Translating latency requirements into resource requirements for game traffic

Chris Majewski[1], Carsten Griwodz[1,2] and Pål Halvorsen[1,2]

[1]IFI, University of Oslo, Norway [2]Simula Research Lab., Norway
e-mail: {krzys, griff, paalh}@ifi.uio.no

Abstract

Networked multi-player games constitute a demanding class of interactive distributed multimedia applications with very high commercial relevance. As such, they attract a growing number of researchers in multimedia networking. Most games use a client-server architecture, largely to prevent cheating. By analyzing the traffic of such games, we confirm that individual client-server flows consume relatively little bandwidth. Thus latency, rather than bandwidth, is the critical parameter when provisioning this class of applications. In order for commercial game services to ensure low-latency operation, resource reservation must be explored. In this paper, we investigate options for a DiffServ-style reservation on part of the path between a game server and sets of clients. We show how a token bucket shaper can be parameterized based on a target end-to-end latency, and discuss the implications for a network infrastructure. We use the shaper to quantify the burstiness of game traffic and the correlation between individual flows, with a view to the limitations this imposes on resource reservation for aggregate (multiplexed) flows.

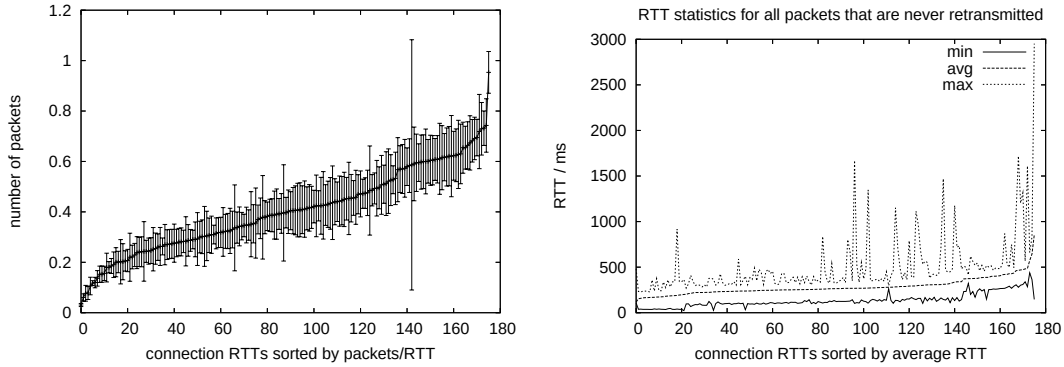
Keywords

Networked games, packet traces, resource requirements

1 Introduction

Networking researchers have recently demonstrated a growing interest in networked multiplayer games as a demanding example for distributed interactive multimedia applications. The computer games industry is large and has been a more important factor in home computer development than other multimedia applications including conferencing, Internet telephony and media streaming. In the context of networking research, games are mostly interesting because the kind of traffic that they create highlights possible developments in future, highly interactive applications. The uptake was initiated by the work on MiMaze (Gautier and Diot, 1998) and has led to the NetGames workshops (Wolf, 2002). In this paper, we investigate the considerations that are necessary to use reservation for aggregate game flows.

The public Internet offers no service guarantees to end users. Thus, neither available bandwidth nor limits to end-to-end latency are guaranteed. Both are mainly due to router queues that grow until packets must be dropped. Usually, each individual data stream backs off in order to alleviate congestion. For games traffic, this is impossible. In Anarchy Online (AO) for example, every single client-server stream is so thin that it is an exceptional occurrence when two packets



(a) Packets per average RTT

(b) Average client RTTs

Figure 1: Packets per average RTT and standard deviation in Anarchy Online

of the same stream are sent within the same round-trip time (RTT) (figure 1). It is therefore necessary to look differently at gaming traffic. Games have several concurrent flows that might contribute to congestion in the vicinity of the server. When competing for the bottleneck bandwidth, these flows will experience considerable packet loss. For large-scale games, network resources reservation, for example under a DiffServ regime, could alleviate the problems by separating games traffic from other traffic. To do this, we must better understand how games traffic behaves in order to recommend a reservation style to achieve the required latencies.

