Machbarkeit eines „perfekten“ Heal-Addons (Mists of Pandaria Classic)

Hintergrund und Ziele

Ein perfektes Heal-Addon soll in Echtzeit die aktuelle Lebenspunkte (HP) eines Zieles mit prognostizierten Änderungen durch eingehende Heilungen und erwarteten Schaden kombinieren. Das Ziel ist, dem Heiler bereits vor dem Wirken eines Zaubers anzuzeigen, wie hoch die HP des Ziels voraussichtlich sein werden, ohne dass Lebensbalken „sprunghaft auf 100%“ steigen bevor die Heilung tatsächlich eintrifft. Dieses Konzept, oft Heilvorhersage genannt, erhöht die Konsistenz der HP-Anzeige und hilft Overheal zu vermeiden.

Wichtige Anforderungen sind dabei:

Live-Projektion: Aktueller Server-HP-Wert plus Precast-Projektion (eigene + fremde laufende Heilungen minus erwarteter Schaden bis Landezeit des Zaubers).

Separate Anzeigeebene: Die Prognose soll als Overlay (z.B. andersfarbiger Segment im HP-Balken) erscheinen, nicht die echten HP überschreiben, um Verwirrung zu vermeiden.

Raid-Tauglichkeit: Das Addon muss in 40-Spieler-Raids funktionieren, auch wenn nicht jeder Heiler eine spezielle Heal-Library installiert hat. (In klassischen WoW-Umgebungen ist z.B. Deadly Boss Mods für viele Pflicht, aber ein Heilungs-Addon oder -Lib nicht.)

Blizzard-API vs. LibHealComm – Datenquellen für eingehende Heilung

Für die Erfassung eingehender Heilungen stehen zwei Hauptansätze zur Verfügung:

Blizzard-API: Die Funktion UnitGetIncomingHeals(unit) ist die offizielle Schnittstelle, um vorhergesagte Heilungen auf einem Ziel abzufragen. Allerdings ist sie in Classic-Clients nur teilweise funktionsfähig: Sie berücksichtigt nur gewirkte Direktheilungen, \*\*Heil-über-Zeit-Effekte (HoTs) und kanalisierten Heilzauber werden nicht einbezogen】

wowpedia.fandom.com

. Blizzard hat dieses Feature in WoW Classic zwar zurückportiert, aber ausdrücklich ohne HoT-Support implementiert

wowpedia.fandom.com

. Mit anderen Worten: In MoP Classic wird UnitGetIncomingHeals Heilung durch z.B. Erneuerung, Wildwuchs oder Tranquility ignorieren, was zu unvollständigen Vorhersagen führt. Community-Rückmeldungen und Bugreports bestätigen, dass die API für Classic „nur für direkte Heals“ funktioniert und z.B. bei kanalisierten Heilungen einfach 0 zurückliefert

github.com

. Entsprechend wird die Genauigkeit der API-Infos als mangelhaft und „buggy“ eingestuft – Addon-Autoren melden, dass UnitGetIncomingHeals() teils unsinnige Werte liefert

wowinterface.com

.

LibHealComm-4.0: Dies ist eine etablierte Addon-Bibliothek, die seit Vanilla/WotLK-Zeiten genutzt wird, um Heilungsinformationen zwischen Spielern auszutauschen. LibHealComm (LHC) registriert, wenn ein Heiler einen Zauber wirkt, und schätzt den zu erwartenden Heal-Wert unter Einbezug von Talenten, Glyphen, Auren und Stats des Wirkenden. Diese Daten werden über den Addon-Kanal an alle Gruppenmitglieder mit LHC gesendet

wowace.com

. Dadurch können Addons sehr präzise anzeigen, wie viel Heilung von wem auf ein Ziel im Anflug ist – inklusive Effekten wie z.B. Beacon of Light, kritischen Heils, HoTs (soweit in der Lib-Datenbank enthalten) usw. LibHealComm gilt als zuverlässig für Direktheilungen, solange alle Heiler im Raid die Library nutzen, da nur dann sämtliche Casts erfasst und verteilt werden

