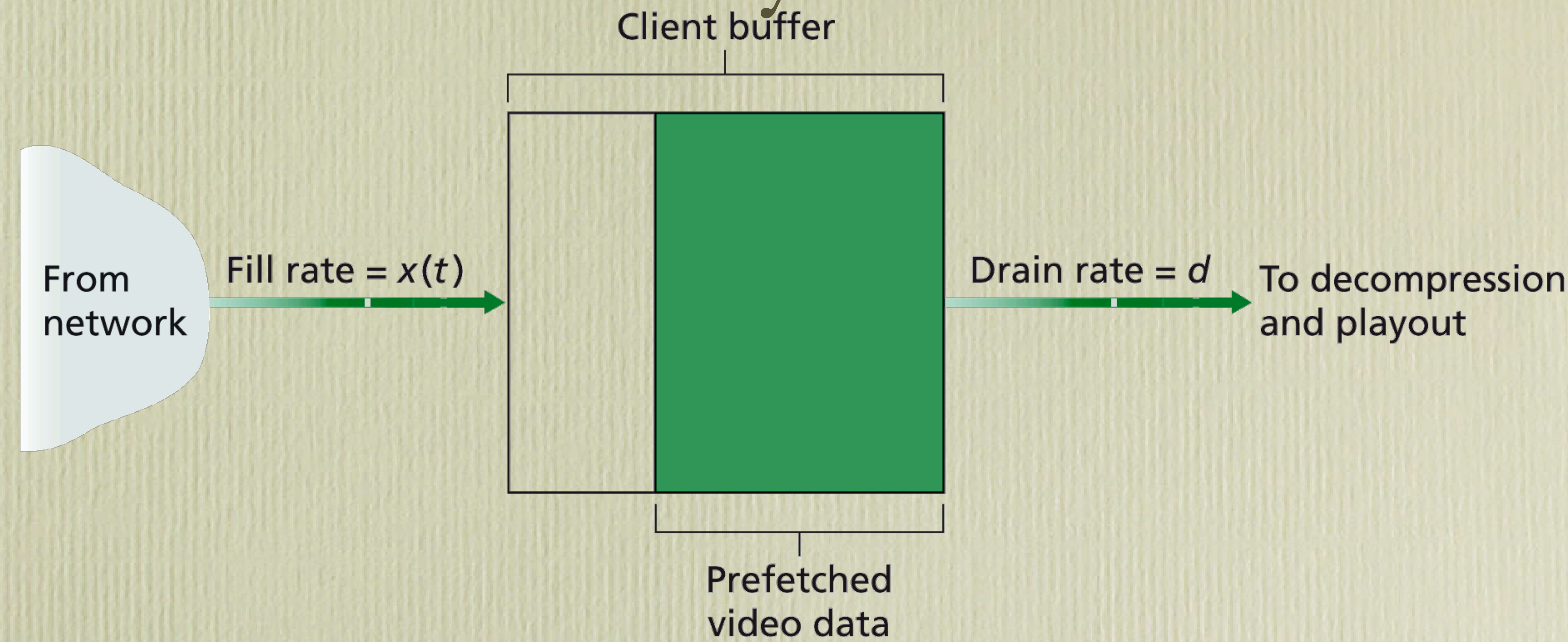
Multimedia networking

- Real-time multimedia is an interesting class of app
  - *delay-sensitive* and *loss-tolerant*
  - crosses layers: issues at application, transport and network layers
- Increasingly important
  - VoIP, streaming audio & video, games
- Bandwidth and delay requirements mean ISPs need to take special care
  - cf. VideoFurnace deployment in Sudikoff
- We will look briefly at audio, video and games

Streaming multimedia

- Media stored at source
  - transmitted to client
  - *streaming* = client playout before all data has arrived
  - still-to-be transmitted data has to arrive in time for playout
    - typical delay requirement for interactivity ~150-250ms
- Can use HTTP server and *meta files*
  - meta file launches media player
  - but is TCP appropriate?
- Use UDP *streaming server*
  - data might arrive out-of-order, stored in *playout buffer*
  - buffer also needed because of *jitter* (variance in network delay)
  - buffer needs to be filled faster than it is drained

Playout



- Drain rate includes *decompression* time
  - most multimedia content is compressed
  - e.g., MPEG-{1,2,4}

RTSP (Real-Time Streaming Protocol)

- How to fast-forward/rewind/pause stream?
  - Rewind is easy, fast-forward not so easy
- RTSP = *out-of-band* protocol (like FTP control channel)
  - doesn't define encoding, transport, buffering
- Retrieve *meta file* via HTTP, browser launches player
  - player sets up data connection and RTSP control connection
- control messages include SETUP, PLAY, PAUSE, TEARDOWN
  - server keeps track of client state using session and sequence numbers

Audio compression

- Analogue audio signal sampled at constant rate
  - e.g., CD @ 44.1KHz = 44,100 samples per second
- Each sample quantised (rounded), e.g. to 2^8 values
  - each quantised value represented by a number of bits
  - e.g., 8KHz, 256 values → 64,000 bps
- Potential redundancy in bits
  - similar patterns, noisy/quiet parts of music
- Receiver converts back to analogue signal
  - e.g., PCM (CD audio): 1.411Mbps
  - MP3: 96, 128, 192, 320 Kbps
  - VoIP: 5.3-13Kbps

