Chapter 10 – Traffic Measurements, Flows and Parameters

Magda El Zarki
Dept. of CS
Univ. of CA, Irvine
Email: elzarki@uci.edu
http://www.ics.uci.edu/~magda

Outline

- Measurements – How, types and errors
- Flows – Types and origins
- Patterns – understanding and analyzing

Traffic Patterns

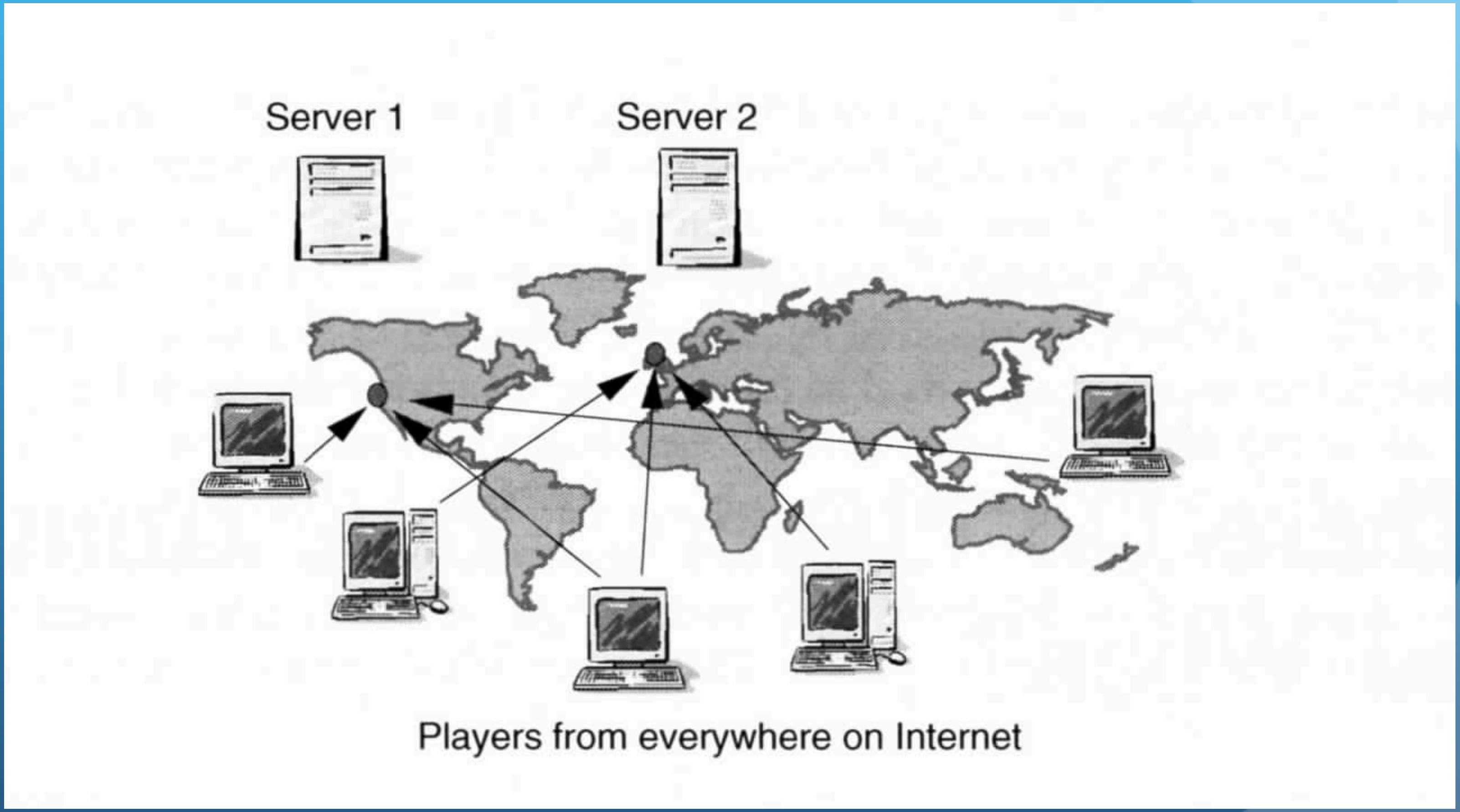
- Measuring Game Traffic
- Understanding your audience
- How accurate is your data

Measuring Game Traffic – Why?

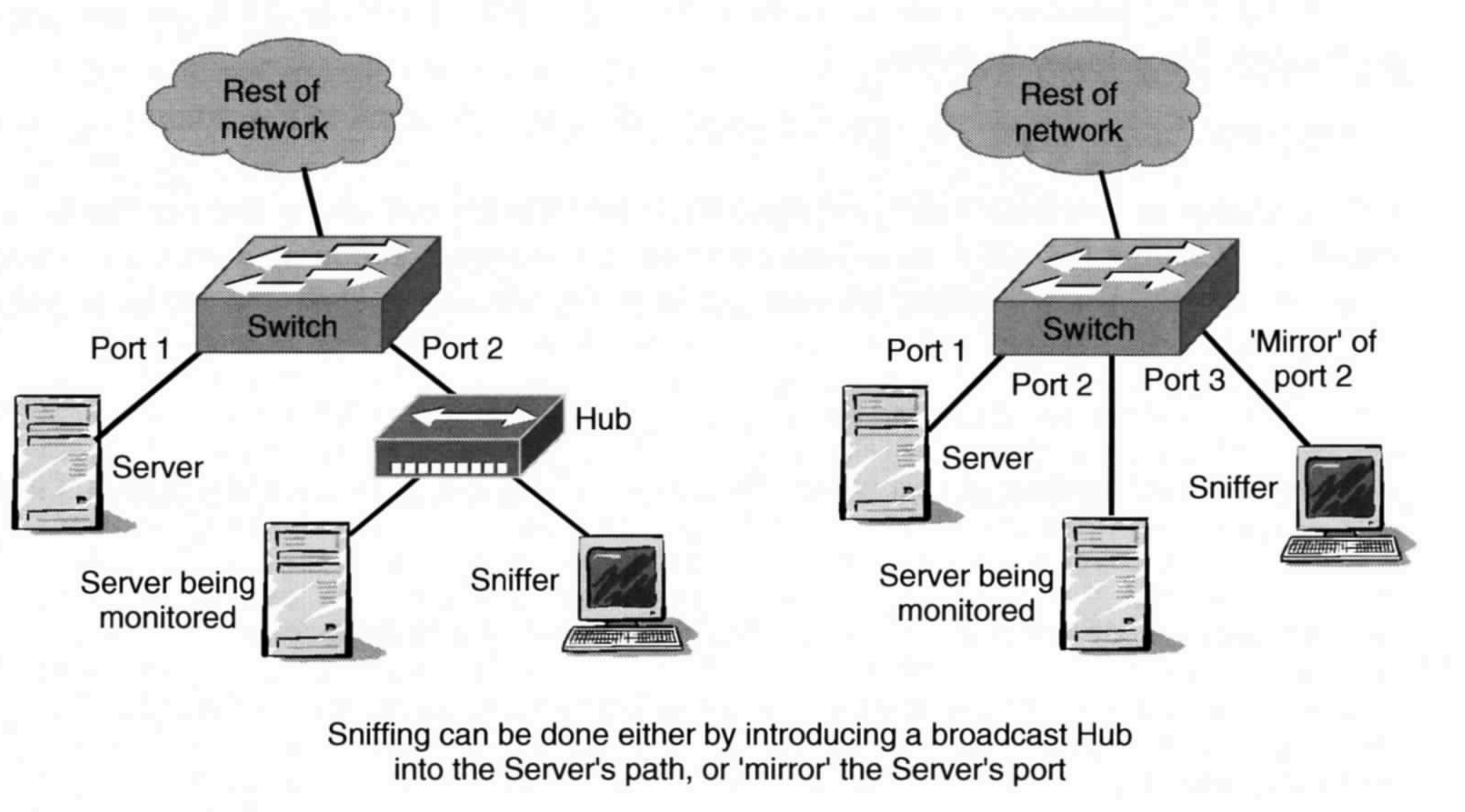
- It is critical for any game provider to understand
  - who the audience is
  - where they are coming from – radius around servers
  - what are the popular playing hours

- This helps for capacity planning
  - on the server side for computing loads
  - on the ISP side for network bandwidth demand

Knowing your audience



How to measure traffic?



Sniffing Traffic

- How to get access to the client server ports to sniff traffic flows?
  - Inserting an Ethernet hub (promiscuous device) between client and server
    - easy to do, can be done on either side
    - hubs are usually low speed devices and thus will bring down the link bandwidth -> lowest common denominator
  - Replicating the client - server router port traffic to an administratively monitored – port mirroring
    - have to get access to router port and have administrative control

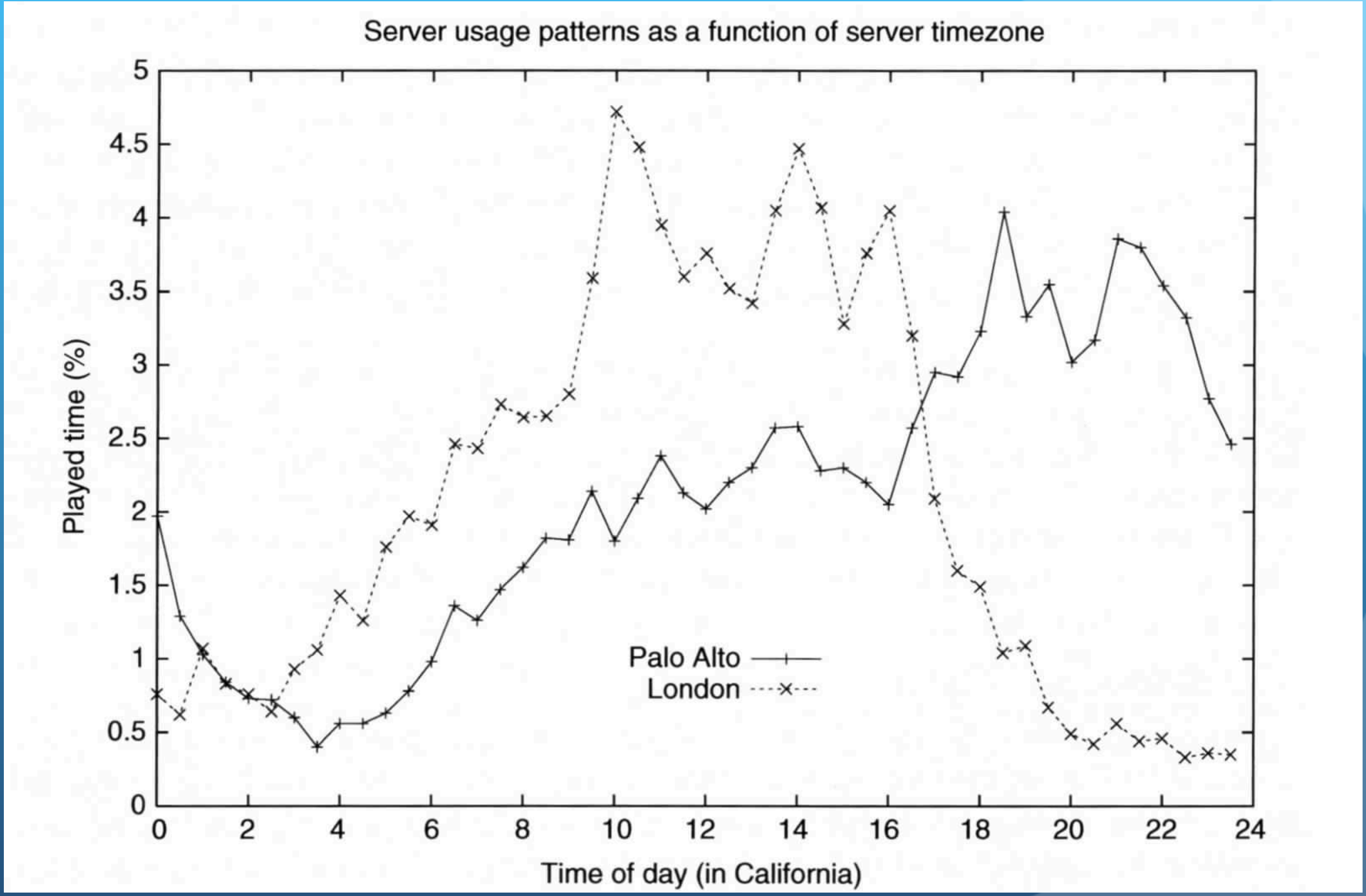
Sniffing Tools

- Real time popular public domain tools:
  - tcpdump – command line tool
  - Ethereal – comprehensive GUI interface
- Both are supported on most OS platforms

