US 20100094962A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2010/0094962 A1**

(43) **Pub. Date: Apr. 15, 2010**

Zuckerman et al.

(54) **INTERNET BACKBONE SERVERS WITH EDGE COMPENSATION**

(75) Inventors: **Gal Zuckerman**, Petah-Tikva (IL); **Gil Thieberger**, Kiryat Tivon (IL)

Correspondence Address:
Gil Thieberger
Active Knowledge Ltd., POB 294
Kiryat Tivon 36010 (IL)

(73) Assignee: **PatentVC Ltd.**, Kiryat Tivon (IL)

(21) Appl. No.: **12/579,375**

(22) Filed: **Oct. 14, 2009**

Related U.S. Application Data

(60) Provisional application No. 61/105,683, filed on Oct. 15, 2008.

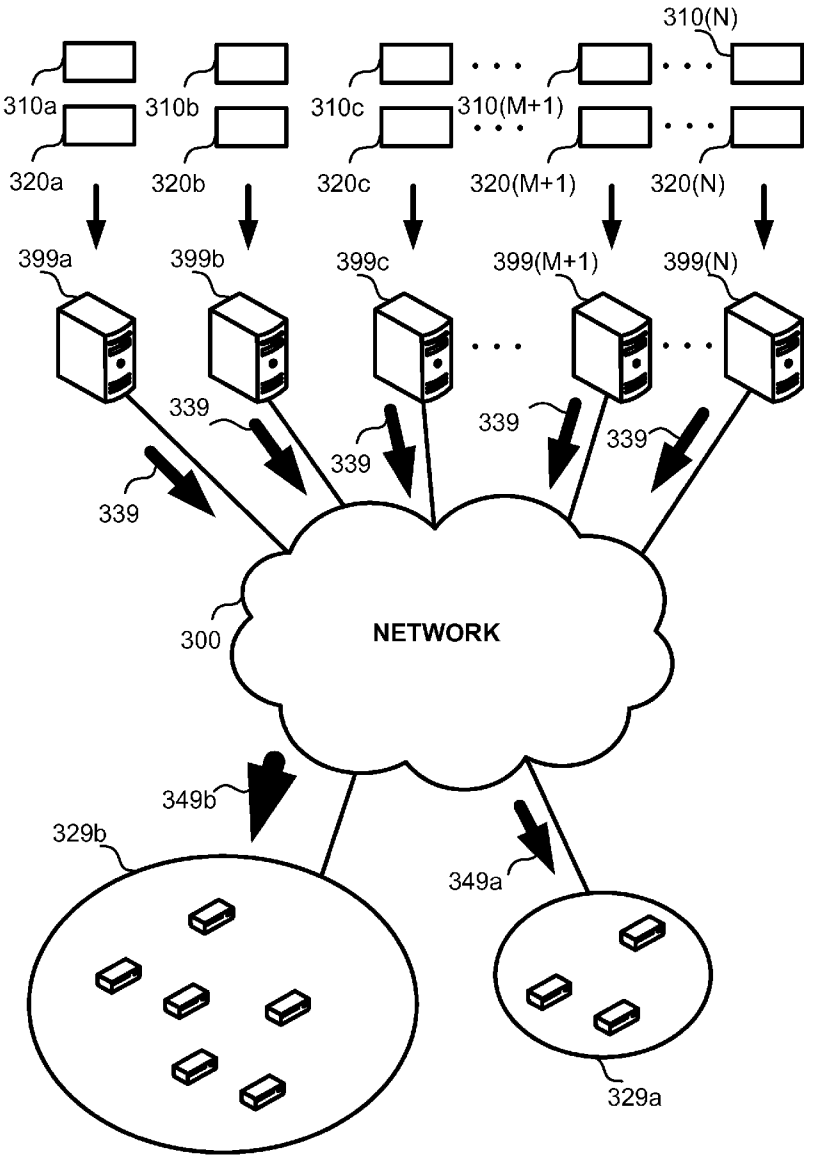
Publication Classification

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/219**; 709/231

(57) **ABSTRACT**

Internet backbone servers with edge compensation and a corresponding assembling device for obtaining erasure-coded fragments from at least one of the Internet backbone servers. Upon a fragment loss, the assembling device uses a fragment pull protocol to retrieve a substitute erasure-coded fragment from a nearby fractional-storage CDN server having low latency in response to the assembling device.



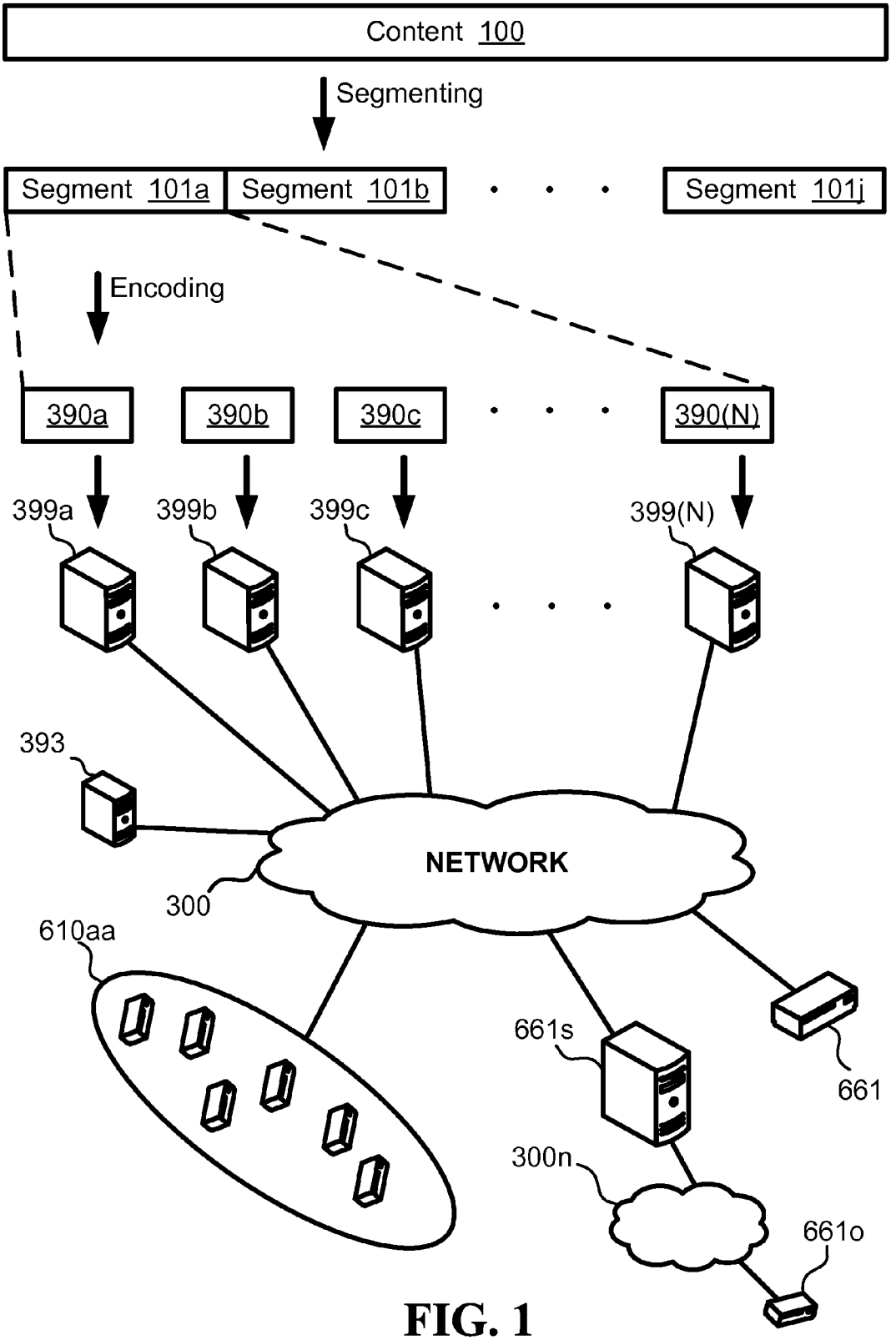

Content 100
Segmenting
Segment 101a
Segment 101b
Segment 101j
Encoding
390a
390b
390c
390(N)
399a
399b
399c
399(N)
393
NETWORK
300
610aa
661s
661
300n
661o

FIG. 1

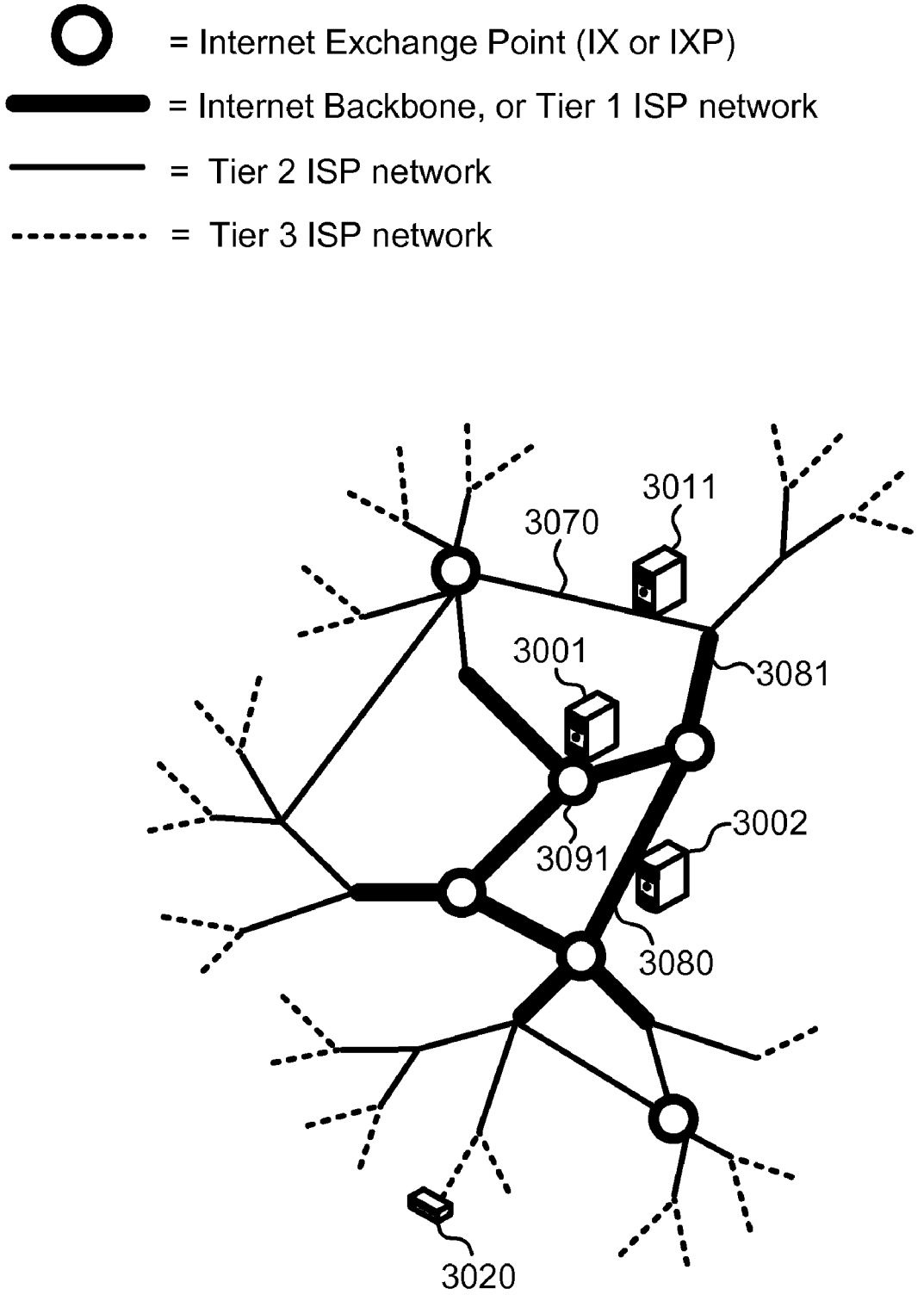

= Internet Exchange Point (IX or IXP)
= Internet Backbone, or Tier 1 ISP network
= Tier 2 ISP network
= Tier 3 ISP network
3011
3070
3001
3081
3002
3091
3080
3020

FIG. 2

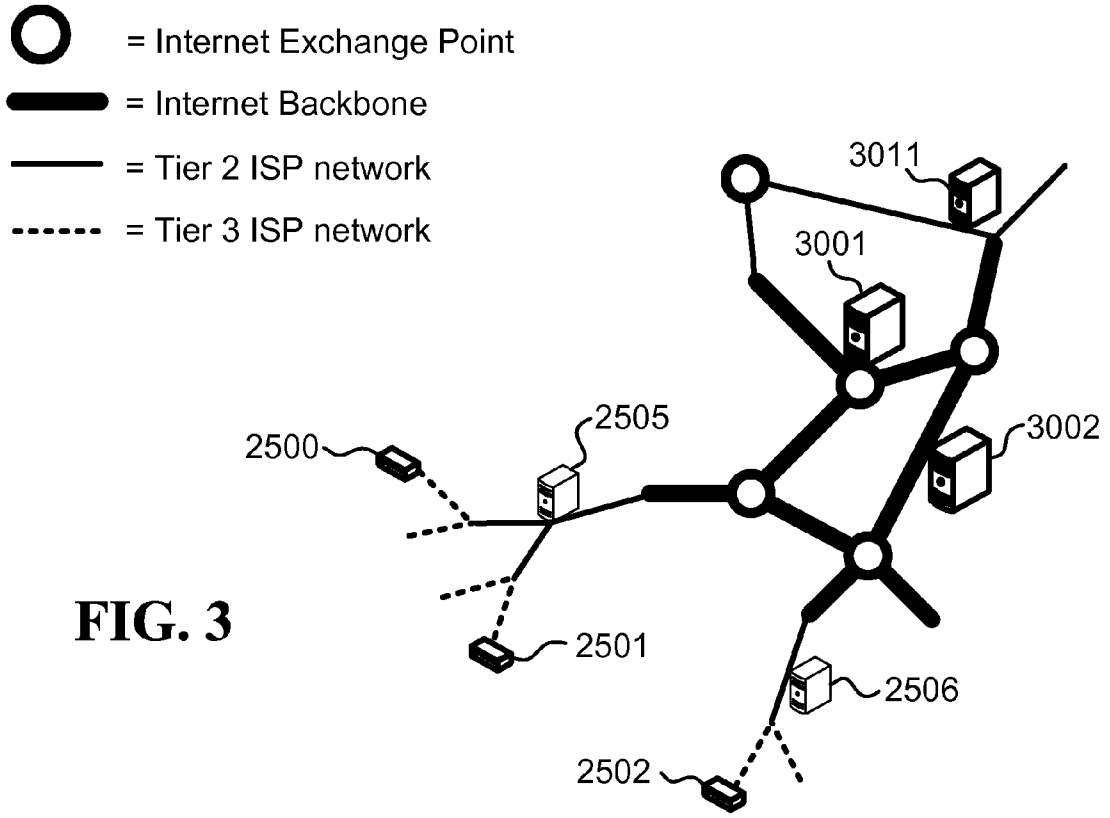

= Internet Exchange Point
= Internet Backbone
= Tier 2 ISP network
= Tier 3 ISP network
3011
3001
3002
2505
2500
2501
2506
2502

FIG. 3

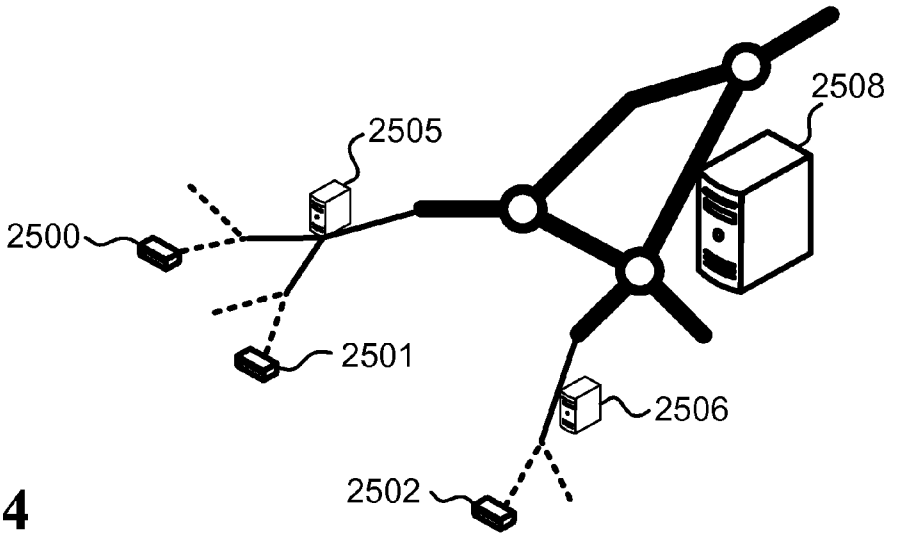

2508
2505
2500
2501
2506
2502

FIG. 4

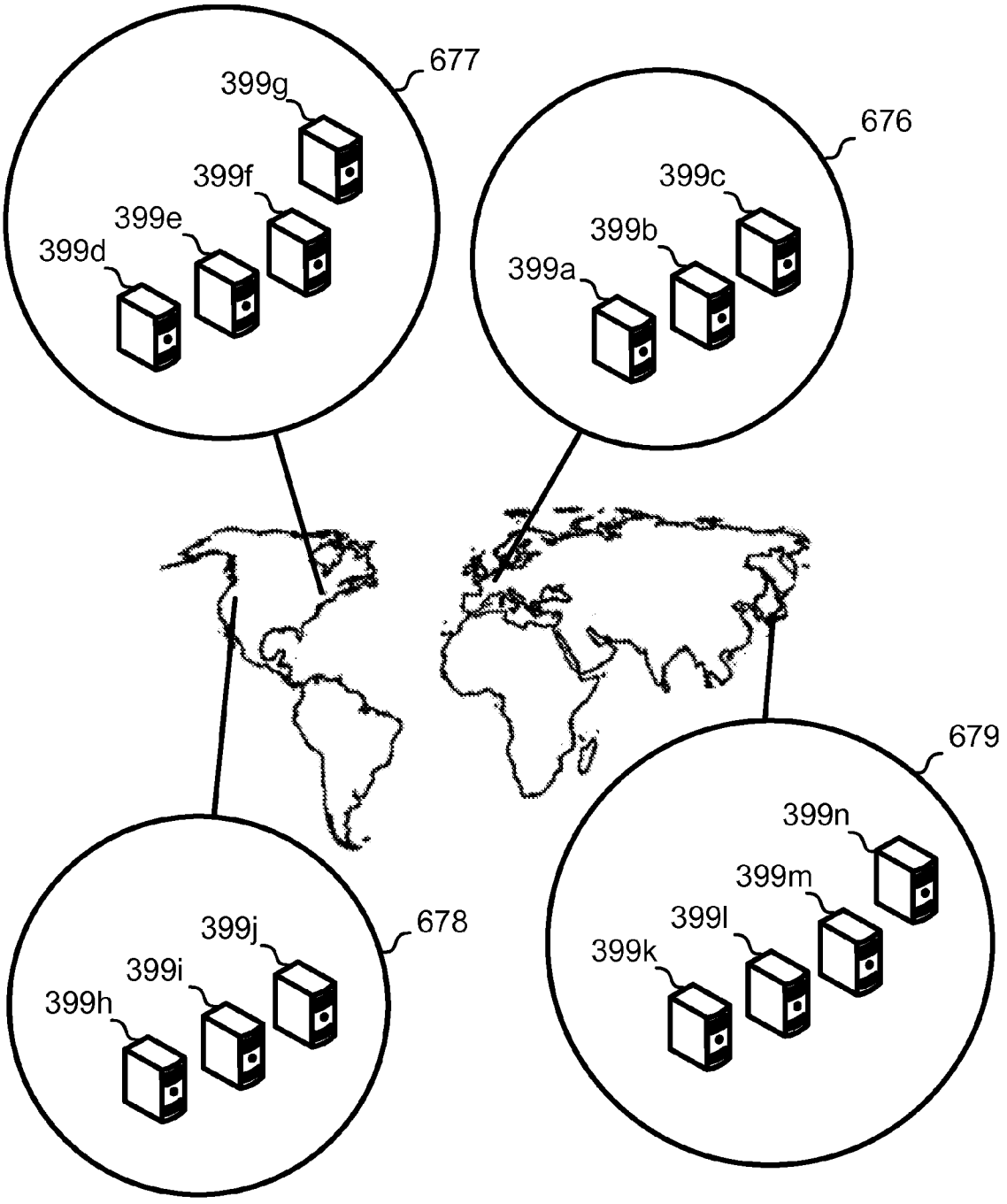

677
399g
399f
399e
399d
676
399c
399b
399a
679
399n
399m
399l
399k
678
399j
399i
399h

FIG. 5

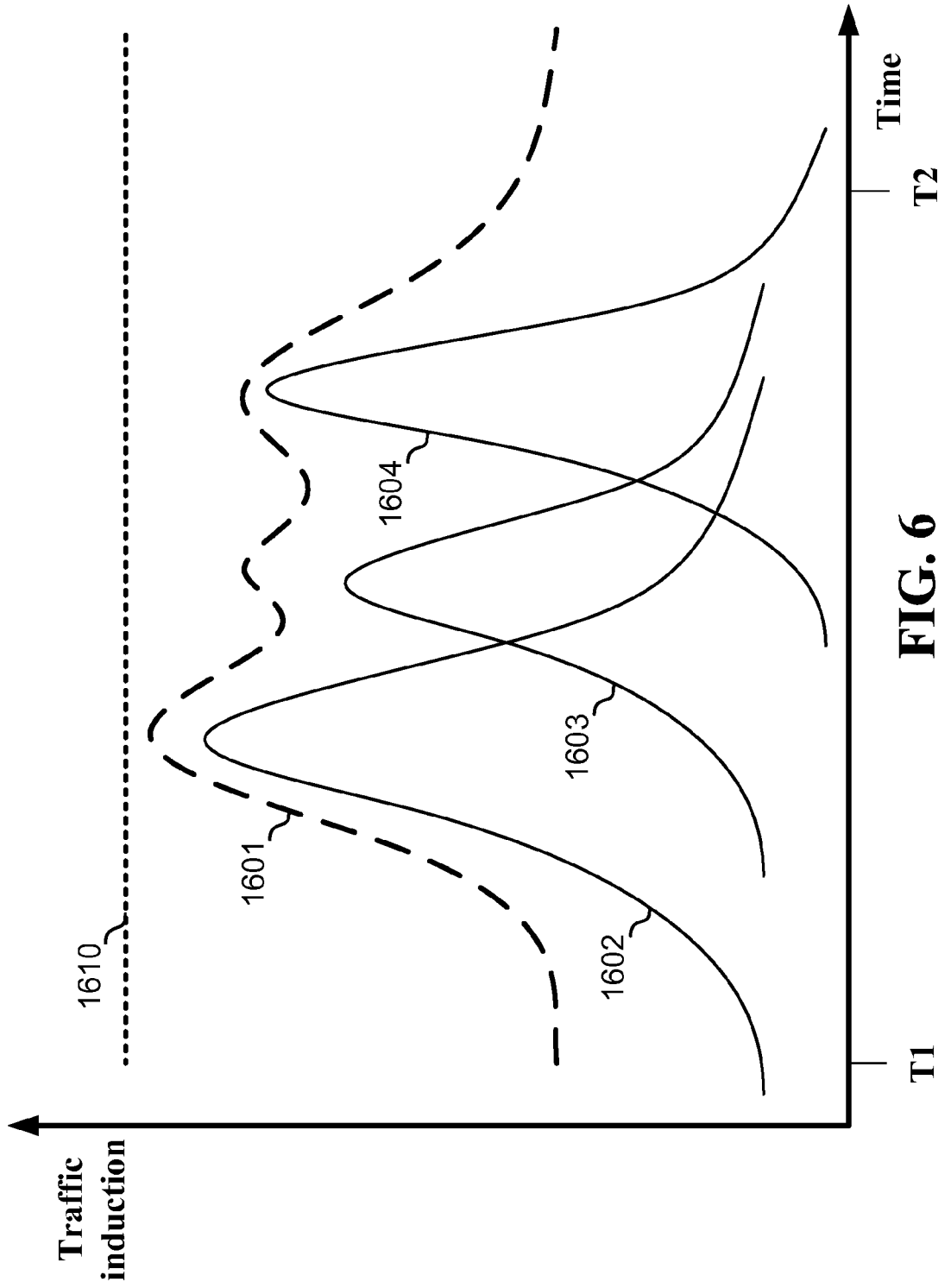


FIG. 6

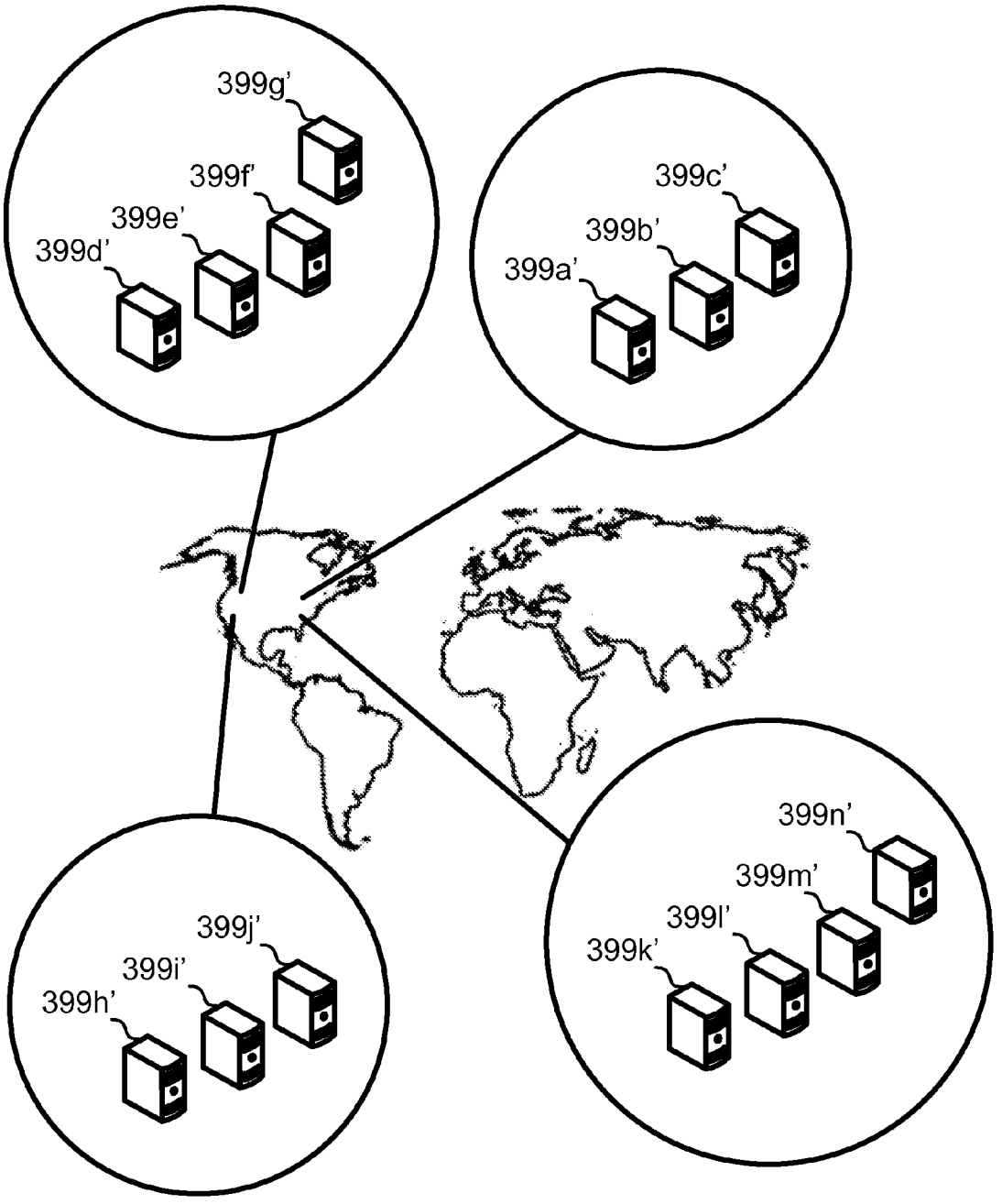

399g'
399f'
399e'
399d'
399c'
399b'
399a'
399n'
399m'
399l'
399k'
399j'
399i'
399h'

FIG. 7

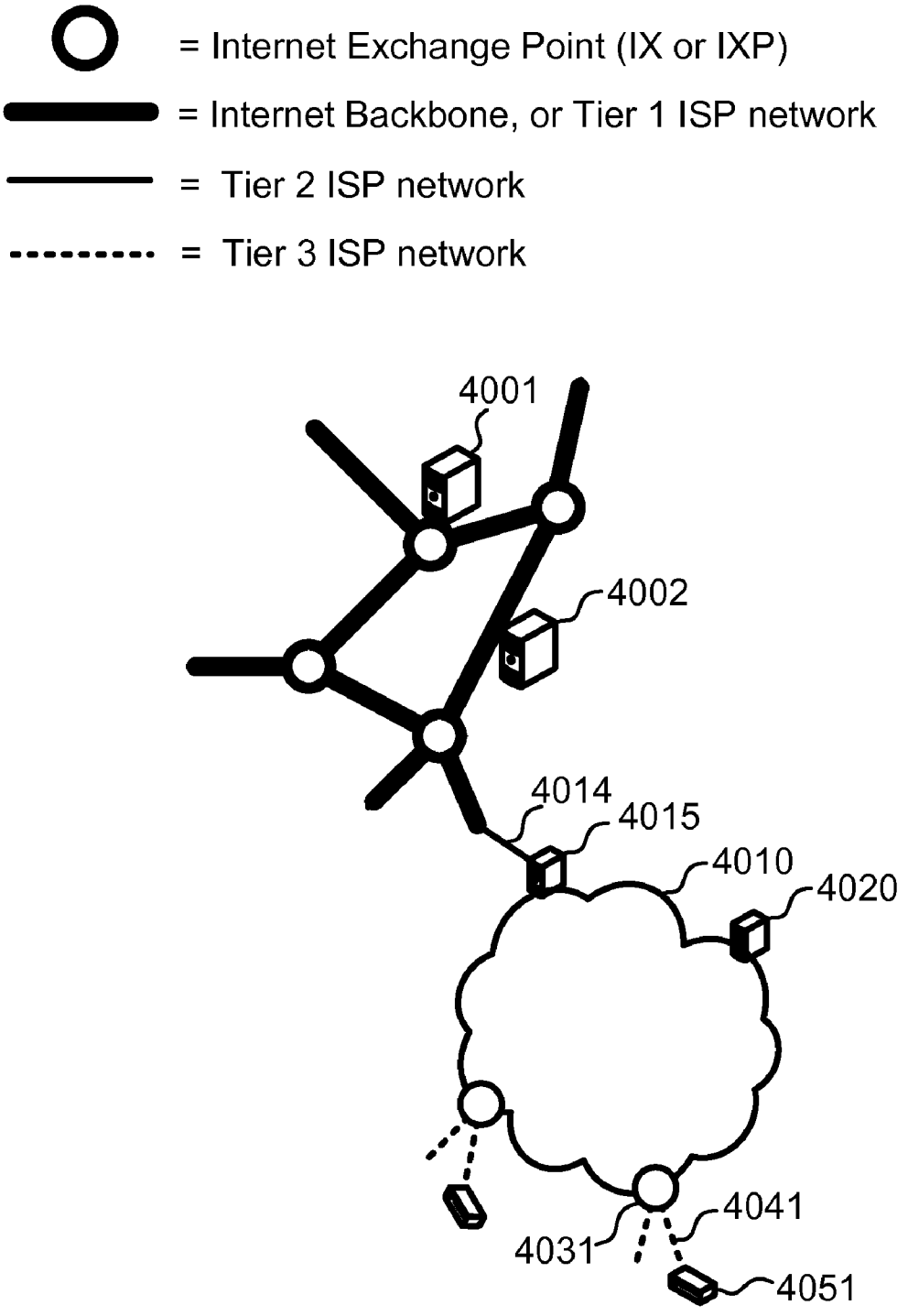

= Internet Exchange Point (IX or IXP)
= Internet Backbone, or Tier 1 ISP network
= Tier 2 ISP network
= Tier 3 ISP network
4001
4002
4014
4015
4010
4020
4041
4031
4051

