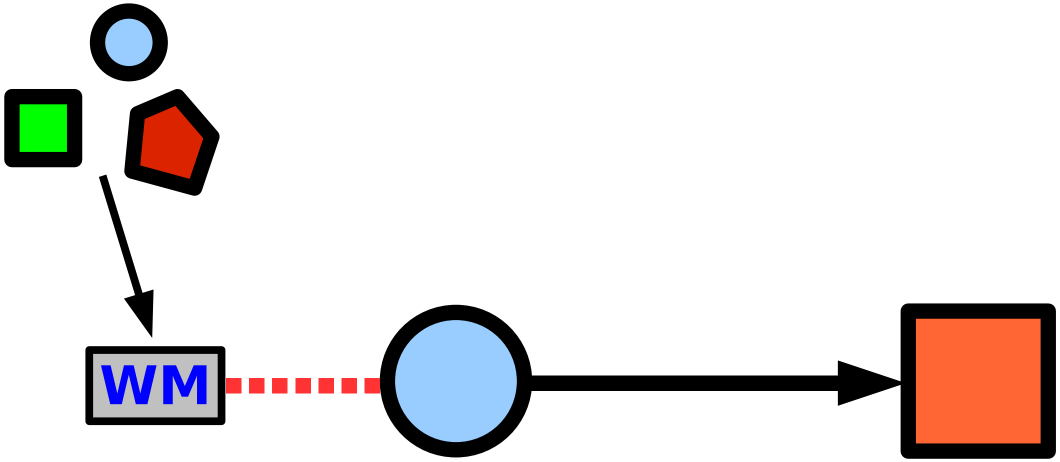
Messages

- Get WriteMessage from SendPort



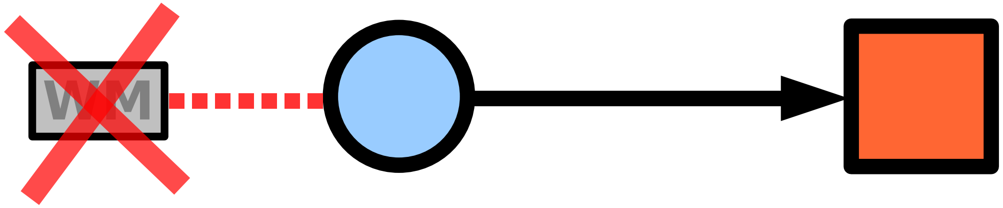
Messages

- Write data into WriteMessage



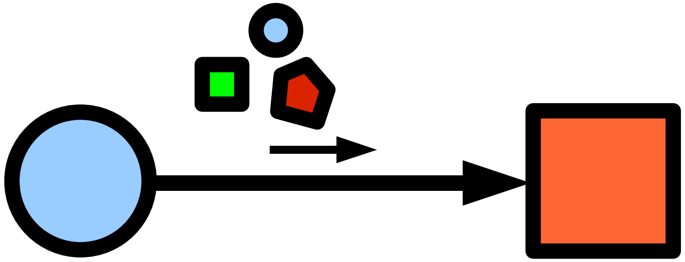
Messages

- Finish the WriteMessage



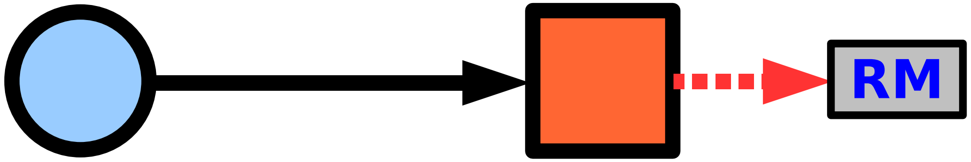
Messages

- Data is send to ReceivePort



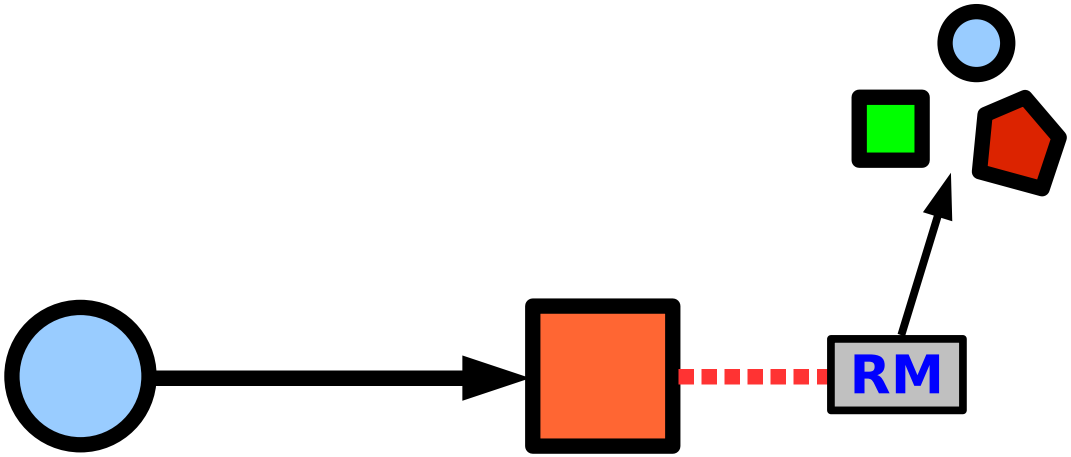
Messages

- ReceivePort produces ReadMessage
  - Explicit receive or callback (upcall)



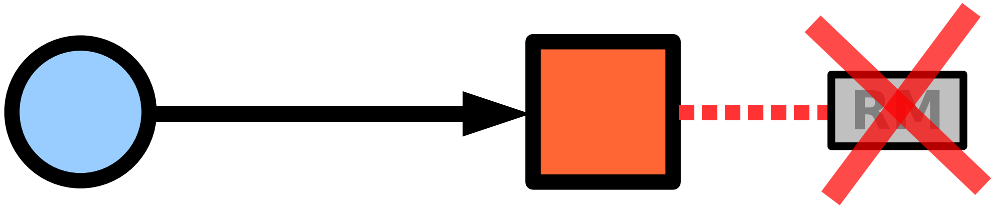
Messages

- Read data from ReadMessage



Messages

- Finish the ReadMessage

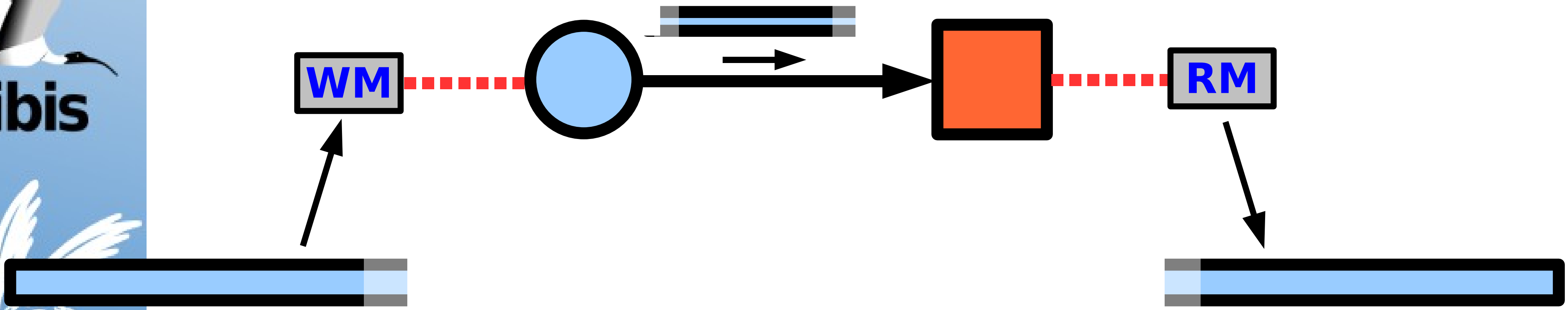


Messages

- Done!

Messages or streams ?

