EDA095
Audio and Video Streaming

Pierre Nugues

Lund University
http://www.cs.lth.se/home/Pierre_Nugues/

May 19, 2010

What is Streaming

Streaming corresponds to playing audio and video files from an Internet server.

This opposes to downloading the corresponding files.

As transmission over the Internet is not synchronous, streaming uses a buffer to store a small part of data.

This buffer dampens irregularities in Internet transmission.

Streaming imposes constraints on the network speed:

Download speed on the average should be at least as fast as playback speed.

Applications of Streaming

Applications are numerous. In addition to data:

- Internet telephony and video conferences
- Digital radios and TV: ordinary broadcast but through Internet, no frontier, no distance
- Audio and video server: on demand movies and concerts.
- Games and virtual reality
- Interaction

Triple play: data (IP), audio (VoIP), video (TVoIP).

It is made possible because of the growing availability of ADSL and fast Internet

Companies in Sweden: NetInsight, PacketFront, Kreatel (now Motorola), Marratech (now Google), etc.

The original TCP/UDP protocols are based on packet transmission and have no quality of service.

Multimedia transmission has to tackle:

Delay. Must be less than 300 ms. (Perception threshold: 150 ms)

Jitter. Packets may use different transmission paths that results into time expansion and compression

Loss. Routers may drop packets when the network load is too high

Problems with TCP/UDP

Audio and video transmission needs buffering and synchronization, possibly error correction, for instance by repeating data

UDP is just a layer to address ports. It is compatible with the requirements.
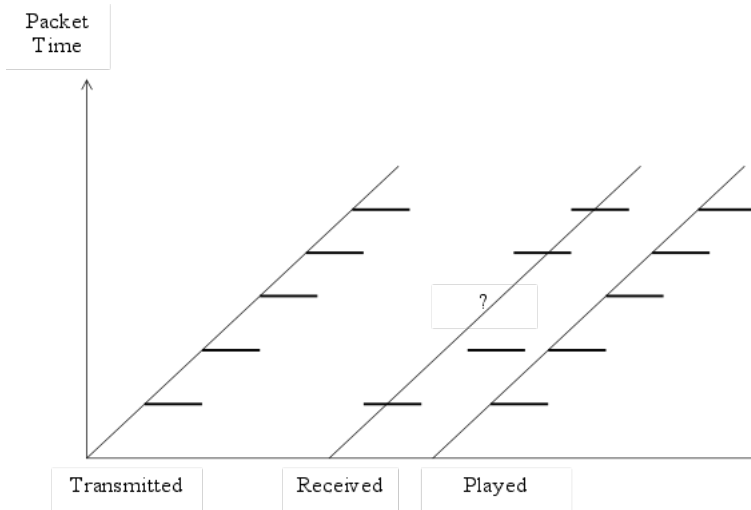
However, there is no congestion control

The datagram congestion control protocol (DCCP) attempts to fill the gap, but is not widely adopted

TCP design makes it less adapted to real-time, fast communications

Real Time Transport Protocol

The Real Time Transport Protocol (RTP)

- Identifies the content
- Adds time stamps
- Adds sequence numbers

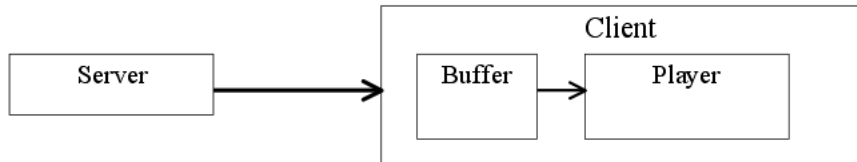RTP can be used with unicast and multicast transmission
RTP does not guarantee a real-time delivery.
RTP needs an application layer to

- Re-order packets
- Attenuate jitter
- Compensate packet loss

Architecture

Real Time Transport Protocol (RTP)

RTP is on top of UDP. It uses even ports and port $+ 1$ is for RTCP

Ethernet	IP	UDP	RTP Media content: MPEG, AIFF, and son on

The simplified RTP header structure is:

Payload type	Sequence Number	Timestamp	Sync. Source ID (SSRC)	Other fields

RTP RFCs are available here: `http://www.ietf.org/rfc/rfc3550.txt`

and `http://www.ietf.org/rfc/rfc3551.txt`
(Or through RFC Editor `http://www.rfc-editor.org/`)
Other reference: `http://csperkins.org/rtp-book.html`

- Payload type (7 bits) corresponds to the packet content: PCM = 0, DVI4=5, MPEG-2 = 33 (http://www.iana.org/assignments/rtp-parameters)
- Sequence number (16 bits) is incremented each time a packet is sent (Nothing guarantees the arrival order with UDP)
- The timestamp (32 bits) corresponds to the sampling instant of the first octet in the RTP data packet. (Clock of the sending machine)
- SSRC (32 bits) is the source of the stream. (A sending machine can have multiple sessions.)