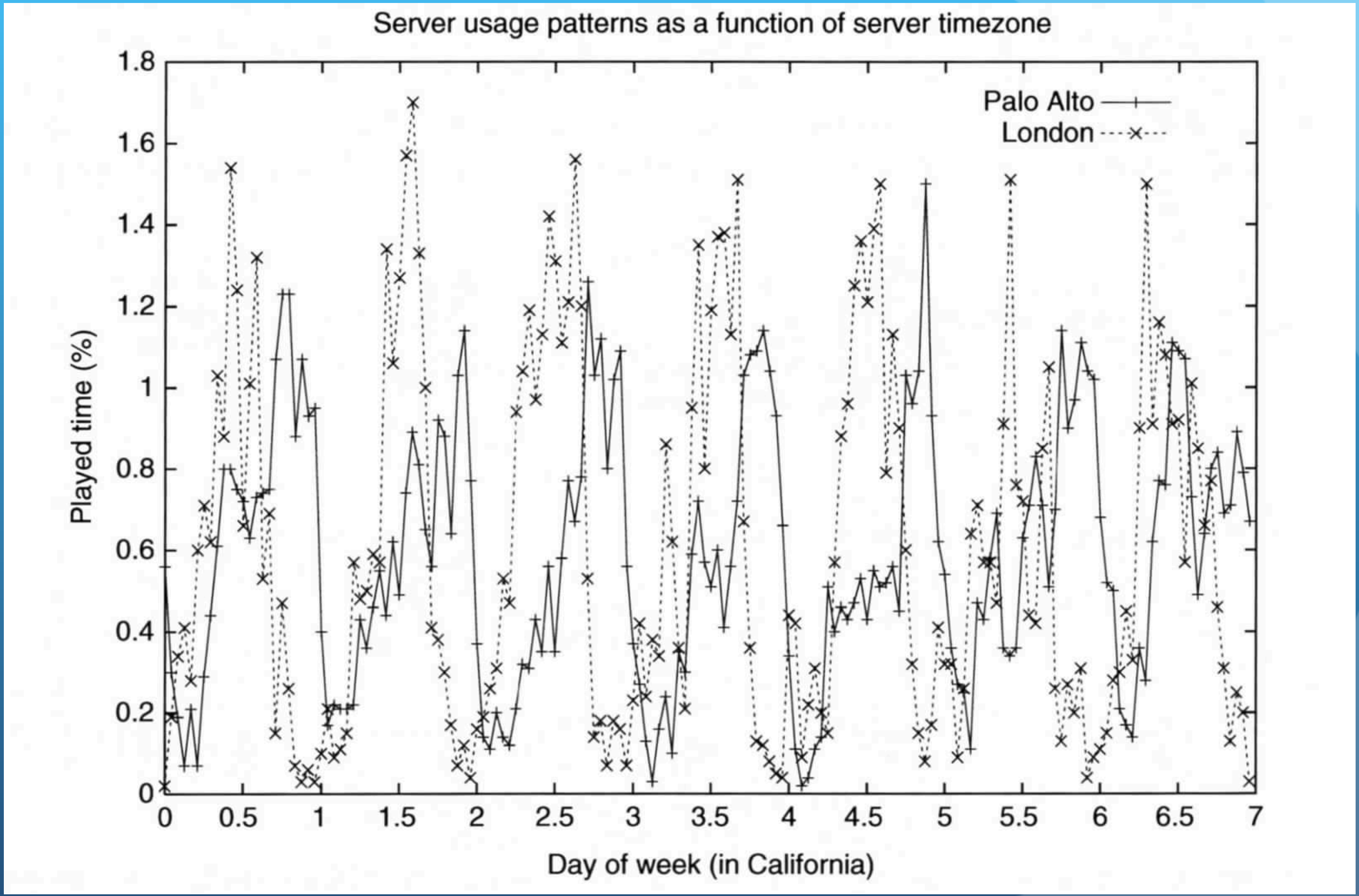
Game Play Trends

- Good to understand when players come on line to play
  - Days of week
  - Time of day – may vary with day of week
  - May vary with continent or even country – different work hours and off days
- Helps to forecast server capacity
- ISPs can plan and accommodate user demands and create SLAs that adapt to player needs

Example of Hourly Server Cycle Load – Quake III Arena



Day by Day Server Cycle Load – Quake III Arena



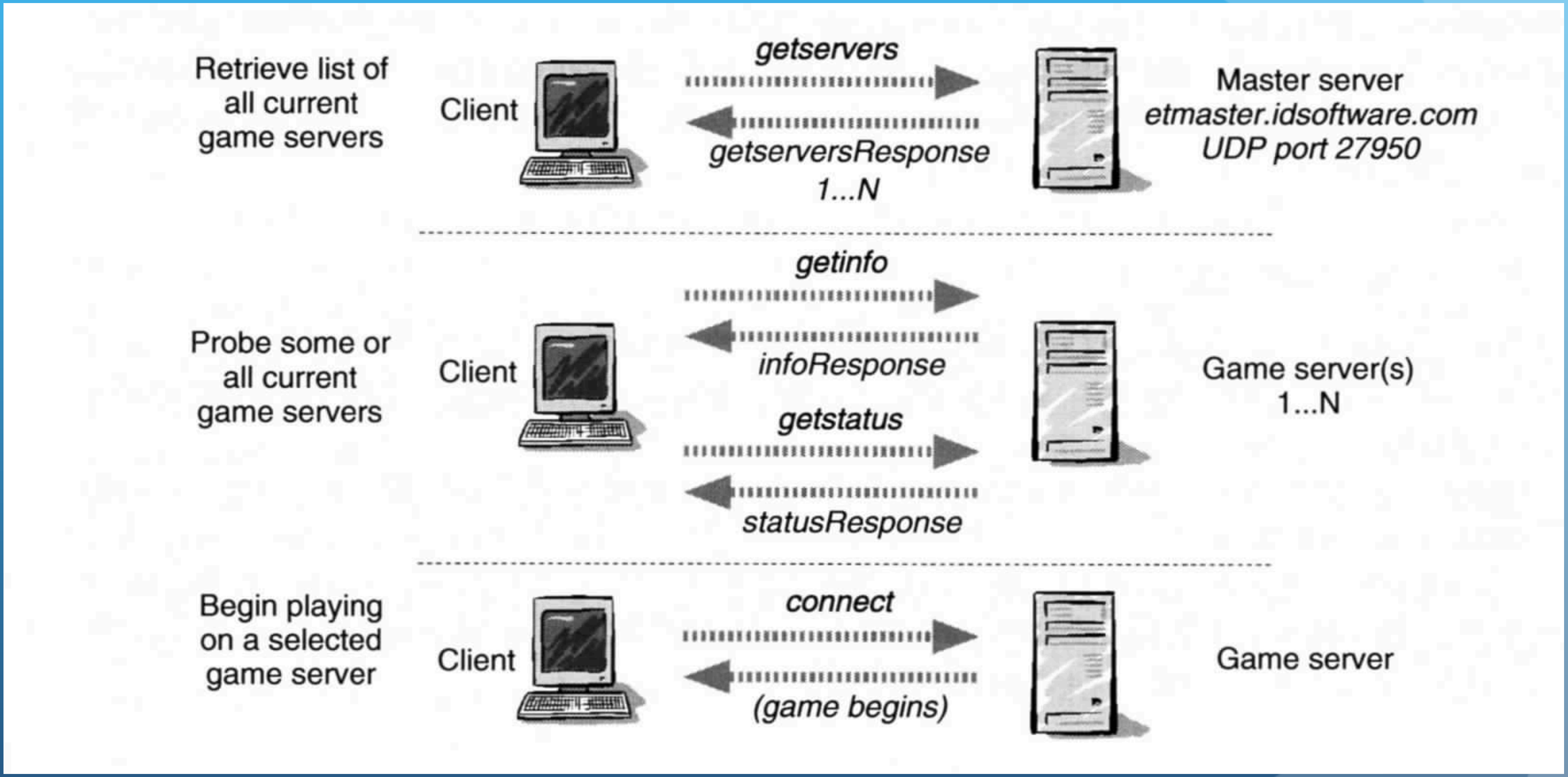
Server Discovery

- Players choosing servers
  - Even though most clients will chose a server within their region, server load may drive some players to servers further away even if it may mean an increase in transmission latency
- Probe traffic
  - Easy to ignore but has some impact – used by clients and server discovery tools to find game servers
  - Indicates where potential players reside (usually only probe neighboring servers)

