Hello

- IP multicast
- JGroups multicast
- JGroups API
- Example
- Demonstration
- How JGroups works
- Conclusions
- Questions
- References

JGroups: a Java system for communicating with a group of machines

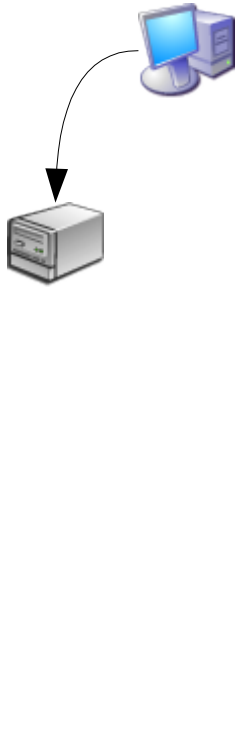
Savaş Ali TOKMEN

MSc in Computer Engineering

UFR IMA, Grenoble, FRANCE

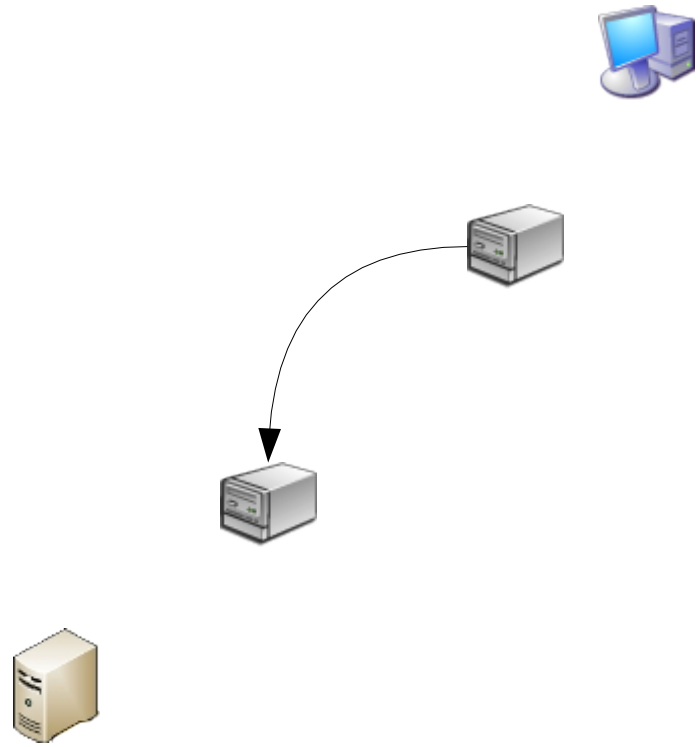
IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



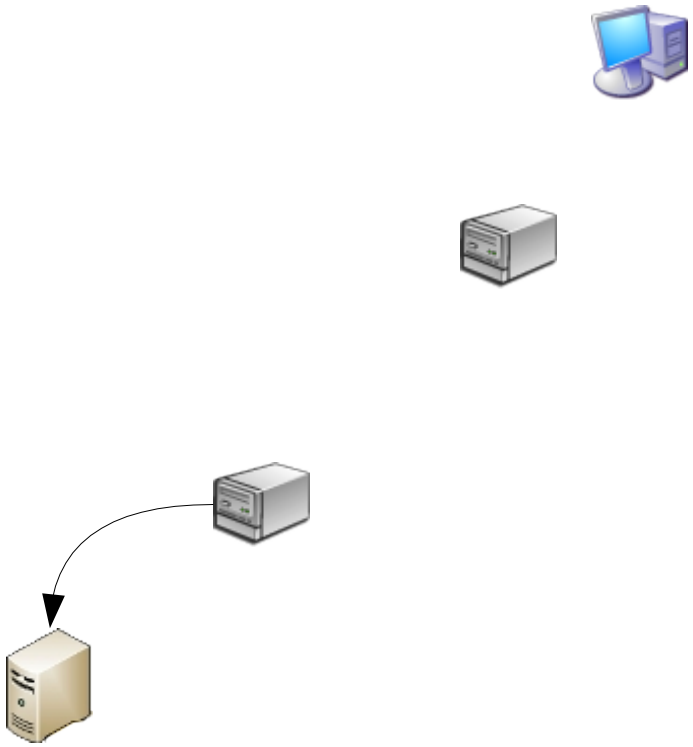
IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



