Towards Discovering Remote Code Execution Vulnerabilities in Apple FaceTime

Tao Huang and Tielei Wang

About us

- Tao Huang
  - Senior researcher at Pangu Lab
  - Focusing on iOS/macOS vulnerability discovery
- Tielei Wang
  - PhD, co-founder of Team Pangu, organizer of MOSEC
  - Leading iOS/macOS security research at Pangu Lab
  - Regularly present research at BlackHat, POC, etc

Motivation

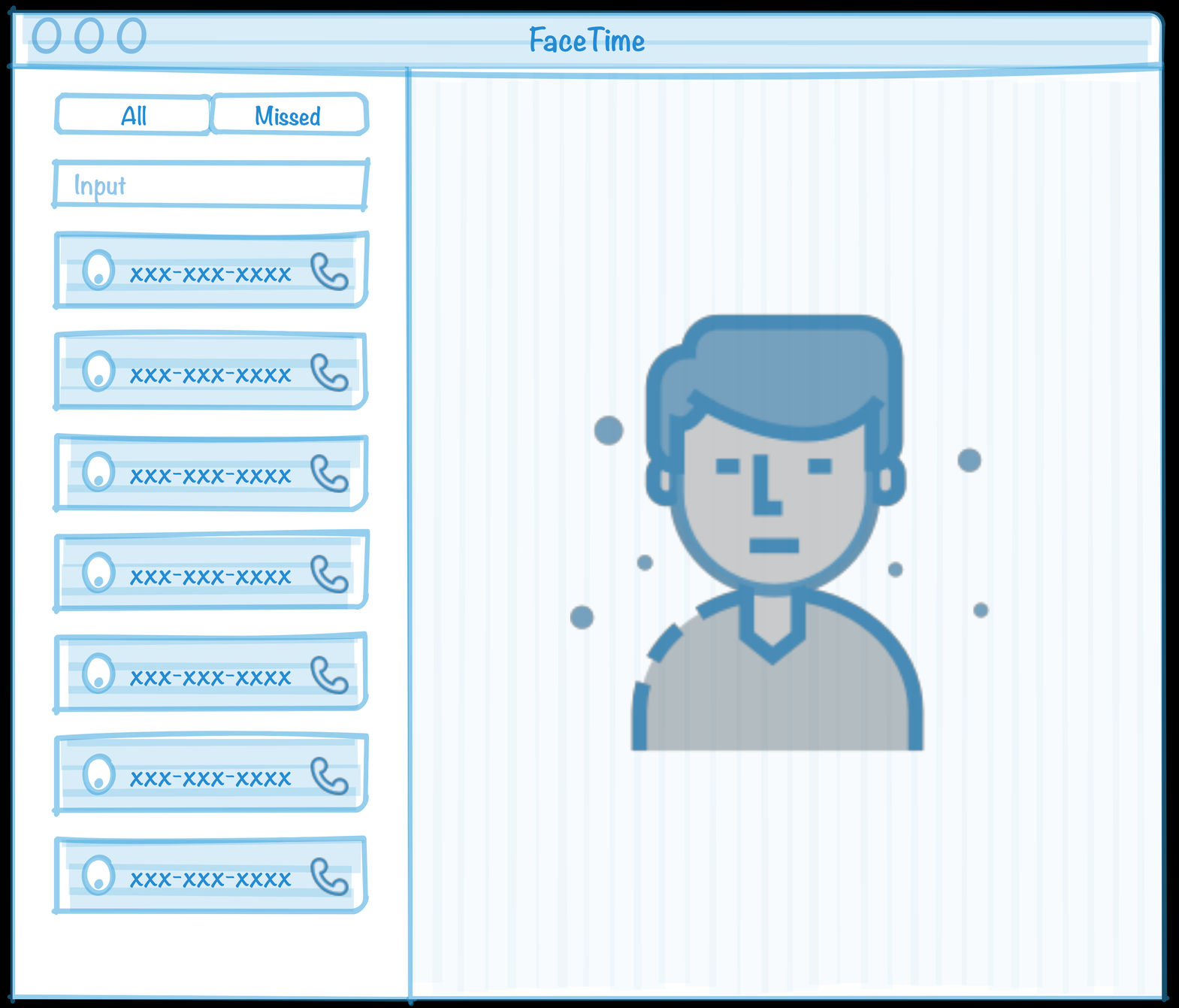
- Messaging apps are becoming a hot security research target
- Google Project 0 released a series of blog posts about fuzzing messaging apps, including WhatsApp, FaceTime
- We decided to take a look at FaceTime

Scope of the talk

- This talk will cover
  - Code execution flows while making a FaceTime call
  - Attack surfaces and vulnerabilities along with the code execution flows
- This talk will NOT cover
  - FaceTime protocol families (e.g., SIP, STUN, RTP/SRTP, etc)
  - Stream encryption, decryption, and storage
    - https://blog.quarkslab.com/resources/2013-10-17_imessage-privacy/slides/iMessage_privacy.pdf

Outline

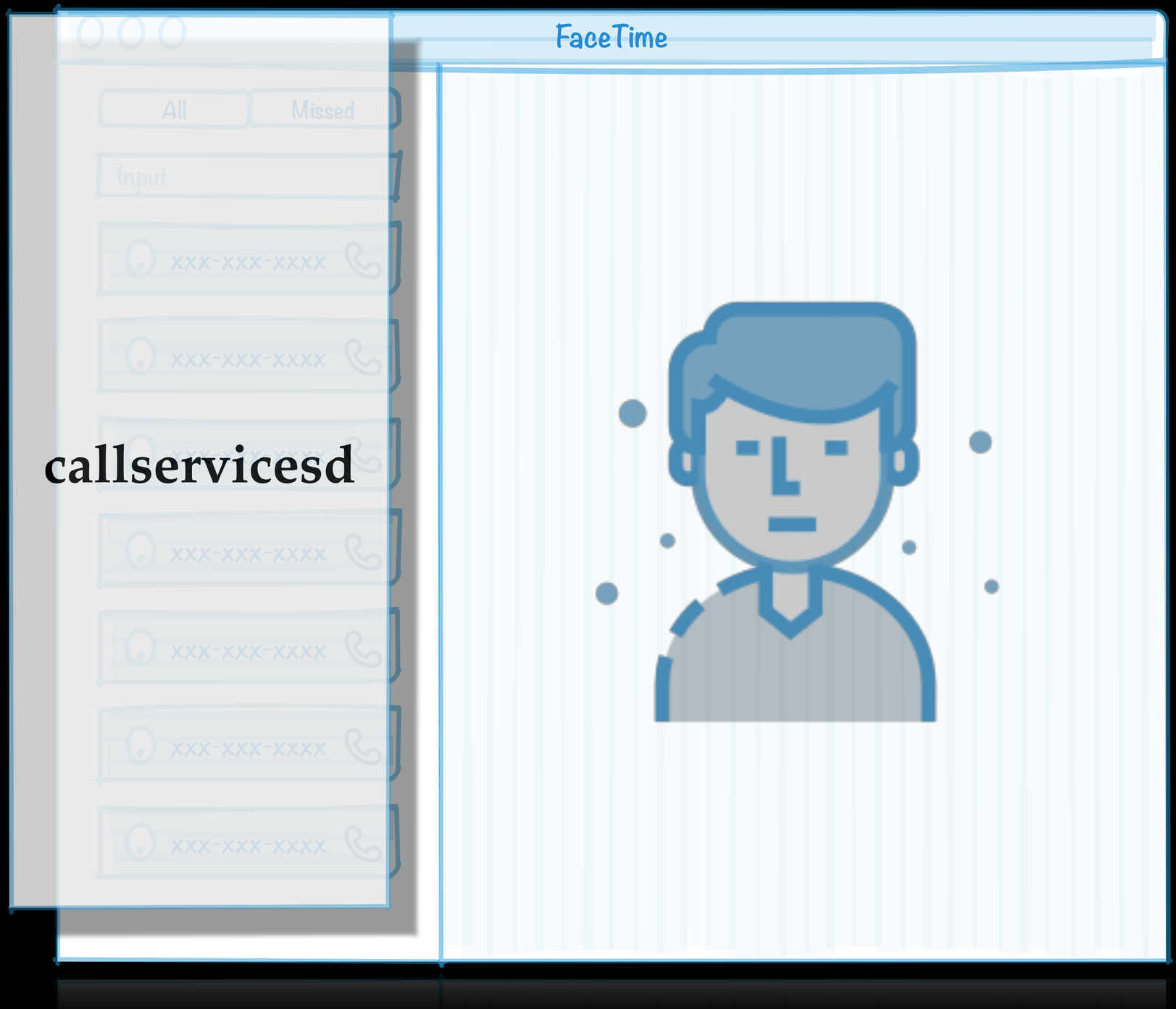
- Reverse-engineering FaceTime
- Attack surface and vulnerabilities analysis
- Conclusion