- Message size is unlimited
  - Data may be forwarded at any time
  - Both S. & R. messages alive at same time
  - There's streaming!

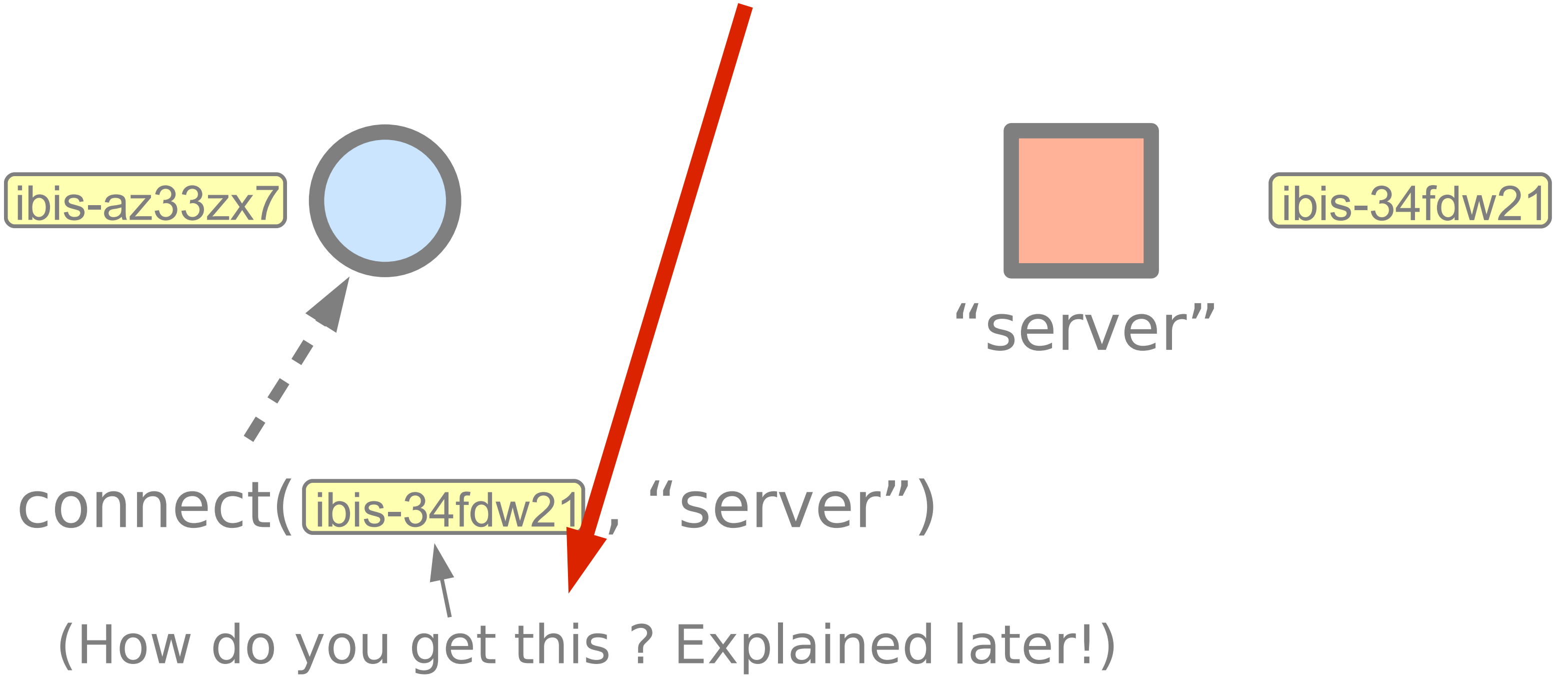


Short Recap

- First create PortType
- PortType creates Send & ReceivePort
  - Type is checked when connecting
- Several ways to connect
  - Abstact addressing
- Use Messages to communicate
  - Allows streaming
  - 3/4 types of serialization

Connection setup (1)

Remember this question ?



ibis-az33zx7

ibis-34fdw21

“server”

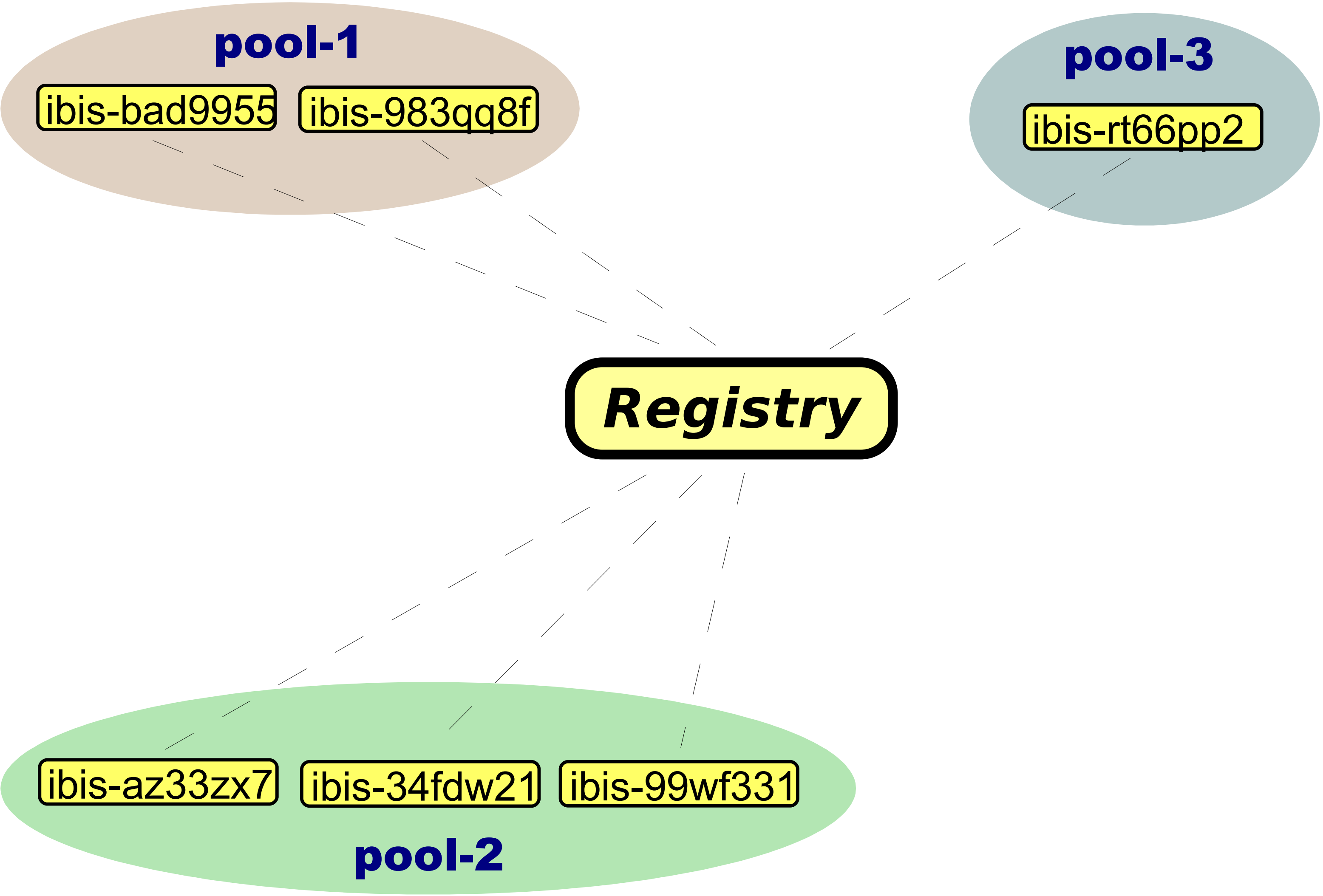
connect(ibis-34fdw21, “server”)

(How do you get this ? Explained later!)

