Distributed Systems Paradigms

José Orlando Pereira

HASLab / Departamento de Informática
Universidade do Minho

2021/2022

Motivation

- Handle a large number of clients and requests with a single server
- The “c10k problem” in 1999:
  - http://www.kegel.com/c10k.html
- Examples:
  - financial, games, ...
  - notifications in mobile apps
  - machine-to-machine (M2M)

Case study

- Simple chat server:
  - Forward all messages to all clients
- Consider:
  - Large number of clients
  - Slow connections

First threaded solution

- For each connection:
  - Handler thread
- When reading, write to all other connections
- Use buffering:
  - At user level (streams): To minimize system calls
  - In the kernel (socket): To cope with slow readers

Sockets in java.net

```
ServerSocket ss=new ServerSocket(12345);

while(true) {
    Socket s=ss.accept();

    InputStream is=s.getInputStream();
    OutputStream os=s.getOutputStream();

    // i/o

    s.close();
}
```

Buffers in java.net

```
ServerSocket ss=new ServerSocket(12345);

while(true) {
        Socket s=ss.accept();

        InputStream is=new BufferedInputStream(s.getInputStream());
        OutputStream os=new BufferedOutputStream(s.getOutputStream());

        // i/o

        os.flush();

        s.close();
}
```

Needed to actually write!

Memory

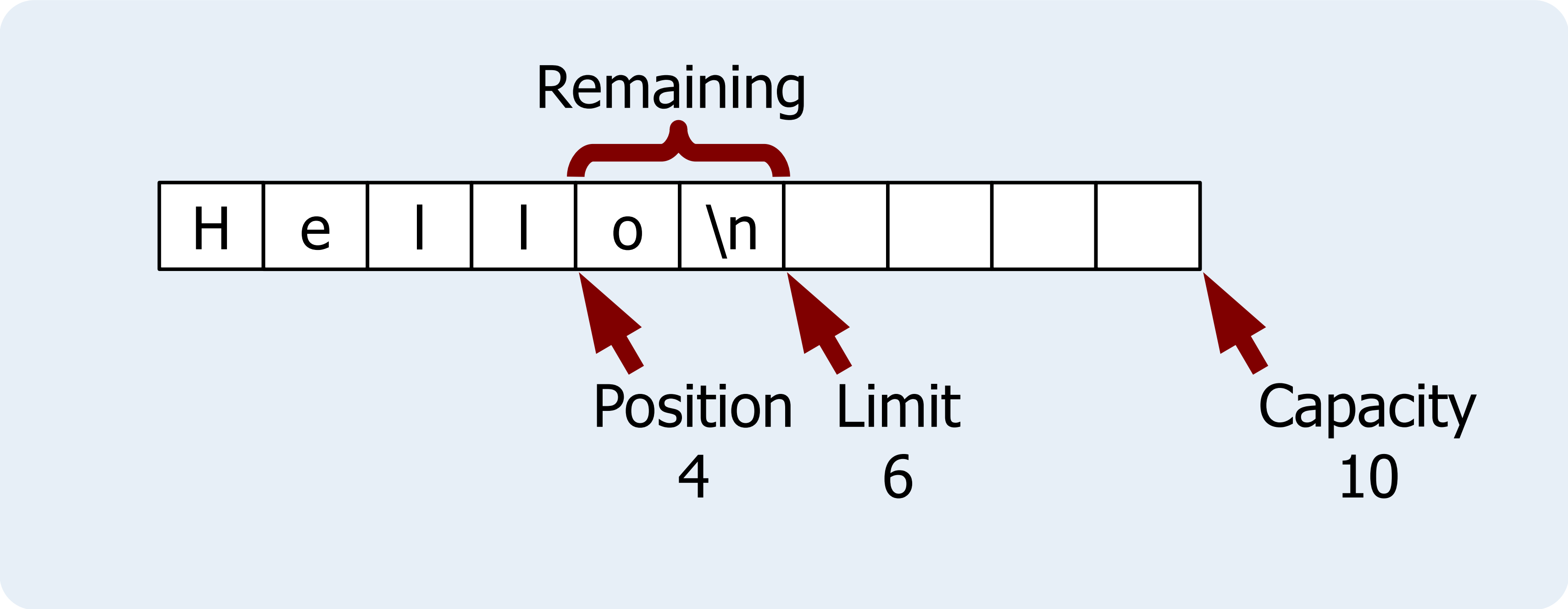
- Memory: *n* connections x messages in transit (~ n^2)
  - Caused by data copying in stacked abstractions
    - Serialization!
  - Overhead in allocation and garbage collection
- Solution: Store transient data in reusable shared buffers
  - Pointers/indexes into statically allocated data

