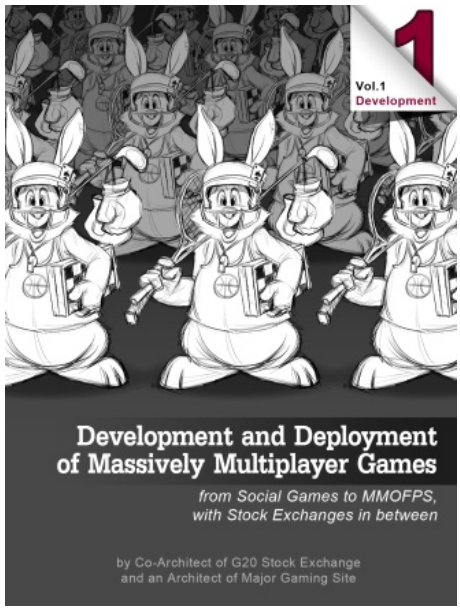
Chapter III. On Cheating, P2P, and [non-]Authoritative Servers from "D&D of MMOG" book

posted November 9, 2015 by "No Bugs" Hare, translated by Sergey Ignatchenko

[[*This is Chapter III from the upcoming book "Development&Deployment of Massively Multiplayer Online Games", which is currently being beta-tested. Beta-testing is intended to improve the quality of the book, and provides free e-copy of the "release" book to those who help with improving; for further details see "Book Beta Testing". All the content published during Beta Testing, is subject to change before the book is published.*

To navigate through the book, you may want to use Development&Deployment of MMOG: Table of Contents.]]



While developing an MMOG, there is one extremely important thing to remember about. This thing is almost non-existent for non-multiplayer games, and is usually of little importance for LAN-based multiplayer games. I'm speaking about player cheating.

Player cheating is One Big Problem for all successful MMO games. The problem is that ubiquitous for such games, that we can even say that if you don't have players cheating – it is either you're not looking for cheaters thoroughly enough, or you are not successful yet.

If you're popular enough, they Will find Reasons to Cheat

Two things are infinite: the universe and human stupidity; and I'm not sure about the universe.
— Albert Einstein —

You may think that players have no reason to cheat for your specific game. For example, if your game has nothing which can be redeemed for money – you may think that you're safe regardless of your number of players. In practice, it is exactly the other way around: if your game is popular enough, they will find a reason to cheat regardless of direct redemption options.

"The thing was that the players were able to push all their "play chips" on the table; while

Just one real-life example. Once upon a time, there was a poker site out there, where players got "play chips" for free, and were able to play with them. There was nothing which can be done with that "play chips", except for playing (so they cannot be redeemed for anything-which-has-real-value). At that time it seemed to the team that there was no reason to cheat on the site, none whatsoever, right? The real life has proven this assumption badly wrong.

The thing was that the players were able to push all their "play chips" on the table; while doing it has made very little sense, they were using the amount of their chips to imply "how good the player I am". And as soon as they started to brag about the

doing it has made very little sense, they were using the amount of their chips to imply "how good the player I am"

play chips, one guy has thought "hey, I can sell these play chips on eBay, and they will pay!" And as soon as eBay sales went on, the cheating went rampant (with lots of multiple accounts to get those free chips, and with lots of "chip dumping" to pass them along).

While I (and probably you) cannot imagine spending 20 real dollars to get two million of "play chips" with no other value than being able to boast that you're a Really Good Player (while you're not) – we know for sure that there is a certain percentage of people out there who will do it. If you're big enough, such things will happen for sure, the only question is about your site popularity and probabilities.

BTW, exactly the same aspect of human nature is currently successfully being exploited for monetization purposes by numerous modern games (especially social games); however, at this point we're not concerned about the exploiting human vices ourselves (it is a job for monetization guys, and beyond the scope of this book), but about technical aspects of preventing cheating.

