

Distributed Systems

Tutorial 5 - Ensemble (Group Communication Middleware)

edited by Alex Kogan (sakogan@cs.technion.ac.il)
winter semester, 2009-2010

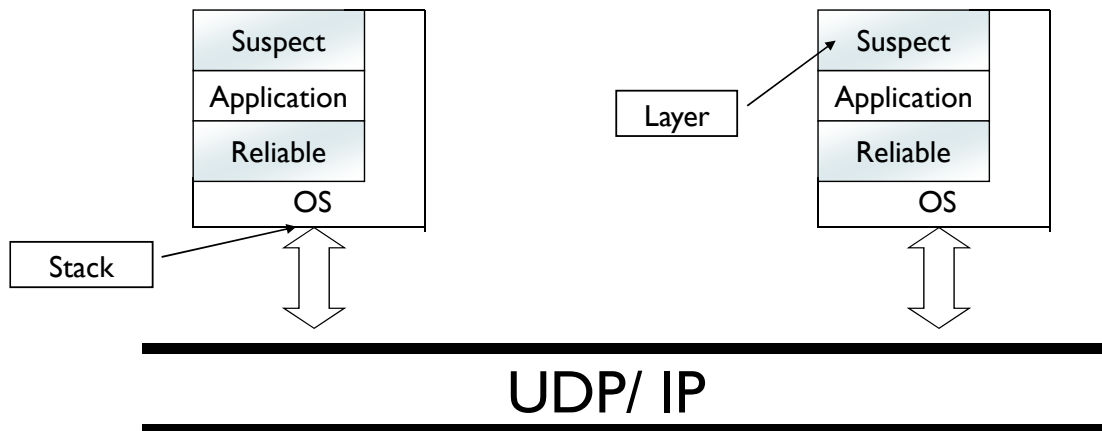
Ensemble

- ▶ A group communications implementation for research
 - ▶ <http://dsl.cs.technion.ac.il/projects/Ensemble/>
 - ▶ also at the course web-site
- ▶ Group communication as a middleware, providing an application with:
 - ▶ group membership / member status
 - ▶ support for various reliable communication and synchronization schemes

Architecture

▶ Modular design

- ▶ provides various micro-layers that may be stacked to form a higher-level protocol



▶ 3

Examples of layers

- **Total** – totally ordered messages
- **Suspect** – failure detection
- **Drop** – randomized message dropping
- **Privacy** – encryption of application data
- **Frag** – fragmentation and reassembly of long messages

▶ 4

Stacks

- ▶ Combinations of layers that work together to provide high-level protocols

Stack creation:

- ▶ A new protocol stack is created at each endpoint of a group whenever the configuration (e.g., the view) of the group changes
- ▶ All endpoints in the same partition receive the same *ViewState* record to create their stack

▶ 5

Intra-stack communication

- ▶ A layer can generate an event and invoke a neighboring layer callback/handler to pass it the event
- ▶ No two callbacks/handlers invoked concurrently
 - ▶ single-threaded runtime
- ▶ Since events are passed by direct non-concurrent procedure calls, we have a synchronized FIFO passage of events in a stack
 - ▶ life is easy!

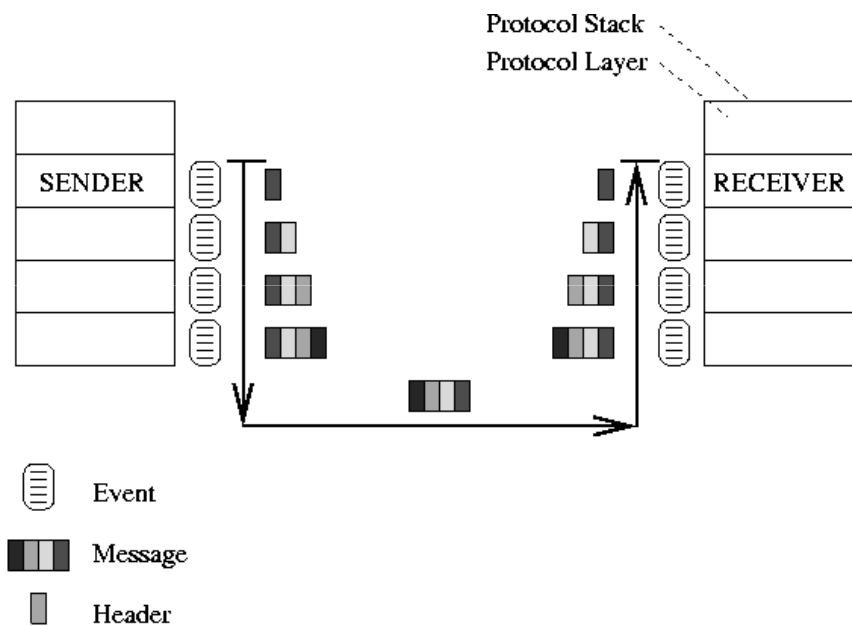
▶ 6

Inter-stack communication

- ▶ A layer in a stack may send a *message* to its corresponding peer on a different stack in the following ways:
 - ▶ generating a new message
 - ▶ piggybacking information to a message received from the layer above
- ▶ Layers never read or modify other layers' message headers

▶ 7

Inter-stack communication cont.



