Part 3: Networking Aspects

Vincent Roca and Christoph Neumann
{firstname.name}@inrialpes.fr

Planète project; INRIA Rhône-Alpes
MIPS'03, Napoli, November 2003

Outline

- introduction
- RTP/RTCP protocol
- Forward Error Correction (FEC)
- group communication services
- QoS management

Introduction – the protocols

many protocols are required by video streaming

- stream description: SDP, SMIL...
  - **describe the session and content**
- stream control: RTSP
  - **remote control the session**
- media transport: RTP
  - **send data and metadata**
- packet transport: multicast routing
  - **… or any alternative group communication service!**
  - **efficient transmission of large amounts**
- resource reservation (if any!): RSVP, DiffServ
  - **make sure the communication path offers appropriate guaranties…**
  - **…otherwise Best-Effort transmissions!**

Introduction… (cont')

we will focus on:

- RTP/RTCP
  - **used to encapsulate real time content**
  - **we discuss:**
    - RTP and RTCP overview
    - an example: RTP framing of H.261 video
- Forward Error Correction (FEC)
  - **required by many streaming techniques**
  - **we discuss:**
    - simple FEC schemes
    - small block versus large block FEC codes
    - partial reliability and FEC

Introduction… (cont')

we will focus on… (cont')

- Group communication services
  - **critical for scalability**
  - **Laurent Mathy will detail alternative group communication services**
  - **we discuss:**
    - multicast briefly
    - ALC (more or less) reliable multicast protocol
    - layered congestion control protocols
- Quality of Service
  - **required by some streaming techniques**
  - **we briefly discuss IntServ versus DiffServ**

Networking defects

- packet erasures
  - Internet is a ***Packet Erasure Channel***
  - it works on packets
  - packets can be erased (i.e. lost)
  - but a packet arriving at a receiving applications is error-free
    - **integrity is checked by physical CRC and TCP/UDP checksum**
  - several loss models (random, burst, long cut-offs)
- end to end delay is not constant (jitter)
  - usually due to buffering in routers, sometimes by the presence of several paths

Outline

- introduction
- RTP/RTCP protocol
  - RTP and RTCP
  - RTP profiles
  - RTP payload format for H.261
- Forward Error Correction (FEC)
- group communication services
- QoS management

RTP overview

- IETF Audio/Video Transport WG
  - RTPv2 RFC 3550 (July 2003)
    - **obsoletes RTPv1 (RFC 1889, January 1996)**
- Real-Time Protocol (RTP)
  - understand: « a framing protocol for real-time applications »
  - does not define any QoS mechanism for real-time delivery!
- Real-Time Control Protocol (RTCP)
  - its companion control protocol, useful to get some feedback and carry control information
  - does not guaranty anything either!

RTP overview… (cont')

based on:

- UDP
  - **TCP is not for real-time!**
  - **typical RTP packet:**

20 bytes	8 bytes	12 bytes	
IP header	UDP header	RTP header	payload

- no fixed UDP ports
  - **negotiated out of band (e.g. specified in the SDP description)**
  - **UDP port for RTCP = UDP port for RTP + 1**
- one media per RTP session (i.e. per port pair)
  - **video and audio are carried in two RTP sessions**
  - **but there are exceptions…**

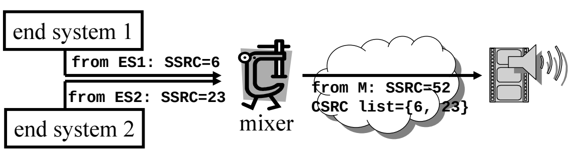
RTP overview… (cont')

design goals:

- flexible
  - provide mechanisms, do not dictate algorithms!
    - ⇒ **instantiations for H261, MPEG1/2/...**
- scalable
  - unicast, multicast, from 1 to ∞
    - ⇒ **limit RTCP overhead**
- provide all the required info/mechanisms
  - timing information for external mechanisms:
    - **intra-media synchronization: remove jitter with playout buffers**
    - **inter-media synchronization: lip-synchro between audio-video**

RTP overview… (cont')

- provide all the required mechanisms… (cont')
  - mixers
    - **a mixer may change the data format (coding) and combine several (e.g. video) streams in any manner**
    - **example: video mixer (~MCU)**



RTP overview… (cont')

two times are defined …

- RTP time
  - random initial offset (for each stream)
  - RTP timestamp present in each ***data packet***
  - increases by the time « covered » by a packet
- NTP time (or wall clock time)
  - absolute time (use Network Time Protocol format)
  - NTP timestamp present in each ***RTCP Sender Report***
  - enables inter-stream synchronization