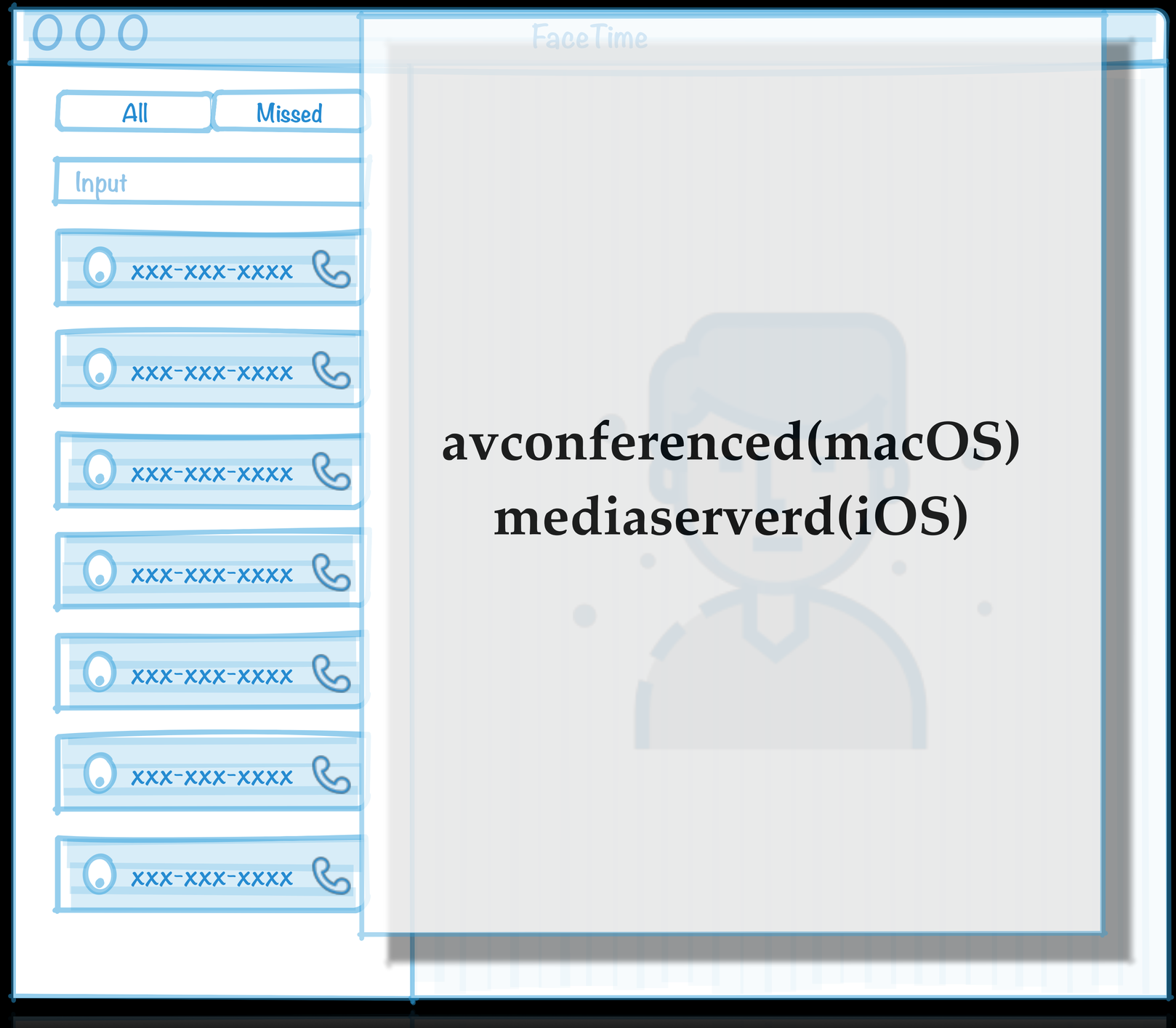
FaceTime is not a single application

- FaceTime.app provides the basic UI framework

- Manage the call status of FaceTime
- Respond to UI triggered events
- Communication bridge between avconferenced and identityservicesd