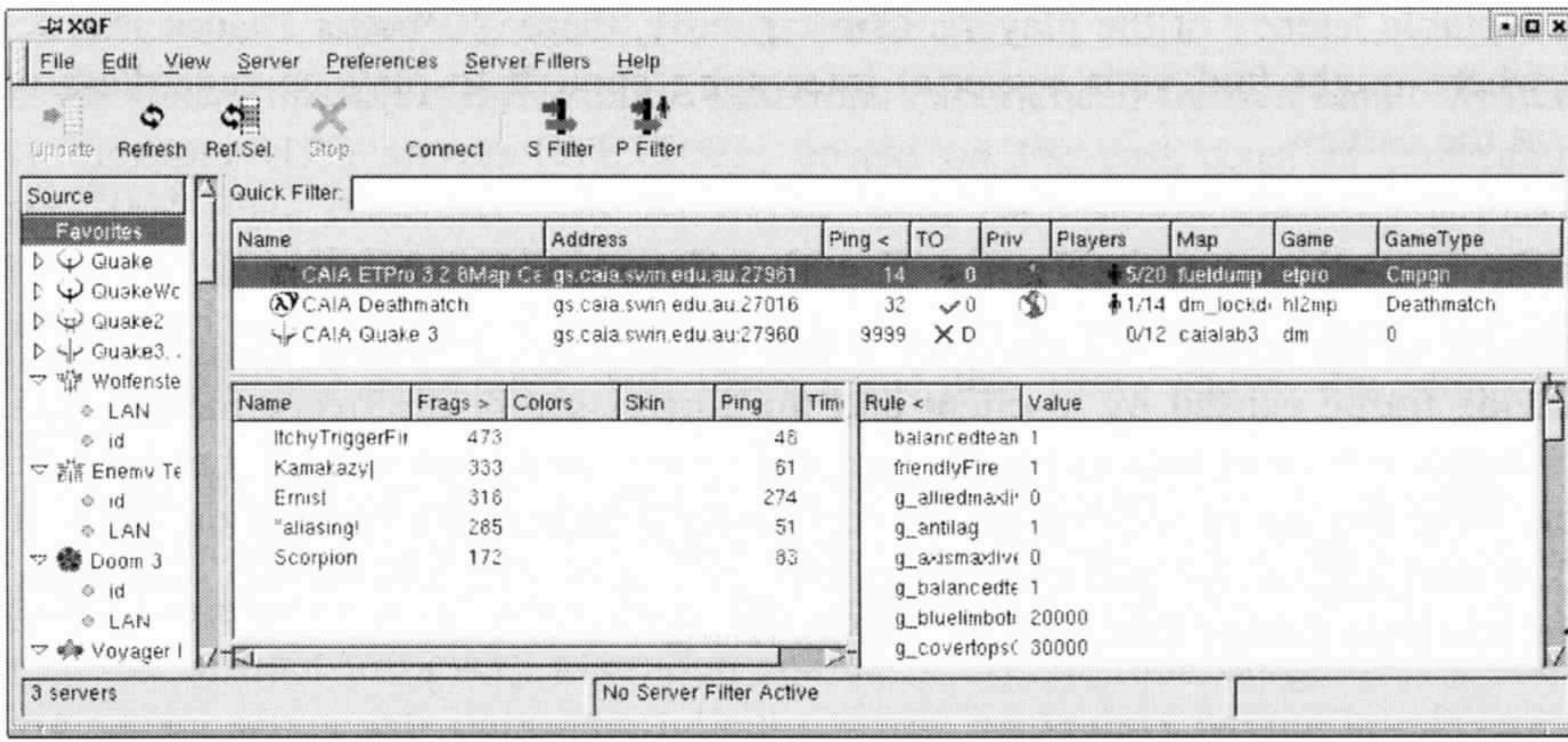
Probe Traffic

- Players trigger an automated search process when they get online and want to initiate a gaming session
- Clients probe a master server that sends a list of IP addresses of current game servers
- Clients then probe each server for information
  - Server type
  - Current map
  - Number of players/teams
  - Number of available slots
  - Etc.

Probe Traffic Interaction



Example of Client Server Discovery Tool – xqf using QStat



Probe Traffic Analysis

- Although probes are not large in size – small IP packet sizes
- The sheer volume creates a large background traffic for game systems – players constantly querying servers for information
- Traffic load unpredictable as probe traffic not limited to players from region. Players will query most if not all servers on the master list and pick a server only after finding one that can serve their needs which beside latency, includes load, available player spots, etc.

Mapping Traffic to Player Locations

- IP addresses and Geographic location – not easy to do
  - Databases exist that provide some mapping between IP addresses and location – GeoLite, Geobytes
  - Reverse-lookup to get their domain name – some ISPs will embed region specific codes and names into the domain names
  - Not very accurate as there is little incentive for ISPs to provide detailed topological information on their clients.
- Latency Tolerance – trying to understand player choices of servers and if latency played a role in the choice. Traceroute and TTL can be used with client IP addresses to collect data during game play and observe client server patterns.

Traffic Measurements

- Accuracy
- Frequency
- Quantity vs Storage and Analysis

Timestamping errors

- The easiest way to collect data is to timestamp it.
- Traffic analysis requires sub millisecond timestamping accuracy – higher resolution then that of game play
- Capture software on devices will claim micro second resolutions BUT
  - The clocks on many devices are not very accurate at that level
  - May not be running real time software that will process the arrival of packets instantaneously.

Hardware Clocks

- Clock - Counter that increments at a fixed known rate (note drifts over time)
- Measured in ticks – X ticks per second
- Operating systems measure time intervals in no. of ticks and use that to estimate an interval of time. The closer the ticks the less error in the estimate - no sub tick estimates measurements.

Improving Data Collection

- Calibrate the hardware and software using a calibration device (data generators that provide accurate timing information). Adjust measurements based on calibration results (i.e., add x msec to average readings)
- Minimize processor load on sniffing device and use a reliable device that doesn't introduce random errors due to unrelated system requirements (ie only polls ports at fixed intervals, or skips a poll when a new service is started, etc.)
- Re-synchronize periodically to counter clock drift.

Ticks, Snapshots and Updates

- Ticks – the smallest unit used by the OS to calculate the length of a time unit (usually a second). E.g., 20 ticks per sec -> 1000/20 = 50msec resolution
- Snapshots – the rate at which the server can send updates to clients. Multiple of ticks
- Clients cannot request a snapshot rate that is higher than the server update rate.
- Can request a slower rate – is always a multiple of the server's tick rate – usually a multiple of the server's update rate. Some games will allow something in between – i.e., custom tick rate.

Server and Client Exchanges

Trade offs – Accuracy and Load

- The rate of updating and the size of the update packets will determine the necessary link bandwidth between the server and the client.
- Both upload and download can be an issue for a client.
- The client may request a slower snapshot rate due to its access link bandwidth to the server.
- Client updates can also be limited in rate and packet size as the upload bandwidth is usually more constrained.
- Caps can be set on both sides to reduce the traffic load

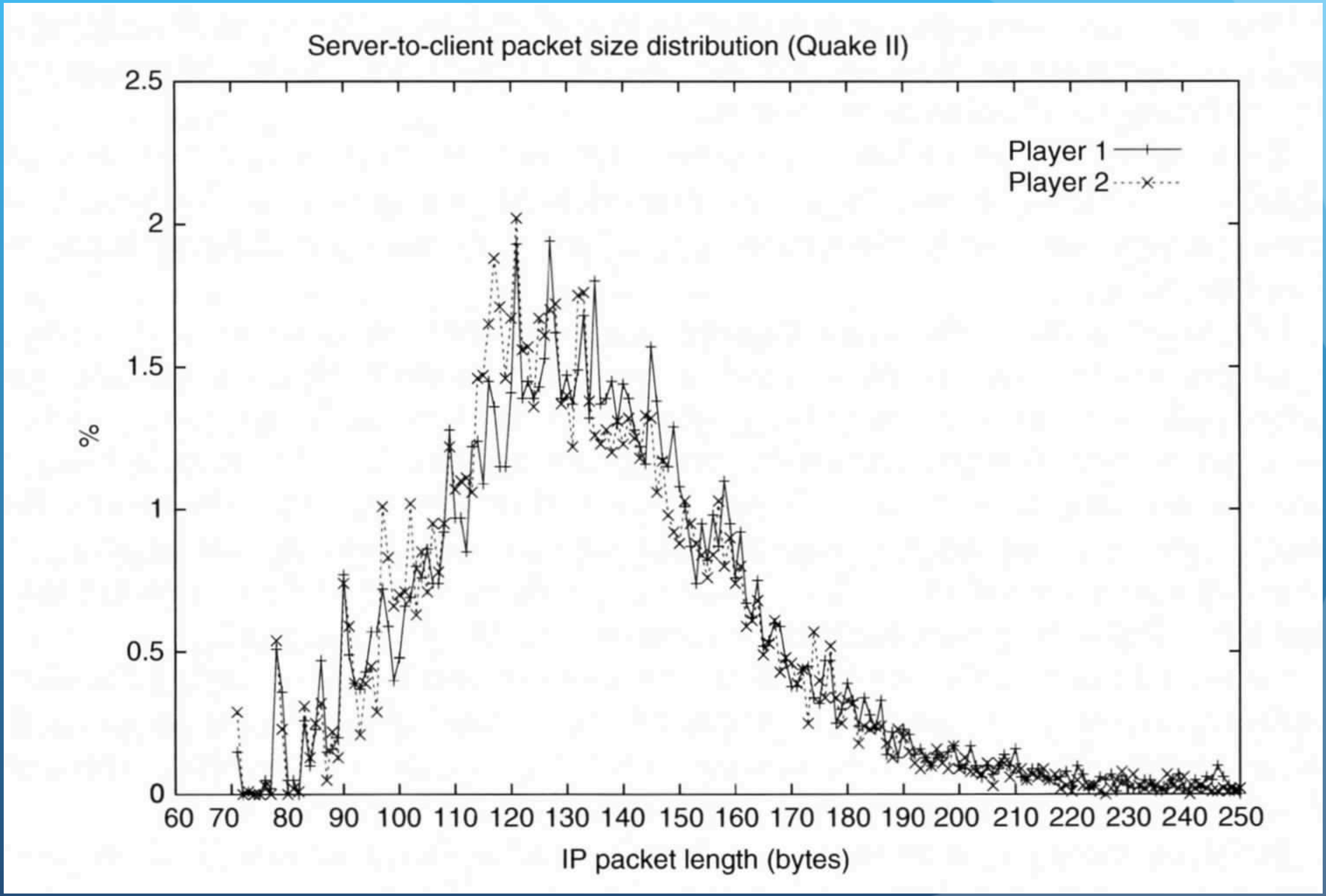
Design Choices

- To reduce the amount of traffic from server to client:
  - Eliminate precise details in the updates
  - Send only data that is of importance to that particular client – i.e., in view (nimbus) information only
  - Send incremental changes between snapshots whenever possible – delta compression
- To reduce the amount of traffic from the client to the server:
  - Only send important changes that affect game play
  - Send incremental changes