RTP header

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|                   payload (audio, video...)                   |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           ...|      padding  |    count      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- ❍version (V) CSRC count (CC)
- ❍padding (P) marker (M)
- ❍extension (X) payload type (PT)
- ❍Sequence number incr. for each RTP packet
- ❍Sync. SouRCe (SSRC) identifies the source

RTP header… (cont')

- RTP header is at least 12 bytes
- …but it can be longer
  - ❍is if mixers are used
    - ❍**add a list of all Contributing SouRCes (CSRC), whose number is indicated by the CC field**
  - ❍is longer with some content formats
    - ❍**H.261 video transport requires an additional H.261/RTP header (4 bytes)**
    - ❍**see later…**

RTCP overview

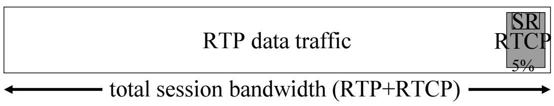
- periodic transmission of control packets
  - ❍use same distribution mechanisms as data packets (i.e. unicast or multicast)
  - ❍but there are exceptions…
    - ❍**e.g. RTP for SSM**
- several functions
  - ❍***feedback*** on the quality of data distribution
  - ❍let everybody evaluate the ***number*** of participants
  - ❍persistent transport-level canonical ***name*** for a source, CNAME
    - ❍**usually: user@host**
    - ❍**will not change, even if SSRC does!**
    - ❍**binding across multiple media tools for a single user**

RTCP overview… (cont')

- five RTCP packets
  - ❍***SR*** ***sender reports***
    transmission statistics from active senders
  - ❍***RR*** ***receiver reports***
    reception statistics from participants
  - ❍SDES source description, including CNAME
  - ❍BYE explicit leave
  - ❍APP application specific extensions

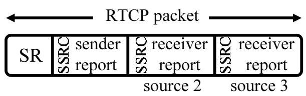
RTCP overview… (cont')

- scalability with session size
  - ❍RTCP traffic should not exceed 5% of total session bandwidth
    - ❍**requires an evaluation of number of participants**
    - ❍**then let:**
      RTCP transm. period = f (estimated number of part.)
  - ❍at least 25% of RTCP bandwidth is for source reports
    let new receivers quickly know CNAME of sources!



SR RTCP packets

- includes
  - ❍SSRC of sender identify source of data
  - ❍NTP timestamp when report was sent
  - ❍RTP timestamp corresponding RTP time
  - ❍packet count total number sent
  - ❍octet count total number sent
  - ❍followed by zero or more receiver report…
  - ❍example:
    source 1 reports, there are 2 other sources



RR RTCP packets

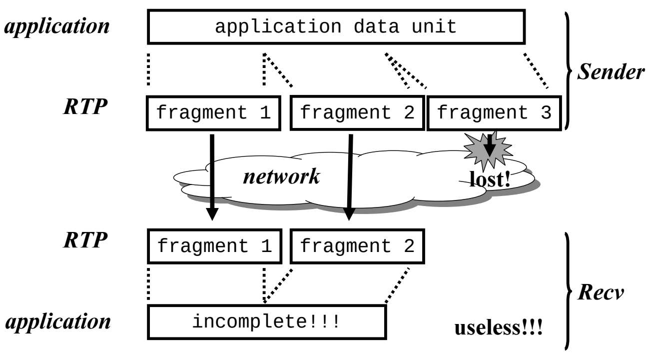
- includes
  - SSRC of source — identify the source to which this RR block pertains
  - fraction lost — since previous RR (SR) sent (= int(256*lost/expected))
  - cumul # of packets lost — long term loss
  - highest seq # received — compare losses
  - inter-arrival jitter — smoothed inter-packet distortion
  - LSR — time when last SR heard
  - DLSR — delay since last SR

RTP profiles

- RTP is generic… define a profile for each target media!
  - example: H.261 video packetization (RFC 2032)
  - must follow general guidelines
    - **"Guidelines for Writers of RTP Payload Format", RFC 2736, December 1999**
  - goal:
    - ***"Every packet received must be useful! "***
  - potential problems: packets sent over the Internet may be:
    - **lost**
    - **reordered**
    - **fragmented by IP if size > MTU (max. transm. unit)**

RTP profiles… (cont')

- Example of what must ***not*** be done!
  - loss multiplication effect due to bad framing



RTP profiles… (cont')

- the ALF (Application Level Framing) paradigm
  - Clark, Tennenhouse, *"Architectural Considerations for a New Generation of Protocols"*, SIGCOMM '90
  - idea:
    - **unit of transmission ≡ unit of control**
    - **each unit is self-sufficient and can be processed as soon as it is received**
  - if a video frame is larger than MTU, the application must define its own fragmentation mechanism so as to make each RTP/UDP/IP packet self-sufficient

