Probe 1:

• Die Darstellungsschicht kann vollständig auf dem Server implementiert werden. → **FALSCH**
→Darstellungsschicht (Presentation Layer) ist für die Umwandlung und Formatierung von Daten zuständig und kann sowohl auf dem Client als auch auf dem Server implementiert werden, um Datenformate zu vereinheitlichen und sicherzustellen, dass die Kommunikation korrekt verstanden wird

• Kausalität ist durch Lamport-Zeitstample gesichert. → **FALSCH**
→Lamport-Zeitstempel sichern nur die Reihenfolge von Ereignissen, nicht jedoch die kausalen Abhängigkeiten. Kausalität erfordert, dass kausale Zusammenhänge explizit erkannt und berücksichtigt werden, was durch Vektor-Zeitstempel besser erreicht wird

• RPC ist immer Asymetrisch → **FALSCH**
→Remote Procedure Call (RPC) bezieht sich auf die Kommunikation zwischen einem Client und einem Server, bei der der Client eine Funktion auf dem Server aufruft. Der Begriff "asymmetrisch" ist hier nicht zutreffend. RPC kann synchron oder asynchron sein

• Threads setzen sich aus mehreren Prozessen zusammen. → **FALSCH**
→Threads sind leichtgewichtige Ausführungseinheiten innerhalb eines Prozesses. Ein Prozess kann mehrere Threads enthalten, aber nicht umgekehrt. Threads teilen sich den gleichen Speicherraum innerhalb eines Prozesses.

• Prozesse setzten sich aus mehreren Threads zusammen → **FALSCH**
→ Innerhalb eines Prozesses können ein oder mehrere Threads existieren. Threads sind Ausführungseinheiten innerhalb eines Prozesses, die sich den gleichen Speicherbereich und Ressourcen teilen.

• Digitale Signaturen benutzen Vektor-Zeitstample. → **FALSCH**
→Digitale Signaturen verwenden kryptografische Verfahren, typischerweise asymmetrische Verschlüsselung (öffentliche und private Schlüssel), nicht Vektor-Zeitstempel.

• Domain Namenauflösung ist nur rekursiv. → **FALSCH**
→Die Domain-Namenauflösung kann sowohl rekursiv als auch iterativ sein. In der rekursiven Auflösung erledigt der DNS-Server die gesamte Nachschlagearbeit, während in der iterativen Auflösung der Client jeden Schritt der Nachschlagekette selbst ausführt.

• DNS hat keine Replikate. → **FALSCH**
→DNS verwendet tatsächlich Replikate. DNS-Server sind häufig repliziert, um Ausfallsicherheit und Lastverteilung zu gewährleisten.

• DNS nutzt ausschließlich rekursive Namensauflösung. → **FALSCH**
→DNS kann sowohl rekursive als auch iterative Namensauflösung verwenden.

• Jeder Ablauf, der kausal konsistent ist, ist auch sequentiell konsistent. → **FALSCH**
→Kausale Konsistenz garantiert nur, dass alle kausalen Abhängigkeiten eingehalten werden, nicht jedoch die Reihenfolge aller Operationen. Sequentielle Konsistenz erfordert, dass alle Operationen in einer globalen Reihenfolge erscheinen, die mit der Reihenfolge der Operationen auf jedem einzelnen Knoten übereinstimmt.

1. Der Bully-Algorithmus funktioniert auch dann fehlerfrei, wenn mehrere Knoten ihn gleichzeitig starten. → **WAHR**
→Der Bully-Algorithmus kann korrekt funktionieren, auch wenn mehrere Knoten gleichzeitig eine Wahl initiieren. Schließlich wird der Knoten mit der höchsten ID gewinnen

2. Symmetrische Verschlüsselungsverfahren können auch für digitale Signaturen verwendet werden. → **FALSCH**
→ Symmetrische Verschlüsselungsverfahren eignen sich nicht für digitale Signaturen, da beide Parteien denselben Schlüssel verwenden. Digitale Signaturen erfordern asymmetrische Verschlüsselungsverfahren, bei denen ein privater Schlüssel zum Signieren und ein öffentlicher Schlüssel zum Verifizieren verwendet wird

3. Das TCP-Protokoll gehört zur Transportschicht. → **WAHR**
→TCP (Transmission Control Protocol) ist ein Protokoll der Transportschicht, das zuverlässige, verbindungsorientierte Kommunikation zwischen Anwendungen ermöglicht.

