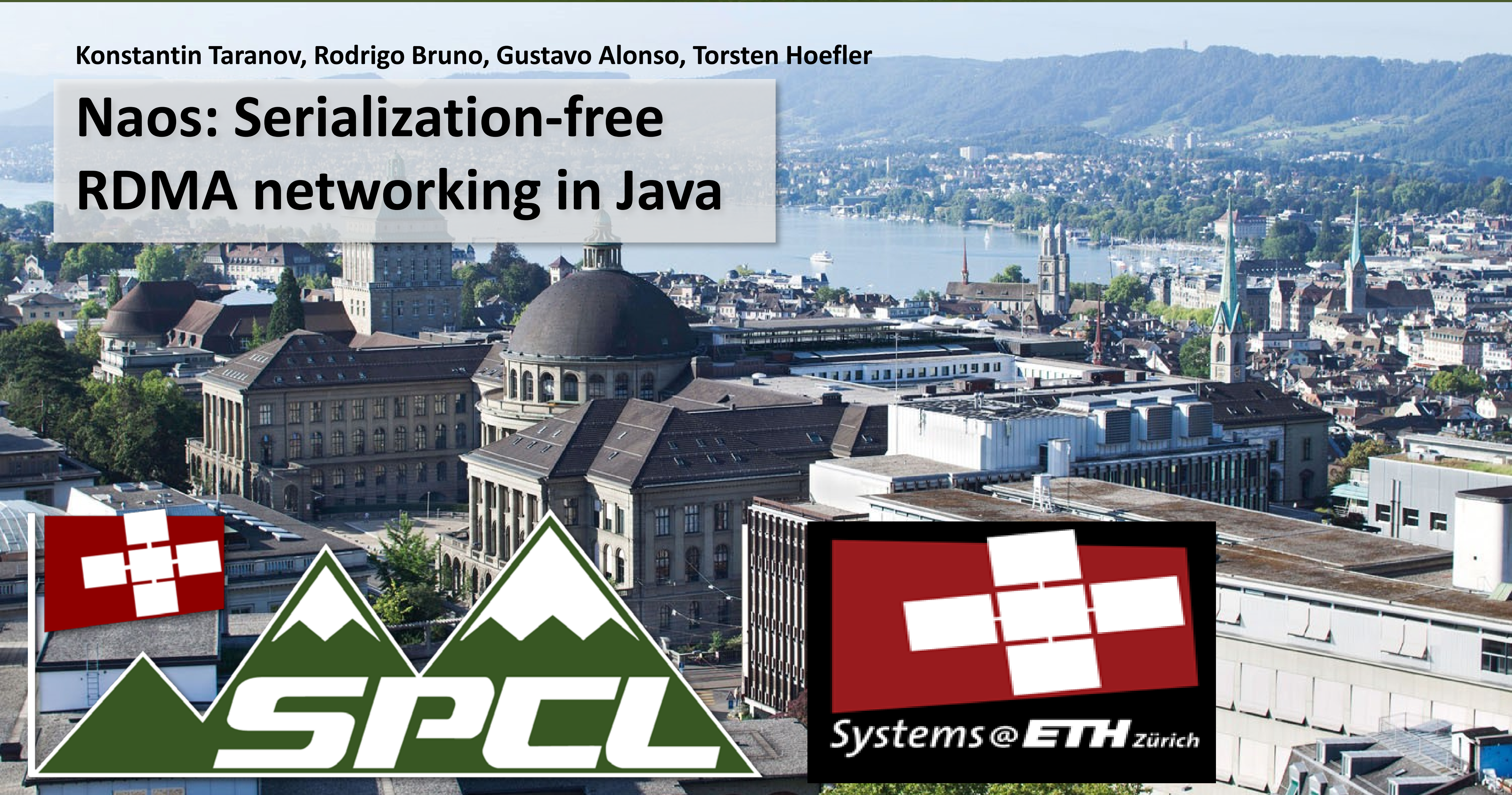
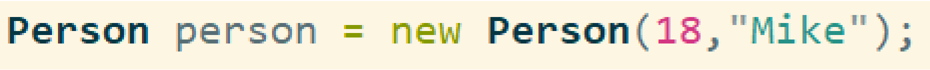
Konstantin Taranov, Rodrigo Bruno, Gustavo Alonso, Torsten Hoefler