Video compression

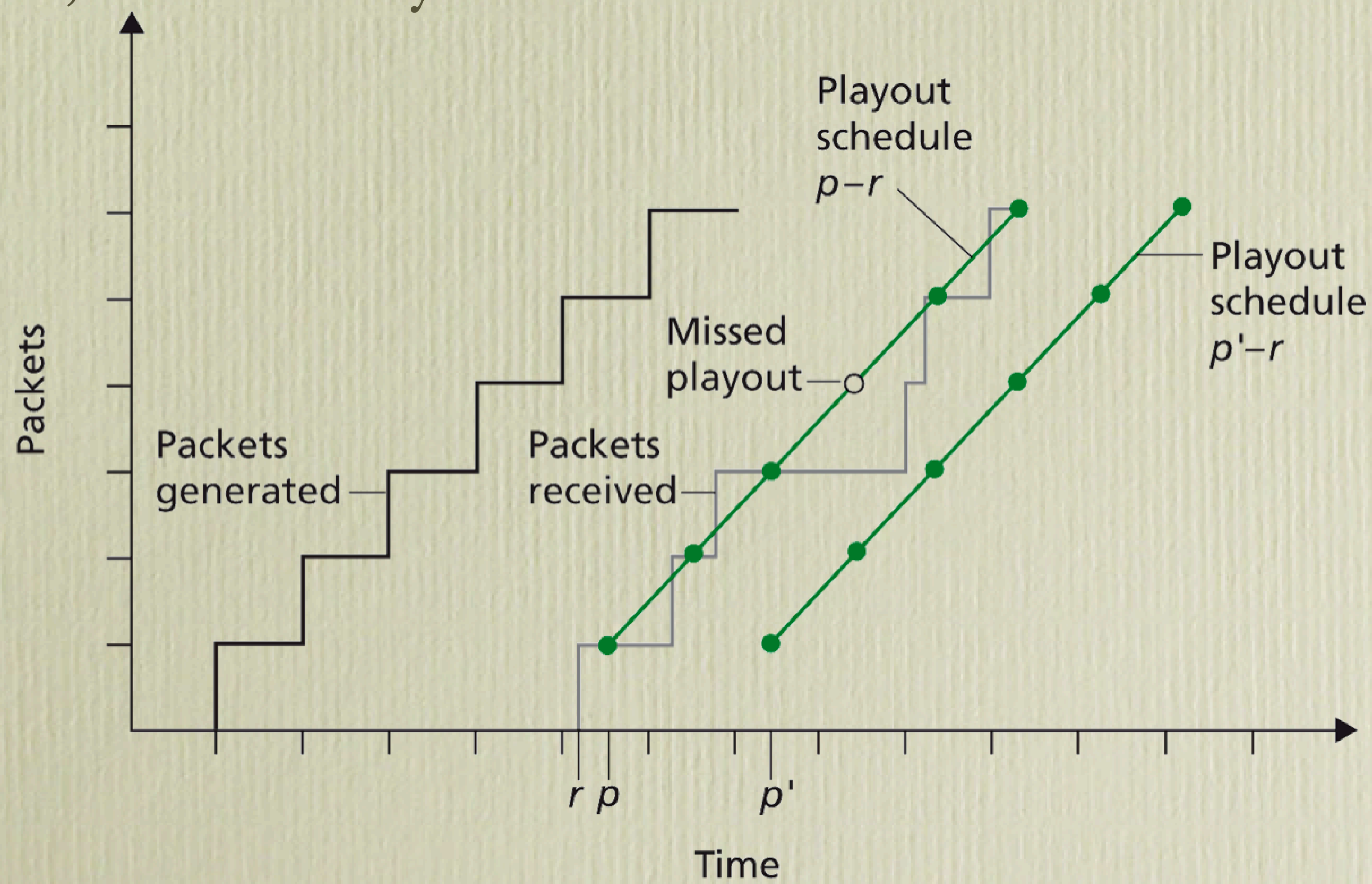
- Video = sequence of images displayed at constant rate
  - e.g., NTSC = 29.97fps, PAL = 25fps
- Digital image = array of pixels
  - each pixel represented by bits
- Bits may be redundant
  - temporal and spatial similarities
- May encode at constant or variable bit rate
  - VBR: use periodic Intra-frames that contain all info, then Predicted and Bidirectional frames that rely on I-Frames
  - e.g., MPEG-1 (VCD) < 1.5 Mbps
  - MPEG-2 (DVD) 3-6 Mbps
  - MPEG-4 (Internet, handheld devices) < 1Mbps

Multimedia over IP

- IP is *best-effort* - may have loss, delay, out-of-order delivery
  - multimedia applications need to cope!
  - use Voice over IP as motivating example
    - voice digitized and sent in chunks (UDP datagrams)
- *Network loss*: IP datagram lost due to congestion
- *Delay loss*: IP datagram arrives too late for playout
- *End-to-end delay*: should be <150ms, must be <400ms
- *Packet jitter*: if receiver plays out chunks as soon as they arrive, resulting audio can be unintelligible