4. RPC ist immer synchron. → **FALSCH**
→ Remote Procedure Call (RPC) kann sowohl synchron als auch asynchron implementiert werden, je nach Anwendungsfall.

5. Jeder Ablauf, der sequentiell konsistent ist, ist auch kausal konsistent. → **WAHR**
   →Sequentielle Konsistenz ist strenger als kausale Konsistenz. Ein sequentiell konsistenter Ablauf ist somit auch kausal konsistent.

6. Die Anwendungsschicht kann auf Client- und/oder Servermaschine implementiert werden → **WAHR**
   →Die Anwendungsschicht kann sowohl auf der Client- als auch auf der Serverseite implementiert werden, abhängig von der spezifischen Anwendung.

7. Das Ethernetprotokoll gehört zur Vermittlungsschicht. → **FALSCH**
   →Das Ethernet-Protokoll gehört zur Sicherungsschicht (Data Link Layer).

8. Der Domain Name Service verwendet ausschließlich rekursive Namensauflösung. → **FALSCH**

   →DNS kann sowohl rekursive als auch iterative Namensauflösung verwenden

9. Bei quorumbasierten Protokollen wird in der Regel gefordert, dass das Schreibquorum mehr als die Hälfte der Knoten umfasst. Damit wird sichergestellt, dass stets der aktuellste Wert gelesen wird. → **WAHR**
   →Quorumbasierte Protokolle fordern typischerweise, dass das Schreibquorum mehr als die Hälfte der Knoten umfasst, um sicherzustellen, dass jede Leseoperation mindestens einen Knoten aus dem Schreibquorum beinhaltet und somit den aktuellsten Wert liest.

10. Mit Lamportzeitstempeln ist es nicht möglich, kausale Zusammenhänge verlässlich zu erkennen. → **WAHR**
   →Lamport-Zeitstempel können die Reihenfolge von Ereignissen bestimmen, aber sie reichen nicht aus, um kausale Zusammenhänge sicher zu erkennen, da sie keine Informationen über die tatsächlichen Abhängigkeiten zwischen Ereignissen enthalten.

---

11. Zugriffstransparenz: Erreicht Amazon Dynamo diese Art der Transparenz?

→ Zugriffstransparenz wird erreicht in gewissen Maße
→ Benutzer und Anwendungs-Daten können unabhängig von der physischen Speicherung und Verteilung der Daten Abgerufen werden.
→ Die Datenbank abstrahiert die Details der Datenverteilung und -speicherung, sodass Benutzer sich nicht darum kümmern müssen, wo genau die Daten gespeichert sind oder wie sie auf verschiedene Server verteilt werden
→wird erreicht durch: -> Abstraktion der Datenverteilung (verteilt Daten automatisch auf verschiedene Partitionen (Knoten), um Skalierbarkeit, Leistung und gleichmäßige Lastenverteilung zu gewährleisten)
->Einfacher Datenzugriff (Benutzer greifen über einfache API-Aufrufe, put() und get(), auf Daten zu, somit werden Unterschiede in Datenrepräsentation und im Zugriff verborgen)
->Replikation (Daten werden automatisch in mehrere Verfügbarkeitszonen innerhalb einer Region repliziert und DynamoDB Global Tables ermöglichen die Replikation von Daten über mehrere AWS-Regionen hinweg