▶ 8

Layer example: multicast stability

Stable layer:

- ▶ Tracks the stability of multicast messages
- ▶ Protocol:
 - ▶ Maintain $Acks [N] [N]$ by unreliable multicast:
 - ▶ $Acks [s] [t]$: #(s ' messages) that t has acknowledged
 - ▶ Stability vector
 $StblVct = \{(minimum\ of\ row\ s): (for\ each\ s)\}$
 - ▶ NumCast vector
 $NumCast = \{(maximum\ of\ row\ s): (for\ each\ s)\}$
 - ▶ Occasionally, recompute $StblVct$ and $NumCast$, then send them down in a *Stable* event

▶ 9

Layer example: reliable multicast

Mnak layer:

- ▶ Implements a reliable FIFO-ordered multicast protocol
 - ▶ messages from live members are delivered reliably
 - ▶ messages from faulty members are retransmitted by live members
- ▶ Protocol:
 - ▶ keep a record of all multicast messages to retransmit on demand
 - ▶ use *Stable* event from Stable layer:
 - ▶ $StblVct$ vector is used for garbage collection
 - ▶ $NumCast$ vector gives an indication to lost messages => recover them
 - ▶ use sequence number to ensure FIFO ordering

▶ 10

Layer example: ordering

Sequencer layer:

- ▶ Provides total ordering
- ▶ Protocol:
 - ▶ members buffer all messages received from below in a local buffer
 - ▶ the leader periodically multicasts an *ordering message*
 - ▶ members deliver the buffered messages according to the leader's instructions

▶ 11

Layer example: failure detector

Suspect layer:

- ▶ Regularly pings other members to check for suspected failures
- ▶ Protocol:
 - ▶ If ($\#unacknowledged\ Ping\ messages\ for\ a\ member > threshold$) send a *Suspect event down*

Slander layer:

- ▶ Share suspicions between members of a partition
 - ▶ The leader is informed so that faulty members are removed, even if the leader does not detect the failures.
- ▶ Protocol:
 - ▶ The protocol multicasts slander messages to other members whenever receiving a new *Suspect event*

▶ 12

Ensemble in practice with C# (Java)

▶ 13

Ensemble C# (JAVA) API

- ▶ The C# (Java) API for Ensemble uses five public classes:
 - ▶ `View` – describes a group membership view
 - ▶ `JoinOps` – specifications for group name and layer stack
 - ▶ `Member` – status of the member within the group
 - ▶ `Connection` - implements the actual socket communication between the client and the server
 - ▶ `Message` - describes a message received from Ensemble
 - ▶ A message can be:
 - a new `View`
 - a multicast message
 - a point-to-point message
 - a block notification
 - an exit notification

▶ 14

Creating a C# (Java) application on top of Ensemble

▶ Step 1: Start a new connection

```
Connection conn = new Connection ();  
conn.Connect ();
```

Upon connecting, one can call the following methods of the object `conn`:

```
public bool Poll(); //non-blocking  
public Message Recv(); //blocking
```

▶ 15

Creating a C# (Java) application cont.

▶ Step 2: Create a `JoinOps` object:

```
JoinOps jops = new JoinOps ();  
jops.group_name = "MyProgram" ;
```

The public `String` field `properties` initially contains the default layers

```
Gmp:Switch:Sync:Heal:Frag:Suspect:Flow:Slender
```

▶ 16

Creating a C# (Java) application cont.

▶ Step 3: Create a Member object:

```
Member memb = new Member(conn);
```

Using the Member object, one can call the following methods:

```
// join a group with the specified options
▶ public void Join(JoinOps ops);

// leave a group
▶ public void Leave();
```