- Produce and handle FaceTime video/audio streams

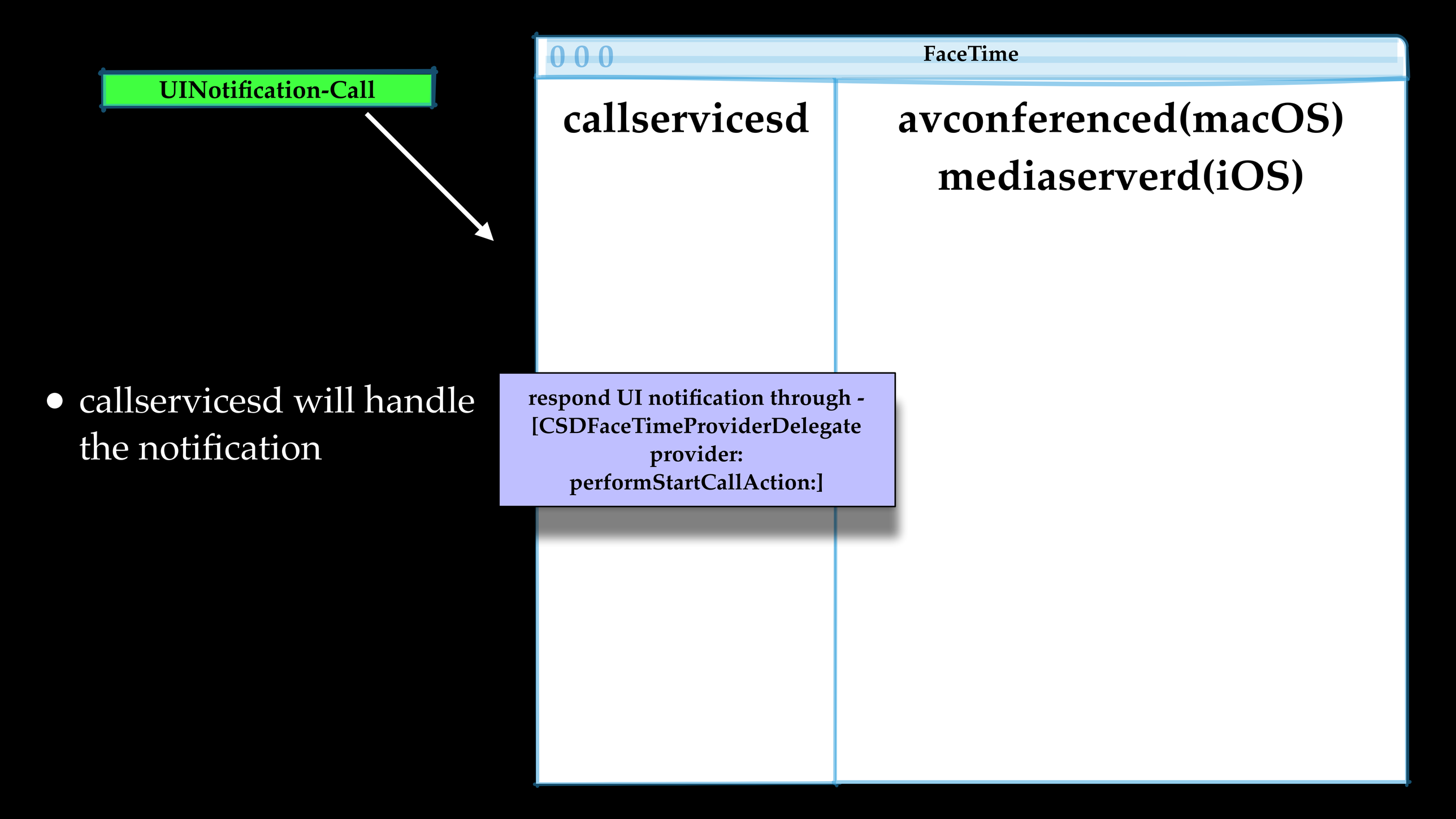


- We can consider that FaceTime consists of the three major components

- When a user tries to make a FaceTime call

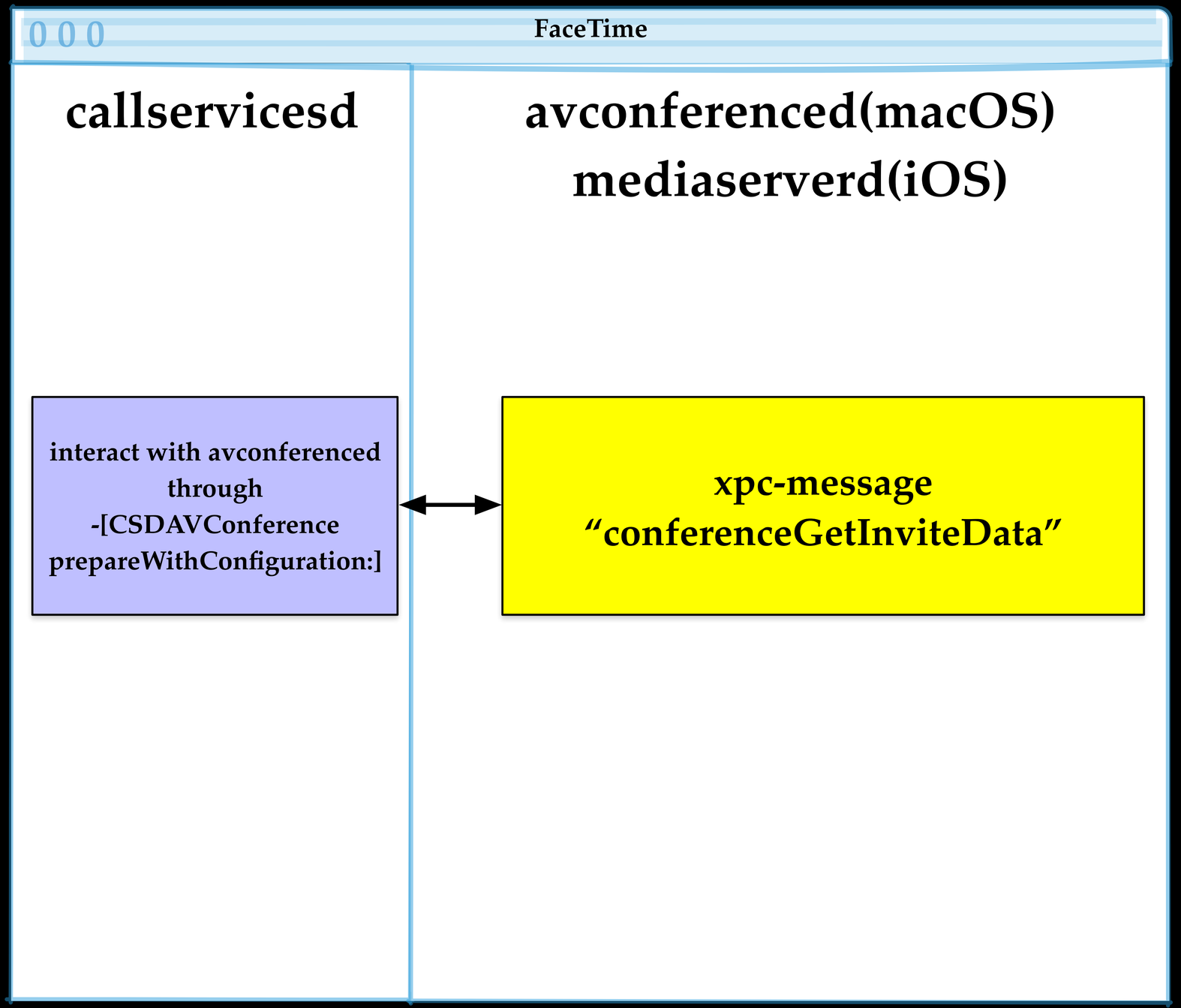
UINotification-Call

- FaceTime.app will generate a notification

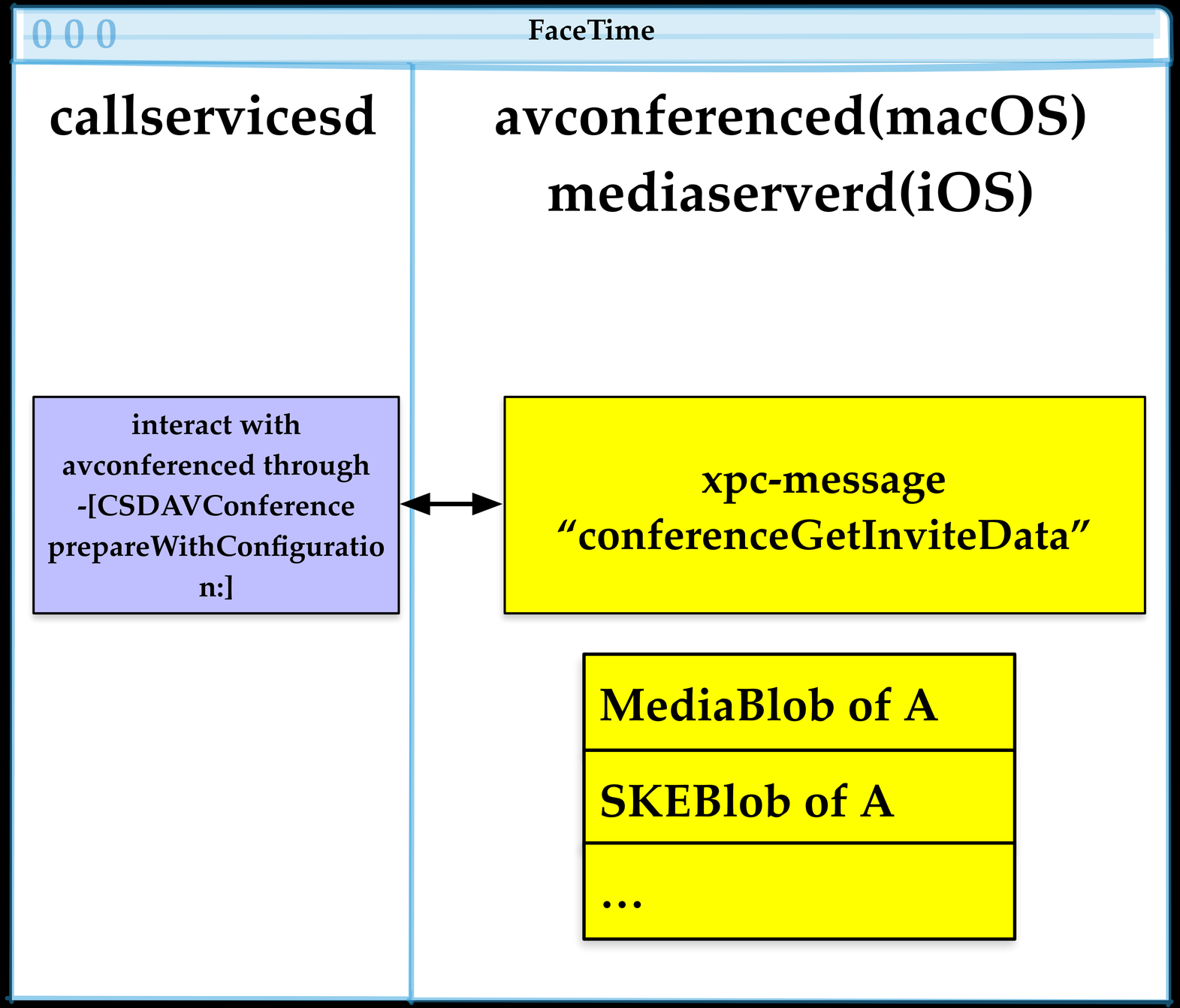


- callservicesd will handle the notification

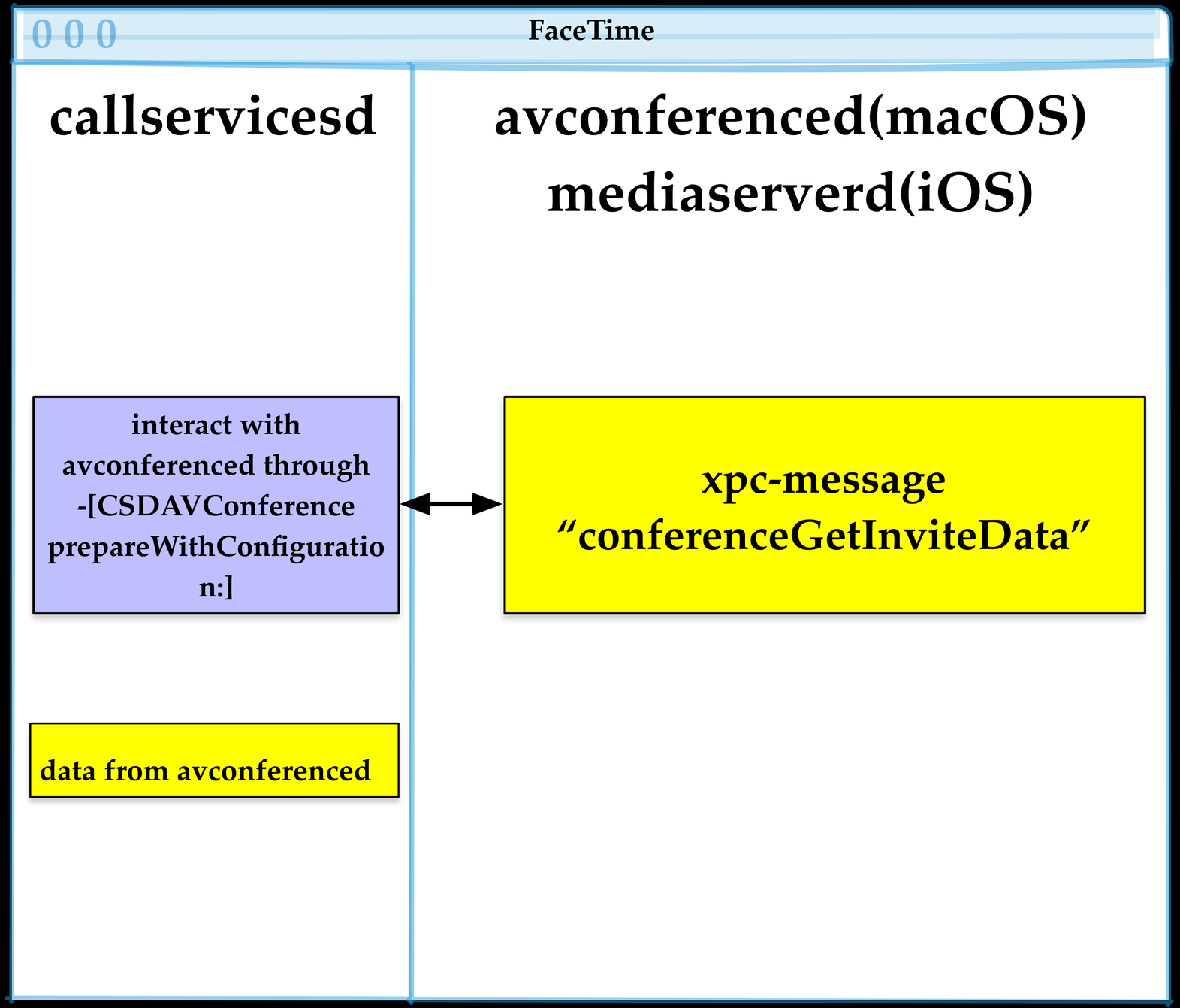