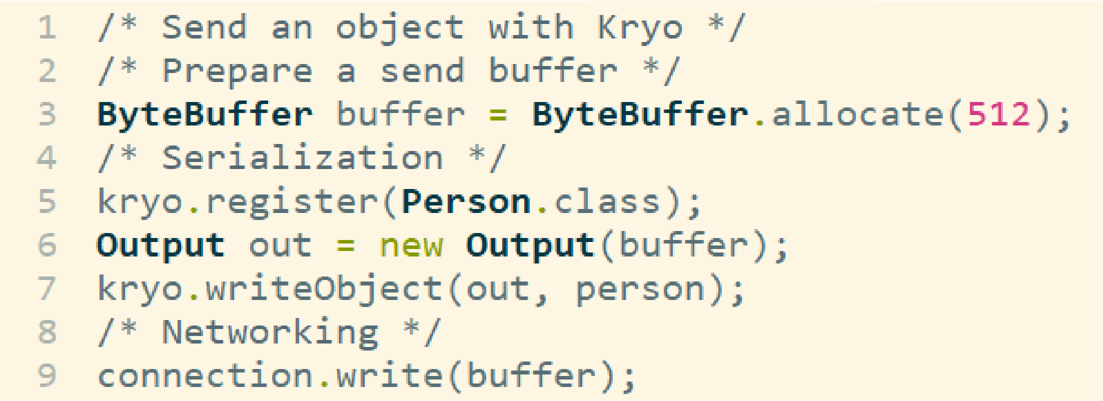
Naos: Serialization-free RDMA networking in Java



Sending and Receiving Objects in Java



```java
Person person = new Person(18,"Mike");
```



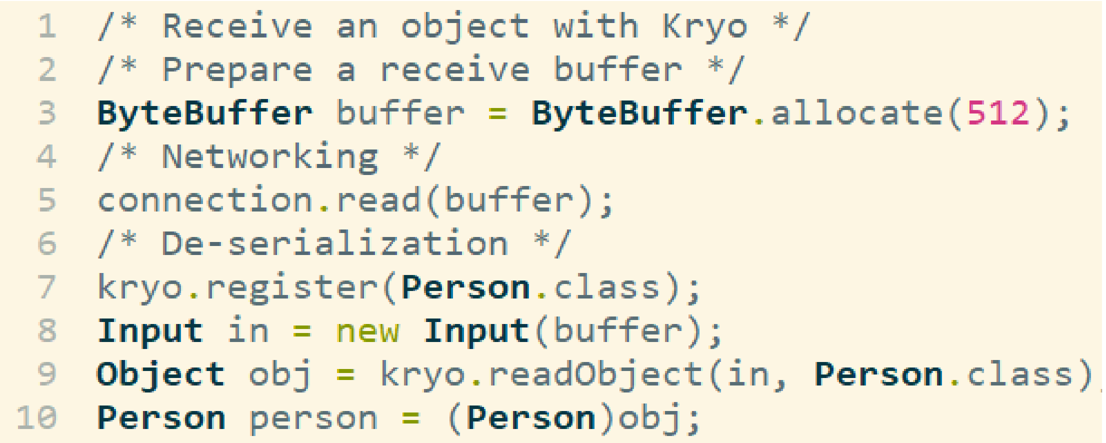
```java
/* Send an object with Kryo */
/* Prepare a send buffer */
ByteBuffer buffer = ByteBuffer.allocate(512);
/* Serialization */
kryo.register(Person.class);
Output out = new Output(buffer);
kryo.writeObject(out, person);
/* Networking */
connection.write(buffer);
```

Memory allocation

Class registration

Serialization

Writing buffer to the network



```java
/* Receive an object with Kryo */
/* Prepare a receive buffer */
ByteBuffer buffer = ByteBuffer.allocate(512);
/* Networking */
connection.read(buffer);
/* De-serialization */
kryo.register(Person.class);
Input in = new Input(buffer);
Object obj = kryo.readObject(in, Person.class)
Person person = (Person)obj;
```