Timestamp According to the RFC

The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations (see Section 6.3.1). The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter (one tick per video frame is typically not sufficient). The clock frequency is dependent on the format of data carried as payload [. . .]

As an example, for fixed-rate audio the timestamp clock would likely increment by one for each sampling period. If an audio application reads blocks covering 160 sampling periods from the input device, the timestamp would be increased by 160 for each such block, regardless of whether the block is transmitted in a packet or dropped as silent.

8 kHz $\rightarrow$ 125 μs, 160 samples $\rightarrow$ 20 ms

Real-Time Control Protocol (RTCP)

The real-time control protocol (RTCP) is part of the RTP protocol and defined in the same RFC.

It sends periodically control packets to all participants in the session and uses a different port, $N + 1$

It provides feedback on the quality of the data from the sender and the receiver: Statistics on packets sent, received, lost, jitter

Should be limited to 5% of the bandwidth

Commands	Description
SR	Sender report
RR	Receiver report
SDES	Source description
BYE	Quit

Encoding Formats

Telephone: 8 kHz, 1 octet (256 values), 64 kbit/s
CD: 44.1 kHz, 16 bits, stereo, 1.4 mbits/s
MP3, compressed, 96, 128, 160 kbit/s
G.732, (Internet telephony in H.323) 5.3 kbit/s or 6.4 kbit/s
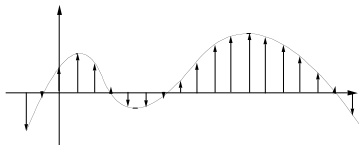MPEG-2 used in DVD, 3-6 mbit/s
Two important concepts in encoding methods:

- Bit rate can be constant (CBR) or variable (VBR)

- From an original format, encoding compress data with or without loss. Lossy compression generally results in better rates but lower quality: sometimes not perceptible. Loss rate can be a parameter of the encoding method.
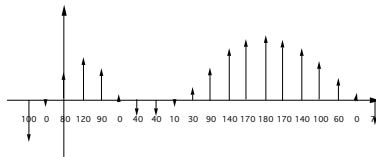
Pulse Code Modulation

Sampling   Digitization

Digitization can be linear or logarithmic: μ-law, A-law

JPEG

Compression standard for still pictures:

- Maps RGB images onto YUV coordinates (luminance and chrominance)
- Applies a Discrete Cosine Transform (DCT)
- Quantizes, which results in a compression with an adjustable loss
- Run-length encoding

MPEG

Initially sequences of frames using JPEG (I Frames)

25 or 30 frames/s

Uses temporal redundancies between images: differences between frames
(P and B Frames)

MPEG 2 has multiple possible resolutions: 720×480, 720×576, 1920×1080...

Multimedia streams contain audio and video data that are synchronized in
MPEG

(http://en.wikipedia.org/wiki/MPEG)

Codecs encode and decode original data streams.

Depending on the media you are sending, you must have the corresponding codec.

Formats supported by the RTP implementation of the Java Media Framework: http://java.sun.com/products/java-media/jmf/2.1.1/formats.html#RTPFormats

Codecs can be found from many sources as:
http://jffmpeg.sourceforge.net/

Real-Time Streaming Protocol (RTSP)

RTSP is a HTTP-like protocol to control streaming media.

It acts as a sort of remote control.

(http://www.ietf.org/rfc/rfc2326.txt)

Commands	Description
SETUP	Causes the server to allocate resources for a stream and start an RTSP session
PLAY	Tells the server to start sending data
RECORD	Records data
PAUSE	Temporarily halts a stream without freeing server resources
TEARDOWN	Frees resources associated with the stream. The RTSP session ceases to exist on the server

RTSP uses 544 as dedicated port.