- callservicesd then invokes the corresponding handler to create a new invitation
- callservicesd sends an XPC message to avconferenced to get invitation data



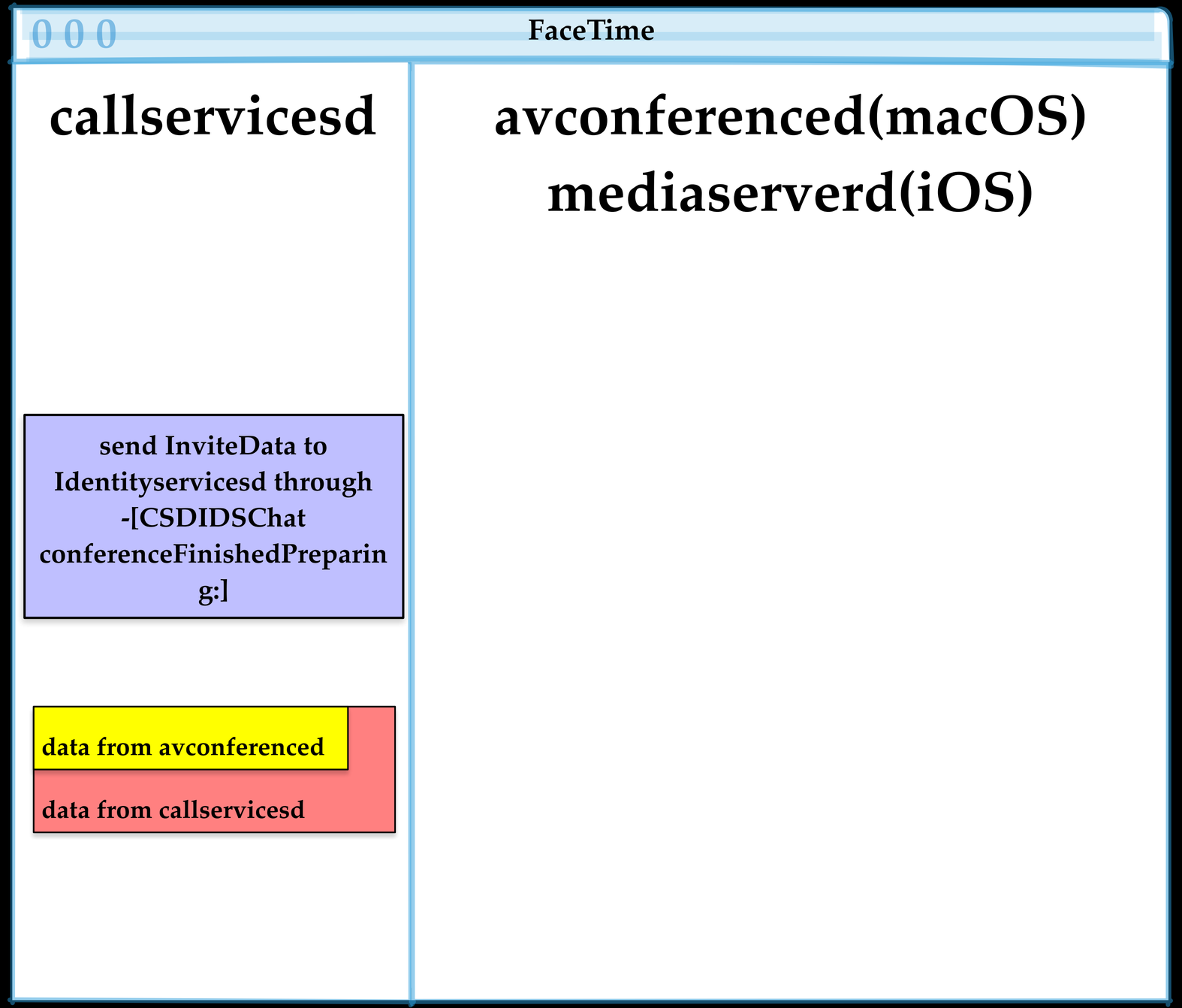
- MediaBlob contains configurations for audio and video streams
- SKEBlob contains encryption and decryption parameters for audio and video streams



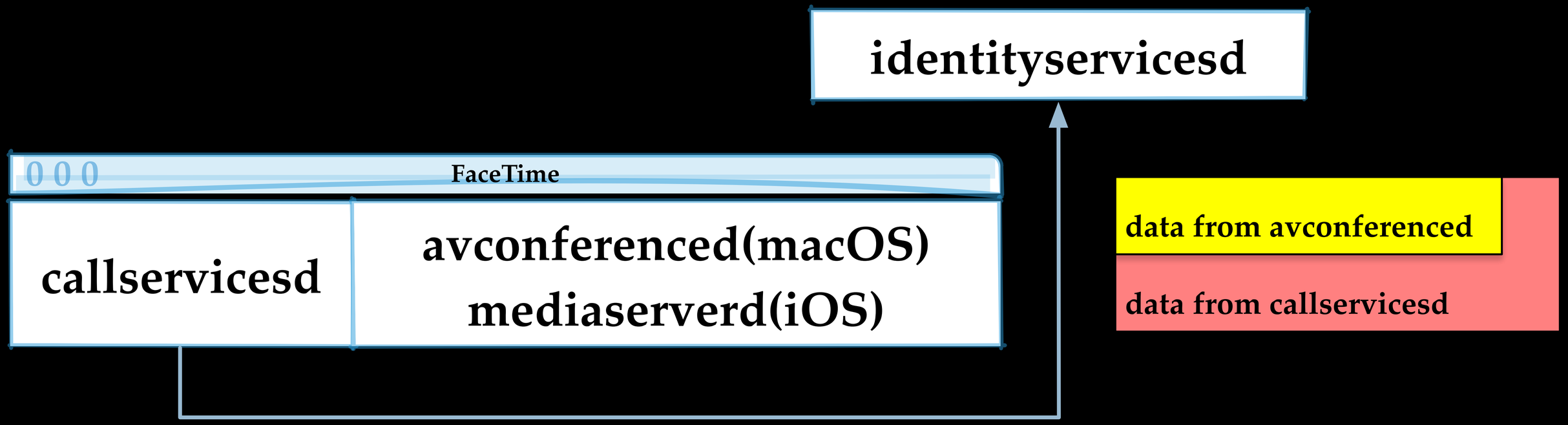
- avconferenced sends MediaBlob and SKEBlob back to callservicesd