2 Relevant Game Types

Improving the performance of interactive games requires an understanding of games traffic. The networking requirements of most multiplayer games fit into one of the following categories, and more recent games may include elements from more than one group:

First Person Shooters (FPS) involve a high percentage of combat that requires low response times. The number of messages (update events such as players' position and actions) is high. Player satisfaction is disturbed mainly by experienced latency, which may be due to network latency or the need to retransmission. Pantel and Wolf (2002) have shown that this latency becomes detectable at about 100 ms and makes game play impossible at approximately 200 ms. This game type is the most demanding in terms of latency requirements.

Role Playing Games (RPGs) are visually similar to FPSs games, but the pace of player interactions in RPGs is usually less intense. The kind of user actions is more varied and can have a large influence on the game state. Online RPGs often aim at supporting a large number of players making scalability an important issue. This game type is most demanding in terms of the number of concurrent flows.

Real-Time Strategy (RTS) games are not particularly sensitive to network latency and jitter. Sheldon et al. (2003) and Bettner and Terrano (2001) studied them in more detail and showed that they have no particular latency or bandwidth requirements, i.e., we ignore them in this paper.

Today, most commercial multiplayer games are implemented in a client-server fashion, using either a single server or a server cluster. Clients are not interconnected and have an individual connection to the server. In this way, game operators reduce the chances for cheating,

they ensure the anonymity of players and simplify administration. Based on two sets of traces from commercial games, we take a closer look at network resource reservation and at the influence and limits of traffic shaping in the games scenario. We find that server-to-client and client-to-server communication must be addressed separately, although bandwidth and latency requirements are similar for both directions. The distinguishing factors are the following:

Server-to-client: Servers identify groups of clients to which they send the same information. They could use multicast for this, but even if this is lacking, they can determine the relevance of data and apply shaping and dropping in a cooperative manner for all flows. It is possible to place servers strategically such that they are well-connected to relevant networks and close to ingress nodes of networks that support reservation.

Client-to-server: Clients may generate events either on behalf of interacting users or cyclically. They will usually share their access network with a small number of competing players and a much larger amount of traffic from other applications. Usually, they will not be close to an ingress of a network that supports reservation.

3 Related Work

Chambers et al. (2005) investigated traces of several games concerning the question of how predictable game workloads are. They considered mainly FPS games, but also the massively multiplayer online role-playing game (MMORPG) Neverwinter Nights™. Their conclusion is that games traffic vary strongly with time and with attractiveness of the individual game. They approach the issue of multiplexing gain as a long-term multiplexing problem, where several games should be hosted together and with other services. In this paper, we consider multiplexing on the time-scale of end-to-end delay.

Fitting multi-player game traffic to probability distributions is described by Borella (2000). The feasibility of aggregating game traffic to achieve statistical multiplexing gains has also been previously studied. Notably, Busse et al. (2004) present an admission control algorithm and its suitability for a simple game whose bandwidth usage approximately follows a Poisson distribution. While this traffic makes resource reservation relatively easy, it is not typical for games traffic. In contrast to earlier resource reservation approaches via ATM and RSVP, the hot approaches at this time, DiffServ and/or MPLS, do not leave room for bursty, latency-critical traffic. However, research is starting to appear that might lead to an understanding of reservation for traffic aggregates that will allow guarantees for it (Allalouf and Shavitt, 2005). Sharing of the bottleneck capacity has been investigated by in an environment without resource reservation has been investigated, e.g. by Balakrishnan et al. (1999). However, this approach requires that some streams give up bandwidth in favor of others.

4 Experiments

Since the acceptable end-to-end delay is known only to the game designer or developer, we must use this as a variable in our investigation. Independently of this, burstiness in flows must be tolerated if events are correlated on the timescale of the acceptable end-to-end delay.

We propose to use resource reservation for aggregates of game flows that connect the network edge of autonomous systems with a large player population to the games server. Such reservations require the means for formulating application-dependent traffic specifications. Aggregating flows should allow for smoothing of the streams, i.e. a burstiness that is less than and an average rate that is higher than for the sum of the individual streams. Smoothing streams

with correlated bursts, however, may require considerable buffering and an increase in end-to-end delay for the individual flow, which is particularly bad for games.