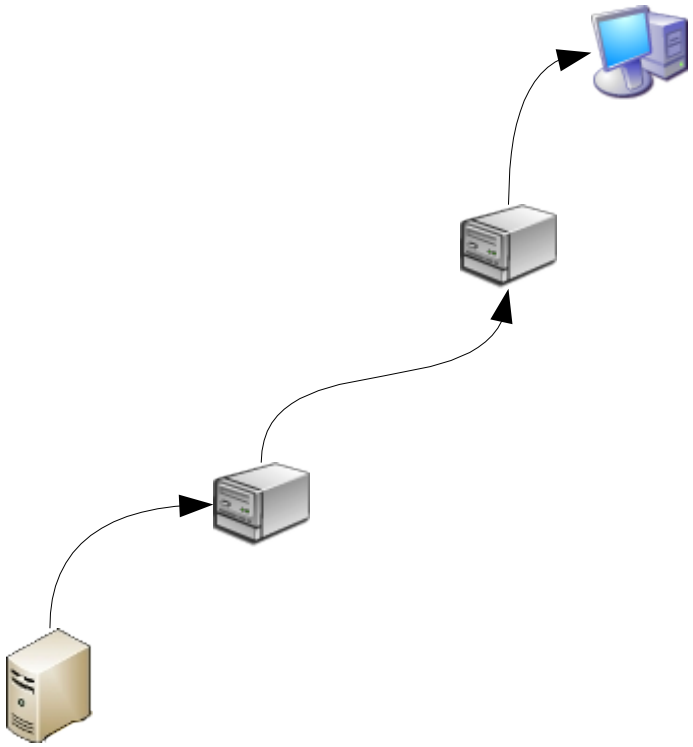
IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



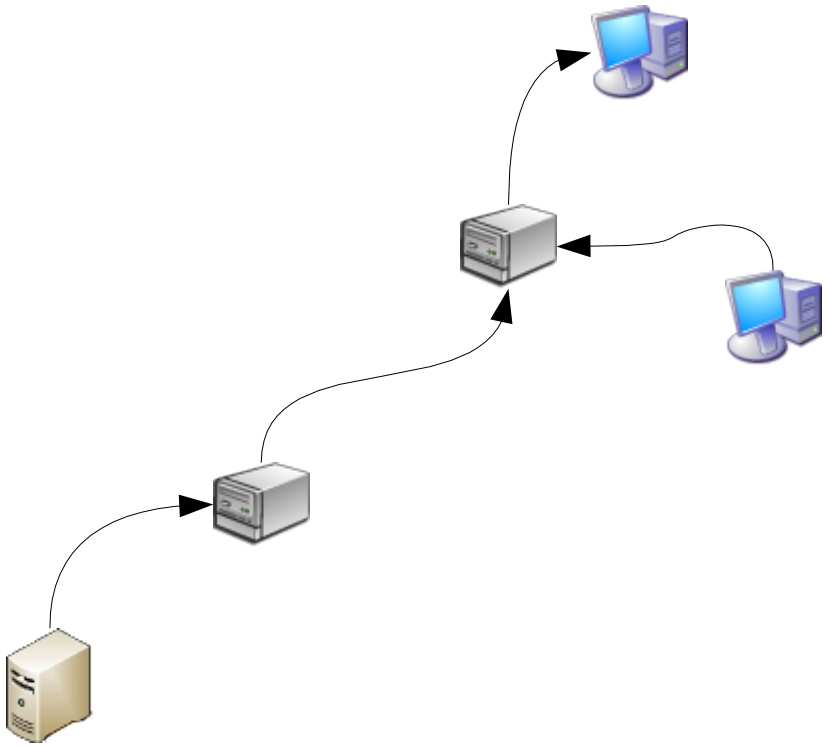
IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