reddit.com

. Viele beliebte Raidframes und Heil-Addons (Grid, VuhDo, HealBot, ElvUI usw.) haben LHC integriert oder als optionalen Bestandteil vorgesehen

reddit.com

reddit.com

. Das bedeutet in der Praxis: In organisierten Raids hat oft ein Großteil der Heiler LHC bereits „mit an Bord“, entweder bewusst oder über ihr Unit-Frame-Addon. LibHealComm rechnet mit den meisten Heileffekten korrekt (z.B. Talente, Mastery-Boni, Glyphen werden berücksichtigt) und besitzt eine interne Spell-Datenbank

wowace.com

wowace.com

. Allerdings muss diese Datenbank für neue Expansions (wie MoP) gepflegt werden – das heißt, Spell-IDs und Formeln für MoP-Heilzauber müssen eingepflegt sein, damit LHC korrekte Werte sendet. Addon-Changelogs weisen explizit darauf hin, dass zum Start einer Expansion die Heilformeln aktualisiert werden müssen (in Patchnotes oft als „Need new spell formulas and coefficients“ vermerkt)

curseforge.com

. Wird LHC ohne Update in MoP Classic verwendet, könnten neue Heilzauber zunächst nicht erfasst werden, bis ein Update erfolgt.

Fazit der Datenquellen: LibHealComm bietet – sofern im Raid verbreitet – die weitaus vollständigere und akkuratere Datengrundlage für ein Heal-Overlay als die Blizzard-API alleine. Die API liefert in MoP Classic nur einen Teilaspekt (Direktheilungen) und ist daher nicht als alleinige Quelle für präzise Precast-Projektionen geeignet

wowpedia.fandom.com

github.com

. Im Idealfall nutzt man also LHC, greift aber zur API zurück, wenn von bestimmten Spielern keine LHC-Daten vorliegen. Andernfalls würden eingehende Heals dieser Spieler komplett unsichtbar bleiben.

Genauigkeit und Raid-Szenario (40 Spieler)

Ein Hauptaugenmerk liegt auf der Nutzbarkeit in 40er-Raids, wo nicht garantiert ist, dass alle Heiler ein Addon oder die LibHealComm installiert haben. Hier stellt sich die Frage der Machbarkeit in Bezug auf Datenvollständigkeit und Performance:

Vollständigkeit der Heal-Daten: Ohne einheitliche Library bleibt ein Teil der Heilleistungen verborgen. In WoW Classic konnte man standardmäßig keinerlei fremde eingehende Heilung sehen – nur wer HealComm nutzt, kann gegenseitig Casts austauschen

reddit.com

. Dieses Problem würde in MoP Classic weiterhin bestehen: Heiler ohne LHC senden nichts; die Blizzard-API fängt solche Casts nicht zuverlässig ab (außer eben die genannten Direktheals, aber ohne Addon-Comm fehlen z.B. Cast-Start-Events anderer Spieler). Die Erfahrung der Addon-Community zeigt, dass manche UI-Pakete (z.B. ElvUI) LHC von Haus aus mitbringen

reddit.com

, wodurch zumindest in vielen Gruppen einige Heiler die Lib verwenden. Aber es wird kaum vorkommen, dass wirklich alle 40 Raidmitglieder sie haben.