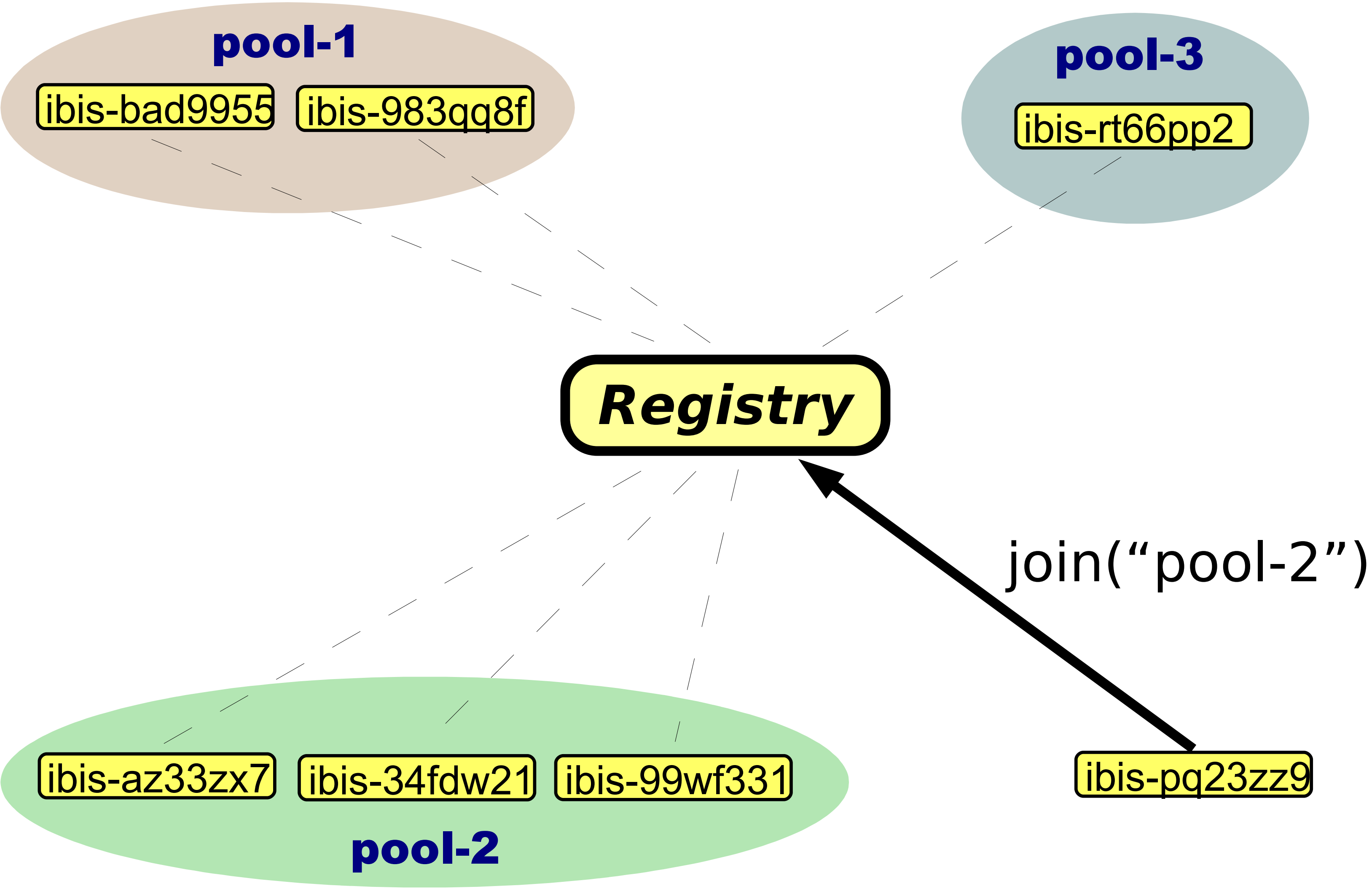
Join Elect Leave (JEL)

- Membership information
  - Can subscribe to information
    - Updates when Ibis instances join or leave
  - Useful for determining who's participating
    - Also used for fault-tolerance
- Ibis instances are part of a **pool**
  - Either variable size or fixed (create-once)
    - Fixed used by 'legacy' MPI-type applications