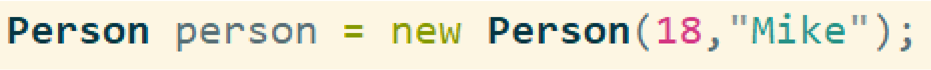
Memory allocation

Reading buffer from the network

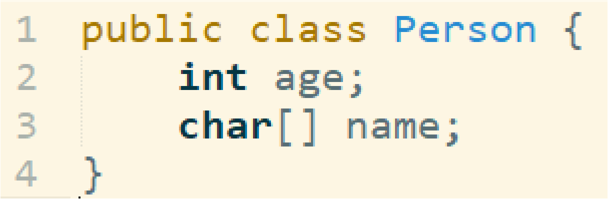
Class registration

De-serialization

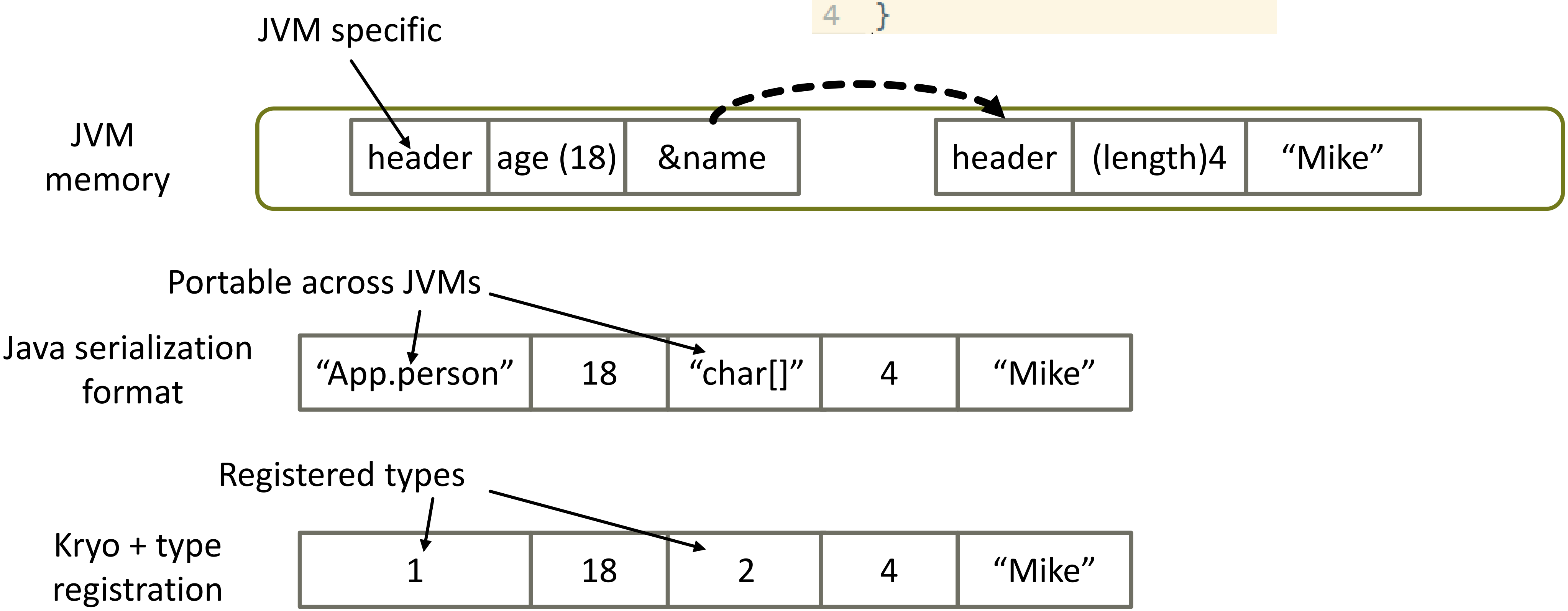
Serialization process



```
Person person = new Person(18,"Mike");
```



```
public class Person {
        int age;
        char[] name;
}
```



JVM specific

JVM memory

header | age (18) | &name

header | (length)4 | “Mike”

Portable across JVMs

Java serialization format

“App.person” | 18 | “char[]” | 4 | “Mike”