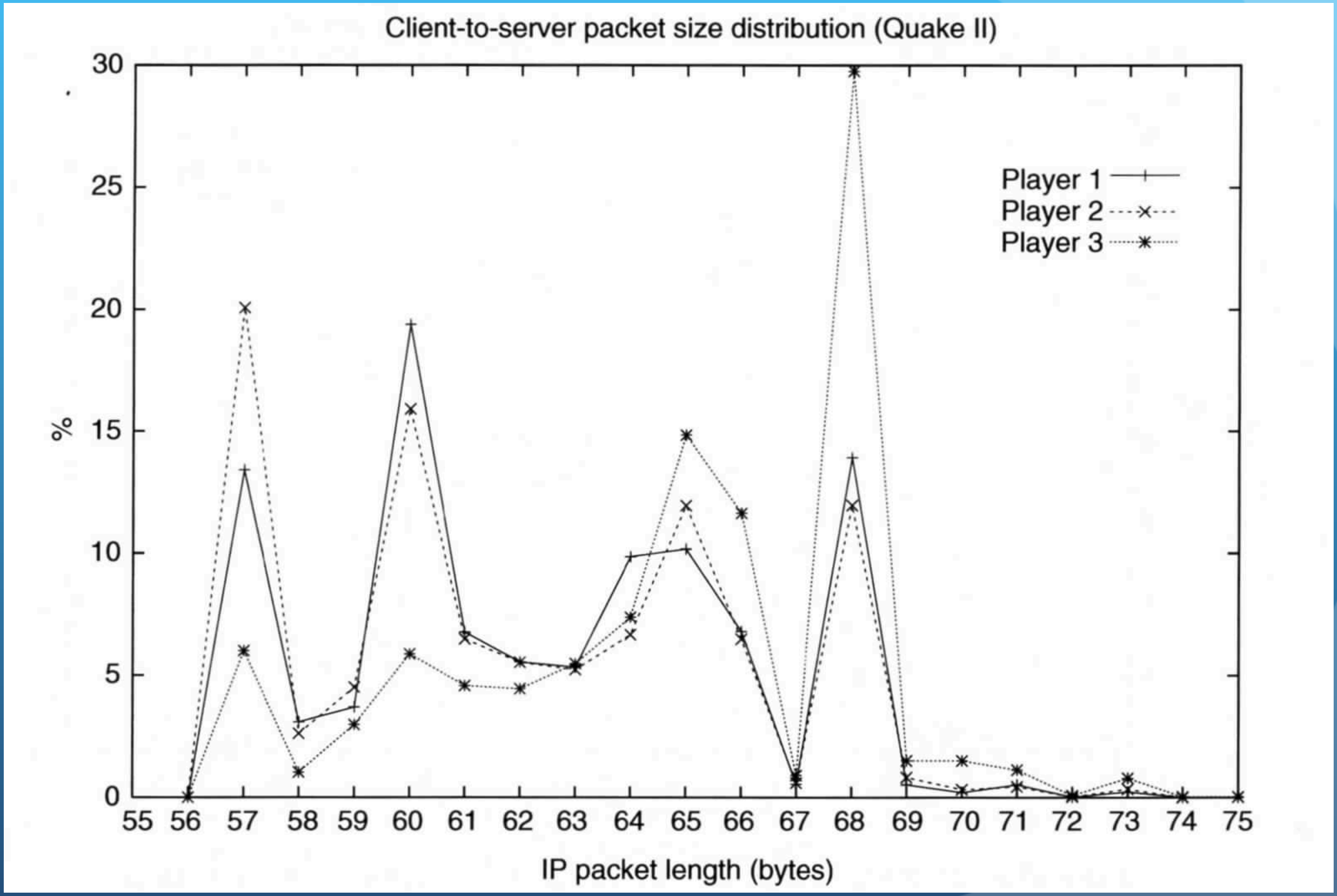
Sub-second Packet Size Distributions

- Server to client much larger packets
- Only interested in “in game” distributions – not pre, post or inter game traffic
- Influence of game map on packet size