Lösungsansatz: Ein Hybrid-Aggregator im Addon. Konkret: Verwende LibHealComm, wenn verfügbar, ansonsten UnitGetIncomingHeals, ansonsten 0. Mit dieser Priorität kann das Addon pro Einheit entscheiden: gibt es verlässliche LHC-Daten (von mindestens einem Heiler)? Wenn nein, greife auf die Blizzard-API zurück, um wenigstens eigene und ggf. direkt gewirkte Gruppenheilungen zu erfassen. Ist auch das nicht verfügbar, wird für diese Quelle keine Healvorhersage einbezogen. So erhält der Nutzer im Worst Case immerhin die Projektion seiner eigenen Heilung plus aller erkannten Direktheals von anderen (aber HoTs fremder Heiler blieben unberücksichtigt). Da reine API-Werte tendenziell unvollständig sind, kann man solche Prognosen ggf. visuell markieren (z.B. durch geringere Deckkraft des Overlaysegments), um anzuzeigen, dass hier Unsicherheit besteht – ganz nach dem Motto: „Incoming heals (teilweise unbekannt)“. Einige neuere Projekte gehen noch weiter: Ein Grid2-Plugin etwa schätzt fehlende Heildaten anderer Heiler ab, indem es deren bereits gewirkten Heilungen beobachtet und daraus die Heil-Power ableitet

curseforge.com

. Nach ein paar Heilzaubern kann das Addon so Prognosen treffen, selbst wenn der andere Heiler kein Lib-Addon hat – beispielsweise wird registriert, dass Heiler X mit Heilungspell Y typischerweise ~5000 Punkte heilt, und bei erneutem Cast von Y auf das Ziel nimmt man diesen Wert als incoming an. Solche Methoden (rollierender Mittelwert fremder Heals) sind komplexer, könnten aber in Betracht gezogen werden, um die Vorhersage weiter zu verbessern. Sie sind insbesondere dann nützlich, wenn man auf HoTs fremder Heiler schließen will (was die Blizzard-API ja auslässt). Der Aufwand dafür ist hoch, daher wäre das eher eine spätere Ausbaustufe – aber machbar ist es, wie das erwähnte Plugin beweist

curseforge.com

.

Performance in 40er-Raids: Die Fülle an Events (Heilungen, Schadensereignisse, HoT-Ticks) in einem großen Raid bedeutet, dass das Addon effizient arbeiten muss. Um die Machbarkeit zu gewährleisten, sollten Berechnungen möglichst ereignisgetrieben und throttled ablaufen. Praktisch heißt das: Updates in ~0,1–0,2s Abständen statt bei jedem Combat-Log-Ereignis. Das Design sieht vor, die Forecast-Berechnung (z.B. DPS-Mittelwert, HoT-Summen) höchstens 5–10 Mal pro Sekunde zu aktualisieren, was erfahrungsgemäß vertretbar ist. Gezielte Sofort-Updates passieren nur bei bestimmten Triggern: etwa wenn ein eigener Cast startet oder stoppt (um die Projektion zu setzen oder zu löschen) oder wenn eine Heilung tatsächlich landet, sodass der „echte“ HP-Wert sich ändert. Dieses Event-getriggerte Vorgehen mit leichter Drosselung wurde von ähnlichen Addons erfolgreich eingesetzt und verhindert auch, dass z.B. das Durchlaufen aller 40 Einheiten bei jedem Mini-Tick die Framerate beeinträchtigt

wowinterface.com

wowinterface.com

.

Latenz- und Timing-Aspekte: In einem internationalen Raid können die Latenzen variieren. Das Addon sollte daher die Netzwerkverzögerung einbeziehen – z.B. über GetNetStats() lässt sich die eigene Latenz ermitteln. Auch die Spell Queue Window (ein klientenseitiges Zeitfenster, typischerweise 250–400ms, in dem man Folgezauber vor Ende des aktuellen Casts in die „Warteschlange“ legen kann) sollte berücksichtigt werden. Diese Faktoren fließen in die Berechnung von T\_land (Zeitpunkt, zu dem ein Cast wirksam wird) ein. Praktisch: T\_land = Castzeit + globaler Cooldown + Queue-Fenster + Latenz. Alle Vorhersagen (eingehende Heals, erwarteter Schaden, HoT-Ticks) werden nur bis zu diesem Zeitpunkt summiert. Das verhindert, dass man Heilungen mitzählt, die eigentlich erst danach landen würden, oder Schaden unterschlägt, der kurz vor Heal-Einschlag noch passiert.