12. Lokationstransparenz: Erreicht Amazon Dynamo diese Art der Transparenz?

→ Zugriffstransparenz wird erreicht
→ Wird meist durch einen generischen Load Balancer erreicht, der alle Anfragen entgegennimmt und entsprechend der Last an Knoten weiterleitet. Es gibt aber auch eine Client-Bibliothek. Dabei werden die Knoten über eine Präferenzliste erreicht
→ Dynamo verwendet eine DHT (verteilte Hashtabelle), um Daten auf die Knoten im Cluster zu verteilen. Die DHT sorgt dafür, dass die genaue Position von Daten auf den Knoten für die Anwendungsebene nicht sichtbar ist. Benutzeranfragen werden über ein konsistentes Hashing-System an die entsprechenden Knoten weitergeleitet, ohne dass die Anwendungslogik Kenntnisse über die physische Lage der Daten benötigt

13. Replikationstransparenz: Erreicht Amazon Dynamo diese Art der Transparenz?

→ Replikationstransparenz wird erreicht
→ Dynamo verwendet eine Replikationsstrategie, bei der Daten auf mehrere Knoten verteilt werden, um sowohl Ausfallsicherheit als auch Datenverfügbarkeit zu gewährleisten. Die Replikation wird von Dynamo intern verwaltet, sodass die Anwendung keine expliziten Anweisungen zur Replikation geben muss
→ Diese Replikate werden basierend auf den Hashes der Schlüssel und der Präferenzliste verteilt

14. Mehrbenutzertransparenz: Erreicht Amazon Dynamo diese Art der Transparenz?

→ Mehrbenutzertransparenz wird nicht wirklich erreicht

→ Man nutzt nur die definierte Schnittstelle. Von anderen Nutzern erfährt man gar nichts

→ Jeder Nutzer hat eigenen Warenkorb und eigene Empfehlungen etc. → keine Interaktion oder Konflikte möglich. Außerhalb davon können mehrere Benutzer Amazon (Warenkorb) verwenden ohne Einschränkungen zu Erfahren

→ Dynamo verwendet ein Eventual-Consistency-Modell, was bedeutet, dass es keine sofortige Konsistenz über alle Replikate hinweg garantiert. In der Praxis bedeutet das, dass verschiedene Benutzer oder Clients möglicherweise unterschiedliche Versionen der Daten sehen, wenn sie gleichzeitig auf dasselbe Datenobjekt zugreifen oder es ändern. Diese Verzögerung in der Konsistenz kann zu inkonsistenten Datenansichten führen, bis alle Replikate synchronisiert sind

→ man kann in einem Amazon Account nichts im Warenkorb verbergen, acuh wenn 2 Personen diesen Account nutzen

15. Fehlertransparenz: Erreicht Amazon Dynamo diese Art der Transparenz?

→Fehlertransparenz wird erreicht

→Wenn ein Fehler auftritt gibt es verschiedene Mechanismen. Bei temporären Fehlern wird Hinted Handoff genutzt bei Permanenten Fehlern wird ein Anti-Entropie Protokoll zur Synchronisierung genutzt

→Hinted Handoff und Sloppy Quorum: Diese Techniken ermöglichen es, dass Daten bei temporären Ausfällen auf gesunden Knoten gespeichert werden und später bei Verfügbarkeit auf die Zielknoten übertragen werden, was die Verfügbarkeit und Fehlertoleranz erhöht.

→Antientropie-Protokoll und Merkle-Bäume: Durch regelmäßige Synchronisierung der Daten zwischen Knoten mittels Antientropie-Protokollen und effizientem Vergleich von Knoteninhalten mit Merkle-Bäumen werden Dateninkonsistenzen erkannt und behoben, um eine kontinuierliche Konsistenz trotz Ausfällen zu gewährleisten.

→Informationsverteilung über Gossip-Protokoll: Gossip-Protokoll, das die Verbreitung von Mitgliedschafts- und Fehlerinformationen ohne zentrale Koordination ermöglicht.