Sockets in java.nio

```
ServerSocketChannel ss=ServerSocketChannel.open();
ss.bind(new InetSocketAddress(12345));

while(true) {
        SocketChannel s=ss.accept();

        // i/o

        s.close();
}
```

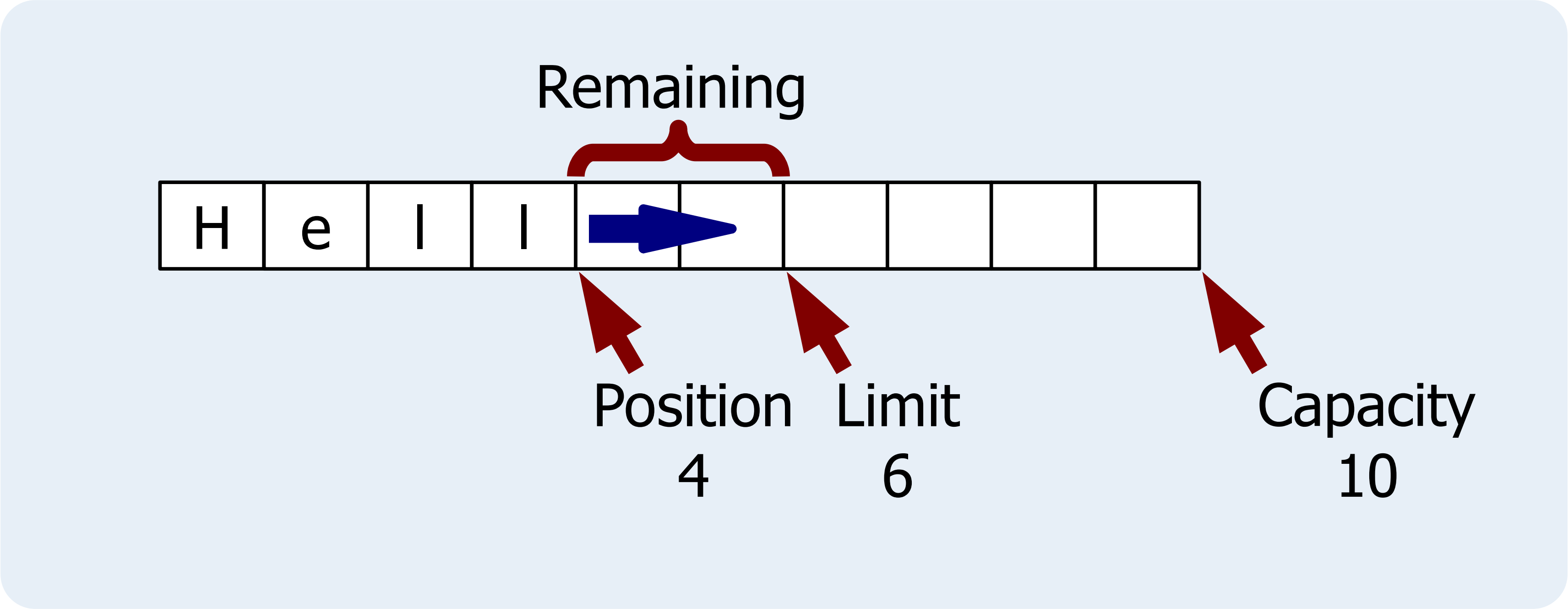
Buffers in java.nio

- Buffer = Array + Indexes:



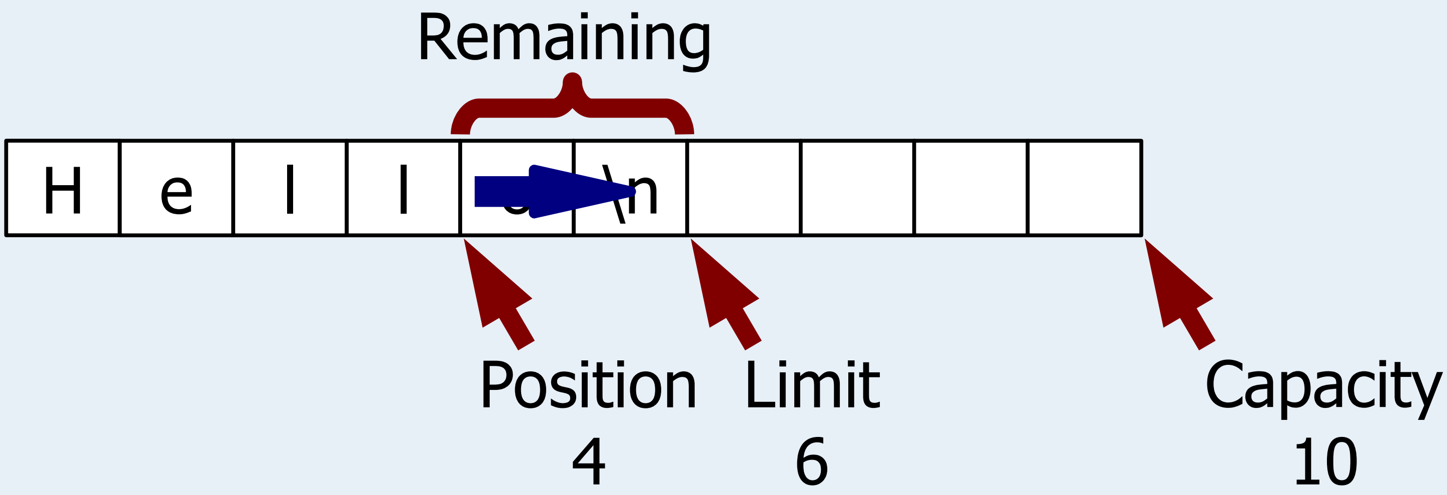
Buffers in java.nio

- Put/read: advances position, sets content



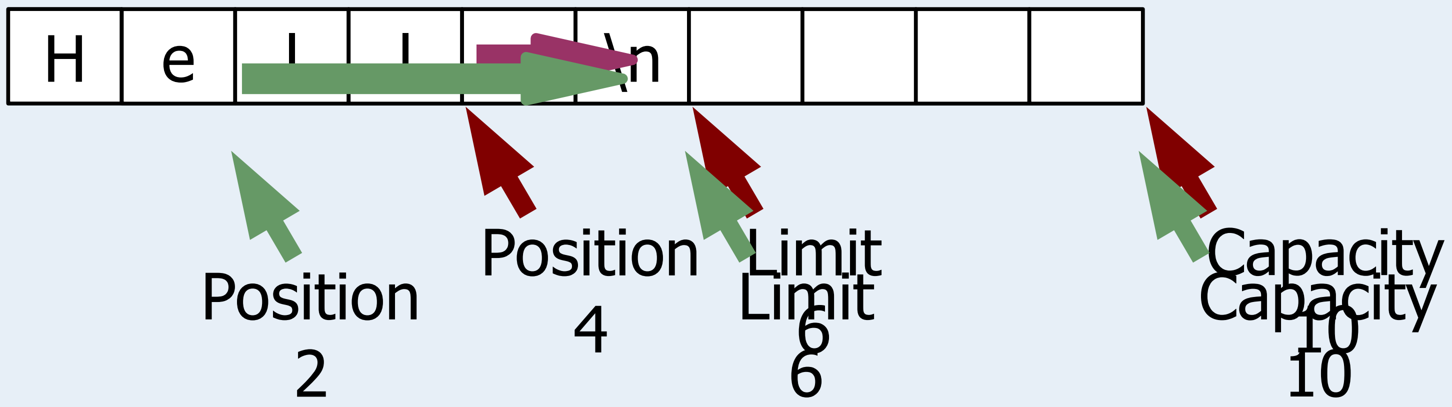
Buffers in java.nio

- Get/write: advances position, gets content

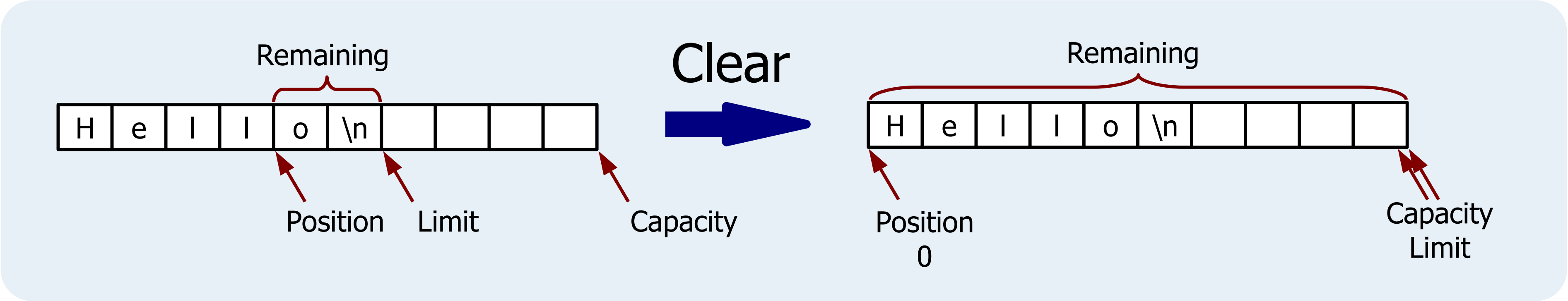


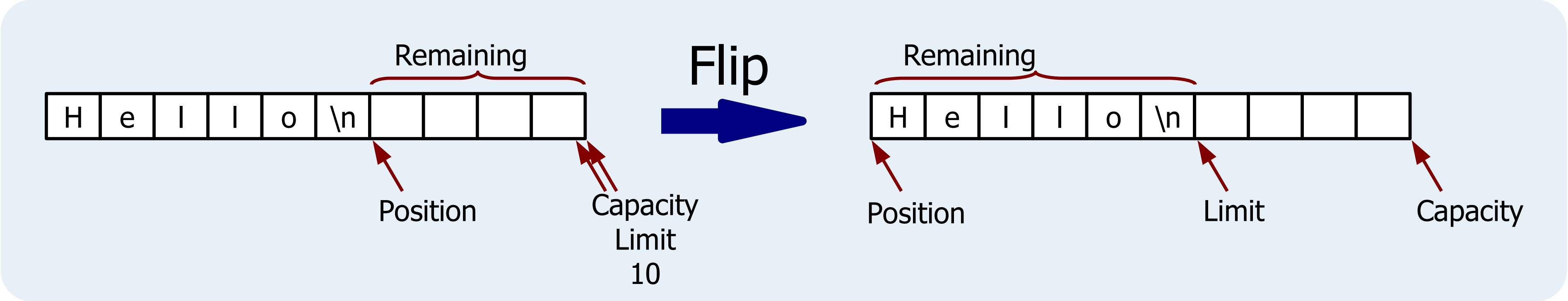
Buffers in java.nio

- Duplicate and wrap: multiple pointers into the same array

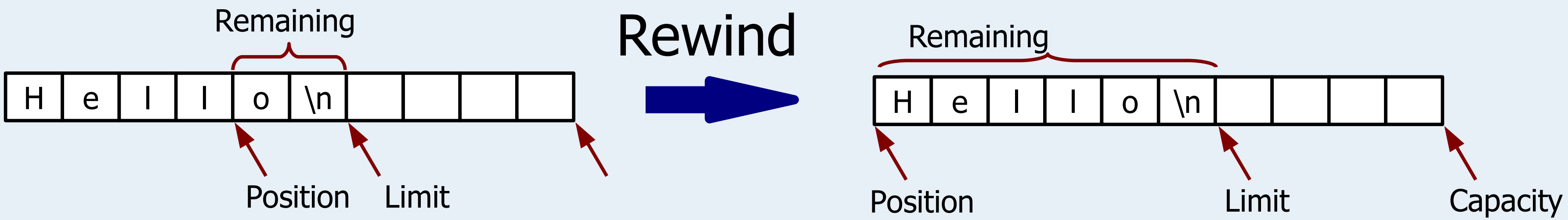


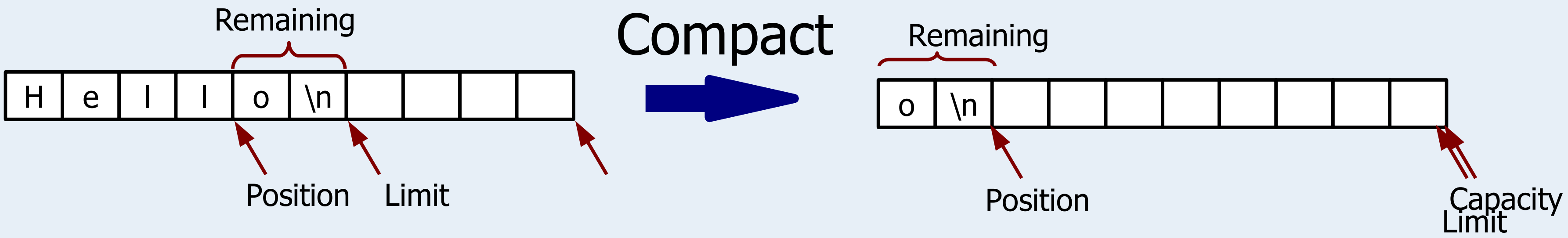
Buffers in java.nio





Buffers in java.nio





Sockets in java.nio

```
try {
        ByteBuffer buf=ByteBuffer.allocate(100);

        s.read(buf);
        buf.flip();
        for(SocketChannel r: receivers) {
                r.write(buf.duplicate());
        }
} catch(IOException e) {
        report(e);
}
```

Shared buffers

- Memory used: messages in transit (~ n)
- Ideally, never allocate or dispose of memory in normal operation:
  - No overhead, but...
  - Needs reference counting to know when to reuse

Flushing buffers

```
ByteBuffer buf=ByteBuffer.allocate(100);
try {
        s.read(buf);
        buf.flip();
        for(SocketChannel r: receivers) {
                r.write(buf.duplicate());
        }
        buf.clear();
} catch(IOException e) {
        report(e);
}
```

What if
write blocks?

Second threaded solution

- For each connection:
  - Reader thread + Pending queue + Writer thread
- When reading, insert in outgoing queues and notify writer threads
  - If a queue overflows, the reader must block, drop some data, or even disconnect the writer
- When writing, remove from queue
  - If readers might have blocked on this queue, notify them now

Threads summary

- Simple programming model
- Problems:
  - Memory overhead (stacks and buffers)
  - Context switches and lock contention
  - “Thundering herd”, hidden queue, and fairness