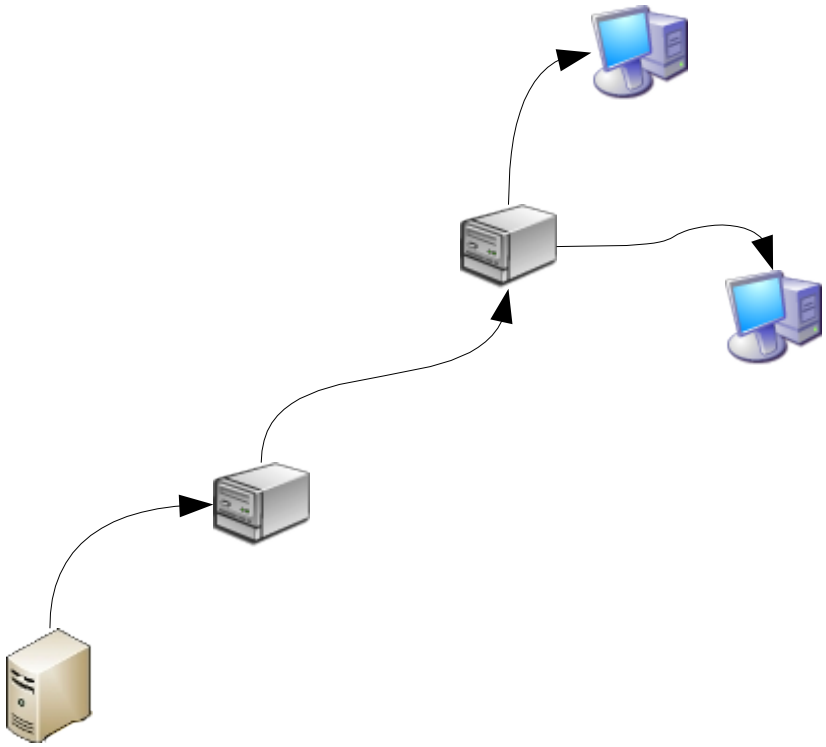
IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



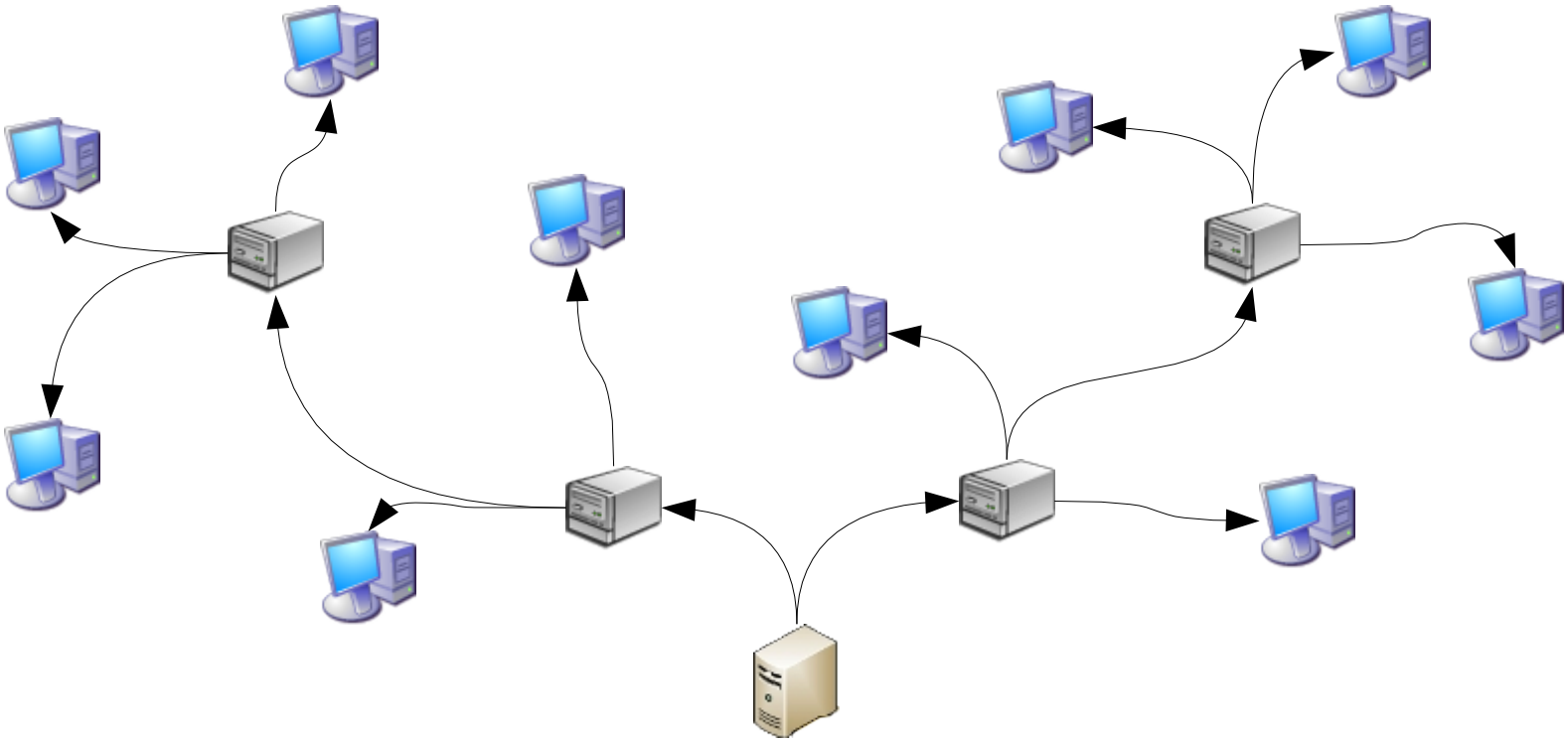
IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