The moral of the story:

Even if you think that players have zero reason to cheat
Given your site is popular enough, they will find such a reason

As soon as your game reaches 1'000 simultaneous players, you're likely to have singular cheaters. And when the number goes up to 100'000, you can be 100% sure that cheaters are there (and if you don't see them – it just means that you're not looking for them hard enough). While it depends on the kind of goodies you provide to your players, and numbers above may easily vary by an order of magnitude, I daresay that chances of you having a game with 100'000 simultaneous players and not having any cheaters, are negligible, pretty much regardless of what exactly is the game you're playing.

The Big Fat Hairy Difference from e-commerce

One thing to be kept in mind is that game cheaters are very different from e-commerce fraudsters. With e-commerce, those who're trying to get around the system, are either those trying to angle the promotions, or outright fraudsters.[1] When speaking about games, the reasons behind cheating are much more diverse. For players, in addition to all the reasons to cheat described above, there are many others.

For example, as it has happened with "play chips" (see [[TODO]] section above), people can cheat just to claim that they're better players than they really are. Or they can cheat because they feel that the game rules are unfair. Or they can cheat just because of (wrong) perception that "everybody else does it anyway", so they need to cheat just to level the field. Or they can just try to save some time by using "bots" for "grinding". Possibilities are really endless here.

This means that the line which separates "cheaters" from "honest players" is much more blurred with games, than in e-commerce. Throw in the fact that e-commerce fraud is an outright crime, and, say, using "bots" to avoid "grinding" is punishable at most by the ban on the site (which can be bypassed rather easily, at least unless the name on your credit cards is Rumpelstiltskin), and you will realize that

some of the people who would never ever cheat in e-commerce, will easily cheat in online games

While the number of "honest players" in online games still exceeds the number of "cheaters" by a wide margin, you cannot rely on your e-commerce experience of "Oh, merely 1% of our customers are cheating". Also you need to keep in mind that, due to much more significant interaction between players in games than in e-commerce,

unlike with e-commerce, even a small number of game cheaters can easily ruin the whole game ecosystem

Just as one example: if enough people are using bots to get an unfair advantage with your game (for example, to react to threats more quickly than a human can), your game will start deteriorate to the point of being completely unplayable. In other words: dealing with cheaters is not all about money, it is about preserving the very substance of your game.

"dealing with cheaters is not all about money, it is about preserving the very substance of your game.

[1] There are also people who want to use your site as testing grounds to improve their hacker skills or to brag about them after breaking you, and hacktivists, but fortunately, they're relatively few and far between.

Dealing with Cheaters

So, cheaters are pretty much inevitable. The question is: what can/should we do about it? In general, there are two things which need to be done.

First and foremost, you need to make sure that your architecture at least doesn't help cheaters. If it does – you will be in a Really Big Trouble as soon as your game becomes popular.

The second aspect of dealing with the cheaters is a direct cheater fighting, and it can usually (well, unless you're a stock exchange) be postponed until you deploy your game; then you need to start actively looking for cheaters, and to fix the problems as they arise. Details of the direct fighting with cheaters will be described in Chapters [[TODO]] and [[TODO]]; for now we just need to ensure that our architecture will allow to perform such cheater fighting without rewriting the whole thing.

Attacks: The Really Big Advantage of the Home Turf

When dealing with cheaters (in the realm of classical security they are usually named "attackers"), it is very important to understand the fundamental differences between two classes of the attack scenarios.

In the first class of scenarios, cheater/attacker tries to affect something which is under your direct control. This "something" can be your server, or a communication channel between the client and server. In this case you essentially have an upper hand to start with; while attacks are always a possibility, for this first class of attacks all of them are inevitably related to the bugs in your implementation.

In other words, whenever you have something which is under *your* control, you're generally safe, saving for implementation problems. Of course, there are lots of bugs to be exploited, but you do have a fighting chance, and as soon as a specific bug is fixed, the attacker will need to find another bug, which is not that easy if you've done your job properly.

Security by Obscurity is the use of secrecy of the design or implementation to provide security. A system relying on security through obscurity may

The second class of the attack scenarios is related to those cases when the attacker has your client software (or even hardware device) under his full control, and can do whatever-he-wants with it. In these cases, things are much much worse for you. In fact, whatever you do with your client software, the attackers are able to reverse engineer it and do whatever-they-want with it from that point.