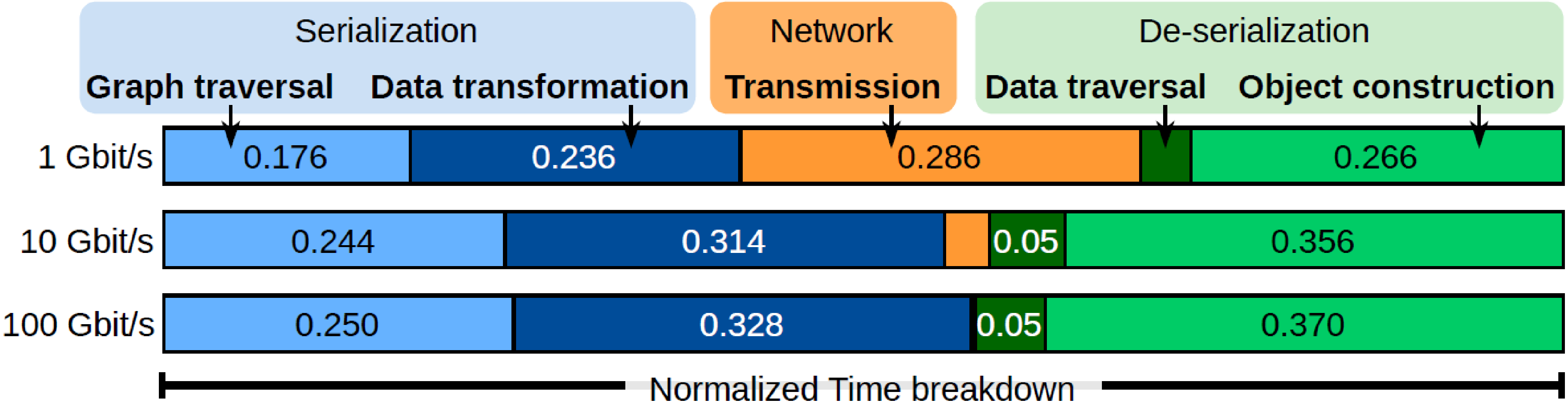
Registered types

Kryo + type registration

1 | 18 | 2 | 4 | “Mike”

Impact of network bandwidth on sending objects between JVMs

- We are sending an array of 1.28M objects.

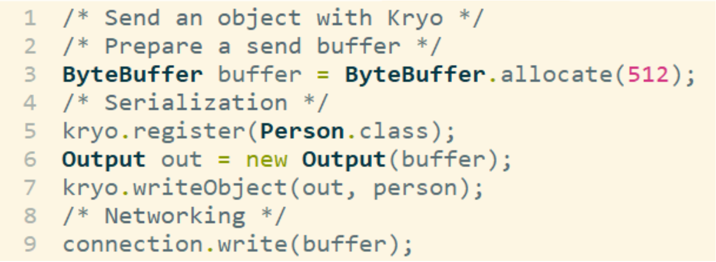


The network is not a bottleneck anymore!

Serialization already accounts for 6% of total CPU cycles at Google datacenters*.

* Svilen Kanev, et al. Profiling a Warehouse-Scale Computer. In Proceedings of the 42nd Annual International Symposium on Computer Architecture. ISCA’15

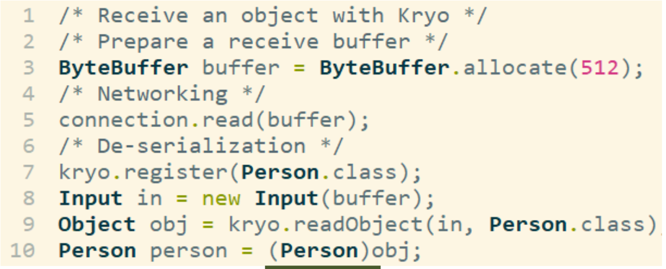
Naos – serialization-free networking library



```
/* Send an object with Kryo */
/* Prepare a send buffer */
ByteBuffer buffer = ByteBuffer.allocate(512);
/* Serialization */
kryo.register(Person.class);
Output out = new Output(buffer);
kryo.writeObject(out, person);
/* Networking */
connection.write(buffer);
```

naos

```
connection.writeObject(person);
```

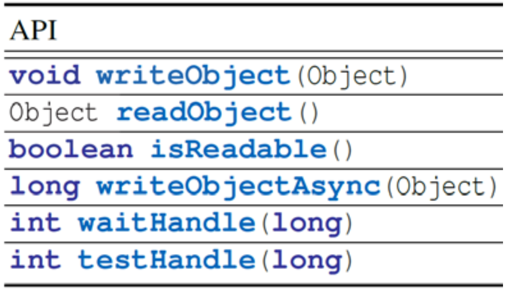


```
/* Receive an object with Kryo */
/* Prepare a receive buffer */
ByteBuffer buffer = ByteBuffer.allocate(512);
/* Networking */
connection.read(buffer);
/* De-serialization */
kryo.register(Person.class);
Input in = new Input(buffer);
Object obj = kryo.readObject(in, Person.class)
Person person = (Person)obj;
```