FIG. 8

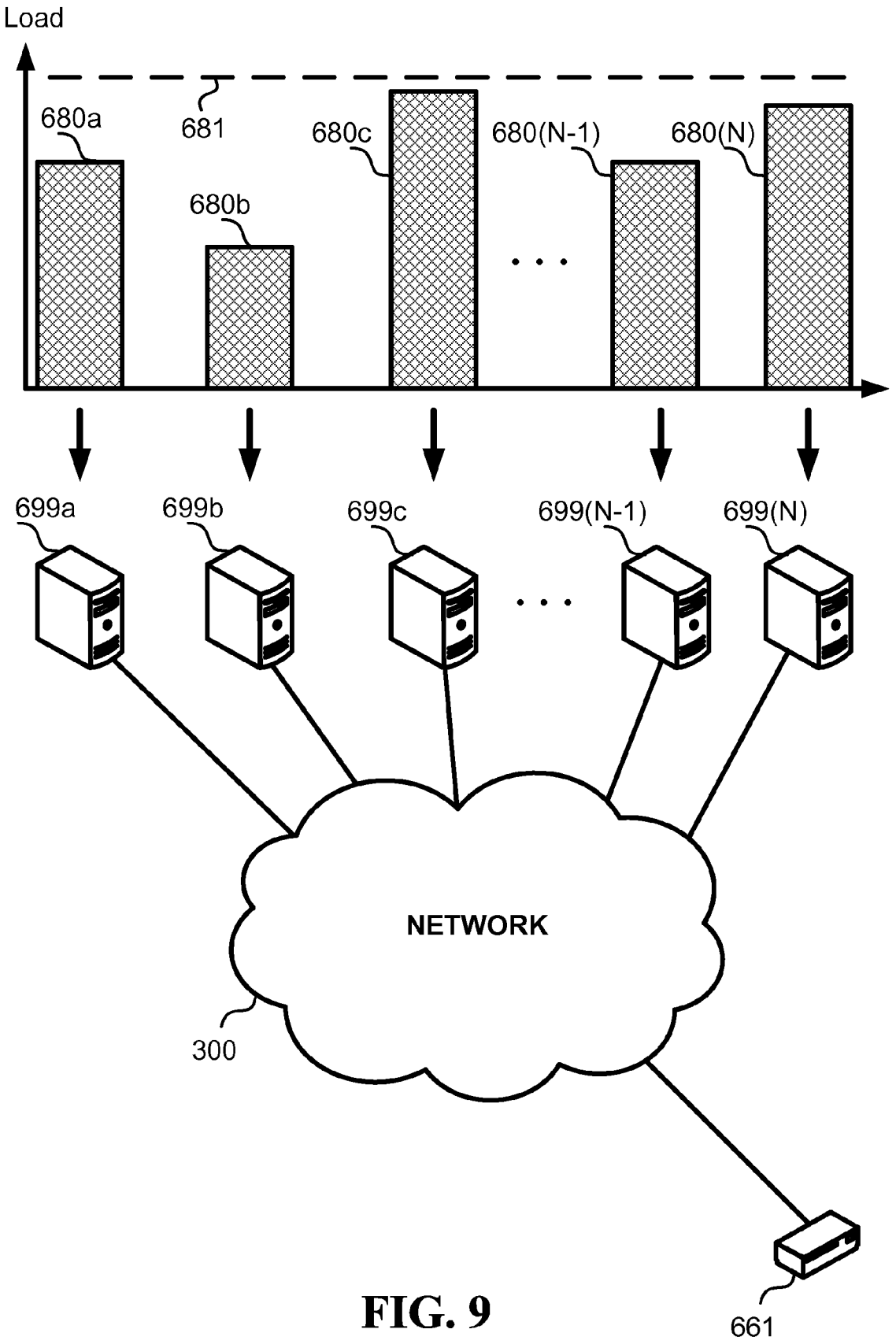

Load
680a
681
680c
680(N-1)
680(N)
680b
699a
699b
699c
699(N-1)
699(N)
NETWORK
300
661

FIG. 9

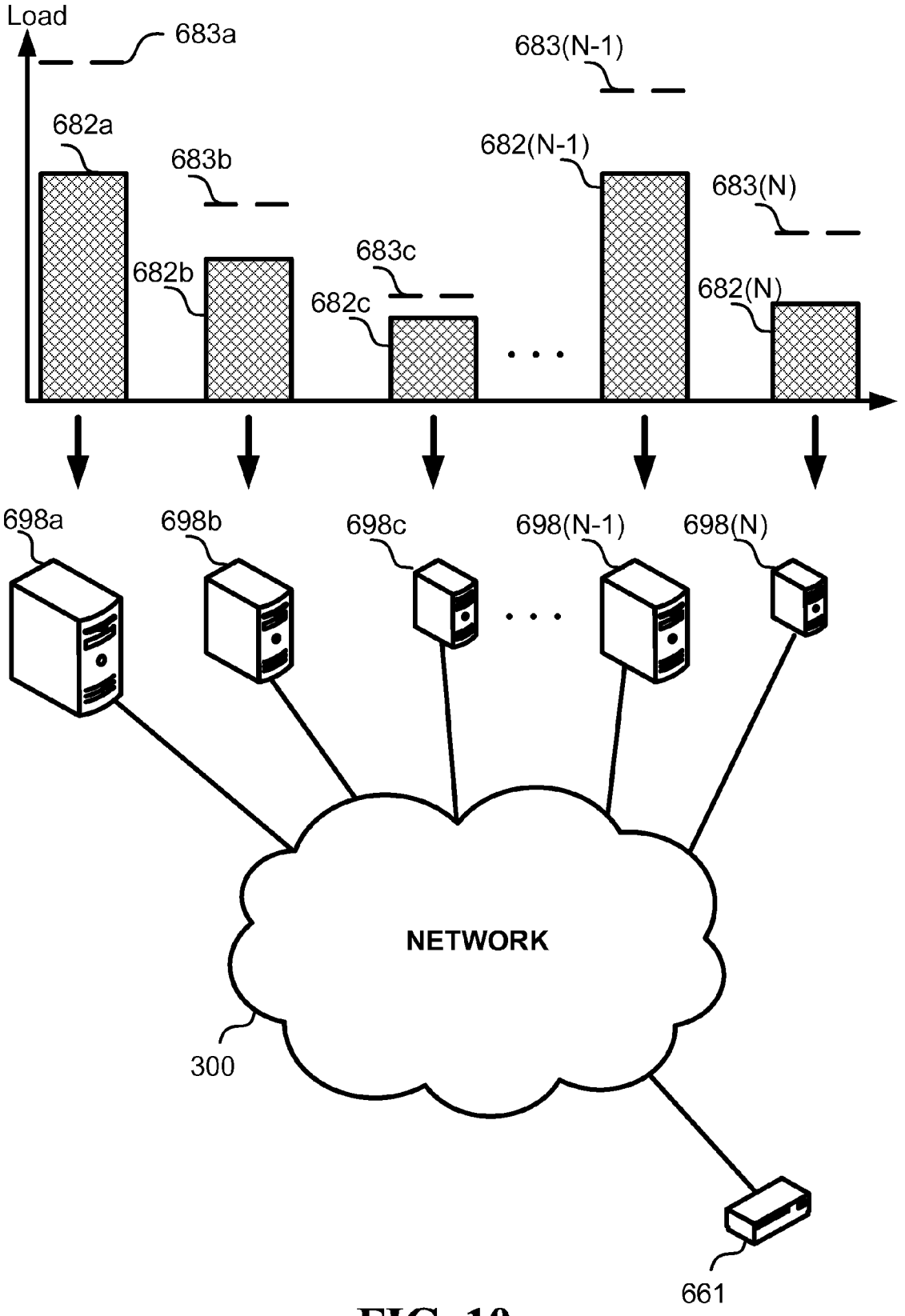

Load
683a
683(N-1)
682a
683b
682(N-1)
683(N)
683c
682b
682c
682(N)
698a
698b
698c
698(N-1)
698(N)
NETWORK
300
661

FIG. 10

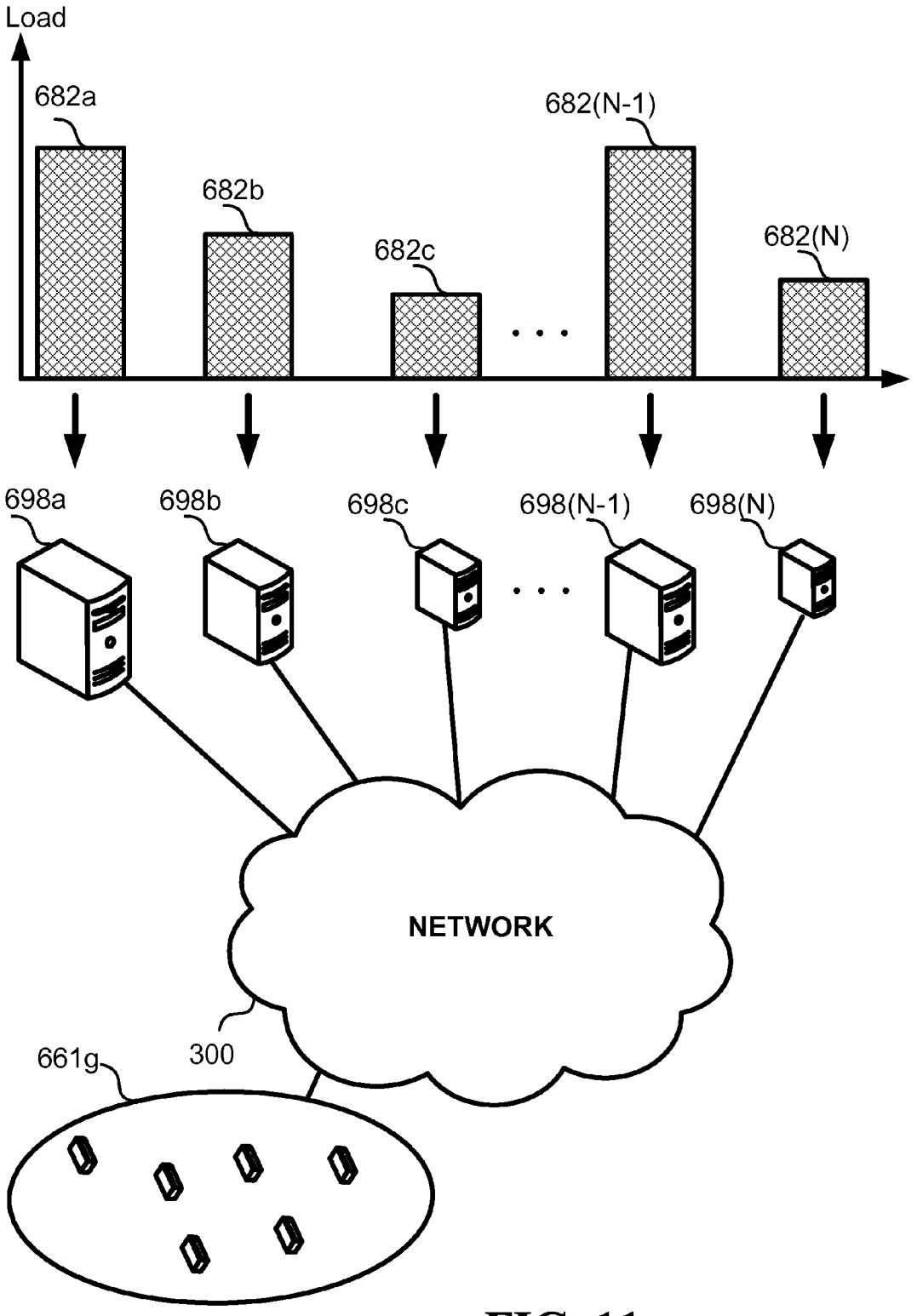

Load
682a
682b
682c
682(N-1)
682(N)
698a
698b
698c
698(N-1)
698(N)
NETWORK
300
661g

FIG. 11

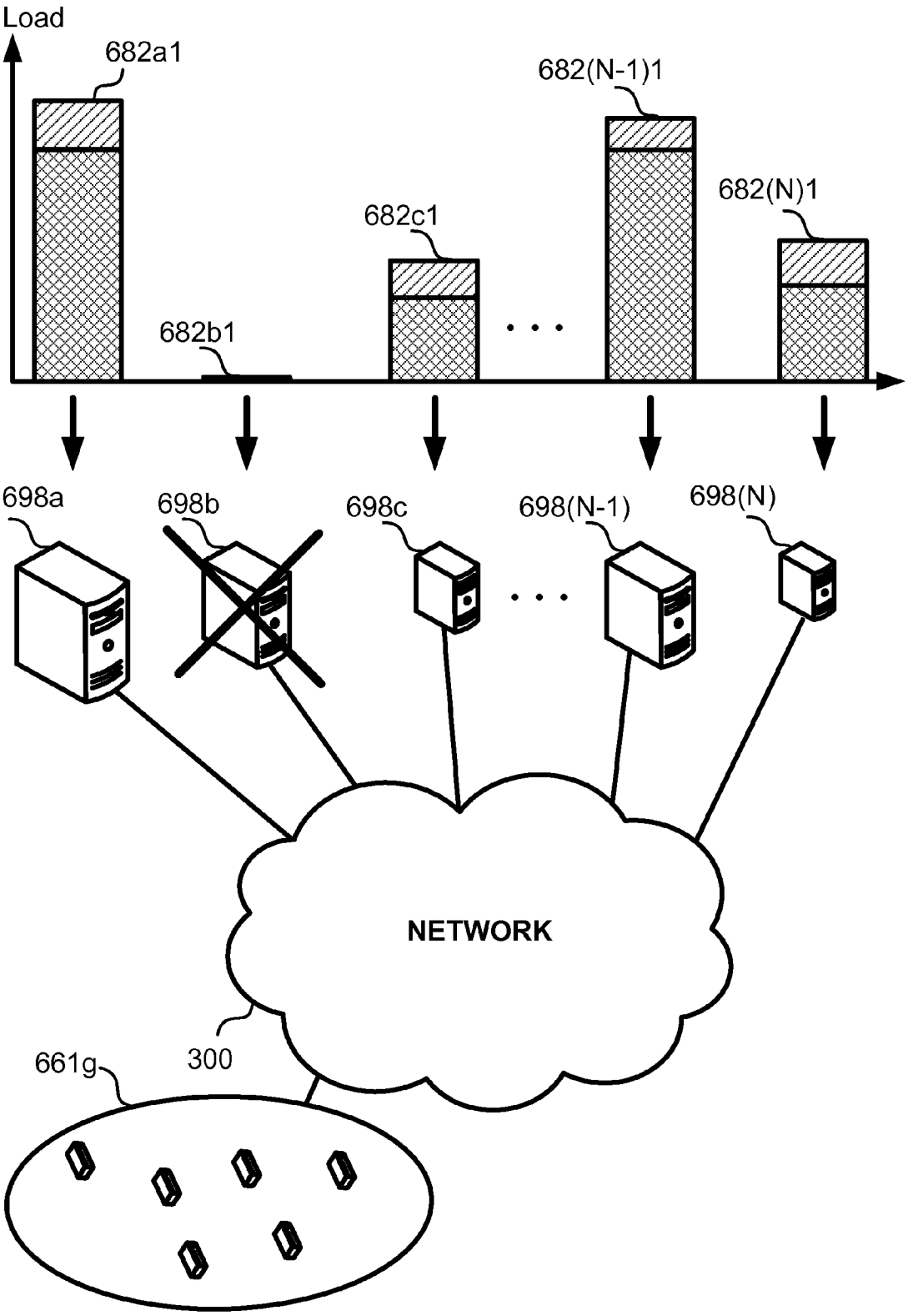

Load
682a1
682(N-1)1
682(N)1
682c1
682b1
698a
698b
698c
698(N-1)
698(N)
NETWORK
300
661g

FIG. 12

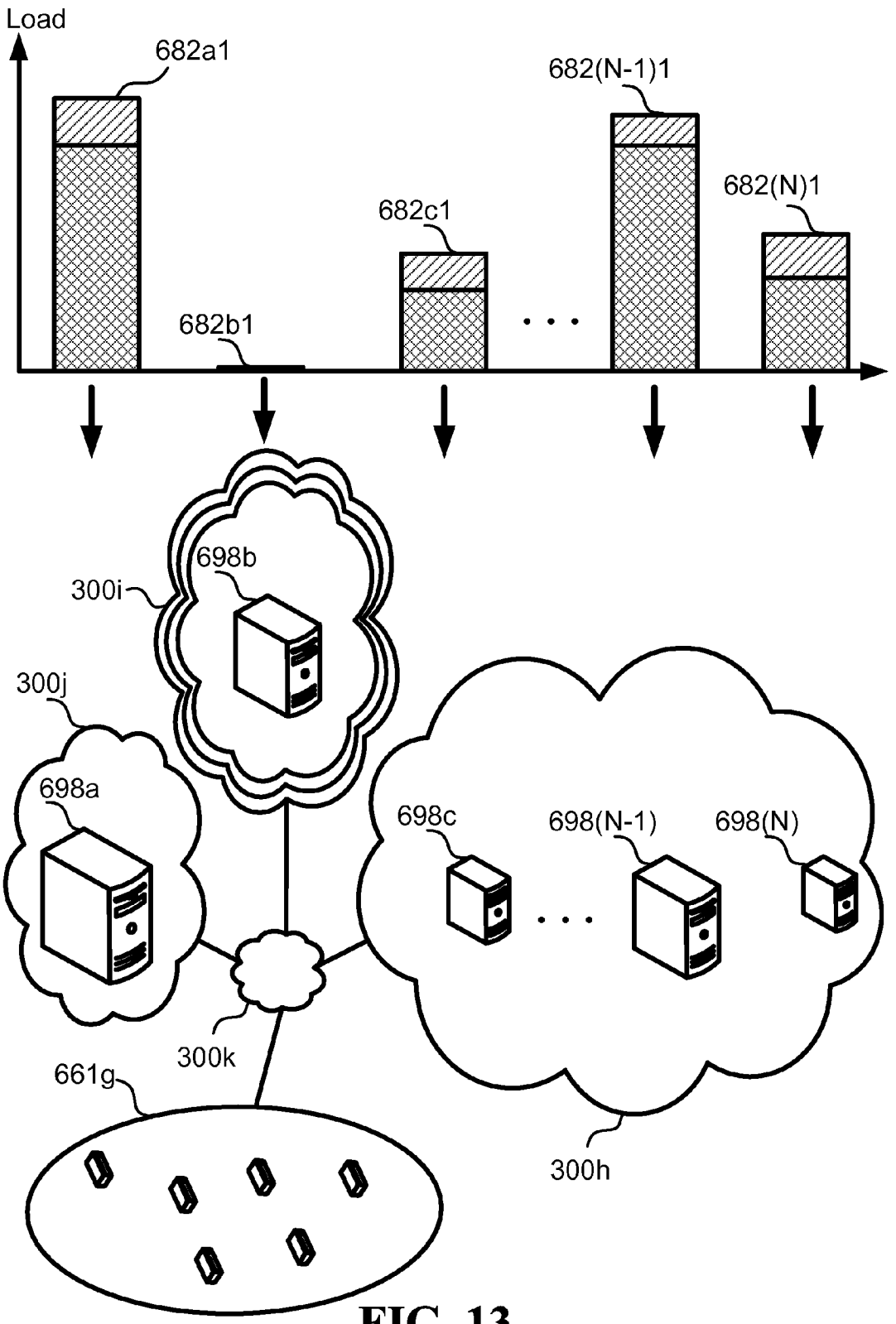

Load
682a1
682(N-1)1
682(N)1
682c1
682b1
698b
300i
300j
698a
698c
698(N-1)
698(N)
300k
661g
300h

FIG. 13

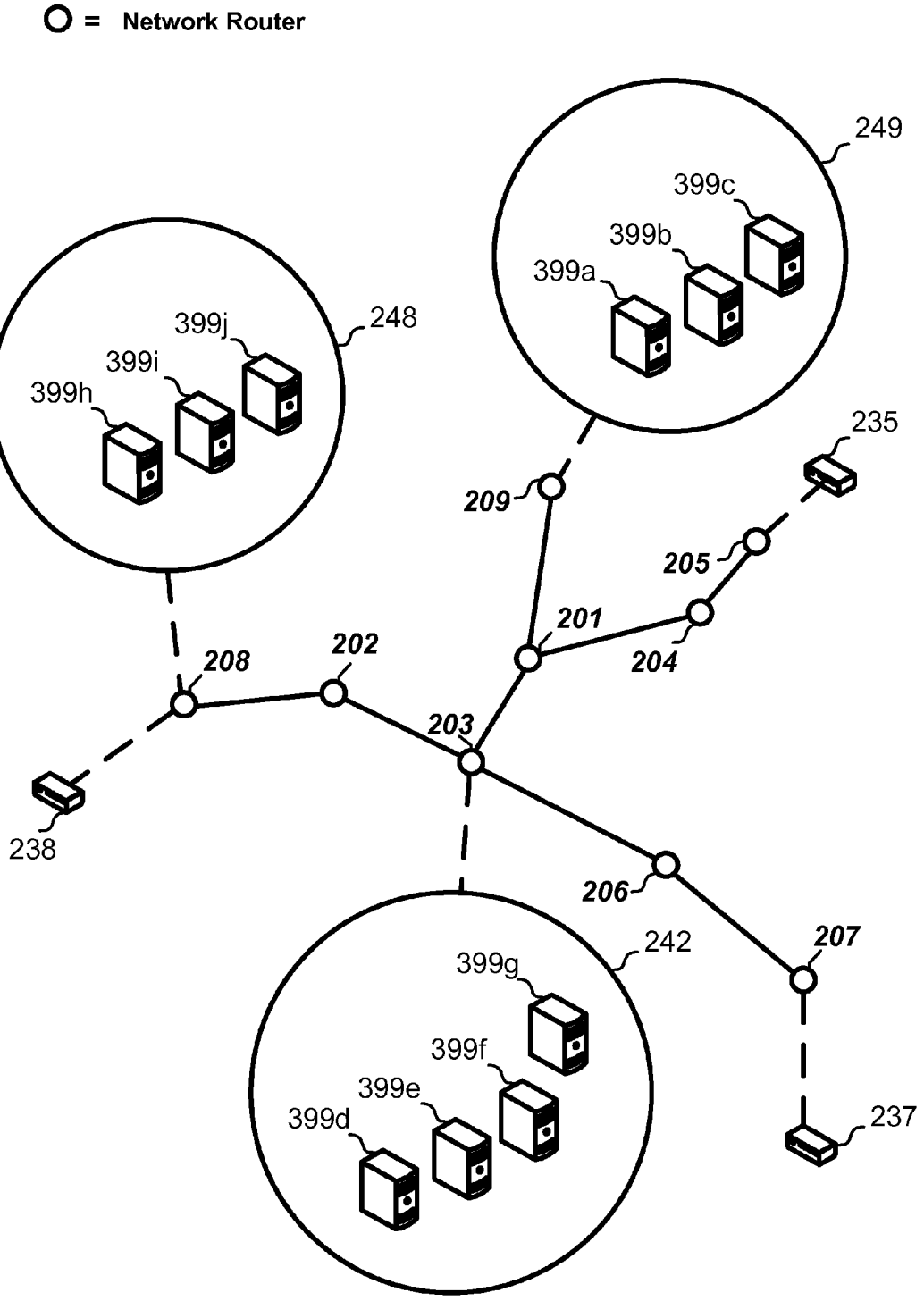

= Network Router
249
399c
399b
399a
248
399j
399i
399h
235
209
205
201
204
208
202
203
238
206
207
242
399g
399f
399e
399d
237

FIG. 14

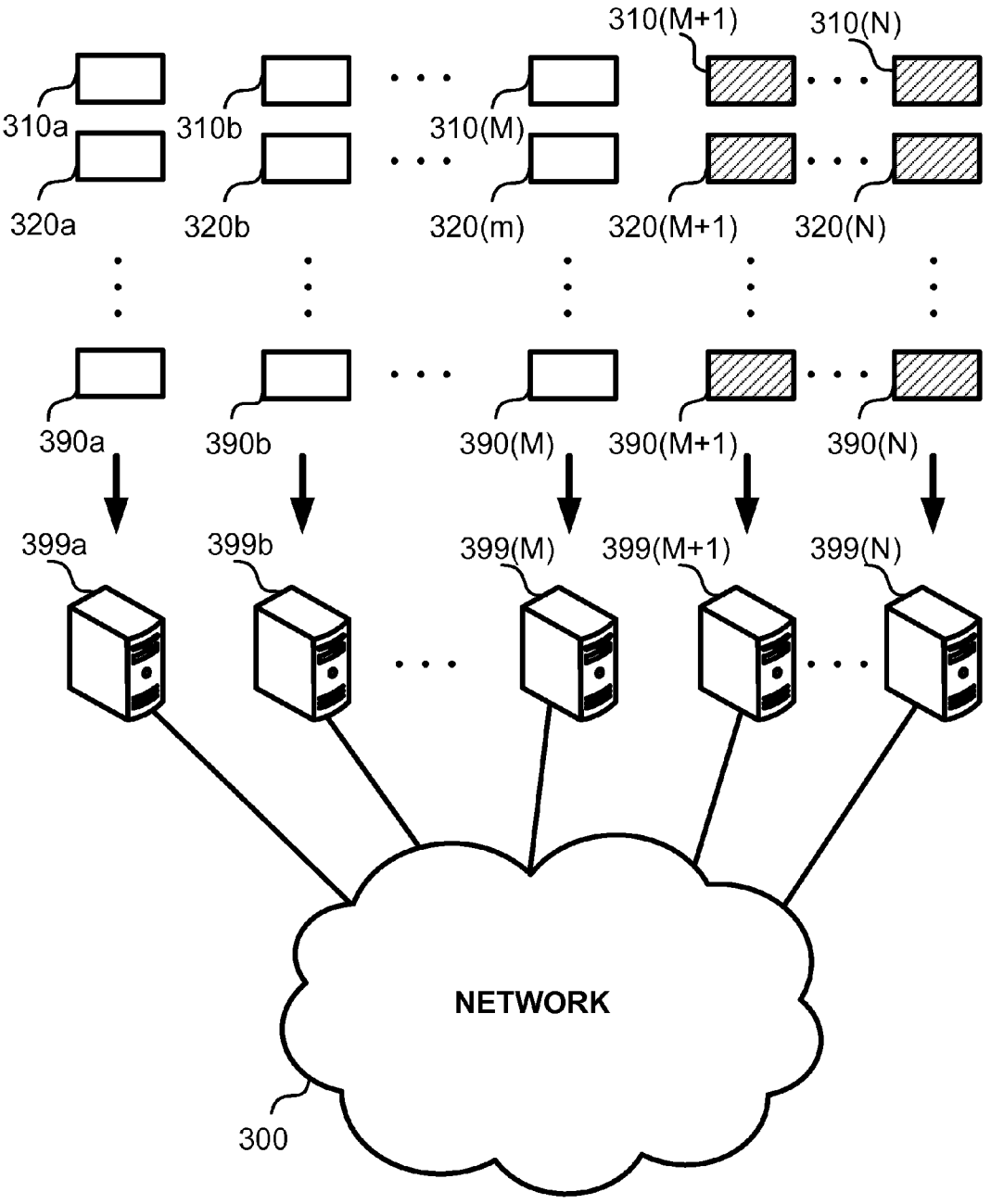

310(M+1)
310(N)
310a
310b
310(M)
320a
320b
320(m)
320(M+1)
320(N)
390a
390b
390(M)
390(M+1)
390(N)
399a
399b
399(M)
399(M+1)
399(N)
NETWORK
300

FIG. 15

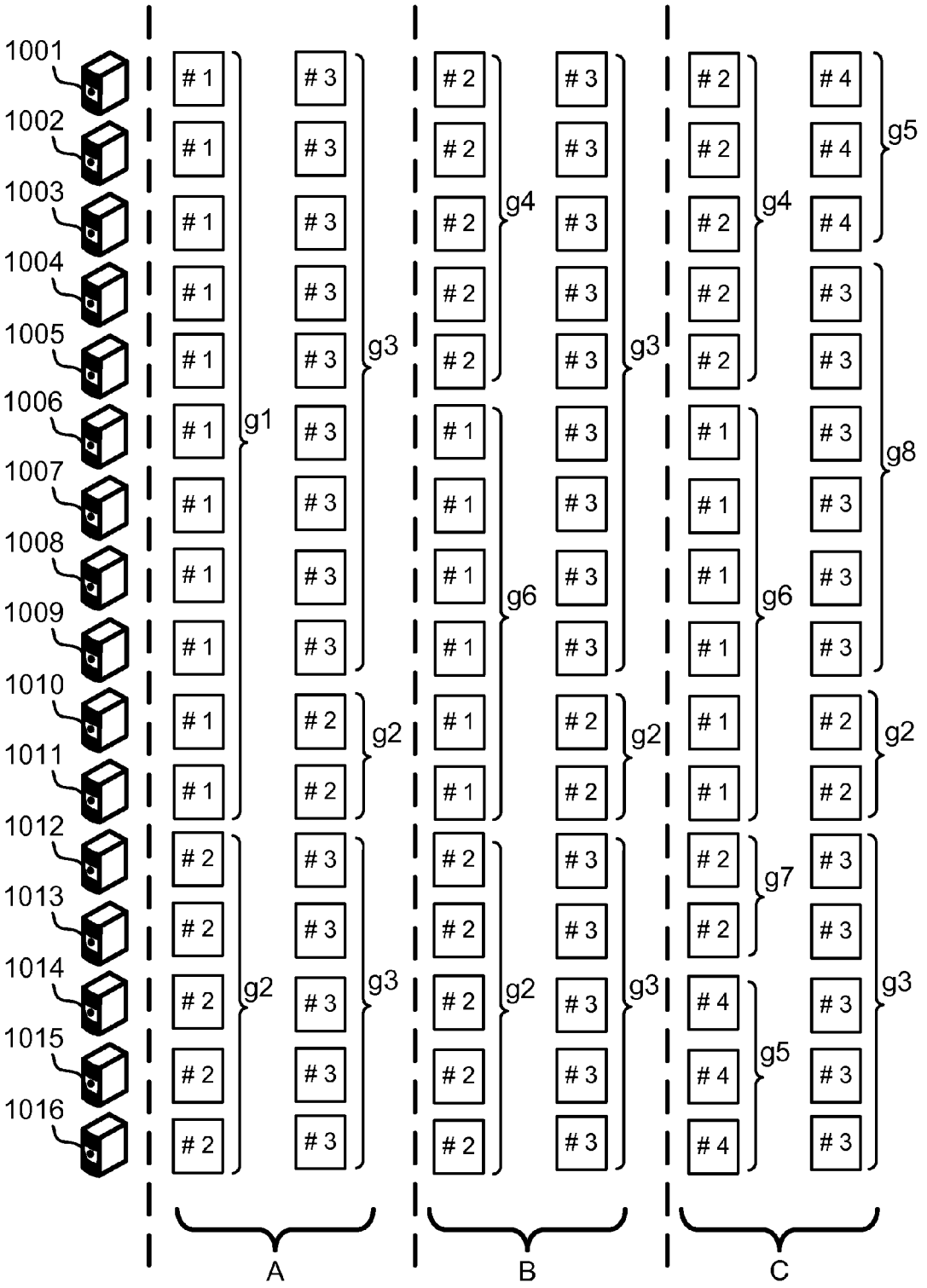

1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
A
B
C
g1
g2
g3
g4
g5
g6
g7
g8