The only protection you have in these attack scenarios, is some kind of obfuscation, but given enough effort (and we're not speaking about "the time comparable with life time of our sun"), any obfuscation can be broken. In terms of classical security, in this second class of attack scenarios, all you have at your disposal, is "Security by Obscurity", which is

> have theoretical or actual security vulnerabilities, but its owners or designers believe that if the flaws are not known, then attackers will be unlikely to find them.
>
> — *Wikipedia* —

traditionally not considered security at all; while we will need to resort to "Security by Obscurity" in some cases[2], we need to realize that

"Security by Obscurity",
while sometimes being the only protection available,
cannot be relied on

To summarize: when speaking about cheaters, an advantage of the "home turf" (having control over software/device) makes a huge difference. In particular, you cannot really protect software which you place into the attacker's hands. The situation in this regard is that bad, that even if you would be able to give each player a device, these devices would also be hacked (to see the spectrum of attack available, see, for example, [Skorobogatov]). In general, whatever-you-give-to-player should be considered hackable; the only thing we can do about it is to increase the cost of hacking, but preventing the hacking completely is out of question.[3]

---

[2] notably for bot fighting and for preventing duplicate accounts, where there are very few other ways of protection, if any

[3] In particular, Skorobogatov (being one the top researchers in the field), says that "given enough time and resources any protection can be broken"

Low-Impact and High-Impact Attacks

As mentioned above, we cannot really prevent 100% of the attacks on our games; some of the attacks (such as bots and duplicate accounts) are protected mostly by Security-by-Obscurity, and protection only by Security-by-Obscurity cannot be considered reliable, so some attackers will be able to slip in, at least for some time. Let's try to see what types of attacks are the most typical in gaming environment, and what is the impact of these attacks if they're successful.

Stealing User DB

One of the worst things which can happen with your game security-wise, is an attack on your user DB (the one which includes all the passwords, e-mails, etc.). It is an extremely juicy target for competitors (to have all the e-mails and to discredit your game at the same time), for disgruntled users [4], and for ordinary cheaters. The impact of such an attack is very high. Fortunately, user DB can be protected beyond "Security by Obscurity". Some details related to protection from stealing

of user DB will be described in Chapter [[TODO]].

---

[4] you can count on having your fair share of disgruntled users as soon as you have millions of players, even if you're 1000% fair and deliver on all your promises

DDoS

DDoS attacks are fairly easily to mount, and the battle really goes both on attacker's "home turf" and on your "home turf" at the same time. Fortunately, DDoS, while painful, usually do not last too long to cause too much trouble.

Affecting Gameplay

If your game is a soccer game, and somebody is able to ensure that they score a goal regardless of actual things happening on the field, you're in trouble. The very same thing applies to any kind of fight (if cheater is able to score a hit when shooting in the opposite direction, the things go pretty bad), and to any other type of competitive game in general. Even not-exactly-competitive games are subject to manipulation in this regard (especially as competitiveness is routinely introduces as different kinds of "leader boards" even to as non-competitive games as social farming).

"If your game is a soccer game, and somebody is able to ensure that they score a goal regardless of actual things happening on the field, you're in trouble.

Cheating-to-affect-gameplay will become known among the players pretty soon, and will break the trust to your game; in the extreme cases your game will become completely unplayable because of number of cheaters being too high. Therefore, the impact of such an attack can be classified as "high" (and becomes "extremely high" if the exploit is published). Whether you can protect from this type of attacks beyond "Security by Obscurity", depends on your architecture. We'll discuss the attacks related to affecting gameplay, in this chapter below.

Duplicate Accounts

Whatever your game is about, there is usually enough motivation for players to have duplicate accounts. As protection from duplicate accounts is mostly based on "Security by Obscurity" (except for paid accounts, where you can use credit card number or equivalent to identify your player), preventing duplicate accounts completely is not realistic, but we can still make it a bit more complicated for the attacker (especially on non-jailbroken phones and consoles). Fortunately, while

duplicate accounts are usually prohibited in T&C, and do affect gameplay in subtle ways, their impact on the game is usually very limited. Some ways of dealing with duplicate accounts will be described in Chapter [[TODO]].