Fixed playout delay

- Receiver tries to playout each chunk *q* msec after chunk generated
  - needs timestamps in the chunk (app-layer header)
  - if chunk with timestamp *t* arrives after *t+q*, chunk discarded
  - if *q* big, cope better with delay loss and jitter
  - if *q* small, interactivity is better



Adaptive playout delay

- Goal: minimise playout delay, keep delay loss rate low
  - Estimate network delay and adjust playout delay at beginning of each talk spurt
  - Silent periods compressed and elongated
  - Estimate network delay in similar fashion to TCP (place more weight on recently-observed chunks)
  - Use sequence numbers to distinguish between loss and gap between talk spurts

Loss recovery

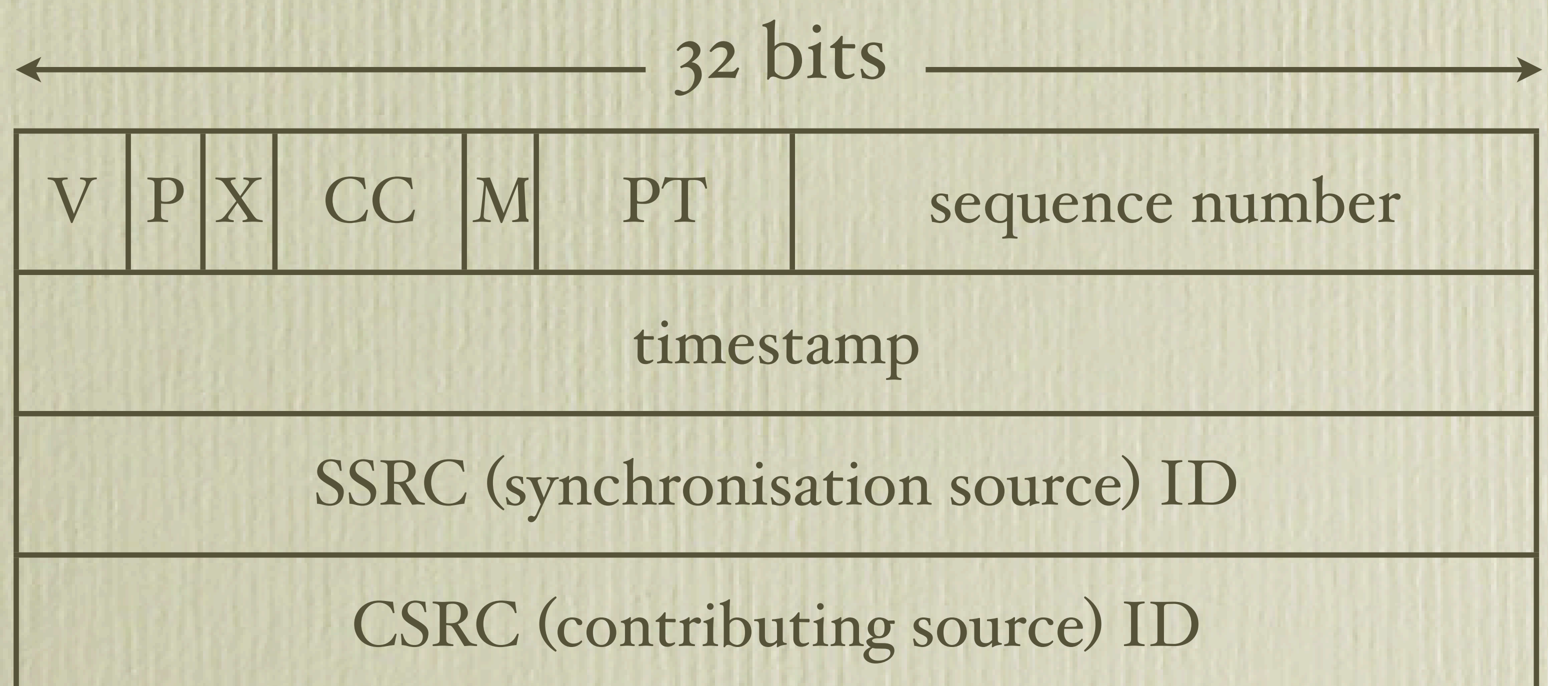
- Forward Error Correction
  - add extra (redundant) info to stream for loss-recovery
  - for every *n* chunks, create and send redundant chunk by XOR-ing
    - can reconstruct *n* chunks if at most one lost chunk out of $n+1$
    - if *n* is small, bandwidth overhead is large
    - if *n* is big, higher probability that 2 or more chunks lost
  - send lower-resolution audio stream
    - e.g., append 2.4kbps LPC audio to 64kbps PCM
    - append lower quality version of n^{th} chunk to $n-1^{th}$ chunk
    - if n^{th} chunk lost, playback lower quality version
- Interleaving
  - resequence data, e.g., chunk 1 contains 1,5,9,13; 2 contains 2,6,10,14 etc.
  - loss leads to multiple small gaps rather than one big gap
- Receiver-based recovery
  - e.g., repeat previous packet in event of loss

