ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Multimedia Technology

Section # 20: Best effort service

Instructor: George Xylomenos

Department: Informatics

Contents

- Limits of best effort
- Dealing with delay
- Dealing with loss

Limits of best effort

Class: Multimedia Technology, **Section #20:** Best effort service

Instructor: George Xylomenos, **Department:** Informatics

An example app (1 of 3)

- Voice over IP
  - Classic (and simple) media app
- Alternates between speaking and silence
  - Each speaker must pause to listen to others
  - We only need to transmit when speaking
- Speech becomes audio samples (8 KHz @ 8 bit)
  - PCM coding at 64 Kbps
  - No compression to avoid delays

An example app (2 of 3)

- Samples encapsulated in packet
  - 20 byte IP header
  - 8 byte UDP header
  - 12 byte RTP header
- Assume 160 bytes of samples sent every 20 ms
  - Header overhead is 20% (without Ethernet/WiFi!)
  - Smaller period -> larger overhead
  - Larger period -> larger delay

An example app (3 of 3)

- Where does delay come from?
  - Delays in the network
    - Processing, queueing, transmission, propagation
  - Delays in the computer
    - Consider the first sample of a packet
    - It waits for 20 ms for the other 159 samples
    - This is called packetization delay
    - Optional: compression/decompression
    - Which induces additional delay

Best effort (1 of 2)

- IP offers a best effort service
  - In each node the packet is queued
  - Queue length is unpredictable
    - Queueing delay is unpredictable
  - The queue may overflow due to load
    - The packet is dropped and never arrives
  - Packets may also be reordered
    - Due to dynamic packet routing

Best effort (2 of 2)

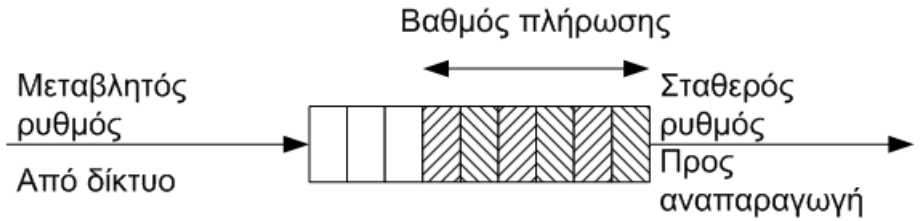
- How good is best effort?
  - Most packets do arrive
    - But not all, due to congestion
  - We rarely get corrupted packets
    - Ethernet and WiFi drop them
  - Sometimes they do arrive out of order
  - Delay is very unpredictable
    - Delay jitter is also unpredictable

Dealing with delay

Class: Multimedia Technology, **Section #20:** Best effort service

Instructor: George Xylomenos, **Department:** Informatics

Jitter removal (1 of 2)



- Network delays are unpredictable
  - Even without TCP!
- But media playback needs constant rate
  - Otherwise, the user is annoyed
- De-jittering buffer between arrival and playback
  - Initially wait for the buffer to fill
  - Then playback from buffer in constant rate

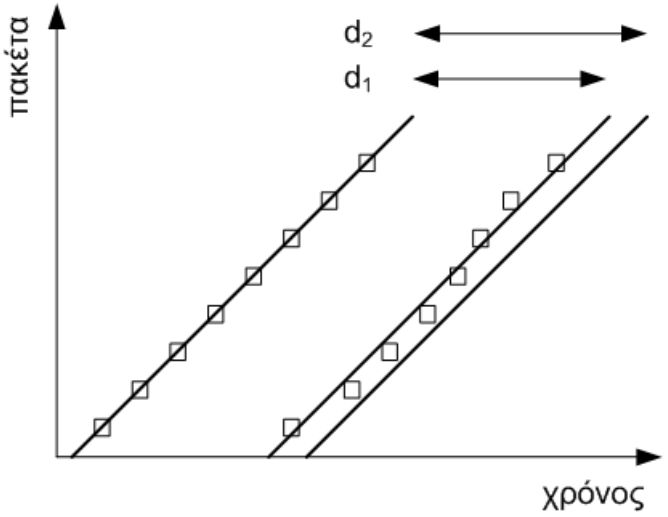
Jitter removal (2 of 2)

- Media is played back after a delay
  - To ensure each piece is available
  - How long should we wait?
    - This also dictates the size of the buffer
  - Fixed or dynamic delay?
- Policies based on RTP headers
  - Sequence No: increased for every packet
  - Timestamp: time first sample was produced

Fixed playback delay (1 of 2)

- Fixed playback delay
  - For a packet with a timestamp t
  - We play it back at time t+d
  - Assumption: end-to-end delay is < d
    - Otherwise, the packet misses its deadline
  - Lost packets are ignored
    - We just leave a gap during playback
  - What is a good estimate for d?