▶ 17

Creating a C# (Java) application cont.

```
// send a multicast message to the group.
▶ public void Cast(byte[] data);

// send a point-to-point message to a list of members
▶ public void Send(int[] dests, byte[] data);

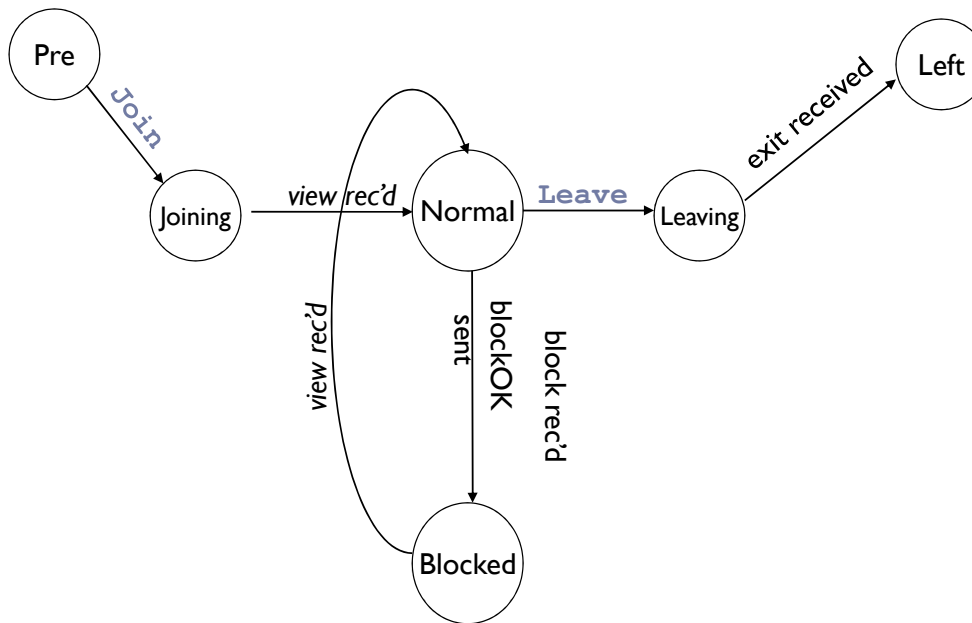
// send a point-to-point message to the specified group member.
▶ public void Send1(int dest, byte[] data);

// report group members as failure-suspected.
▶ public void Suspect(int[] suspects);

// send a BlockOk
▶ public void BlockOK();
```

▶ 18

Member: state diagram



▶ 19

Creating a C# (Java) application cont.

- ▶ **Step 4:** Upon joining and receiving a VIEW-type message, look at `msg.view`:

```
public class View {
    public int nmembers;           /** number of members in the view */
    public String version;        /** The Ensemble version */
    public String group;          /** group name */
    public String proto;          /** protocol stack in use */
    public int ltime;             /** logical time */
    public boolean primary;       /** this a primary view? */
    public String parameters;     /** parameters used for this group */
    public String[] address;      /** list of communication addresses */
    public String[] view;         /** list of endpoints in view */
    public String endpt;          /** local endpoint name */
    public String addr;           /** local address */
    public int rank;              /** local rank */
    public String name;           /** my name. This does not change
                                   throughout the lifetime of this member*/
    public ViewId view_id;        /** view identifier */
}
```

▶ 20

Example: mtalk (chat application)

```
public static void Main(string[] args)
{
    conn = new Connection ();
    conn.Connect();

    JoinOps jops = new JoinOps();
    jops.group_name = "CS_Mtalk" ;

    // Create the endpoint
    memb = new Member(conn);
    memb.Join(jops);

    MainLoop();
}
```

▶ 21

Example: mtalk cont.

```
static void MainLoop()
{
    // Open a special thread to read from the console
    Mtalk mt = new Mtalk();
    Thread input_thr = new Thread(new ThreadStart(mt.run));
    input_thr.Start();

    while(true) {
        // Read all waiting messages from Ensemble
        while (conn.Poll()) {
            Message msg = conn.Recv();
            switch(msg.mtype) {.....}
        }
        Thread.Sleep(100);
    }
}
```

▶ 22

Example: mtalk cont.

```
switch(msg.mtype) {
    case UpType.VIEW:
        // Got new View
        break;
    case UpType.CAST:
        // Got broadcast message
        Console.WriteLine("CAST (" + msg.origin + ") " +
            System.Text.Encoding.ASCII.GetString(msg.data));
        break;
    case UpType.SEND:
        // Got point to point message
        break;
    case UpType.BLOCK:
        // Last chance to send urgent message here
        memb.BlockOk();
        break;
    case UpType.EXIT:
        break;
}
```

► 23

Example: mtalk cont.

```
// A method for the input-thread
void run () {
    while(true) {
        // Parse an input line and perform the required operation
        string line = Console.ReadLine();
        lock (conn.send_mutex)
        {
            if (memb.current_status == Member.Status.Normal)
                memb.Cast(System.Text.Encoding.ASCII.GetBytes(line));
            else
                Console.WriteLine("Blocked currently, try later");
        }
        Thread.Sleep(100);
    }
}
```

► 24