RTP payload format for H.261

- H.261 generates a variable bit-rate flow
  - in practice frame sizes range from a few 10s of bytes up to 20 Kbytes
  - size of a CIF frame must not exceed 32 Kbytes
    - **GOB size ≤ 3 Kbytes**
    - **MB size ≤ 90 bytes**
    - **block size ≤ 15 bytes**
- H.261 packetization
  - ADU == MB
  - a packet contains a few ADUs (i.e. MB)
  - sometimes all MBs of a frame are in the same packet…
  - …and sometimes a frame is split in ~20 packets

RTP payload format for H.261… (cont')

- Add a H.261/RTP header to the RTP header

12 bytes	4 bytes				
RTP header	**H261/RTP header**	**MB**	**MB**	**…**	**MB**

H.261 data

  - goal is to make all packets self-sufficient

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|SBIT |EBIT |I|V| GOBN  |   MBAP  |  QUANT  |  HMVD   |  VMVD   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- **I** — **INTRA frame**
- **GOBN** — **GOB number, 0 if packet starts with a GOB header**
- **HMVD, VMVD** — **horizontal/vertical movement vector**

Some recent RTP extensions (subset)

- http://www.ietf.org/html.charters/avt-charter.html
- extensions for new services and environments
  - **"RTCP Extensions for Single-Source Multicast Sessions with Unicast Feedback",** *<draft-ietf-avt-rtcpssm-05.txt>, October 2003*
  - **"RTP Control Protocol Extended Reports (RTCP XR)",** *<draft-ietf-avt-rtcp-report-extns-06.txt>, May 2003*
  - **"The Secure Real-time Transport Protocol",** *<draft-ietf-avt-srtp-09.txt>, July 2003*
- extensions for new content formats
  - **"RTP Payload Format for Transport of MPEG-4 Elementary Streams",** *< draft-ietf-avt-mpeg4-simple-08.txt>, August 2003*
  - **"RTP Payload Format for JPEG 2000 Video Streams",** *<draft-ietf-avt-rtp-jpeg2000-04.txt>, October 2003*
  - **"RTP Payload Format for Uncompressed Video",** *<draft-ietf-avt-uncomp-video-04.txt >, October 2003*
  - **"RTP Payload Format for MPEG1/MPEG2",** *<draft-ietf-avt-mpeg1and2-mod-00.txt>, October 2003*
  - **"An RTP Payload Format for Erasure-Resilient Transmission of Progressive Multimedia Streams",** *<draft-ietf-avt-uxp-06.txt>, October 2003*

Outline

- introduction
- RTP/RTCP protocol
- Forward Error Correction (FEC)
  - introduction to FEC
  - simple forms of FEC
  - FEC codes
    - **small block versus large block FEC codes**
  - FEC and streaming
- group communication services
- QoS management

Introduction to FEC

- FEC (Forward Error Correction)
  - add some redundancy to the data flow
- reliable multicast is almost impossible without FEC
  - a single redundant FEC packet can recover many different losses at different receivers

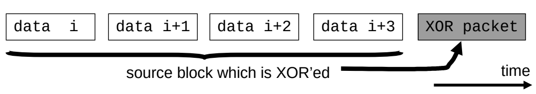
    ⇒ improves ***scalability*** by reducing the need for feedback messages and retransmissions
- and it is useful to many other applications…
  - including loss recovery in real-time flows
    - **no time to retransmit!**
- we only consider a Packet Erasure Channel

Simple forms of FEC

- packet repetition
  - trivial solution, send each packet several times
  - but too inefficient to be used for streaming
- repeat previously received data in case of an erasure
  - e.g. a missing block in a frame is replaced by the corresponding block in the previous frame
  - takes advantage of the redundant nature of the audio/video content
  - no transmission overhead
  - loss of information ⇒ only for audio/video streams

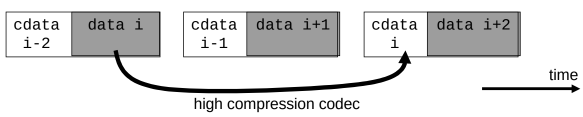
Simple forms of FEC… (cont')

- XOR of packet streams
  - every k packets, add a k+1 packet which is the XOR of the previous k packets
  - simple scheme, well suited to packet streams
  - but limited erasure recovery capabilities
    - **1 loss per block of k packets**
  - increased latency
    - **k packets of block must be received to recover an erasure in the block**



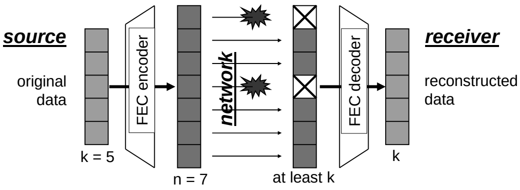
Simple forms of FEC… (cont')

