The road to the remote debugger

Remote debugger behind the scenes

The road to the remote debugger

TostSerializer

Basys

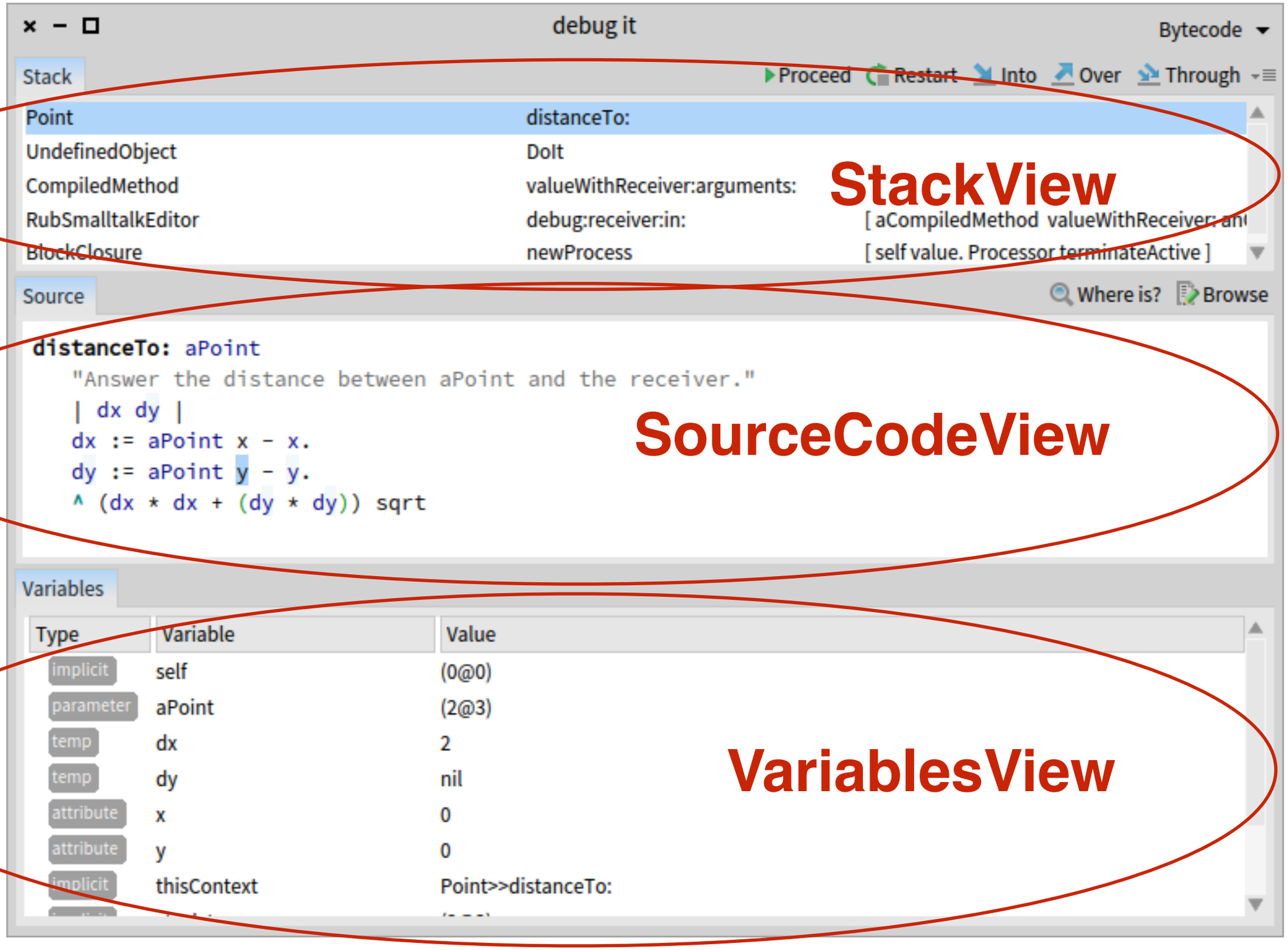
Seamless

ObjectTravel

ObjectStatistics

GT Extensions

GTInspector
GTDebugger


debug it
Bytecode
Stack
Proceed
Restart
Into
Over
Through
Point
distanceTo:
UndefinedObject
DoIt
CompiledMethod
valueWithReceiver:arguments:
RubSmalltalkEditor
debug:receiver:in:
[aCompiledMethod valueWithReceiver: anI
BlockClosure
newProcess
[self value. Processor terminateActive]
StackView
Source
Where is?
Browse
distanceTo: aPoint
"Answer the distance between aPoint and the receiver."
| dx dy |
dx := aPoint x - x.
dy := aPoint y - y.
^ (dx * dx + (dy * dy)) sqrt
SourceCodeView
Variables
Type
Variable
Value
implicit
self
(0@0)
parameter
aPoint
(2@3)
temp
dx
2
temp
dy
nil
attribute
x
0
attribute
y
0
implicit
thisContext
Point>>distanceTo:
VariablesView

Process suspendedContext
Exception signalerContext

Stack
Sender context 1
Sender context 2
Sender context 3
...
Root context

Each context
receiver
selector
arguments
temps
method

class
instVarNames

methodClass
temp names
arg names
source code

Each object
printString

Demo for slow remote debugger

Seamless

- New Remote Smalltalk implementation
- Started at 2012 by Nikolaos Papoulias
- Redesigned this year
    - http://smalltalkhub.com/#!/~Pharo/Seamless