FIG. 16

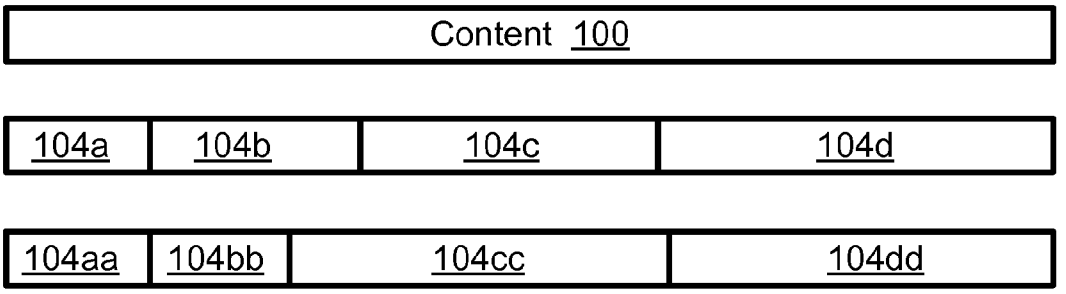

Content 100
104a
104b
104c
104d
104aa
104bb
104cc
104dd

FIG. 17

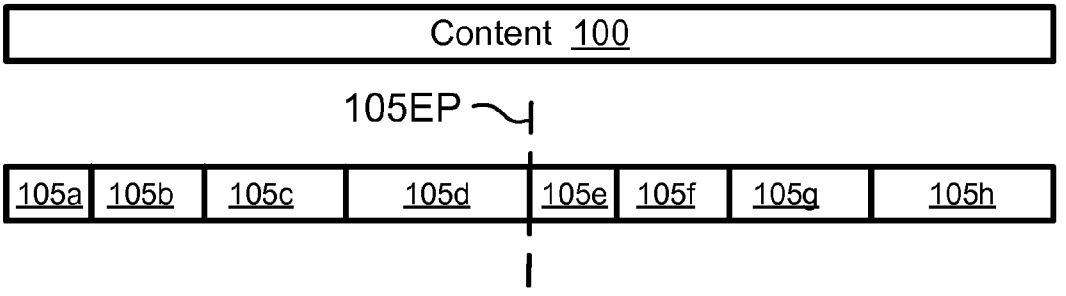

Content 100
105EP
105a
105b
105c
105d
105e
105f
105g
105h

FIG. 18

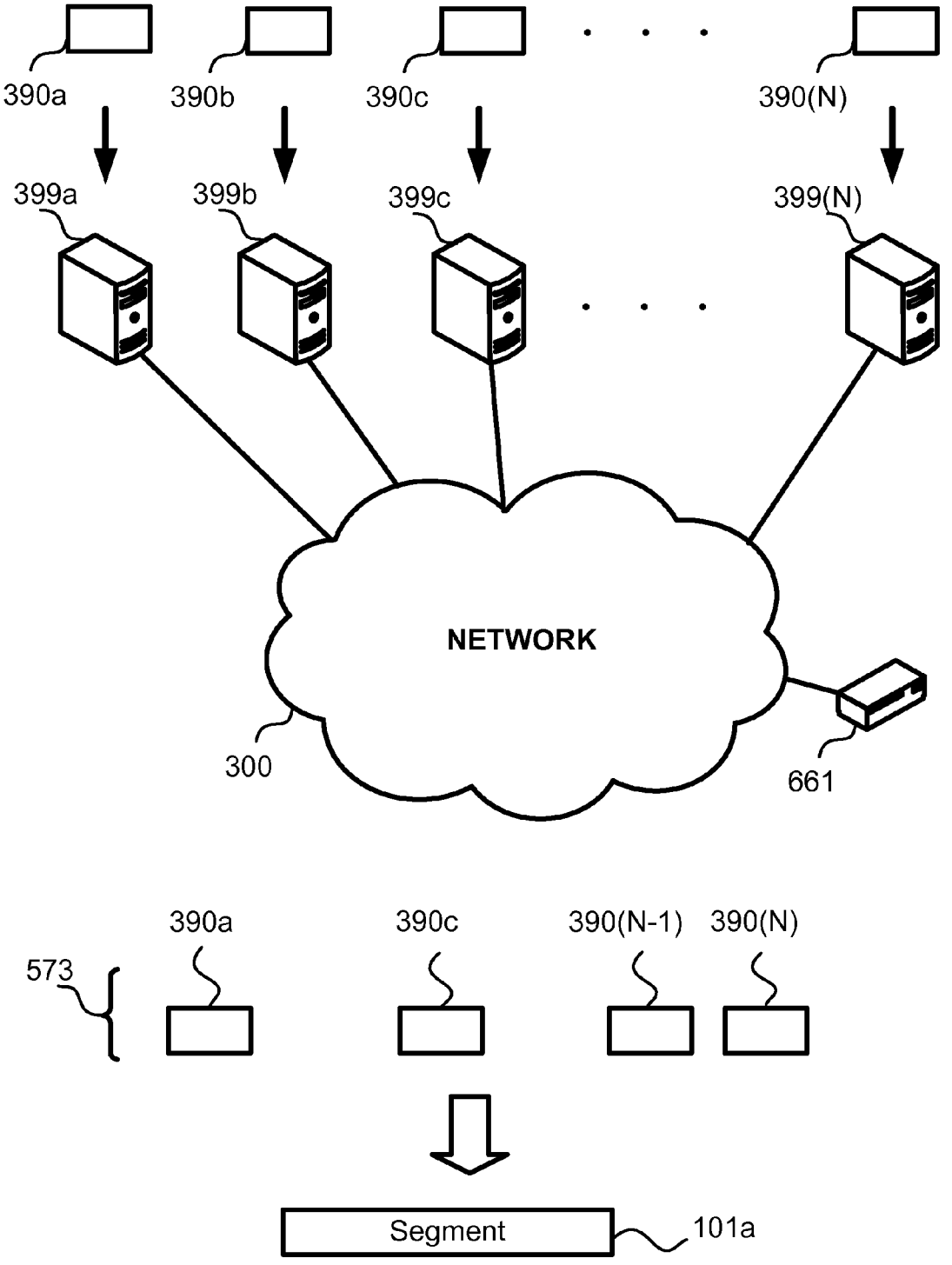

390a
390b
390c
390(N)
399a
399b
399c
399(N)
NETWORK
300
661
390a
390c
390(N-1)
390(N)
573
Segment
101a

FIG. 19

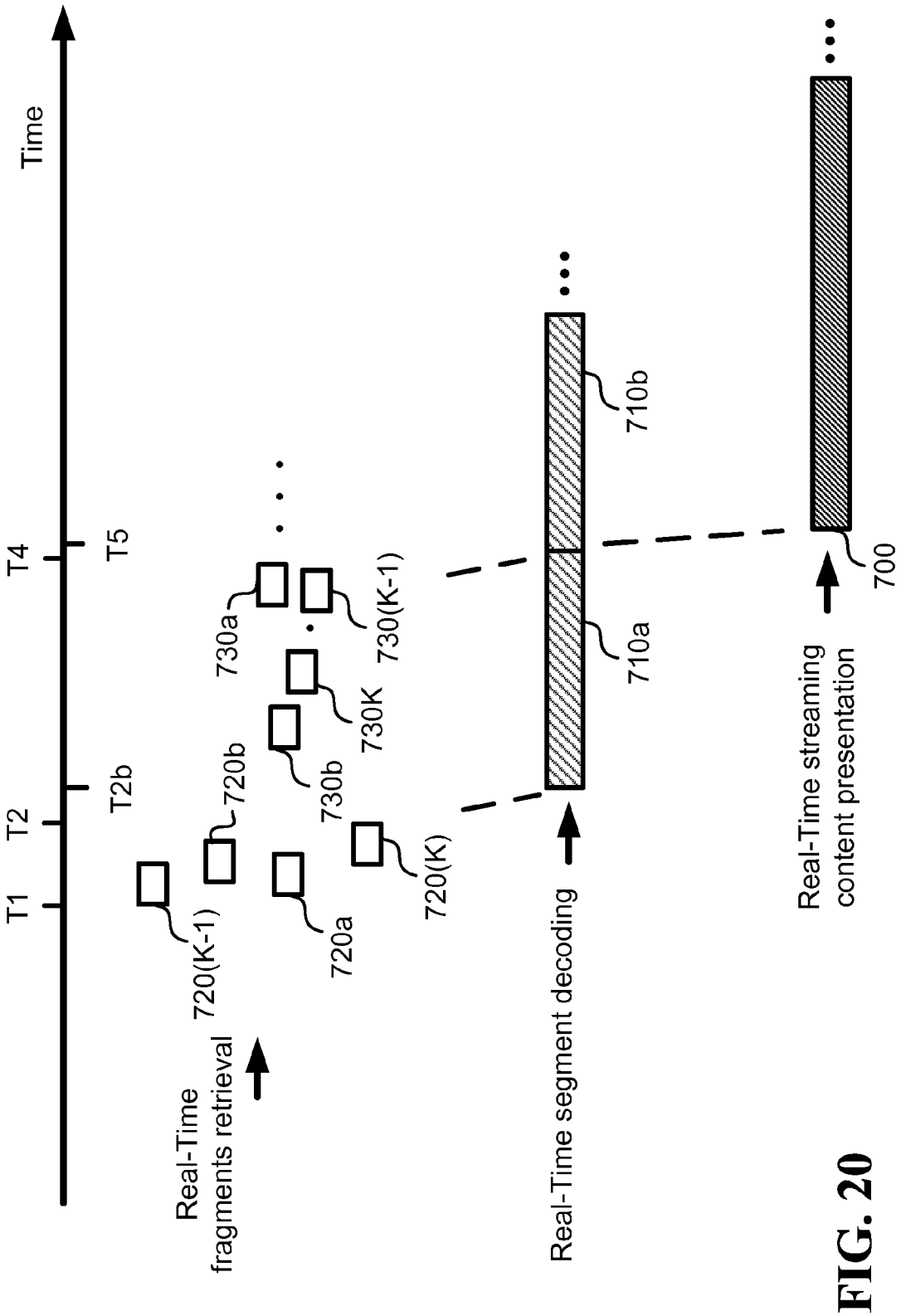

Time
T1
T2
T2b
T4
T5
Real-Time fragments retrieval
720(K-1)
720a
720b
730b
730K
730a
730(K-1)
720(K)
Real-Time segment decoding
710a
710b
Real-Time streaming content presentation
700

FIG. 20

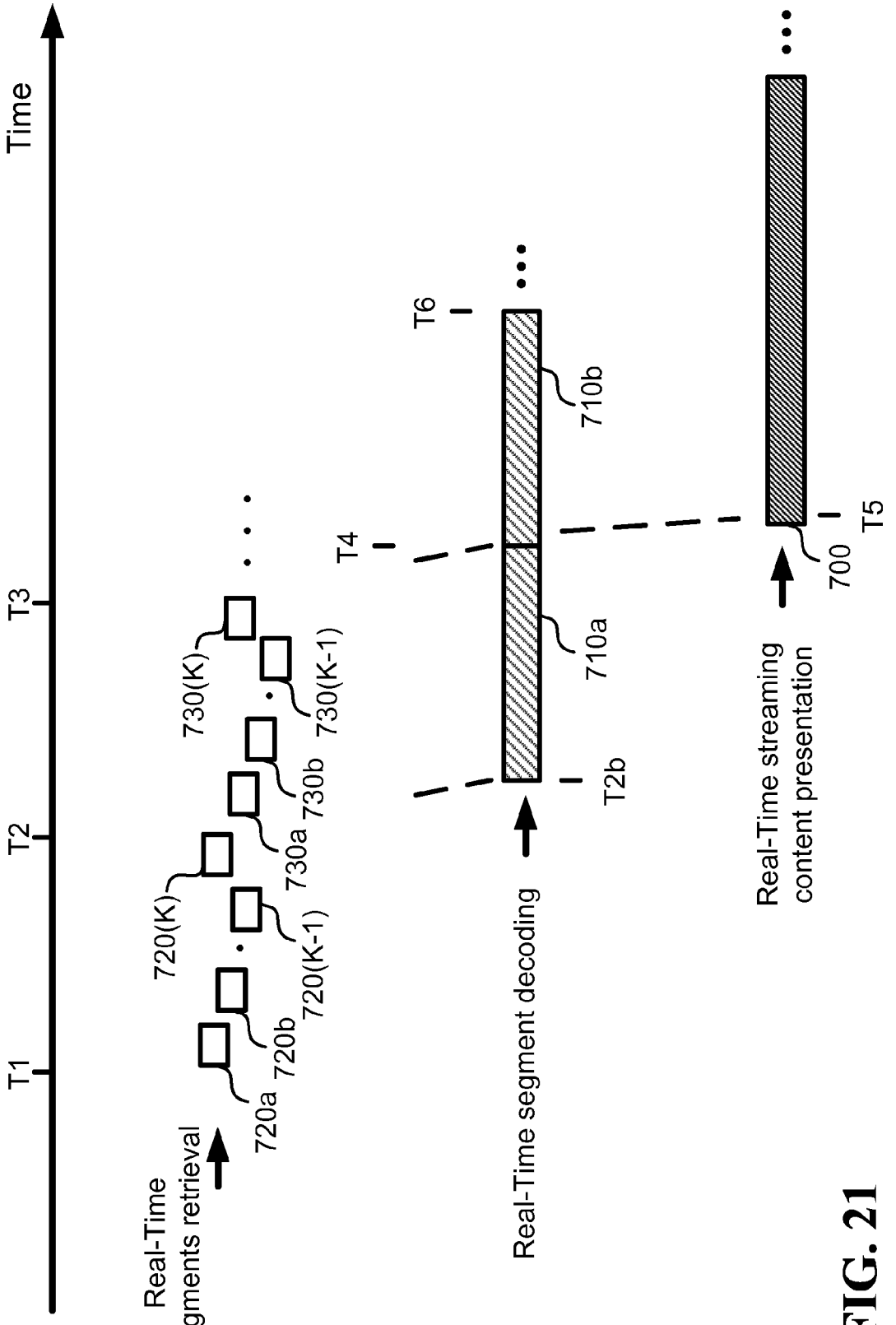

Time
T1
T2
T3
T4
T5
T6
T2b
Real-Time fragments retrieval
720a
720b
720(K-1)
720(K)
730a
730b
730(K-1)
730(K)
Real-Time segment decoding
710a
710b
Real-Time streaming content presentation
700

FIG. 21

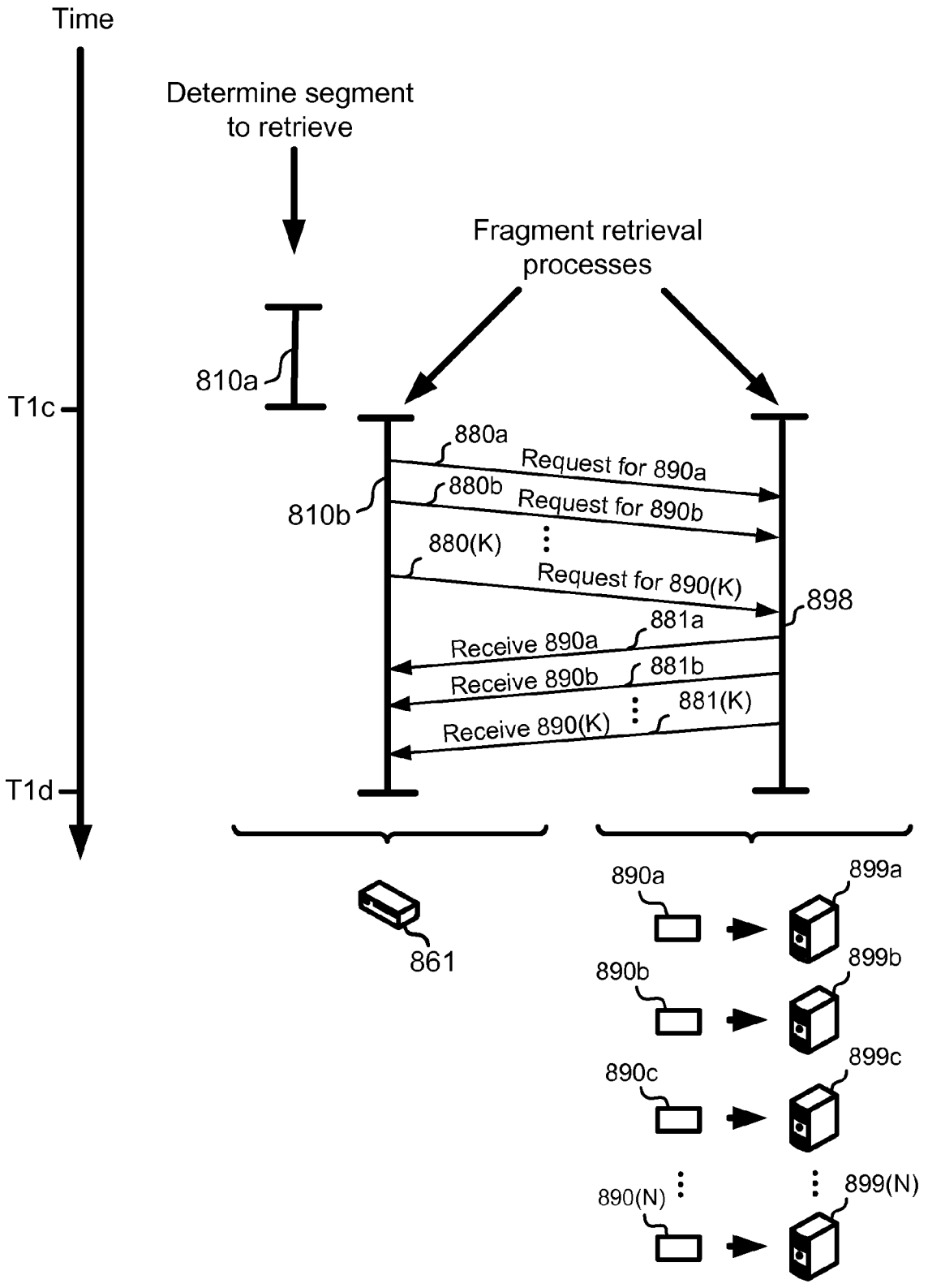

Time
Determine segment to retrieve
Fragment retrieval processes
810a
T1c
880a
Request for 890a
880b
Request for 890b
810b
880(K)
Request for 890(K)
898
881a
Receive 890a
881b
Receive 890b
881(K)
Receive 890(K)
T1d
861
890a
899a
890b
899b
890c
899c
890(N)
899(N)

FIG. 22

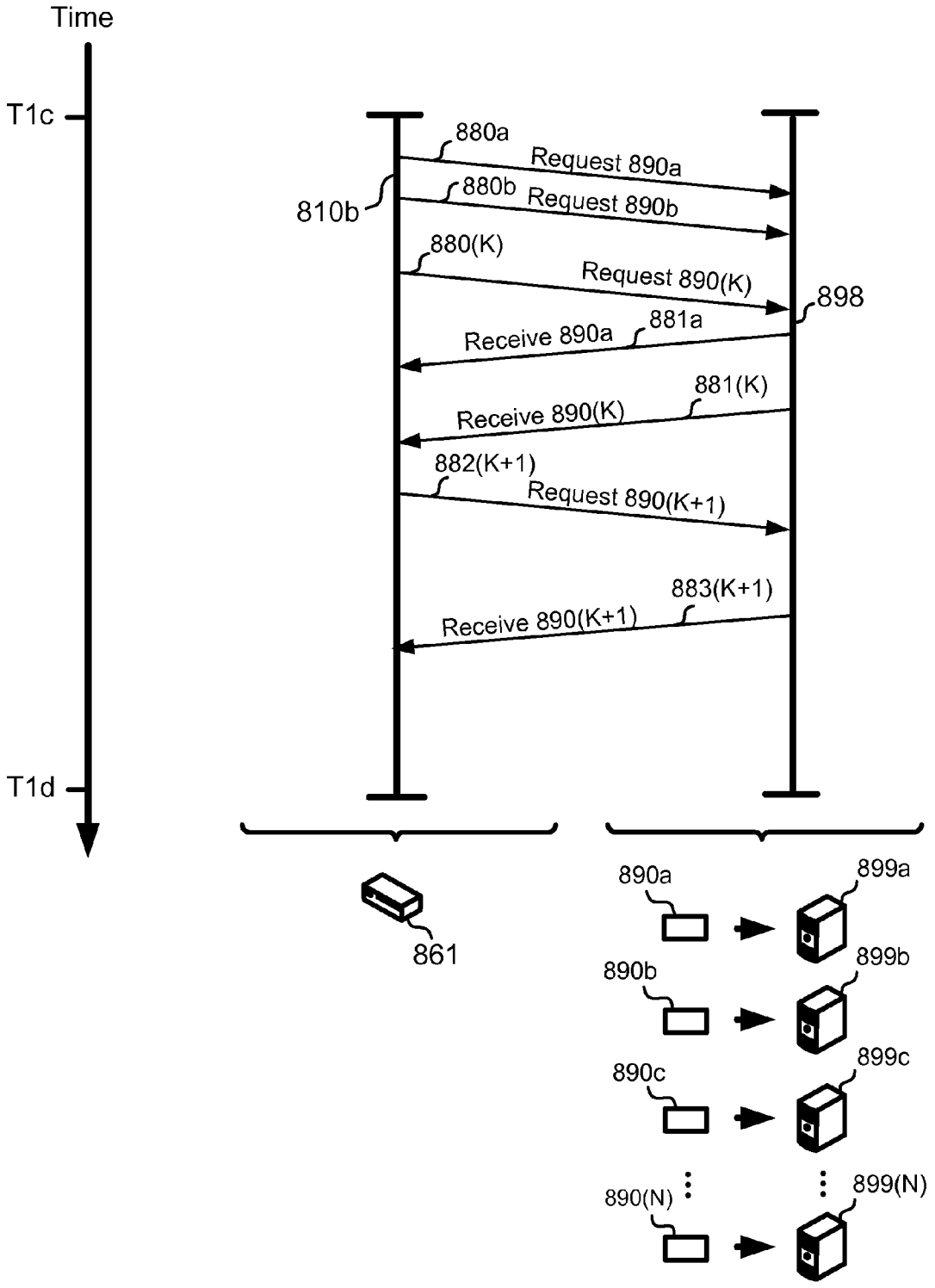

Time
T1c
T1d
880a
Request 890a
880b
Request 890b
810b
880(K)
Request 890(K)
898
881a
Receive 890a
881(K)
Receive 890(K)
882(K+1)
Request 890(K+1)
883(K+1)
Receive 890(K+1)
861
890a
899a
890b
899b
890c
899c
890(N)
899(N)

FIG. 23

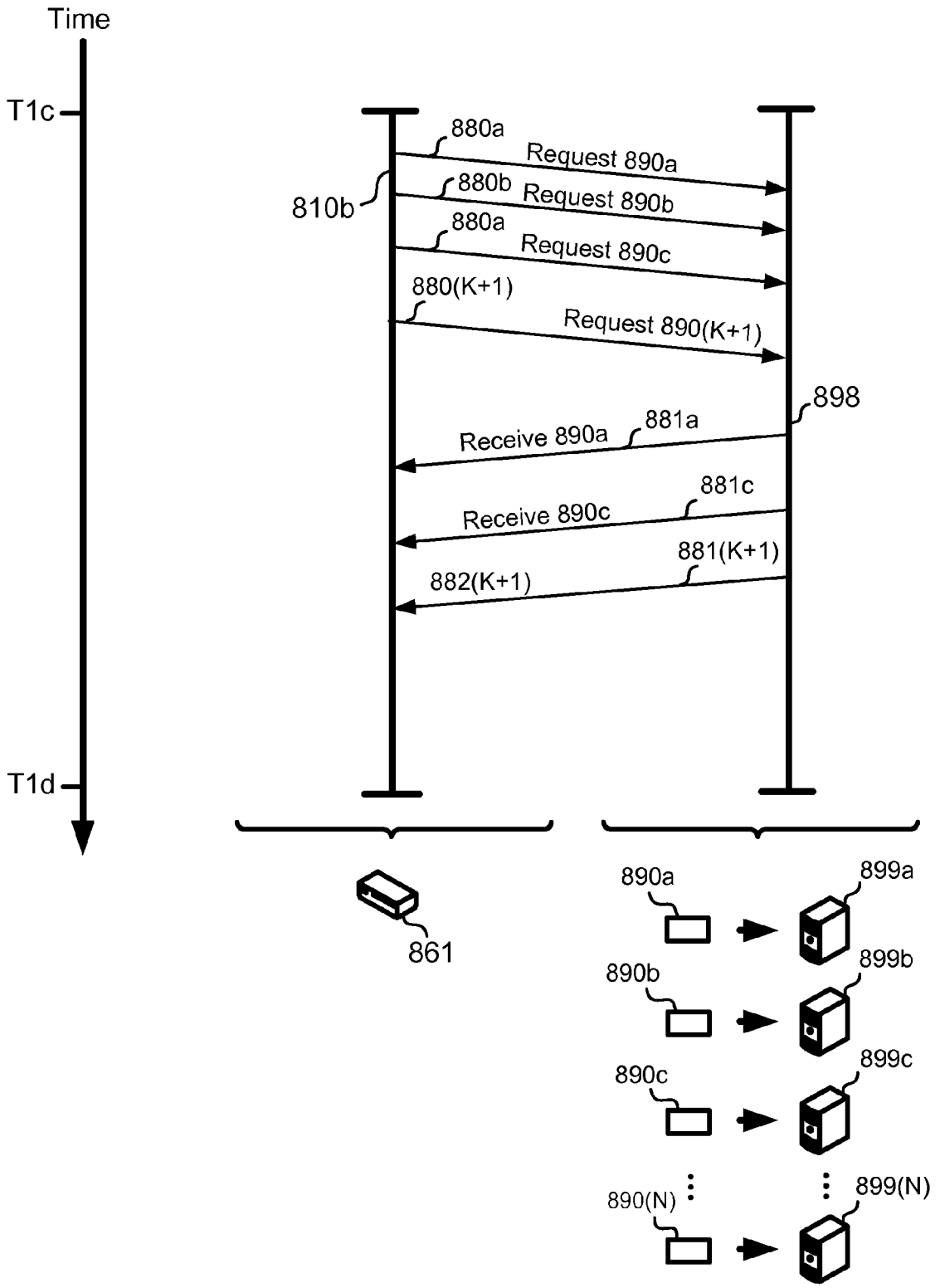

Time
T1c
T1d
880a
Request 890a
880b
Request 890b
880a
Request 890c
880(K+1)
Request 890(K+1)
810b
898
881a
Receive 890a
881c
Receive 890c
881(K+1)
882(K+1)
861
890a
899a
890b
899b
890c
899c
890(N)
899(N)

FIG. 24

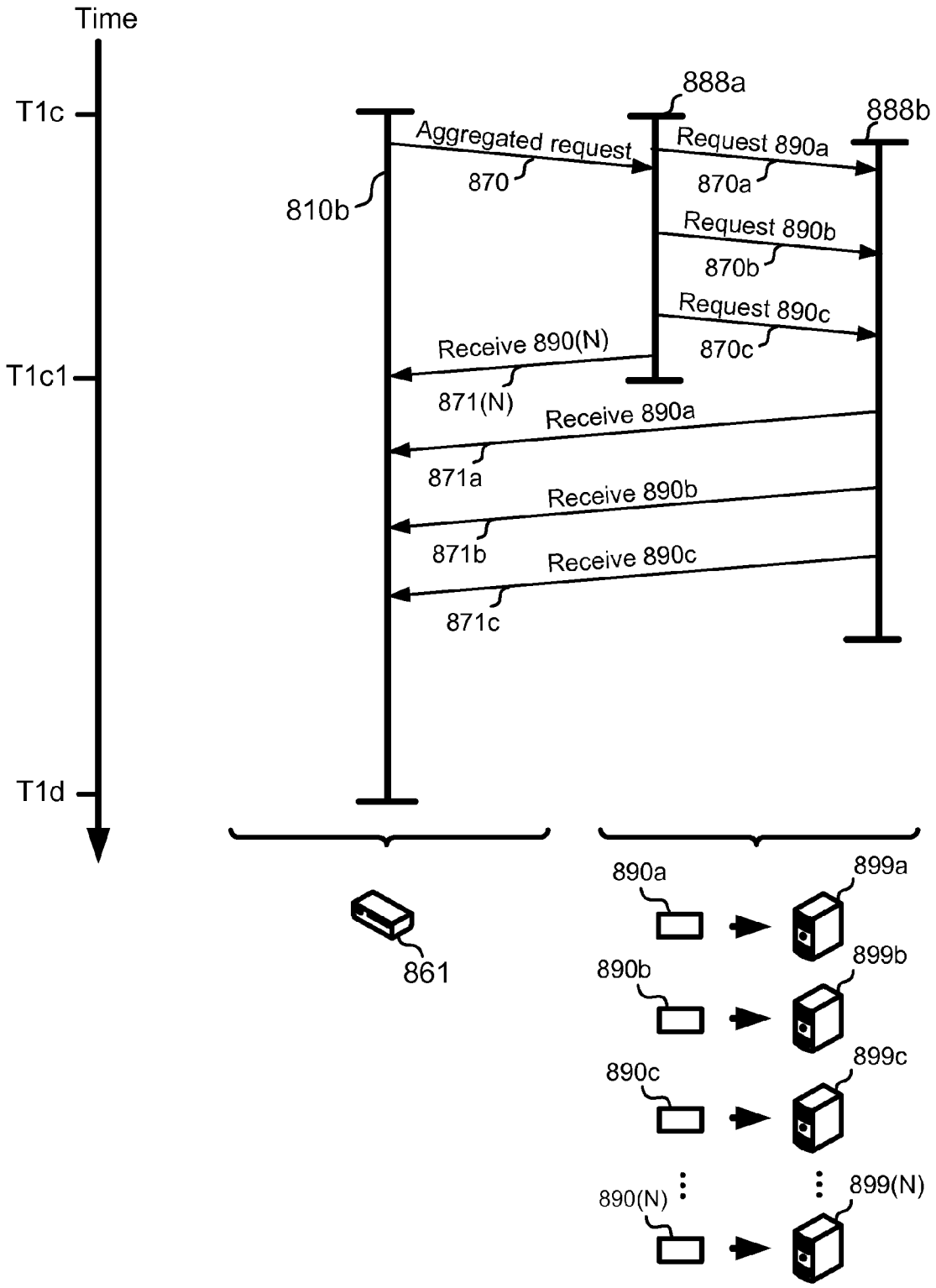

Time
T1c
T1c1
T1d
888a
888b
Aggregated request
870
Request 890a
870a
Request 890b
870b
Request 890c
870c
810b
Receive 890(N)
871(N)
Receive 890a
871a
Receive 890b
871b
Receive 890c
871c
861
890a
899a
890b
899b
890c
899c
890(N)
899(N)