Summary: multimedia tricks

- Use UDP to avoid TCP congestion control
- client-side adaptive playout delay
- server-side matches *stream bandwidth* to available bandwidth
  - choose among pre-encoded stream rates
  - dynamic server encoding rate
  - layered multicast
- error recovery (on top of UDP)
  - FEC, interleaving
  - retransmissions (if appropriate and time permitting)
  - error-concealing: repeat recent data

RTP (RFC 3550)

- Real-Time Protocol
  - specifies packet structure for carrying A/V data
  - provides: payload type ID, sequence numbers, timestamp
  - runs on top of UDP (typically multicast)
    - applications, e.g. VoIP, run on top of RTP
  - enables interoperability between A/V applications
- one RTP session for each CM type (audio, video, etc)



RTP header elements

- RTP timestamp = sampling instant of first octet in packet
  - requires high-resolution timers and accurate clocks
  - used to synchronise multiple senders
- RTP payload
  - payload types registered by IANA, e.g., PCM, G.722, G.729
  - encoding type can change during a session
- Marker bit
  - beginning of talk spurt
- SSRC = Synchronisation Source
  - each sender in a conference has its own SSRC
- CSRC = Contributing Source
  - multiple sources can be combined or transcoded by RTP mixer

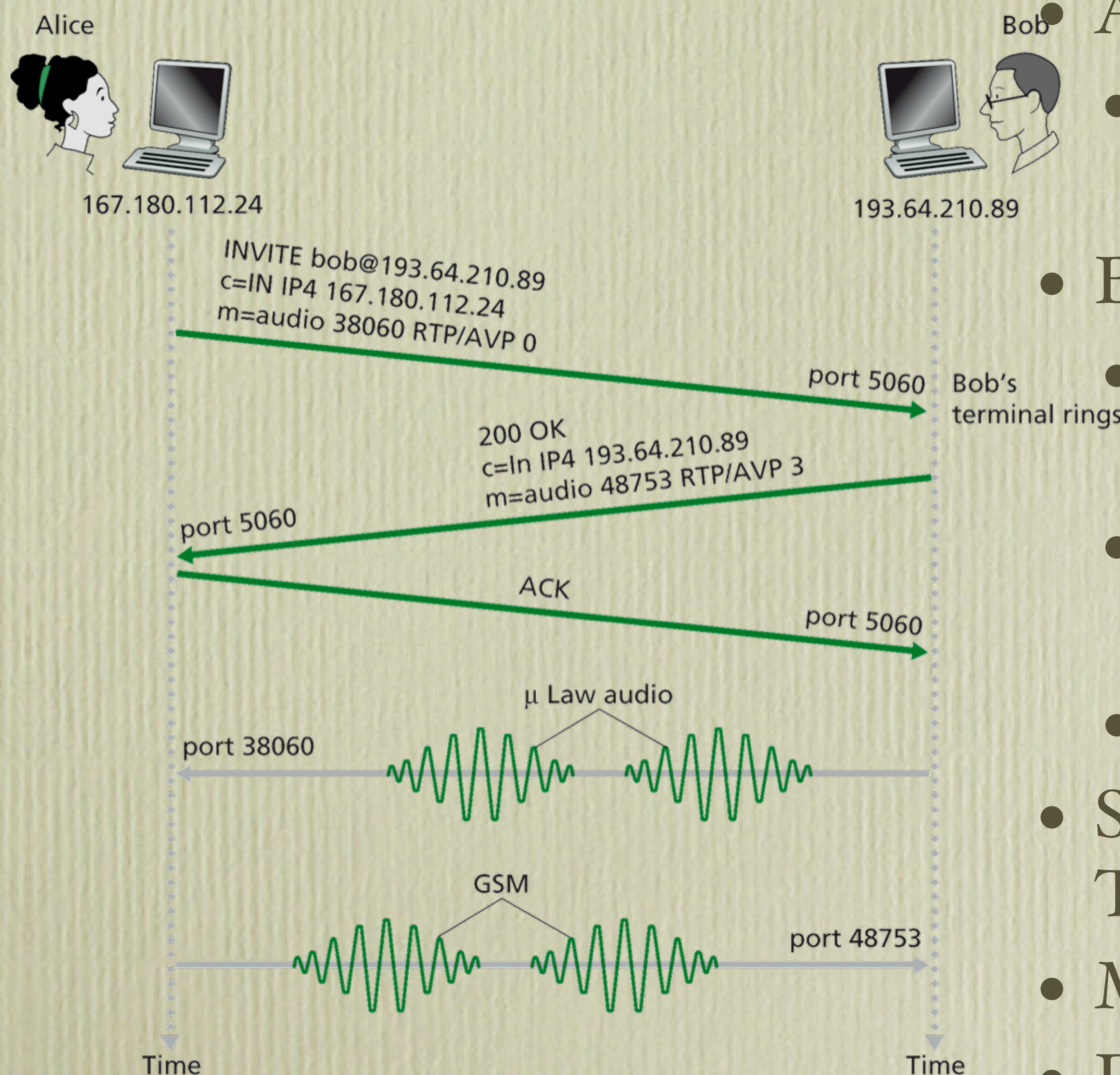
RTCP

- Real-Time Control Protocol
  - RTP on even port, RTCP on odd port
- Includes information about:
  - packet loss, jitter, delay, signal level, call quality, etc...
  - send *receiver reports* for each SSRC
- Feedback from reports can be used to control performance
  - e.g., sender may change encoding to use less bandwidth