Zusammengefasst ist die Machbarkeit im Raid gegeben, wenn auch mit sorgfältiger Optimierung. Andere Addons wie VuhDo oder HealBot zeigen bereits eingehende Heals und Absorbs für 40 Leute, was von der Last her vergleichbar ist. Die zusätzliche Schadensprognose (via DPS-Mittelwert) erhöht den Rechenaufwand moderat, dürfte aber bei vernünftiger Implementierung (Exponentieller Gleitmittelwert über z.B. 5 Sekunden Schaden pro Unit) ebenfalls tragbar sein. Wichtig ist, potenziell teure Schritte wie das Scannen aller Auren für HoT-Ticks nur bei Bedarf durchzuführen – z.B. nur für Einheiten, auf denen überhaupt ein relevanter HoT aktiv ist, statt pauschal für alle. Mit solchen Checks bleibt das Addon performant.

Umsetzung: Kernfunktionen und Empfehlungen

Um das Vorhaben zu realisieren, sind mehrere Funktionsblöcke nötig (die bereits konzeptuell definiert wurden). Hier eine Übersicht, wie sie umgesetzt werden können und wo mögliche Stolpersteine liegen:

Server-HP-Kern: Die aktuellen Lebenspunkte und Max-HP jeder Einheit zieht man zuverlässig über UnitHealth(unit) und UnitHealthMax(unit) aus dem WoW-API. Zusätzlich werden UnitIsDeadOrGhost (für Totenerkennung) und UnitGetTotalAbsorbs (aktuelle Absorptionsschilde) genutzt, um einen echten Effektiv-HP-Wert zu ermitteln. Diese Werte sind unmittelbar und autoritativ vom Server. (Kein Problem, diese Funktionen sind performant und exakt.)

Precast-Engine (Zeitberechnung): Für jeden Heilzauber, den der Spieler wirken will, wird die Landezeit T\_land berechnet. Daten: Zauberzeit des Spells (inkl. Haste-Effekte), globaler Cooldown (idR 1,5s, modifiziert durch Haste), eingestelltes SpellQueueWindow (z.B. 0,25s) und die aktuelle Latenz. Diese Summanden sind alle aus dem API greifbar (GetSpellCooldown für GCD-Reste, GetNetStats für Latenz, Spellcasting-Infos via Events). (Kein externer Datenbedarf, reine lokale Berechnung.)

Incoming-Heal-Aggregator: Wie oben diskutiert, werden eingehende Heilungen bis T\_land gesammelt. Eigene ausgehende Heals kennt das Addon (Caststart auf Ziel etc.). Fremde Heals bis T\_land kommen entweder aus UnitGetIncomingHeals(unit) – mit den genannten Einschränkungen – und/oder über LibHealComm-4.0. Im Optimalfall hört das Addon auf LHC-Events (z.B. „HealComm\_HealStarted“ etc.), um genau zu wissen: Heiler X castet großen Heal für 2,5s auf Ziel Y, Wert ~8000. Dann kann man diesen Beitrag zu Y’s HP-Prognose addieren. Hat ein Heiler kein LHC, und man will über die API hinausgehen, könnte man wie erwähnt dessen letzte Heilung aus dem Kampflog auswerten, um den kommenden Wert abzuschätzen

curseforge.com

. (Empfehlung: Zunächst LHC-Unterstützung einbauen. API als Fallback, obwohl unvollständig – es schadet nicht, wenigstens Direktheals zu addieren. Später optional verbesserte Schätzung für Non-LHC-Heiler ergänzen.)