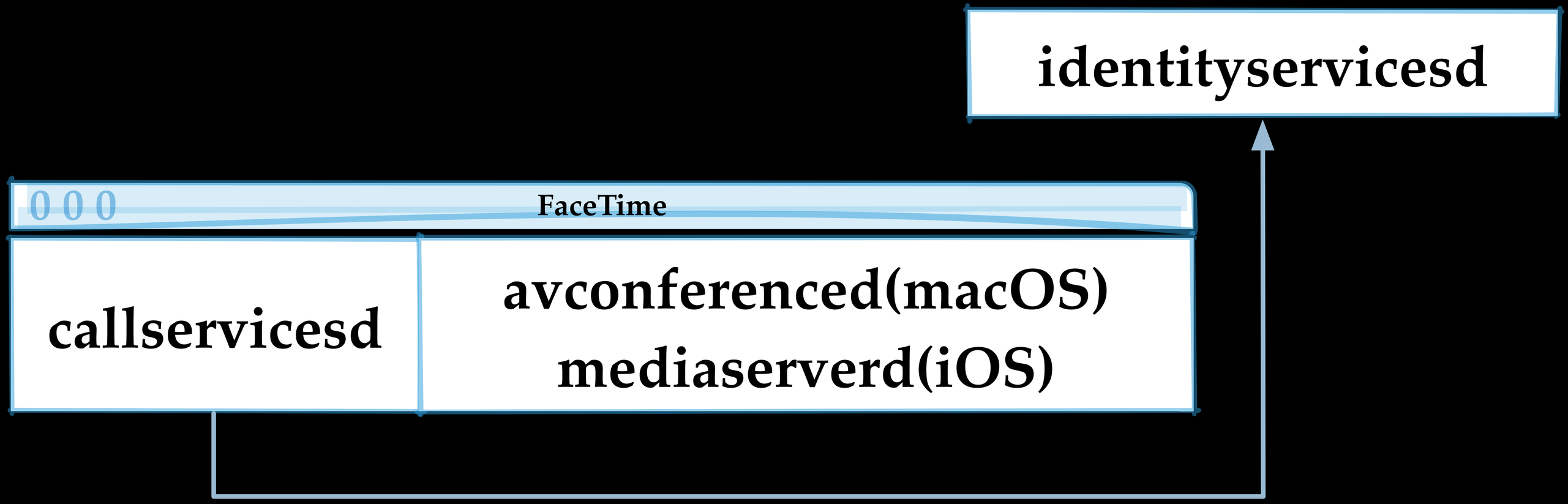
- callservicesd continues to encapsulate more information



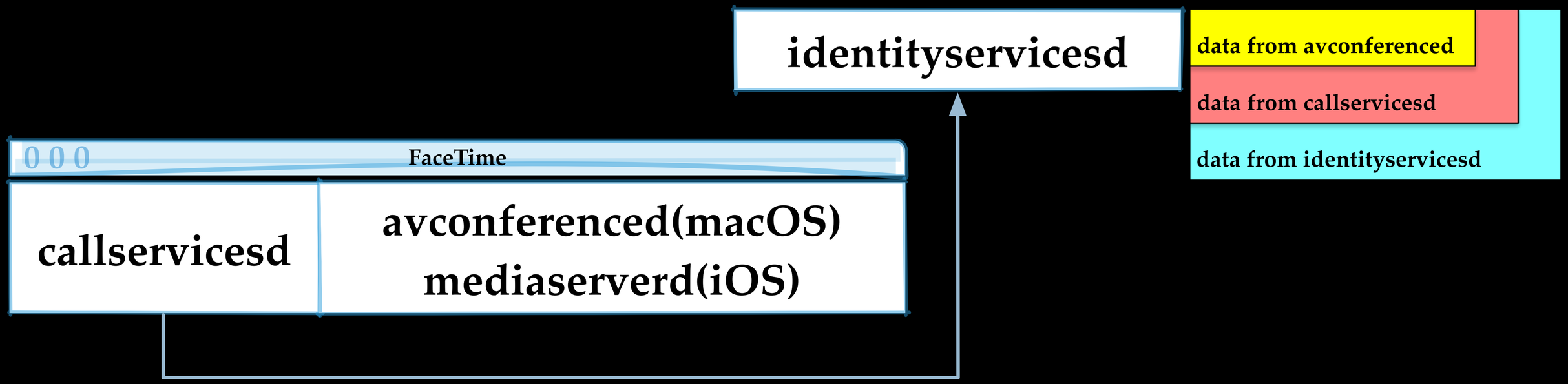
- callservicesd passes the invitation data to identityservicesd for further encapsulation



- Identity Services Daemon is a system process that handles credentials for various services, including iCloud and iMessage. It also connects to computers and iOS devices on your local network to coordinate phone calls across multiple devices.

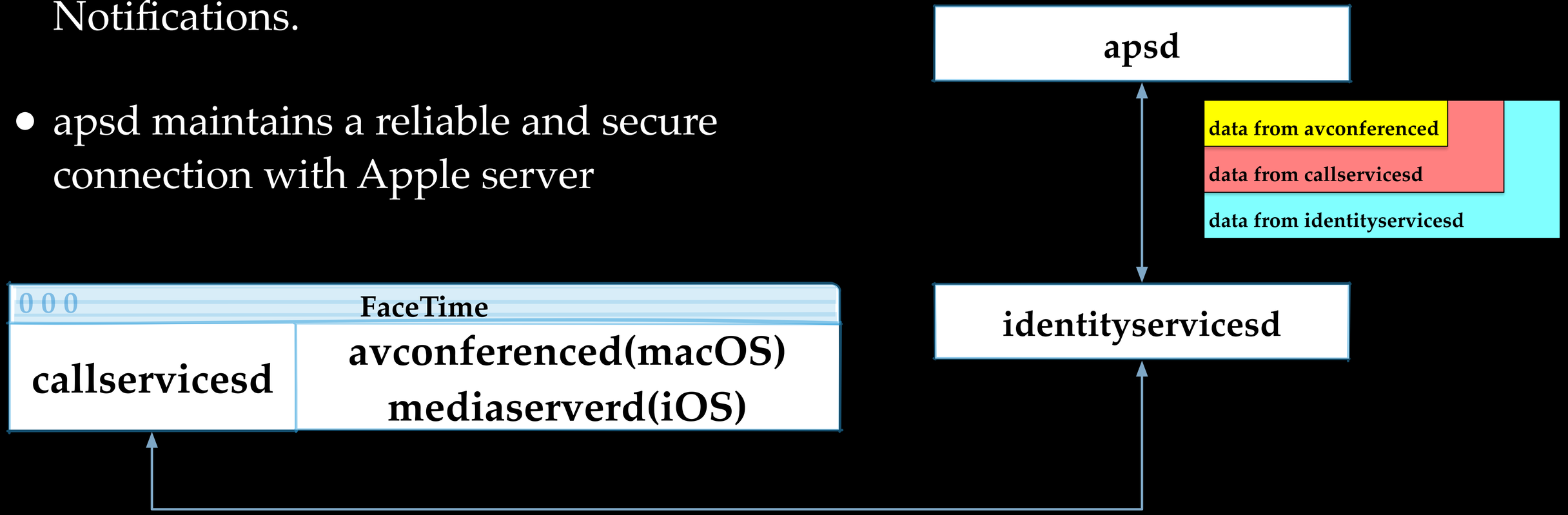


- identityservicesd continues to encapsulate more information, and then passes it to apsd