SIP (Session Initiation Protocol)

- RFC 3261
- Session = exchange of data between participants
  - VoIP, video, text messaging, etc.
- SIP allows users (SIP endpoints) to discover each other and agree on a session characterisation
  - encoding, etc
  - in other words SIP handles call *signalling*
- Endpoints identified by names or e-mail addresses
  - not phone numbers
  - can reach callee wherever they are, irrespective of IP/device

Call to known IP address



- Alice sends SIP INVITE msg
  - indicates port, IP address and preferred encoding (μ law)
- Bob sends 200 OK msg
  - indicates port, IP address and preferred encoding (GSM)
  - could send 606 (Not Acceptable) if Bob can't do μ law
  - could reject call if Bob busy
- SIP messages can be sent over TCP, UDP or RTP
- Media sent over RTP
- Default SIP port = 5060

SIP messages

```
INVITE sip:bob@domain.com SIP/2.0
Via: SIP/2.0/UDP 167.180.112.24
From: sip:alice@hereway.com
To: sip:bob@domain.com
Call-ID: a2e3a@pigeon.hereway.com
Content-Type: application/sdp
Content-Length: 885

c=IN IP4 167.180.112.24
m=audio 38060 RTP/AVP 0
```

- HTTP syntax
- SDP = Session Description Protocol
- Call-ID: unique for each call

- Don't know Bob's IP address in this call
- Alice specifies in Via: header that she will use UDP

Finding SIP callees

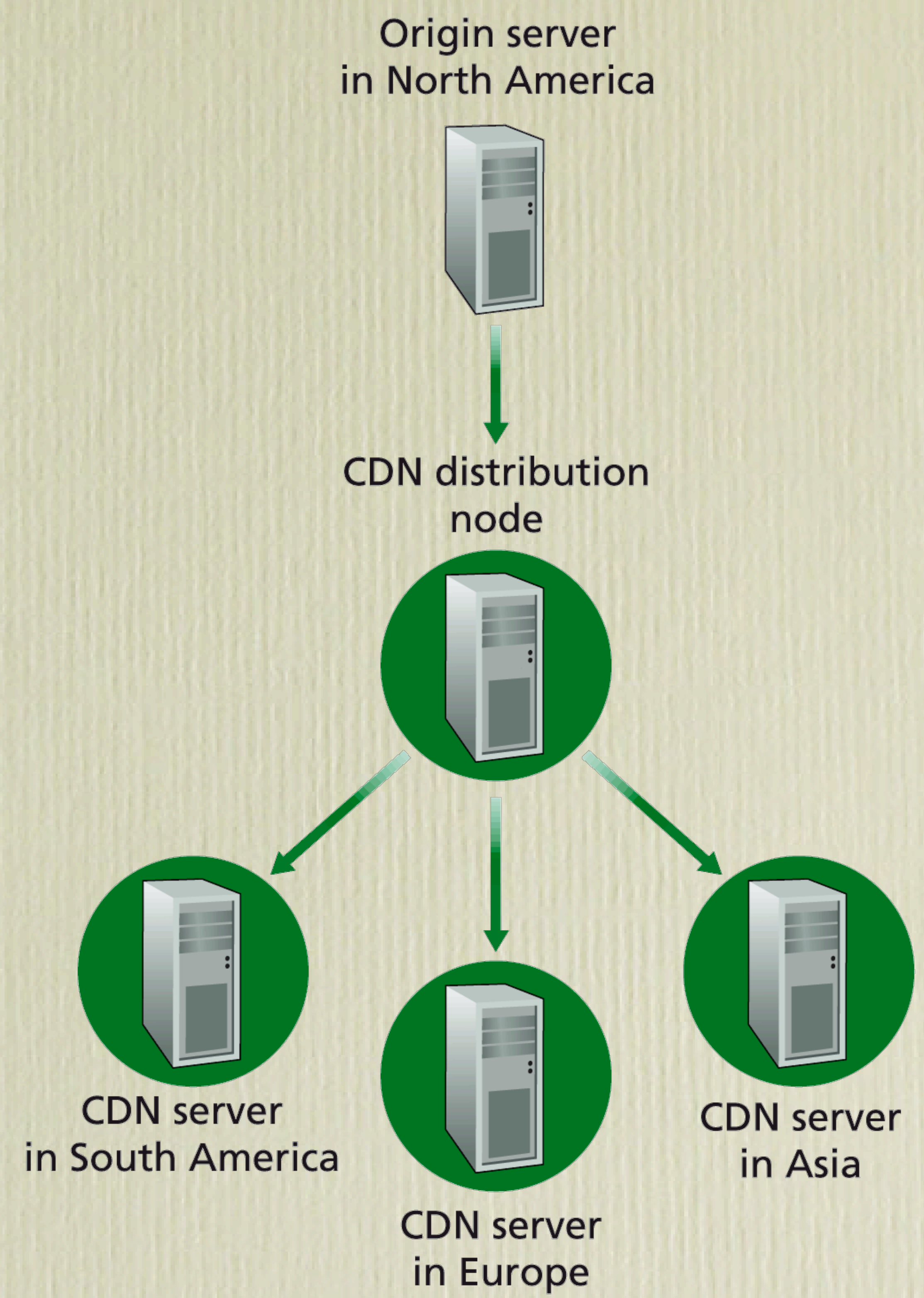
- Caller wants to find callee by name/e-mail
    - callee's device might change
    - on startup, SIP client sends SIP REGISTER msg to SIP *registrar server* (similar to IM)
    - SIP *proxy servers* responsible for routing SIP messages to callee
        - proxy will return callee's current IP address
        - similar to a local DNS server
- SDP can be used to advertise conferences to world
    - use a session directory tool to browse session announcements