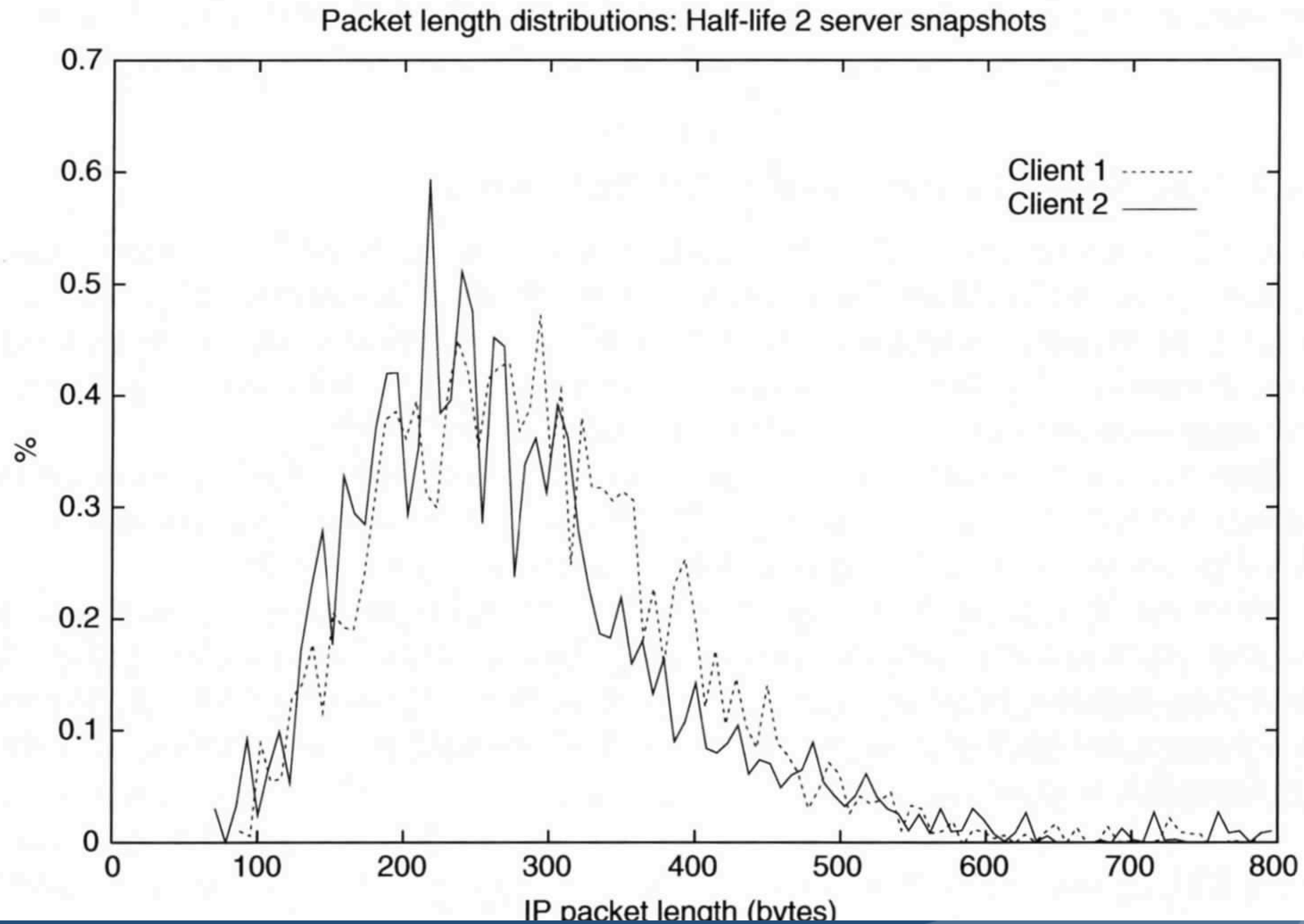
Server to Client for Quake II



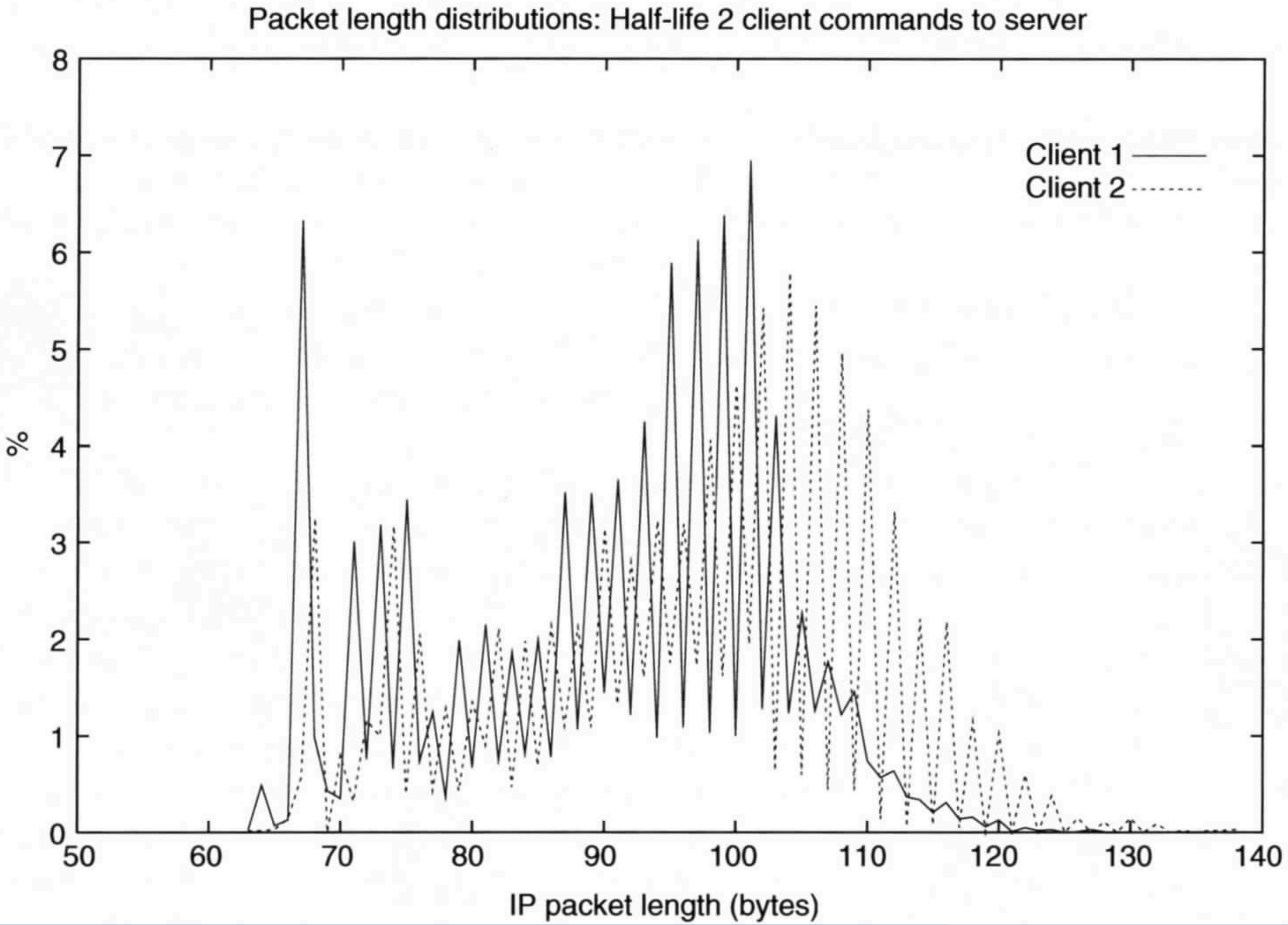
Client to Server – Quake II



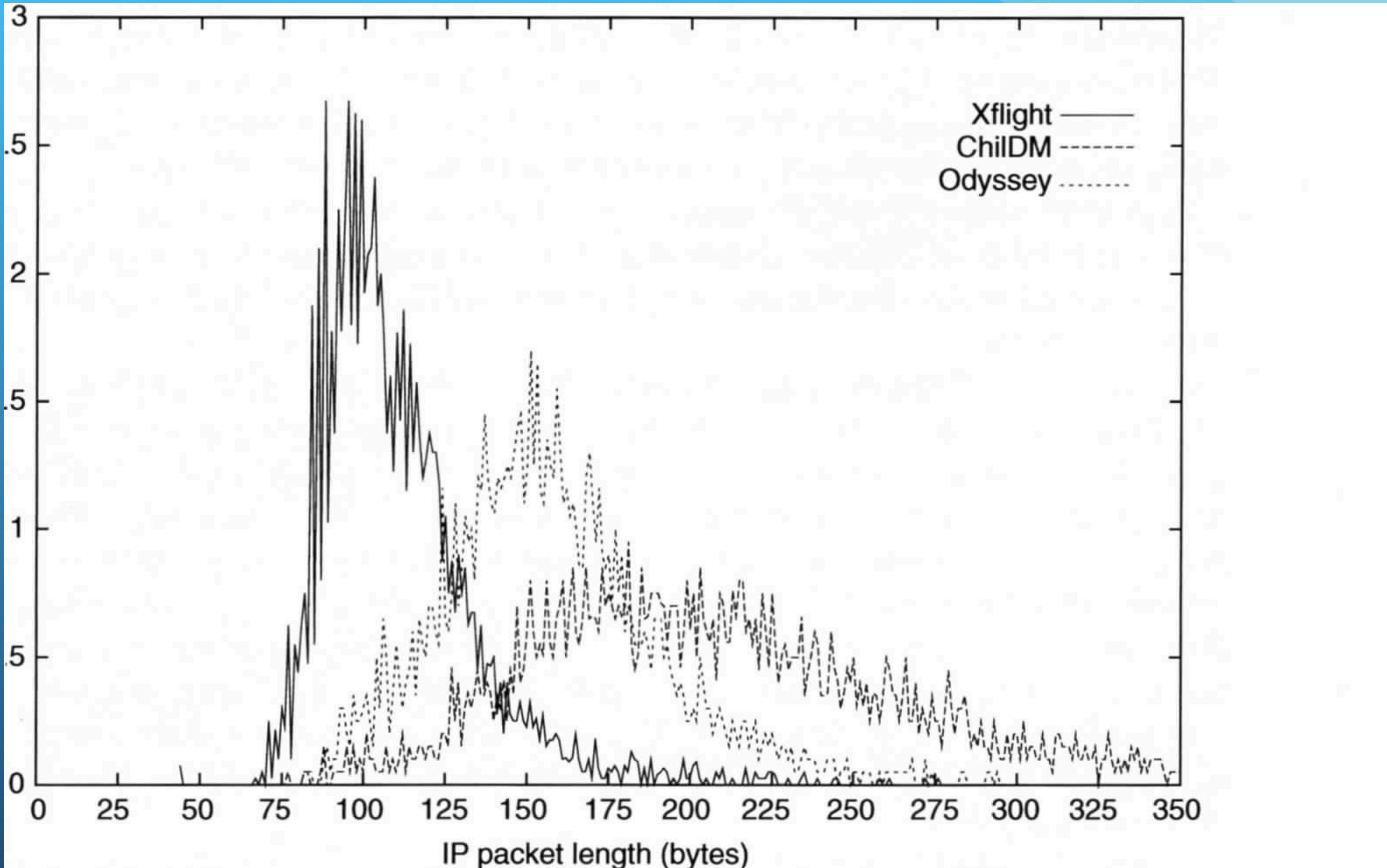
Distributions for Half Life S-C



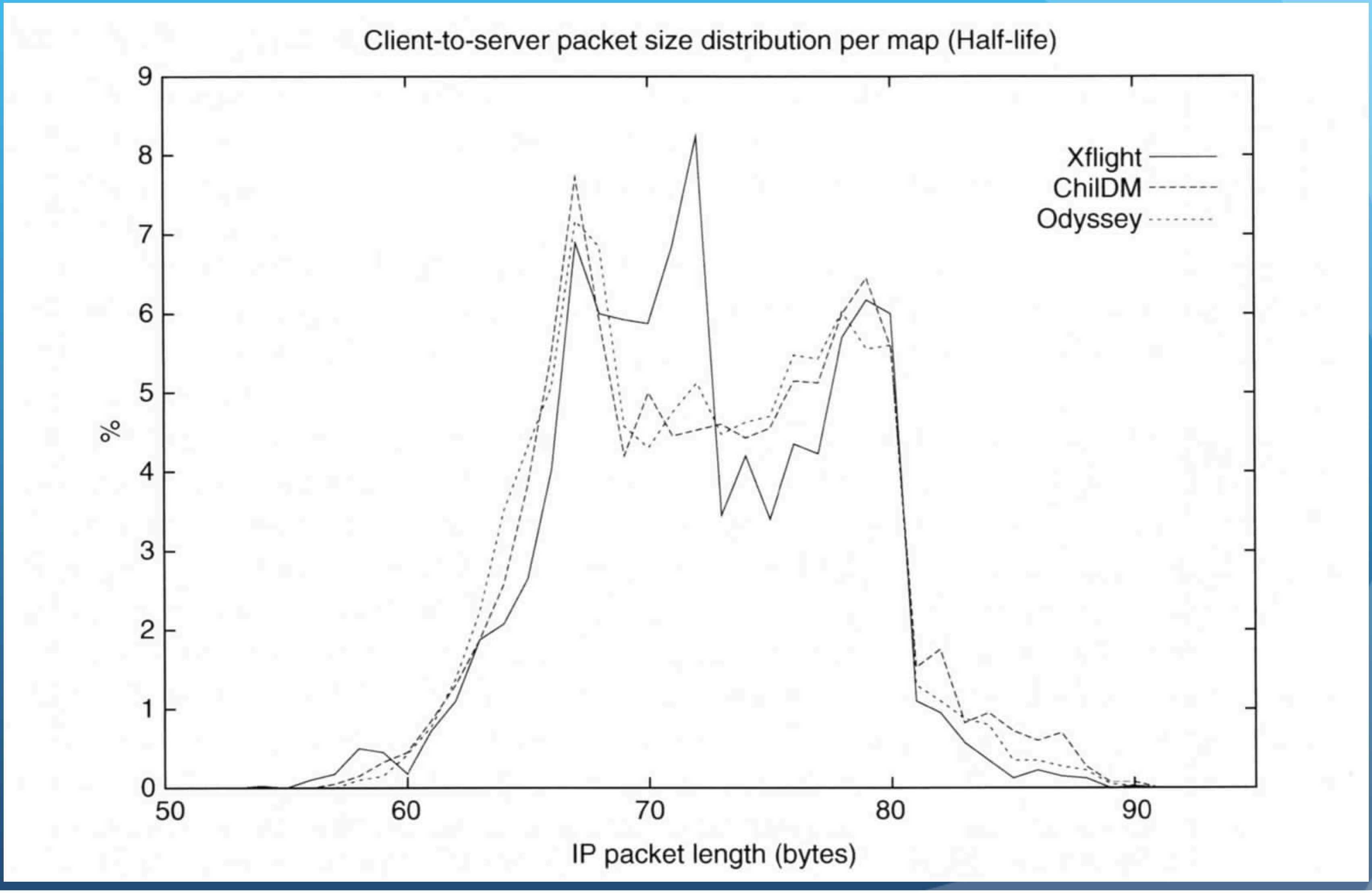
Distributions for Half Life C-S



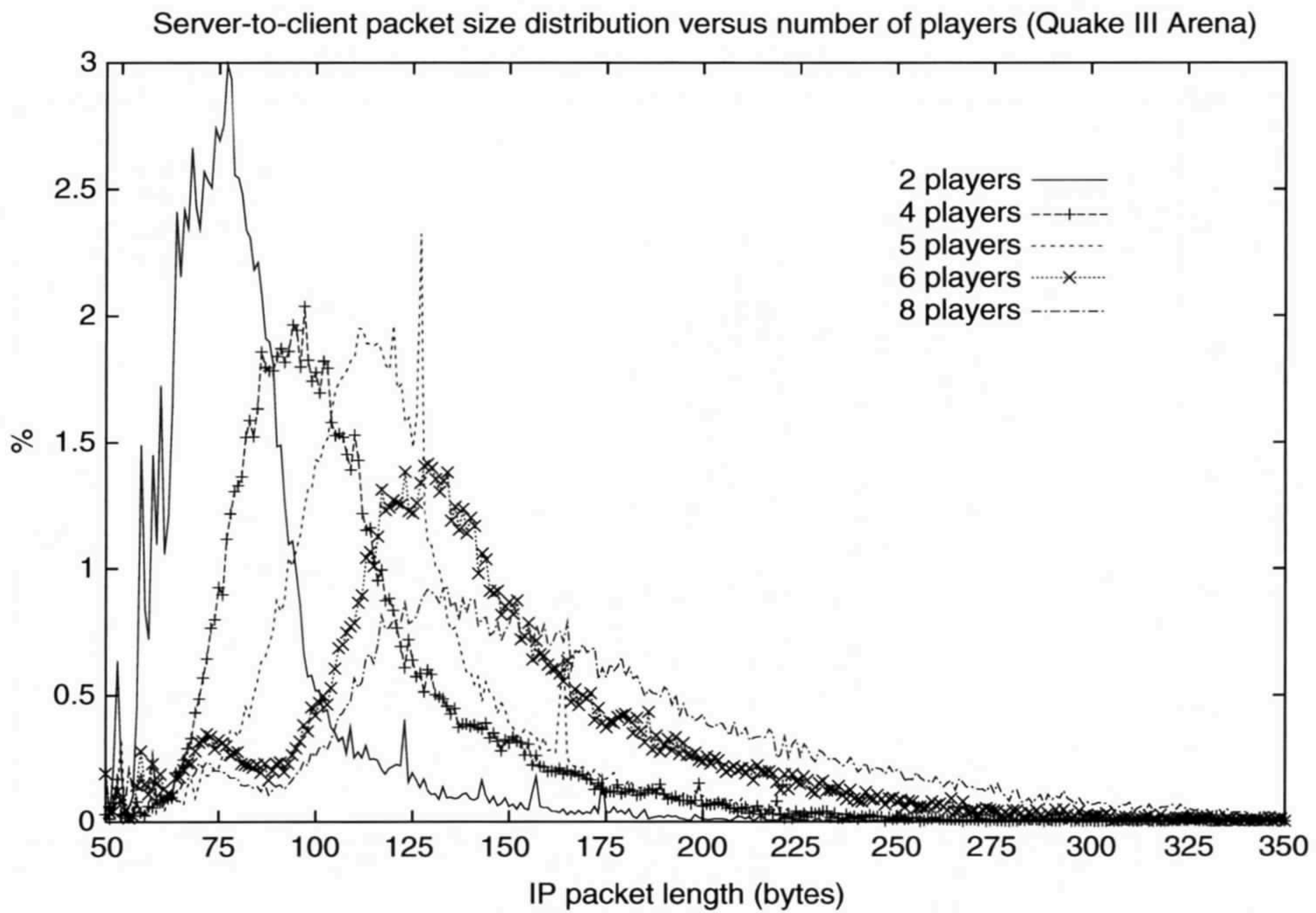
Influence of Maps on S-C packet sizes – Half Life