FIG. 25

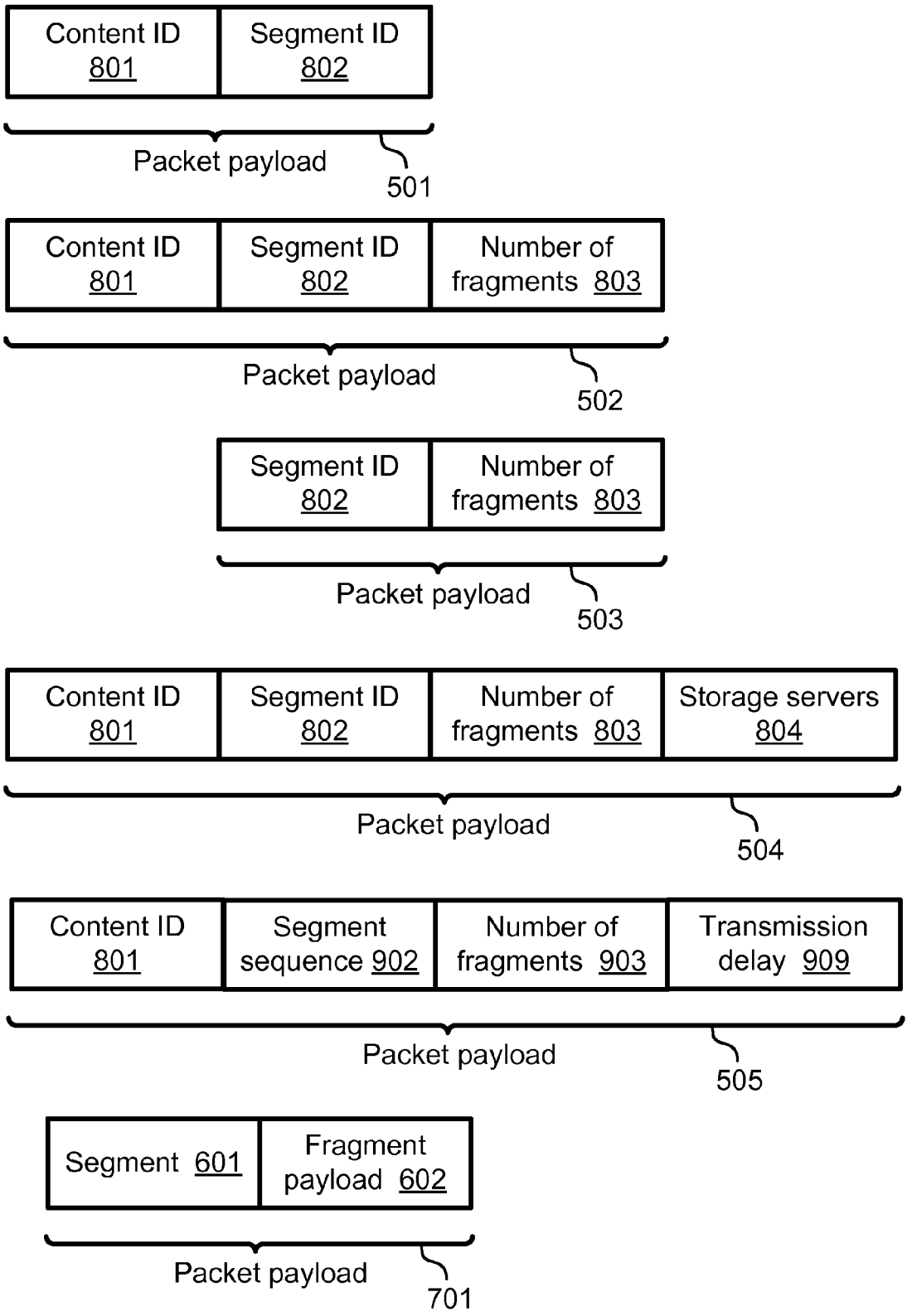

Content ID 801
Segment ID 802
Packet payload
501
Content ID 801
Segment ID 802
Number of fragments 803
Packet payload
502
Segment ID 802
Number of fragments 803
Packet payload
503
Content ID 801
Segment ID 802
Number of fragments 803
Storage servers 804
Packet payload
504
Content ID 801
Segment sequence 902
Number of fragments 903
Transmission delay 909
Packet payload
505
Segment 601
Fragment payload 602
Packet payload
701

FIG. 26

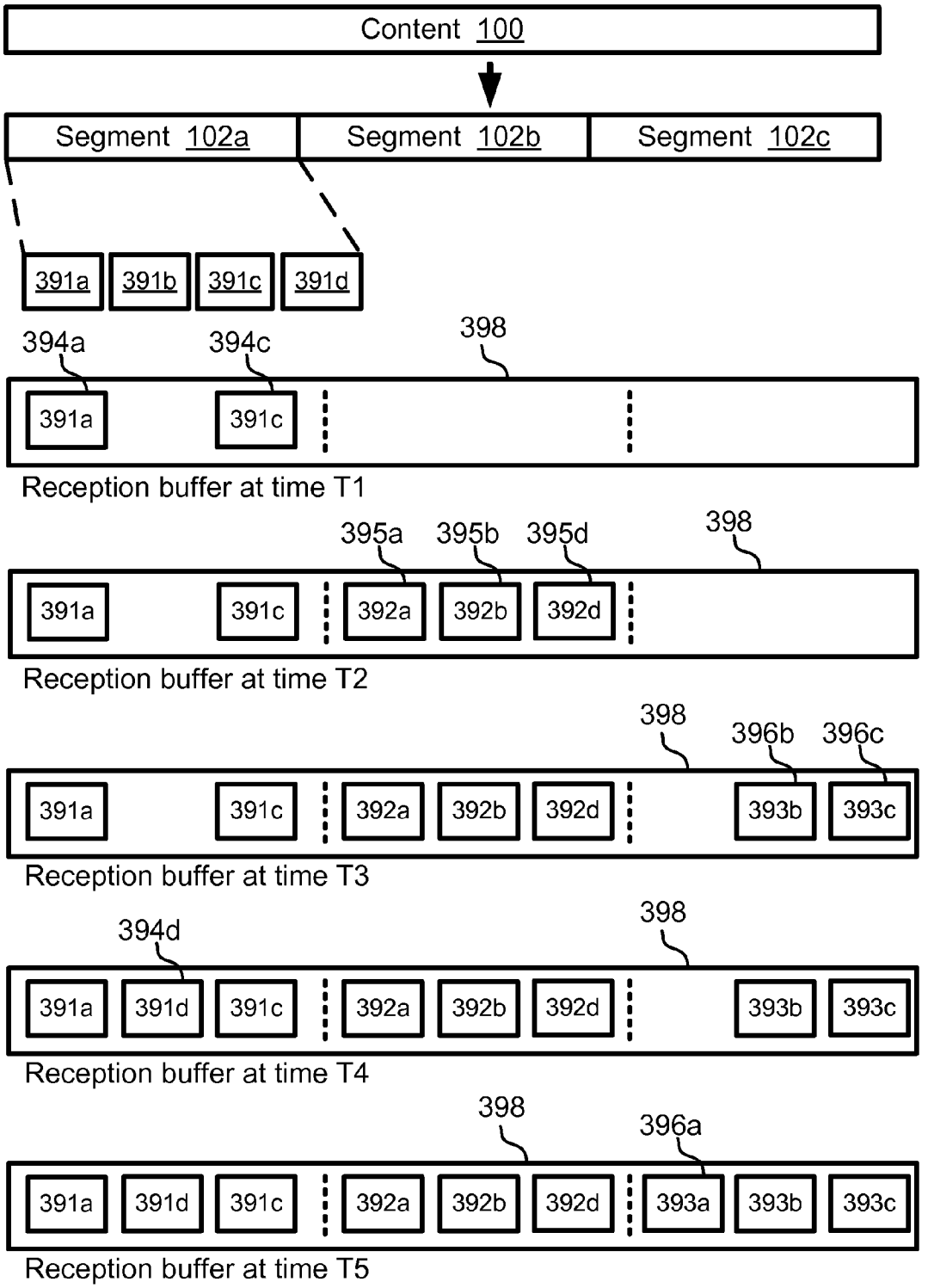

Content 100
Segment 102a
Segment 102b
Segment 102c
391a
391b
391c
391d
394a
394c
398
391a
391c
Reception buffer at time T1
395a
395b
395d
398
391a
391c
392a
392b
392d
Reception buffer at time T2
398
396b
396c
391a
391c
392a
392b
392d
393b
393c
Reception buffer at time T3
394d
398
391a
391d
391c
392a
392b
392d
393b
393c
Reception buffer at time T4
398
396a
391a
391d
391c
392a
392b
392d
393a
393b
393c
Reception buffer at time T5

FIG. 27

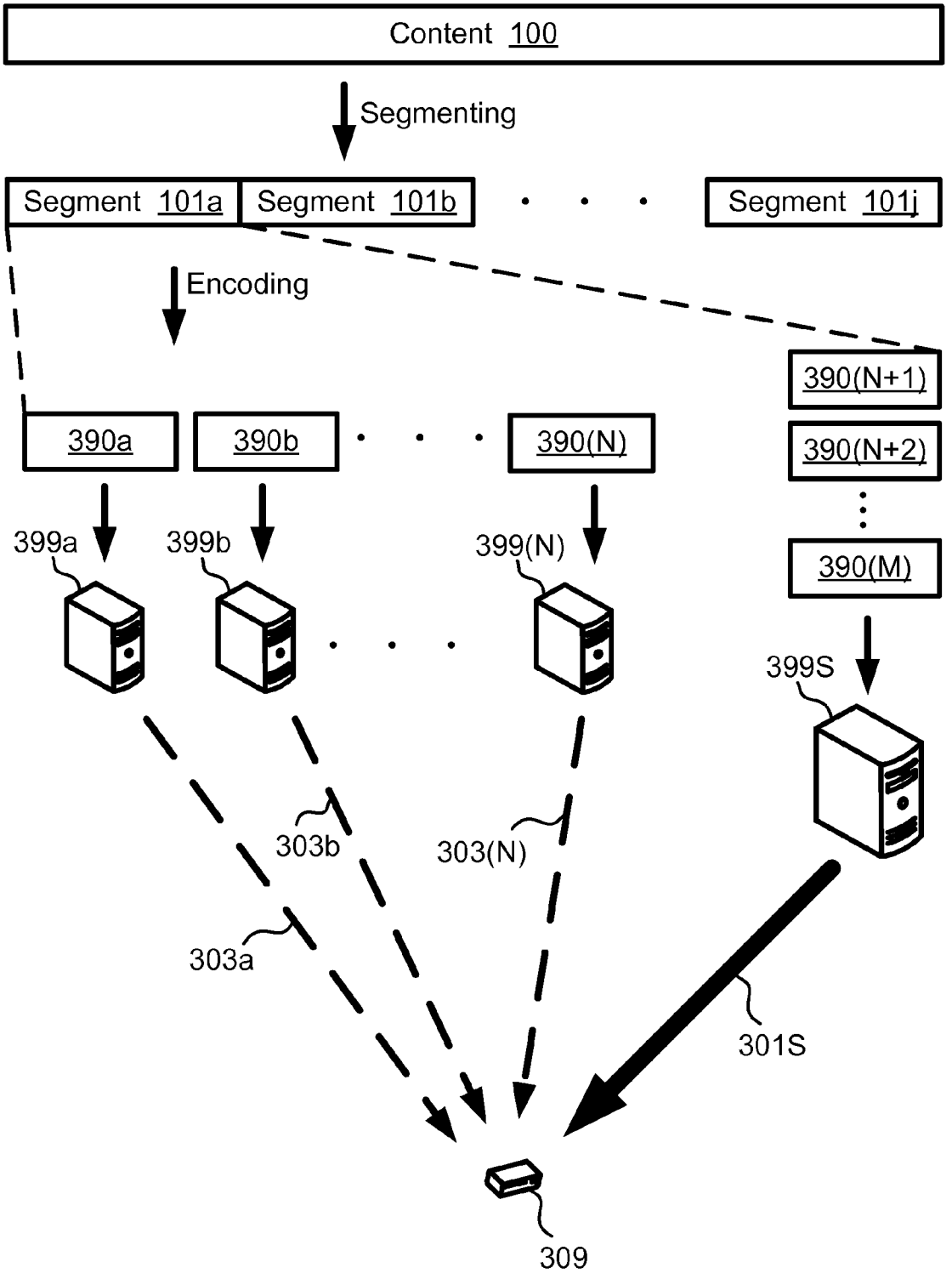

Content 100
Segmenting
Segment 101a
Segment 101b
Segment 101j
Encoding
390a
390b
390(N)
390(N+1)
390(N+2)
390(M)
399a
399b
399(N)
399S
303b
303(N)
303a
301S
309

FIG. 28

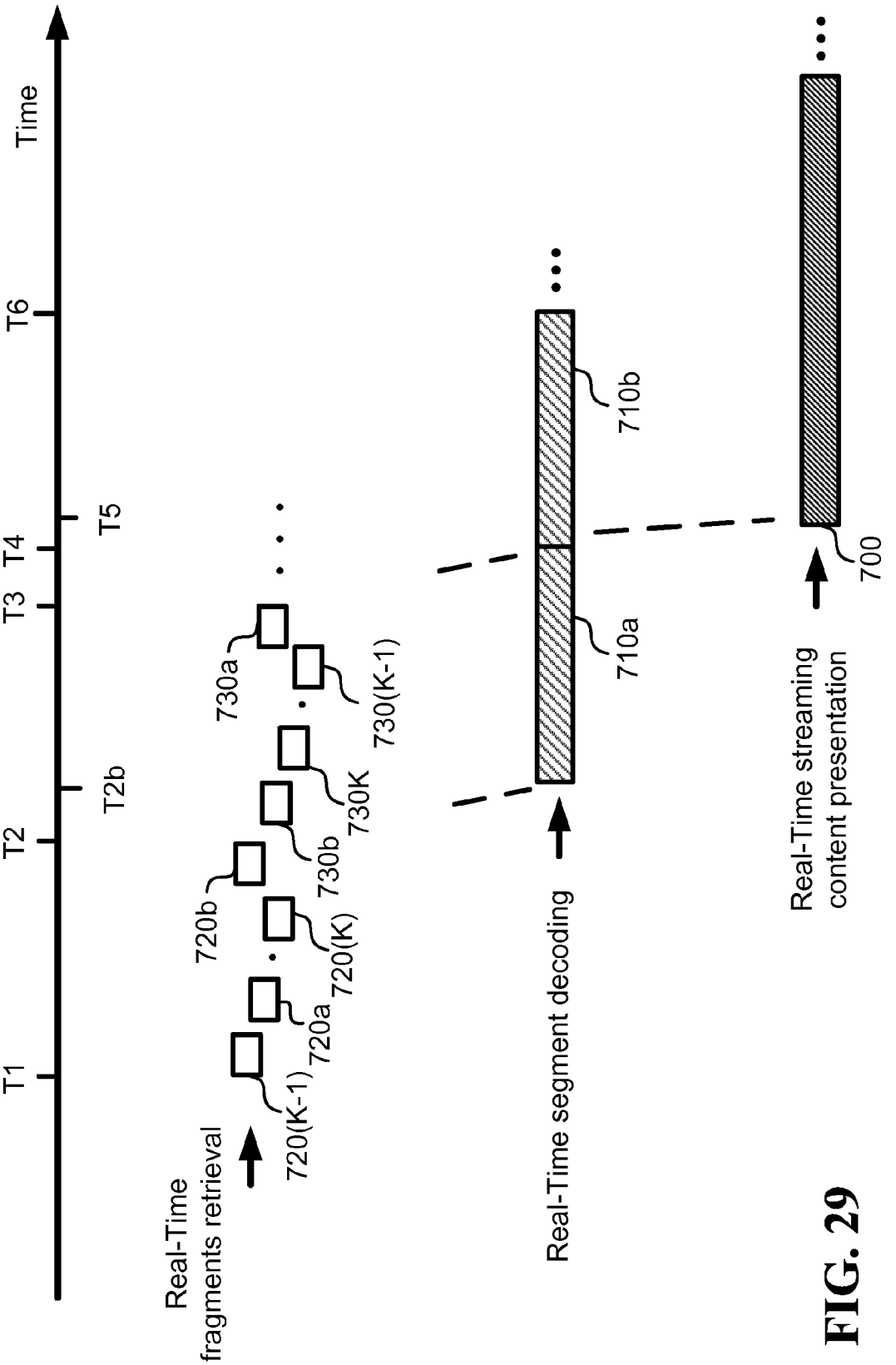

Time
T1
T2
T2b
T3
T4
T5
T6
Real-Time fragments retrieval
720(K-1)
720a
720(K)
720b
730b
730K
730a
730(K-1)
Real-Time segment decoding
710a
710b
Real-Time streaming content presentation
700

FIG. 29

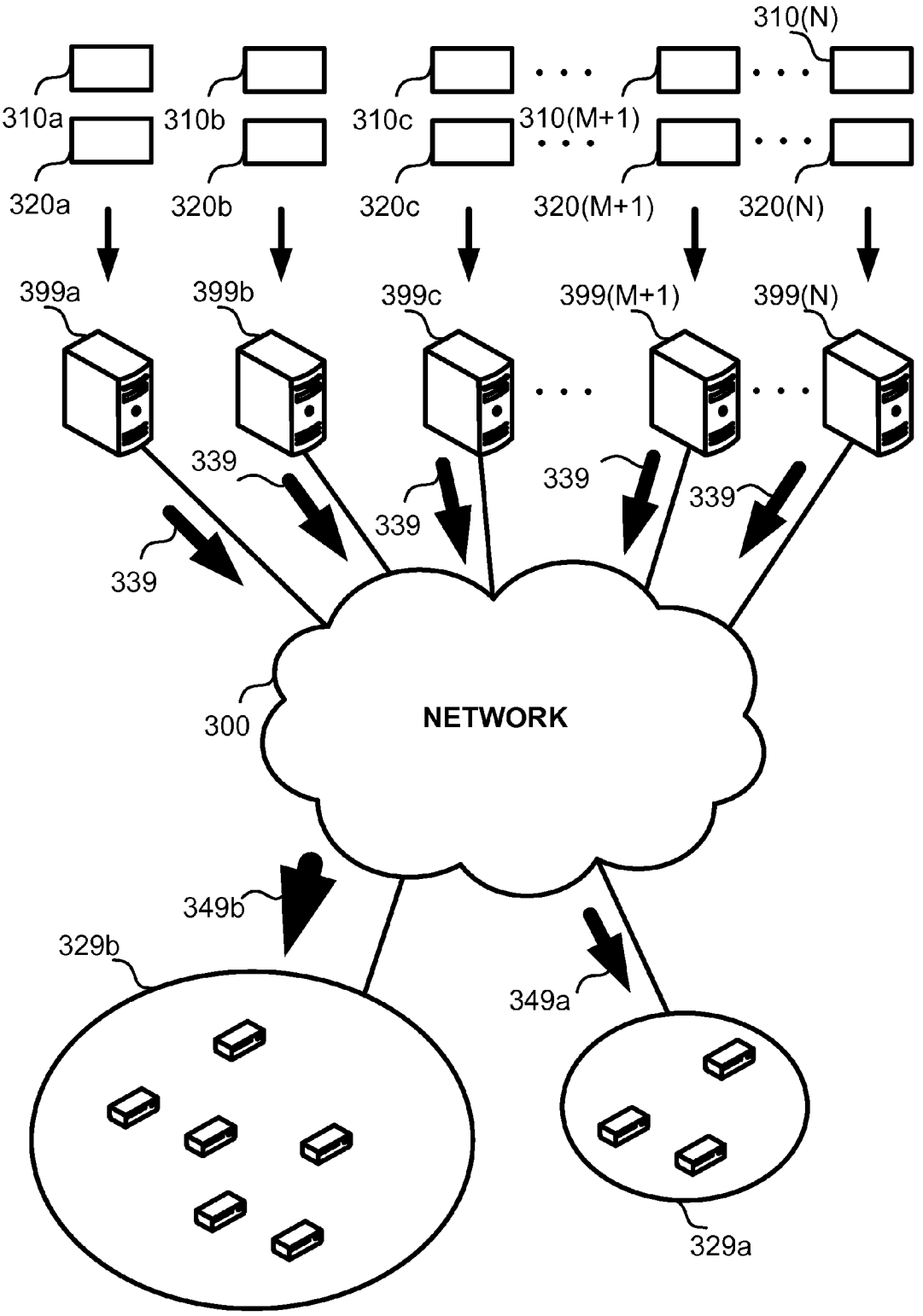

310(N)
310a
310b
310c
310(M+1)
320a
320b
320c
320(M+1)
320(N)
399a
399b
399c
399(M+1)
399(N)
339
339
339
339
339
NETWORK
300
349b
329b
349a
329a

FIG. 30

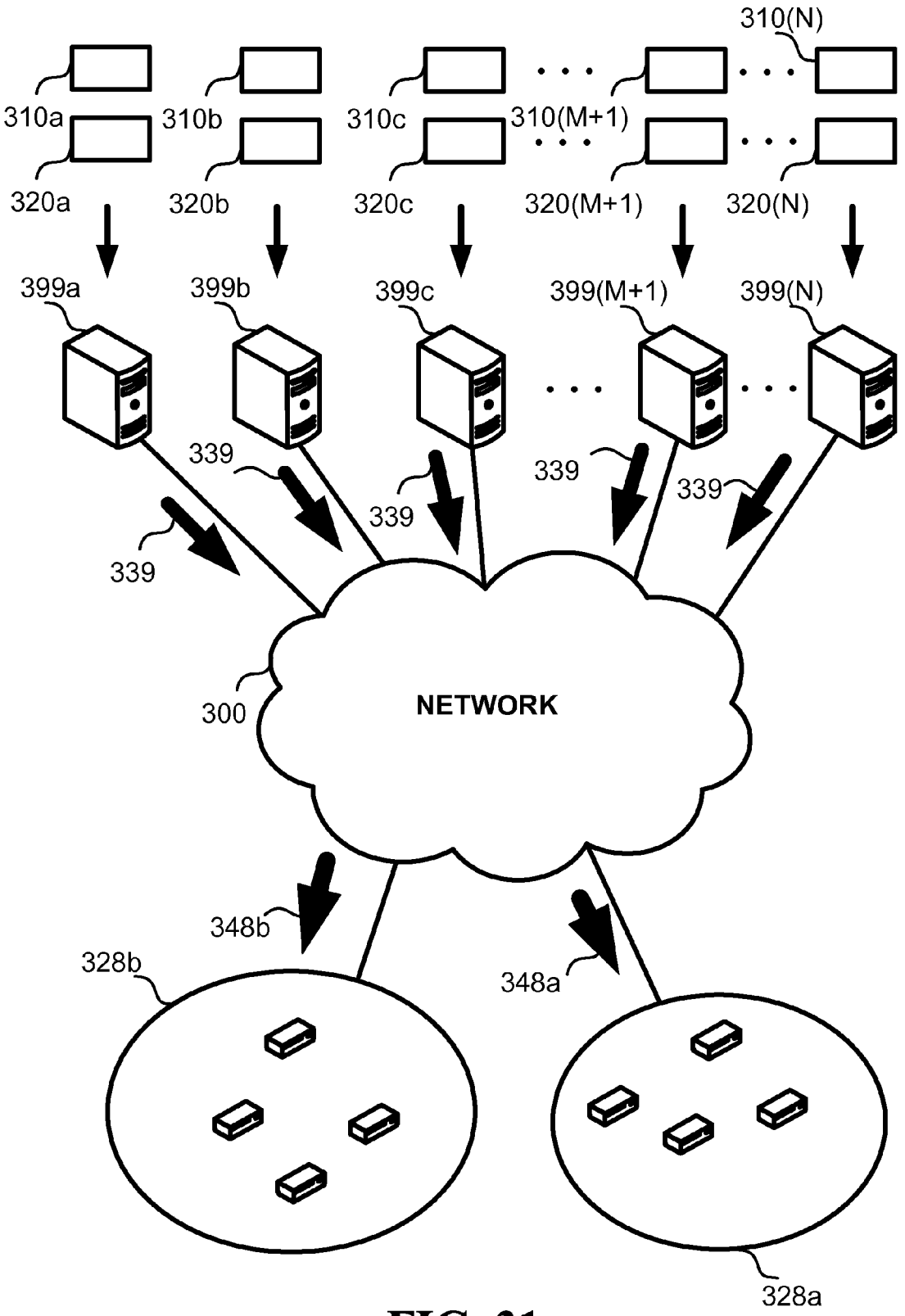

310(N)
310a
310b
310c
310(M+1)
320a
320b
320c
320(M+1)
320(N)
399a
399b
399c
399(M+1)
399(N)
339
339
339
339
339
NETWORK
300
348b
348a
328b
328a

FIG. 31

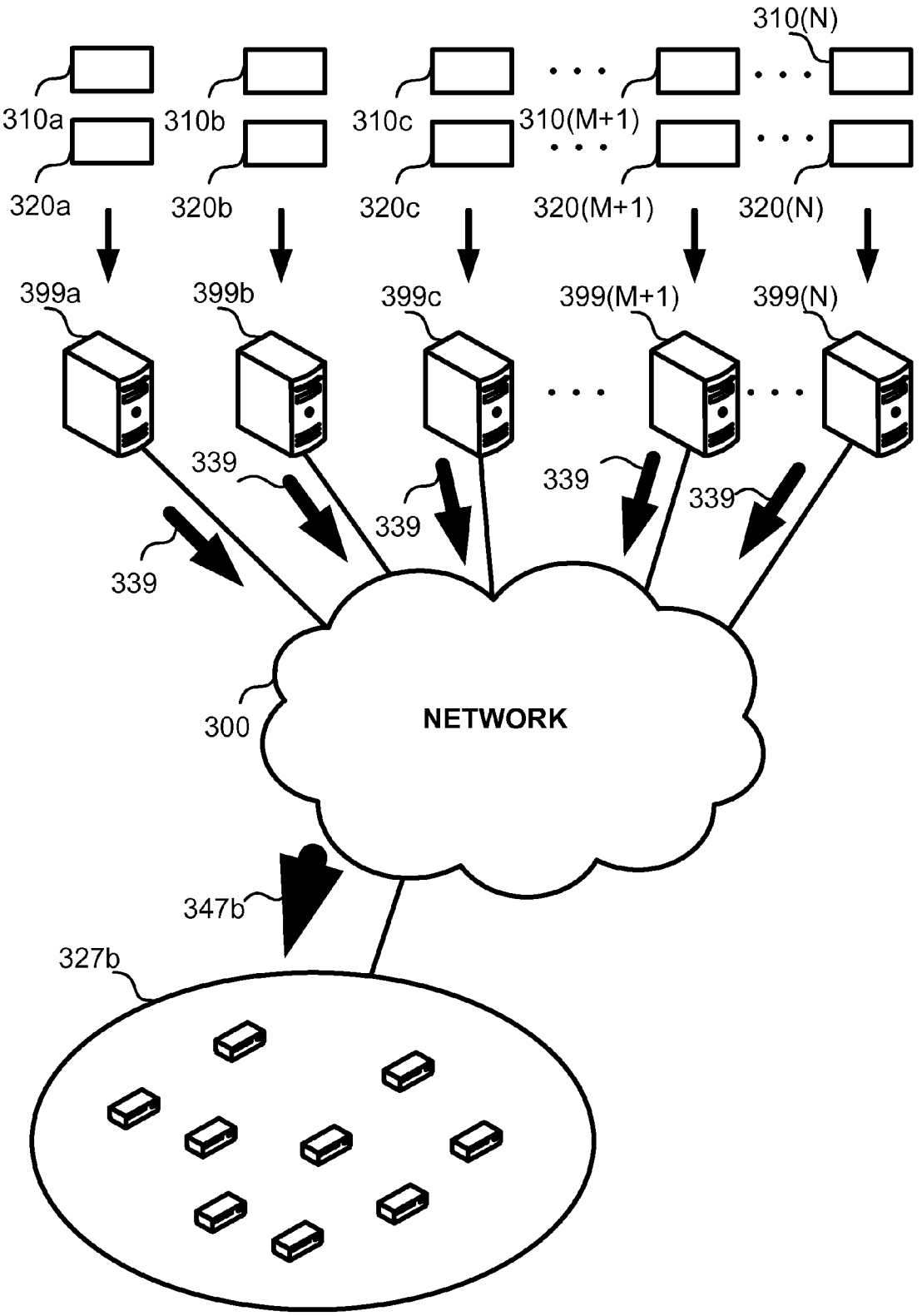

310(N)
310a
310b
310c
310(M+1)
320a
320b
320c
320(M+1)
320(N)
399a
399b
399c
399(M+1)
399(N)
339
339
339
339
339
NETWORK
300
347b
327b

FIG. 32

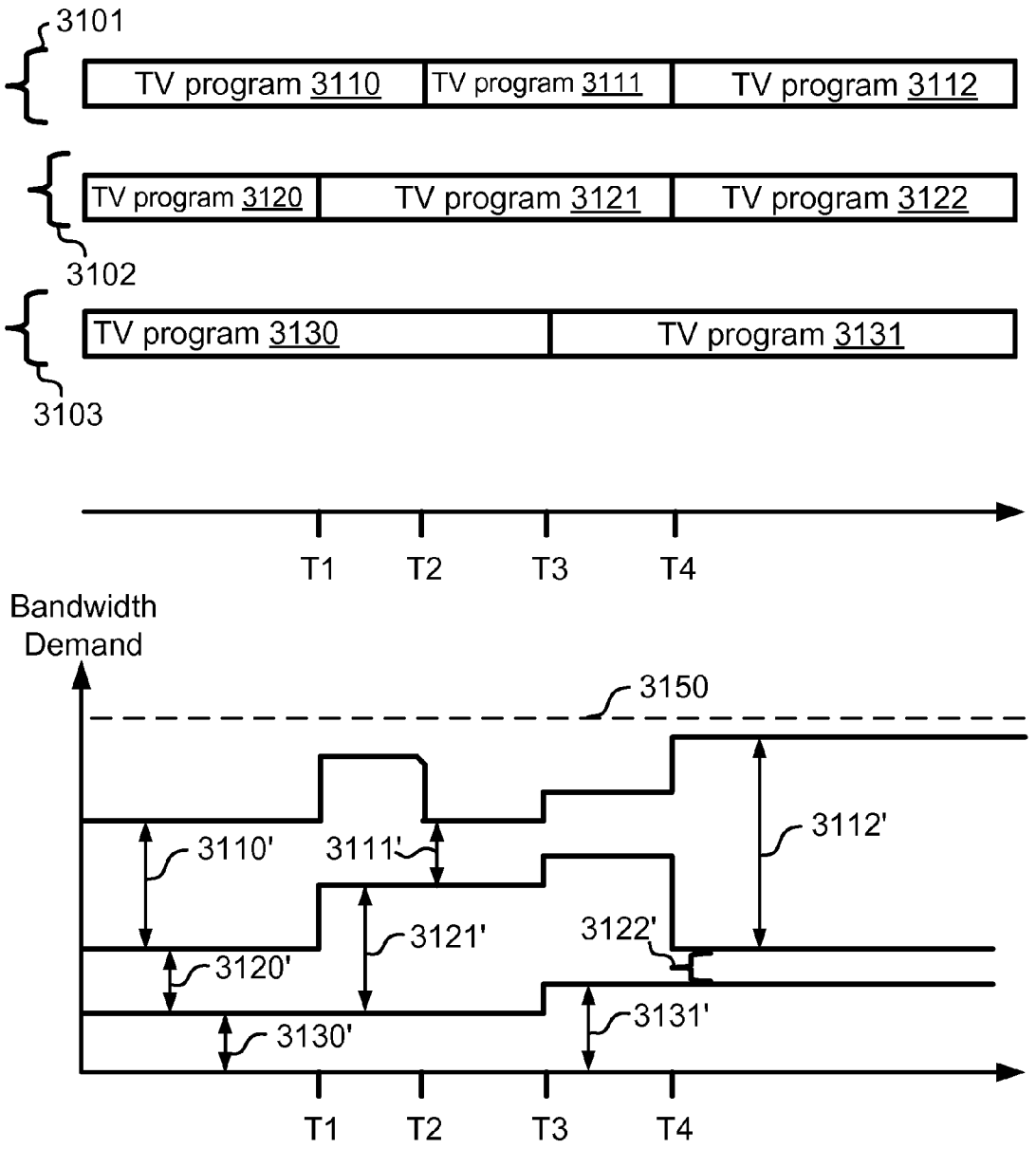

3101
TV program 3110
TV program 3111
TV program 3112
TV program 3120
TV program 3121
TV program 3122
3102
TV program 3130
TV program 3131
3103
T1
T2
T3
T4
Bandwidth Demand
3150
3110'
3111'
3112'
3121'
3122'
3120'
3131'
3130'
T1
T2
T3
T4

FIG. 33

INTERNET BACKBONE SERVERS WITH EDGE COMPENSATION

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 61/105,683, filed Oct. 15, 2008.