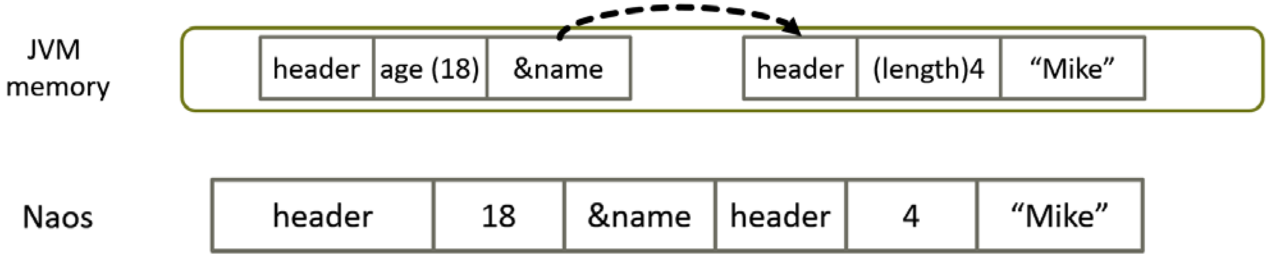
naos

```
Person person = (Person)connection.readObject();
```

Naos supports TCP and RDMA



API
`void writeObject(Object)`
`Object readObject()`
`boolean isReadable()`
`long writeObjectAsync(Object)`
`int waitHandle(long)`
`int testHandle(long)`



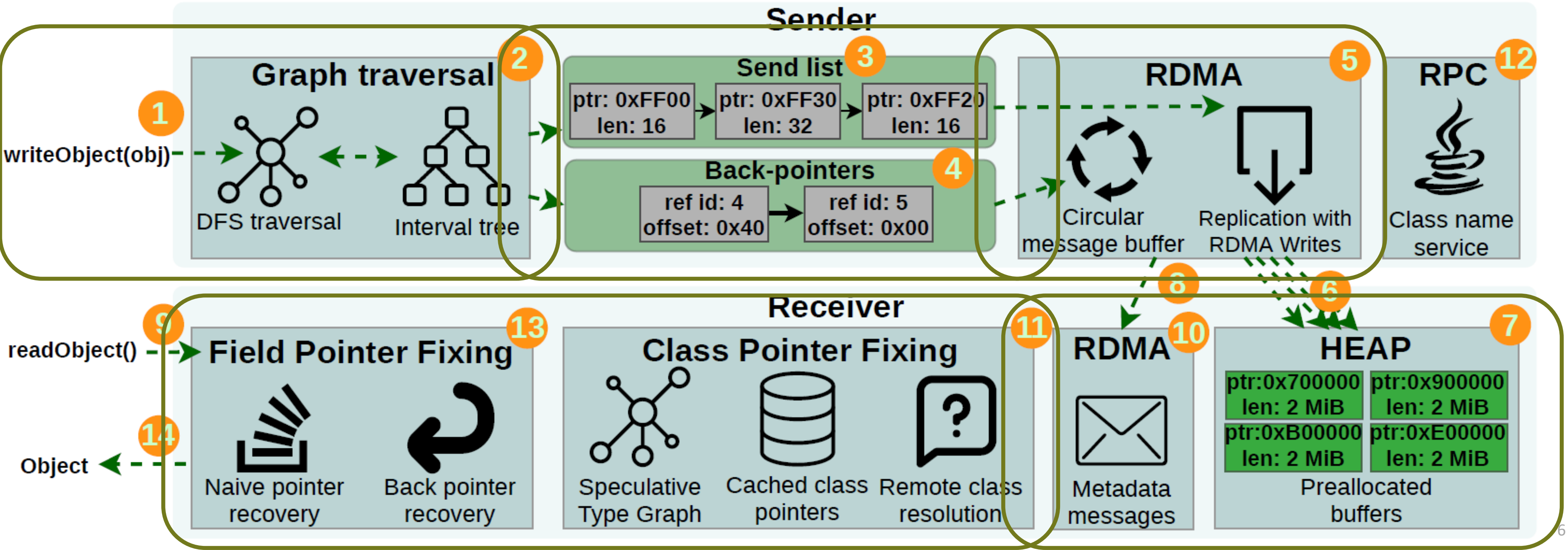
JVM memory: header | age (18) | &name | header | (length)4 | “Mike”