We analyze packet traces (`tcpdump` files) from two games. CounterStrike (CS) is a multiplayer FPS using a client-server architecture to communicate over UDP. The upstream bandwidth per client is in the order of 1500 bytes/sec on average, with around 20 packets being sent per second. More details can be found in Wu-chang Feng et al. (2002). Anarchy Online (AO) is an MMORPG using a client-server architecture that communicates using TCP. The upstream bandwidth per client is 250 bytes/sec on average, with about 3 packets being sent per second.

4.1 Burstiness

Burstiness is typically expressed as some measure of the arrival rate distribution X, such as the variation coefficient $\sigma^2(X)/E^2(X)$. We deal with flows which may have varying packet rates and packet sizes. Thus, in order to quantify the arrival distribution, we would need to identify a relevant time interval Δt over which the arrival rate is measured. Since bandwidth is typically measured in bits *per second*, one second is the typical choice for Δt. However, games require sub-second delay bounds, and we therefore use a simulated token bucket filter (Wang, 2001) to characterize burstiness independently of time scale.

The peak rate is set arbitrarily high relative to the arrival rate; for instance, the line speed of a core Internet router. The initial token rate is set to the average arrival rate λ of the traffic (in bytes per second). The initial bucket depth is set to zero bytes. We now systematically increase the token rate and bucket depth (or FlowSpec), and measure the resulting queuing delays. The ranges of these parameters are chosen so as to show an eventual convergence of queuing delay to nearly zero for the given traffic data. In the experiment shown in figures 2 and 3, the token rate was increased in 8 exponentially increasing steps, up to 4λ bytes per second. The bucket depth was increased in 20 steps, up to $\lambda * 1sec$ bytes.

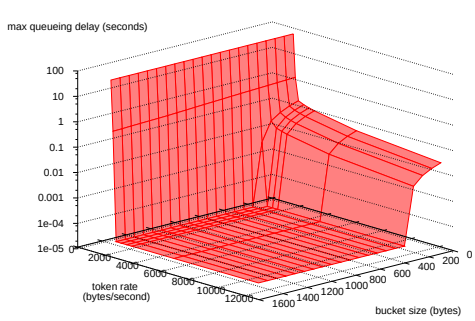
By plotting maximum queuing delay against the token bucket parameters (see for example Figure 2(a)), we obtain a visualization of the flow's burstiness. For each scan line in the plot corresponding to a constant token rate, each drop-off in delay indicates a burst. Specifically, if a drop-off occurs at depth d, it indicates the presence of d-byte bursts. This is because a bucket of depth d allows bursts of up to d bytes to "pass through" at the (significantly higher) peak rate, so that they no longer contribute to the queue length.

As the token rate is increased, queue lengths gradually converge to nearly zero. The token rate of interest is the one at which, for some reasonable[1] bucket depth, the corresponding queuing delay meets the application-specific delay bound ζ. The higher this rate is with respect to the average arrival rate, the more bursty the flow.

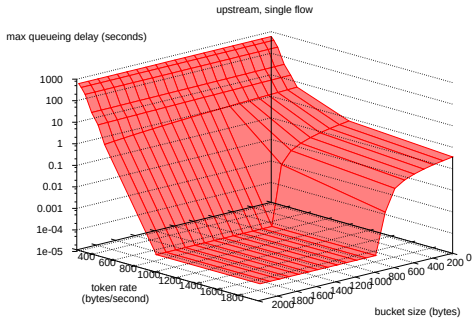
Upstream traffic: Figure 2(a) shows the upstream component of a typical individual client-server flow of our CS trace, and figure 2(b) shows the same for an AO trace. Consider that the arrival of each b-byte packet constitutes a burst of size b at a time scale corresponding to the packet rate. The largest upstream IP packet size in the CS flow is 289 bytes. This corresponds to the bucket depth at which the ridge in the right rear of Figure 2(a) has rolled off to height (i.e. queuing delay) zero [2]. In the AO flow, the largest packet size is 704 bytes.

[1] The definition of "reasonable" depends on what the network operator provides and what we are willing to pay.

[2] Actually, the delay floor is just above zero, due to the service delay resulting from a finite peak rate.