BACKGROUND

[0002] When designing a large-scale streaming system, it is possible to locate the streaming servers on the Internet backbone or at the edges of the Internet. An edge-based architecture may provide fast response, may reduce backbone traffic, and provide a better protection against different types of network failures, while a backbone-based architecture may be less expensive and simpler to maintain.

[0003] Commonly used techniques of synchronizing multiple servers for delivering distributed content utilize inter-server communication. These techniques are complex to realize, and often do not optimally utilize resources such as communication bandwidth and storage space.

BRIEF SUMMARY

[0004] In one embodiment, an apparatus comprising: an assembling device configured to obtain erasure-coded fragments from at least one CDN server located close to or on the Internet backbone; upon a fragment loss, the assembling device is further configured to use a fragment pull protocol to retrieve a substitute erasure-coded fragment from a nearby fractional-storage CDN server having low latency in responding to the assembling device.

[0005] In one embodiment, an apparatus comprising: an assembling device configured to receive erasure-coded fragments from at least one CDN server located close to or on the Internet backbone using a push protocol; upon a fragment loss, the assembling device is configured to pull a substitute erasure-coded fragment from a nearby fractional-storage CDN server having low latency.

[0006] In one embodiment, a streaming media delivery network, comprising: at least one CDN server, located close to or on the Internet backbone, configured to provide to an assembling device a first set of erasure-coded fragments of streaming media; and a plurality of fractional-storage CDN servers, located at the edges of the Internet, configured to provide to the assembling device a second set of erasure-coded fragments of the streaming media; wherein the first set is at least 5 times larger than the second set, and the first and second sets comprise enough fragments to enable the assembling device to start playing the streaming media within a short period of time following a request.

[0007] Implementations of the disclosed embodiments involve performing or completing selected tasks or steps manually, semi-automatically, fully automatically, and/or a combination thereof. Moreover, depending upon actual instrumentation and/or equipment used for implementing the disclosed embodiments, several embodiments could be achieved by hardware, by software, by firmware, or a combination thereof. In particular, with hardware, embodiments of the invention could exist by variations in the physical structure. Additionally, or alternatively, with software, selected functions of the invention could be performed by a data processor, such as a computing platform, executing software instructions or protocols using any suitable computer operating system. Moreover, features of the embodiments may be combined.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The embodiments are herein described, by way of example only, with reference to the accompanying drawings. No attempt is made to show structural details of the embodiments in more detail than is necessary for a fundamental understanding of the embodiments. In the drawings:

[0009] FIG. **1** illustrates one embodiment of segmenting content, encoding the segments into erasure-coded fragments, distributing the fragments to fractional-storage servers, and obtaining the fragments by assembling devices and assembling servers.

[0010] FIG. **2** illustrates fractional-storage servers located on the Internet backbone.

[0011] FIG. **3** illustrates CDN servers located on the Internet backbone, supported by fractional-storage CDN servers located on edges of the Internet.

[0012] FIG. **4** illustrates a content delivery center located on the Internet backbone, supported by fractional-storage CDN servers located on edges of the Internet.

[0013] FIG. **5** illustrates geographically distributed fractional-storage servers.

[0014] FIG. **6** illustrates peak-to-average traffic ratios generated by assembling devices distributed over different time zones.

[0015] FIG. **7** illustrates US-based fractional-storage servers delivering erasure-coded fragments to assembling devices spread over the globe.

[0016] FIG. **8** illustrates an assembling server located at a network juncture.

[0017] FIG. **9** illustrates fractional-storage servers having the same bandwidth capability.

[0018] FIG. **10** illustrates fractional-storage servers having different bandwidth capabilities.

[0019] FIG. **11** and FIG. **12** illustrate a case where a fractional-storage server has failed.

[0020] FIG. **13** illustrates a server failure due to network congestion.

[0021] FIG. **14** illustrates retrieving fragments according to locality.

[0022] FIG. **15** illustrates distribution and storage of erasure-coded fragments on fractional-storage servers.

[0023] FIG. **16** illustrates three examples of changes made to redundancy factors according to changes in demand.

[0024] FIG. **17** and FIG. **18** illustrate different embodiments of content segmentation.

[0025] FIG. **19** illustrates an assembling device obtaining erasure-coded fragments from fractional-storage servers.

[0026] FIG. **20** illustrates fast real time fragment retrieval.

[0027] FIG. **21** illustrates real time fragment retrieval, segment reconstruction, and content presentation.

[0028] FIG. **22** to FIG. **25** illustrate various embodiments of fragment pull protocols.

[0029] FIG. **26** illustrates various aggregated and non-aggregated fragment request messages.

[0030] FIG. **27** illustrates retrieving fragments and compensating for failures.

[0031] FIG. **28** illustrates operation of hybrid pull and push protocols.

[0032] FIG. **29** illustrates real time fragment retrieval in random order.

[0033] FIG. **30** to FIG. **32** illustrate changes in content consumption.

[0034] FIG. **33** illustrates utilization of the entire aggregated bandwidth of fractional-storage servers for multiple content delivery.

DETAILED DESCRIPTION

[0035] FIG. **1** illustrates one embodiment of a fractional-storage system configured to store erasure-coded fragments. Content **100**, which may optionally be streaming content, is segmented into content segments **101***a*, **101***b* to **101***j* (for brevity referred to as segments). Each of the segments is encoded into erasure-coded fragments. For example, segment **101***a* is encoded into erasure-coded fragments **390***a* to **390**(N). The erasure-coded fragments are distributed to the fractional-storage servers **399***a* to **399**(N) and/or to the bandwidth amplification devices **610***aa*. The erasure-coded fragments are then obtained by assembling devices like **661** or proxy servers like proxy server **661***s* from the fractional-storage servers **399***a* to **399**(N) and/or the bandwidth amplification devices **610***aa*. The obtained erasure-coded fragments are decoded to reconstruct the segments. The proxy server **661***s* may broadcast/multicast and/or re-stream the reconstructed content, optionally using standard streaming technique, to its client(s) **661***o*, optionally over network **300***n*. In some embodiments, the content distribution is performed in real time. In some embodiments, the content assembly is performed in real time and the presentation starts a short time after the content request.

[0036] Similarly to content **100**, additional contents are segmented, encoded into erasure-coded fragments, and distributed to the fractional-storage servers and/or to the bandwidth amplification devices. Each segment may be reconstructed independently of other segments by obtaining and decoding enough erasure-coded fragments generated from that segment.

[0037] In some embodiments, the encoding scheme is erasure codes and/or rateless codes. In some embodiments, the fractional-storage servers **399***a* to **399**(N) are Content Delivery Network (CDN) servers, optionally accessed over the public Internet. In some embodiments, the control, management, content reception, content segmentation, segment encoding, erasure-coded fragment distribution, allocation of bandwidth amplification devices, and/or other kind of central supervision and operation may be performed by managing server(s) **393**, which may be a part of the CDN network. It is noted that the term "fractional-storage server" is not limited to a large server and, according to the context, may include a fractional-storage bandwidth amplification device, a fractional-storage peer server, or other types of fractional-storage servers.

[0038] The term "erasure coding" as used herein denotes a process in which a sequence of erasure-coded fragments can be generated from a segment such that the segment can be reconstructed from any or almost any subset of the erasure-coded fragments of size equal to or somewhat larger than the size of the segment (sometimes may be referred to as "enough erasure-coded fragments" or "sufficient subset of fragments"). Examples of erasure codes include, but are not limited to, rateless codes, Reed-Solomon codes, Tornado codes, Viterbi codes, Turbo codes, any Block codes, any Convolutional codes, and any other codes that are usually used for forward error correction (FEC).

[0039] The term "rateless coding" as used herein denotes a type of erasure coding in which a very long, potentially limitless, sequence of rateless-coded fragments can be generated from a segment such that the segment can be reconstructed from any or almost any subset of the rateless-coded fragments of size equal to or somewhat larger than the size of the segment (sometimes may be referred to as "enough rateless-coded fragments"). Examples of rateless codes include, but are not limited to, Raptor codes, LT codes, online codes, any Fountain codes, and any other Rateless codes.

[0040] The term "erasure-coded fragment" denotes a fragment comprising data encoded with an erasure code (which may also be a rateless code in some embodiments). The term "rateless-coded fragment" denotes a fragment comprising data encoded with a rateless code.

[0041] The term "assembling device" as used herein denotes a computing device that retrieves erasure-coded fragments from servers over a network. The assembling device may perform one or more of the following: (i) Decode the retrieved erasure-coded fragments into segments. (ii) Present the content reconstructed from the retrieved erasure-coded fragments. (iii) Act as a bandwidth amplification device, by receiving, storing, and forwarding erasure-coded fragments. In some embodiments, the assembling device may be any device located at the user premises, like an STB, PC, gaming console, DVD player, PVR device, or any other device able to retrieve erasure-coded fragments from a communication network. In some embodiments, the assembling device may be an assembling server. In some embodiments, the assembling device may be any computational device with access to a communication network, located at a central office, data center, BRAS location, ISP premises, or any other place with direct network connectivity. In one embodiment, the assembling device is coupled to a display device used for content presentation.

[0042] The abbreviation CDN denotes "Content Delivery Network". The term "CDN server" as used herein denotes a server having one or more of the following characteristics: (i) A bandwidth (CDN_BW) that is much greater than the average bandwidth consumed by a user premises device (User_BW) receiving video streaming content. In some examples, the CDN_BW is at least 10 times, 100 times, 1,000 times, or 10,000 times greater than the User_BW. (ii) The server is located outside the last mile communication infrastructure of the end users, such that the CDN server and the end users are located in different networks. For example, the CDN server is not located under a BRAS, while the end users are located under a BRAS. Moreover, in some embodiments, the CDN servers are deployed over a wide area across the Internet and optionally located close to or on the Internet backbone. In some embodiments, the CDN server does not usually retrieve and play streaming content. In some embodiments, the CDN server has a much greater storage space than the storage space of an average player of streaming content.

[0043] The term "fractional-storage server" in the context of erasure-coded fragments (also applicable to "fractional-storage CDN server"), as used herein denotes a server that (i) stores less than the minimum quantity of erasure-coded fragments required to decode the erasure-coded fragments, and (ii) where at least a meaningful quantity of the stored erasure-coded fragments is not stored in order to be consumed by the fractional-storage server.

[0044] The term "streaming content" as used herein denotes any type of content that can begin playing as it is

being delivered. Streaming content may be delivered using a streaming protocol, a progressive download protocol, or any other protocol enabling a client to begin playing the content as it is being delivered. Moreover, the term "streaming protocol" includes "progressive download protocol". In addition, the verb "streaming" refers to using a streaming protocol, using a progressive download protocol, or using any other protocol enabling the receiver to begin playing the content as it is being delivered.

[0045] The term "approximately random" as used herein refers to, but is not limited to, random, pseudo random, and/or based on a long list of numbers featuring very low autocorrelation and very low correlation with other similar lists of numbers.

[0046] In some embodiments, expressions like "approximately sequential segments" may denote one or more of the following non-limiting options: segments that are sequential (in time or according to a file's order), segments that are approximately sequential (such as segments with some interlace, or segments without a great amount of non-sequential data), segments generated sequentially and/or approximately sequentially from different components of content (such as storing the i-frames and p-frames of a compressed content in different segments), and/or other sequential or approximately sequential segmentation after classification or separation into different components and/or elements.

[0047] In one embodiment, a CDN is created by the aggregated bandwidth and storage capacity of the participating erasure-coded fractional-storage servers. In one example, a large scale CDN includes several hundreds or thousands of fractional-storage servers connected to the Internet. These servers send erasure-coded fragments to a large number, potentially millions, of assembling devices. In order to keep costs low for sending a large number of fragments from fractional-storage servers to assembling devices, the servers are located on the Internet backbone, or close to it.

[0048] The current Internet backbone primarily comprises different Tier one ISP (or other) networks that interconnect at various Internet Exchange Points (IX or IXP), using peering agreements. Tier one ISPs, or other backbone-forming network entities, can reach any portion of the Internet via other Tier one ISPs or other backbone-forming networks, without paying any Internet transit fee, and solely by utilizing mutual peering agreements. In order to gain access to large amounts of inexpensive bandwidth, the fractional-storage servers are typically located on the Internet backbone. This means that the servers are either co-located (and connected) with a core switching router that interconnects the Internet backbone networks at an IXP, or, alternatively, co-located (and connected) with a router which is part of the backbone network, typically located at a data center or co-location center. Fractional-storage servers can also be located close to the Internet backbone, which means that they are co-located (and connected) with a router which is part of a Tier two ISP network, which has a high bandwidth connection with at least one Tier one operator, to which it pays transit fees in order to potentially reach all portions of the Internet. FIG. **2** illustrates one example of a fractional-storage server **3001**, which is one of a plurality of servers forming a large-scale CDN, located on the Internet backbone by being connected to the Internet backbone via IXP **3091**. In a second example, fractional-storage server **3002** is located on the Internet backbone by being connected to a Tier one backbone network **3080**. In a third example, fractional-storage server **3011** is located close to the Internet backbone by being connected to a Tier two ISP network **3070**, which is connected to the backbone via Tier one ISP network **3081**. In one embodiment, a typical fractional-storage server is located on the backbone or close to the backbone by being attached to a switching router via a high bandwidth port, such as a 1 Gbps, 10 Gbps, or a higher bandwidth port, such as high-speed Ethernet port, usually carried over a fiber, or suitable short-distance copper lines. In one embodiment, in a typical deployment using high bandwidth connections (in 2009 terms), each of about 1,000 fractional-storage servers is located on the backbone or close to the backbone and is connected to the backbone via a dedicated (guaranteed bandwidth) 1 Gbps Ethernet port, resulting in an aggregated throughput of 1,000 Gbps, which can serve about one million subscribers of standard definition streaming video, such as client device **3020**, simultaneously. Such aggregated bandwidths would have required a substantially larger number of fractional-storage servers, had they been connected to other locations in the Internet, such as at edges of the Internet (close to last mile networks), Tier 3 ISPs, or at the user premises. Moreover, in some embodiments, the cost of streaming the mentioned 1,000 Gbps when the fractional-storage servers are located on the Internet backbone, or close to the Internet backbone, is expected to be significantly lower than what is expected when the servers are located elsewhere as mentioned before.

[0049] In one embodiment, an assembling device operating trick play modes needs to obtain new erasure-coded fragments within a short period, to replace lost erasure-coded fragments. Therefore, the new fragments are retrieved from one or more nearby fractional-storage servers having low latency responses to the assembling device. The nearby fractional-storage servers should have sufficient bandwidth to supply the new fragments needed for the trick play, but because most of the fragments are not obtained from the nearby servers, these nearby servers may have relatively low bandwidth and may store relatively small portions of the information.

[0050] In one example, a distant server/s stores approximately all erasure-coded fragments needed to reconstruct segments by an assembling device. The assembling device attempts to obtain enough of the fragments to reconstruct segments. However, due to fragment loss conditions optionally resulting from the distance that the fragments need to traverse from the server/s to the assembling device, only about 97% of transmitted fragments actually reach the assembling device. The assembling device therefore needs to supplement the lost fragments with additional fragments needed to reconstruct the segments. Therefore, the assembling device requests an additional amount of fragments equal to about 3% of the total fragments sent by the distant server/s, from a nearby fractional-storage server. The additional fragments are quickly received from the nearby server, and most likely without any fragment loss, due to the proximity of the nearby server. The nearby server needs to store only a small fraction of the fragments per segments of content, since it is required to supplement only a small portion of lost fragments, which corresponds in percentage to the fragment loss ratio. With 3% fragment loss condition, the nearby server can store only about 3% of the fragments per segments of content, such that if 200 fragments are needed to reconstruct a segment, the nearby server can store only 6 or 7 fragments per segment. Moreover, the nearby server can supplement the small fraction of the fragments with a rela-

tively low bandwidth communication link. For example, for a 1 Mbps fragment throughput sent by the distant server, the nearby server needs only 3%, or 30 Kbps, in order to supplement the missing fragments.

[0051] In one embodiment, the nearby fractional-storage server may store more fragments than needed just for fragment loss compensation. In one example, the nearby server stores 30% of the fragments needed for segment reconstruction. This may improve response times, reduce some of the backbone traffic, and provide a better protection against different types of network failures.

[0052] In one embodiment, only certain sections of the contents support trick play and a significant portion of the erasure-coded fragments stored on the nearby fractional-storage servers are associated with these particular sections. This embodiment reduces the storage requirements, and, optionally, also the bandwidth requirements, from the nearby fractional-storage servers. In one example, only 10% of the segments support trick play operation, meaning that the assembling device can start a content presentation from only 10% of the content's segments. In this case, and still assuming a 3% fragment loss condition from the distant server/s, the nearby fractional-storage server can store only 10%×3%=0.3% of the content's fragments. It is also estimated that in this case the nearby server's bandwidth requirements will be lowered as well.

[0053] FIG. **3** illustrates one embodiment of CDN servers **3001**, **3002**, and **3011**—which may be fractional-storage or may store full replicas—located close to or on the Internet backbone, supported by fractional-storage CDN servers **2505**, **2506** located on edges of the Internet. The assembling devices attempt to obtain the required erasure-coded fragments from CDN servers **3001**, **3002**, and **3011**, which may have high latency relative to the servers on the edge. Upon a fragment loss, the assembling devices pull a substitute erasure-coded fragment from the nearby fractional-storage CDN server, which has low latency. For example, assembling devices **2500** and **2501** pull substitute erasure-coded fragment from fractional-storage CDN server **2505**, and assembling device **2502** pulls from server **2506**. Because the nearby fractional-storage CDN servers **2505** and **2506** are approached mainly after a fragment loss (or other predefined occasions), servers **2505** and **2506** may have a much smaller storage and bandwidth relative to the total content consumed by the assembling devices **2500**, **2501**, and **2502**.

[0054] FIG. **4** illustrates one embodiment of a content delivery center **2508** housing from ten to thousands of servers located close to or on the Internet backbone, supported by fractional-storage CDN servers **2505**, **2506** located on edges of the Internet. Using push or pull protocols, the assembling devices attempt to obtain the required erasure-coded fragments from the content delivery center **2508**, which may have high latency relative to the servers on the edge. Upon a fragment loss, the assembling devices pull a substitute erasure-coded fragment from the nearby fractional-storage CDN server, which has low latency. In this embodiment, fragment loss may include a fragment that was not received, a fragment received after a predefined duration, or an erred fragment.

[0055] In one embodiment, erasure-coded stream is received by an assembling device from one or more distant servers featuring high latency, which may be susceptible to frequent fragment loss. In order to avoid requesting retransmissions and avoid requesting extra fragments to compensate for the fragment loss, especially while using trick play, the assembling device stores a partial buffer of erasure-coded fragments, which, in one example is used to compensate for the fragment loss. Optionally, the erasure-coded fragments of the partial buffer are distributed to the assembling device before it requests the content. Optionally, the erasure-coded fragments of the partial buffer are added to a stream transmitted to the assembling device, in order to reduce the time needed to perform a future trick play request.

[0056] In one embodiment, lost fragments associated with segments designed for trick play are substituted by fragments from the nearby fractional-storage CDN serves, while lost fragments associated with segments not designed for trick play are substituted by erasure-coded fragments from the CDN server/s located close to or on the Internet backbone.

[0057] In one embodiment, the streaming content is played within a short period following a request by an assembling device, wherein the short period comprises supplemental fragment requests by the assembling device, and corresponding reception of the supplemental fragments from nearby servers. Optionally, the short period of time following a request is less than 15 seconds for a high-definition full-length movie.

[0058] FIG. **5** illustrates one example of geographically distributed fractional-storage servers **399***a* to **399***n*, in which servers **399***a* to **399***c* are located in Europe **676**, servers **399***d* to **399***g* are located on the east coast of the US **677**, servers **399***h* to **399***i* are located on the west coast of the US **678** and servers **399***k* to **399***n* are located in Japan **679**. Assembling devices all over the world obtain erasure-coded fragments from the globally distributed fractional-storage servers. The characteristics of the fractional-storage system, according to some embodiments, allow the globally distributed assembling devices to exploit the outgoing bandwidth of the globally distributed fractional-storage servers approximately up to the point where all servers **399***a* to **399***n* utilize their available outgoing bandwidth for content delivery.

[0059] In one embodiment, the main demand for fragments shifts between the different global locations as the day elapses. For example, at 8 pm Pacific Standard Time, the main fragment demand is generated from the US west coast. At that time, the local time in the east coast is late evening, the time in Europe and Japan is early morning and noon respectively, and thus very little fragment demand is generated from these regions. The high fragment demand load generated from the west coast is spread across all of the fractional-storage servers. As the day elapses, the load generated from the west coast declines, and the main load shifts to Japan as time there becomes afternoon. When that happens, the servers are still able to supply all content demands, as they are still able to deliver maximal bandwidth to assembling devices in Japan. As the cycle continues, the main load shifts again from Japan to Europe, from Europe to the US east coast, and from there back to the US west coast, following a 24-hour cycle. In some embodiments, the servers are able to deliver maximal fragment traffic, resulting from peak demands occurring during a day cycle, to anywhere on the globe.

[0060] In one example, there are 14 globally distributed fractional-storage servers; each server has a bandwidth of B, and the total capacity of the array is 14×B. Assuming the total global peak demand during the daily cycle does not exceed Bg, then the system is balanced and can meet all demands during the daily cycle if $Bg<14\times B$, meaning that $B>Bg/14$. In this example, all servers may be at, or may approach, their peak bandwidth capabilities for a relatively long period, and

feature relatively short idle periods. In one example, the number of servers in the global array is 10,000, from which 2,500 are located on the US west coast, 2,500 on the east coast, 2,500 in Europe and 2,500 in Japan. In one example, the number of servers in the global array is 1,000, from which 100 are located on the west coast, 700 on the east coast, 100 in Europe and 100 in Japan.

[0061] In one embodiment, multiple contents originating from multiple global locations (and therefore expected to require high loads at different times of day), are all stored on the globally distributed fractional-storage servers. Therefore, the system's bandwidth capacity equals the aggregated bandwidth of its server members, optionally regardless of which content generates high load, regardless of when the load is generated during the day, and regardless of where the load is generated from.

[0062] In one embodiment, at some point in time, some portions of the Internet may become congested at some global locations. The global system assures that servers not affected by the congestion handle the excess load, such that operation close to peak bandwidth performance is still possible.

[0063] In one embodiment, the globally distributed assembling devices retrieve fragments from the fractional-storage servers using a fragment pull protocol, and determining which servers deliver fragments to which assembling devices load balances the distributed system. In one embodiment, the globally distributed assembling devices obtain fragments from fractional-storage servers using a push protocol with multiple sub-transmissions, and determining which servers deliver fragments via the sub-transmissions to which assembling devices load balances the distributed system.

[0064] FIG. **6** illustrates one embodiment in which assembling devices distributed over different time zones together induce fragment traffic having a reduced peak-to-average traffic ratio, as compared to the fragment traffic induced by assembling devices located in any single time zone. Graph **1602** illustrates the fragment traffic induced by assembling devices located at a first time zone. The peak of graph **1602** occurs during the late afternoon, local time of the first time zone. Similarly, graphs **1603** and **1604** illustrate induced traffic from second and third time zones. Since the first, second and third time zones are different, the peak traffic of each graph occurs at a different time. The peak-to-average fragment traffic ratios of graphs **1602** to **1604** are relatively high, since most of the traffic is generated close to the peak demand. In the case of video traffic, a daily peak-to-average traffic ratio of about six is expected during one day, starting at T**1** and ending at T**2**. The combined traffic induced by all assembling devices is the sum of graphs **1602** to **1604**, which is schematically illustrated as graph **1601**. Since the peaks of graphs **1602** to **1604** occur at different times, the combined traffic **1601** behaves much more smoothly and has peaks close to the peaks of graphs **1602** to **1604**, resulting in a much lower peak-to-average traffic ratio, which in some embodiments is about two or three. This means that the fractional-storage servers can be utilized during longer periods of the day when servicing assembling devices located at different time zones. In one embodiment, the distribution of the assembling devices to the different time zones results in an approximately flat traffic during the day, having a peak-to-average traffic ratio approaching one. Such a distribution is challenging in real life deployments, but can be approached by engineering the distribution of the assembling devices over the globe.

[0065] Still referring to FIG. **6**, in one embodiment, the severs are connected to the Internet using guaranteed fixed bandwidth communication links, and can together deliver to the Internet fragment traffic of **1610** all day. In this case, it is clear that traffic graph **1601** utilizes the fixed bandwidth capacity **1610** better than any of the graphs **1602** to **1604**, since it approaches the maximal capacity for longer periods over the day.

[0066] In one embodiment, the servers are spread over two or more continents, and some of the fragments associated with the same segments are stored on different servers located on different continents. This achieves content placement diversity, and results in better immunity to different network and server faults.

[0067] FIG. **7** illustrates one embodiment in which US-based fractional-storage servers **399***a*' to **399***n*' deliver erasure-coded fragments to assembling devices spread over the globe. The assembling devices spread over the globe induce a total fragment traffic from the US-based servers having a reduced peak-to-average traffic ratio, as compared to the fragment traffic induced by assembling devices located in any single time zone. In one example, 5,000 fractional-storage servers are located in the US and service 10 million assembling device subscribers spread over the globe. At a first period during the day, the servers deliver erasure-coded fragments concurrently to 2 million assembling devices located primarily in Japan. At a second period during the day, the servers deliver erasure-coded fragments concurrently to 2 million assembling devices located primarily in Europe. At a third period during the day, the servers deliver erasure-coded fragments concurrently to 2.5 million assembling devices located primarily on the East Coast, and ½ million assembling devices located primarily on the West Coast. At a fourth period during the day, the servers deliver erasure-coded fragments concurrently to ½ million assembling devices located primarily on the East Coast, and 2.5 million assembling devices located primarily on the West Coast. According to this example, the servers are capable of delivering a peak fragment traffic resulting from the demand of at least 3 million assembling devices concurrently.

[0068] In one embodiment, a distributed system is located in a few to dozens of data centers (also known as server farm or datacenter), located close to or on the Internet backbone, together housing at least 100 fractional-storage CDN servers. The servers store erasure-coded fragments associated with approximately sequential segments of streaming contents, with a storage gain of at least 5, and transmit the stored fragments on demand to assembling devices approximately according to the sequential order of the segments. In many cases, the data centers provide a convenient place to place the CDN servers close to or on the Internet backbone. A data center can be also a collocation center, or an Internet Exchange Point. In one example, a single data center can house many fractional-storage CDN servers.

[0069] Assuming all segments have approximately the same size and all fragments generated from the segments have approximately the same size (without limiting any of the embodiments), the term "storage gain" as used herein denotes the following ratio: (size of a segment)/(size of an erasure-coded fragment). If the server stores more than one erasure-coded fragment per segment, the storage gain denotes the following ratio: (size of segment)/((size of erasure-coded fragment)*(number of stored erasure-coded fragments per segment)).

[0070] In one example, a streaming system comprising at least several hundreds of fractional-storage CDN servers located close to or on the Internet backbone, storing erasure-coded fragments encoded with a redundancy factor greater than one, and associated with approximately sequential segments of streaming contents. At least 100,000 assembling devices concurrently obtain fragments from the CDN servers, wherein the system achieves efficient load balancing and fault tolerance between the various CDN servers by determining for each of the assembling devices from which servers to obtain the fragments.

[0071] The term "redundancy factor" as used herein denotes the following ratio: (total size of the unique erasure-coded fragments generated from a segment and actually stored on the servers)/(size of the segment).

[0072] In one example, a system comprising at least 1,000 fractional-storage CDN servers is connected to the public Internet. The servers store erasure-coded fragments associated with approximately sequential segments of streaming contents, with a storage gain greater than 5, and transmit the stored fragments on demand to assembling devices approximately according to the sequential order of the segments. Wherein the aggregated bandwidth utilized by the servers for transmitting the fragments to the assembling devices exceeds 1 Giga bit per second times the number of the CDN servers. In one optional example, the system comprises at least 10,000 fractional-storage CDN servers and the aggregated bandwidth utilized by the servers exceeds 10 Giga bit per second times the number of the CDN servers.

[0073] In one embodiment, the assembling device categorizes the servers into two categories: (i) fastest responding servers, and (ii) slower responding servers, and approximately avoids initial fragment requests from the fastest responding servers, such that if additional fragments are needed, they are quickly retrieved from the fastest responding servers. Avoiding retrieval from the fastest responding servers when initially requesting the fragments of a segment increases the chances of retrieving a substitute fragment, needed to compensate for the lost fragments, from the fastest responding servers, and enables fast compensation that is needed for fast presentation of the streaming content. Categorizing the servers may be performed by registering measured latencies of servers responding to fragment requests by the assembling device.

[0074] In one embodiment, a plurality of fractional-storage servers, which may be located almost anywhere around the globe, configured to store erasure-coded fragments associated with segments of streaming content. An assembling device, which may be located almost anywhere around the globe, configured to request, using a fragment pull protocol over the Internet, a set of fragments. The assembling device is further configured to compensate for lost fragments by requesting additional erasure-coded fragments that are needed to reconstruct the segments. wherein the bandwidth of the streaming content is bounded approximately only by the incoming bandwidth of the assembling device.

[0075] In one embodiment, fractional-storage CDN servers configured to store erasure-coded fragments associated with approximately sequential segments of streaming content. An assembling device located at a point featuring an average one-way network-related latency of more than 50 milliseconds between the assembling device and the servers obtains a first set of fragments, approximately according to the sequential order of the segments, and compensates for lost fragments by obtaining a second set of erasure-coded fragments that are needed to reconstruct the segments. Wherein the bandwidth of the streaming content is bounded approximately only by the incoming bandwidth of the assembling device. Optionally, the assembling device is configured to utilize a fragment pull protocol to obtain the fragments. Optionally, the assembling device utilizes a push protocol to obtain the fragments.

[0076] Referring again to FIG. **1** with device **6610** as a non-assembling CPE, such as a STB, PC or gaming console, capable of performing standard request, reception, and decoding of video over IP network. In one embodiment, server **661***s*—also referred to as proxy server, assembling server, and in some cases assembling device—performs three primary functions: (i) receipt of content requests from non-assembling client device **6610**; (ii) assembly of content, as requested by client **661***o*, from the fractional-storage servers and optionally from the bandwidth amplification devices; (iii) optionally, conversion of the assembled content into a streaming format; and (iv) transmission of the streaming content to the requesting client **661***o*. Client **6610** can then store the content, or present it. In one embodiment, the assembled content is a general web content, including HTML, FLASH or any other data format that can be found in a web-based site.