---

16. Verteilte Systeme sollen in unterschiedlichen Dimensionen skalieren.
Nennen Sie diese Dimensionen und diskutieren Sie (kurz), ob und ggf. wie Amazon Dynamo diese unterstützt.

• **in Größe**: → konstante Antworten im Millisekunden-Bereich at any scale und damit die Skalierbarkeit in der Anzahl der Nutzer, Menge der Daten und Zahl von Prozessen

→Datenmengen: Nutzt konsistentes Hashing und virtuelle Knoten zur gleichmäßigen Datenverteilung und effizientem Datenmanagement.

→Anfragen (Throughput): Ermöglicht durch Datenreplikation und eventual consistency, stets schreibbar zu bleiben, und durch Sloppy Quorum und Hinted Handoff auch bei Ausfällen eine hohe Verfügbarkeit zu gewährleisten.

→Anzahl der Knoten (Nodes): Dezentrale Peer-to-Peer-Architektur und virtuelle Knoten ermöglichen elastische Skalierung und gleichmäßige Lastverteilung.

• **In maximalen Entfernung zwischen Knoten**:

→ist skalierbar, weil amazon global aufgestellt ist, aber vielleicht auch nicht und es werden lieber mehr Knoten mit Replikaten platziert, als die Entfernung zu erhöhen (hoch skalierbar in horizontaler Richtung, also Anzahl Nutzer, nicht Entfernung)

→Präferenzliste eines Schlüssels wird so erstellt, dass die Speicherknoten über mehrere Rechenzentren verteilt sind.

• **In Zahl der beteiligten Organisationen:**

→macht keinen Sinn, da Amazon Dynamo lediglich intern verwendet wurde (nicht skalierbar). Mit dem Wechsel auf DynamoDB und der Nutzung auch außerhalb von Amazon ist es dann auch in der Zahl der beteiligten Organisationen skalierbar.

→ es wird skaliert durch: Dezentrale Architektur, die Organisationen unabhängig hinzufügen lässt. Konsistentes Hashing und virtuelle Knoten, die eine einfache Integration neuer Knoten ermöglichen. Flexible Replikationsstrategien, die es ermöglichen, Replikate auf verschiedenen Knoten, auch von unterschiedlichen Organisationen, zu verteilen. Gossip-Protokoll, das die Verbreitung von Mitgliedschafts- und Fehlerinformationen ohne zentrale Koordination ermöglicht. Modulare Struktur, die Anpassungen und verschiedene Implementierungen erlaubt.

→In der Zahl der beteiligten Organisationen (Skalierung durch die Integration mit AWS IAM (Identity and Access Management), das feinkörnige Zugriffskontrollen ermöglicht. Administratoren können die Zugriffsrechte für Benutzer und Anwendungen präzise steuern und verwalten)

Im Dynamo Papier wird in den Abschnitten 42 und 4.3 eine Datenverteilung und Replikation beschrieben. Die folgenden Fragen beziehen sich auf diese Abschnitte.

17. Amazon Dynamo erlaubt es, Server, die auf unterschiedlicher Hardware laufen (und damit unterschiedlich leistungsfähig sind) zu nutzen. Über eine Hashfunktion werden Daten auf Knoten verteilt. Wie wird dabei die unterschiedliche Leistungsfähigkeit von Servern berücksichtigt?

→ durch virtuelle Knoten wird es berücksichtigt
→ Leistungsstarke knoten haben viele virtuelle Knoten (oder mehr als Leistungsschwache)
→Leistungsschwache Knoten haben wenig virtuelle Knoten

→ Grundprinzip der virtuellen Knoten:
- Anstatt jedem physischen Knoten (Server) im Cluster direkt eine feste Menge an Daten zuzuweisen, wird der Datenraum in viele kleinere virtuelle Knoten unterteilt.
- Jeder physische Knoten im Cluster wird dann für eine Reihe von virtuellen Knoten verantwortlich gemacht.