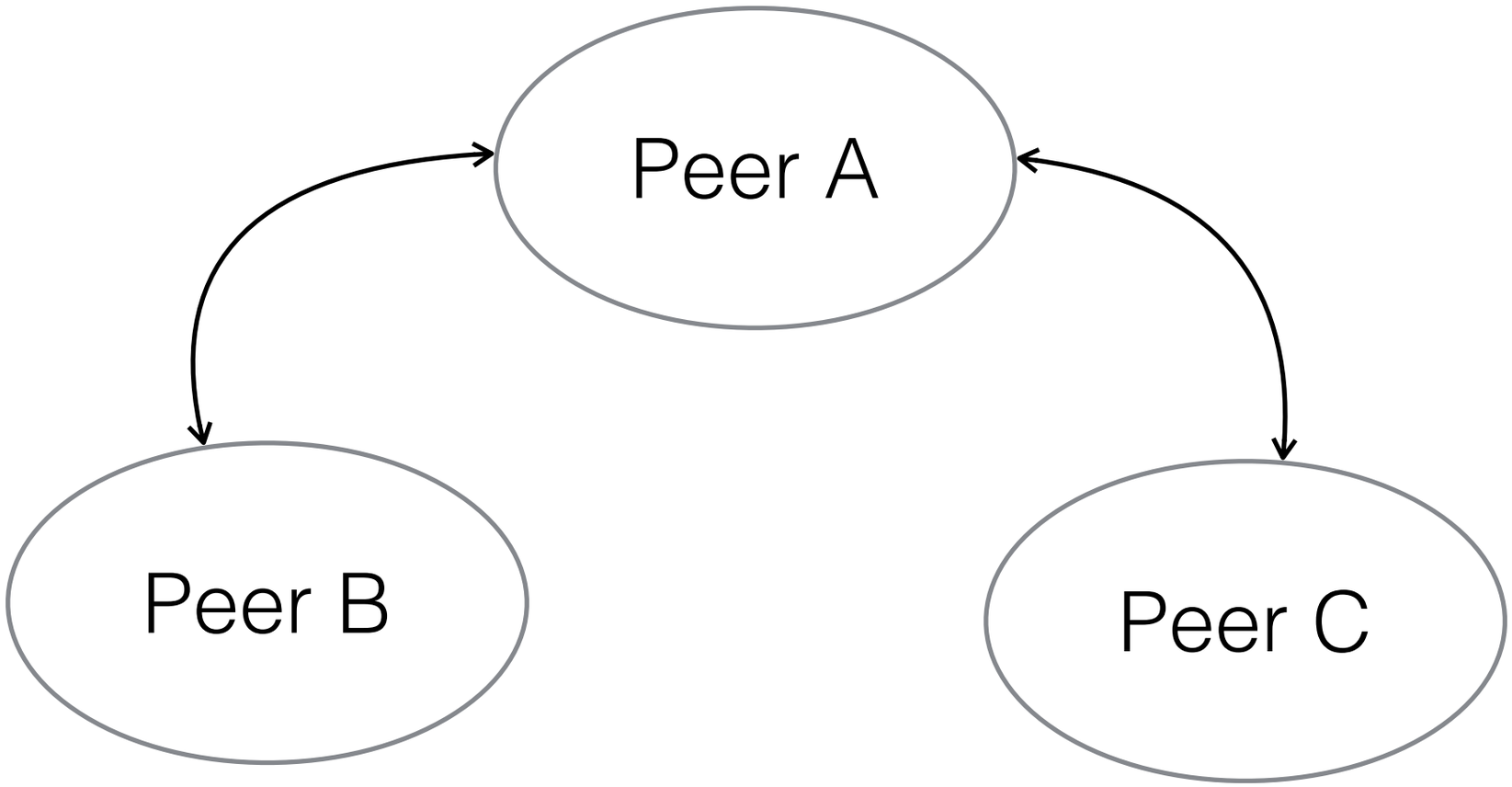
Asynchronous network

- Simultaneous sending and receiving data
  - Data sending not depends on data receiving
- Asynchronous processing of received data
  - Data processing not blocks new incoming data
  - Every received data is processed in separate thread