Damage-Forecast: Hier geht es um die Prognose des Schadens, der das Ziel bis T\_land voraussichtlich erleidet. Vollständig vorhersagen lässt sich zukünftiger Schaden nicht, aber man kann Trends nutzen. Ein pragmatischer Ansatz ist, die in den letzten X Sekunden erhaltene DPS des Ziels zu ermitteln (z.B. mittels exponentiell gleitendem Mittelwert der letzten ~5–10 Sekunden an Schaden aus COMBAT\_LOG\_EVENT\_UNFILTERED). Multipliziert man diesen durchschnittlichen DPS-Wert mit der Zeitspanne T\_land, erhält man einen groben erwarteten Schaden D\_pred bis zum Heal-Einschlag

wowinterface.com

. Natürlich wird dieser Wert begrenzt (Clamp), um unrealistische Ausreißer zu vermeiden. Beispielsweise könnte man annehmen, dass max. der letzte Trefferwert einmalig erneut eintritt, falls der EMA sehr hoch ist. Dieses Feature hat moderaten Nutzen – es ist vor allem in Situationen hilfreich, wo konstante DoT-Effekte ticken oder der Boss AoE-Schaden pulsiert. Es ist aber kein Muss für Version 1.0. Viele Heal-Addons kommen ohne explizite Schadensvorhersage aus, da Heiler oft selbst einschätzen, ob ein Ziel weiter Schaden nehmen wird. (Empfehlung: Evtl. als späteres Upgrade „Damage-Forecast“ integrieren. Anfangs reicht es, die vorhandenen aktuellen HP zu nehmen; Overheal-Erkennung funktioniert auch so schon gut, da eingehende Heals + aktuelle HP im Verhältnis zur Max-HP betrachtet werden können.)

Tick-Scheduler (HoT/DoT): Um genauer zu sein, kann das Addon kommende HoT-Ticks bis T\_land berücksichtigen. Dafür scannt man die Auren des Ziels nach eigenen HoTs (und evtl. fremden, falls auffindbar) – WoW liefert via Aura-APIs die Tick-Intervalle und nächste Tickzeiten mancher Effekte. Beispiel: Erneuerung tickt alle 3 Sekunden. Wenn T\_land 2 Sekunden in der Zukunft liegt, würde kein voller Tick mehr kommen, wohl aber ein Teil vielleicht gerade nach T\_land. Solche Details sind tricky; realistisch würde man nur ganze Tick-Ereignisse bis zur Landezeit addieren. Gleiches gilt für DoTs: falls das Ziel einen DoT hat, der innerhalb der nächsten 2s tickt, könnte man diesen Schaden in D\_pred einfließen lassen. (Dieser Block ist technisch machbar, aber aufwändig. Er erhöht die Präzision leicht, aber nicht dramatisch. Empfehlung: Kann implementiert werden, aber mit niedrigerer Priorität. Wichtiger ist, dass direkte Schadenstrends und Heals stimmen.)

Absorb-Handling: Bereits aktive Schilde (Absorbs) werden über UnitGetTotalAbsorbs erfasst

wowpedia.fandom.com

. Für die Effektiv-HP-Berechnung (EHP) ist das relevant: Ein Ziel mit 5000 HP und 2000 Absorb hat effektiv 7000, bevor es Schaden nimmt. Bei der Overheal-Berechnung (wie viel der geplanten Heilung würde über MaxHP hinausgehen) müssen Absorbs mitbedacht werden – allerdings heilen Zauber i.d.R. keine Absorbs auf, sondern nur verlorene HP. Daher gilt: Overheal entsteht, wenn HP\_now + Absorb\_now + alle Heals - erwarteter Schaden > MaxHP. Überschreitet die Projektion MaxHP, wird eben nur bis 100% angezeigt. (Absorb-Anzeige ist heute Standard in Frames; die Einbindung ist problemlos möglich und sollte erfolgen, um z.B. Power Word: Shield etc. sichtbar zu machen.)