→Verteilung der vnodes:
- Bei der Verteilung der vnodes wird berücksichtigt, wie leistungsfähig ein physischer Server ist. Leistungsfähigere Server können mehr vnodes zugewiesen bekommen, während weniger leistungsfähige Server entsprechend weniger vnodes erhalten.
- Diese Verteilung sorgt dafür, dass die Last proportional zur Kapazität und Leistungsfähigkeit der physischen Server verteilt wird

→ Durch virtuelle Knoten können Leistungsstärkeren physischen Knoten im Overlaynetzwerk mehrere virtuelle Knoten zugeteilt werden, während die schwächsten physischen Knoten nur einen virtuellen Knoten zugeteilt bekommen. (Zuweisung von einem physischen Knoten an mehrere Stellen im Ring)

18. Dynamo repliziert jedes Datenelement auf N Knoten, wobei N instanzspezifisch ist. Zusätzliche Replikate können im Fehlerfall angelegt werden (Abschnitt 4.6 im Artikel).
In der Vorlesung hatten wir drei Arten von Replikaten kennengelernt: *Permanente Replikate, Serverinitiierte Replikate* und *Clientinitierte Replikate.* Welche davon spielen bei Amazon Dynamo eine Rolle? Begründen Sie Ihre Antwort kurz.

→ **Permanente Replikate:**
- in Dynamo wird jeder Datensatz auf N Knoten repliziert, wobei N konfigurierbar ist. Diese Replikate sind dauerhaft und sorgen dafür, dass Daten auch bei Ausfällen einzelner Knoten verfügbar bleiben.
- normalerweise grundlegend durch das Datenbankdesign gegeben und sehr wenige in ihrer Anzahl.

→**Serverinitiierte Replikate:**
- Beispielsweise nutzt Dynamo „hinted handoff“, bei dem ein Knoten, der temporär die Verantwortung für die Daten eines ausgefallenen Knotens übernimmt,
- temporär und werden angefordert, wenn eine hohe Last bemerkt wird

→ **Clientinitiierte Replikate:**
- Spielen in Dynamo eine weniger bedeutende Rolle, da die Verwaltung der Replikation und Konsistenz serverseitig erfolgt. (jeder Knoten ist gleichberechtigt → P2P Struktur)

---

19. Wir hatten in der Vorlesung unterschiedliche Varianten kennengelernt, wann und wie Replikate aktualisiert werden können. Nennen Sie diese Varianten und ordnen Sie das bei Dynamo verwendete Verfahren ein.

→Es gibt Datenzentrische (continuous consistency, Eintrittskonsistenz, eventual consistency) und Klientenzentrische Konsistenz (monotones lesen/schreiben und read-your-writes/write-follows-reads)

→Dynamo verwendet die eventual consistency (letztendliche Konsistenz), eine schwächeres Konsistenzmodell, bei welchem lediglich garantiert wird, dass alle Änderungen irgendwann überall ankommen
(eignet sich für Systeme mit wenigen Schreib-Schreib-Konflikten → Warenkorb wird nur vom Kunden befüllt)

→Hinted Handoff: Dynamo verwendet das sogenannte „Sloppy Quorum“. Hierbei werden Schreib- und Leseoperationen auf den ersten N gesunden Knoten ausgeführt.

→Speicherung des Replikates auf einem anderen Knoten (inkl. Hinweis auf orig.Empfänger) – „Hinted Replica“ . Transfer des Replikates bei erneuter Verfügbarkeit des orig. Empfängers

→Anti-Entropie-Protokoll zur Synchronisierung (Umgang mit permanenten Ausfällen (Replikat-Synchronisierung))

→Optimierung durch die Verwendung von Merkle-Bäumen mit effizientem Vergleich von Knoteninhalten (Objekte) zur Identifikation von Inkonsistenzen (Umgang mit permanenten Ausfällen (Replikat-Synchronisierung))

Die folgenden Fragen beziehen sich auf das von Amazon Dynamo verwendete Konsistenzmodell und -protokoll und damit zusammenhängende Aspekte.