Attacking Another User's Device

One of less common scenarios is placing a keylogger or some other kind of backdoor onto another player's device (PC/phone). Usually the aim for such an attack is to steal the user's password, but things such as "being able to make an action in the name of victim player while he's playing" are not unheard of. While technically this kind of attack is not our problem, from the user's perspective it is ("hey, somebody has logged in as me and lost that Great Artifact I had to somebody else without me even knowing about it!"), so this may need to be addressed if value of the things on player's account is high enough. Fortunately, impact of these attacks on the game ecosystem tends to be low. Dealing with them (if this is deemed necessary) is usually done with so-called two-factor authentication, which will be described in Chapter [[TODO]].

Bots

Bots (automated players) are well-known to be a part of any popular-enough MMOG. As soon as you have "grinding", there is an incentive to bypass the "grinding" and get the end result without spending hours (yes, for a good game, many people find that the "grinding" itself is fun, but this doesn't mean that *all* the players will agree with it). For the other games, reasons behind bots are different, but they do exist pretty much regardless; hey, bots are known to represent a Big Problem even for poker sites!

"Abuse scenarios using bots are endless.

Abuse scenarios using bots are endless. Just as one example, if there are goodies associated with new accounts, bots may automatically register, play enough to get those goodies, and then to pass them along to a consolidation account; then consolidation account can be used, say, to sell the stuff on eBay. Been there, seen that.

Bot fighting itself is a Big Task and if your game is really successful, you're likely to end up with a whole department dealing with bots; this is especially true as bot fighting is inherently in "Security-by-Obscurity" domain, so it is always pretty much going back and forth between you and the bot writers. Impact of bots on the game depends on them being published or not. For the unpublished bots, the impact is usually fairly low; for the published/commercially available bots, the impact is significantly higher. Fortunately, when you're fighting bots which are already published, it is you who has a "home turf" advantage half of the time (as you can get/buy the bot and dissect

it's logic pretty much in the same way as the bot writers have done with your program when they wrote the bot).

Attack Type Summary

Let's summarize the attacks mentioned above in one table:[5]

Attack	Impact	"Home turf" Advantage	Chapter where ProtectionWill Be Discussed
Stealing User DB	Very High	Yours	[[TODO]]
DDoS	Medium	None	[[TODO]]
Affecting Gameplay	High to Extremely High	Depends on Architecture	This one
Duplicate Accounts	Very Low	Cheaters'	[[TODO]]
Another User's Device	Low	None	[[TODO]] (only protecting logins will be discussed)
Non-published Bots	Low	Cheaters'	[[TODO]]
Published Bots	High	Back and Forth	[[TODO]]

As we can see from this table, only one of the attack types heavily depends on the architecture. Moreover, as it has one of the highest possible impacts on the game, we'd better to take a look at it before making any decisions related to the game architecture. Let's take a look at three different approaches to MMOG from this perspective.

[5] As usual, only typical values are provided, and your mileage may vary

Peer-to-Peer: Pretty Much Hopeless

SPOF
A single point of failure (SPOF) is a part of a system that, if it fails, will stop the entire system from working
— Wikipedia —

From time to time, a question arises in various forums: why bothering about servers, when we can have a SPOF-free, perfectly scalable system using P2P (as in "peer-to-peer")? Moreover, there exists an article [Skibinsky] which argues (I presume, with a straight face) that the client-server is not scalable, and that the future lies with MMO games being P2P.

With P2P, each client performs its own calculations, which are then used to determine the state of the game world. The way how the state of the game world is determined based on the results of individual computations, needs to be described in detail when you define your P2P game; ironically, I didn't really see any specific architectures specifying "how it should work", not even in [Skibinsky].

Still, let's take a look at P2P for gaming purposes, first of all from the point of view of "Affecting Gameplay" type of attacks. With P2P, we're essentially operating on "attackers home turf", making us to resort to "Security by Obscurity". [Skibinsky] recognizes it (albeit using different terminology), and proposes essentially three techniques to address this security issue.