Predictive Solver (pro Einheit, pro Spell): Dies ist das Herzstück: Für ein anvisiertes Ziel und einen gewählten Heilzauber berechnet das Addon: H\_proj\_on\_land = clamp(HP\_now - D\_pred + IncHeals\_until\_T + HealValue(spell), 0, MaxHP). Damit erhält man den projizierten HP-Stand bei Landezeit (gekappt zwischen 0 und MaxHP). Diese Projektion wird nur als eigener Layer geführt, der Server-HP wird nie direkt überschrieben. So kann man den Overlay-Balken z.B. grün/blau einfärben und an den aktuellen HP-Balken „anhängen“. Springt der Wert nicht über 100%, hat man eine fortlaufende Anzeige; würde er über MaxHP gehen, sieht man, dass der Overlay-Balken das Ende des HP-Balkens erreicht – ein Zeichen für komplettes Auffüllen (und potenzielles Overheal). Wichtig: Selbst wenn die Projektion 100% erreicht, soll der eigentliche HP-Balken nicht abrupt voll werden, sondern erst, wenn der Heal wirklich eintrifft. Diese Trennung verhindert visuelles Flackern und „Geister-Heilungen“. Blizzard’s Default-UI macht es genauso: Dort wird eingehende Heilung als separater Farbabschnitt dargestellt, der nie über den Rahmen des Gesundheitsbalkens hinausgeht

reddit.com

.

HealValue(spell) – Heilwert-Ermittlung: Um die erwartete Heilmenge eines Zaubers zu kennen, gibt es mehrere Ansätze. LibHealComm liefert diese Werte für alle Casts anderer Spieler (da dort schon eingerechnet). Für den eigenen Cast kann das Addon die Werte aus einer Spell-Datenbank nehmen (falls vorhanden) oder dynamisch lernen. Der Entwickler schlägt eine Kombination vor: ein rollierender Mittelwert aus den letzten eigenen Heilungen pro Zauber, der persönliche Stats (Krit, Mastery, Versatility in MoP) berücksichtigt. So „lernt“ das Addon, dass z.B. Große Heilung bei Deinem Gear im Schnitt 8500 heilt, mit einem Streubereich je nach Krit. Anhand der Varianz könnte man ein Konfidenzintervall angeben (z.B. als transparenten Bereich: minimaler und maximaler kürzlich beobachteter Wert). Als Fallback dient eine statische Spell-DB (entweder selbst gepflegt oder aus LHC entlehnt), damit das Addon von Anfang an Werte hat. Bei fremden Heilern ohne LHC, wie oben erwähnt, kann man deren HealValue ebenfalls über Beobachtung der SPELL\_HEAL-Ereignisse ermitteln. Dieser Aufwand lohnt, wenn man wirklich präzise sein will, ist aber optional. (Empfehlung: Eigene Heilleistungen aufzeichnen und nutzen – das erhöht die Genauigkeit der Overlay-Anzeige für den Spieler spürbar. Bei fremden ist es nett, aber optional, da Overheal-Vermeidung vor allem die eigene Heilung betrifft.)

Desync-Guard (Anzeigeglättung): Ein praktischer Kniff ist, die Overlay-Anzeige beim CastStart kurz einzufrieren (100–150 ms), um Netzwerklatenz bedingte Flackereien zu umgehen. Beispiel: Du beginnst einen Heal-Cast – dein Addon zeigt sofort den Overlay. Angenommen, ein anderer Heiler heilt in genau dem Moment das Ziel voll, das Event kommt aber mit 100ms Verzögerung an. Ohne Guard würde dein Overlay erst erscheinen, dann sofort verschwinden, weil das Ziel „schon voll“ geheilt wurde – das sieht wie ein Zucken im Balken aus. Besser: Bei Castbeginn kurz keine sofortigen Updates zulassen, sondern z.B. 0,1s warten, dann den neuesten Stand setzen. So „rastet“ die Anzeige stabil ein. Solche kleinen Delay-Filter verbessern die Nutzererfahrung erheblich und sind in bestehenden Addons üblich (man kennt es etwa von Zauberleisten, die eine Latenzfarbe am Ende haben – ähnliches Prinzip).