Naos: header | 18 | &name | header | 4 | “Mike”

Naos sends objects directly from local JVM memory to remote JVM memory *without data transformations.*

Core challenges

- Naos cannot modify sender’s memory, as the data is sent directly from the heap
- Garbage collector can concurrently touch/move objects
- RDMA-capable NICs can concurrently access the memory



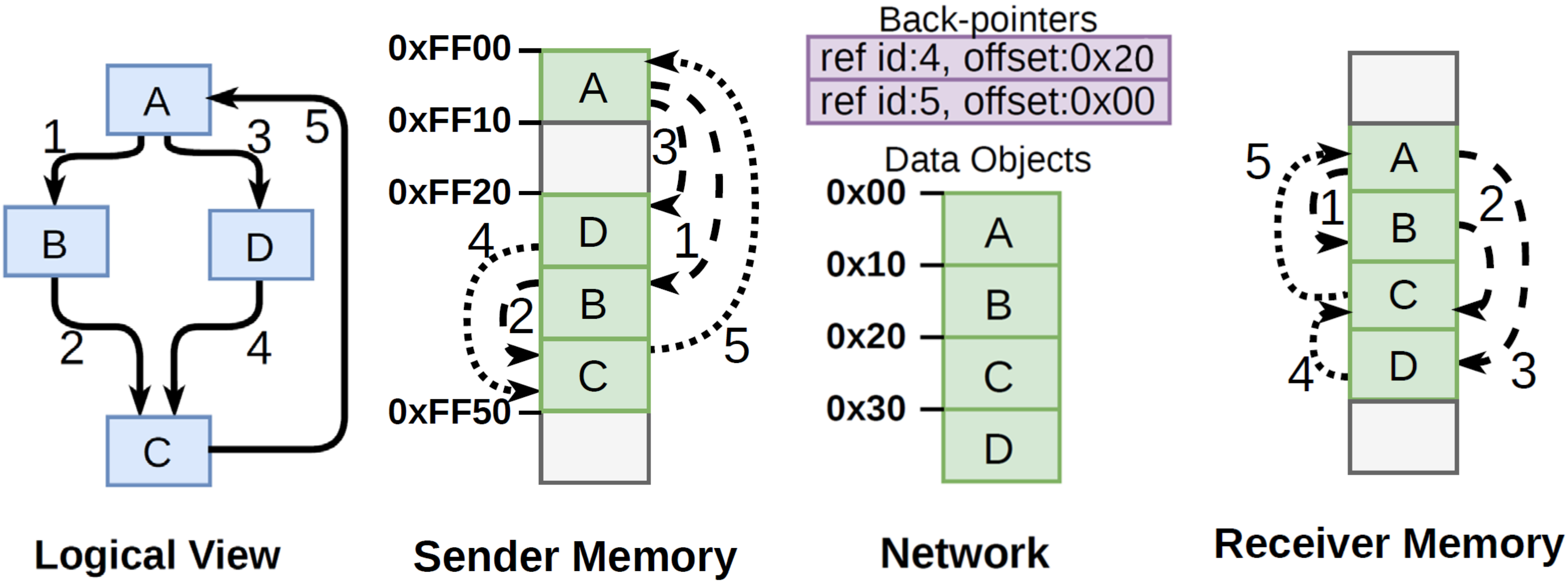
Sending object with Naos

Naos	header	18	&name	header	4	“Mike”

Naos proposes a novel way of the graph traversal:

- It traverses object graphs in depth-first search (DFS) order on both sender and receiver