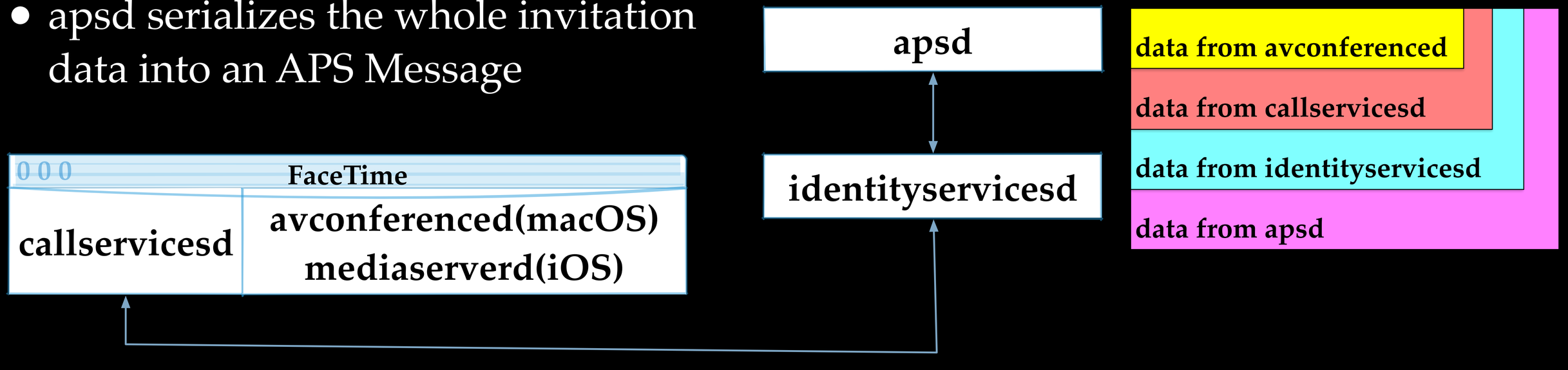


- The Apple Push Service Daemon (apsd) is responsible for sending and receiving Push Notifications.

- apsd maintains a reliable and secure connection with Apple server

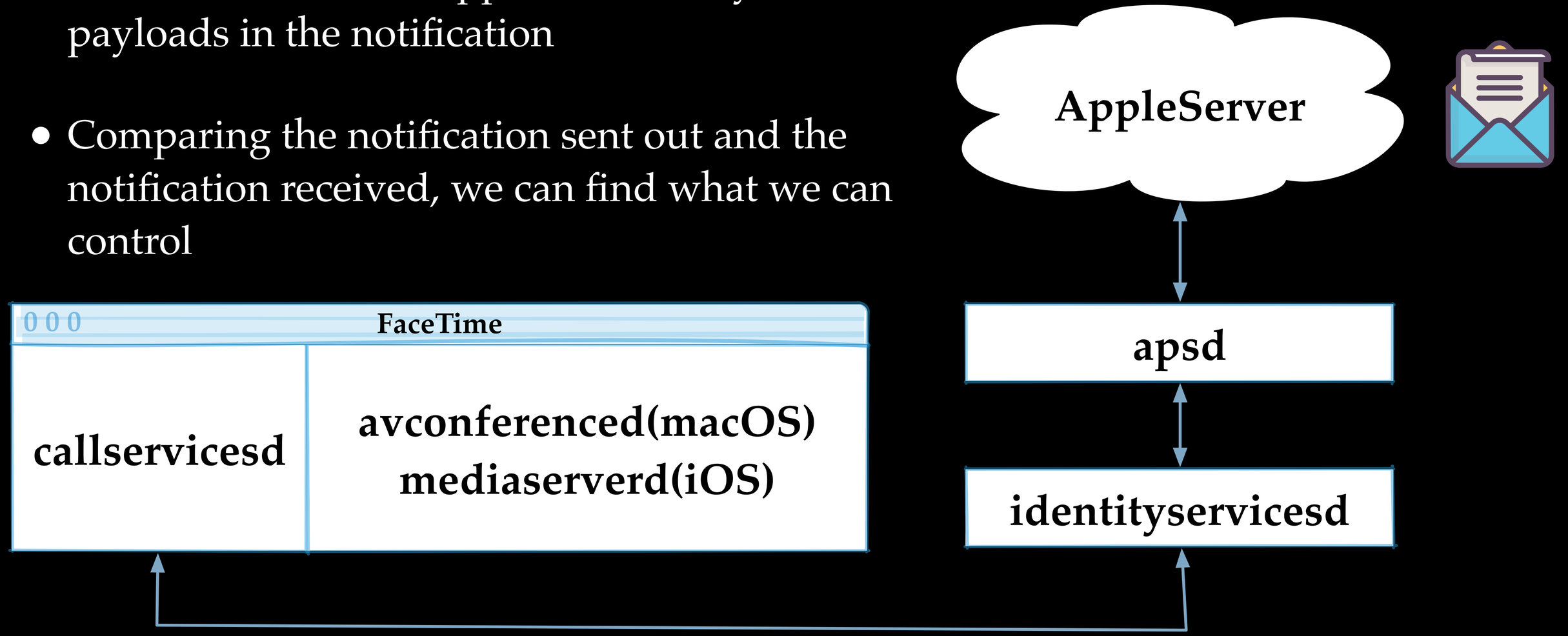


- apsd serializes the whole invitation data into an APS Message

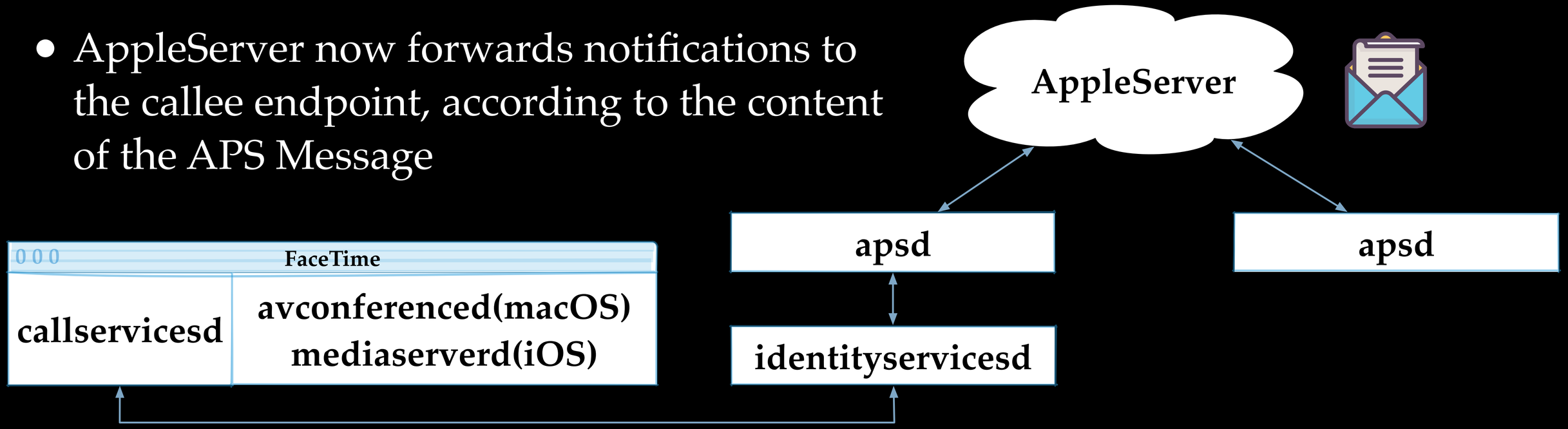


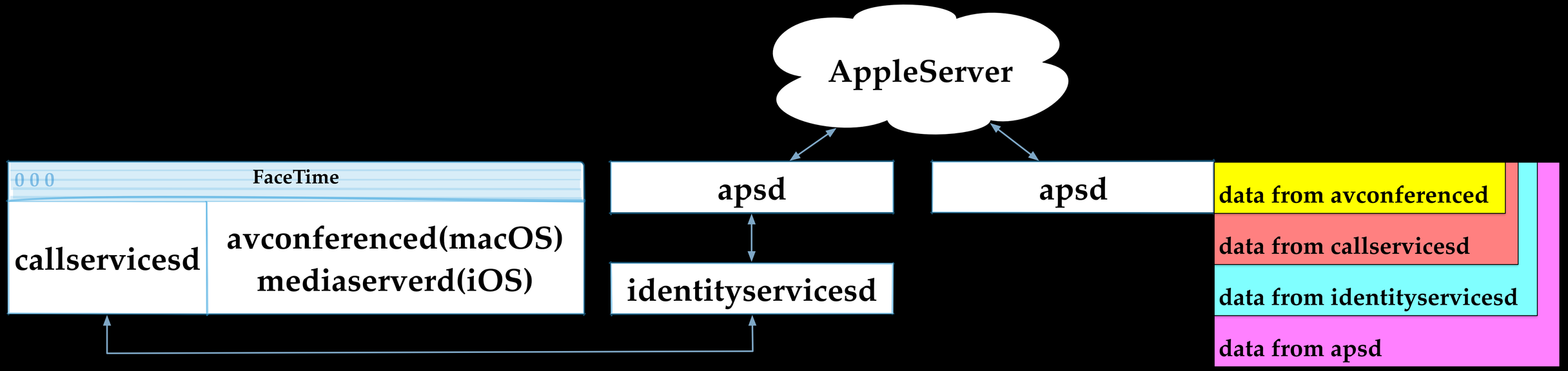
- When an endpoint sends a notification to Apple, Apple does not simply forward it to the destination, instead Apple will modify payloads in the notification