- repeat with compressed information
  - each packet contains fresh data + lower quality data from a previous packet
    - **e.g. fresh audio uses PCM encoding, low quality audio uses LPC encoding**
  - easy way to counter random erasures…
  - but not long bursts of erasures
  - popular for audio content
  - loss of information ⇒ only for audio/video streams



FEC codes

- high error recovery power
  - **Sender: uses FEC (k, n)**
    - for ***k*** original data symbols, add ***n-k*** FEC encoded redundant symbols
    - ⇒ total of ***n*** symbols (or packets) sent
  - **Receiver:**
    - as soon as it receives ***any k*** symbols out of the n, it reconstructs the original k symbols



FEC codes… (cont')

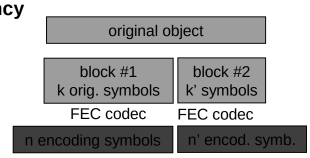
- classification based on the (k, n) parameters
  - small block FEC codes (small k)
    - **Reed-Solomon**
  - large block FEC codes (large k)
    - **LDPC, LDGM, Tornado ©**
  - expandable FEC codes (large k and n)
    - **LT ©, Rateless code**
  - RFC 3453 gives some more info, but with a ***very partial***, Digital Fountain centric eye !

Small block FEC codes

- key features
  - e.g. Reed-Solomon codes (RSE) [Rizzo97]
  - (k, n) with a k parameter limited to a few tens for computational reasons
    - **in practice: 0 ≤ k ≤ n ≤ 255**
  - it's an MDS code (Minimum Distance Separation)
    - **any set of exactly k packets is sufficient for decoding**
  - high quality open-source implementation available
    - **see Luigi Rizzo's home page**

Small block FEC codes... (cont')

- RSE in practice
  - binary result
    - **if r ≥ k packets are received, decoding is possible**
    - **otherwise no decoding at all, and only the source symbols in the r packet received can be useful**
    - **not very flexible!**
  - leads to inefficiencies with large objects
    - **large objects must be split into several blocks**
    - **limits the correction capability of a FEC symbol**
    - **limits the global efficiency**



Large block FEC codes

- why « large block » ?

  large block == "k amounts to 10,000s or more packets"

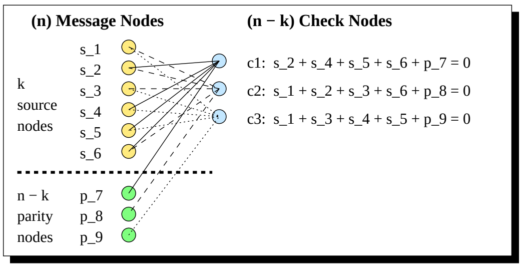
  - since a parity packet can recover an erasure only in its block, the optimal solution is to have the file encoded as a ***single block…***
  - …which is only possible if large blocks can be used !

Large block FEC codes… (cont')

- key features
  - e.g. LDPC, LDGM codes
  - (k, n) with a **very large k**
  - but n is limited (e.g. n = 2k)
  - decoding requires (1+ε)k, i.e. a bit more than k symbols
  - high-speed encoding/decoding
    - **237 Mbps encoding with our LDGM-staircase codec, PIII 1GHz, 10MB block, 5MB parity**
  - best codes (e.g. Tornado ©) are patented, but LDPC/LDGM are good enough and patent-free
  - open source implementation available:
    - **http://www.inrialpes.fr/planete/people/roca/mcl/**

Low Density Generator Matrix (LDGM)

- fundamentals
  - based on XOR
  - two representations: ***bipartite graph*** and ***matrix***
  - notations:
    - **s_i are source packets, p_i are FEC packets, c_i are check (A.K.A. constraint) nodes (not sent)**



LDGM… (cont')

- dual (k x n) matrix representation:

$$[H \mid Id_3] = \begin{array}{c} \begin{array}{cccccc|ccc} s_1 & & .. & & & s_6 & p_7 & .. & p_9 \end{array} \\ \left(\begin{array}{cccccc|ccc} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \end{array} \begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array}$$

e.g. it says that for c_1: $s_2 + s_4 + s_5 + s_6 + p_7 = 0$

- encoding
  - encoding is simple since a + a = 0 (bitwise XOR)
  - each p_i is the sum of the source symbols in the associated constraint equation

    e.g. for c_1: $p_7 = s_2 + s_4 + s_5 + s_6$
  - simple and highly efficient: *O(n-k)*

LDGM… (cont')

- iterative decoding algorithm
  - solve a system of linear equations using a trivial algorithm:

    e.g. for c_1: $s_{2\,(missing)} + s_4 + s_5 + s_6 + p_{7\,(known)} = 0$

    then you have: $s_2 = s_4 + s_5 + s_6 + p_7$
  - ***step 1:*** so, you look for equations (set of constraints) where all the variables are known except one, and if one such equation exists you directly find the missing variable.
  - ***step 2:*** each time a packet is received or recovered, you replace its value in the equations, and go to step 1.