[0077] In one embodiment, although server **661***s* is illustrated as being connected to network **300** on one side and to network **300***n* on the other, server **661***s* may also be connected to another network element, such as a router, which makes the topological connection between networks **300** and **300***n*. In that case, server **661***s* communicates with both networks **300** and **300***n* via the other network element.

[0078] FIG. **8** illustrates one example where an assembling server **4020** is located at the juncture **4010** between two networks: the first network is an ISP transit network **4014** that connects the juncture to the Internet and provides Internet transit via a switching router **4015**, and the second is a last mile network **4041** that connects end users **4051** to the Internet via a switch **4031** (located, for example, inside a Central Office, a Head-End, or a street-level cabinet). In one embodiment, the juncture **4010** is a network operated by a local ISP that pays transit fees for Internet traffic passing through the transit network **4014**, and last mile fees for traffic passing through the last mile network **4041**. A unique property of the juncture **4010** is that it is possible for an assembling server **4020** located at the juncture to receive erasure-coded fragments sent by fractional-storage servers, such as **4001** and **4002**, to assemble content, and to stream the content to a client **4051** via the last mile network **4041**, without incurring any additional costs in comparison to other scenarios, such as where Internet packets flow from the Internet backbone to a Tier two ISP network to the Internet backbone and to the last mile network. In other words, since the assembling server **4020** is located at the juncture, it does not create any extra traffic via networks **4014** and **4041**. The assembling server can also be located at or close to an edge of the Internet, which may include the juncture, or a point above server **4015**, such as at the transit network **4014** connecting the juncture to the Internet. When located at or close to an edge of the Internet, the assembling server has the potential not to incur additional transit fees as a result of the relaying operation, since approximately the same traffic would have to pass via the same transit network in a normal scenario. Another beneficial location for the assembling server is at the home premises, since, clearly, a relaying operation performed there does not add any significant traffic to higher levels of the network. In contrast to

the above-suggested locations, in some cases an assembling server may be located at an arbitrary point on the backbone, or at other high-level points of the Internet, where it incurs additional transit fees, as fragments assembled by the server flow once over an Internet transit network going from a fractional-storage server to the assembling server, and then a second time when streamed by the assembling server to a destination client over an Internet transit network.

[0079] FIG. **9** illustrates one example of a fractional-storage system comprising servers **699***a* to **699**(N) having a bandwidth capability **681**. In other words, no server can send data at a rate higher than **681**. Assembling device **661** can select from which servers to obtain erasure-coded fragments for reconstruction of a segment. In one example, each server stores one relevant, unique, erasure-coded fragment. Therefore, from the N servers storing N possible unique fragments, the assembling device needs only K erasure-coded fragments for complete reconstruction of the segment (K<N). Since it is not important which K fragments from the N are retrieved, the assembling device may retrieve from the least loaded servers, so as to keep the load between the different servers balanced. When many assembling devices assemble contents in parallel, and since all assembling devices can select the least loaded servers, the end effect is that the load on the servers is balanced, with the potential for most servers to approach their maximal bandwidth capabilities. Optionally, that load balancing is achieved without significant coordination between the servers.

[0080] In the example of FIG. **9**, assuming that K=3, the assembling device **661** may select servers **699***b*, **699**(N−1), and **699***a* for fragment retrieval, as they have the lowest load of all N servers. Servers **699***c* and **699**(N), as an example, will not be chosen, as they have relatively higher loads.

[0081] The assembling device may select the least loaded servers using any appropriate method, such as, but not limited to (i) accessing a central control server having data about the load conditions on the various servers, or (ii) periodically querying the various servers on their load conditions.

[0082] In one embodiment, instead of, or in addition to, selecting the least loaded servers, the assembling device **661** tries a random set of K servers from the N, and retrieves erasure-coded fragments from all servers reporting a load below a threshold, while higher loaded servers will be replaced by least loaded servers from the possible N servers. The end result is that the server array is balanced because the K erasure-coded fragments are retrieved from servers loaded below the threshold.

[0083] In one embodiment, the assembling device does not know which of the servers store erasure-coded fragments related to the content to be retrieved, but the assembling device knows over how many servers (from the total number) the erasure-coded fragments are distributed. Therefore, the assembling device compensates for the infertile requests by enlarging the number of requests for erasure-coded fragments. Optionally, the requested servers are selected based on approximately random algorithm.

[0084] FIG. **10** illustrates one embodiment of different servers **698***a* to **698**(N) having different bandwidth capabilities of **683***a* to **683**(N) correspondingly. Assembling device **661** selects from which K servers, out of the possible N, to retrieve the fragments for segment reconstruction, wherein each server may have different unutilized bandwidth and different bandwidth capability. When many assembling devices assemble contents in parallel, while rejecting servers with a high load, the end effect is that the server array is approximately balanced and most servers can approach their maximal bandwidth capabilities. In one embodiment, the server array is balanced by enabling many assembling devices to select the least loaded servers.

[0085] Still referring to FIG. **10**, in the example, and assuming that K=3, servers **698***a*, **698**(N−1) and **698**(N) will be selected, as they have the highest unutilized bandwidth. In another example, the servers having the highest percentage of unutilized bandwidth will be selected.

[0086] In one embodiment, servers **698***a* to **698**(N) represent completely different types of server hardware, operating systems and capabilities, all put together in an array, and achieving load balance without the need for significant inter-server coordination. In one example, the fragments are distributed to at least two different classes of servers; the first class comprises high bandwidth CDN servers directly connected to the Internet backbone, and the second class comprises lower bandwidth CDN servers not directly connected to the Internet backbone.

[0087] In one embodiment, the servers are selected for fragment retrieval according to their unutilized fragment delivery bandwidth. For example, the servers report their unutilized bandwidth, and the assembling devices, or a control server, obtain the report and decide which servers to use for fragment delivery based on the unutilized bandwidth of each server.

[0088] In one embodiment, the servers are selected for fragment retrieval according to their ability to support additional fragment delivery load. For example, the servers report their ability to support additional fragment delivery loads. And the assembling devices, or a control server, obtain the report, and select the servers that report an ability to support additional fragment delivery loads.

[0089] In one embodiment, the assembling device, or a control server, looks for a pool of servers that may be used as replacements for servers that are loaded to a degree that does not allow continuation of fragment delivery. For example, the assembling device looks for potential unloaded servers, while retrieving fragments from other servers. The assembling device may sample relevant servers approximately randomly, and/or according to indications from a control server. The sampling process may comprise querying the potential server for load information, or measuring the latency or latency variance to the servers in order to estimate the current load on the server.

[0090] In one embodiment, it is desired to replace one or more servers by other servers for the delivery of erasure-coded fragments, wherein the replacement servers are selected using a second criterion from a pool of servers identified using a first criterion. For example, the first criterion for identifying the pool of replacement servers comprises looking for servers capable of increasing their fragment delivery throughputs, and the second criterion for selecting the replacement servers from the pool comprises selecting the best latency response server from the pool. In one example, the first criterion is a latency criterion, and the second criterion is a load criterion. In another example, the first criterion is a latency criterion, and the second criterion is a latency variance criterion. In another example, the second criterion is an approximately random selection. In one embodiment, a server selected using the second criterion is compared to the server to be replaced based on the second criterion. For

example, the second criterion is latency, and the replacing server, selected from the pool, has a smaller latency than the server it replaces.

[0091] In one embodiment, the server to be replaced is identified by comparing the actual performance level of the server with a threshold performance level. For example, when the compared performance is latency, a server having response latency above a certain threshold is replaced. In another example, the compared performance is the load on the server, which may be measured in terms of the amount of the unutilized fragment delivery bandwidth, or in terms of the percent of the server's unutilized fragment delivery bandwidth, or measured by any other appropriate technique.

[0092] In some embodiments, the assembling devices use a fragment pull protocol to retrieve the fragments and approach the servicing servers. In some embodiments, the assembling devices use a push protocol to obtain the fragments and approach the servicing servers, possibly by obtaining multiple sub-transmissions comprising fragment sequences.

[0093] FIG. **11** illustrates one embodiment of a fractional-storage system. Assembling device group **661***g* obtain erasure-coded fragments from the servers, such that the resulting outgoing bandwidth utilizations of each server in the array is **682***a* to **682**(N) correspondingly. FIG. **12** illustrates a case where server **698***b* has failed, its bandwidth capability **682***b***1** is zero, and is therefore unable to provide erasure-coded fragments. The assembling devices from group **661***g*, which previously obtained fragments from server **698***b*, may attempt to access it again for additional fragments, but are now unable to get a response. These assembling devices therefore obtain fragments from alternative servers. The end effect is that bandwidth **682***b* is now loaded on the still available servers, such that the total bandwidth **682***a***1** to **682**(N)**1** approximately increases by a total amount equal to **682***b*, optionally with no inter-server coordination, and simply by the fact that each assembling device selects alternative available servers for obtaining fragment on-the-fly. In one example, instead of obtaining from server **682***b***1**, the assembling devices obtain from the least loaded available servers. In one embodiment, a control server selects the alternative server/s for the assembling devices. In one embodiment, the assembling devices use a fragment pull protocol to obtain the fragments, and approach the alternative servers. In one embodiment, the assembling devices use a push protocol to obtain the fragments, and approach alternative servers, possibly by obtaining multiple sub-transmissions comprising fragment sequences. In this case, the sub-transmissions of the faulty server are discontinued and compensated for by other sub-transmissions from the alternative servers.

[0094] FIG. **13** illustrates an example similar to FIG. **12** with the difference that servers **698***a*, **698***b*, and **698***c* to **698**(N) reside within, or get serviced via, first, second, and third Internet backbone providers **300***j*, **300***i*, and **300***h* correspondingly. The group of assembling devices **661***g* is connected to the Internet via network **300***k*, which has access to all three backbones, such that communication between the assembling devices and servers **698***a* to **698**(N) pass via at least one of the backbones, or more. If server **698***b* is made unavailable to the assembling devices, optionally not due to a server failure, but rather due to congestion or a failure of the second Internet backbone provider **300***i*, assembling devices **661***g* compensate for the lost bandwidth by switching to the available servers on-the-fly. In one embodiment, networks **300***h*, **300***i*, and **300***j*, are different physical sub-nets of one network connected to the Internet. In one embodiment, the assembling devices are connected to networks **300***h*, **300***i*, and **300***j*, via network **300***k*, and then via one or more Internet Exchange Points ("IX/IXP").

[0095] FIG. **14** illustrates a few examples of retrieving fragments according to locality. In one example, the fractional-storage servers are connected to a data network or networks comprising the routers **201** to **209**. Assembling devices **235**, **237**, and **238** are connected to the same data network or networks, and K=3, meaning that any assembling device needs to obtain 3 erasure-coded fragments per segment from optionally 3 different servers out of the 10 in order to successfully reconstruct the segment.

[0096] Each assembling device tries to obtain erasure-coded fragments from fractional-storage servers that are closest to it topologically. In one embodiment, the topological distance is a function of the number of separating routers. Assembling device **238** can select three servers from groups **242**, **248** or **249**. According to the minimal path criterion, it retrieves the erasure-coded fragments from servers **399***h* to **399***i* of group **248**, since they are only one router **208** away. Groups **242** and **249** are three (**208**, **202**, **203**) and five (**208**, **202**, **203**, **201**, **209**) routers away, and are therefore not selected for retrieval. Similarly, device **237** selects three servers out of group **242**, and device **235** can select any three servers from groups **242** and **249**, since both are located four routers away.

[0097] In one embodiment, if topologically close servers do not respond to the assembling device, or report a bandwidth limitation, the assembling device will attempt to obtain an erasure-coded fragment from the next topologically closest server.

[0098] In one embodiment, the assembling device attempts to obtain erasure-coded fragments from servers featuring the lowest latency. Upon no response, for whatever reason, the assembling device will attempt to retrieve from the next lowest latency server. In one embodiment, the assembling device obtains information regarding the unutilized fragment delivery bandwidths of servers, and then attempts to retrieve from the lowest latency servers out of the servers having enough unutilized bandwidth. In one embodiment, the assembling device obtains information regarding the unutilized fragment delivery bandwidths of the servers, and then attempts to retrieve from the topologically closest servers out of the servers having enough unutilized bandwidth.

[0099] Still referring to FIG. **14**, in one embodiment the assembling devices select servers according to a latency criterion, such as selecting servers with the shortest time between fragment request and fragment delivery, or selecting servers having latency below a dynamic or static threshold. Assembling device **237** assembles content from servers **399***c*, **399***f*, **399***g*, and assembling device **235** assembles content from servers **399***b*, **399***c*, **399***g* (both use a mixture of servers from groups **242** and **249**). At a certain point in time, router **209** becomes congested or blocked, and prevents the erasure-coded fragments from servers **399***b* and **399***c* from arriving at assembling devices **235** and **237**, or causes the fragments to arrive with an increased delay. Therefore, assembling device **235** switches to three servers of group **242**, and assembling device **237** switches from server **399***c* to server **399***e*.

[0100] In one embodiment, the assembling device selects fractional-storage servers according to the following criterion: first, servers with adequate unutilized fragment delivery bandwidth are considered, then out of these, those with

latency below a threshold are considered, and out of these, the servers with minimal topological routing path are selected.

[0101] In some embodiments, the assembling devices use a fragment pull protocol to retrieve the fragments, and approach servers having low latency or low hop count as compared to other servers. In some embodiments, the assembling devices use a push protocol to retrieve the fragments, and approach servers having low latency or low hop count as compared to other servers, optionally by obtaining multiple sub-transmissions comprising fragment sequences.

[0102] In one embodiment, a plurality of unsynchronized retrieving assembling devices, which optionally use fragment pull protocol, choose the least loaded servers from which to retrieve the erasure-coded fragments. Optionally, the servers have almost no inter-communication between them and the load balancing calculation is performed by the retrieving assembling devices. Because the assembling devices can select the least loaded servers, the assembling devices manage the load balancing. When the erasure-coded fragments stored by the servers are unique erasure-coded fragments, the retrieving assembling device may retrieve erasure-coded fragments from any relevant server. Therefore, it may be enough for the retrieving assembling device to have indication of the load on its targeted servers, and retrieve enough erasure-coded fragments from the least loaded servers.

[0103] In one embodiment, a server signals the retrieving assembling device that it is close to its bandwidth limit and the assembling device searches for an alternative server. Optionally, the assembling device selects the server according to one or more of the following parameters: locality, cost, latency, or reliability. In one embodiment, the servers register their loads on a central server, and the assembling device selects the server to retrieve from, from the registered servers. In one embodiment, a central server, holding the loads of the various servers, determines for the assembling devices from which server to retrieve the erasure-coded fragments.

[0104] In one embodiment, the assembling devices measure the latency of the different servers in responding to fragment requests, and then use the latency information to estimate the loads on the servers. In one example, a high latency may indicate a high load on the server.

[0105] In one embodiment, the topological router hop count between an assembling device and fragment delivering servers is used to estimate the latency of the servers in responding to fragment requests.

[0106] In one embodiment, the latency of fragment delivering servers in responding to fragment requests by an assembling device is used to estimate the topological router hop count between an assembling device and the servers.

[0107] In one embodiment, the assembling devices perform several latency measurements for the different servers in responding to fragment requests, and then use the latency variance information to estimate the loads on the servers. In one example, a high latency variance may suggest a high load on server.

[0108] In one embodiment, the fractional-storage servers, from which the fragments are obtained for reconstructing a segment, are selected based on an approximately random selection algorithm from all of the servers storing the relevant fragments. In one example, an approximately random selection algorithm weighted according to the unutilized bandwidth of the servers is used for the approximately random selection of servers. The weighted random selection algorithm assigns servers with selection probabilities proportional to the amount of unutilized bandwidth for fragment delivery in each of the servers, such that the probability to select a server having a larger amount of unutilized bandwidth is higher than the probability to select a server having a lower amount of unutilized bandwidth.

[0109] The following embodiments describe processes for on-the-fly selection and re-selection of fractional-storage servers from which to obtain erasure-coded fragments.

[0110] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on the unutilized bandwidth of the servers, includes the following steps: (i) accessing data regarding servers storing relevant fragments (referred to as the relevant servers); (ii) accessing data regarding the unutilized bandwidth of the relevant servers. Optionally, the data is received by the assembling device from the relevant servers; and (iii) obtaining fragments from enough of the relevant servers having approximately the highest unutilized bandwidth; or obtaining fragments from enough of the relevant servers selected randomly and having unutilized bandwidth above a certain threshold.

[0111] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on latency, includes the following steps: (i) accessing data regarding the relevant servers; (ii) accessing data regarding the latencies from the relevant servers to the assembling device; and (iii) obtaining fragments from enough of the relevant servers having the lowest latencies; or obtaining fragments from enough of the relevant servers selected randomly and having latencies below a certain threshold.

[0112] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on bandwidth and latency, includes the following steps: (i) accessing data regarding the relevant servers; (ii) accessing data regarding the unutilized bandwidth of the relevant servers; (iii) identifying more than enough relevant servers having the most unutilized bandwidth; or randomly identifying more than enough relevant servers having unutilized bandwidth above a certain threshold; (iv) accessing data regarding the latencies from the identified servers to the assembling device; and (v) obtaining fragments from enough of the identified servers having the lowest latencies; or obtaining fragments from enough of the relevant servers selected randomly and having latencies below a certain threshold.

[0113] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on latency and bandwidth, includes the following steps: (i) accessing data regarding the relevant servers; (ii) identifying more than enough relevant servers having latencies to the assembling device below a certain threshold; or randomly identifying more than enough relevant servers having latencies to the assembling device below a certain threshold; (iii) accessing data regarding the unutilized bandwidth of the identified servers; and (iv) obtaining fragments from enough of the identified servers having the highest unutilized bandwidth; or obtaining fragments from enough of the relevant servers selected randomly and having the highest unutilized bandwidth.

[0114] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on locality, includes the following steps: (i) accessing data regarding the relevant servers; (ii) accessing data regarding the network topology distance (locality) from the relevant servers to the assembling device; and (iii) obtaining fragments from enough of the topologically closest relevant servers; or

obtaining fragments from enough of the relevant servers that are located in the same sub-network as the assembling device, or located in the closest sub-networks.

[0115] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on bandwidth and locality, includes the following steps: (i) accessing data regarding the relevant servers; (ii) accessing data regarding the unutilized bandwidth of the relevant servers; (iii) identifying more than enough relevant servers having the most unutilized bandwidth; or randomly identifying more than enough relevant servers having unutilized bandwidth above a certain threshold; (iv) accessing data regarding the network topology distance from the relevant servers to the assembling device; and (v) obtaining fragments from enough of the topologically closest relevant servers; or obtaining fragments from enough of the relevant servers that are located in the same sub-network as the assembling device, or located in the closest sub-networks.

[0116] In one embodiment, a method for selecting enough new servers from which to obtain fragments, based on latency and locality, includes the following steps: (i) accessing data regarding the relevant servers; (ii) identifying more than enough relevant servers having latencies to the assembling device below a certain threshold; or randomly identifying more than enough relevant servers having latencies to the assembling device below a certain threshold; (iii) accessing data regarding the network topology distance from the relevant servers to the assembling device; and (iv) obtaining fragments from enough of the topologically closest relevant servers; or obtaining fragments from enough of the relevant servers that are located in the same sub-network as the assembling device, or located in the closest sub-networks.

[0117] In one embodiment, a method for selecting enough new servers from which to obtain fragments is based on bandwidth, latency, locality, and, optionally, one or more additional relevant parameters. The method may weigh the different parameters in various ways, all of them are intended to be covered by the embodiments. For example, the method may include the following steps: (i) accessing data regarding the relevant servers; (ii) receiving data regarding the unutilized bandwidth latencies to the assembling device, and topology distances to the assembling device; (iii) weighting the received data and identifying a quantity of the most proper relevant servers, which can provide enough fragments to reconstruct content; and (iv) obtaining the fragments from the identified servers. In another example, the method may include the following steps: (i) accessing data regarding the relevant servers; (ii) identifying a set of more than enough relevant servers having the most unutilized bandwidth; or randomly identifying a set of more than enough relevant servers having unutilized bandwidth above a certain threshold; (iii) from the set, identifying a sub-set of more than enough relevant servers having latencies to the assembling device below a certain threshold; or randomly identifying more than enough relevant servers having latencies to the assembling device below a certain threshold; and (iv) obtaining fragments from enough of the topologically closest relevant servers out of the sub-set; or obtaining fragments from enough of the relevant servers out of the sub-sets, which are located in the same sub-network as the assembling device, or located in the closest sub-networks.

[0118] FIG. **15** (without the fragments marked with dashed lines) illustrates one example of distributing the erasure-coded fragments to 'M' CDN servers **399***a* to **399**(M), connected to a network **300**. Encoded fragments **310***a* to **310**(M) of a first segment are sent for storage in servers **399***a* to **399**(M) respectively. Similarly, erasure-coded fragments **320***a* to **320**(M) of a second segment are sent for storage in servers **399***a* to **399**(M) respectively. In addition, other erasure-coded fragments associated with other segments of other contents, illustrated as erasure-coded fragments **390***a* to **390**(M), are sent for storage in servers **399***a* to **399**(M) respectively. The number of unique erasure-coded fragments from each segment that are stored on the servers (**399***a* to **399**(M)) is equal to M in this example, where M may be smaller than the maximum number of unique erasure-coded fragments, meaning that only a subset of the potential erasure-coded fragments are actually stored. It is also possible to store the maximum number of unique erasure-coded fragments, or store more than one unique erasure-coded fragment per segment per server. The network **300** may be the Internet for example, or any other data network connecting multiple nodes, such as a private IP network, or a Wide Area Network ("WAN"). In one embodiment, the fragments marked with dashed lines illustrate one example where (N−M) additional servers are added to the array, and (N−M) new unique erasure-coded fragments per segment per content (**310**(M+1) to **310**(N), **320**(M+1) to **320**(N), and **390**(M+1) to **390**(N)) are generated and added to the array. In one embodiment, only M out of the maximum possible erasure-coded fragments (L) are actually generated for storage in the first place. In one embodiment, when the additional N-M erasure-coded fragments are needed for storage (e.g., when additional servers are made available), the remainder of the N-M erasure-coded fragments are actually generated. Any time that additional unique erasure-coded fragments are needed, this process of calculating the additional erasure-coded fragments is repeated, up to the point that all L possible erasure-coded fragments are used.

[0119] In one embodiment, and especially when using rateless coding, L may be chosen as a sufficiently large number to account for any realistic future growth of the server array. For example, a segment of 96 Kbytes is expanded using a rateless code with a ratio of 1 to 2^16 original symbols to encoded data, into an encoding symbol of potential size 6.29 GBytes. Assuming a 1500 Bytes erasure-coded fragment size, then potentially 4.19 million unique erasure-coded fragments can be generated. Now, it is safe to assume that for all practical uses, the server array will not grow to more than 4.19 million nodes, and may contain several thousands of servers, meaning that the encoded data can be used in all cases where additional unique erasure-coded fragments are needed, by generating new erasure-coded fragments out of the segment. Optionally, a server may store erasure-coded fragments for only some of the segments.

[0120] In one example of redundancy factor and storage gain (without the fragments marked with dashed lines), server **399***a* stores only erasure-coded fragment **310***a* from a first segment, erasure-coded fragment **320***a* from a second segment, and erasure-coded fragment **390***a* from a third segment. Assuming that: (i) the segment size is 1024 Kbytes; (ii) the segment is encoded using erasure code into a 4096 KByte encoded segment; (iii) the encoded segment is segmented into 256 erasure-coded fragments of size 4096/256=16 KByte; and (iv) the erasure-coded fragments are stored on 256 servers (M=256); it turns out that each server stores only a 1/64 portion of the original size of the segment. This means that each server can manage with only 1/64 of the storage require-

ments in comparison to a situation where it had to store the entire segment. In addition, there are 256 erasure-coded fragments altogether from each encoded segment, meaning that an assembling device that is assembling the erasure-coded fragments from the servers need only select slightly more than 64 erasure-coded fragments in order to completely reconstruct the segment, and it can select whichever slightly more than 64 erasure-coded fragments it desires out of the 256 possibly available. The redundancy factor in this example is approximately 256/64=4. All contents in this example enjoy a factor of 64 in storage gains, meaning that server **399***a*, for example, stores only 1/64 of the information associated with the first segments and any additional segments belonging to other contents. In one example, each server supports high volume storage of between about 500 GByte and 500 TBytes, optionally utilizing hard drive, Solid State Drive, or any other high volume storage device(s). In these cases, each server may store many millions of erasure-coded fragments, associated with millions of segments, belonging to hundreds of thousands of different contents, and possibly more.

[0121] In one embodiment, new content initially encoded with a low redundancy factor is distributed to an initial number of fractional-storage servers. As the content is distributed to more servers, additional unique fragments are encoded and therefore the redundancy factor increases. Optionally, as the content's popularity increases, and/or as the load on the fractional-storage servers increases, the redundancy factor is increased, and vice versa.

[0122] In one embodiment, multiple unique erasure-coded fragments per segment of a new content are distributed to an initial number of fractional-storage servers with a low storage gain (i.e. each server stores multiple unique erasure-coded fragments per encoded segment). As the content is distributed to more fractional-storage servers, some of the erasure-coded fragments stored on the initial number of fractional-storage servers are removed and thereby the storage gain is increased. Optionally, as the demand for the content increases, the storage gain is decreased, and vice versa.

[0123] FIG. **16** illustrates three examples (each depicted by one of the columns A-C) of changing the redundancy factor according to the demand. Column A illustrates one simplified example of a storage array including 16 servers (**1001** to **1016**). Each server stores up to 2 different erasure-coded fragments, and can service an erasure-coded fragment transmission bandwidth of up to B. Assuming three contents (#1, #2, and #3) processed to segments and erasure-coded fragments with a storage gain of 4.

[0124] Assuming content #1 is the most popular, and requires a peak bandwidth of 11×B. Since each server can service up to bandwidth B, at least 11 servers are needed to service content #1 bandwidth requirements. Content #1 is therefore encoded into 11 unique erasure-coded fragments per segment, illustrated as group g**1** of erasure-coded fragments stored on servers **1001** to **1011**. Out of these 11 erasure-coded fragments, it is sufficient to obtain slightly more than 4 erasure-coded fragments in order to reconstruct a segment of content #1. Therefore, the resulting redundancy factor of the stored fragments associated with content #1 is approximately 11/4=2.75. Content #2 requires less bandwidth, and manages with a peak of 7×B. It is therefore encoded into 7 unique erasure-coded fragments per segment, illustrated as group g**2** of erasure-coded fragments on servers **1010** to **1016**. Therefore, the redundancy factor of the stored fragments associated with content #2 is 7/4=1.75. Content #3 requires a peak bandwidth of 5×B, but for some reason (for example, being a more critical content), it is encoded into 14 erasure-coded fragments per segment, illustrated as group g**3** of erasure-coded fragments on servers **1001** to **1009** and **1012** to **1016**. Therefore, the redundancy factor of the stored fragments associated with content #3 is 14/4=3.5. This concludes the storage availability of the servers in this example, as every server stores two erasure-coded fragments.

[0125] Column B illustrates an example where content #2 becomes more popular than content #1, and therefore requires more bandwidth and hence more of a redundancy factor. This is achieved by eliminating **5** erasure-coded fragments associated with content #1 that were previously stored on servers **1001** to **1005**, and replacing them with 5 new unique erasure-coded fragments g**4** associated with content #2. This brings the total number of erasure-coded fragments per segments of content #1 and #2 to 6 and 12 respectively. In column C, new content #4 is stored on servers **1001** to **1003** and **1014** to **1016** (illustrated as g**5**), by eliminating **3** erasure-coded fragments of content #1 and 3 erasure-coded fragments of content #2.

[0126] Throughout the examples of FIG. **16**, a record of "what erasure-coded fragments are stored where" may be: (i) kept in each of the servers **1001** to **1016**. In this case, when an assembling device is assembling content #2, it will send a query to servers **1001** to **1016**, asking which one is storing erasure-coded fragments of content #2; (ii) kept in a control server. In this case, an assembling device will ask the control server to send back a list of all servers storing erasure-coded fragments of its required content.

[0127] In one embodiment, different quantities of erasure-coded fragments are generated per different segments. In one embodiment, some segments store data that is considered more important than data stored in other segments, and relatively more erasure-coded fragments are generated from the segments storing the more important data than from the segments storing the less important data.

[0128] In some embodiments, the content is segmented into a plurality of segments to enable beginning to play the content as it is being obtained, and optionally enable trick play. The different segments may or may not be of the same size.

[0129] The following embodiments discuss different methods for segmenting the content. In one embodiment, at least one portion of the content is segmented into multiple segments in sizes within a first size range, and the remainder of the content is segmented into a plurality of segments in sizes within a second size range (additional size/s may be added similarly). The sizes included in the second size are larger than the sizes included in the first size range. Pluralities of erasure-coded fragments are generated from each of the segments. The segments of sizes within the first size range are better suited for fast retrieval, and the segments of sizes within the second size range are better suited for high-gain storage. In one example, the segments in sizes within the first size range belong to approximately the beginning of the content. In one example, the segments in sizes within the first size range belong to locations within the content requiring trick play access. In one embodiment, the segments of the first type are encoded into fewer fragments than the segments of the second type. This allows a fast retrieval of the shorter segments.

[0130] The term "fragment pull protocol for high latency" as used herein denotes a protocol enabling an assembling device to request one or more fragments from one or more

providing sources, wherein the time to transmit the one or more fragments in response to the assembling device request, through the slowest communication link connecting the responding source and the assembling device, is smaller than the round trip communication delay between the assembling device and the responding source, excluding the processing time of the providing source. For example, if the round trip communication delay between Israel and the USA is about 200 ms, the assembling device requests one fragment sized about 1500 bytes, and the slowest communication link is an ADSL line connecting the assembling device at 1.5 Mbps, then the time it takes to transmit the requested fragment through the slowest communication link is about 1500*8/1500000=8 ms, which is much smaller than the round trip delay.

[0131] In one example, the content **100** is a 1 GByte encoded H.264 file, storing a 2-hour motion picture, and is segmented into approximately 10,000 segments of approximately 100 Kbytes each. In another example, the content **100** is a 4 MByte web-site information (HTML, FLASH, or any other combination of information that encodes the presentation of a website), and is segmented into 4 segments of approximately 1 MByte each.

[0132] In one example, the content supports streaming presentation, and the segments are small enough to enable presentation shortly after beginning the reception of the first segment(s). For example, each segment may include 96 KByte, allowing a 5 Mbps receiver to download the segment in approximately 0.2 seconds, and optionally begin the presentation shortly thereafter. In one embodiment, the time to play is reduced by segmenting certain portions of the content into smaller segments, while the remaining portions are segmented into larger segments. A smaller segment can be retrieved faster, while a larger segment may be better optimized for storage gain and/or efficient transmission.

[0133] In one embodiment, the short segments are 96 Kbytes in size, and the long segments are 960 Kbytes in size. The redundancy factors used for encoding short and long segments into fragments are 100 and 5 respectively. 1500 Bytes fragments are used for both sizes. The short segments are therefore encoded into (96K/1500)×100=6,400 fragments, from which only about 64 are needed for reconstruction, and the long segments are encoded into (960K/1500)×5=3,200 fragments, from which only about 640 are needed for reconstruction. Short segments are reconstructed more quickly than long ones, as they require fewer fragments to be decoded. Optionally, each fragment is stored on a different server, resulting in a storage gain of 64 for short segments, and 640 for long segments.