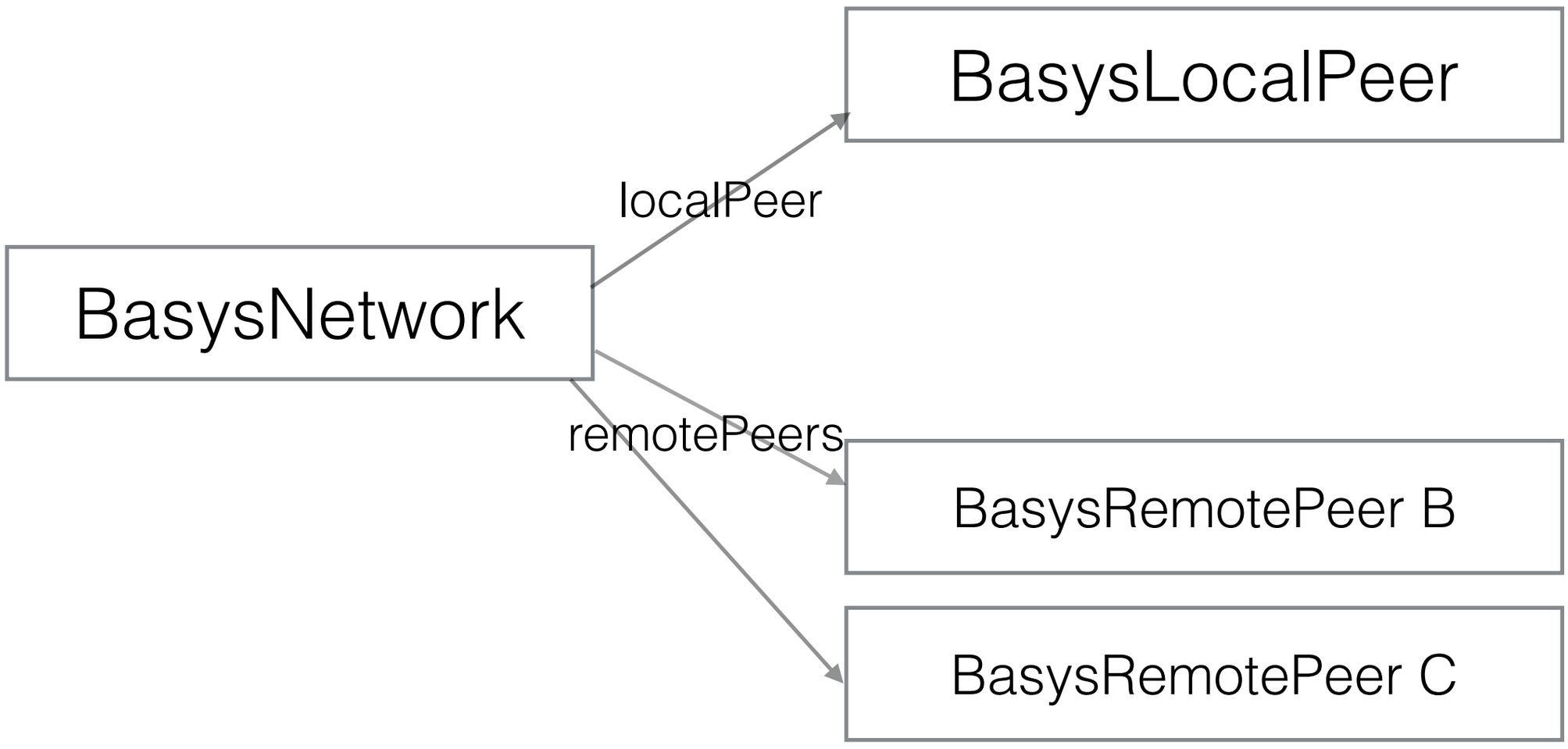
Basys

- **B**idirectional **asy**nchronou**s** network
    - Client can send data to server
    - Server can send data to client
    - Both directions are equivalent
        - But usually server can’t establish new connections
    - Asynchronous data transfer