- It does not modify JVM memory and sends zero metadata about “trivial” pointers



Related Optimizations

For back-pointer detection

Interval tree

To recover object pointers at the receiver

Naive pointer recovery

Back pointer recovery

To recover class pointers at the receiver

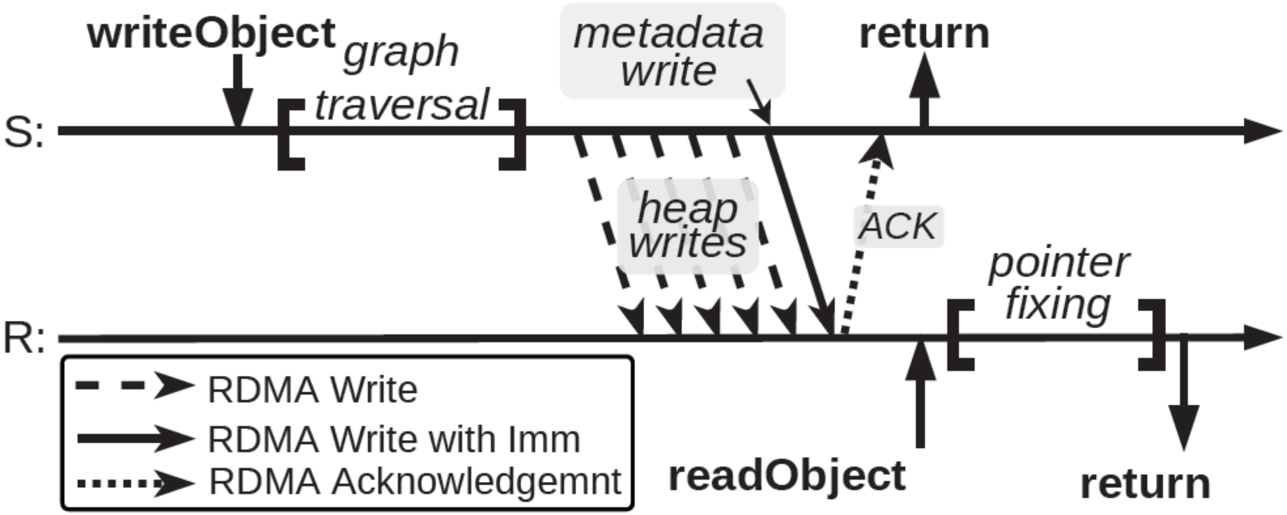
Speculative Type Graph

Cached class pointers

Remote class resolution

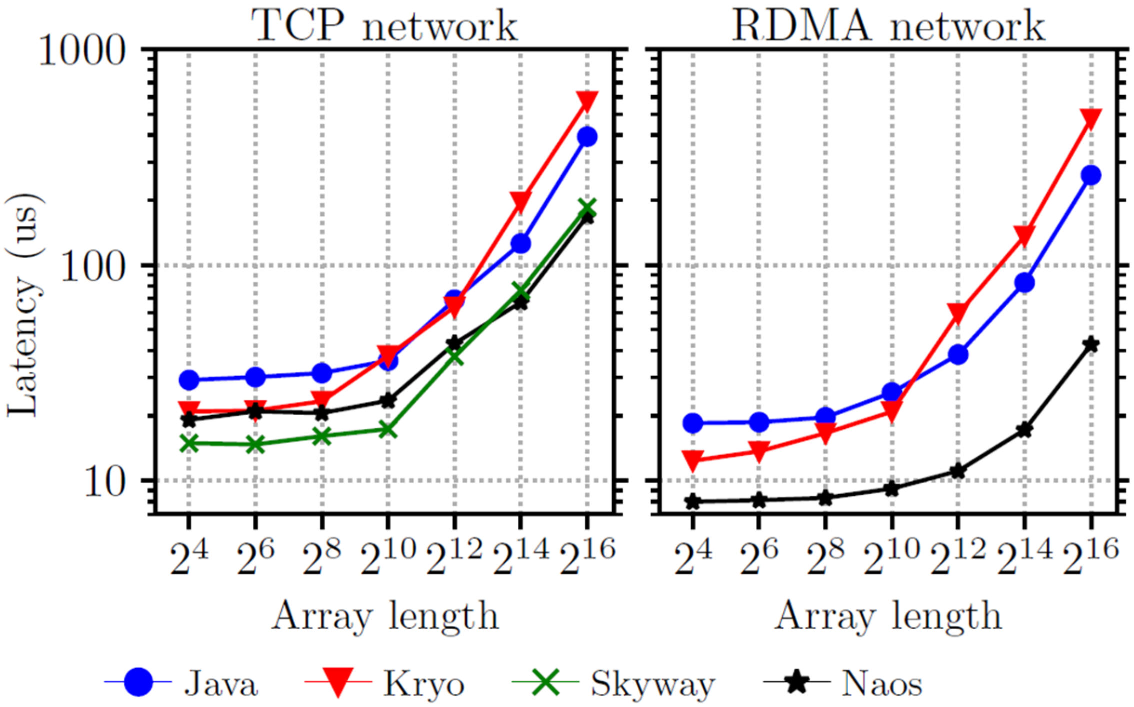
RDMA design

- The receiver reserves heap memory for receiving objects
- The sender “silently” writes object to the remote heap using RDMA writes