RTSP servers typically use RTP for the audio/video transmission

An Example of RTSP Exchange (Modified From the RFC)

C→S SETUP rtsp://example.com/foo/bar/baz.rm RTSP/1.0
      CSeq: 1
      Transport: RTP/AVP;unicast;client_port=4588-4589
S→C RTSP/1.0 200 OK
      CSeq: 1
      Date: 23 Jan 1997 15:35:06 GMT
      Session: 12345678
      Transport: RTP/AVP;unicast;client_port=4588-4589;server_port=62
C→S PLAY rtsp://audio.example.com/audio RTSP/1.0
      CSeq: 2
      Session: 12345678
      Range: npt=10-15
. . .
C→S PAUSE rtsp://example.com/audio RTSP/1.0
      CSeq: 3
      Session: 12345678
S→C RTSP/1.0 200 OK
      CSeq: 3

Java Media Framework

The Java Media Framework (JMF) is an "API for incorporating time-based media into Java applications and applets."
It features RTP capabilities
JMF makes it possible to build media applications assembling modules.
Just follow the examples
Source available under Sun's license (SCSL)
http://java.sun.com/javase/technologies/desktop/media/jmf/
Native implementations for Windows, Linux, and Solaris
Java implementation for the others, including MacOS
No active development from Sun: latest release in 2003, MP3 update in 2004, but active list of users on the Sun forum
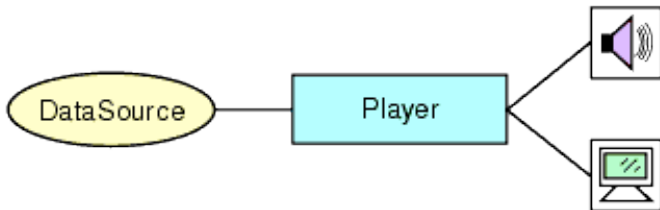
The main components are:

- The `Clock` interface keeps track of time in a media stream
- Managers: `Manager`, `PackageManager`, etc. construct `Players`, `Processors`, `DataSources`, etc
- JMF programs report their current state using `MediaEvents`. Many subclasses include `ControllerEvent` and `RTPEvent`
- `DataSources` manage media transfer. They contain the location of the media (URL), protocol, and software.
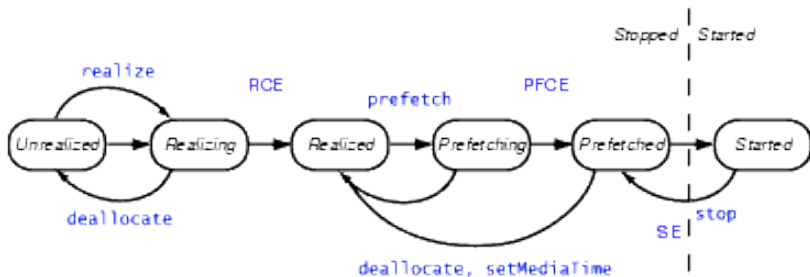
A player reads a `DataSource` and renders it

(From Sun's documentation)

(From Sun's documentation)