Fixed playback delay (2 of 2)



- Left line: small d
  - Higher loss / lower latency
- Right line: large d
  - Lower loss / higher latency

Adaptive playback delay (1 of 4)

- Adapting delay dynamically
  - Continuously adapt the delay
  - Tradeoff loss for latency
  - Based on TCP RTO algorithm
- Adaptive delay algorithm
  - t_i = timestamp of packet i
  - r_i = time packet received
  - p_i = time played back
  - r_i-t_i = transmission delay

Adaptive playback delay (2 of 4)

- Estimation of average delay d_i

$$d_i = \alpha d_{i-1} + (1-\alpha)(r_i - t_i)$$

- Smoothing constant α in [0,1]
  - Large α: more smooth changes in d_i
  - Small α: more abrupt changes in d_i
- Estimation of average variance v_i

$$v_i = \beta v_{i-1} + (1-\beta)\,|\,r_i - t_i - d_i\,|$$

  - Constant β works like α but for variance

Adaptive playback delay (3 of 4)

- Calculate d_i and v_i for every packet
  - But only adapt delay after a silent period
  - This prevents gaps in the audio
- Time to playback first packet in period

$$d = d_i + \gamma v_i$$

  - γ: small positive constant (say, γ = 4)
  - We add γ times the variance to the delay
  - Most packets will arrive on time

Adaptive playback delay (4 of 4)

- Time to playback next packets

$$p_j = t_j + d$$

  - We do not change the delay again
- How do we know a new talk period starts?
  - No loss:
    - Timestamp grows by > 20 ms
  - With loss:
    - Sequence no grows by > 1
    - Not really silence, but a gap works just as well!

Fixed or adaptive?

- Fixed is much simple but inflexible
  - Either wait a lot in the beginning
  - Or risk stalls during playback
- Adaptive follows network conditions
  - But it needs an adaptation point
  - In voice we have natural pauses
  - In other audio we may wait for loss or stall
  - What if audio and video are synchronized?

Dealing with loss

Class: Multimedia Technology, **Section #20:** Best effort service

Instructor: George Xylomenos, **Department:** Informatics

What does loss mean?

- For playback, loss means the packet is late!
  - Some packets will never arrive
    - May have been corrupted during transmission
    - May have been dropped due to congestion
  - Some packets arrive too late
    - After the playback delay
    - We cannot wait for them
  - How can we hide such losses?

Retransmissions

- If the packet is not on time, we retransmit
  - Various retransmission schemes (ARQ)
  - Positive or negative acknowledgments
    - Positive: every arriving packet ACKed
    - Negative: only missing packers NACKed
  - Retransmit only when needed
  - Very unpredictable delay
    - Retransmission costs at least one RTT

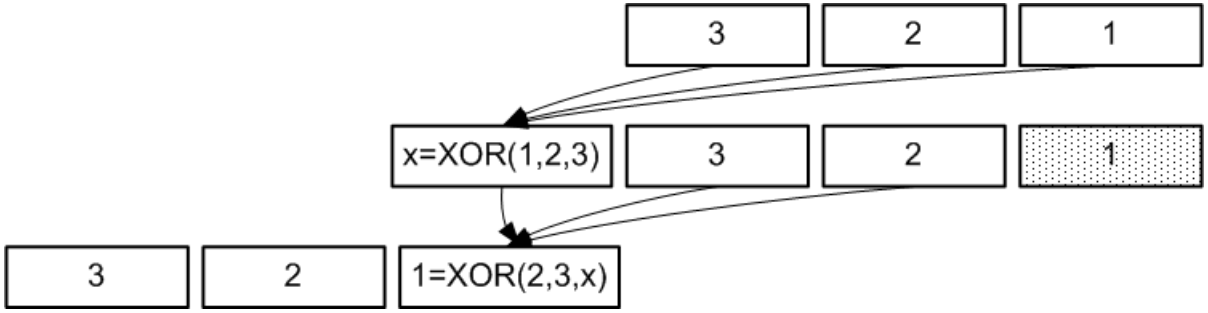
Forward error correction

- Forward error correction (FEC)
  - We add redundancy in the transmitted stream
    - Transmission rate is increased
    - May fully or partially mask losses
  - Alternative to retransmissions
    - No need for feedback from receiver
  - Wasteful when there is no loss
    - The overhead is sent in all cases

Parity packets (1 of 2)