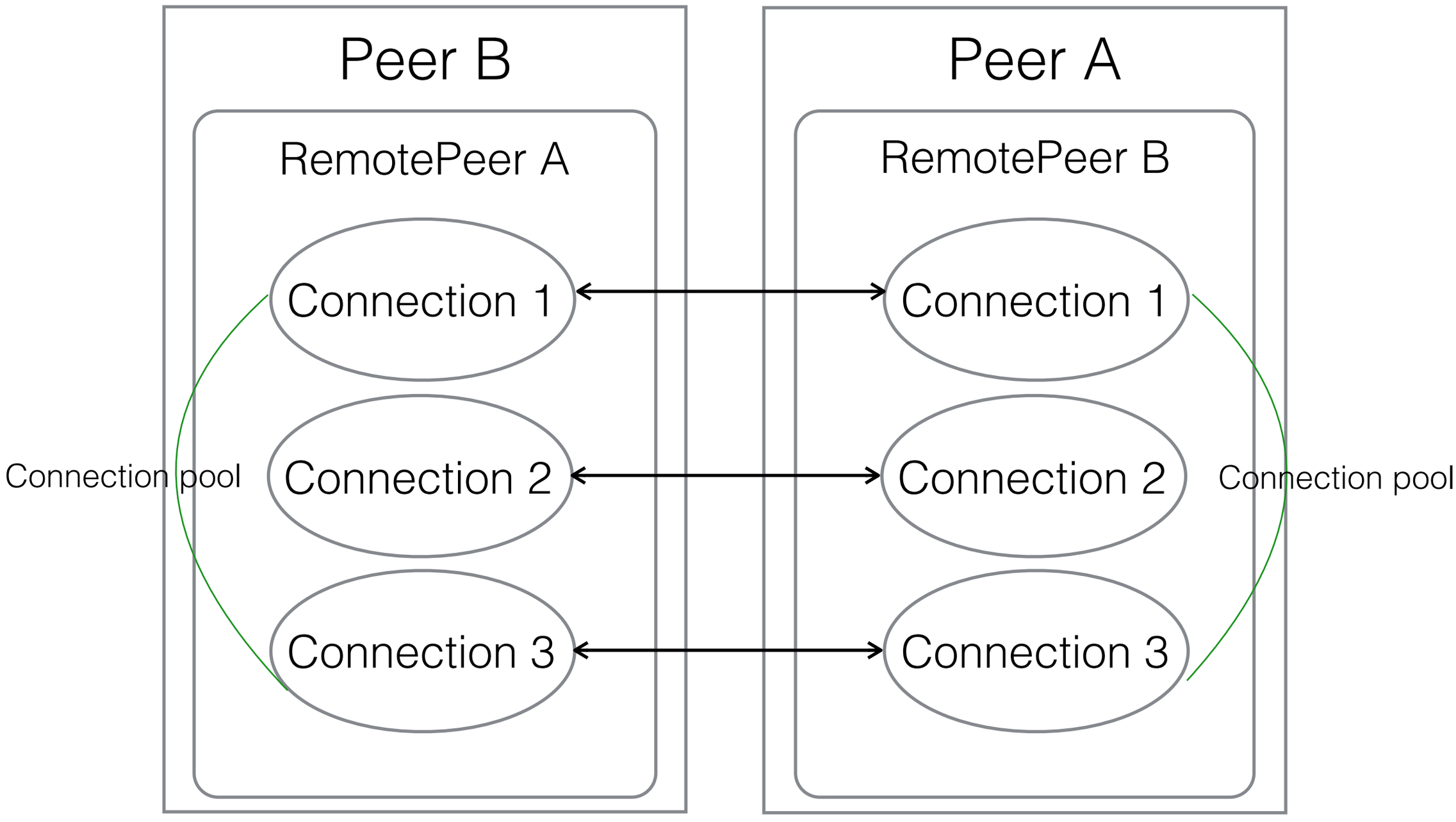
Basys models network as connected peers



Network peer structure



Peers interact by connection pool



Peers interact by connection pool

- Users not work directly with connections
- Users ask peer to send data to remote side

```
peer := network remotePeerAt: tcpAddress.
peer sendDataPacket: dataObject
```

- Connections are established by demand
- Established connections are reused

Seamless implements Basys network

- **What is data?**
    - aSeamlessRequest (MessageSendRequest, DeliveryResultRequest)
- **What to do with data?**
    - aSeamlessRequest executeFor: senderPeer

Seamless implements Basys network

- What is data?
    - aSeamlessRequest (MessageSendRequest, DeliveryResultRequest)
- What to do with data?
    - aSeamlessRequest executeFor: senderPeer
- **How to send data?**
    - objects are serialized on connection stream by serialization library.
    - transfer strategies are applied to each node of given object graph to decide how to transfer them:
        - by value
        - by reference
        - others
    - strategies are object specific and could be redefined for application
- **How to receive data?**
    - objects are materialized from connection stream by serialization library.
    - on receiver side they could be represented by specific objects (proxies, local globals)

Seamless

- First class strategies to transfer objects
    - by value
    - by reference
    - by reference with cached properties
        - properties can be transferred by reference too
    - by referenced copy
    - by deep copy
    - by global name (to transfer well known globals)
    - other specific strategies

Remote debugger tuning

network transferByReference: (Kind of: CompiledMethod) withCacheFor: #(selector methodClass isTestMethod argumentNames).

network transferByReference: (Kind of: Context) withCacheFor: #(receiver method methodClass methodSelector isBlockContext home tempNames isDead selector sender debuggerMap outerContext outerMostContext closure).

network transferByValue: (Kind of: Slot).

network transferByReference: (Kind of: ClassDescription) withCacheFor: #(name allInstVarNames allSlots).

network transferByValue: (Kind of: OrderedCollection).

network transferByValue: (Kind of: Set).

network transferByValue: (Kind of: Interval).

network transferByValue: (Kind of: Array).