"As soon as end-user himself wants to break code signing – it becomes at best "Security by Obscurity"

The first technique proposed in [Skibinsky] is code signing. However, the problem with the code signing of the game (as with any other code signing), is that *as soon as end-user himself wants to break code signing – it becomes at best "Security by Obscurity".* This is a direct result of the fact that when we're operating on attacker's "home turf", then all the signing keys (both private keys and root certificates) are under control of the attacker, making them essentially useless. BTW, [Skibinsky] also recognizes this weakness, stating "That still doesn't provide 100% security".

The second technique proposed in [Skibinsky], is a kind of "web of trust" system, with some of the nodes being trustworthy, and some being untrustworthy. While the idea looks attractive at the very first glance, there are two problems when applying it to the MMOG. Without (a) a reliable way to identify nodes, and with (b) not-really-tied-to-real-world-gaming-logins I expect any kind of "web of trust" thing to fall apart very quickly when facing a determined adversary; this is not to mention that those trusted nodes will quickly become a target for "Another User's Device" attack, which we cannot really do much about beyond protecting user's login.

The third technique to address inherent vulnerability of P2P systems to "Affecting Gameplay" attacks, is about cross-checking of the calculations-made-by-our-potential-cheater by other peers. While the idea sounds nice, on this way there are several Big Problems too.

First of all, the cross-checks are themselves vulnerable to cheating. Note that even Bitcoin system (which solves only a singular problem which is extremely narrow compared to general gaming) has an inherent 50% attack, and with inevitably selective nature of the cross-checks for gaming worlds (we simply cannot perform *all* the calculations on *all* the nodes), the number of nodes necessary to "take over the gaming world" is going to be drastically lower.

Second, all these cross-checks inevitably lead either to additional delays (which is unacceptable for the vast majority of the games), or to cross-checks being performed not in real time, but "a bit later". The latter approach raises another Big Question: "What shall we do with the game world when the cheater is caught?" Sure, we can ban the cheater for life (or more precisely, "until he opens a new e-mail account and registers again"), but what should we do with consequences of his cheating actions? This question, to the best of my knowledge, has no good general answer: leaving cheater deeds within the world is at best unfair to the others (not to mention that a cheater may cheat in the interests of another player), and rolling the whole world back whenever the cheater is found, is usually impractical ☹ (not to mention frustration of all the players not affected by cheating, but needing to lose significant time of their play).

As a result,

as of now, I don't see how peer-to-peer game (which goes beyond closed communities where people can trust each other), can provide reasonable protection from the cheaters

And we didn't even start to mention issues related to P2P complexity, which tends to go far beyond any complications related to client-server systems (especially if we're considering "web of trust" and "cross-checks").

That being said, I don't mean that P2P MMOG is unfeasible in principle; what I mean is that as of now, there is no good way to implement gaming over P2P (and whether such a way will ever be found, is an open question).

Non-Authoritative Client-Server: Simpler but still Hopeless

The whole idea of "non-authoritative server" has arisen when developers tried to convert classical single-player 3D game into an MMO. And classical single-player 3D game is very often pretty much built around 3D engine (which usually also performs some physics-related calculations).

To convert the classical-built-around-3D-engine game to an MMO, the simplest way was to just send the data about player movements to the other players, which then reflect these movements in their 3D engines. To send the data between the players, a "non-authoritative server" was routinely used, which is usually nothing more than taking the data from the clients and forwarding it pretty much "as is" to the other clients interested in what's going on in this part of the world.

"While simple to implement, essentially this "non-authoritative server" model is as much vulnerable as P2P model.

While simple to implement (and also avoiding NAT problems which tend to be quite unpleasant with P2P), essentially this "non-authoritative server" model is as much vulnerable as P2P model. With a non-authoritative server, you need to rely on clients not cheating; for example, if your client controls all the movements of your character, then it can say "hey, my coordinates just changed to the ones being 5 meters back" to avoid being hit, and server won't prevent your client from doing it. And if you try to detect this kind of cheating on the other clients, you will inevitably (pretty much the same way as described in section on P2P) get into need to a kind of "vote" on "who's good and who's bad", with different versions of the worlds appearing on different clients and needing to be consolidated, with a question "what to do if a cheater is detected", and so on.