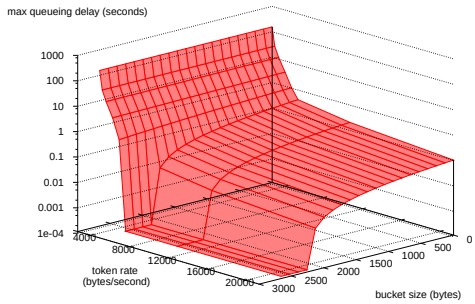


(a) Counterstrike traffic

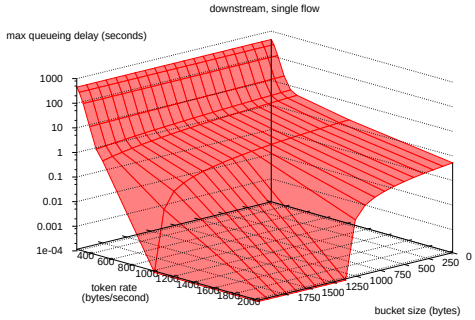


(b) Anarchy online traffic

Figure 2: Upstream token bucket filter for a single stream



(a) Counterstrike traffic



(b) Anarchy online traffic

Figure 3: Downstream token bucket filter for a single stream

Downstream traffic: Figures 3(a) and 3(b) show the downstream components of typical individual client-server flows in CS and AO, respectively. The largest downstream IP packet size in the CS flow is 1428 bytes, 1500 bytes in the AO flow. For CS, the ridge in the right rear of the plots rolls off at bucket depth greater than size of the largest packet, indicating bursts where two or more packets arrive in rapid succession. For AO, the situation is less clear. The role-off occurs for packet sizes less than the MTU size, and increasing the token rate case leads to a visible reduction in queuing delay even for bigger token bucket sizes. This is due to the use of TCP; packets are not the unit of event generation but packets are sent in pairs when more data must be sent than a single TCP segment can hold. The delay bound does therefore correspond to the length of actual bursts, as soon as the accumulated tokens are consumed.

In both downstream cases, delays converge more slowly along the token rate axis than in the upstream case: a relatively higher rate is required for the token bucket to take effect, indicating higher overall burstiness with respect to the upstream traffic. However, the effect is much less pronounced for AO than for CS. We can understand this when we consider that each CS client requires updated information about many other clients at any given time, while an AO client receives only updates for the player's immediate surroundings as well as effects of other players' actions.

4.2 Correlation and Multiplexing Gain

Traffic specification in many distributed applications makes the starting assumption that users' interactions are mutually independent and can be modeled by Markov processes. In the case of games, however, it is natural to assume that game events are strongly correlated. Both FPSs and RPGs lead to co-location of players in the virtual world, and actions are frequently determined by knee-jerk reactions. However, these reaction are short compared client-server delay and developer can compensate for a certain amount of delay ζ. We must therefore investigate the correlation to determine whether flow aggregation would yield a relevant multiplexing gain at a temporal resolution that is implicitly defined by the maximum latency that the games developer considers acceptable. To do this we consider a limited peak rate that we call the *minimal service rate* ρ that upholds ζ. A limited ρ has two effects. First, the bucket depth and token rate alone are no longer sufficient to stay within the delay bound ζ, they have to be defined in such a way that packets drained from the buffer are processed early enough. Second, if the expressive power for a resource reservation system is limited, as in the case of DiffServ's assured forwarding per hop behavior, the peak rate must be reserved to meet the delay bound ζ. The ratio ρ/λ gives a measure of overall burstiness. For one representative game flow for both CS and AO, figure 4 plots this ratio over different values of ζ. Obviously, ζ must be very large (in the order of 1 to several seconds) to achieve a peak rate close to the average data rate for individual flows.