20. Dynamo verwendet "Vector Clocks". Diese haben eine ähnliche Funktion wie Vektorzeitstempel. Allerdings gibt es auch Unterschiede zwischen beiden.
Nennen und erläutern Sie (kurz) zwei Aspekte, unter denen sich die beiden Dinge voneinander unterscheiden.

→ C: Datenelementspezifisch; Z: Prozessspezfisch

→ C: sicheres Wissen; Z: Annahme über Aktivitäten

→ C: spiegelt Schreiboperationen wider; Z: kann unterschiedliches widerspiegeln je nach Konfiguration

→ C: beschreibt Aktualität (Version) der Daten -> bezieht sich auf Objekt
Z: beschreibt Tun eines Prozesses, aber nicht was es tut

21. Im Gegensatz zu klassischen Quorumbasierten Verfahren fordert Dynamo nicht, dass R+W > N sein muss.
Welche Eigenschaft geht damit verloren? Wie löst Dynamo dadurch entstehende Probleme?

→Quorumbasierte (sloppy Quorum) → Hierbei müssen eine bestimmte Anzahl von Knoten (ein Quorum) eine Operation bestätigen, um sicherzustellen, dass die Daten konsistent bleiben. (Unterschied zu normalem Quorum ist, dass die Anzahl der benötigten Knoten nicht W + R > N, sondern W + R < N ist)

→ Datenkonsistenz geht verloren

22. Das CAP-Theorem besagt, dass ein verteiltes System stets nur zwei der folgenden drei Eigenschaften erreichen kann: Konsistenz (Consistency), Verfügbarkeit (Availability) und Toleranz gegenüber Netzwerkpartitionen (Partition tolerance). Beschreiben Sie für Dynamo oder ein anderes System ihrer Wahl (z.B. ein klassisches RDBMS) welche der beiden Eigenschaften erreicht werden. Begründen Sie, warum die Entscheidung auf diese beiden Eigenschaften fiel ODER begründen Sie warum nicht alle drei gleichzeitig erreicht werden können.

→ Das CAP-Theorem beschreibt, dass in einem verteilten System nur zwei der drei Eigenschaften gleichzeitig erreicht werden können (consistency, availability, partition tolerance) → Also CA, CP und AP
(dabei ist wichtig, dass die letzte nicht komplett ignoriert wird, sondern nur in abgeschwächter Form existieren kann z.B. bei AP kann nur eingeschränkte Konsistenz gewährleistet werden ≠ gar keine Konsistenz) Amazon Dynamo fokussiert sich dabei auf AP und verwendet die abgeschwächte eventual consistency. In DynamoDB gibt es auch die Möglichkeit auf strengere Konsistenz, wobei die Verfügbarkeit schlechter wird (CP mit schwächerem A)

---

23. Dynamo so wie es im Artikel beschrieben wurde, verfügt über keinerlei Sicherheitsmechanismen.
Nehmen Sie an, Sie wurden beauftragt, Dynamo bzgl. Sicherheit so weiter zu entwickeln, dass es nicht nur rein intern, sondern auch für Anwendungen mit externen Beteiligten (also menschlichen Benutzern oder Software, die zu anderen Organisationen gehören) benutzt werden kann. Welche Sicherheitsmaßnahmen schlagen Sie vor? Decken Sie möglichst viele der in der Vorlesung behandelten Aspekte ab.

→**Authentifizierung und Autorisierung**:

- **API-Schlüssel und Token:** Implementierung von API-Schlüsseln und Tokens, um sicherzustellen, dass nur authentifizierte Benutzer auf die Datenbank zugreifen können.