LDGM staircase

- principles
  - replace the identity matrix by a "Staircase" matrix

$$[H \mid Staircase_5] = \begin{array}{c} \begin{array}{cccccc|ccccc} s_1 & & .. & & & s_6 & p_7 & .. & & & p_{11} \end{array} \\ \left(\begin{array}{cccccc|ccccc} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right) \end{array}$$

  - encoding:
    - **calculate the first parity packet: p7 = s2 + S4 + S5**
    - **calculate the remaining parity packets, in the order:**

      p8 = p7 + …

      p9 = p8 + …, etc.
  - this code has a better erasure recovery property, because parity packets are themselves protected

LDGM staircase… (cont')

- LDGM-staircase in practice
  - it introduces a ***small decoding inefficiency***
    - **(1+ε)k packets must be received for decoding to finish, where $\varepsilon \geq 0$**
    - **k = 10000, n-k = 5000, we found:**

      average_ε = 6.9%, worst_ε = 7.7%
  - but it is ***highly efficient***
    - **high encoding/decoding speed**
    - **blocks of several tens of MB**
  - and is excellent for ***partially reliable*** sessions
    - **the decoding process can be stopped at any time**
    - **if r < (1+ε)k packets are received, some erasures may still be recovered**
    - **≠ RSE**

FEC and video streaming

- we've seen some theoretical aspects…
  - but we only covered a subset of FEC
  - other codes exist
    - **e.g. rate-less codes**
    - **e.g. for symmetric binary channels**
- …we'll see some practical aspects later
  - within ALC, which can be used for video streaming (cf. SVSoA, part 4)
  - within other streaming schemes
  - for implementing an unequal erasure protection scheme

Outline

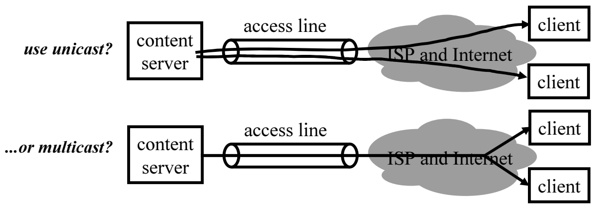
- introduction
- RTP/RTCP protocol
- Forward Error Correction (FEC)
- group communication services
  - multicast (briefly)
  - reliable multicast protocols and ALC
  - congestion control protocols for ALC and other layered approaches
- QoS management

Introduction to Multicast

- definition
  - group communications means...
    - **1 → n** — **e.g. file distribution**
    - **as well as n → m** — **e.g. video-conference**
  - ideally a physical link sees at most a single copy of a packet
  - multicast routing is one way of implementing this group communication service

Introduction to Multicast… (cont')

- why should we use multicast?
  - scalability...
    - **scales to an unlimited number of users**
  - reduced costs...
    - **cheaper equipment and access line**
  - increased speed...
    - **increases the delivery speed**



Introduction to Multicast… (cont')

- why should we use multicast… (cont')
  - useful for discovery protocols (RFC 1112)

```
224.0.0.0 - 224.0.0.255 (224.0.0/24) Local Network Control Block
-----------------------------------------------------------------
224.0.0.0    Base Address (Reserved)          [RFC1112,JBP]
224.0.0.1    All Systems on this Subnet       [RFC1112,JBP]
224.0.0.2    All Routers on this Subnet       [JBP]
224.0.0.4    DVMRP Routers                    [RFC1075,JBP]
224.0.0.5    OSPFIGP OSPFIGP All Routers      [RFC2328,JXM1]
224.0.0.6    OSPFIGP OSPFIGP Designated Routers [RFC2328,JXM1]
224.0.0.7    ST Routers                       [RFC1190,KS14]
224.0.0.8    ST Hosts                         [RFC1190,KS14]
224.0.0.9    RIP2 Routers                     [RFC1723,GSM11]
224.0.0.10   IGRP Routers                     [Farinacci]
224.0.0.11   Mobile-Agents                    [Bill Simpson]
224.0.0.12   DHCP Server / Relay Agent        [RFC1884]
224.0.0.13   All PIM Routers                  [Farinacci]
…
```

Introduction to Multicast… (cont')