- Comparing the notification sent out and the notification received, we can find what we can control

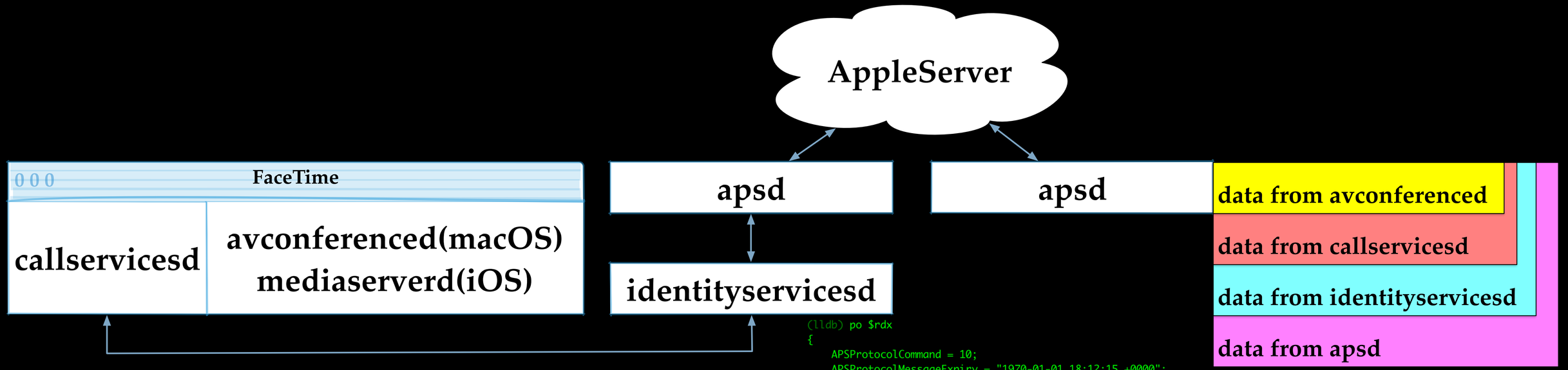


- AppleServer now forwards notifications to the callee endpoint, according to the content of the APS Message



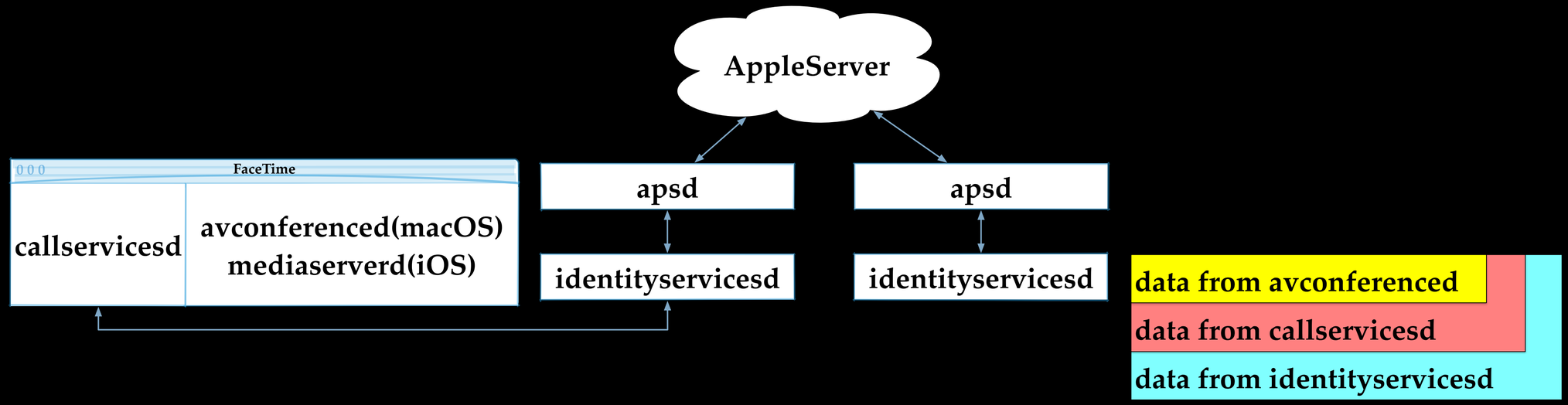


- The callee apsd deserializes the APS message



- We can set breakpoints in APSMessage deserialization functions in a debugger, and check the received notification
  - APSProtocolCommand = 10
  - APSProtocolPayload = <…>
  - APSProtocolTopicHash = <…>