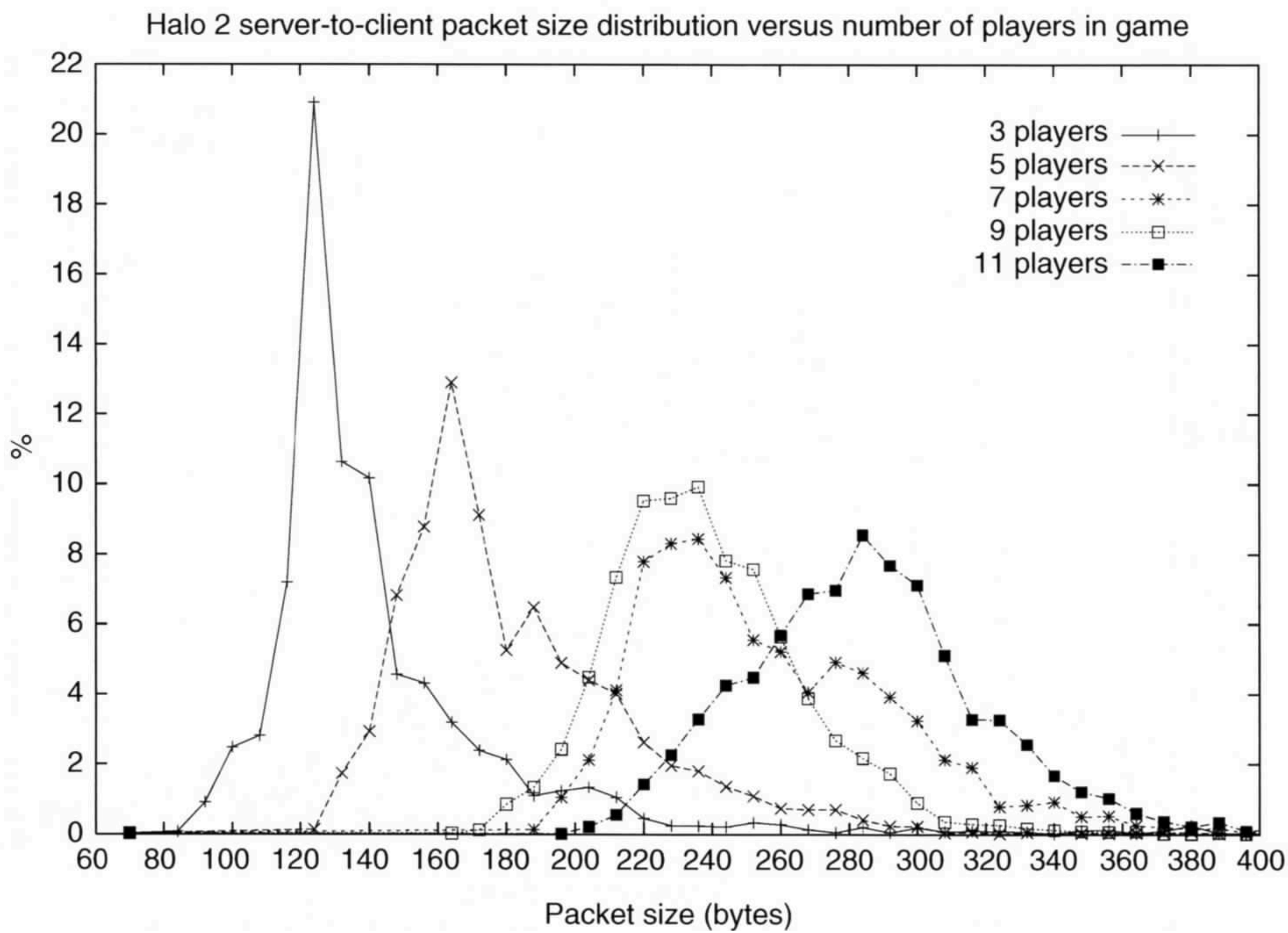
C-S packet sizes for different maps – Half Life



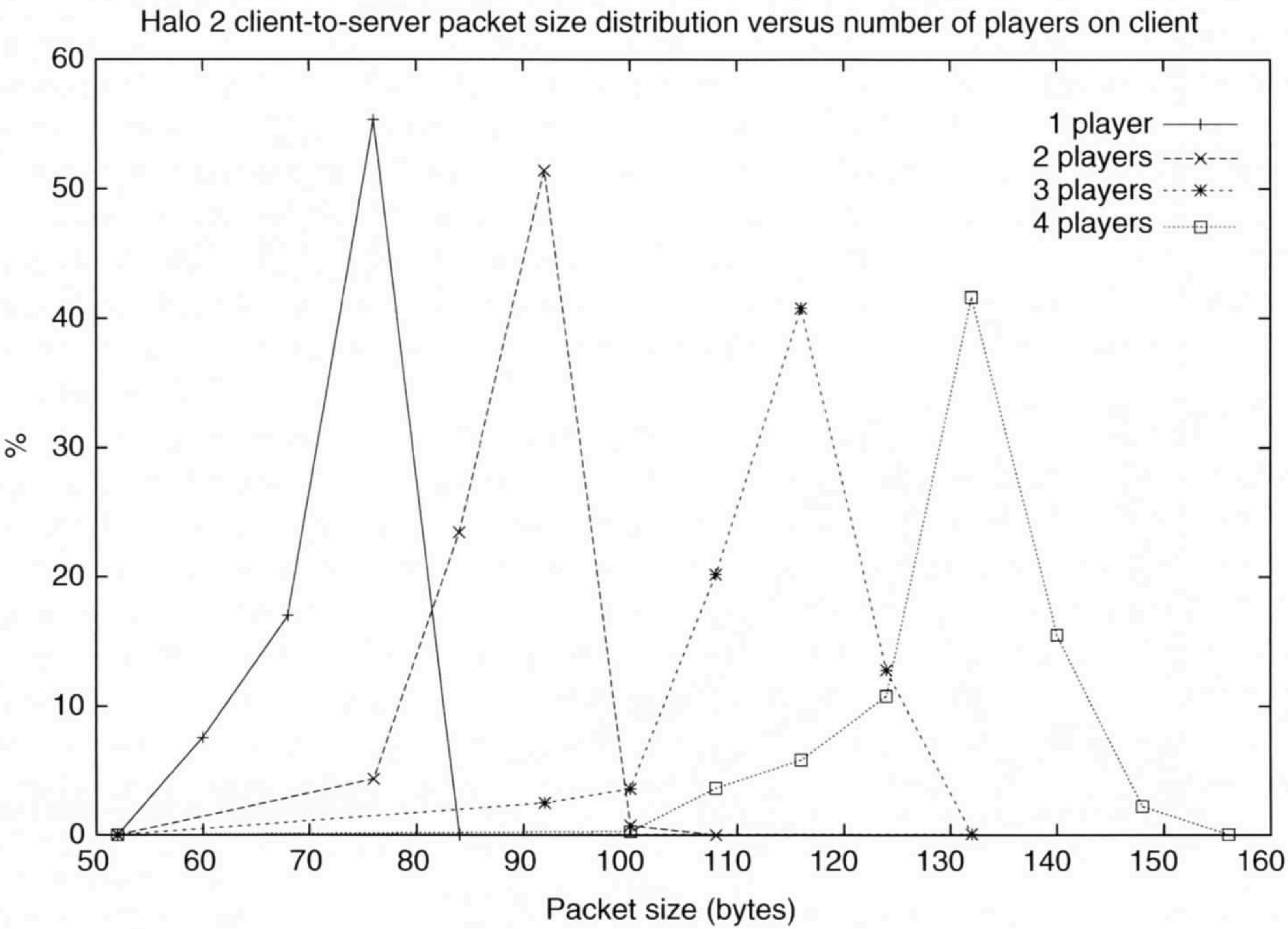
S-C packet size distribution vs number of players



S-C packet size vs no. of players – Halo 2 – Xbox Game



C-S for Halo 2 – no. of players on client

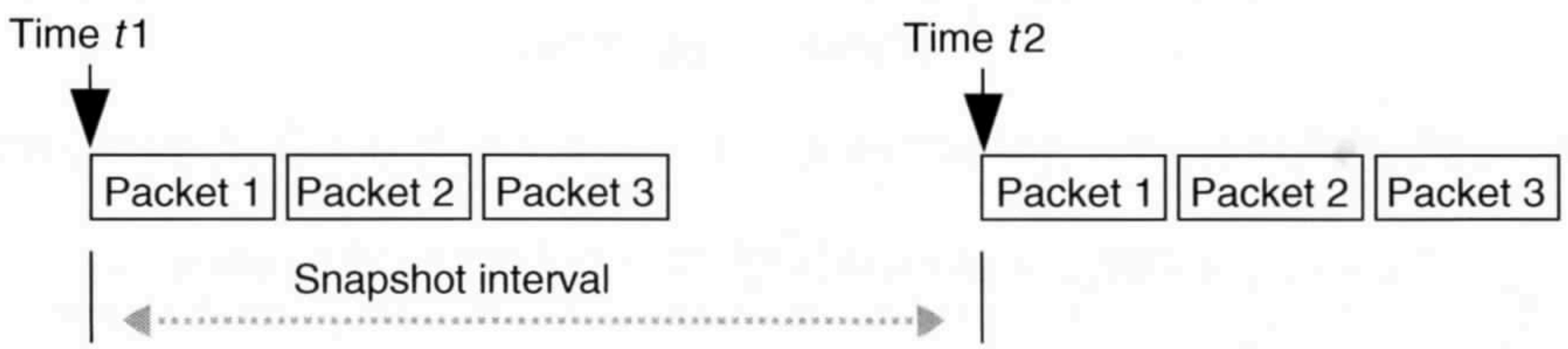


Conclusions on Packet Size

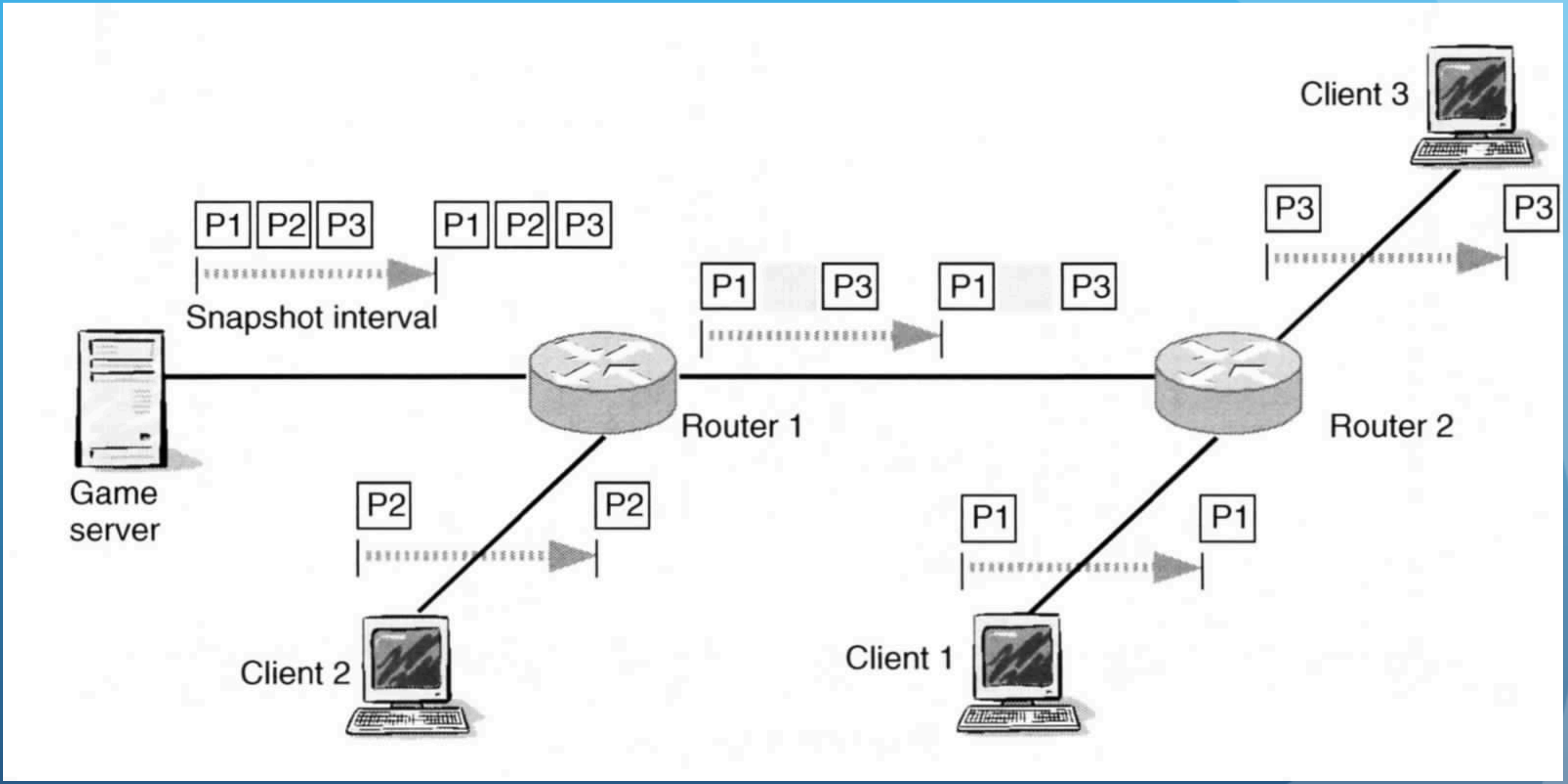
- Can make no assumptions on what the packet size for a game is going to be
- Depends on the game
- Depends on the maps
- Depends on the client
- Depends on the no. of players on a client and platform
- ………