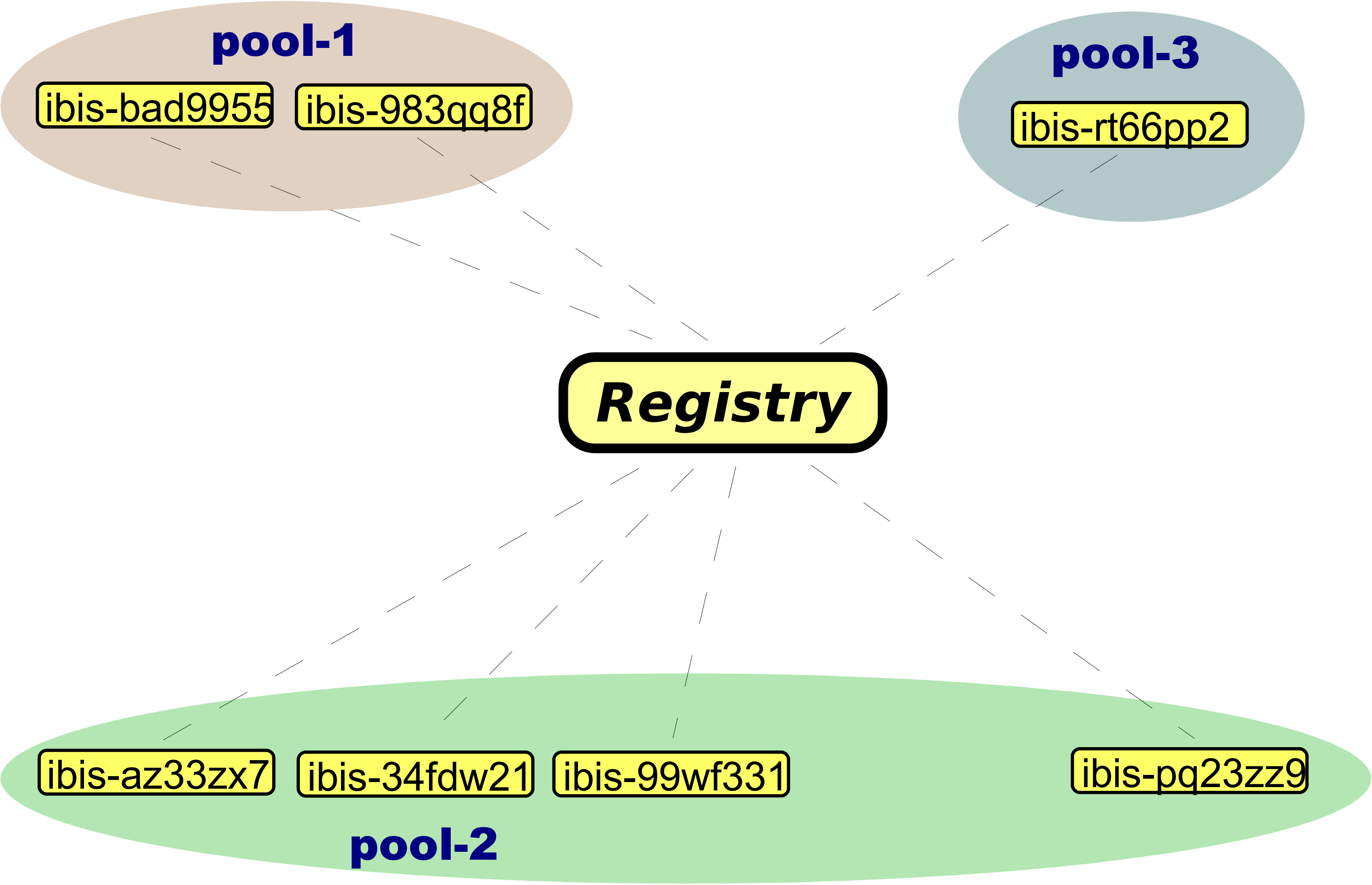
Pools & Malleability



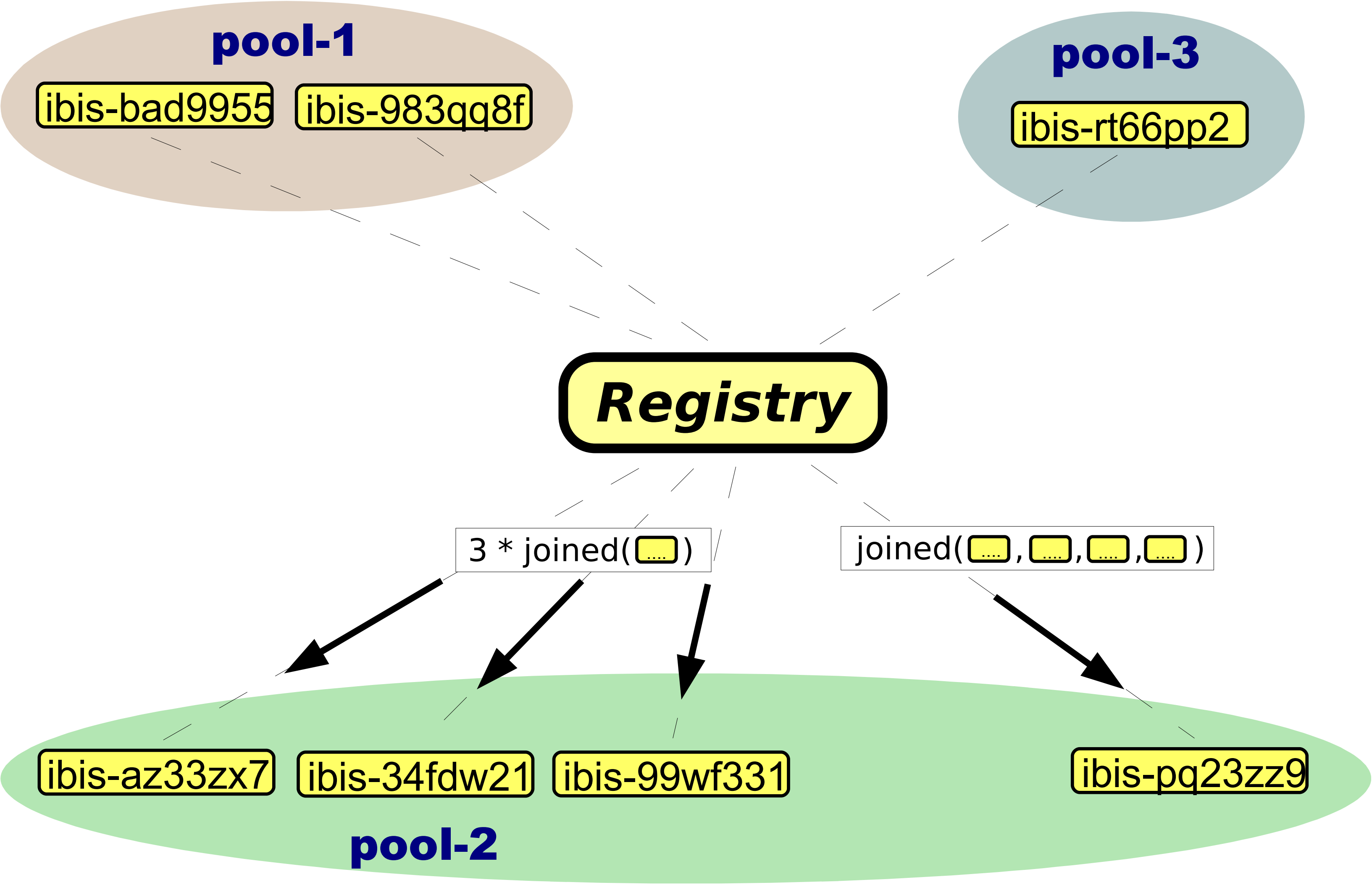
Pools & Malleability



Pools & Malleability



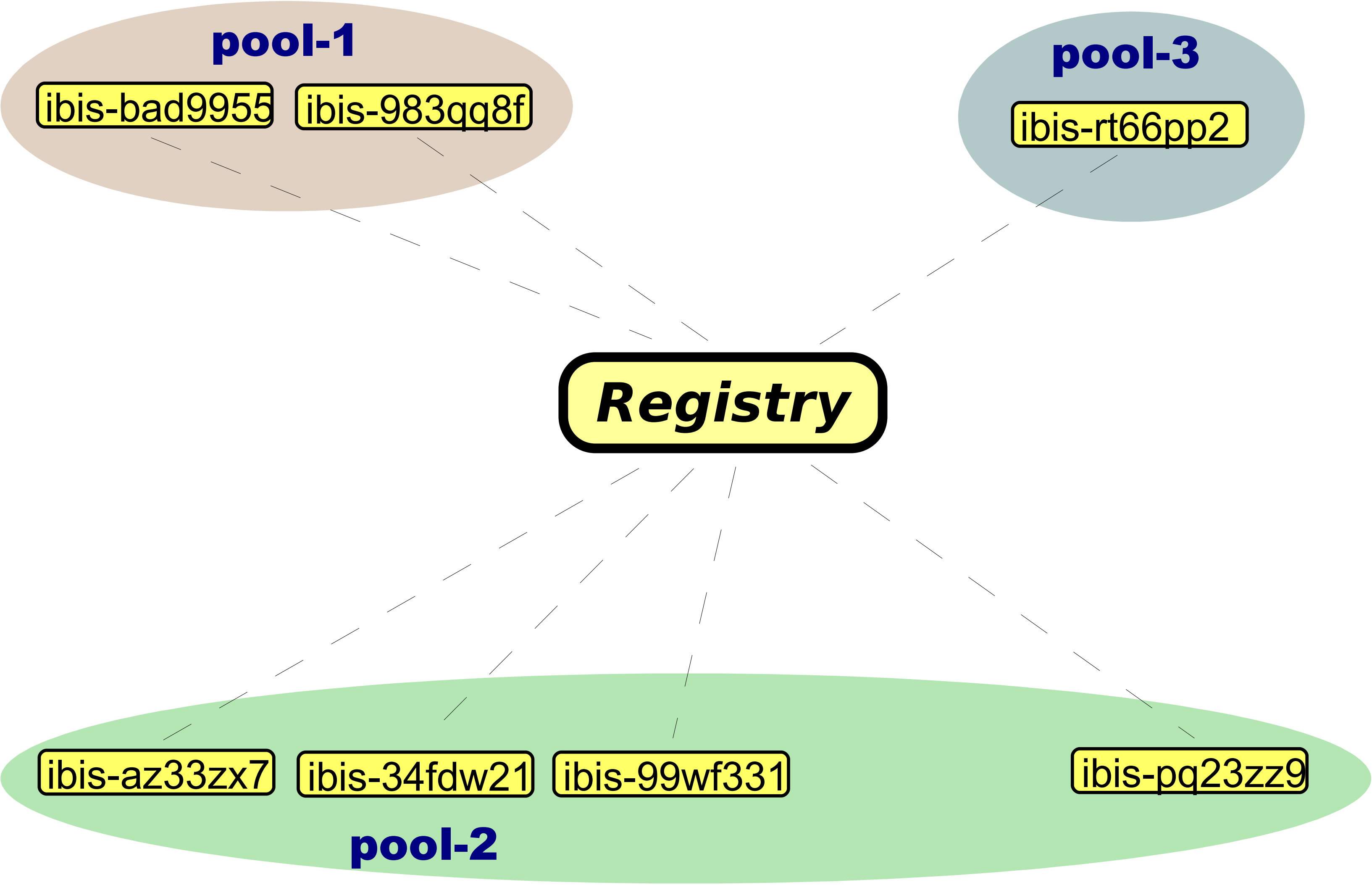
Pools & Malleability



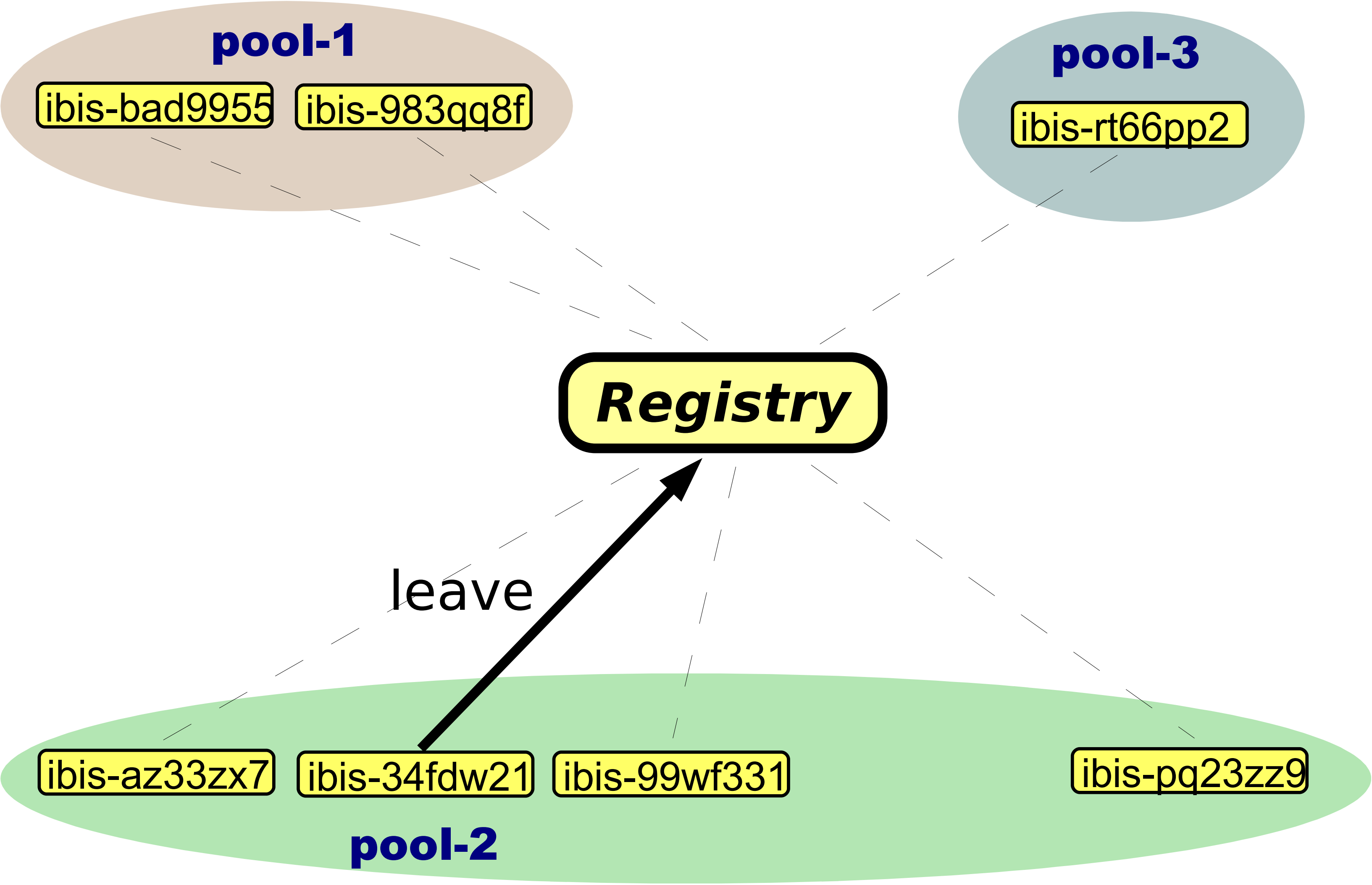
ibis

vrije Universiteit

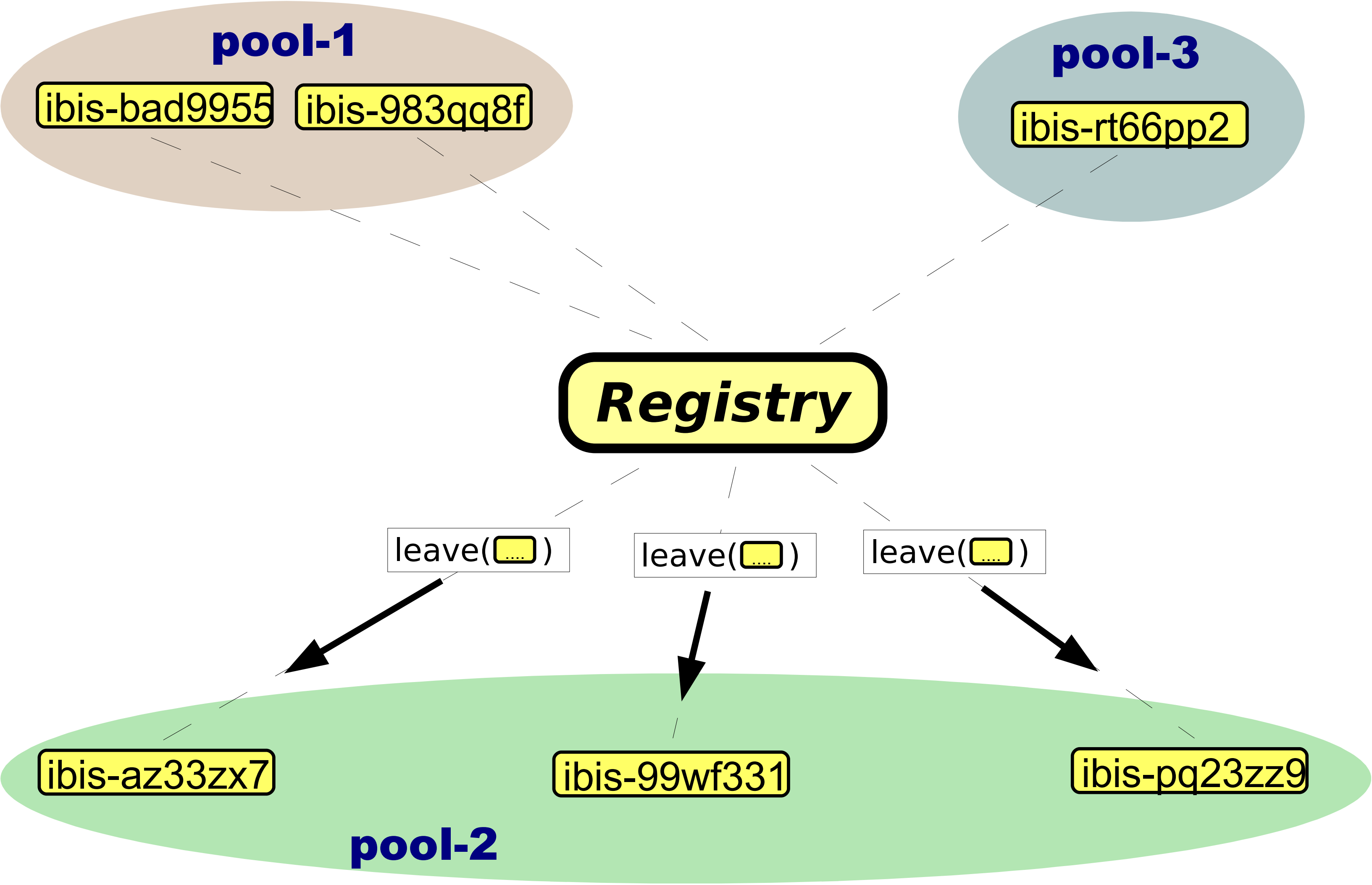