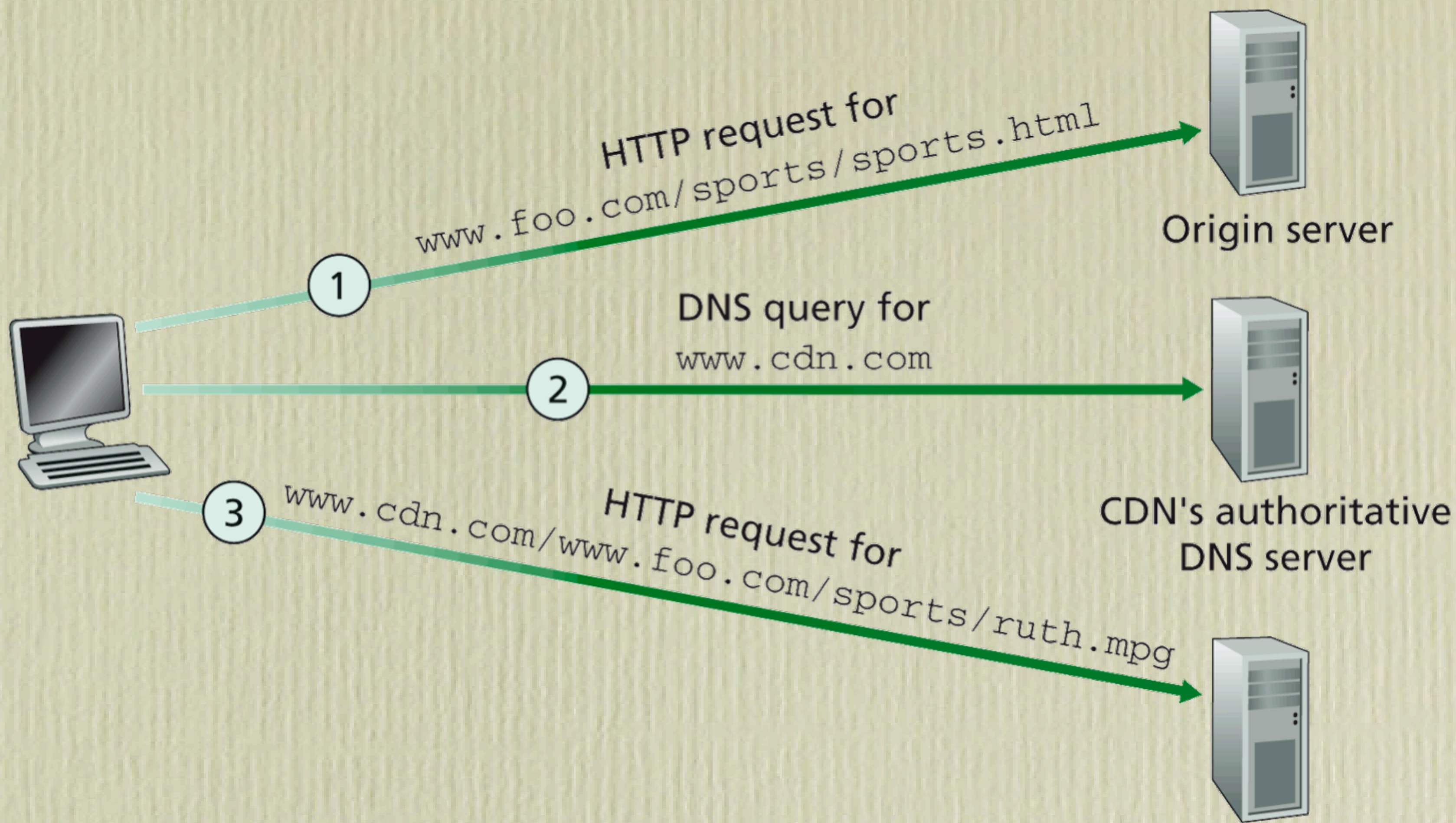
Open vs. closed protocols

- Let's beat this dead horse one more time...
- SCCP ('skinny')
  - used by Cisco, Cisco and Cisco
- Skype
  - used by Skype, Skype and Skype
- SIP
  - used by Vonage, Packet8, AT&T, BroadVoice, MCI, BT, ... (see www.sipforum.org or www.freeworlddialup.com)
  - even Cisco now uses SIP
- Imagine a telephone that only works with one provider ☹

Content Distribution Networks

- How to distribute content?
- Difficult to stream large files from single server in real time
  - Solution: replicate content at lots of servers throughout Internet
  - content downloaded to CDN servers ahead of time
  - place content “close” to user → short path means lower loss, delay
  - CDN servers typically located in edge/ access networks
- CDN (e.g., Akamai) customer is content provider (e.g., CNN)
  - CDN replicates customers’ content on CDN servers. When provider updates content, CDN updates servers.