- Parity: simple FEC scheme
  - Every n packets, send a FEC packet
    - Consists of a bitwise XOR of the n packets
    - This is basically bitwise parity
  - Corrects one lost packet out of n
    - Just bitwise XOR the n-1 received and the FEC
    - Need to know loss probability to set n
  - Some delay due to recovery
    - We must receive all packets to XOR them

Parity packets (2 of 2)

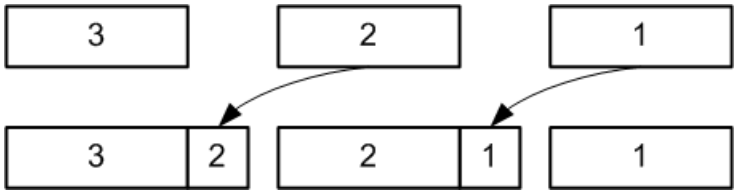


- Overhead: 1/(n+1)
  - Larger n means less overhead
- Maximum delay: n
  - Occurs if we lose first packet
    - So, we normally always wait for n packets
  - Smaller n means less delay n
- Recovery: exactly 1 loss per n packets

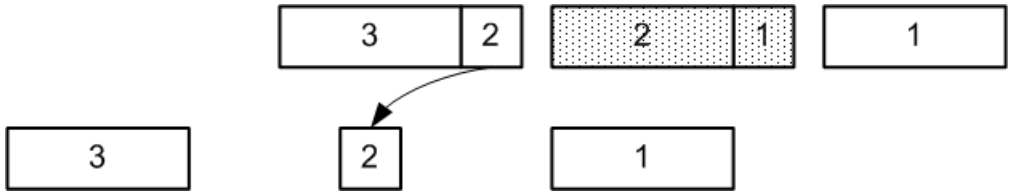
Calculating overhead

- Say that we use the XOR packets
  - For n data packets, 1 parity packet
- Two ways to express overhead
  - Over the original flow: 1/n
  - Over the total flow: 1/(n+1)
  - Not the same!
  - Should clarify what we mean by overhead

Redundant flow (1 of 2)



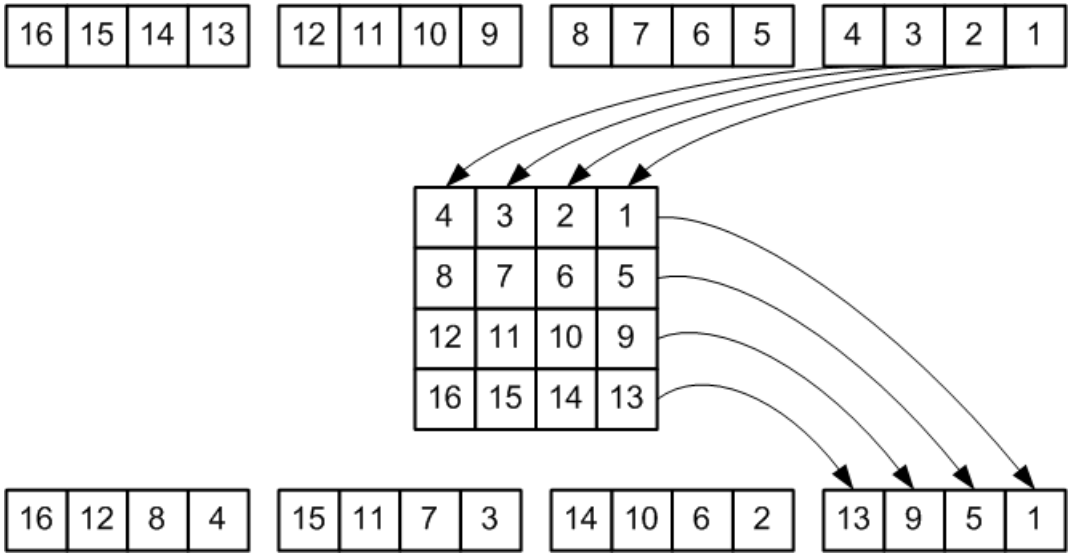
- Send extra flow with reduced quality
  - Original: G.711 at 64 Kbps
  - Redundant: G.729 at 8 Kbps
  - Packet i: original i + i-1 of redundant
  - Redundant part can recover from a single loss



Redundant flow (2 of 2)