Therefore, from a cheating prevention standpoint, non-authoritative servers are pretty much the same as P2P systems; the difference is that non-authoritative servers are simpler to implement, but at the cost of paying significantly more for server traffic (and arguably worse scalability).

However, these differences are pretty much irrelevant from preventing-cheating perspective, and

because of the cheating issues, I don't recommend using non-authoritative servers

While in theory, there might be games which can be protected using non-authoritative servers (as in "I don't have a formal proof that such games don't exist"), think more than twice when choosing to use non-authoritative servers. Oh, and make sure to re-read the "If You're Popular Enough, They Will Find Reasons to Cheat" section above.

Authoritative Server: As Safe As They Go

"With authoritative server, it is the server which moves the players (and other stuff) around; it is also the server who makes all the decisions about collisions, hits, etc.

The most popular approach when it comes to MMOG, is a so-called "authoritative server". In the usual approach to authoritative servers for a virtual world game, clients usually have a 3D engine, but this 3D engine is used purely for rendering, and not for decision-making. On the other hand, all the movements (not "coordinates resulting from movements", but more or less "player keypresses and mouseclicks themselves") are sent to the server, and it is the server which moves the players (and other stuff) around; it is also the server who makes all the decisions about collisions, hits, etc. Moreover, with an authoritative server, it is the server who makes all the changes in it's own copy of the game world (and the server's copy is an authoritative copy of the game world, which is then replicated to the clients to be rendered).

It means that for Virtual World games with an authoritative server, it is the server (and not the clients) who needs to implement the physics engine (though 3D rendering engines are still on the clients).

However, for fast-paced games, the delays of going-to-server-and-back-to-client with every keystroke are often not acceptable. In such cases, client often implements some kind of extrapolation (usually referred to as "client-side prediction") based on it's own picture of the game world; in other words, client shows the game world *assuming* that it's own picture of the game world is the same as the server one. For example, it may even show hits based on its own understanding of the game world. On the other hand, the client copy is *not* authoritative, so if the vision of the server and the vision of the client become different, it is server's copy which always "wins" (i.e. in such cases it is perfectly possible to see the hit which should have killed the opponent, only to see that the opponent is well alive). More details on client-side prediction for fast-paced games based on an authoritative servers will be provided in Chapter [[TODO]].

RTT is the length of time it takes for a signal to be sent plus the length of time it takes for an acknowledgement

From the point of view of cheaters trying to affect gameplay, authoritative servers are by far the best thing you can have. If you have enough checks on the server side, you always can enforce game rules with a relative ease. And while when using client-side prediction, temporary disagreements between clients and server are possible, it is always clear how to resolve the conflict; also these disagreements are always of a very-limited time (of the order of magnitude of RTT), which makes them not that noticeable in practice (except for first-

of that signal to be received.
— Wikipedia —

person shooters and fast-paced combat in RPGs).

It is worth noting that merely using authoritative server doesn't necessarily imply security; it merely *provides means* to make your game secure, and you will need to work on actual security later; fortunately, usually most of special work in this regard can be pushed down the road, after your game becomes more-or-less popular.

Authoritative Servers: Scalability is Imperfect, But is Workable

There is only one objection against this theory, and it is that the theory is wrong.
— C.N. Parkinson —

Before committing to go with authoritative servers, let's consider one common argument pushed by proponents of using-P2P-for-gaming; this is that client-server systems are not scalable. In particular, such an argument is presented in [Skibinsky].

The first line of the argument of alleged non-scalability of client-server games, revolves around the "$O(W^2)$ traffic estimate". The idea behind the argument goes as follows: first, let's consider a game world with W players within; now let's consider each player making some kind of change every N seconds, and let's assume that this change needs to be communicated to all the W-1 of the other players. Hence (they argue), for W players in the world, we need to push $O(W^2)$ bytes of traffic per second, making client-server non-scalable.