A First Example: A Player

MDIApp.java (http://java.sun.com/products/java-media/jmf/2.1.1/solutions/MDIApp.java)

```java
if ((url = new URL(mediaFile)) == null) {
  ...
  player = Manager.createPlayer(url);
  ...
  mplayer = player;
  mplayer.addControllerListener((ControllerListener) this);
  mplayer.realize();
  ...
  public void controllerUpdate(ControllerEvent ce) {
    if (ce instanceof RealizeCompleteEvent) {
      mplayer.prefetch();
    } else if (ce instanceof PrefetchCompleteEvent) {
```

```
if (visual != null)
  return;
if ((visual = mplayer.getVisualComponent()) != null) {
  Dimension size = visual.getPreferredSize();
  videoWidth = size.width;
  videoHeight = size.height;
  getContentPane().add("Center", visual);
} else
  videoWidth = 320;
if ((control = mplayer.getControlPanelComponent()) != nu
  controlHeight = control.getPreferredSize().height;
  getContentPane().add("South", control);
}
setSize(videoWidth + insetWidth,
videoHeight + controlHeight + insetHeight);
validate();
mplayer.start();
```
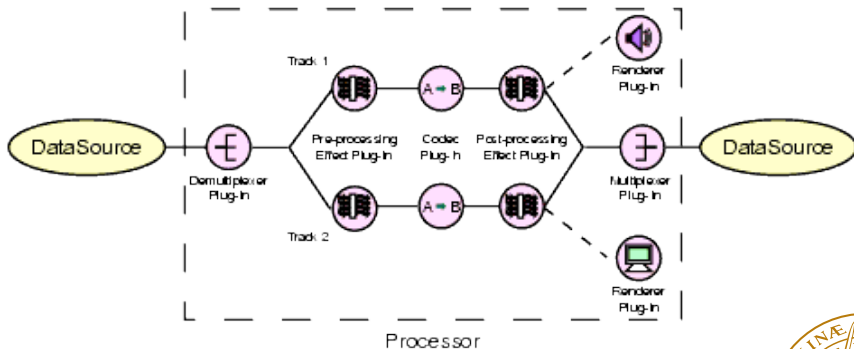
```
  } else if (ce instanceof EndOfMediaEvent) {
    mplayer.setMediaTime(new Time(0));
    mplayer.start();
  }
}
```

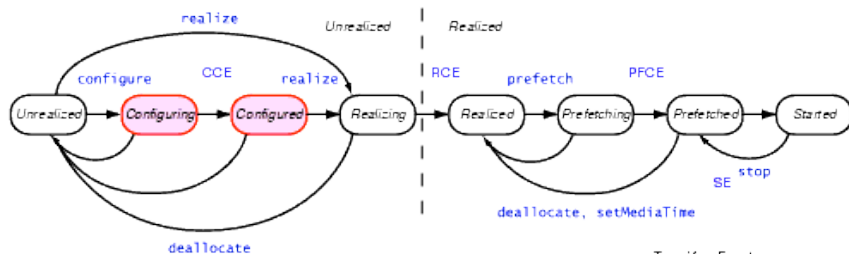A `Processor` is a specialized player that can output data to a
`DataSource` and carry out some processing

realize

Unrealized   Realized

configure   CCE   realize | RCE   prefetch   PFCE

Unrealized → Configuring → Configured → Realizing | Realized → Prefetching → Prefetched → Started

SE   stop

deallocate, setMediaTime

deallocate

Transition Events:
CCE   ConfigureCompleteEvent
RCE   RealizeCompleteEvent
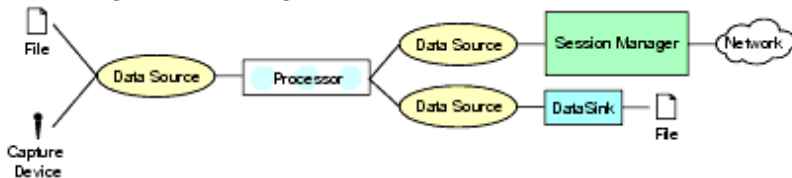PFCE PrefetchCompleteEvent
SE   StopEvent

(From Sun's documentation)
Configure connects to the `DataSource` and demultiplexes the stream.

Sending Audio

AudioTransmit.java was available from `http://java.sun.com/`
`products/java-media/jmf/2.1.1/solutions/index.html`

(From Sun's documentation)

```
Usage: AudioTransmit <sourceURL> <destIP> <destPort>
$ java -cp ../../lib/jmf.jar:. samples/AudioTransmit
  file:../../media/birds.aiff localhost 25000
```

```
private String createProcessor() {
    if (locator == null)
        return "Locator is null";
    DataSource ds;
    DataSource clone;
    try {
        ds = Manager.createDataSource(locator);
```

```
    } catch (Exception e) {
        return "Couldn't create DataSource";
    }
    // Try to create a processor to handle the input media loc
    try {
        processor = Manager.createProcessor(ds);
    } catch (NoProcessorException npe) {
        return "Couldn't create processor";
    } catch (IOException ioe) {
        return "IOException creating processor";
    }
    // Wait for it to configure
    boolean result = waitForState(processor, Processor.Config
    if (result == false)
        return "Couldn't configure processor";
```