Pools & Malleability



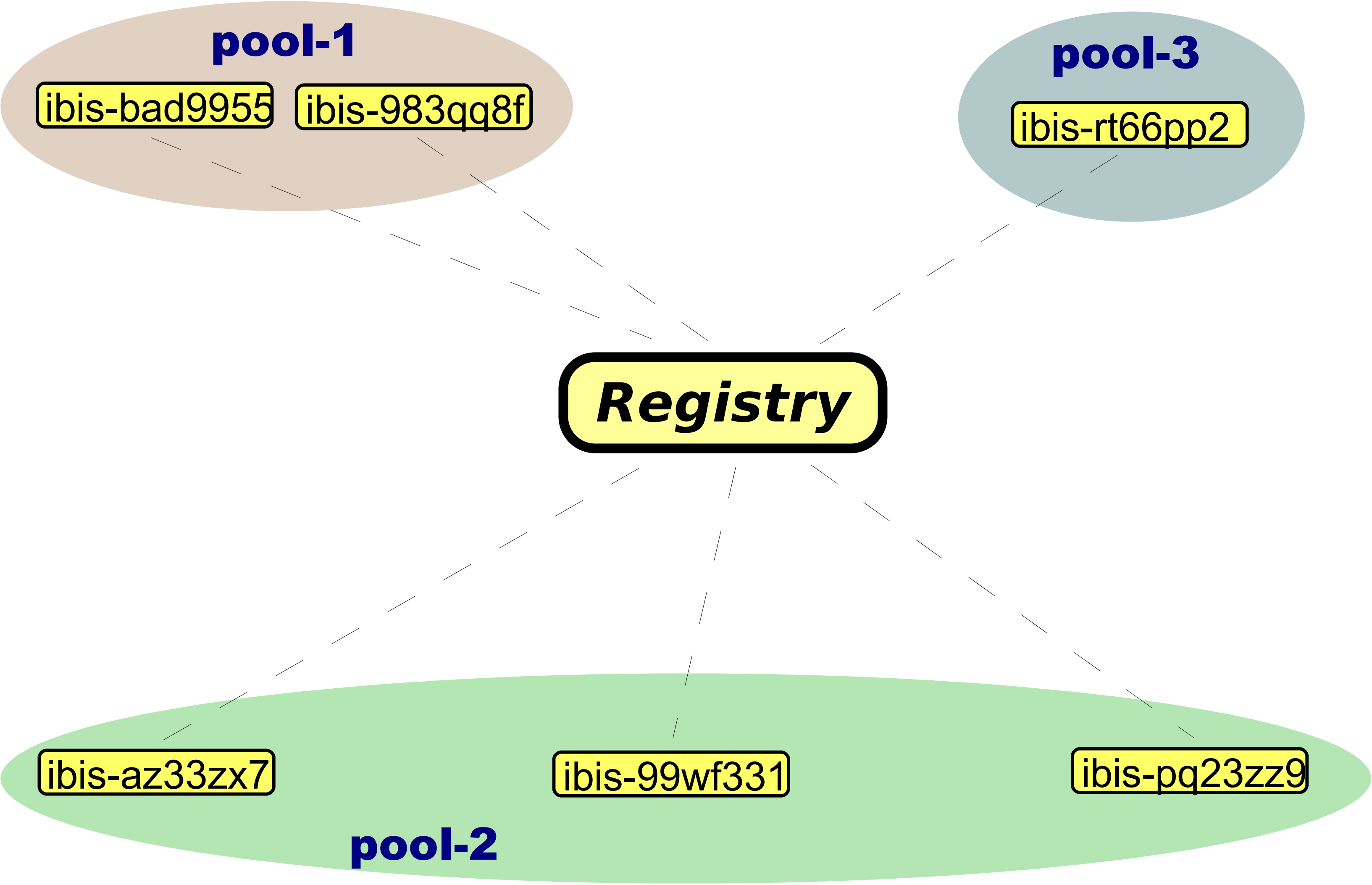
Pools & Malleability



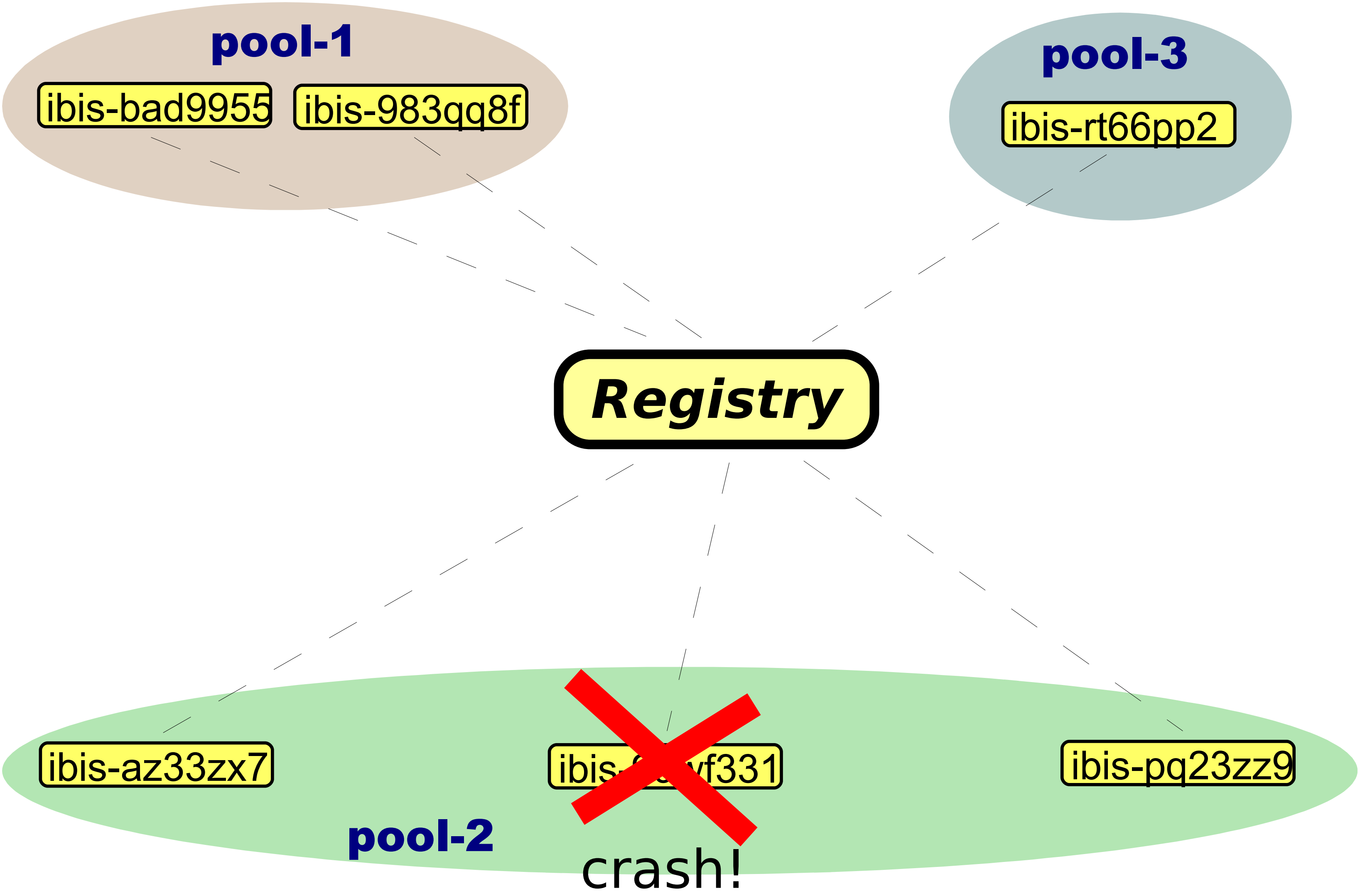
Pools & Malleability



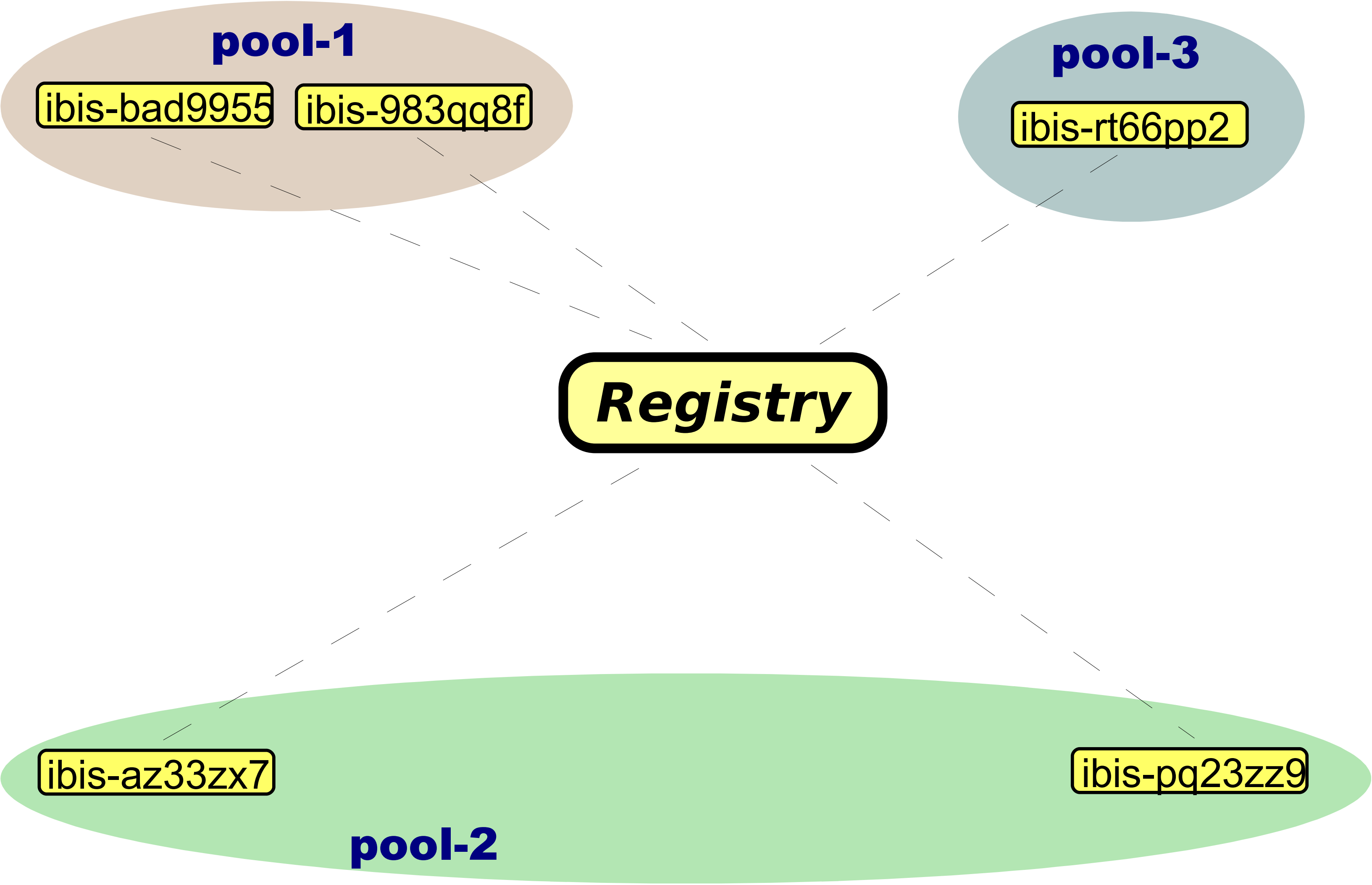
Pools & Malleability



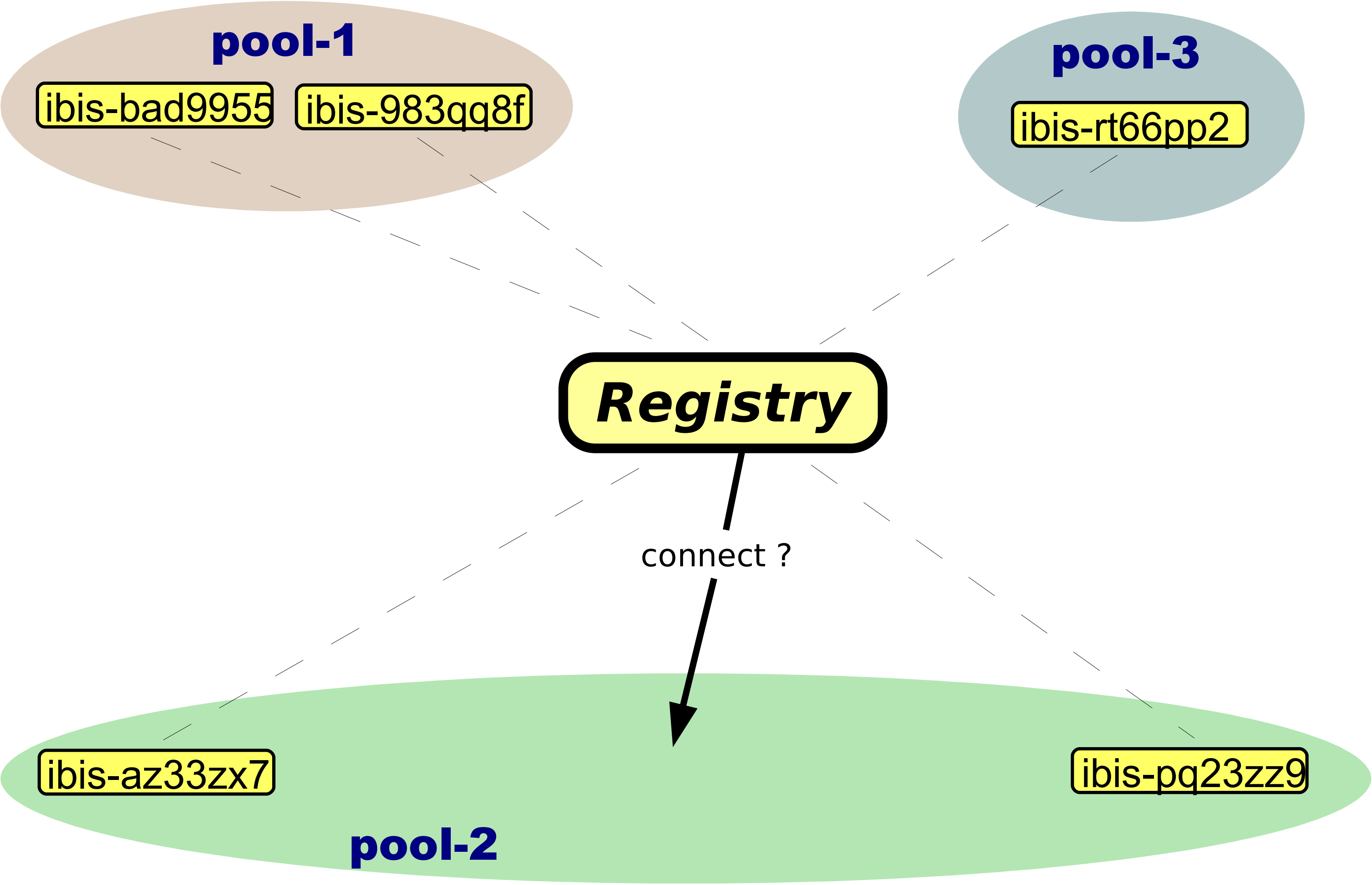
Pools & Malleability



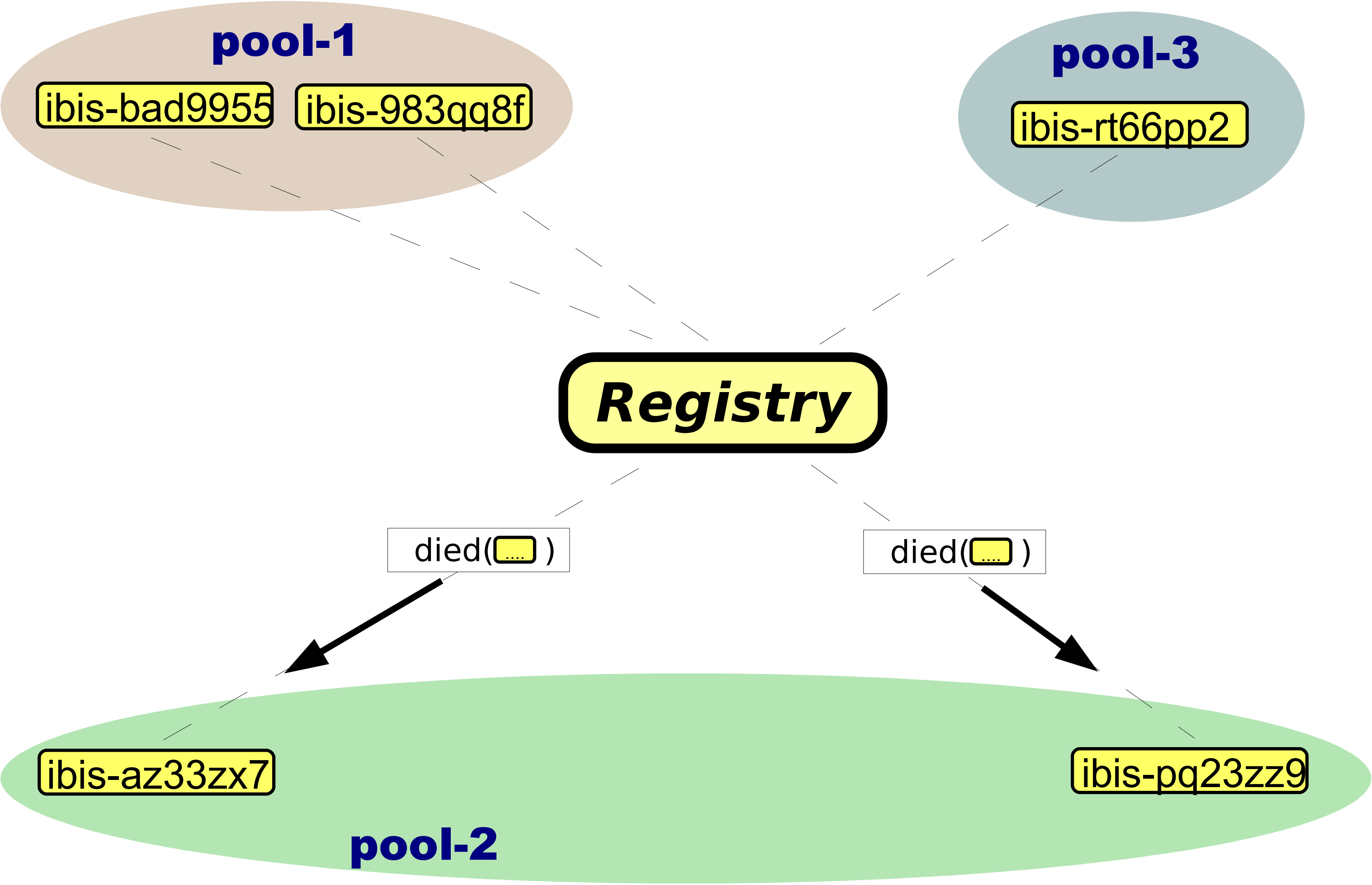