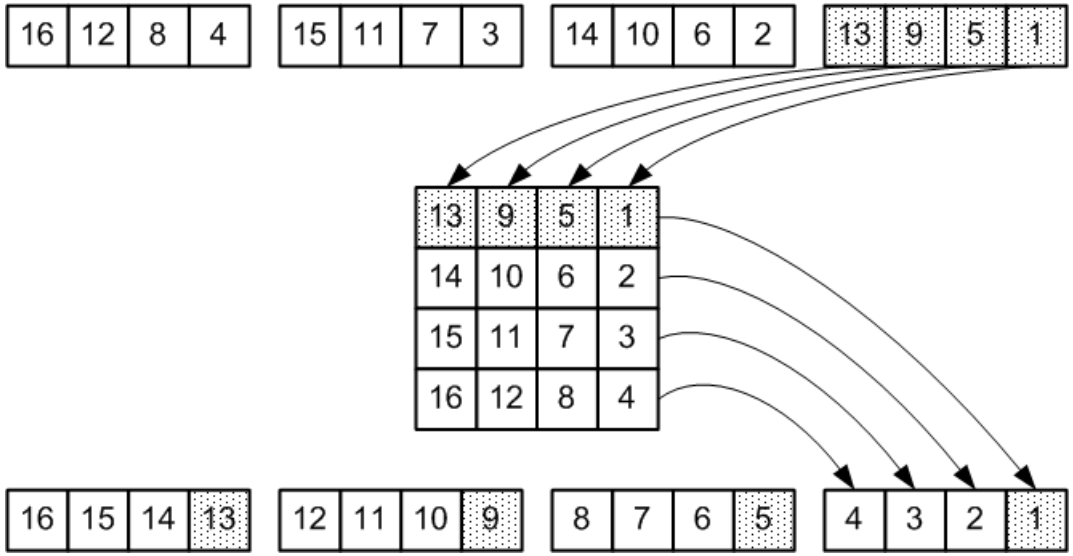
- Cause variable playback quality
  - Good quality for rare losses
  - Small recovery delay (1 packet)
  - Increases transmission rate
    - Depending on quality of redundant flow
  - Generalized to multiple redundant flows
    - Lower quality version of two previous packets
    - Can handle to continuous losses

Interleaving (1 of 4)



- Data interleaving
  - We break down the audio packets into chunks
  - We reorder the chunks before transmission
    - Contiguous chunks are transmitted far away

Interleaving (2 of 4)

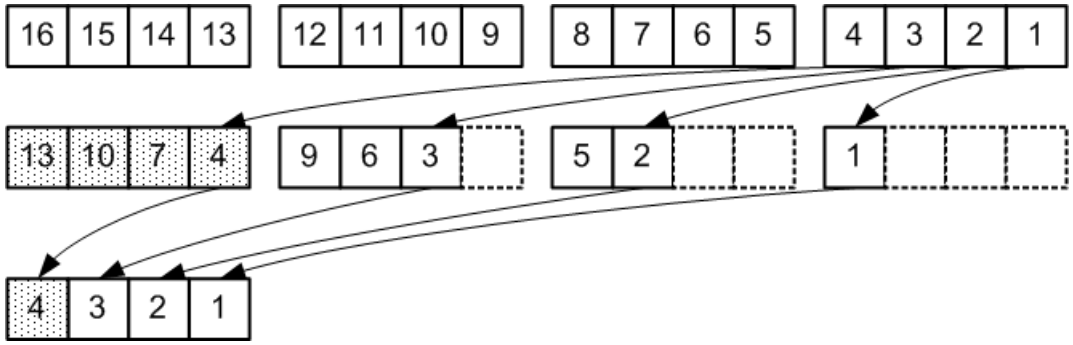


- Data interleaving
  - We reverse this at the receiver
  - No change in data rate
  - No change in loss rate

Interleaving (3 of 4)

- Why bother with interleaving?
  - It may reduce the impact of losses
  - By itself, it does not recover from loss
  - But it turns large losses to small ones
    - Can be combined with parity packets
  - But it also increases delay
    - We need to wait before we rearrange packets
    - We need to wait before we reassemble packets

Interleaving (4 of 4)



- Rolling interleaving
  - In regular interleaving we deal with N packets
  - In rolling we do not wait for the entire block
  - Packet fragments added to the next N packets
    - N-1 packet times delay at the transmitter
    - No additional delay in the receiver

Error correcting codes

- Block codes (used at higher layers)
  - Handle a fixed block of data
  - Calculate error recovery for the block
  - Decoding when the entire block is received
- Convolution codes (used at lower layers)
  - Data and error correction interleaved
  - Create from a window in the data
  - Decoding occurs as we receive data

End of Section # 19

Class: Multimedia Technology, **Section # 19:** Streaming

Instructor: George Xylomenos, **Department:** Informatics