Sending Audio

```
    // Get the tracks from the processor
    TrackControl [] tracks = processor.getTrackControls();
    // Do we have at least one track?
    if (tracks == null || tracks.length < 1)
        return "Couldn't find tracks in processor";
    boolean programmed = false;
    AudioFormat afmt;
// Search through the tracks for a Audio track
    for (int i = 0; i < tracks.length; i++) {
        Format format = tracks[i].getFormat();
        if (tracks[i].isEnabled() &&
                format instanceof AudioFormat &&
                !programmed) {
            afmt = (AudioFormat)tracks[i].getFormat();
            AudioFormat ulawFormat =   new AudioFormat(AudioF
            // afmt.getSampleRate(),
            // afmt.getSampleSizeInBits(),
```

```
        tracks[i].setFormat(ulawFormat);
        System.err.println("Audio transmitted as:");
        System.err.println("  " + ulawFormat);
        // Assume successful
        programmed = true;
    } else
        tracks[i].setEnabled(false);
}
if (!programmed)
    return "Couldn't find Audio track";
// Set the output content descriptor to RAW_RTP
ContentDescriptor cd = new ContentDescriptor(ContentDescri
processor.setContentDescriptor(cd);
// Realize the processor. This will internally create a t
// graph and attempt to create an output datasource for UI
// Audio frames.
result = waitForState(processor, Controller.Realized)
```

```
private synchronized boolean waitForState(Processor p, int sta
    p.addControllerListener(new StateListener());
    failed = false;

    // Call the required method on the processor
    if (state == Processor.Configured) {
        p.configure();
    } else if (state == Processor.Realized) {
        p.realize();
    }
    // Wait until we get an event that confirms the
    // success of the method, or a failure event.
    // See StateListener inner class
    while (p.getState() < state && !failed) {
```

```
        synchronized (getStateLock()) {
            try {
                getStateLock().wait();
            } catch (InterruptedException ie) {
                return false;
            }
        }
    }
    if (failed)
        return false;
    else
        return true;
}
```

Sending Audio (III)

```
// Creates an RTP transmit data sink. This is the easiest way
// an RTP transmitter. The other way is to use the RTPSessionM
// Using an RTP session manager gives you more control if you
// fine tune your transmission and set other parameters.
private String createTransmitter() {
    // Create a media locator for the RTP data sink.
    // For example:
    //     rtp://129.130.131.132:42050/audio
    String rtpURL = "rtp://" + ipAddress + ":" + port + "/audi
    MediaLocator outputLocator = new MediaLocator(rtpURL);

    // Create a data sink, open it and start transmission. It
    // for the processor to start sending data. So we need to
    // output data source of the processor. We also need to st
    // processor itself, which is done after this method retur
```

```
    try {
        rtptransmitter = Manager.createDataSink(dataOutput, ou
        rtptransmitter.open();
        rtptransmitter.start();
        dataOutput.start();
    } catch (MediaException me) {
        return "Couldn't create RTP data sink";
    } catch (IOException ioe) {
        return "Couldn't create RTP data sink";
    }
  return null;
}
```

Sending Video

VideoTransmit.java http://java.sun.com/products/java-media/
jmf/2.1.1/solutions/index.html

```java
for (int i = 0; i < tracks.length; i++) {
    Format format = tracks[i].getFormat();
    if (  tracks[i].isEnabled() &&
            format instanceof VideoFormat &&
            !programmed) {
        // Found a video track. Try to program it to output JP
        // Make sure the sizes are multiple of 8's.
        Dimension size = ((VideoFormat)format).getSize();
        float frameRate = ((VideoFormat)format).getFrameRate()
        int w = (size.width % 8 == 0 ? size.width :
            (int)(size.width / 8) * 8);
        int h = (size.height % 8 == 0 ? size.height :
            (int)(size.height / 8) * 8);
```
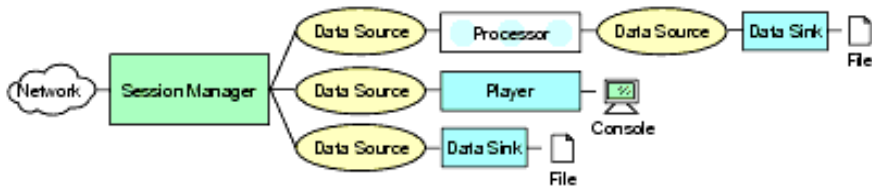
```
            VideoFormat jpegFormat = new VideoFormat(Vide
            new Dimension(w, h),
            Format.NOT_SPECIFIED,
            Format.byteArray,
            frameRate);
    tracks[i].setFormat(jpegFormat);
    System.err.println("Video transmitted as:");
    System.err.println("  " + jpegFormat);
    // Assume succesful
    programmed = true;
} else
    tracks[i].setEnabled(false);
}
```