Sub-second Inter Packet Arrival Times

- Servers send packet bursts for back to back updates to its clients every snapshot period.
- No. of packets per snapshot depends on the game design and its snapshot update strategy



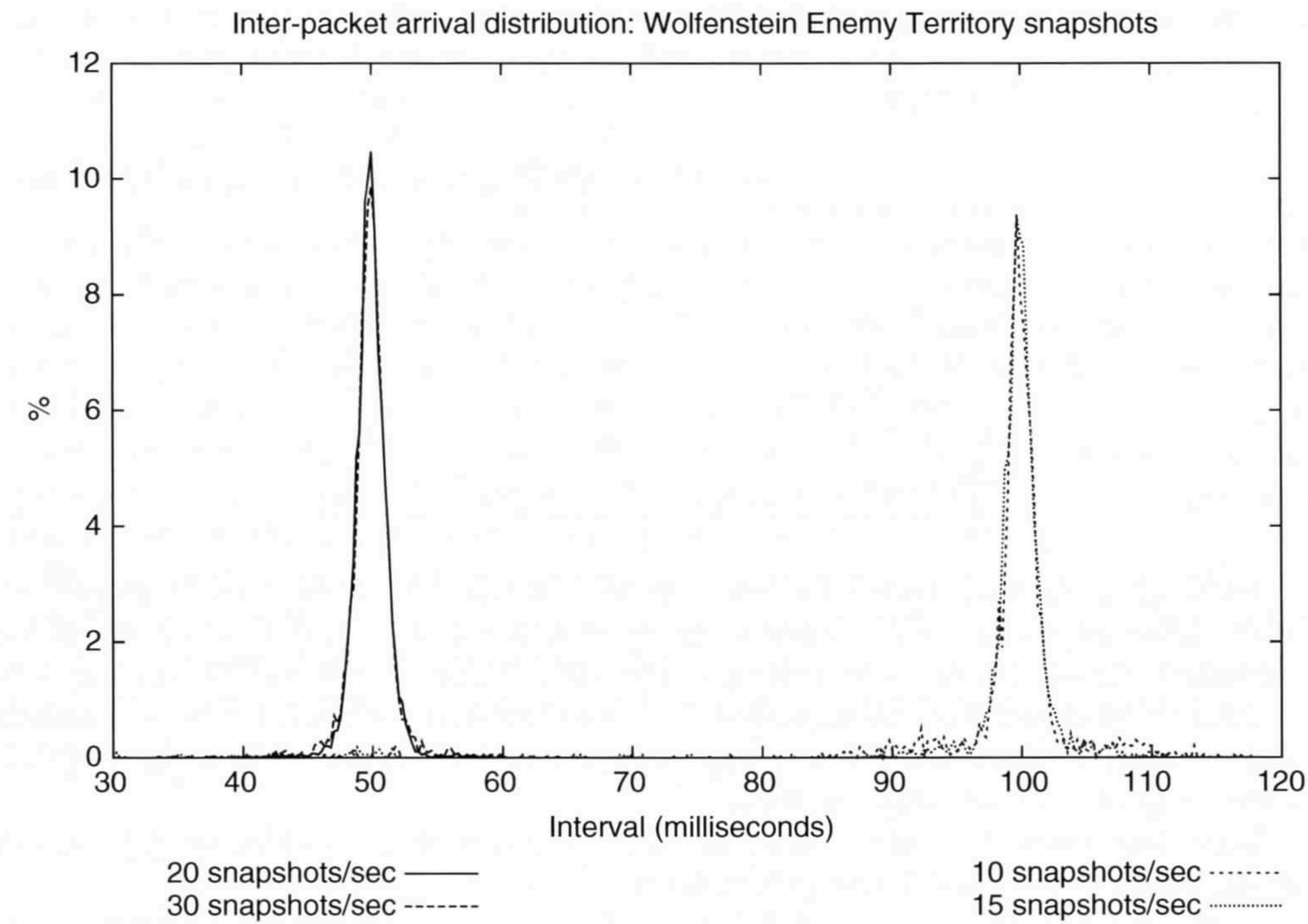
Server update scheme



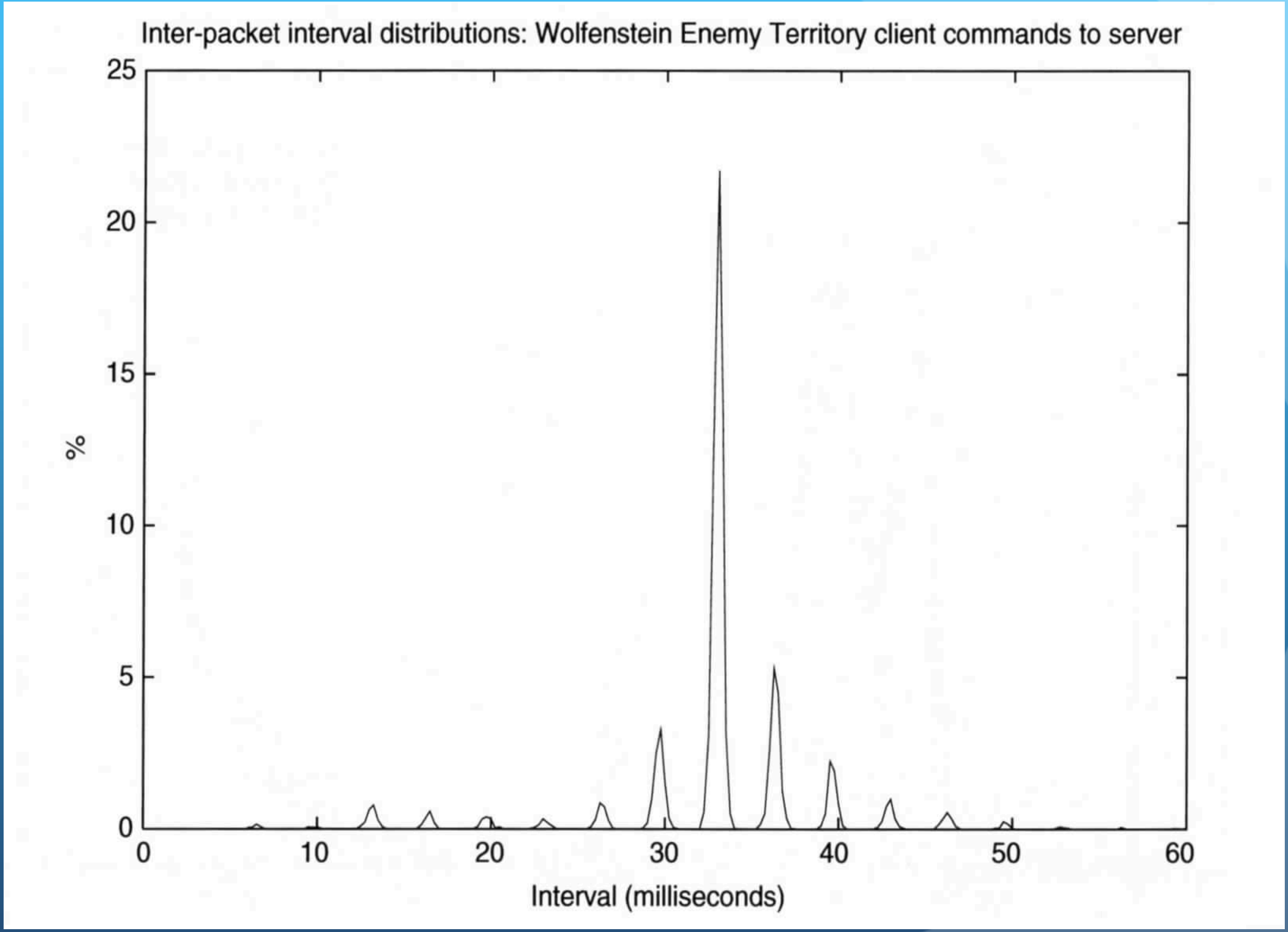
Inter packet arrival times

- Snapshot intervals do not generally vary with map choice or number of players
- Intervals will vary though with processor load - tick timing (slippage) can fluctuate by a few milliseconds resulting in jitter of snapshot transmissions
- Client to server updates more unpredictable:
  - Depends on choices client makes for updates
  - Depends on player behaviour
  - Uncorrelated streams – larger spread in C-S packet arrivals at server