[0134] FIG. **17** illustrates one example in which the content **100** is segmented into segments, such that the first segment **104***a* is smaller than the consecutive segment **104***b*, which is smaller than following segments **104***c* and **104***d*. In another example, the content **100** is segmented into segments, such that the first several segments (e.g. **104***aa* and **104***bb*, which are the same size), are smaller than consecutive segments (e.g. **104***cc* and **104***dd*, which are the same size).

[0135] FIG. **18** illustrates one example in which the content **100** is segmented into cyclic sets of successive segments increasing in size. For example, **105***b* is equal or larger in size than **105***a*, and so on, up to segment **105***d*; **105***f* is equal or larger in size than **105***e*, and so on, up to segment **105***h*. In one example, segment **105***e* is equal in size to segment **105***a*. Point **105**EP represents the ending of the first set, and the beginning of the second set.

[0136] In one embodiment, the segments are created on-the-fly, such as during a live event or when the content is made available to the segmentation process as an on-going stream. In one embodiment, the content supports streaming presentation, and the segments are of the small size, to enable content presentation shortly after beginning the reception of the first segment (or any other segment). In addition, the erasure-coded fragments are kept as small as possible, while still enabling efficient transport over an IP network. For example, each erasure-coded fragment is about 1500 Bytes and can be transported using one IP packet.

[0137] It is to be noted that streaming content may also be manifested as an intermediate product of a process. For example, in a case where a video camera outputs erasure-coded fragments that can be decoded into streaming content, the intermediate data from which the erasure-coded fragments are generated is considered to be streaming content (even if the video camera does not output that intermediate data). Moreover, streaming content may include: content that is produced and then immediately transmitted to a receiving server, content that is produced but stored for any length of time before being transmitted to a receiving server, content that is transmitted to a receiving server and then immediately sent from the receiving server to a client, content that is transmitted to a receiving server, then buffered for some time at the receiving server and then sent from the receiving server to a client, content that is solely played at a client, and content that is manipulated or changed or reacted to at the client while a continuation of the content is still being played at the client.

[0138] FIG. **19** illustrates one embodiment of a server array including fractional-storage servers **399***a* to **399**(N) storing erasure-coded fragments **390***a* to **390**(N) associated with content. In order for assembling device **661** to reconstruct a segment **101***a* of the content, it has to retrieve at least K erasure-coded fragments. In one example, k=4 and the assembling device **661** chooses approximately randomly from which servers to retrieve the 4 different erasure-coded fragments. It chooses to retrieve fragments **390***a*, **390***c*, **390**(N−1) and **390**(N), which are noted as group **573**, and reconstruct the segment **101***a*. Consequent segments of the content are reconstructed in a similar fashion, and the content may eventually be fully retrieved by combining all relevant segments. If the assembling device **661** cannot reconstruct the segment **101***a*, it retrieves one or more additional unique erasure-coded fragments, and tries again.

[0139] Referring back to FIG. **19**, in one embodiment, the content being distributed supports stream presentation, and segment **101***a* is of small size, to enable content presentation by assembling device **661** shortly after beginning the reception of the segment (or any other segment of the content). For example, segment **101***a* is 96 KByte, allowing a 5 Mbps download speed receiver to obtain the entire segment (by requesting enough erasure-coded fragments to enable the reconstruction of the segment, and such that the total size received of all requested erasure-coded fragments is slightly larger than the segment) after approximately 0.2 seconds from request, and beginning the presentation shortly or right after the successful decoding and reconstruction of segment **101***a*.

[0140] The following embodiments describe processes for on-the-fly erasure-coded fragment retrieval from fractional-storage servers.

[0141] In one embodiment, a method for obtaining erasure-coded fragments from fractional-storage servers to reconstruct a segment includes the following steps: (i) identifying the next segment to be obtained; optionally, the segments are approximately sequential segments of streaming content obtained according to their sequential order; (ii) optionally, determining the minimum number of fragments needed to reconstruct the segment; (iii) are enough identified relevant servers (i.e. servers storing the required fragments) available from the process of obtaining prior segment/s? (iv) if no, identifying enough relevant servers; (v) if yes, requesting enough fragments from the identified relevant servers; if less than enough fragments are obtained from the identified relevant servers, go back to step iv and identify additional relevant server/s; (vi) reconstruct the segment from the obtained fragments; and (vii) optionally, go back to step i to obtain the next segment.

[0142] In one embodiment, a method for obtaining erasure-coded fragments from fractional-storage servers to reconstruct multiple segments includes the following steps: (i) identifying multiple segments to be obtained, optionally according to their sequential order; (ii) optionally, determining the minimum number of fragments needed to reconstruct the segment; (iii) optionally, determining the number of fragments to be obtained approximately in parallel; (iv) are enough identified relevant servers available from the process of obtaining prior segment/s? (v) if no, identifying enough relevant servers; (vi) if yes, requesting enough fragments from the identified relevant servers, optionally in parallel and according to the sequential order of the segments; (vii) if less than enough fragments are obtained from the identified relevant servers, go back to step iv and identify additional relevant server/s; (viii) reconstructing the segment/s from the obtained fragments; and (ix) optionally, go back to step i to obtain the next segments.

[0143] In one embodiment, a method for obtaining erasure-coded fragments from fractional-storage servers to reconstruct a segment in a burst mode includes the following steps: (i) identifying the next segment to be obtained; (ii) optionally, determining the minimum number of fragments needed to reconstruct the segment; (iii) are more than the minimum number of relevant servers available from the process of obtaining prior segment/s? (iv) if no, identifying more than the minimum relevant servers; (v) if yes, requesting more than the minimum number of fragments needed to reconstruct the segment; if less than enough fragments are obtained, go back to step iv and identify additional relevant server/s; (vi) reconstructing the segment from the obtained fragments; and (vii) optionally, go back to step i to obtain the next segment.

[0144] The various methods for obtaining erasure-coded fragments from the fractional-storage servers for reconstructing one or more segments may be combined as needed. In one example, the initial segment/s are obtained using a burst mode and the following segments are retrieved without requesting extra fragments. In another example, the initial segment/s are obtained approximately in parallel and optionally using a burst mode, and the following segments are obtained one by one and optionally without requesting extra fragments. The fragments may be obtained using a pull protocol and/or a push protocol. Moreover, the servers from which to retrieve the fragments may be selected according to one or more of the various discussed methods for selecting the servers and/or load balancing the servers.

[0145] In some embodiments, the fragments are small enough to be contained in one packet. In one embodiment, each fragment is about 1400 bytes, and can fit into one UDP or RTP packet transmitted over Ethernet. The stateless nature of UDP and RTP allows the servers to send one packet with one fragment very quickly, without the need for any acknowledgement or hand shaking In some embodiments, the fragment pull protocol requests use one stateless packet, like UDP or RTP. In one embodiment, the assembling device requests about 100 fragments approximately in parallel, using 100 separate requests or one or few aggregated requests. About 100 servers respond by sending about 100 fragments, each encapsulated in one stateless packet, after a short delay, and the assembling device receives the fragments within a fraction of a second. Assuming an Internet round trip delay of 100 ms, and server processing latency of 100 ms, then after 200 ms the assembling device starts receiving all 100 fragments. With a modem of 5 Mbps, and assuming 1400 bytes per fragment, all 100 fragments are received 1400×100×8/5 Mbps=224 ms after the initial delay, meaning that content can be presented 200+224=424 ms after request (decoding and other process time has been ignored in this example).

[0146] FIG. **20** illustrates one embodiment of real time streaming content retrieval from fractional-storage servers, wherein erasure-coded fragments **720***a* to **720**(K) are retrieved in a fast cycle, meaning that several erasure-coded fragments are obtained approximately in parallel. As a result, the interval T**2** minus T**1** is more or less limited only by the download bandwidth of the assembling device's modem. Referring to the example of FIG. **21**, T**2** minus T**1** can be reduced from 0.77 seconds to 0.15 seconds, if the modem operates at 5 Mbps (instead of 1 Mbps).

[0147] In one embodiment, T**1** to T**2** represents a fragment fetch cycle that corresponds to the beginning of streaming content to be presented (in that case, segment **710***a* is the first segment of the content, and presentation **700** corresponds to the beginning of the streaming content), or corresponds to a certain point within the streaming content to be presented starting this point onwards (in that case, segment **710***a* is a segment within the content, and presentation **700** corresponds to playing the content starting not from the beginning, but rather from segment **710***a*, located somewhere within the content). This is also known as trick play. In one embodiment, erasure-coded fragments **720**(*a*) to **720**(K) are obtained such as to result in approximately a maximum utilization of the download capabilities of the assembling device, and such that the rate of requesting erasure-coded fragments results in a data arrival rate that on average utilizes the assembling device's maximum download bandwidth.

[0148] For example, if each erasure-coded fragment is 1500 Bytes, and the maximal download rate of the assembling device is 5 Mbps, then erasure-coded fragments **720**(*a*) to **720**(K) are requested at an average rate of a 5 Mbps/(1500 Bytes×8 bit/byte)=approximately 417 erasure-coded fragment requests per second. This results in a fast retrieval time for segment **710***a*. Consequent segments, such as **710***b* can be retrieved either in the same manner as segment **710***a*, or by sending fragment requests at a lower rate and, possibly, at a rate that results in a fragment data reception bandwidth that approximately equals the average content presentation rate, or by sending fragment requests at a rate that results in a

fragment data reception bandwidth that is higher or slightly higher than the average content presentation rate for accumulating a content buffer throughout the presentation of the content.

[0149] In one embodiment, more than the minimum number of unique erasure-coded fragments needed to correctly reconstruct a segment are requested per segment, such that even if some fragment requests are not followed by actual fragment reception, the segment can still be reconstructed provided that at least the minimum number of unique erasure-coded fragments are actually received.

[0150] In one embodiment, the fragment pull protocol request includes a priority indication. A high priority indication means that the servers should give a preference to responding with a fragment transmission. High priority requests are served before other requests. Optionally, high priority requests are served even if the server's bandwidth quota is exceeded. In one embodiment, the high priority requests are used by the assembling devices for receiving priority in the reception of the first segment, or several first segments, in order to facilitate fast starting of content presentation after content request by the user (either when starting to play a content, or in trick play mode, when starting to play a content from a certain point).

[0151] FIG. **22** illustrates one embodiment of a fragment pull protocol. Assembling device **861** (also represented by protocol diagram element **810***b*) obtains erasure-coded fragments from fractional-storage servers **899***a* to **899**(N) (also represented by protocol diagram element **898**), utilizing the following steps: (i) deciding **810***a* which segment to retrieve; (ii) device **861** sending requests to some of the fractional-storage servers for erasure-coded fragments associated with the desired segment. For example, requests **880***a* to **880**(K) for erasure-coded fragments **890***a* to **890**(K), from servers **899**(*a*) to **899**(K), correspondingly; and (iii) the servers respond by sending the requested erasure-coded fragments. For example, servers **899***a* to **899**(K) send **881***a* to **881**(K) erasure-coded fragments **890***a* to **890**(K) to device **861**. The fragment request and receipt process begins at T**1***c* and ends at T**1***d*. At time T**1***d*, device **861** has enough erasure-coded fragments (K) to reconstruct the segment selected at **810***a*. In one embodiment, the process from T**1***c* to T**1***d* occurs in real time, in support of streaming content presentation.

[0152] FIG. **23** illustrates a similar process to FIG. **22**, where request **890***b* fails to result in a reception of erasure-coded fragment **890***b* for any reason (such as a server fault, network congestion, or abnormal latency conditions). Assembling device **861** therefore issues another request **882**(K+1) for erasure-coded fragment **890**(K+1) in response, and receives **883**(K+1) the additional erasure-coded fragment **890**(K+1) needed to reconstruct the segment.

[0153] FIG. **24** illustrates a similar process to FIG. **22**, where one or more extra erasure-coded fragments (in addition to the needed K) are requested in advance (illustrated as request **880**(K+1) for erasure-coded fragment **890**(K+1)), such that if, as an example, request **890***b* fails to result in a reception of erasure-coded fragment **890***b*, assembling device **861** does not have to request new erasure-coded fragments to reconstruct the segment, since there are still at least K erasure-coded fragments that were successfully received and therefore the segment can be reconstructed.

[0154] FIG. **25** illustrates a similar process to FIG. **22**, where requests for erasure-coded fragments are loaded into one aggregated request **870**, that is sent to one of the fractional-storage servers (the receiving server is illustrated as protocol diagram element **888***a*, and will be also referred to as a "relay server"). In one example, if the relay server is **899**(N), then, it will forward the request to additional servers **899***a* to **899***c* (protocol element **888***b*) via new requests **870***a* to **870***c* (on behalf of assembling device **861**). Servers **899***a* to **899***c* will then respond by sending the erasure-coded fragments **890***a* to **890***c* (**871***a* to **871***c*) to the assembling device **861**. Server **899**(N) will send **871**(N) fragment **890**(N) to the assembling device.

[0155] In one embodiment, the aggregated request **870** contains a list of servers to be approached by the relay server **888***a*. In one embodiment, the aggregated request **870** does not contain a list of servers to be approached, and it is up to the relay server **888***a* to select the servers. In one embodiment, the relay server **888***a* is not necessarily one of the fractional-storage servers, and may be a control server, or other network device like a router, or an assembling device.

[0156] Still referring to FIG. **24**, in one embodiment, more fragments than needed to reconstruct a segment are requested, such that the additional requested fragments approximately compensate for fragment failure conditions. If, statistically, F fragment requests are expected not to result in the reception of a fragment (i.e. fragment loss), out of a total number of K+F fragment requests (wherein K is the minimal number of fragments needed to reconstruct a segment), then it is possible to request K+F fragments instead of just K. In one embodiment, more than K+F fragments are requested, since the quantity of the received fragments is a statistical variable. In this case, K+F+S fragments are requested, wherein S is a safeguard amount of additional requests to assure that at least K fragments are received. In one embodiment, the fragment loss F changes over time, and the assembling device handles the change by increasing or decreasing the number of fragments requested per segment. In one embodiment, the assembling device may determine F based on previous fragment failure rates.

[0157] In one embodiment, requesting K+F+S fragments for a segment will almost always result in the reception of at least K fragments, and therefore the assembling device may request K+F+S without being concerned about which fragment has not arrived, and without trying to actively compensate for fragment failures by issuing additional fragment requests. In this case, the assembling device requests the fragments in an "open loop" fashion, meaning that it requests the K+F+S fragments, and moves on to another segment. In one embodiment, even when requesting K+F, or K+F+S fragments per segment, it is still possible not to receive the needed K fragments. Therefore, the assembling device may compensate for undelivered fragments by issuing additional fragment requests (a "closed loop" operation).

[0158] In one embodiment, the K+F, or K+F+S fragment requests are issued approximately in parallel, in order to achieve the fastest response possible for reconstructing a segment. In this case, the fragments start to arrive at the assembling device a short while after being requested, such that as soon as at least K out of the requested fragments arrive, the assembling device may immediately proceed with reconstructing the segment.

[0159] In some embodiments, a push protocol is used to obtain fragments. A push protocol may be implemented using one transmission carrying fragments from a source server to a destination receiver, or may be implemented using a plurality of sub-transmissions. When using sub-transmissions, each

sub-transmission transports a fraction of the fragments needed for segment reconstruction. Segments may be reconstructed from fragments received via sub-transmissions after obtaining decodable sets of erasure-coded fragments; optionally one set per segment. A sub-transmission may be transported using an IP stream such as RTP, an HTTPS session, or any other protocol suitable for transporting a sequence of fragments between a source server and a destination assembling device.

[0160] FIG. **19** illustrates one embodiment, in which content is segmented and erasure-coded. Fragments **390***a* to **390**(N), belonging to a first segment, are distributed to servers **399***a* to **399**(N) respectively. Other fragments belonging to subsequent segments are similarly distributed to servers **399***a* to **399**(N). The servers may use a push protocol to transport the fragments to an assembling device. A push protocol sub-transmission may comprise a sequence of fragments associated with multiple segments. In one example, the fragments are ordered according to the sequential order of the segments in a streaming content. Server **399***a* sends a first sub-transmission to a destination assembling-device. Optionally, the first sub-transmission comprises a sequence of fragments starting with fragment **390***a*, associated with the first segment, and continuing with fragments belonging to subsequent segments. Server **399***c* sends a second sub-transmission to the destination assembling-device, optionally starting with fragment **390***c*, associated with the first segment, and continuing with fragments belonging to subsequent segments. In a similar fashion, servers **399**(N−1) and **399**(N) send additional sub-transmissions to the destination assembling-device, each comprising a unique fragment sequence.

[0161] When using a push transmission, the assembling device does not explicitly ask for each fragment, but instead instructs each of the different servers to start sending it a fragment sequence using a sub-transmission. The destination assembling-device receives the sub-transmissions sent by servers **399***a*, **399***c*, **399**(N−1) and **399**(N). It gathers **573** the first fragment from each sub-transmission to reconstruct the first segment **101***a*. In a similar fashion, additional fragments belonging to subsequent segments are obtained from the sub-transmissions, and used to reconstruct the segments. It is noted that any combination of sub-transmissions may be used, as long as a decodable set of fragments is obtained per each segment. It is also noted that FIG. **19** illustrates a non-limiting embodiment and a sub-transmission may include two or more unique erasure-coded fragments per segment.

[0162] In one embodiment, the push sub-transmissions is synchronous (all servers sending the fragments of each segment at approximately the same time). In another embodiment, the push sub-transmission is asynchronous and the arrival of different fragments associated with a specific segment at the assembling device side may be spread over a long period. This may occur, as an example, when some push servers are faster than others. In one embodiment using asynchronous sub-transmissions, the assembling device aggregates whatever fragments it can before presentation time of each segment, and then optionally supplements fragments using a pull retrieval process. A server that does not send fragments fast enough, and therefore usually causes supplemental requests, may be ordered to stop the sub-transmission. Another server may be requested, optionally by the assembling device, to replace the slow server by initiating a new sub-transmission.

[0163] In one embodiment, the push-transmissions carry more erasure-coded fragments than needed for segment reconstruction. In one embodiment, the push transmissions carry fewer erasure-coded fragments than needed for segment reconstruction, and the remaining fragments are pulled by the assembling device.

[0164] In one embodiment, a method includes the steps of obtaining, by an assembling device from fractional-storage servers, erasure-coded fragments associated with a segment of streaming content; detecting a first quantity of at least one fragment associated with the segment, which have failed to arrive at the assembling; requesting via a fragment pull protocol for high latency approximately the first quantity fragments; and repeating the steps of detecting and requesting until enough fragments have been obtained for reconstructing the segment. Optionally, the steps are repeated approximately sequentially on the segments. Optionally, the streaming content comprises approximately sequential segments. Optionally, detecting the failure comprises determining which fragment has failed after not obtaining the fragment within a predetermined period from issuing its request, and/or receiving a message that does not contain the actual fragment's payload. And optionally, the fragments are obtained via sub-transmissions transmitted by the servers.

[0165] In one embodiment, a method for retrieving erasure-coded fragments includes the steps of: determining, by an assembling device, the time remaining to reconstruct a segment of streaming content; estimating the probability of receiving a sufficient quantity of already-ordered erasure-coded fragments to reconstruct the segment during the remaining time; and if the estimated probability is below a predefined threshold, issuing one or more additional fragment requests, using a fragment pull protocol, until the estimated probability equals or passes the predefined threshold. Optionally, the already-ordered fragments and the additional fragment requests are received from fractional-storage CDN servers. Optionally, the already-ordered fragments are received via at least two sub-transmissions transmitted by the servers. Optionally, the already-ordered fragments are received via a fragment pull protocol. And, optionally, estimating the probability includes considering the probability of each request to result in a fragment reception during the remaining time.

[0166] In one embodiment, the assembling device may aggregate several fragment requests into one message. The aggregated message is then sent to a fractional-storage server, possibly in a payload of a single packet, and optionally in order to conserve outgoing bandwidth and/or to reduce the number of packets needed to convey the requests. The fractional-storage server may then read the aggregated message and act accordingly by sending a plurality of fragment responses to the assembling device. The fragment responses may include one fragment at each payload, as is the case of responding to a single fragment request, or it may include an aggregated response including multiple fragments at each payload.

[0167] In one embodiment, fragment aggregation is used for retrieving fragments associated with segments of streaming content, and each aggregated message requests fragments associated with a specific segment. For example, three fractional-storage servers store together 12 fragments associated with a certain segment, such that each stores four fragments. The assembling device needs the 12 fragments in order to reconstruct the segment and therefore issues three aggregated

fragment request messages—one for each server. The three servers receive the aggregated request messages, and each server responds by sending its four fragments to the assembling device. Therefore, only three aggregated request messages were needed to retrieve the 12 fragments. The assembling device may request fragments associated with the next segment(s) in a similar manner using additional aggregated requests, optionally until receiving all required segments.

[0168] FIG. **26** illustrates various examples of aggregated fragment request messages. In one example, the aggregated fragment request messages **503**, which may be transported using one packet payload, uses a format (or a data structure) comprising a segment identification **802** and the number-of-fragments **803** requested by the assembling device. The receiving fractional-storage server uses the segment identification information to locate the relevant fragments associated with the identified segment, and then uses the number-of-fragments parameter **803** to determine how many fragments, out of the located fragments, should be transmitted to the requesting device as a response.

[0169] In one embodiment, the fragments are erasure-coded fragments and the fractional-storage servers store unique erasure-coded fragments. The assembling device receives multiple erasure-coded fragments from multiple servers, such that the number of received fragments is at most the sum of all number-of-fragments **803** values as has appeared in all of the aggregated requests messages. In this case, the various fractional-storage servers need no inter-coordination to respond to message **503**, as the assembling device does not care which of the fragments associated with the identified segment were received, as long as at least the requested number of unique erasure-coded fragments were received. In some embodiments, aggregated fragment request messages and single fragment requests are used concurrently.

[0170] In another example, an aggregated fragment request message **502** further comprises a content identification field **801**. In still another example, an aggregated fragment request message may comprise requests for fragments associated with different segments of streaming content. In this case, and according to one example, the aggregated request **505** comprises a sequence of identified segments **902** containing the identification of all segments for which the assembling device requests fragments. Optionally, in the absence of additional information in the aggregated message, the fractional-storage server may assume that one fragment per each of the segments identified in sequence **902** is required. In this case, the server will locate such fragment per each of the identified segments, and will send them to the requesting device. The requesting device may include information regarding how many fragments are required per identified segment, as a number-of-fragments parameter **903**. The number-of-fragments **903** may be a scalar value that indicates how many fragments are requested per each identified segment, or it may be a vector value, indicating the number of required fragments per each of the identified segments in the sequence.

[0171] In one embodiment, the fractional-storage server responds to a message, comprising aggregated requests for fragments associated with multiple segments, by sending all of the requested fragments to the requesting device, or by sending all of the requested fragments to the requesting device in a certain order. The order may follow the sequential order of the segments in streaming content. In one example, the fractional-storage server first sends the fragments associated with the first identified segment, and then sends the fragments associated with the next identified segments. Packet payload **505** illustrates one example of an aggregated fragment request message comprising a transmission delay **909** instructing the fractional-storage servers to incorporate intended delays while transmitting the different fragment responses. In one example, the transmission delay **909** sets the desired time delay between transmission of each group of fragments associated with the segments identified in the sequence **902**. In this case, the fractional-storage server responds to the aggregated request message by transmitting a sequence of fragments, associated with the identified segments, at a duty cycle determined by the transmission delay **909**. In one example, the segments belong to streaming content and the effective rates at which the servers transmit their responses are controlled using the transmission delay **909**.

[0172] In one embodiment, the fragments are erasure-coded fragments and the assembling device uses multiple aggregated fragment request messages for obtaining the required content. Each message comprises multiple fragment requests associated with one sequence of segment(s) and addressed to a different fractional-storage server storing the relevant fragments. Each such sequence of segments may be referred to as a portion of streaming content, whereby the assembling device uses multiple aggregated messages to obtain each portion of the streaming content at a time. In one embodiment, the assembling device uses a wireless interface, such as WiFi, to connect to the Internet and communicate with the fractional-storage servers, and the fragment request aggregation techniques may dramatically reduce the number of time such an assembling device needs to gain access to the outgoing wireless interface. Moreover, the fragment request aggregation techniques may be combined with many of the disclosed embodiments for retrieving erasure-coded fragments.

[0173] Still referring to FIG. **26**, in one embodiment, requests for fragments are transmitted via an IP network, in the form of packet payloads. The packet payload may be, as an example, the payload of a UDP packet carried over IP. In one embodiment, packet payload **501** contains a fragment request comprising content identification **801** and segment identification **802**. A server receiving such a request uses the content and segment identifications to locate a relevant erasure-coded fragment, and transmits it back to the requester. Optionally, if no references are made as to how many fragments are requested per the identified segment, the server may assume that only one fragment is requested.

[0174] In one embodiment, the fragment responses are transported over an IP network, using packet payloads. In one example, packet payload **701** includes an actual requested fragment payload **602**, and, optionally, information regarding the segment **601** to which the fragment payload belongs. The segment information may be needed if the requester retrieves fragments associated with more than one segment, and therefore it must know to which segment the fragment payload belongs. In one example, the fragment response is transported over UDP/IP, or TCP/IP, such that the payload **701** is a UDP or TCP payload.

[0175] In one embodiment, multiple segments of content, which, in one example, is streaming content, are reconstructed by an assembling device retrieving multiple erasure-coded fragments associated with the multiple segments. Since a fragment request does not always result in a reception of the fragment, some requested fragments may fail to arrive at the assembling device. Therefore, the assembling device

checks (from each of the segments for which fragments have already been requested) which requested fragments have failed to result in a correct reception of a fragment. For each such failure, the assembling device issues an additional request for a fragment. The additional requests are associated with segments for which fragments have already been requested before, and therefore, in one example, the resulting fragment retrieval process includes the following two sub-processes: a first sub-process of requesting fragments associated with new segments to be reconstructed, and a second sub-process of requesting additional fragments needed to complement already requested fragments, in order to reconstruct the segments. The first and second sub-processes work together, such that the second sub-process may complement fragments associated with a first segment, while the first sub-process runs ahead in an attempt to obtain fragments needed to reconstruct a second segment; wherein the second segment is located ahead of the first segment. The first and the second sub-processes can also be described as two different quantities of fragments being requested: a first quantity associated with the first sub-process requests, and a second quantity associated with the second sub-process requests.

[0176] FIG. **27** illustrates one example of retrieving fragments and compensating the failures. Content **100** is segmented into segments **102***a*, **102***b*, and **102***c*, and each segment is erasure-coded into four fragments, as illustrated for segment **102***a*, which is coded into fragments **391***a* to **391***d*. This example assumes that each segment can be reconstructed by obtaining any three fragments associated with it. Prior to time T**1**, the assembling device requests fragments **391***a*, **391***b*, and **391***c* in order to reconstruct segment **102***a*. At time T**1**, only two of the requested fragments **391***a* and **391***c* have resulted in fragment reception, and were placed **394***a*, **394***c* in the buffer **398**. Fragment **391***b* has not yet been received at time T**1**, but can still be received later, and therefore at time T**1** the assembling device does not yet try to complete the missing fragment with an additional fragment request. Instead, it proceeds and requests fragments associated with segment **102***b*. At time T**2**, all of the fragments requested for segment **102***b* have arrived, and have been placed **395***a*, **395***b*, **395***d* in the buffer **398**. Prior to time T**2**, the assembling device transmits additional requests for fragments associated with segment **102***c*, and at time T**3** two out of the requested fragments have arrived, and have been placed **396***b*, **396***c* in the buffer **398**. At time T**3**, the assembling device realizes that the chances on receiving the previously requested fragment **391***a* (associated with segment **102***a*) are too small. This may be concluded, for example, as a long time having elapsed since the request, or by receiving a message from a fractional-storage server saying it is too loaded to respond with a fragment. Either way, the assembling device chooses to request an additional fragment **391***d*, instead of the previously requested **391***b*. At time T**4**, the additional request is met with the reception of fragment **391***d*, and with its placement **394***d* in the buffer **398**. At time T**5**, the third fragment previously requested for segment **102***c* has finally arrived and has been placed **396***a* in the buffer **398**, so there is no need to complement with an additional fragment request. At time T**5** all fragments needed to reconstruct segments **102***a* to **102***c* are stored in the buffer **398**. It is noted that only one additional fragment request was needed in order to account for the lack of reception of fragment **391***b*, and that this additional fragment request was issued after consequent fragments had already been requested for consequent segments.

[0177] In one embodiment, significant communication latency and/or other latencies between requesting and receiving a fragment exists. A significant latency may result in a case where the average latency in responding to fragment requests is in the order of magnitude of the total transmission time of all fragments needed to reconstruct a segment. As an example, if a segment needs 64 fragments of 1500 Bytes each to be reconstructed, and the assembling device has a 1.5 Mpbs incoming connection, then it takes about (64[fragments]×1500[bytes per fragment]×8[bits per byte])/1.5 Mbps=0.512 seconds to transmit the fragment via the incoming connection. If the average latency is 0.2 seconds (which is within the order of magnitude of 0.512 seconds), then from the time of requesting the first fragment to the time all fragments have arrived, a period of no less than 0.512+0.2=0.712 seconds may elapse. If the process takes 0.712 seconds, the resulting effective incoming throughput will be only (64[fragments]×1500[bytes per fragment]×8[bits per byte])/0.712[seconds]=1.07 Mbps, which is significantly less than the potentially 1.5 Mbps. In a case where some fragments are lost, and need to be requested again, the total time for segment retrieval may reach as high as 0.512+0.2+0.2=0.912, and the effective incoming throughput down to only 842 Kbps. The significant latency therefore adversely affects the effective incoming throughput. The effective throughput can be made to approach the incoming bandwidth available to the assembling device by utilizing the above-described fragment retrieving process comprising the two sub-processes of requesting fragments and complementing the failures. In this case, the first sub-process can be made to result in an average targeted fragment reception throughput, and span multiple segments, without handling the lost fragments. The second sub-process can then complement with additional needed requests, approximately per each fragment request that seems not to result in an actual fragment reception. According to another view, the first sub-process is an open loop retrieval process, in which the assembling device does not wait to check whether enough fragments have arrived per segment. And the second sub-process is the process, which closes the loop on fragments arrival, in order to make sure that every segment has enough fragments to enable reconstruction.

[0178] In one embodiment, the assembling device may control the erasure-coded fragment reception throughput by controlling the rate of fragment request. For example, each of n fragments has a known size S**1** to Sn. Therefore, issuing n requests over a period of T will result in an average fragment reception throughput of (S**1**+S**2** . . . +Sn)/T. In one example, if each fragment is 1500 Bytes, and 64 fragment requests are issued over a period of 0.5 seconds, then the average expected fragment arrival throughput is (64×1500×8)/0.5=1.53 Mbps. The fragment requests do not need to be uniformly spread over the period of 0.5 seconds, although such a spread may result in a more stable throughput, which means that less communication buffering will be needed. Using the above-described rate-control technique may result in one or more of the following: retrieving the content at a target fragment reception throughput; preventing communication buffer spill at the last mile network resulting from uncontrolled fragment requests; and/or reducing fragment loss due to averaging the fragment traffic.

[0179] In one embodiment, the controlled rate is set to an initial value estimated to support a streaming operation, or alternatively to a predefined initial value. The controlled rate is then gradually increasing until the fragment loss resulting

from requesting fragments at the controlled rate reaches a predefined threshold. At this point, the controlled rate remains at approximately a steady state, or alternatively is slightly reduced in order to create a margin.

[0180] In one embodiment, the fragment loss resulting from requesting fragments at a controlled rate is monitored. The controlled rate is then gradually decreased in order to improve the fragment loss. The controlled rate is decreased approximately up to the point that the fragment loss drops below a predefined threshold.