- Content provider tags objects to be delivered by CDN
  - e.g., `www.foo.com/big.jpg` → `www.cdn.com/www.foo.com/big.jpg`
- CDN uses authoritative DNS server to route requests
  - determines "best" server for client
    - "secret sauce" - RTT, BGP, measurement data
  - returns best server's IP address to DNS queries
  - an application-layer *overlay* network

Networked games

- A very important and interesting type of application
  - popular - millions of players every day
  - heterogeneous devices - PCs, consoles, handhelds, cell phones
  - highly-interactive
  - multiplayer - simultaneous interaction between every client
    - unlike video-conferencing or VoIP
  - strong incentive to cheat
    - unlike video-conferencing or VoIP
- Two main popular types of networked game
  - First-Person Shooter (FPS)
  - Massively Multiplayer Online Role-Playing Game (MMORPG)

- FPS: e.g., Doom, Quake, Halo
- lots of servers set up by individuals
  - small number of players per server (typically 16-32)
- premise: run around fighting and blowing things up

- MMORPG: e.g., Everquest, Age of Empires, Lineage
- Few servers set up by games publishers
  - hundreds of thousands of players per server
- premise: run around, collecting items/money/experience, and occasionally fighting and blowing things up

Client-server architecture

- Clients (players) connect to a central server
  - UDP, small fixed-size packets at constant rate
    - e.g., Half-Life: 60-300 bytes every 60ms
  - packets contain state updates
  - bandwidth not an issue
    - delay and jitter are most important
- Server is *authoritative*
  - manages *inconsistent* state
  - prevents cheating

Distributed Interactive Simulation

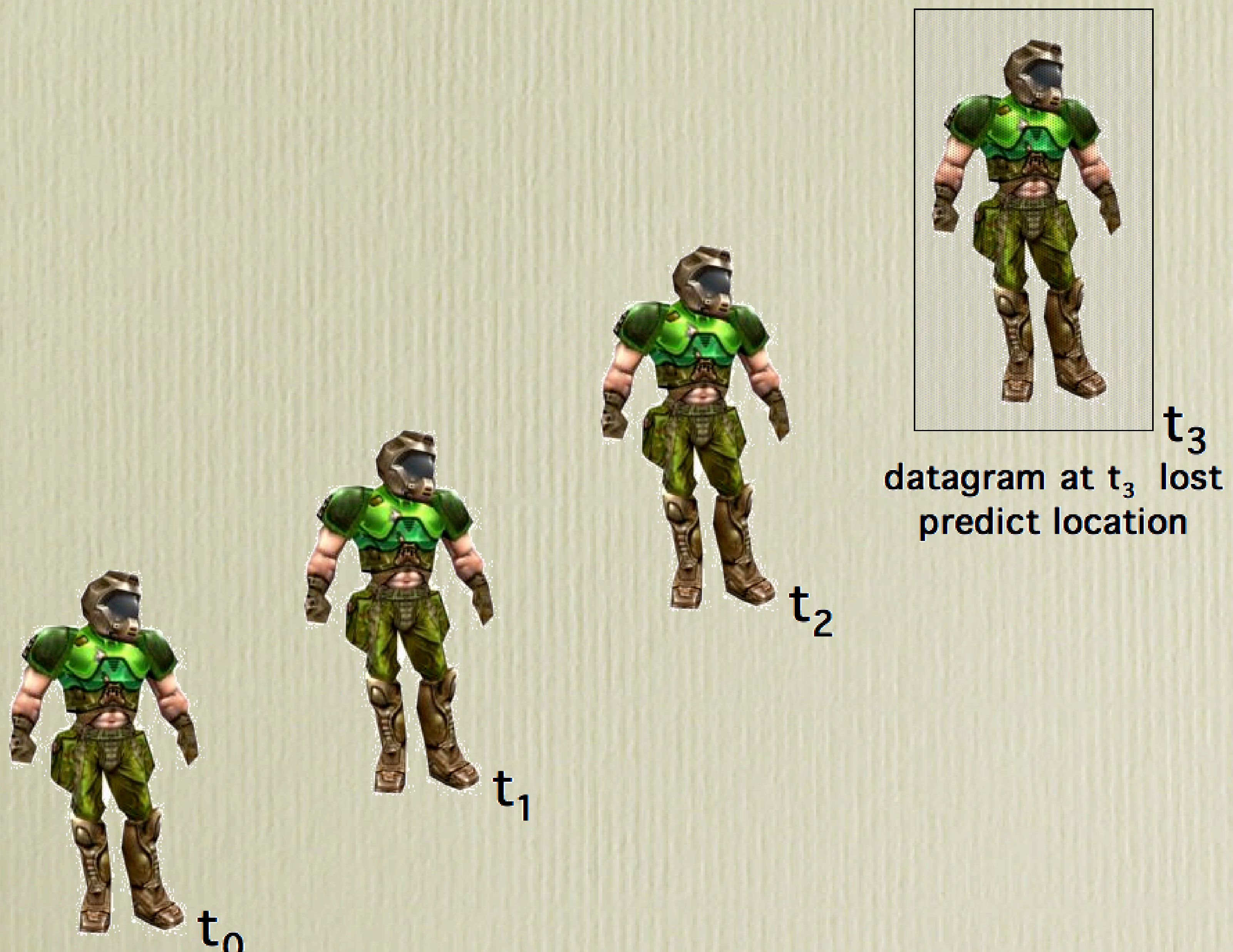
- IEEE standard for simulations
  - Derived from military SIMNET tank-training program
- Designed to simulate many tasks: training drivers of tanks/aircrafts/ships, military offensives
- Distributed peer-to-peer architecture
  - all clients broadcast state to all other clients
- Entity-event model
  - Entities: vehicles, missiles; Events: firing, moving
- Introduced lots of gaming concepts
  - dead reckoning, consistency
- But not everything relevant to gaming
  - clients are trusted in DIS
  - reliability paramount (98% of packets **must** be delivered)