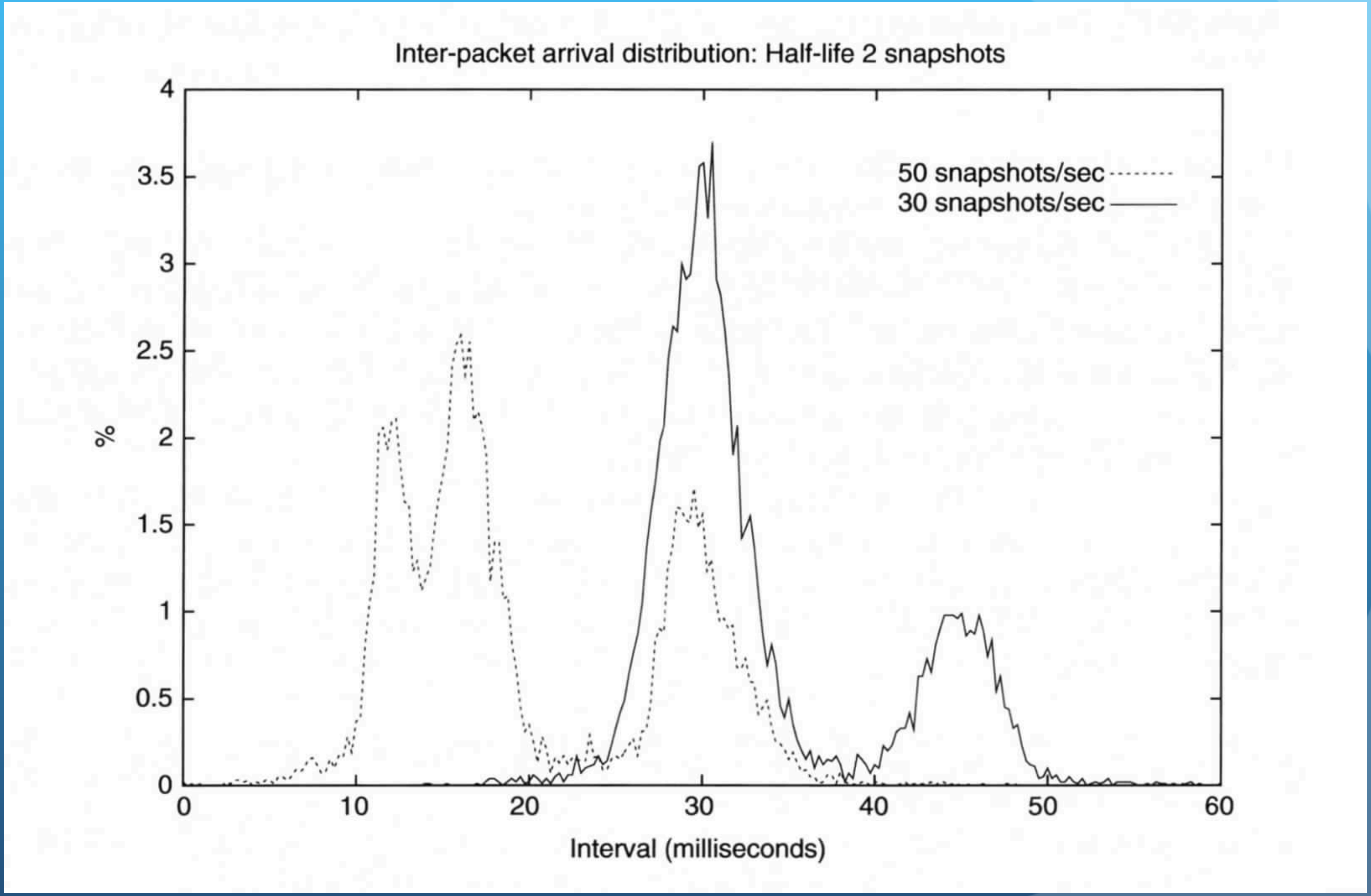
Distributions of snapshot inter-arrival times



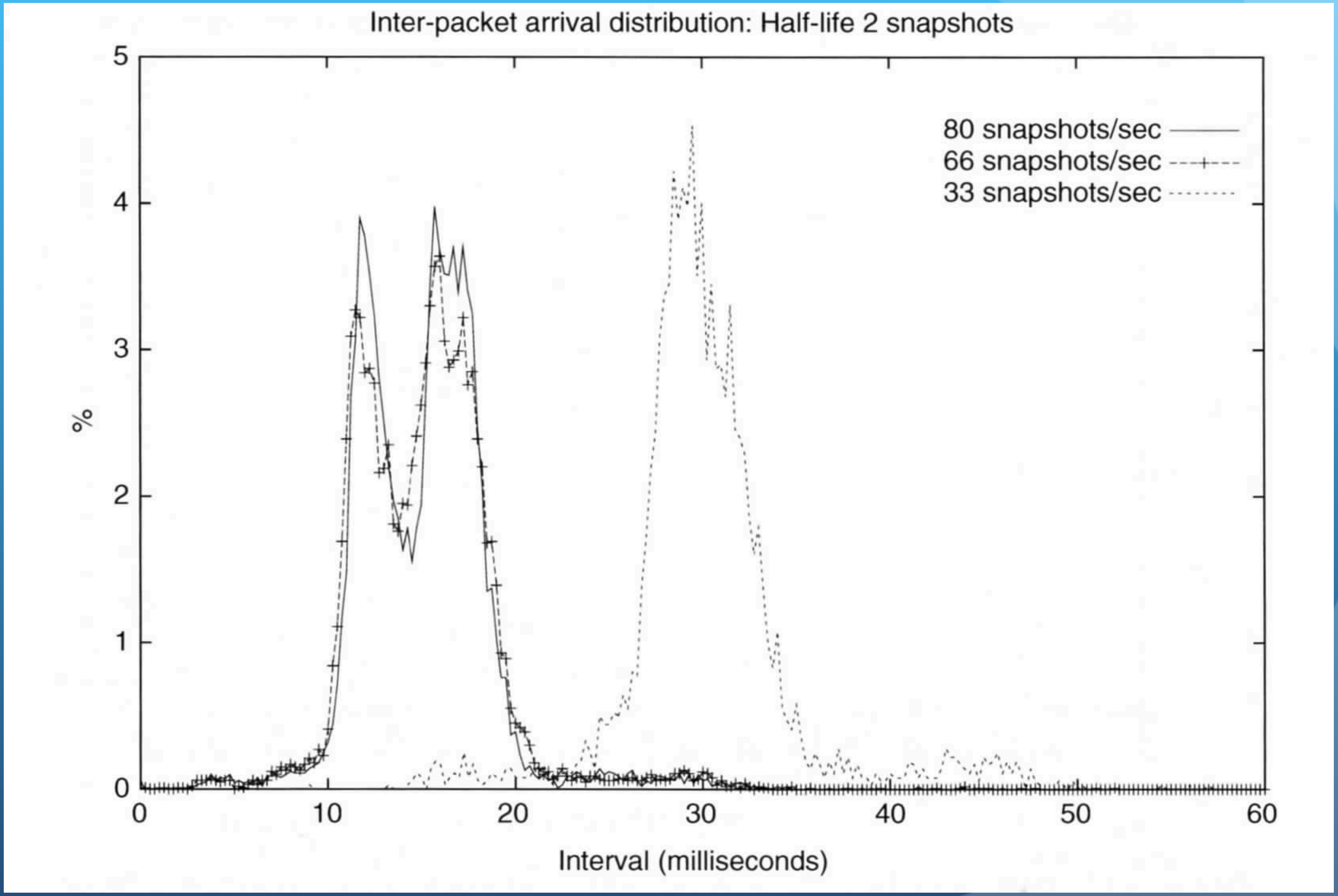
Client Commands to server



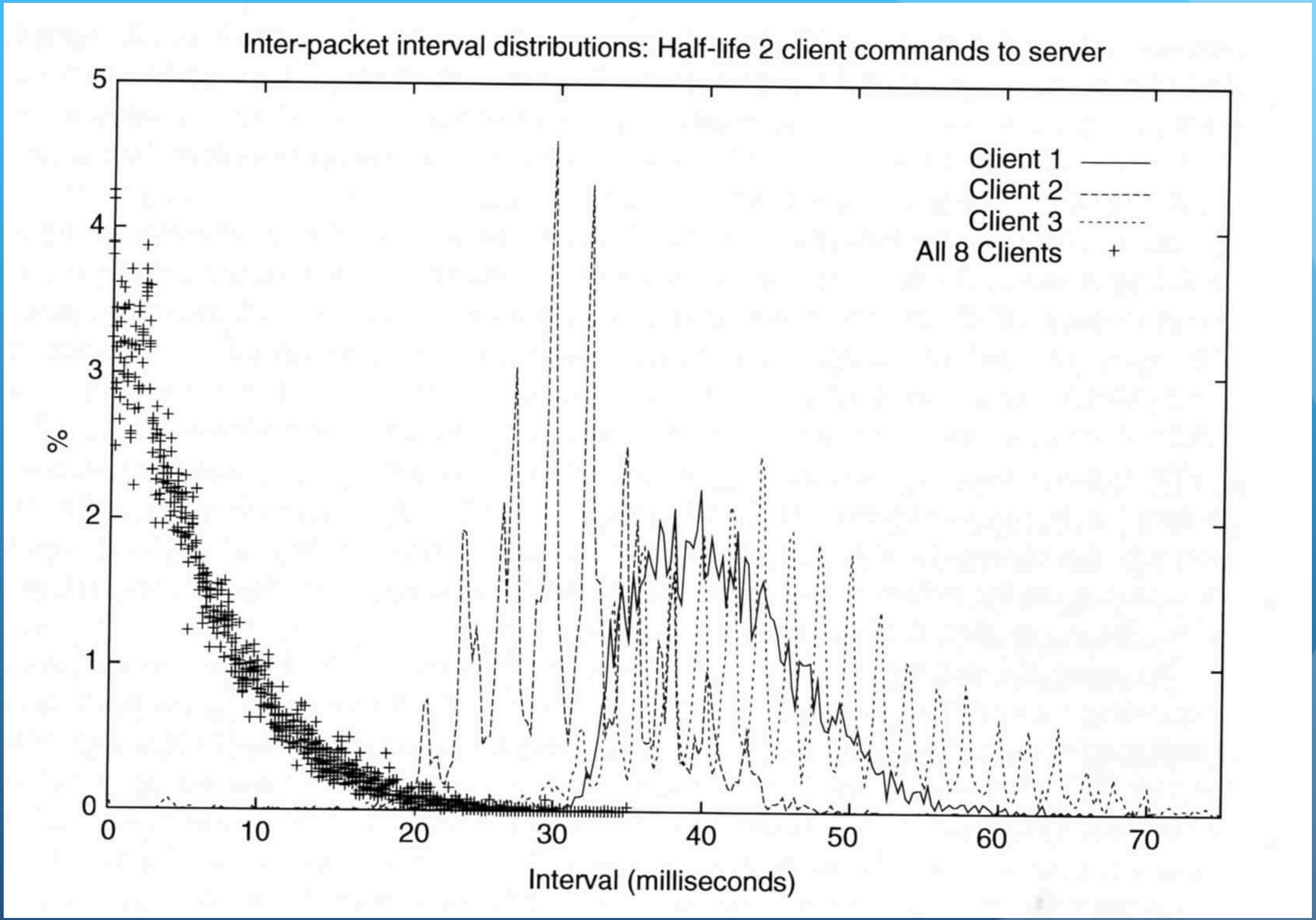
Packet intervals for custom snapshot updates – 33snapshots/sec is system update rate



Packet intervals for custom snapshot updates – 33snapshots/sec is system update rate



Client Commands for Half Life 2



Estimating Loads

- Using single value metrics (i.e., average values) hides the packet by packet realities we have seen in the inter-arrival time and packet size distributions.
- Server link bandwidth also plays a role in packet arrival distributions for network performance and load estimates

Example Scenarios

- Server sending 15 players updates every 50ms -> 300 packets per second with average interarrival time of 3.3ms.
- For a 160byte packet size -> 384Kbits/sec link required. For a 350byte packet size -> 840Kbits/sec link required
- However packets are send in a burst (all players get updated at the same time)
- Assuming two access links for the server – 100Mbs Ethernet and a 1.5Mbs T1 link – we get very different packet streams

Example Continued

- For 100Mbs link: 160 bytes -> 1472bit Ethernet transmission -> 15microsecs per IP packet, for 15 players -> burst of 225microsecs every snapshot update.
- For Ti link: 160bytes -> 1344bit PPP transmission -> 883microsecs per IP packet, for 15 players -> burst of 13.2msecs every snapshot update.
- Case 1 is much more bursty than case 2 -> worse behaviour for network
- Case 2 would not function well with a packet size of 350bytes – too close to the link bandwidth.