[0181] In one embodiment, the fragment loss resulting from requesting fragments at a controlled rate is monitored. The controlled rate is then gradually increased in order to improve the fragment reception rate. The controlled rate is increased approximately up to the point that the fragment loss rises above a predefined threshold.

[0182] Using the fragment pull protocol may result in a significant amount of requests, which may consume a significant percent of the outgoing bandwidth of the assembling device. In one embodiment, the assembling device aggregates two or more fragment pull protocol requests for erasure-coded fragments in one aggregated request. The aggregated request is transmitted to a relay server, which, in turn, distributes the aggregated requests between two or more fractional-storage servers. Aggregating the requests may save bandwidth and significantly reduce the number of packets needed to convey the requests. For example, a request for a single fragment, transported over UDP/IP or similar protocols, requires about 60 bytes of overhead. Meaning that even if one request, containing only several bytes of request-related payload, is needed, the resulting message will probably exceed 80 bytes. Therefore, transmitting, as an example, 64 fragment requests per single segment will result in a total uplink bandwidth requests of about 80×64=5 Kbyte. Using the relay approach can result in a much more efficient communication. In a case where 64 fragment requests are aggregated into one message to a relay server, and assuming that each fragment request has a payload of about 10 bytes (the payload may comprise segment and content information, as an example), then one aggregated message may contain about 60+64×10=700 bytes, instead of the 5 Kbytes. In addition, one packet containing the aggregated requests may be used instead of 64 separate packets needed in the non-aggregated case. This may be significant when, as an example, the assembling device is connected to the Internet via a wireless connection, such as a WiFi connection. In this case, instead of having to gain access to the air interface 64 times per segment, the assembling device accesses the air interface only once per segment, under the assumption that 64 fragment requests are loaded into one aggregated request.

[0183] In one embodiment, the assembling device transmits an aggregated message containing multiple fragment requests to a relay server. The relay server, in turns, creates multiple fragment requests, and transmits the requests to multiple fractional-storage servers on behalf of the assembling device.

[0184] In one embodiment, an aggregated fragment request message sent to a relay server identifies the destined storage server(s). In one example, the destined storage servers are identified by their IP addresses. The relay server, in turn, uses the identification information in the message to create multiple requests for fragments on behalf of the assembling device. In this embodiment, the assembling device determines the destined servers (via the aggregated relayed message), and the relay server creates the multiple fragment requests according to the assembling device's instructions.

[0185] Referring back to FIG. **26**, in one embodiment, a single request payload **504** is used to relay multiple fragment requests to multiple storage servers via a relay server. The payload **504** comprises the identified content **801** and segment **802**, the number of requested fragments **803**, and a list of servers **804**, which contain relevant fragments. The relay server, in turns, relays multiple fragment requests to the servers identified in the list **804**, according to some embodiments.

[0186] In one embodiment, an assembling device transmits aggregated messages to a relay server, including the number of fragments needed per certain segment, but without identifying the storage servers from which fragments are to be requested. The relay server selects the appropriate storage servers to which the fragment requests are to be transmitted, and transmits discrete or aggregated fragment requests, corresponding to the number of fragments requested by the assembling device, to the selected storage servers. The storage servers receive the fragment requests from the relay server, and transmit the requested fragment to the assembling device.

[0187] The relay server may select the storage servers according to one or more criteria, as long as the selected storage servers store relevant fragments. Optionally, the relay server forwards the address of the assembling device to the selected storage servers, and/or adds the address of the assembling device to the fragment requests transmitted to the selected servers, in order to enable the storage servers to transmit the fragment response to the assembling device. Referring back to FIG. **26**, in one example, the assembling device transmits a single request payload **502** to the relay, which identifies the content **801**, the segment **802**, and the number of requested fragments **803**. The relay server, in turns, selects the relevant storage servers, generates the fragment request messages, and transmits the messages to the selected storage servers on behalf of the assembling device.

[0188] Shifting the process of selecting the storage servers from the assembling device to the relay server enables the design of a relatively thin and simple assembling device, having a relatively simple software, since all the assembling device has to decide in order to issue an aggregated fragment request to the relay server is how many fragments it needs per segment and, optionally, when it needs them.

[0189] In one embodiment, an assembling device transmits aggregated messages to a relay server, comprising general information regarding a portion of streaming content for which fragments are needed. Optionally, the portion of the streaming content comprises several consecutive segments. In one embodiment, the portion is defined by a starting point and an ending point within the streaming content, and the relay server uses these points to determine the actual segments comprising the portion. Then the relay generates and transmits the corresponding fragment requests to the relevant storage servers.

[0190] In one embodiment, the starting point and ending point are time stamps within the streaming content. In one embodiment, the portion is indicated by including a starting point and duration of the portion.

[0191] In one embodiment, an assembling device transmits an aggregated message to a relay server, comprising fragment pull protocol requests that identify the destined storage servers. The relay server analyzes the fragment pull protocol requests and may change one or more of the destined storage

servers based on the network related information it holds, such as network congestion, server load, and/or cost.

[0192] FIG. **28** illustrates one embodiment, wherein segment **101***a* of content **100** is encoded into erasure-coded fragments **390***a* to **390**(M), such that any sufficient subset of the fragments can be used to reconstruct segment **101***a*. Fragments **390***a* to **390**(N) are stored in fractional-storage servers **399***a* to **399**(N) respectively, and fragments **390**(N+1) to **390**(M) are stored in streaming server **399**S. In one example, fragments **390**(N+1) to **390**(M) form a group of fragments which are sufficient to reconstruct segment **101***a*. Subsequent segments **101***b* to **101***j* of content **100** may be similarly encoded into additional fragments stored on the servers (not illustrated). Assembling device **309** uses two different protocols approximately simultaneously to retrieve fragments for segment reconstruction: (i) a push protocol, and (ii) a fragment pull protocol. The push protocol **301**S is used to deliver fragments **390**(N+1) to **390**(M) to assembling device **309**. The push protocol may be RTP based or TCP-connection based, or any other type of transmission that does not require assembling device **309** to explicitly ask for each of fragments **390**(N+1) to **390**(M). In one example, fragments **390**(N+1) to **390**(M) are delivered to the assembling device using a single RTP stream **301**S, such that upon reception of the fragments from the stream, the assembling device can immediately reconstruct segment **101***a*. The fragment pull protocol is used by the assembling device to retrieve additional fragments that may be needed to reconstruct segment **101***a* if one or more fragments out of fragments **390**(N+1) to **390**(M) fail to reach the assembling device. In one example, fragment **390**(N+2) fails to reach the assembling device due to Internet packet loss conditions (referred to as fragment loss). The assembling device, after concluding that fragment **390**(N+2) is missing, uses a fragment pull protocol to retrieve a substitute fragment out of one of the fractional-storage servers **390***a* to **390**(N), and uses this fragment to complete the reconstruction of the segment **101***a* (any one of fragments **390***a* to **390**(N) will do). For example, the assembling device chooses fragment **390***a* as the one additional fragment, by requesting and receiving it **303***a* from server **399***a*, using a fragment pull protocol. If more fragments out of fragments **390**(N+1) to **390**(M) fail to reach the assembling device **309**, it may compensate by pulling substitute fragments from some or all of servers **399***a* to **399**(N), illustrated as fragment pull protocol requests and responses **303***a* to **303**(N)).

[0193] In one embodiment, the fragment pull protocol requests for additional needed fragments are not made to fractional-storage servers **399***a* to **399**(N), but are rather made to server **399**S. In this case, the assembling device asks server **399**S to retransmit the fragment which has failed to arrive. In this embodiment, only fragments that fail to reach the assembling device via the push transmission **301**S cause an added communication overhead in the form of explicit fragment pull protocol requests, such that if no fragments are actually lost over transmission **301**S, there is no need for fragment pull requests **303***a* to **303**(N).

[0194] In some embodiments, the push protocol is implemented using one or more sub-transmissions. Optionally, a push protocol transmission is implemented using multiple sub-transmissions, each transporting a fraction of the fragments transmitted by the push protocol transmission. A sub-transmission may be transported using an IP stream such as RTP, an HTTPS session, or any other form of transporting a sequence of fragments between a source server and a destination assembling device.

[0195] In one embodiment, the assembling device starts retrieving fragments using only fragment pull protocol processes, and then, when concluding that a specific server is responsive enough, instructs it to start sending a push-transmission for the remaining segments. In this case, the assembling device may start with pure pull-protocol based fragment retrieval, and gradually switch to push-protocol transmissions, up to the point that approximately all fragments are delivered using push-transmissions, and using the pull requests only as a means to overcome failure of obtaining specific fragments by the assembling device. In one embodiment, the fragment pull protocol and the push protocol are used interchangeably to obtain enough fragments to reconstruct segments. In one example, the assembling device may start to obtain fragments using a push protocol and then switch to a fragment pull protocol. In one example, the assembling device may use both fragment pull protocol and push protocol to obtain fragments at the same time, wherein the assembling device may change the ratio Fpull/Fpush on-the-fly to any value between zero and infinity, where Fpull denotes the number of fragments associated with a certain segment that are obtained using a fragment pull protocol, and Fpush denotes the number of fragments associated with the certain segment that are obtained using a push protocol.

[0196] In the claims, sentences such as "wherein the assembling device is configured to use a fragment pull protocol to obtain the fragments" and "wherein the assembling device is configured to use sub-transmissions to obtain the fragments" are to be interpreted as open claim language. Therefore, an assembling device configured to use a fragment pull protocol to obtain fragments may also obtain fragments using sub-transmissions, and vice-versa.

[0197] FIG. **21** illustrates one embodiment of real time streaming content retrieval from fractional-storage servers. An assembling device begins a process of obtaining streaming content **700** for presentation. Starting at T**1**, the assembling device requests erasure-coded fragments **720***a* to **720**(K). By T**2**, all K erasure-coded fragments are obtained, and at time T**2***b* until T**4**, erasure-coded fragments **720***a* to **720**(K) are decoded into segment **710***a*. The retrieval time of the erasure-coded fragments and the segment decoding time should be equal to or faster than the corresponding presentation time, in order to enable a continuous presentation, once presentation begins at T**5**. T**2***b* minus T**2** is a short delay, and can be fractions of a second. Subsequent erasure-coded fragments **730***a* to **730**(K) are retrieved between T**2** and T**3**, and are decoded into subsequent segment **710***b* between T**4** and T**6**.

[0198] In one example, the streaming content **700** is encoded at 1 Mbps, and the segment size is 96 Kbytes. The presentation of each segment takes about 0.77 seconds. Retrieving fragments **720***a* to **720**(K) takes no more than 0.77 seconds, meaning that the assembling device's connection bandwidth must be 1 Mbps or higher. Decoding segment **710***a* takes no more than 0.77 seconds. If a small delay of 0.2 seconds is assumed for both T**2***b* minus T**2** and T**5** minus T**4**, then T**5** can start at 0.77+0.2+0.77+0.2=1.94 seconds after T**1**, meaning that presentation can begin about 2 seconds following request of the first erasure-coded fragment. In another example, the retrieval process and the decoding process are performed faster than the real time presentation

bounds, therefore enabling a shorter time to play and a download rate that exceeds the presentation rate.

[0199] FIG. **29** illustrates one embodiment where the erasure-coded fragments **720***a* to **720**(K) are retrieved in approximately random order **720**(K−1), **720***a*, **720**(K), **720***b*, or any other order, as long as at least the K erasure-coded fragments needed for decoding the segment **710***a* are available until time T**2**. Similar retrieval in random order is applied to erasure-coded fragments **730***a* to **730**(K) and all other subsequent fragments.

[0200] In one embodiment, the fragments associated with sequential segments of streaming content are delivered to an assembling device as a plurality of sub-transmissions. In this case, each fractional-storage server participating in the delivery of the fragments to the assembling device sends a transmission to the assembling device comprising a sequence of erasure-coded fragments. This transmission is referred to as a sub-transmission. In one example, each sub-transmission contains at least one fragment per each sequential segment of the streaming content. In one example, the sub-transmission starts at a segment indicated by the assembling device, and continues from that point onwards, approximately according to the sequential order of segments, until the assembling device instructs the server to stop, or until reaching the last segment of the content. Each sub-transmission carries only a fraction of the fragments (per segment) needed to reconstruct the segments of the streaming content, such that the combination of at least two sub-transmissions received by the assembling device from the servers allows the assembling device to obtain enough fragments needed to reconstruct each segment.

[0201] In one embodiment, each sub-transmission is delivered to the assembling device via a streaming session, such as an RTP session, wherein the RTP packets transport the fragment sequence approximately according to the order of the sequential segments. In one embodiment, each sub-transmission is delivered to the assembling device via an HTTP connection, or other closed-loop data transfer mechanisms over TCP/IP. In one embodiment, the assembling device may change one or more transmitting servers on the fly, by instructing the server(s) to stop sending an already active sub-transmission—as may be needed in a case of an RTP session, and initiating new sub-transmissions from other servers instead. Replacement of transmitting servers on the fly may be needed in a case of a server failure, network failure, or high load or latency conditions.

[0202] In some embodiments, a broadcast-like effect is achieved by distributing to and retrieving from fractional-storage servers a broadcast channel/live content in real time, using a combination of real time distribution and real time retrieval techniques. In a broadcast-like effect, a given channel or content for broadcasting is distributed to at least one assembling device, optionally by means of pushing relevant fragments to the assembling device, or by pulling the relevant fragments by the assembling device, and potentially to many assembling devices at approximately the same time, which creates a similar effect to traditional broadcasting.

[0203] In one embodiment, once starting to retrieve a broadcast-like stream, the assembling device may use one of the following methods to synchronize the retrieval of the stream's segments with the ongoing availability of new segments of the stream: (i) The assembling device retrieves additional segments such that the average rate of obtaining new frames approximately equals the average rate of presenting frames. (ii) The assembling device retrieves additional segments such that it does not try to retrieve segments that are not yet indicated as being available. And (iii) The assembling device retrieves additional segments so as to approximately maintain a constant distance (in segments) between the most currently available segment and the segment currently being retrieved.

[0204] In one embodiment, the assembling device presents the broadcast-like stream at approximately the same frame rate as the rate of producing new frames for the broadcast-like stream. In one example, the frame rate is constant throughout the stream, such as the case of fixed 24, 25, 50, or 60 frames per second.

[0205] In one embodiment, the assembling device obtains an indication regarding the most newly available segment (per specific broadcast-like stream) for retrieval. The assembling device then starts to retrieve from the most newly available segment. In one example, the most newly available segment is the last segment that was distributed to the fractional-storage servers. In another example, the most newly available segment is a segment that was recently distributed to the fractional-storage servers, but wherein there are newer distributed segments, which are not yet indicated as being available.

[0206] In one embodiment, the broadcast-like stream is of a pre-recorded content, such that it is possible to distribute the entire content to the fractional-storage servers, and after any period of time allow the real time consumption of the content by any number of assembling devices. In such a case, an indication is made to the assembling devices regarding the real time allowance to retrieve the related segments. The allowance can start at a certain point in time (which corresponds to the beginning of the broadcast-like "transmission") for the first segment, and then the allowance may continue for subsequent segments, at a rate that approximately corresponds to sustaining the frame rate of the broadcast-like stream.

[0207] By using a pull protocol or a push protocol with multiple sub-transmissions, the assembling device can obtain erasure-coded fragments from one, two or more different arrays of CDN servers and/or bandwidth amplification devices seamlessly.

[0208] In one embodiment, when a CDN server receives a request for an erasure-coded fragment, it may supply the erasure-coded fragment or supply an address of a bandwidth amplification device having an image of the requested erasure-coded fragment. Optionally, a bandwidth amplification device storing one erasure-coded fragment of a specific content also stores an image of some or all other erasure-coded fragments associated with the specific content (which are stored on the specific CDN server). Alternatively, the bandwidth amplification device stores unique erasure-coded fragments generated from the same segments used for generating the erasure-coded fragments stored on the specific CDN server. In these cases, the assembling device may approach the bandwidth amplification devices instead of the CDN server for the relevant erasure-coded fragments of the specific content until (i) the end of the content; (ii) a predefined time period elapses; (iii) receiving an appropriate message; or (iv) a combination of the aforementioned.

[0209] In one embodiment, an assembling device tries to obtain an erasure-coded fragment or sub-transmission from the relevant server, and if the server does not have the necessary bandwidth to respond with fragment/s, the server relays

the fragment request/s to relevant bandwidth amplification devices. The relevant bandwidth amplification devices can then send the fragment/s directly to the assembling device.

[0210] In one embodiment, unique erasure-coded fragments can be distributed between two types of devices: (i) high bandwidth fractional-storage servers, such as CDN servers, and (ii) relatively low bandwidth and storage devices acting as bandwidth amplification devices, such as peer-to-peer (P2P) devices. Since the fragments distributed between the two types of devices are unique, any combination of devices, from both types, can be used to obtain a decodable set of fragments, if the combination of devices stores a decodable set of fragments. In one embodiment, there are at least ten times more bandwidth amplification devices than high bandwidth servers, and the redundancy factor used in decoding the fragments is greater than 10. In this case, the servers can be used all or most of the time, and the bandwidth amplification devices can be used from time to time, according to bandwidth requirements, and according to the availability of the bandwidth amplification devices. In one embodiment, the processes of obtaining a fragment from a server and from a bandwidth amplification device are essentially the same, and the fragments are essentially identical in construction and format. In one embodiment, the high redundancy factor needed to support a large hybrid array of servers and bandwidth amplification devices is achieved using rateless coding techniques.

[0211] In one embodiment, the fractional-storage system is approximately insensitive to the mixture of the consumed contents as long as the aggregated throughput is below the total throughput of the fractional-storage servers.

[0212] FIG. **30** illustrates one example of a fractional-storage server array, including N servers (**399***a* to **399**(N)), and storing content A, which includes erasure-coded fragments **310***a* to **310**(N), and content B, which includes erasure-coded fragments **320***a* to **320**(N). Each server is connected to the network **300** with a fragment delivery bandwidth capability B **339**. Therefore, the N servers have an aggregated bandwidth of B×N. A first group of assembling devices **329***a* consumes content A at an average bandwidth Ba **349***a*. A second group of assembling devices **329***b* consumes content B at an average bandwidth Bb **349***b*. Since all of the servers participate in the transmission of the two contents, the first and second groups can potentially consume all server bandwidth, up to the limit where Ba+Bb=N×B, with any ratio of demand between the first and second contents, and with no special provisions to be made when storing the erasure-coded fragments related to the two contents in the fractional-storage server array.

[0213] FIG. **31** illustrates the case where the first group **328***a*, which consumes content A, becomes larger than **329***a*, with a larger bandwidth Ba **348***a*. The second group **328***b*, which consumes content B, becomes smaller than **329***b*, with a smaller bandwidth Bb **348***b*, such that Ba is about the same as Bb. In this case, the array can still be exploited up to the aggregated bandwidth, since, as before, Ba+Bb can still be almost as high as N×B. FIG. **32** illustrates the case where the first group has disappeared, allowing the second group **327***b*, which consumes content B, to extract an aggregated bandwidth of Bb **347***b* that can potentially reach the limits of the server array, such that Bb=N×B. Again, this is achieved without updating the erasure-coded fragments associated with content A and content B, and without using inter-server interaction.

[0214] In some embodiments, the ability to utilize the aggregated bandwidth of approximately all of the participating servers, for the delivery of about any mixture of contents with about any mixture of content bandwidth demand, is made possible by one or more of the following: (i) each assembling device selecting a subgroup of the least loaded fractional-storage servers from which to retrieve the necessary number of erasure-coded fragments to reconstruct a segment or several segments (least-loaded server selection criterion); or (ii) each assembling device approximately randomly selecting a subgroup from which to reconstruct a segment or several segments, such that when many assembling devices select at random, the various fractional-storage servers are selected approximately the same number of times (or in proportion to their available resources, such as unutilized bandwidth), which in turn balances the load between the participating servers (random server selection criterion). It is noted that (i) the selections may be made by either the assembling devices themselves, or may be made for the assembling devices by a control server, which then communicates the selections to each of the assembling devices; (ii) the selections may be made approximately for each segment, or for a group of segments, or only once per content at the beginning of the content; (iii) some assembling devices may use an approximately random server selection criterion, while other assembling devices may use least-loaded server selection criterion; (iv) the least-loaded selected servers may be selected out of a portion of all available fractional-storage servers. For example, the least-loaded servers may be selected from fractional-storage servers with low latency response or with low hop count to the assembling device; (v) the least-loaded servers may include servers having the most unutilized bandwidth. Additionally or alternatively, it may include servers having any unutilized bandwidth left to serve additional assembling devices; (vi) an approximately random or least-loaded selection of servers may be made such that all servers are selected to determine a subgroup, or it can be made such that every time selections are made, only some servers are selected, while the others remain as before. In these cases, the assembling device runs a process in which only a small portion of the servers currently in the serving subgroup are reselected. In the case of approximately random selection, the assembling device may randomly select the number of servers in the serving subgroup for random selection (reselection in this case, since they are replacing other servers already in the serving subgroup of the specific assembling device), such that eventually, over time, all servers within the serving subgroup have the chance to be randomly reselected. In the case of least-loaded server selection, only the most loaded servers within the serving subgroup may be selected and replaced by less-loaded servers.

[0215] FIG. **33** illustrates one embodiment of using the entire aggregated bandwidth of the fractional-storage servers for delivering multiple contents. Approximately any number of contents having any mixture of bandwidth demand per content may be delivered, as long as the aggregated bandwidth demand does not exceed the aggregated bandwidth of the fractional-storage servers. In one example, broadcast-like streams **3101**, **3102**, and **3103** are delivered to multiple assembling devices via multiple fractional-storage servers. Each stream is a live TV channel carrying multiple TV programs. For example, stream **3101** comprises TV programs **3110** to **3112**, each spanning a specific time interval. The other streams comprise of multiple TV programs as well.

Before time T1, stream **3130** has a bandwidth demand of **3130'** (meaning that all assembling devices that are currently retrieving stream **3130'** use a total bandwidth of **3130'** out of the fractional-storage servers). The other streams **3120** and **3110** have bandwidth demands of **3120'** and **3110'** respectively. The total bandwidth demand of the three streams **3130'+3120'+3110'** does not exceed the aggregated bandwidth of the fractional-storage servers **3150**, and therefore all streams are fully delivered to the assembling devices. The load of the three streams is spread approximately equally among the participating fractional-storage servers, optionally because of a mechanism that selects the least-loaded servers to serve each assembling device, and/or a mechanism that approximately randomly selects servers to serve each assembling device. At time T1, TV program **3120** ends, and TV program **3121** starts. Program **3121**'*s* demand **3121'** is higher than the previous demand **3120'**, and therefore a higher aggregated bandwidth is drawn from the fractional-storage servers. Still, the aggregated bandwidth demand of all three streams (**3130'+3121'+3110'**) is lower than the maximum possible **3150**, and therefore the newly added bandwidth demand is fully supported by the servers. Optionally, the additional demand created by TV program **3121** (**3121'** minus **3120'**) is caused by the addition of new assembling devices that join stream **3102** and retrieving additional erasure-coded fragments. Additionally or alternatively, the additional demand created by TV program **3121** is caused by a higher bandwidth demand of TV program **3121**, such as 3D data or higher resolution. Newly added assembling devices may choose fractional-storage servers from which to retrieve, according to a least-loaded server selection criterion and/or an approximately random server selection criterion, and therefore the total load is still spread approximately equally among the participating servers. At time T2, TV program **3110** ends, and a new program **3111** begins, which is less popular, and therefore creates a lower bandwidth demand **3111'**. The result is a decrease in the total delivered bandwidth. At time T3 TV program **3130** ends, and TV program **3131** starts with a higher bandwidth demand of **3131'**. At time T4 both TV programs **3111** and **3121** end, and two new programs **3112** and **3122** start. TV program **3112** is highly popular and therefore generates a large bandwidth demand **3112'**. Program **3122** is not popular, and therefore generates a limited bandwidth demand **3122'**. Some of the additional bandwidth needed by program **3112** is taken from servers that stop serving assembling devices previously retrieving program **3121**, such that the aggregated bandwidth of all three streams (**3131'+3122'+3112'**) is still below the maximum possible bandwidth **3150**, despite the fact that program **3112** is generating a large bandwidth demand. This example illustrates how the fractional-storage servers support almost any demand mixture, as long as the aggregated demand of all streams is kept below the aggregated maximum capacity of the servers **3150**. Consequently, the distribution of all of the streams to the fractional-storage servers is approximately unrelated to the changes in bandwidth demand for programs carried by each stream; each stream can be regarded as a sequence that is segmented, erasure-encoded, and distributed to the participating servers. There is no need to account for demand variations during the distribution of each stream, nor is there a need to know in advance the bandwidth demand for each stream or for each program within each stream. It is noted that the demand variations are illustrated as instant variations, but may also be gradual and may occur during a program and not necessarily when one program ends and the other begins.

[0216] Many of the disclosed embodiments using fragment pull protocol may use fragment pull protocol for high latency for retrieving the fragments.

[0217] In the claims, a sentence such as "the erasure-coded fragments support source-selection diversity" is to be interpreted as fragments encoded using any kind of erasure-code that can produce N unique fragments, from which C combinations of decodable sets of fragments can be selected, wherein C is much greater than N. Standard parity checks, standard checksums, and standard cyclic redundancy checks (CRC) are examples of codes that do not support source-selection diversity.

[0218] In this description, numerous specific details are set forth. However, the embodiments of the invention may be practiced without some of these specific details. In other instances, well-known hardware, software, materials, structures and techniques have not been shown in detail in order not to obscure the understanding of this description. In this description, references to "one embodiment" mean that the feature being referred to may be included in at least one embodiment of the invention. Moreover, separate references to "one embodiment" or "some embodiments" in this description do not necessarily refer to the same embodiment. Illustrated embodiments are not mutually exclusive, unless so stated and except as will be readily apparent to those of ordinary skill in the art. Thus, the invention may include any variety of combinations and/or integrations of the features of the embodiments described herein.

[0219] Although some embodiments may depict serial operations, the embodiments may perform certain operations in parallel and/or in different orders from those depicted. Moreover, the use of repeated reference numerals and/or letters in the text and/or drawings is for the purpose of simplicity and clarity and does not in itself dictate a relationship between the various embodiments and/or configurations discussed. The embodiments are not limited in their applications to the details of the order or sequence of steps of operation of methods, or to details of implementation of devices, set in the description, drawings, or examples. Moreover, individual blocks illustrated in the figures may be functional in nature and do not necessarily correspond to discrete hardware elements. While the methods disclosed herein have been described and shown with reference to particular steps performed in a particular order, it is understood that these steps may be combined, sub-divided, or reordered to form an equivalent method without departing from the teachings of the embodiments. Accordingly, unless specifically indicated herein, the order and grouping of the steps is not a limitation of the embodiments. Furthermore, methods and mechanisms of the embodiments will sometimes be described in singular form for clarity. However, some embodiments may include multiple iterations of a method or multiple instantiations of a mechanism unless noted otherwise. For example, when a controller or an interface are disclosed in an embodiment, the scope of the embodiment is intended to also cover the use of multiple controllers or interfaces.

[0220] Certain features of the embodiments, which may have been, for clarity, described in the context of separate embodiments, may also be provided in various combinations in a single embodiment. Conversely, various features of the embodiments, which may have been, for brevity, described in

the context of a single embodiment, may also be provided separately or in any suitable sub-combination.

[0221] Embodiments described in conjunction with specific examples are presented by way of example, and not limitation. Moreover, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the embodiments. Accordingly, it is intended to embrace all such alternatives, modifications and variations that fall within the spirit and scope of the appended claims and their equivalents.

What is claimed is:

1) An apparatus comprising: an assembling device configured to obtain erasure-coded fragments from at least one CDN server located close to or on the Internet backbone; upon a fragment loss, the assembling device is further configured to use a fragment pull protocol to retrieve a substitute erasure-coded fragment from a nearby fractional-storage CDN server having low latency in responding to the assembling device.

2) The apparatus of claim **1**, wherein the CDN servers located close to or on the Internet backbone are fractional-storage CDN servers.

3) The apparatus of claim **1**, wherein the assembling device is located at the user premises.

4) The apparatus of claim **1**, wherein the assembling device is configured to use a fragment pull protocol to obtain the fragments from the at least one CDN server located close to or on the Internet backbone.

5) The apparatus of claim **4**, wherein the erasure-coding is rateless-coding potentially resulting in fragments having a limitless redundancy factor.

6) The apparatus of claim **1**, wherein the fragments are generated from segments of streaming content, and the assembling device is further configured to obtain and retrieve the fragments approximately according to the segment presentation order.

7) The apparatus of claim **6**, wherein the nearby fractional-storage CDN server is located at the edge of the Internet.

8) The apparatus of claim **6**, wherein the assembling device is configured to obtain the erasure-coded fragments from the at least one CDN server using a fragment pull protocol for high latency.

9) The apparatus of claim **6**, wherein the assembling device is configured to obtain at least 5 times more erasure-coded fragments from the at least one CDN server located close to or on the Internet backbone than from the nearby fractional-storage CDN server; and the assembling device is configured to obtain the fragments from the at least one CDN server located close to or on the Internet backbone via a push protocol.

10) The apparatus of claim **1**, wherein the erasure-coded fragments are generated from segments of streaming content; and wherein for lost fragments associated with segments designed for trick play, the assembling device pulls the substitute erasure-coded fragment from the nearby fractional-storage CDN server; and for lost fragments associated with segments not designed for trick play, the assembling device pulls the substitute erasure-coded fragment from the CDN server located close to or on the Internet backbone.

11) An apparatus comprising: an assembling device configured to receive erasure-coded fragments from at least one CDN server located close to or on the Internet backbone using a push protocol; upon a fragment loss, the assembling device is configured to pull a substitute erasure-coded fragment from a nearby fractional-storage CDN server having low latency.

12) The apparatus of claim **11**, wherein the nearby fractional-storage CDN server is located at the edge of the Internet, the CDN servers located close to or on the Internet backbone are fractional-storage CDN servers, and the quantity of the pushed fragments is at least 5 times larger than the quantity of the pulled fragments.

13) A streaming media delivery network, comprising: at least one CDN server, located close to or on the Internet backbone, configured to provide to an assembling device a first set of erasure-coded fragments of streaming media; and a plurality of fractional-storage CDN servers, located at the edges of the Internet, configured to provide to the assembling device a second set of erasure-coded fragments of the streaming media; wherein the first set is at least 5 times larger than the second set, and the first and second sets comprise enough fragments to enable the assembling device to start playing the streaming media within a short period of time following a request.

14) The streaming media delivery network of claim **13**, wherein the second set of fragments is provided in response to fragment pull protocol requests issued by the assembling device.

15) The streaming media delivery network of claim **14**, wherein the erasure-coding is rateless-coding potentially resulting in fragments having a limitless redundancy factor, and the assembling device is located at the user premises.

16) The streaming media delivery network of claim **14**, wherein the assembling device is located at the user premises, and the at least one CDN server located close to or on the Internet backbone are fractional-storage CDN servers.

17) The streaming media delivery network of claim **14**, wherein the short period of time following a request is shorter than 15 seconds for a high-definition full-length movie.

18) The streaming media delivery network of claim **14**, wherein the assembling device is configured to pull the second set of fragments from one or more of the nearby fractional-storage CDN servers upon a fragment loss.

19) The streaming media delivery network of claim **18**, wherein the fragment loss comprises not receiving an expected fragment within a predefined period of time, and the erasure-coded fragments support source-selection diversity.

20) The streaming media delivery network of claim **14**, wherein the erasure-coded fragments are generated from segments of the streaming media, some of the segments are designed for trick play, and the majority of the fragments provided by the edge servers are associated with the segments designed for trick play.

* * * * *