Dealing with network loss/delay

- Client-server increases delay
  - clients send state back to server, wait for response before verifying their own state
  - e.g., A shoots B
    - B has to wait for A's bullet datagram to leave A, reach server, be sent from server to A
    - A might shoot someone else in the meantime
    - leads to inconsistency
  - server has to process state updates from all clients before responding
  - clients typically render local state immediately, then *warp* if state is later found to be inconsistent

Dead reckoning

- Use previous state information to *predict* existing state
  - error-concealing method (like repeating audio chunks)
- Use velocity, knowledge about game
- Can be used client-side or server-side



Timewarp

- Server keeps track of global state
  - take periodic snapshots of state
- If inconsistency is detected, *rollback* the game clock
  - revert to last known consistent snapshot
- Used in some games, e.g., *Half-Life*
- Computationally-expensive
  - could use multiple game servers instead
  - each server keeps different snapshot
- What do players think?
  - player can "come back to life" if clock rolled back

Area of interest management

- Player is not interested in every other player in game
  - only care about the area in which they are playing
- Divide game world into areas
- Each area becomes a separate multicast group
- Clients subscribe only to those relevant groups
- Saves bandwidth, improves scalability
- Requires ISP multicast support
- More appropriate to MMORPG than FPS
  - in FPS, few players per server, and players may quickly traverse the entire game world - requires lots of multicast group joins

Jitter and relative delay

- Jitter - variance in network latency
  - difficult to eliminate - queues and bursty packet arrivals
  - large effect on game
  - bullets can take variable amounts of time to reach targets
- Relative delay - difference between players' RTT
  - if one player is further away from server than everyone else, that player's experience is impacted
  - players prefer uniform high delay to different relative delay

Peer-to-peer gaming

- Fully-distributed - no central server
- Not very popular (yet)
  - difficult to control cheating
  - difficult to make money
- Wireless games may be peer-to-peer
  - e.g., ad hoc Sony PSP and Nintendo DS
- Some games claim to be "peer-to-peer"
  - no preconfigured central server
  - one of the peers acts as server
    - other peers connect to that peer

Cheating in games

- Lookahead cheat
  - peer-to-peer game with timestamped packets
  - cheater Alice lies about delay (fake timestamp)
  - may get to see other player's (Bob's) packets before them
- Suppress-correct cheat
  - exploit dead reckoning
  - Alice deliberately drops packets (don't send)
  - Bob will dead reckon Alice's position
  - Alice sends a packet just before Bob is about to disconnect Alice
    - this packet may give Alice an advantage
- Display-driver cheats
  - "seeing through" walls
  - seeing parts of world that Alice isn't supposed to see

Cheat-proofing games

- Client-server
  - Central server controls overall state
  - doesn't prevent display-driver cheats
- Area of interest management
  - prevents display-driver cheat, but need to verify subscriptions
- Signed updates
  - use keys so that updates can only be decrypted at appropriate time
- Lockstep
  - game cannot progress until all players have sent updates
  - can slow things down (but can pipeline updates)
- Game registration (CD key)
  - cheaters can be banned, which might deter them

Money

- Games are increasingly big-business
  - larger revenues than Hollywood (according to some spurious stats)
- Increasing number of tournaments, sponsorship, etc.
  - Global Gaming League, World Cyber Games
  - Starcraft games are often broadcast on Korean TV
- Charging models
  - FPS: pay for program, free to run servers
  - MMORPG: pay monthly subscription to access server
- What do economics mean for cheat-proofing?
  - Rollback in the final game of the world championships?