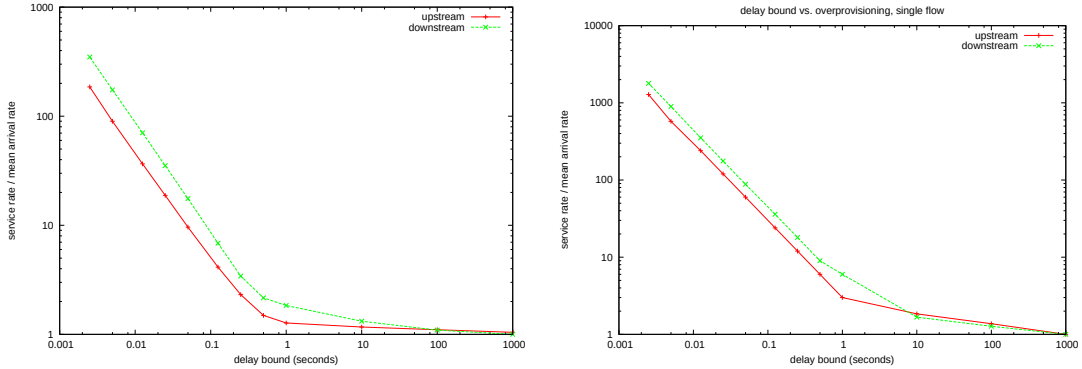
We are therefore interested in determining whether the minimal service rate for a flow aggregate is closer to the average rate of the aggregate. We do this by looking at the multiplexing gain. Consider a set S of concurrent flows. Comparing $\rho(S)$, the minimal service rate for the aggregate, with the sum of the minimal service rates $\rho(i)$ for each individual flow $i \in S$ gives us the multiplexing gain $mg, 0 \leq mg < 1$: $mg = 1 - \frac{\rho(S)}{\sum \rho(i)}$

One would expect that a set of flows with mutually independent Poisson packet arrival distributions would tend to yield high multiplexing gain. Conversely, highly correlated flows would offer little multiplexing gain, because their peaks are likely to coincide. As an example, we look at the 10 largest flows that overlap for at least 5 minutes in both the CS and AO traces, and consider a queuing delay bound $\zeta = 25ms$. We consider all combinations of these flows. Figure 5 shows the multiplexing gain as several flows are aggregated. Both graphs show that significant multiplexing gain is possible, suggesting that peaks in individual flows are not strictly correlated in the aggregate. It does not seem possible to improve downstream aggregation beyond 50% in either game, while aggregation of upstream flows yields an important gain. Especially the upstream flows in AO are nearly perfectly uncorrelated at this time-scale.

5 Discussion

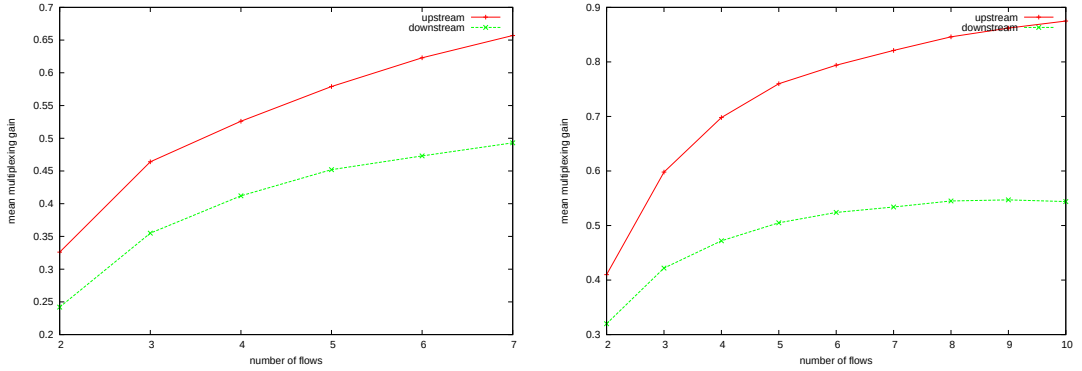
For the upstream traffic in games, we would have expected significant burstiness at timescales of one second or more, due to their interactive nature. Its absence implies that game design prevents a considerable increase in upstream traffic in highly active situations. At sub-second resolution, their burstiness results primarily from packetization of the data. Also any correlation between bursts in several streams is so low that it seems to be a random occurrence. We attribute this to the observation that expected correlation is hidden because the variations in end-to-end delays are on the same timescale as human reaction time, which varies as well. Thus, correlation between the upstream components of concurrent flows is sufficiently limited to allow significant statistical multiplexing gains.