→**Datenverschlüsselung**:

- **Verschlüsselung im Ruhezustand**: Verschlüsselung der gespeicherten Daten (at rest) mit starken Verschlüsselungsalgorithmen
- **Netzwerksicherheit**: Einsatz von VPCs, um den Datenverkehr zu isolieren und sicherzustellen, dass nur autorisierter Verkehr in und aus den dynamischen Clustern fließen kann.

→**DDoS-Schutz (Distributed Denial of Service):**

- **AWS Shield und AWS WAF (Web Application Firewall):** Einsatz von AWS Shield zum Schutz vor DDoS-Angriffen und AWS WAF zum Schutz vor Web-basierten Angriffen wie SQL-Injektionen und XSS (Cross-Site Scripting).

→ **Datenintegrität und Verfügbarkeit**

- **Backups und Snapshots**: Regelmäßige Erstellung von Backups und Snapshots der Datenbank, um im Falle eines Datenverlusts eine Wiederherstellung zu ermöglichen

→Authentifizierung mit Session Keys → Zu Beginn asymmetrische Verschlüsselung (weil teuer) und dann weiter mit symmetrischer Verschlüsselung
→Digitale Signaturen mithilfe von public/private Key und Hashwerten (Vergleichen der Werte bzw. des Inhalts)
Autorisierung (Zugriffsmöglichkeit) mit ACM/ACM oder via Kompetenzen

---

Probe 2:

1.a) Transparenzen erklären:

A. Zugriffstransparenz: Verberge, wo sich eine Ressource befindet
B. Lokationstransparenz: Verberge, wo sich eine Ressource befindet
C. Replikationstransparenz: Replikation von Daten werden so verwaltet wird, dass die Benutzer und Anwendungen nicht direkt damit interagieren oder sich um die Details der Replikation kümmern müssen
D. Mehrbenutzertransparenz: Verberge, dass mehrere Nutzer konkurrierend auf eine Ressource zugreifen
E. Fehlertransparenz: Verbergen von Fehler und das Wiederanlaufen einer Ressource

1.b) Zu welchen Transparenzen tragen folgende Mechanismen bei? (Buchstaben aus a danebenschreiben)

RPC:
Replikation von Daten:
Namensauflösung:
Middleware:
Wechselseitiger Ausschluss:

---

2. Drei Unterschiede zwischen Chord & Amazon Dynamo's distributed Hashtable ?

→ Anpassung der Anzahl virtueller Knoten je physischen Knoten

→ Replikation: Jede Dynamo-Instanz entscheidet über eine Zahl N, welche angibt, wie oft ein Datenelement repliziert wird. Währen Chord keine flexible Replikationsstrategie gibt, die sich an die Bedürfnisse einer Anwendung anpassen lässt.

→ Konsistenzmodell: -> Chord verwendet ein einfaches Konsistenzmodell während
-> Dynamo setzt auf ein „eventual consistency"-Modell. Es erlaubt inkonsistente Daten

→ Fehlertoleranz und Verfügbarkeit:
-> Chord: Fehlertoleranz durch „successor lists" und „finger tables".
-> Amazon Dynamo: Hohe Verfügbarkeit und Fehlertoleranz durch Quorum-Protokolle und „hinted handoff".

3.a) Vektorzeitstempel für folgende Ereignisse angeben (Empfangen wird mitgezählt) in allen drei Prozessen:

→ Senden wird immer gezählt → hier extra mit Empfangen
→ Vektorzeitstemple soll zeigen wie viel welcher Prozess etwas getan hat
→ am Anfang ist Stempel immer auf Vektor 0 → dh Index ist auch 0
→ Vektor ist so groß wie Anzahl der Prozesse

Ereignis 1: P1 sendet eine Nachricht an P2
[1, 0, 0] -> P1: was weiß P1 über P1 | was weis P1 über P2 | was weis P1 über P3
[0, 0, 0] -> P2: analog
[0, 0, 0] -> P3: analog