IP multicast

- Aim: resource-efficient data transmission to multiple targets
- IGMP protocol



- Radio / Video transmission, OSPF

JGroups multicast

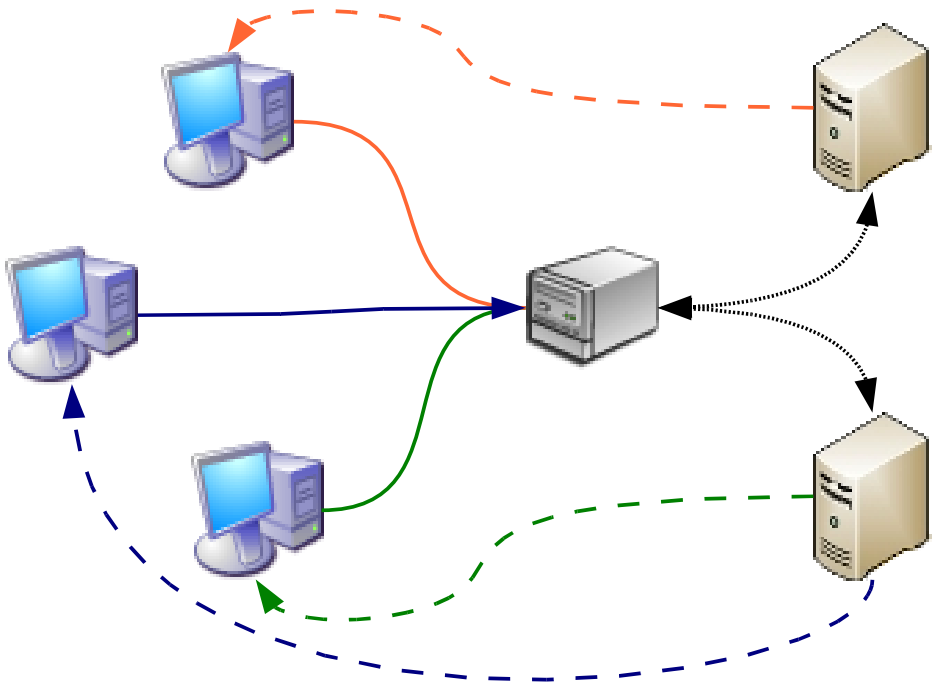
- Initial aim: to provide a Java class for IP multicast
- Extensions:
  - Choice between protocols: UDP, TCP, JMX
  - Subscription and unsubscription notification
  - Crashed member detection
  - Reliable and ordered transmissions
  - Point-to-point messaging
  - Fragmentation of big messages
  - Scheduling policies: atomic (all or none), FIFO, total order
  - Encryption