"In practice this $O(W^2)$ estimate doesn't really stand

If $O(W^2)$ would be indeed the case, then we'd indeed have quite significant scalability problems. However, in practice this $O(W^2)$ estimate doesn't really stand; let's take a closer look at it.

First, let's note that in real-world the number of people we're directly interacting with, has no relation to the number of people in the world. In virtual worlds it is normally *exactly the same thing* – the number of people (or other entities) players are interacting with, is limited not by the world population, but by our immediate vicinity, which in most cases has nothing to do with the world size. This is the point where $T=O(W^2)$ estimate falls apart (assuming reasonable implementation), and is replaced with $T=O(W)*C$ (when $W->infinity$), where C is the constant representing "Immediate Vicinity"[6] From this point, the estimate is no longer $T=O(W^2)$, but just $T=O(W)$ (with mathematicians among us sighing in relief).

Second, if $T=O(W^2)$ would be the case, it would mean that limits on the bandwidth of individual users would be hit pretty soon, so that even if somebody designs a world with everybody-to-everybody direct interaction all the time, it still won't run regardless of architecture (i.e. it won't run in client-server, but it won't run in P2P either).[7]

These observations are also supported by practical experiences; while the dependency of traffic from the world size is usually a bit worse than simple $T=O(W)$, it is never as bad as $T=O(W^2)$. So, we can make an observation that

In a properly implemented Client-Server game, for large enough world population W, traffic T is much closer to O(W) than to $O(W^2)$

This observation has one very important consequence: as soon as T is close to O(W), it means that your traffic is roughly proportional to world population W, which means that your expenses E is also proportional to W. On the other hand, within certain non-so-implausible assumptions, your income I is also more or less proportional to W. If this stands, it means that both your income I and your expenses E grow more or less proportional to W; this in turn means that if you were making money with 10'000 players, you will still make money (and even more of it) with 1'000'000 players.

---

[6] in [Skibinsky] this effect is referred to as *immediate action/reaction manifold,* and it is relied on to ensure P2P scalability, though for some reason it is mentioned only in the P2P context

[7] This effect is mentioned in [Skibinsky] too, though strangely enough, the way to mitigate this problem is once again mentioned only in the P2P context

A Very Example Calculation

To bring all the big-O notation above a bit more down to earth and to demonstrate these effects from a more practical perspective, let's consider the following example.

Let's consider a game where you can interact directly only with at most C=1000 other players, regardless of the world size and regardless of world population W. Of course, architecting and implementing your game to make sure that you don't send your updates to those-players-who-don't-really-need-them will be a challenge, but doing it is perfectly feasible[8]

Let's take the traffic estimate per player-interacting-with-another-player, from

[Skibinsky], i.e. as 50/3 bytes/sec (in practice, your mileage will vary, but if you're doing things right, usually it won't be off by more than an order of magnitude, so we can take it as a rather reasonable estimate). Let's also assume that all your players are paying you $10/month as a subscription fee. And let's further assume that your servers are residing in the datacenter (not in your office, see Chapter [[TODO]] why), and that you're paying $1000 per Gbit/s per month in your datacenter (once again, YMMV, but again, this number is not that far off – that is, if you're paying per GBit/s and have spent your time on finding a reasonably good deal; there is no doubt you can get it for a *lot* more money than that).

"At the same time, with your subscription fees you'll be making around $100'000/month, which means that your traffic costs are negligible.