Real debugger demo

- On server side:
    - RemoteUIManager registerOnPort: 40423
- On client side:
    - RemoteDebugger connectTo: anAddress

GT extensions

- GTInspetor on proxies:
    - Raw tab for remote state
    - Proxy tab for internal proxy state
- DoIt by SeamlessRemoteClassCompiler
    - all variables and globals are bound to proxies
    - self is bound to proxy
    - #doIt method is compiled locally but executed on remote side
        - remote side could not have compiler

TostSerializer

- **T**ransient **o**bject**s** **t**ransport
    - not for persistence
    - serialize on sender and materialize on receiver
    - No meta information for objects
        - no versioning
        - no migration support
- Objects are stream of references
    - which directly written on output stream in same order
    - which directly read from input stream in same order
- Support objects with cyclic references
    - duplicated objects are encoded by stream position of original object
- Support for object substitutions
    - substitutions are just injected into object stream
- Compact encoding for well known objects and classes
    - one byte for encoding

TostSerializer in Seamless

- One pass object traversal
    - With Fuel it was two:
        - Fuel itself analyses object graph
        - Seamless traverse object graph to build substitution map
- Very compact for small objects
    - Smallest communication unit (integer return):
        - 21 bytes for Tost versus 400 bytes for Fuel
- Many possibilities for new features and optimizations:
    - references should not send cache back to server
    - objects state synchronization between client and server
        - cache should be updated when reference is received again from server

ObjectTravel

- Main part of TostSerializer

ObjectTravel

- Tool to stream objects
    - traversal stream of inst vars and indexed fields

```
traveler := ObjectTravel on: (1@2 corner: 3@4).
traveler nextReference. “ => 1@2”
traveler nextReferece; nextReference. “=> 1”
   “or”
traveler referencesDo: [:each | ].
```

- Support cyclic object graphs
- Allow inject external objects

```
traveler referencesDo: [:each |
    each = 2 ifTrue: [traveler atNextStepVisit: 5@6]].
```

- Allow replace references

```
traveler referencesDo: [:each |
    each = 2 ifTrue: [traveler replaceCurrentReferenceWith: 5]].
```

ObjectTravel

- Useful methods:

```
traveler := ObjectTravel on: (1@2 corner: 3@4).
```

  - traveler countReferences “=> 6”
  - traveler collectReferences “=> {1@2. 3@4. 1. 2. 3. 4}”
  - traveler copyObject “=> deep copy of rectangle”
  - traveler findAllPathTo: 2 “=> { {1@2} }”

ObjectStatistics

- Tool to analyze set of objects
    - computes different kind of metrics from different perspective (dimensions)
        - simplistic OLAP Cube in objects space.
- Implements suitable GT extension
    - Metrics and dimensions shown in tree way inside GTInspector

SeamlessStatistics

```
stat := ObjectStatistics new.
stat
	countAllAs: 'requests';
	countDifferent: [ :r | r receiver ] as: 'instances' for: (Kind of: SeamlessMessageSendRequest);
	countAllSuch: #isOutgoing as: 'outgoing';
	countAllSuch: #isIncoming as: 'incoming'.
stat
	dimension: [ :r | r class ] named: 'requests';
	for: (Kind of: SeamlessMessageSendRequest) with: [
		stat
			dimension: [ :r | r receiver nameForSeamlessStatistics ] named: 'classes';
			with: [
				stat dimension: [ :r | r selector ] named: 'msgs'].
		stat
			dimension: [ :r | r selector ] named: 'msgs';
			with: [
				stat dimension: [ :r | r receiver nameForSeamlessStatistics ] named: 'classes']].

stat accumulateAll: requests.
```

Future work

- Remote browser
- More optimizations
- Better presentation of remote contexts
- Support for stepInto for remote call
    - distributed stack in debugger
- Distributed garbage collection
    - now it is absent
    - “debugger disconnect” cleans everything

The end

- follow me on https://dionisiydk.blogspot.com
- questions?