Event-Hooks (Cancel/Update): Die Projektion muss verworfen oder angepasst werden, sobald bestimmte Events eintreten: Bei UNIT\_SPELLCAST\_STOP/INTERRUPTED (Abbruch) wird die Precast-Projektion für diesen Spell sofort entfernt – der Balken verschwindet, da kein Heal mehr unterwegs ist. Bei COMBAT\_LOG\_EVENT\_UNFILTERED für SPELL\_HEAL oder SPELL\_ABSORBED usw. wird festgestellt, dass Heilung tatsächlich eingetroffen ist – dann aktualisiert man den echten HP-Wert (was ohnehin das Spiel-API tut) und passt auch den Overlay an (bzw. entfernt ihn, wenn es der eigene Heal war). Diese Hooks stellen sicher, dass Anzeige und Realität synchron bleiben. Ebenso muss auf harte HP-Änderungen reagiert werden (wenn z.B. ein Boss-Mechanismus MaxHP verändert oder ein Spieler stirbt/rezzed), um die Prognose zu invalidieren.

Latenz-Kompensation: (siehe oben bei Punkt „Precast-Engine“ und „Desync-Guard“). Im Kern: Verwende GetNetStats() um die aktuellen Latenzzeiten einzurechnen. Bei hoher Latenz wird der Overlay-Balken etwas konservativer agieren (da mehr ungewisse Zeit). Optional kann der Benutzer auch das SpellQueueWindow (CVar) einstellen, welches das Addon berücksichtigen soll – die meisten lassen es auf Standard (~400ms), aber es könnte als Feinschliff angeboten werden.

Abschließend noch ein Wort zur Machbarkeit im Gesamtbild: Die aufgezählten Komponenten mögen komplex wirken, aber jeder einzelne Aspekt stützt sich auf bereits bekannte Mechanismen und vorhandene Datenquellen im Spiel

wowpedia.fandom.com

wowace.com

. Viele moderne Heal-Addons implementieren Teile hiervon schon heute: Eingehende Heilbalken und Absorbs sind Standard, Reichweitenprüfung (IsSpellInRange für Farbmarkierung out-of-range), Klick-Cast-Bindings auf Raidframes etc. – all das ist erprobt und machbar. Die wirklich neue Innovation hier ist die Precast-Projektion unter Einbeziehung fremder Heals und Schaden. Doch selbst das wurde ansatzweise schon versucht: So hatte etwa HealBot früher eine Overheal-Anzeige, und das genannte Grid2-Plugin prognostiziert Overheal durch Zähler der Heiler auf dem Ziel

curseforge.com

.

Wichtig ist die Balance aus Nutzen vs. Kosten: Die Planungs-Tabelle im Konzept zeigt ja, welche Features hohen Nutzen bei vertretbarem Aufwand haben („KEEP“) – darauf sollte man sich zuerst konzentrieren (z.B. Incoming Heals Aggregator, Precast-Solver, Latenz-Kompensation sind Kernfunktionen und als „Hoch“ machbar eingestuft). Auf Features mit geringem Mehrwert aber hoher Komplexität (z.B. SmartCast-Automatismen, Mehrziel-Heilungsoptimierung) kann man zunächst verzichten, um Ressourcen nicht zu verzetteln.

Fazit

Aus der Recherche und den bestehenden Informationen lässt sich ableiten, dass ein fortschrittliches Heal-Addon für MoP Classic durchaus realisierbar ist, wenn man die richtigen Datenquellen nutzt und technische Hürden beachtet:

LibHealComm als Schlüssel: Um präzise und vollständige Heal-Vorhersagen zu bekommen, führt kaum ein Weg an LibHealComm vorbei. Die Library bietet solide Daten für alle Heilungen der Mitspieler – inklusive Berücksichtigung von Modifikatoren – solange alle Beteiligten sie installiert haben

wowace.com

. In klassischen Raids hat sich gezeigt: Je mehr Heiler HealComm nutzen, desto besser die Abdeckung und desto weniger Überheilung passiert, weil man sich gegenseitig nicht ins Geheilt „reinheilt“

reddit.com

. Es ist also sinnvoll, das Addon LHC-kompatibel zu machen und die Raid-Mitglieder (vor allem Heiler) zu ermuntern, ein Addon mit HealComm zu verwenden. Für den Rest muss ein Fallback vorhanden sein.