Therefore, when you have 10'000 simultaneous players, you'll have traffic of *at most* 50/3*10000*1000 bytes/sec ~= 1.6e8 bytes/sec ~= 1.3 GBit/s; this will cost you around $1300/month. At the same time, with your subscription fees you'll be making around $100'000/month, which means that your traffic costs are negligible.[9]

When you grow to 1'000'000 simultaneous players, then your traffic per user will increase. As noted above, T won't grow as $T \sim W^2$, but there will be modest increase in per-user traffic because while each part of traffic T' (with sum of all T's being T) can be in most cases optimized to plain $T' \sim W$, in practice usually you're too lazy (or have too little time) to optimize all of them. For the purposes of our example let's assume that the traffic has grown 5-fold (you should be rather lazy – or busy – to get to 5x per-user traffic increase, but well, it can happen). Then, when you grow to 1'000'000 simultaneous players, your traffic will grow 500-fold, bringing it to 650 GBit/s, costing you $650000/month (in practice, the price will go lower for this kind of traffic, but let's ignore it for the moment). While this may sound as tons of money, you should note that with your $10/month subscription fee and million of users you'll be making $10M/month, which is still much more than enough to cover traffic bills (and note that if it becomes a problem, you still have that about-5x-times overhead, most of which can be recovered given sufficient development time).

---

[8] this has been demonstrated by numerous games-which-are-making-money out there, all of which, to the best of my knowledge, are client-server.

[9] Don't rush to buy that house on Bahamas though – while traffic costs are indeed negligible, other costs, especially advertisement costs to keep new players coming, are not

Summary: Authoritative Server is not ideal, but is The

Only Thing Workable

Let's summarize our findings about three different approaches in the following table:

	Scalability	Resilience to Cheating	Complexity
P2P	Good	Poor	from High (when not dealing with cheating) to Very High (Otherwise)
Non-Authoritative Server	Ok	Poor	from Low (when migrating from classical 3D game *and* not dealing with cheating), to High (if dealing with cheating)
Authoritative Server	Ok	Good	Medium

Actually, this table is not that different from that of in [Skibinsky]; the main difference between this book and [Skibinsky] is about estimating the impact of the differences between the approaches, in the real world. In particular, I'm sure that 'Poor' resilience to cheating is bad enough to rule out the relevant models, especially when there is an "Authoritative Server" model which has 'Ok' scalability (which is explained above) and 'Good' resilience to cheating. This point of view seems to be supported by MMOG developers around the world: as far as I know, as of now there are no real MMOGs which are implemented as P2Ps, there are quite a few of those based on non-authoritative servers, but the players are complaining about it, and there are lots of MMOGs based on authoritative servers.

Bottom Line: Yes, It is Going to Be Client-Server

TL;DR for Chapter III:

- Cheating is One Big Problem for MMOGs
- Players will cheat even if you're sure they have a zero reason to
- Gameplay cheating is one of the Big Potential Problems for your game

- P2P and non-authoritative servers provide very poor protection against Gameplay Cheating
- Despite some claims to the contrary, Client-Server (in particular, authoritative servers) can be made scalable
- Given the balance of pros and cons, Authoritative Servers look as the best option as of now; some (including myself) will even argue that in most cases it is *the only viable option*. While exceptions are theoretically possible, they are quite unlikely.

BTW, when speaking about Client-Server, I'm not ruling out multiple datacenters on the server side (this is referred to as "Grid Computing" in [Skibinsky]); on the other hand, delegating any kind of authority and decision-making to the client looks way too risky for practical MMO.

[[To Be Continued…

This concludes beta Chapter III from the upcoming book "Development and Deployment of Massively Multiplayer Games (from social games to MMOFPS, with social games in between)". Stay tuned for beta Chapter IV, "DIY vs Re-use"

EDIT: beta Chapter IV. DIY vs Re-Use: In Search of Balance, has been published.

]]

[–] References

[Skibinsky] Max Skibinsky, "The Quest for Holy Scale", in "Massively Multiplayer Game Development 2", pp. 339-373.
[Skorobogatov] Sergey Skorobogatov, "Hardware Security of Semiconductor Chips: Progress and Lessons"

Acknowledgement

Cartoons by Sergey Gordeev from Gordeev Animation Graphics, Prague.

« *Chapter II: "Game Entities and Interactions" from upcoming b…*

Chapter IV. DIY vs Re-Use: In Search of Balance from upcomi… »

Filed Under: Distributed Systems, Network Programming, Programming, System Architecture
Tagged With: authoritative server, client-server, game, multi-player