(From Sun's documentation)
AVReceive2.java (`http://java.sun.com/products/java-media/jmf/`
`2.1.1/solutions/index.html`)
It opens RTP sessions for each track
It receives events from the RTP streams
It creates one player per stream

```
Usage: AVReceive2 <session> <session> ...
     <session>: <address>/<port>/<ttl>
$ java -cp ../../lib/jmf.jar:. samples/AVReceive2
  pierre/22222/1
```

Documentation: `http://java.sun.com/products/java-media/jmf/`
`reference/docs/index.html`
Program examples available from Sun:
`http://java.sun.com/products/java-media/jmf/2.1.1/`
`solutions/index.html`
Active discussion list:
`http://forum.java.sun.com/forum.jspa?forumID=28`
Codecs available from SourceForge:
`http://jffmpeg.sourceforge.net/`

Session Initiation Protocol (SIP)

There is no procedure to establish a session with a remote host in UDP or RTP.

SIP is a protocol to handle it. Defined by IETF in RFC:

`http://www.ietf.org/rfc/rfc3261.txt`

SIP enables to set up a call, negotiate the parameters, manage, and close the session.

Borrows many ideas from HTTP and uses UDP or TCP.

Once the session is established on port 5060, the media transmission can use RTP or something else.

A SIP Example (From the RFC)

Alice from Atlanta.com sends an INVITE request addressed to Bob's SIP URI at Biloxi.com.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>;
Content-Type: application/sdp
Content-Length: 142
```

Via contains the address (pc33.atlanta.com) at which Alice is expecting to receive responses to this request

To contains a SIP URI (sip:bob@biloxi.com) towards which the request was originally directed.

From also contains a SIP URI (sip:alice@atlanta.com) that indicates the originator of the request.

Call-ID contains a globally unique identifier for this call.

CSeq or Command Sequence contains an integer, incremented for each new request within a dialogue

Contact contains a SIP URI that represents a direct route to contact Alice. While Via tells where to send the response, Contact tells where to send future requests.

A Content Example

The session description protocol, SDP, specifies details of the connection using name-value pairs. (http://www.ietf.org/rfc/rfc4566.txt)

```
v(ersion)=0
o(wner)=bell 53655765 2353687637 IN IP4 128.3.4.5
c(onnection)=IN IP4 135.180.144.94
m(edia)=audio 3456 RTP/AVP 0 3 4 5
```

A SIP Example (From the RFC)

Alice	Proxy	Proxy	Bob
INVITE F1 →			
	INVITE F2 →		
← 100 TRYING F3		INVITE F4 →	
	← 100 TRYING F5		
		← 180 RINGING F6	
	←180 RINGING F7	←200 OK F9	
	←200 OK F10		
←180 RINGING F8			
←200 OK F11			
	ACK F12 →		
	Media		
	← BYE F13		
	OK F14 200 →		

SIP Methods

Methods	Descriptions
INVITE	Invites a session
ACK	Acknowledges
OPTIONS	Server capabilities
BYE	Closes a session
CANCEL	Cancels a pending request
REGISTER	

SIP Registrar

When the SIP client starts, it registers its location.
Proxies can find people in different places using multiple devices.

```
REGISTER sip:registrar.biloxi.com SIP/2.0
Via: SIP/2.0/UDP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@192.0.2.4>
Expires: 7200
Content-Length: 0
```

SIP Registrar (II)

C→S:

```
REGISTER sip:bell-tel.com SIP/2.0
Via: SIP/2.0/UDP pluto.bell-tel.com
To: sip:watson@bell-tel.com
From: sip:jon.diligent@bell-tel.com
Call-ID: 17320@pluto.bell-tel.com
CSeq: 1 REGISTER
Contact: sip:tawatson@example.com
```

H.323

H.323 is a competitor to SIP.

It has been promoted by the ITU – the telephone companies

Complete and in the beginning more complex then SIP

Good integration with telephone systems