Blizzard-API als Fallback: Die integrierte API UnitGetIncomingHeals liefert in MoP Classic nur einen Teil der Wahrheit

wowpedia.fandom.com

. Sie alleine zu nutzen, wäre fahrlässig, da wichtige Bestandteile (HoTs, Channels) fehlen und sogar fehlerhafte Werte gemeldet wurden

github.com

wowinterface.com

. Aber als Fallback ist sie besser als nichts – zumindest werden gerade gewirkte große Heilungen anderer einigermaßen reflektiert. Das Addon sollte also die API-Werte einbeziehen, wo LibHealComm-Daten fehlen, jedoch im Hinterkopf behalten, dass diese unvollständig sind. Gegebenenfalls kann man durch heuristische Methoden (siehe Schätzen fremder Heals) die Lücken füllen

curseforge.com

.

Overheal- und Overlay-Anzeige: Durch die getrennte Overlay-Anzeige wird verhindert, dass Lebensbalken unrealistisch springen. Das Konzept „niemals Server-HP überschreiben“ hat sich bewährt – eingehende Heilungen werden stets nur angehängt, nicht direkt eingerechnet. Somit bleibt die Echtzeit-Konsistenz gewahrt: Was als echter HP-Wert angezeigt wird, ist immer vom Server bestätigt, und alles Weitere ist prognostisch und als solches erkennbar (z.B. blasser oder andersfarbig). Dies entspricht auch der Blizzard-Philosophie bei offiziellen Frames

reddit.com

.

Machbarkeitsbewertung: Die Implementierung erfordert zwar einigen Aufwand und sorgfältige Optimierung, aber keine unlösbaren technischen Hürden. Alle notwendigen Informationen sind entweder über offizielle APIs verfügbar oder lassen sich durch Addon-Kommunikation beschaffen. Existierende Addons liefern Blaupausen: VuhDo z.B. zeigt, dass man eingehende Heals + Absorbs in Classic darstellen kann (mit LHC), und community-getriebene Projekte haben Wege gefunden, fehlende Daten durch Beobachtung zu ergänzen. Die größten Herausforderungen – Performance im 40er-Raid und Datenaktualität – sind durch modernes Hardware und aktive Addon-Communities gut handhabbar. Sollte Blizzard in zukünftigen Classic-Patches die Incoming-Heal-API verbessern (etwa HoTs mit einbeziehen), würde das die Aufgabe weiter erleichtern; bis dahin muss man auf bewährte Addon-Lösungen setzen.

Bottom Line: Das Vorhaben eines perfekten Heal-Addons für MoP Classic ist ambitioniert, aber machbar. Mit LibHealComm als Fundament und cleveren Ergänzungen für Latenz, Schadensprognose und UI-Glättung lässt sich ein System bauen, das dem Heiler frühzeitig zeigt, was passieren wird – ein großer Vorteil für Healing-Assignments und Mana-Management im Raid. Wichtig ist, die Erwartungen richtig zu managen: 100% Exaktheit wird es nie geben (Krit-Heils und plötzlicher Schaden bleiben unvorhersehbar), aber schon eine nahezu aktuelle HP-Anzeige mit Berücksichtigung laufender Heals ist ein gewaltiger Schritt nach vorn, der laut Spielern zu „viel weniger Overheal/verschwendetem Mana“ führt

reddit.com

. In diesem Sinne steht der Umsetzung – bei Beachtung der genannten Punkte – nichts Fundamentales im Wege.

Quellen: Die Einschätzungen stützen sich auf die dokumentierten Erfahrungen aus Addon-Entwickler-Foren und -Logs (u.a. WowAce, WowInterface) sowie auf WoW-API-Dokumentation und Community-Rückmeldungen aus der Classic-Ära

wowace.com

wowpedia.fandom.com

github.com

wowinterface.com

reddit.com

curseforge.com

. Diese zeigen einheitlich, dass präzise Heal-Prediction zwar Aufwand bedeutet, aber technisch erreicht werden kann – und von Heilern geschätzt wird, wenn sie denn funktioniert.