JGroups API

- Channel creation: `new JChannel(props);` with properties such as:
  - Transport protocol
  - Reliable or non-reliable transmissions
  - Scheduling policies
- `connect` method to join the group
- `send` method for multicast or unicast sending
- `receive` method (blocking) or publish / subscribe alerts for receiving
- `disconnect` method to quit group

Example

- Request distribution system



Communication types:

HTTP request
HTTP response
JGroups

Machine types:

Client
Server
Network entry point

Demonstration

- Distributor's code

```
public void OnRequest( HTTPRequest request ) {
    // Get the list of available servers
    Vector<Address> servers = channel.getView().getMembers();
    if( servers.size() <= 1 ){
        // In this case, there are no servers!
        OnError( 501 );
    } else {
        int gatewayPosition = servers.indexOf(channel.getLocalAddress());
        int targetServer = gatewayPosition;
        // Pick a server to process request, make sure it really
        // is a server and not the distributor
        while ( targetServer  == gatewayPosition ) {
            targetServer = random.nextInt(servers.size());
        }
        // Redirect request to server. Server will respond to client directly.
        channel.send(new Message( servers.get(targetServer), null, request ));
    }
}
```

- Servers' code

```
while( true ) {
    // Wait for a request
    Object o = channel.receive(0);
    // Only process normal messages
    if( o instanceof Message ) {
        o = ((Message) o).getObject();
        // Verify that it's an HTTP request
        if( o instanceof HTTPRequest ) {
            // Call the standard HTTP server program with request
            ProcessRequest( (HTTPRequest) o );
        }
    }
}
```

How JGroups works

- Ethereal

JGroups - filtered.pcap - Ethereal

File Edit View Go Capture Analyze Statistics Help

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.25.131	228.8.8.8	IGMP	V2 Membership Report
2	0.007931	192.168.25.131	224.0.0.75	IGMP	V2 Membership Report
3	0.084068	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
4	1.150895	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
5	2.443899	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
6	9.965719	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
7	11.028988	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
8	11.146212	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
10	21.598201	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
11	22.212492	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
12	27.688201	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
13	28.136554	192.168.25.131	228.8.8.8	UDP	Source port: 1033 Destination port: 45566
14	29.132816	192.168.25.131	228.8.8.8	UDP	Source port: 1033 Destination port: 45566
15	30.118251	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
16	30.207289	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
17	30.400785	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
18	30.407729	192.168.25.131	228.8.8.8	UDP	Source port: 1033 Destination port: 45566
20	45.593701	192.168.25.131	228.8.8.8	UDP	Source port: 1033 Destination port: 45566
21	45.612956	192.168.25.131	228.8.8.8	UDP	Source port: 1033 Destination port: 45566
22	48.458657	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
23	48.734481	192.168.25.131	228.8.8.8	UDP	Source port: 1029 Destination port: 45566
24	51.434913	192.168.25.131	228.8.8.8	UDP	Source port: 1033 Destination port: 45566
25	51.554688	192.168.25.131	224.0.0.2	IGMP	V2 Leave Group
26	51.582727	192.168.25.131	224.0.0.2	IGMP	V2 Leave Group

Pros and perspectives

- Pros:
  - Stable system for dynamic groups
  - Very modular architecture (OSI)
  - Ships with 19 examples and 117 tests
- Open projects
  - Ethereal plugin
  - Bluetooth support
  - Compatibility with Java standards: JavaSpaces, ...
  - HTTP load balancer
  - ...

Cons

- Functionnalities not offered by JGroups but offered by rivals such as .NET:
  - Packet coalescing
  - Priorities
  - Per-message transmission options
  - Subgroups
  - Stream exchanging
  - State exchange
- JGroups doesn't have a logo!
- User's manual partially empty

Questions ?

- IP multicast
- JGroups multicast
- JGroups API
- Example
- Demonstration
- How JGroups works
- Conclusions
- **Questions**
- References

Questions ?

References

- JGroups web site and links in it (mostly JBoss presentations)
- Web sites such as Wikipedia or Cisco for information about grouping and load balancing protocols in TCP/IP
- Ethereal
- Didier Donsez' lecture notes on Jini
- Microsoft's MSDN library

Thank you

Thank you

Documents available on

http://scholar.alishomepage.com/Master/JGroups/