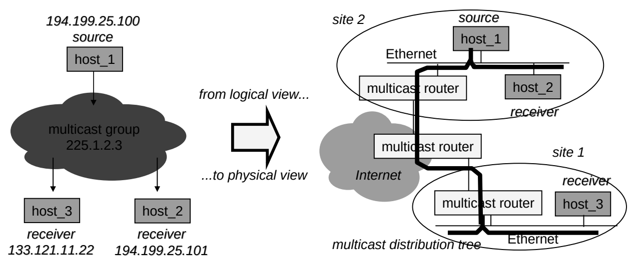
- group identification
  - a group is identified by a class D IPv4 address
    - 224.0.0.0 to 239.255.255.255
  - or a IPv6 address with prefix FF::/8

```
    8          4      4            112 bits
11111111  | 000T | scope |         group ID
```

    - "T" bit identifies transient addresses
    - "scope" the packet scope
  - a group address is an ***abstract notion***
    - **does not identify any host!**

Introduction to Multicast… (cont')

- from logical to physical views



Introduction to Multicast… (cont')

- local-area multicast
  - use the potential diffusion capabilities of the physical layer (e.g. Ethernet)
    - `Ethernet mcast addr = 01:00:5e:00/25 + least significant 23 bits of IP mcast addr`
    - **enables network card level filtering**
    - **works with both hubs and switches**
  - efficient and straightforward

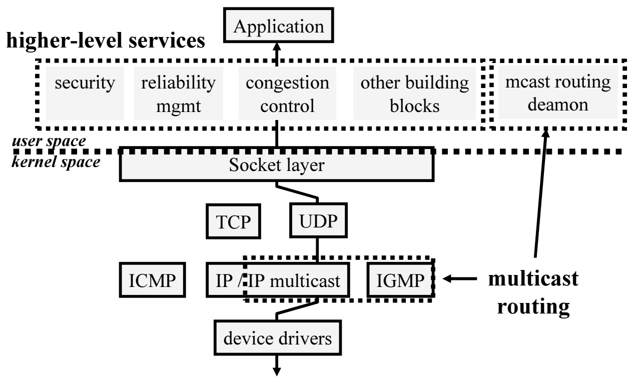
Introduction to Multicast… (cont')

- wide-area multicast
  - requires to go through multicast routers...
    - **e.g. DVMRP, PIM-DM, PIM-SM, PIM-SSM, etc.**
  - requires routers to be informed of local receivers
    - **goal of IGMP**
  - multicast routing in the same administrative domain is simple and efficient
  - inter-domain multicast routing is complex and not always operational…

Introduction to Multicast… (cont')

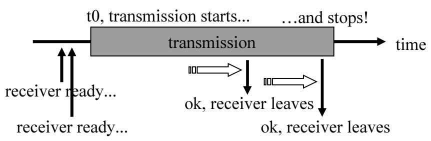
- well, sometimes there's no multicast routing at all
  - quite frequent !
  - see Laurent Mathy MIPS'03 tutorial:
    - ***"Group Communication Routing Services for Multimedia in the Internet"***
  - and/or our common paper:
    - **<advertisement>**
    - ***A. El-Sayed, V. Roca, L. Mathy*, ``A survey of Proposals for an Alternative Group Communication Service'', IEEE Network magazine, January/February 2003.**
    - **</advertisement>**

Multicast and the TCP/IP layered model



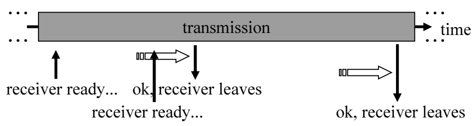
Three delivery modes

- *model 1: push* delivery
  - *sender oriented*
  - synchronous model where delivery is started at t0
  - usually requires a fully reliable delivery, but with a limited number of receivers



Three delivery modes... (cont')

- *model 2: on-demand* delivery
  - *receiver oriented*
  - popular content (video clip, software, update, etc.) is continuously distributed in multicast
  - users arrive at any time, download, and leave
  - possibility of millions of users, no real-time constraint



Three delivery modes... (cont')

- *model 3: streaming* (e.g. for audio/video)
  - long-lasting data flow
  - receivers arrive at any time, usually listen for a long time
  - requires real-time, semi-reliable delivery
  - large amount of data is sent

Reliable Multicast Transport Protocols

- "reliable" means
  - either fully reliable (useful for file delivery)
  - or partially reliable (e.g. ALC)
  - often depends on the way the protocol is used!
    - **e.g. ALC in on-demand mode offers a fully reliable service**
    - **ALC in push mode only offers a partially reliable service**
- a complex problem
  - not NP-complete… but at least extremely complex

Reliable Mcast Transport Protocols... (cont')

- "One size does not fit all"
  - *"requirements"* x *"conditions/problems"* matrix is too large for a single solution!!!
  - define ***Building Blocks*** (BB)
    - **logical, reusable component**
    - **used by the PI**
    - **example: Forward Error Correction (FEC) BB**
  - define ***Protocol Instantiations*** (PI)
    - **non reusable**
    - **glue between the various BBs**
    - **provides an operational solution**