Pools & Malleability



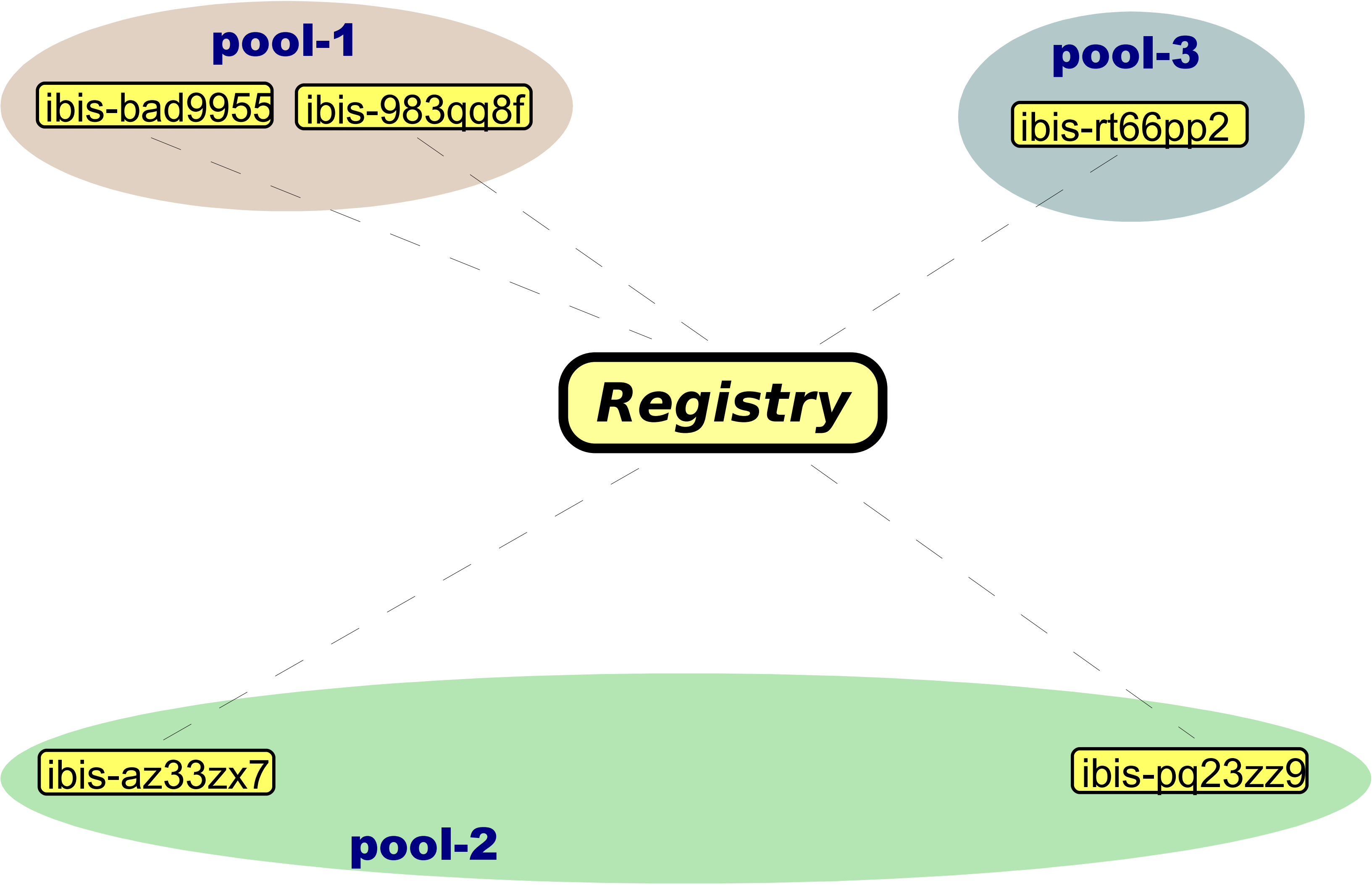
Pools & Malleability



Pools & Malleability



Pools & Malleability



Elections

- Registry offers an 'election' mechanism
  - Allows a group to determine who's **special**

- Each election
  - Has a name (String)
  - Produces IbisIdentifier of the winner
  - Is not democratic
  - You can also be 'an observer'

Registry

- Example shows centralized version
  - also have broadcast tree and gossiping implementations (improve scalability)
- You can select the functionality and consistency that is needed
  - reducing functionality or consistency further improves scalability

Summary

- IPL offers an abstract model
  - Connection oriented message passing
  - Hides network details (for portability)
- Supports fault tolerance / malleability
  - **No application-level fault tolerance!**
  - Only offers the means to implement this!
- Higher level models (Satin) do offer this at application level