Ereignis 2: P2 erhält Nachricht von P1 | -> P2 einzige was sich ändert, P1 und P3 können übernommen werden
[1, 0, 0] -> übernommen
[1, 1, 0] -> P2: P2 vergleichen mit P1 -> immer höherer nehmen | -> empfangen mitzählen (P2 weis über P2)
[0, 0, 0] -> übernommen

Ereignis 3: P2 sendet eine Nachricht an P3
[1, 0, 0]
[1, 2, 0]
[0, 0, 0]

Ereignis 4: P3 erhält eine Nachricht von P2 | -> wenn erhalten, dann die Vektoren vergleichen
[1, 0, 0]
[1, 2, 0]
[1, 2, 1]

Ereignis 5: P3 sendet eine Nachricht an P1
[1, 0, 0]
[1, 2, 0]
[1, 2, 2]

Ereignis 6: P1 erhält eine Nachricht von P3
[2, 2, 2]
[1, 2, 0]
[1, 2, 2]

2.b) Kausal zugeordnete Nachrichten: P1 hat jetzt Zeitstempel [14, 7, 5], wie muss eine Nachricht von P2 aussehen, damit sie sofort zugestellt wird? Wie muss sie aussehen, damit sie zurückgehalten wird? Kurz begründen.

→Nachricht von P2: → es muss gelten: ts(m)[i] > V C_j[i] →2te Index von P2 muss größer sein als der von P1
→1te Index von P2 muss kleiner gleich sein als der von P1
→3te Index egal

2.c) Selbe wie Aufgabe 2.a) nur mit Lamportzeitstempel.

Ereignis / Prozess	P1	P2	P3
Ereignis 1: P1 sendet eine Nachricht an P2	1	0	0
Ereignis 2: P2 erhält Nachricht von P1	1	2	0
Ereignis 3: P2 sendet eine Nachricht an P3	1	3	0
Ereignis 4: P3 erhält eine Nachricht von P2	1	3	4
Ereignis 5: P3 sendet eine Nachricht an P1	1	3	5
Ereignis 6: P1 erhält eine Nachricht von P3	6	3	5

→wenn erhalten wird: - Sendende wird mit Erhaltenden verglichen und Maximum + 1

→ bei Senden: die Prozesse die nicht aktiv sind werden normal übernommen, nur aktueller Prozess erhöht um 1

3.)

P1	W(x)a
P2	R(x)a W(x)b W(x)c
P3	R(x)b R(x)a R(x)c
P4	R(x)a R(x)b R(x)c

3.a) Welche Konsistenz ist das? Kurze Begründung Linearisierbarkeit, sequentiell, kausal, FIFO.

→ FIFO

FiFo-Konsistenz: • Alle Schreiboperationen eines Prozesses werden von allen anderen Prozessen in derselben Reihenfolge wahrgenommen.
• Schreibzugriffe verschiedener Prozesse können von unterschiedlichen Prozessen in unterschiedlicher Reihenfolge gesehen werden.
• Nur Schreiboperationen eines einzelnen Prozesses werden in der Betrachtung berücksichtigt.

3.b) Ein Datenelement x wird auf 10 Knoten gespeichert. Es wird viel gelesen und wenig geschrieben. Welches Lese- und Schreibequorum würden Sie einsetzen? Begründen

→ da man wenig schreibt und viel liest will man eine hohe konsistenz haben das heißt das Schreibquorum sollte sehr hoch sein und das Lesequorum niedrig, weil da man sehr wenig schreibt hat man auch verhätlnismäßig geringe kosten als wenn man viel schreiben würde und wenig lesen tut. man will also schnelle lesezugriffe haben und sich schnell sicher sein, dass man eine aktuelle version liest ein beispiel dafür wäre das das Schreibquorum = 9 und das lesequorum = 2 dadurch ist 9 + 2 = 11 > 10

3.c) Mit welchem Modell könnte man monotones Lesen durchsetzen?

3.d) CAP-Theorem: Beispiele und Begründung für CA, CP und AP