Downstream traffic, on the other hand, was more bursty than we anticipated. Although the



(a) CounterStrike (b) Anarchy online

Figure 4: Burstiness of a representative stream



(a) CounterStrike (b) Anarchy Online

Figure 5: Multiplexing gain vs. number of concurrent flows

packet rate of this traffic is fairly constant, the packet sizes can vary between $50 - 1500$ bytes. The downstream traffic is similar between flows, presumably because the game state updates sent out by the server are largely the same for all clients. Thus, the downstream components of concurrent flows are less suitable for aggregation.

If a games provider intends to exploit multiplexing to protect games traffic from cross-traffic, a trace of the game traffic can provide the relevant token bucket parameters for a DiffServ service specification. For upstream traffic, aggregation works well, so reservations can be made for the aggregate flow. For downstream, a game provider could establish proxy servers at the egress of a reservation. It would then become possible to use multicast between the server and the proxies, where traffic could be converted to unicast streams for the individual client. However, the alternative would be not to perform upstream aggregation at all because the traffic is reasonably smooth, and only use reservation for downstream traffic. In the latter case, no proxy would be required, only DiffServ marking or MPLS labeling.

6 Conclusion and Future Work

We have presented a method for characterizing burstiness and correlation of concurrent game flows at different time resolutions. We applied this method to the FPS game CounterStrike and the MMORPG Anarchy Online, which use the UDP and TCP protocols, respectively. The ap-

proach, can determine the multiplexing gain and minimal service rate for aggregate game flows at these resolutions. We find that upstream flows are only weakly correlated, allowing resource reservation with little overhead. Downstream flows are highly correlated. Through aggregation, a considerable multiplexing gain is still achievable, which makes it possible to reservation a higher average rate in favor of a low peak reservation compared to the unaggregated streams.

In future work, we will takes closer look at the issues that lie in very thin TCP streams. For the game flows that we observed, neither congestion control nor fast retransmission work because of it, and remedies must be investigated. We will also look at kernel enhancements and off-loading engines that make fast packet aggregation and filtering on servers and proxy servers possible.

Acknowledgments

We are grateful to Wu-chang Feng and his colleagues at Portland State University for packet traces from their highly popular CounterStrike server. We would also like to thank Funcom for the traces of their commercial MMORPG, Anarchy Online. In accordance with the Condor license agreement, we acknowledge the Condor cluster at `ifi.uio.no` for performing some of our simulations.

References

M. Allalouf and Y. Shavitt. Achieving bursty traffic guarantees by integrating traffic engineering and buffer management tools. In *Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Student Workshop*, Miami, FL, USA, 2005.

H. Balakrishnan, H. Rahul, and S. Seshan. An integrated congestion management architecture for Internet hosts. In *Proceedings of the ACM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 175–187, September 1999.

P. Bettner and M. Terrano. 1500 archers on a 28.8: Network programming in Age of Empires and beyond. In *Game Developers Conference*, San Jose, CA, USA, 2001.

M. S. Borella. Source models of network game traffic. *Elsevier Computer Communications*, 23(4):403–410, Feb. 2000.

M. Busse, B. Lamparter, M. Mauve, and W. Effelsberg. Lightweight QoS-support for networked mobile gaming. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 85–92, Portland, OR, USA, 2004.

C. Chambers, Wu-chang Feng, S. Sahu, and D. Saha. Measurement-based characterization of a collection of online games. In *Proceedings of the USENIX Internet Measurement Conference (IMC)*, pages 1–14, Berkeley, CA, USA, 2005.

L. Gautier and C. Diot. Design and evaluation of MiMaze, a multi-player game on the Internet. In *Proc. of IEEE Multimedia Systems Conference*, June 1998.

Wu-chang Feng, F. Chang, Wu-chi Feng, and J. Walpole. Provisioning on-line games: a traffic analysis of a busy Counter-strike server. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement*, pages 151–156, Marseille, France, 2002.

L. Pantel and L. Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Miami Beach, FL, USA, 2002.

N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The effect of latency on user performance in Warcraft III. In *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, pages 3–14, Redwood City, CA, USA, 2003.

Z. Wang. *Internet QoS: Architectures and Mechanisms for Quality of Service*. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 2001. ISBN 1558606084.

L. Wolf, editor. *Proceedings of the Workshop on Network and System Support for Games (NETGAMES)*, Braunschweig, Germany, Apr. 2002. ACM Press.