- Then the sender sends metadata that indicates the completion of a send

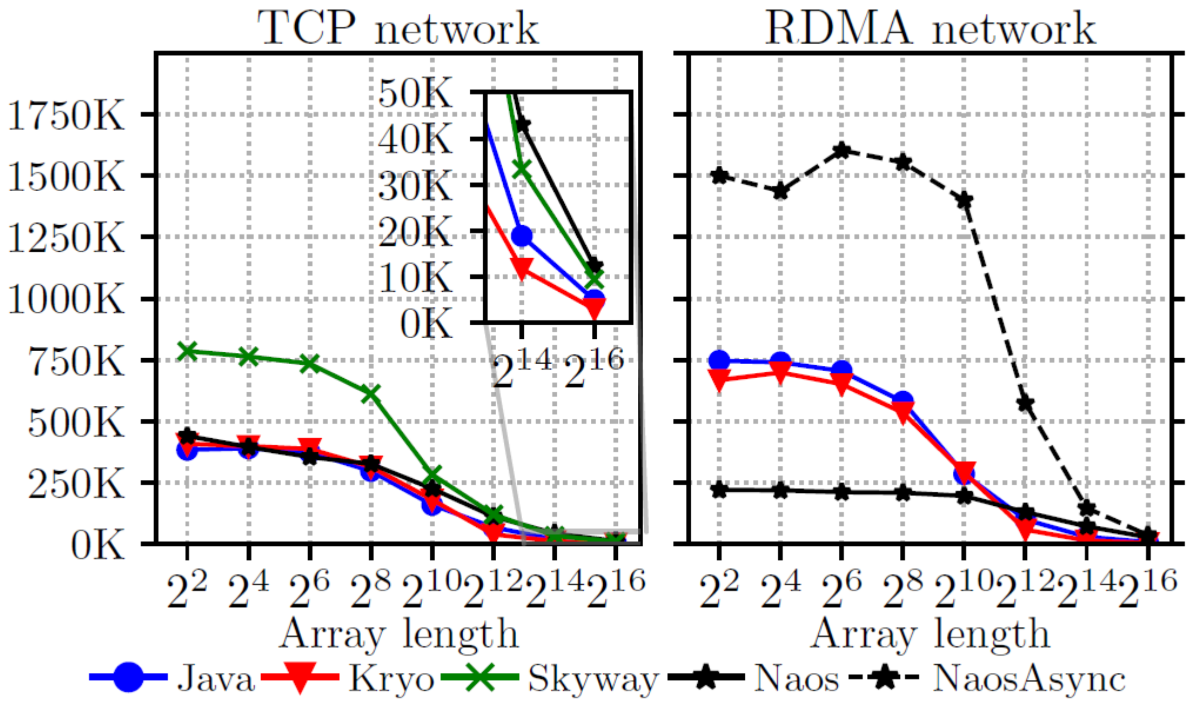


Evaluation

RTT latency for sending float[]



Throughput for sending float[]



Thank you for your attention!

- Naos is the first serialization-free communication library for JVMs
- Naos does not perform excessive data copies during communication
- Naos can directly send objects across Java heaps
- Naos supports RDMA networking

Naos implementation:

Speaker:
Konstantin Taranov
For questions:
konstantin.taranov@inf.ethz.ch