Reliable Mcast Transport Protocols... (cont')

- single layer: NORM
  - for small to medium sized groups
  - simplicity, uses ACK / NACK
    - **Internet Draft under progress**
    - **SRM, PGM belong to that category**
- layered approach: ALC
  - for all sizes of groups, unlimited scalability
  - uses layered transmission
    - **RFC 3450, RFC 3451, RFC 3452**

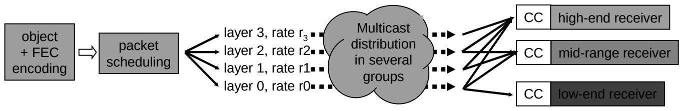
The Asynchronous Layered Coding (ALC) PI

- RFC 3450, RFC 3451
- offers unlimited scalability (no feedback)
- supports receiver heterogeneity
- support ``push'', ``on-demand'', and ``streaming'' delivery modes
- suited to the distribution of popular content
- massive use of pro-active FEC

- building blocks required by the ALC PI
  - LCT (glue between BBs + header definition)
  - FEC
  - layered congestion control (FLID-SL, WEBRC)
  - … e.g. security

The ALC PI... (cont')

- how does it work?
  - multi-rate transmissions, over several multicast groups, one per layer
  - the congestion control BB (e.g. FLID-SL) tells a receiver when to add or drop a layer



The ALC PI... (cont’)

- how does it work... (cont’)
  - mix in a **random** manner all the data+FEC packets and send them on the various layers
  - required to counter losses and random layer addition/removal
  - more intelligent organizations are possible
    - **and can avoid duplications**
  - …but only work in an ideal world!
    - **in practice losses, layer dynamic, layer de-synchronization lead to catastrophic performances!!!**

The ALC PI... (cont’)

- a transmission approach completely different from NORM
- file transmission with NORM

```
source recvs:                   NAK(2)                NAK(4)
source sends: 0  1  2  3  4  5  6  FEC1 7  8  9  10  11  FEC2 12  13  14  END
```

- file transmission with ALC (just an example!)

```
              Layer 3  F3 F12 F0 F1 F4 F11 F6 F5 F14 F7 F8 F2 F9 F10 F13 END
              Layer 2  2  4   10 8  5  9   11 14  7  3  0  12  1  6  13  END
              Layer 1  F12 F9 F2 F1 F10 F7 F6 F4 F13 F3 F5 F11 F14 F0 F8 END
source sends: Layer 0  11  2  4  9  0  13  10  7  8  1  3  14  5  12  6  END
```

The ALC PI... (cont’)

- what is ALC really good at ?
  - on-demand delivery mode
    - **yes, this is the only RM solution supporting it**
  - streaming delivery mode
    - **yes, partial reliability is possible too**
  - push delivery mode
    - **no for the general case, yes when there is no feedback channel (e.g. satellite)**

The ALC PI... (cont’)

- what is ALC really good at… (cont’)
  - scalability
    - **yes, this is the only RM solution having an unlimited scalability**
  - heterogeneity
    - **yes, this is the only RM solution supporting receiver heterogeneity**
  - robustness
    - **yes, reception can be stopped and restarted several times without any problem**
    - **a source is never impacted by the receiver behavior, neither are other receivers**

Congestion Control protocols

- general goals of congestion control
  - be ***fair*** with other data flows (be “TCP friendly”)
    - **should a multicast transfer use as much resource as a TCP connection or n times as much ?**
    - **no single definition**
    - **be responsive to network conditions**
  - be ***stable***, i.e. avoid oscillations
  - use network resources ***efficiently***
    - **if only one flow, then use all the available bandwidth**

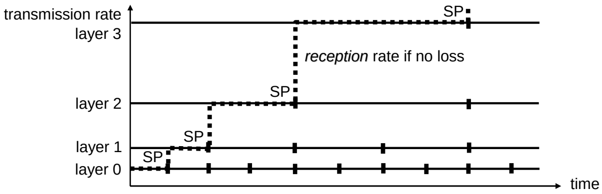
Congestion Control protocols... (cont’)

- single layer versus layered transmissions
  - two completely different schemes
  - single layer
    - ***sender oriented***
    - **transmission rate/window are based on ACK / NACK feedbacks**
    - **used by NORM**
    - **e.g. PGMCC, TFMCC**
  - layered ← focus here…
    - ***receiver oriented***
    - **based on losses experienced**
    - **used by ALC**

Layered Congestion Control protocols

- RLC (Receiver Driven Layered Cong. Ctrl)
  - add synchronization points (SP)
    - **adding a layer is only possible at a SP if no loss has been experienced before**
    - **exponential spacing of SP among the layers**

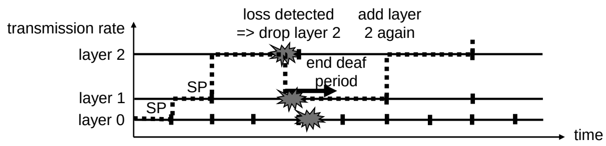
      ⇒ more difficult to add higher layers than lowers



Layered Congestion Control… (cont')

- RLC… (cont')
  - in case of error, drop the highest layer immediately
  - because of IGMP leave latency, after dropping a layer, wait some time before measuring packet loss again

    ⇒ deaf period



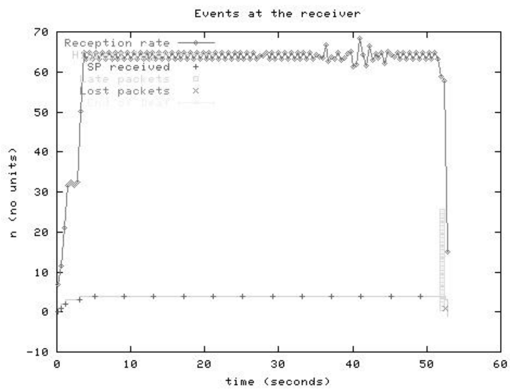
Layered Congestion Control… (cont')

- RLC limitations
  - limited by IGMP leave latency (a few seconds)
  - only adapts to packet loss, not to RTT

    different from TCP where: $a \approx \frac{c}{R\sqrt{p}}$
  - coarse transmission rate distribution
    - **power of 2 distribution to mimic TCP behavior after a loss (divide exp./linear threshold by 2)**
    - **minimum and maximum rate are fixed, the number of intermediate values too**
    - **cannot adapt to the fair TCP share precisely**
  - introduces instability
    - **periodic periods of congestion**

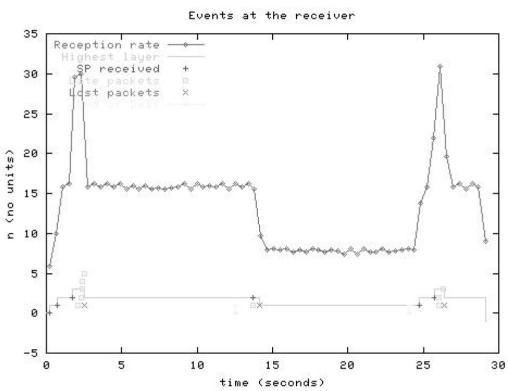
Layered cong. control : an example

- ALC session, receiver events, no loss



Layered cong. control : an example… (cont')

- ALC session, receiver events, with losses



Layered Congestion Control… (cont')

- Other more older and less efficient (!) protocols exist...
  - RLM (Receiver Driven Layered Multicast)
    - **RLM: McCanne, Jacobson, SIGCOMM'96**
    - **same general spirit as RLC**
    - **no time to detail it…**

Layered Congestion Control… (cont')

- Other more efficient protocols exist...
  - FLID-SL (Fair Layer Increase/Decrease - Static Layering)
    - **similar to RLC, without SP, with explicit timing**
  - FLID-DL (Dynamic Layering)
    - **completely different approach**
    - **behaves better than RLC/FLID-SL that are limited by IGMP leave latency**
  - WEBRC
    - **uses the same idea of dynamic layering as FLID-DL**
    - **improves throughput estimation**
    - **but leads to high IGMP/Routing protocol signaling and dynamic**
    - **probably the best solution today...**

Outline

- introduction
- RTP/RTCP protocol
- Forward Error Correction (FEC)
- group communication services
- QoS management

QoS management

- two possible approaches
  - improved service, no guaranties ⇒ *DiffServ*
  - guaranteed service ⇒ *IntServ*
- requires
  - a contract (Service Level Agreement, SLA)
    - **the user expresses its wishes and requirements**
  - admission control
    - **check that resources are in line with the user wishes**
  - signaling mechanisms (e.g. RSVP) with IntServ
    - **synchronize all routers, reserve resources**
  - traffic policing
    - **check the traffic conforms to the contract**
  - traffic control within routers (e.g. WFQ)

QoS management… (cont')

- no time to go into the technical details during this tutorial!
- general solution: DiffServ
  - simple
  - scalable
  - suited to many situations and needs
- specific solution: IntServ
  - for critical applications
    - **tele-medecine, large distributed simulations, etc.**
  - uses a dedicated backbone
  - use MPLS instead ?

QoS management… (cont')

- QoS is sometimes assumed by academic streaming proposals
  - e.g. to protect the base layer of a scalable video stream
  - assuming a large scale deployment of IntServ is not realistic… technically and economically
  - DiffServ will probably be commercially offered by ISPs sooner or later…