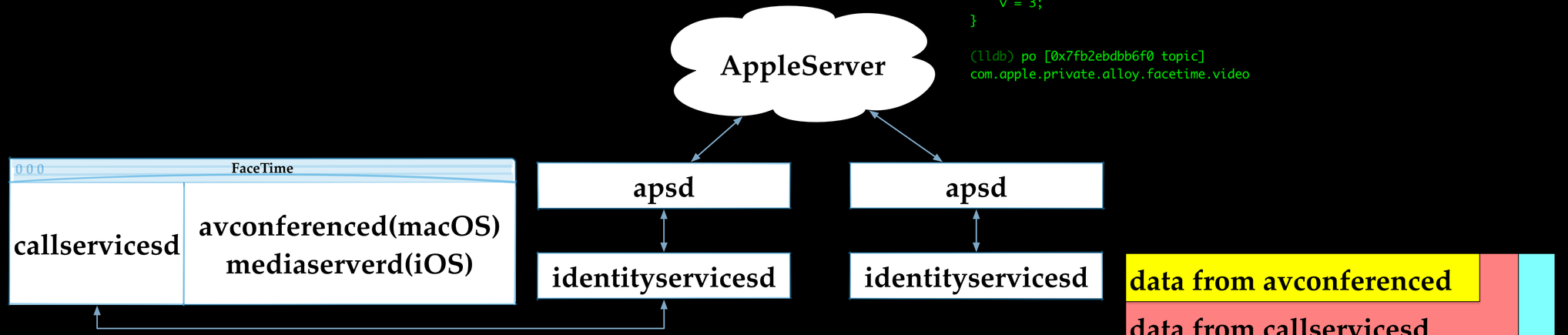
```
(lldb) po $rdx
{
    APSProtocolCommand = 10;
    APSProtocolMessageExpiry = "1970-01-01 18:12:15 +0000";
    APSProtocolMessageID = <78e5cae4>;
    APSProtocolMessageTimestamp = 1562140214245861022;
    APSProtocolPayload = <62706c69 73743030 da010203 04050607 08090a0b 0c0d0e0f 10111213 14527350 51745163 52655851 45517651 6
052 74505155 5f10176d 61696c74 6f3a3430 39333837 33383640 71712e63 6f6d4f10 2025837a bb6fef1b 5de3c659 65bf956d a0185b9f 8e060
672fc72f 3f59befc 9f10e810 00547061 69721003 1315add6 3f8b0bc8 844f1105 a502055a 554439d5 9e5c7f11 8f8a9186 bd69160e fcef0c66
7624 e41a60a1 f5fa7cbe 1ca6dbb7 88fa92d7 3bce923d 56121a1f 50c01a28 1a69ff7a 08a205d7 a7344e03 23c9e258 6933a201 9cc8da35 fdab
 2bc0148e 8753e660 757db443 ccbba2d9 ba090eb5 b3ef17ce ea84068a 8d4f2742 f30c93a2 d273ed5b 66e38a61 5b90817f be125f74 7458c85a
e1995 608d8703 a4cddb45 c10e8380 13e15998 2b8f02dc c0f601d8 4329a236 c85fd5e8 d36294aa 57cf3b70 cb5da541 d2cfdc35 e5df60b3 d41
1 b9169b76 7a2054d4 41e847e4 3f7d1c67 4f7468e1 27ae2f3a 4e00db3a 43f028d1 6f2b028d f2cbeaba 5c8cb831 7243133e 108e13c9 e15abcb
c827f7 095c64d1 c1901852 796a1b03 1eb53d58 73420942 faf98c2f 85adcf34 c2719ab7 c5d585f1 d6376833 5dac456c 8c76a629 93151c29 98
4d f97bbcd3 b8fd893e 2d2e91b3 4a0aa45c 9398dfe9 1beb9e62 03cb0d5d 6cf4189d 07a9a852 606378ee b2663cf4 7a097d34 3bee5991 372026
fd67e89 9e72f631 898847cf 61e2095b f9c1a6dc 37b81e15 88a8e168 9adac339 e6b6a38c d652808e 230f1917 41a326d9 7941fdc8 852fb7bc 3
814 6e5c9a22 d321de8a 17d62d97 73d57e6c bda088df 3c77396a f0f3083c 0d6b050c cfbc797f fd2e29b9 12cb243f 63ef633c c3985b91 9d720
e7ad0e76 0ed05f14 ab08c096 57117ece 96e43f43 fb35f0ce d4be3598 a4f915b4 8c09d28a 4c58003a f3ff1e69 2b8595f4 76d87787 6f127eb7
26f4 af836923 e871170e 77acdb13 8fa10102 64f87579 47469988 9ff7fdc2 acc44a1d 64d7a126 33095001 ebc7653c a0f1568b ef41f92f 189d
 de935a36 84036037 13f7b89a 0be83d6a 89b98efc 22c26dcc 541cb950 44db0760 553dfd85 d90c92db 13638fcb ece30c29 8af7e982 6bf2df4d
6e5b8 e5d944e1 fb8bcd10 2d2d9ed9 618bf6f1 bee58e91 b34e6c5d fd416cb5 c5a6a524 5a1594d1 5c883e73 49581adf 74fc6918 c710ae67 4d0
7 08cea2c2 3edc8be2 6159f3aa 63636b84 df9b83a4 20040df8 0d809d93 f29e625e 4e3904ba 9ab2324a 2a411e8e ccc6bcd8 3f32c71f a0f0b92
f670b7 cc7fb086 a102448c 4fd4e490 e35bda9a 05d6f09f 71cafa72 a15c3d86 18f315ab 2a92890e 0c29ae9c 75e5d62f 315c0df6 e3f455b8 51
10 8f07513f ef7f8ada 8e387ec2 0a2e9f7f 55741877 e17b45cf 8c5cb328 f9b6b515 4d096eb8 b0513a89 29700ebb 2770b6bd 594a4cfa a8b309
9861bce 866cae1d 0a5f343d 49113e39 7d191146 bb240085 07bcec3c 3585f75e 47ac032e 661f2388 c7b53d17 93d81e54 084abafd abe87a1a 0
9bf 8a765652 0b31096f 2bc407b1 8c70c307 b1fd68e6 0d6ea80c 9b00bc2b 9a7064d1 1da2f84f 7a5420c1 7df2fe69 87f2baec ad64a5ca f3de1
392e43e0 173510de db5c7ba8 c8e46d95 7465d3ce 853042bb b7ea9a6d 4ceaa3c2 f2b94efa 92127579 a46f7784 5325b450 370c0206 9e112cad
100f 08106173 d19eb74a 227e6802 034989cf c2364604 22817bc3 4b9e95d6 6e224751 dd123d2e 8620fcd0 98c277e0 dc356d86 69e6b48f 0143
 1b8152b5 e803c50c b99d78b5 1a9e5f5c 1014ec9b bb986ee0 33c345d2 42a3bfc0 32ad1f20 088ba0b3 08033752 384b1bcb 5defbdb1 ce81d66c
51062 df2b98d2 1ef80a88 871fe3b4 4d78916b 4de31922 d192140a 0b11e137 1424186c 3a919754 5b42e870 3084e6aa 94fac71c 53b50116 422
7 3692e662 1cc66268 61483ed2 fe68acd1 388f91a6 22314b89 e143e612 8c372402 6312c188 3060e45c 5580482a 017e3ed7 fd9bf5e5 159960f
d38cd2 cb8ee2a5 43584f67 2c12923a 7f0ef2b6 249f1a29 1d402626 9de92081 8e61d816 a2870654 50b2b6f6 d15acdf4 dd3f86e1 6874fd1f 1c
a0 4aa0d8b5 c3124512 6eba45b7 3d02ed7e e8384730 45022100 c695ff08 933a7dfe 5939d0dd 18c95ffd 67fa5bfe 1584615a c85c22d1 0929ea
2201e46 3161993b 83a3bc79 ebc6efe6 84be5721 63650e9f cadf59fc 5698092c 88a35f10 1d6d6169 6c746f3a 74757269 6e672e68 75616e67 4
d61 696c2e63 6f6d4f10 103eadc0 c986fa4c 5f80d5f0 338dc399 a3000800 1d002000 22002400 27002900 2b002d00 2f003200 34004e00 71007
75007a00 7c008506 2e064e00 00000000 00020100 00000000 00001500 00000000 00000000 00000000 000661>;
    APSProtocolToken = [redacted]
    APSProtocolTopicHash = <5dd96535 5dda3aab a83fb0c6 a129c430 cde5d01d>;
}
```



- According to APSProtocolTopicHash and APSProtocolCommand, apsd delivers the payload of the notification to identityservicesd

- We can set breakpoints in identityservicesd and check the received invitation

```
(lldb) po [0x7fb2ebdbb6f0 userInfo]
{
    E = pair;
    P = <02055a7b 3c4adccc 4495ae7f 7f140d29 13f8f24d 9fc48c62 c88fd217 e2b2f6f8 43524a05 d5712358 471931a2 11bc302b eb3d5486 eada
53c1 cc475e8a 71ec8ac8 6fb56c76 7284678a df8df42c e9d8f23a c3dc7dae 1d7af80c faa868b4 c8ba0800 126b3d3a 0b914fc9 c9dfaa99 0b889dda
 4cab19fb 0cade846 cbfb3130 c3f847d0 4c756343 e5fce26a fa6e5d73 0000066c 7b8039a4 29acd55c d76c078f 141b9c3a 1a0d67bd 6238c8f7 630
e4485 5b79c98a 2e289da5 7e62d0e9 bfc26ae7 27efa387 46c5a06a ca9ebb47 eda35dfa e7d7df2d 977f0a8b 5d395ada 2b896a50 7f9eceee 2f2831c
f dd957a87 27ca52cd fc455512 5c5f4d73 ca4eaf52 4a04491a da50a249 03adbac9 b26eba59 f6a27887 36ad5954 5d36ae9b 086cb4b1 848f835c fc
ed4736 936427c6 bf758079 2001c0a2 c311f8ed ea7a67e6 b5d61e55 a423cd4b 0443bcf1 92c79f47 1505c31a ec86415e 2d1a93f6 b5fbd80c 774b48
af d4533c7d fdf20fcd f3bacce9 62b24e9c a40963b5 ffe3050a ac04ca8d 7abb657c 1085b4ae b48a1190 d9eeb4dc f430eda6 da502566 da6c5ae5 4
2d8d362 bfe91488 58071bfa 2c8e9edd 31224718 e4e18760 cbc4a4b2 eb968094 db6f74eb ffb28a27 69067923 cb3adbe7 1c4c9822 17ab2dec 5173f
85a 4273df8c 2dbd2dcf e169588f ff54799d ea09fa4c 2e79a249 091dd613 2d04add1 910cb405 533684d7 bafa84ef 85552073 9e3c1aec 33aa7a4a
9508fbe6 a6b8133d 40098616 73267310 f467bb4f 09b879b0 1f84e02e 4fb762d2 85852480 a44b62ba 288dbf24 0532bc17 29798243 ca1fbe12 d149
97e7 58a86fe8 05def00e 353fa58c cb7ebd1f 0f239de2 b196abae 9c89e486 a74d30b2 11eee732 0a048e62 88c51097 32e95562 59e689f8 7c66181d
 c9b43831 e5841164 fcddd4bb 88ce7227 cd0f1019 cb2dfd3f 43c008a4 1f7bf93c 65e7ead6 431b49a1 6f5bec10 82de2fbf 2e21adc5 767e8633 f44
3989e 9cb5d10a 4697ca37 7494feb5 8dd3b9b5 8f758d0f f2ae8d60 5ebc7255 6e9d93a4 07232f91 2add387b 026eef2b 83b8e68e 73dfdf94 98cc485
b 28dde4e5 8601b559 f1088584 de5730ba d2cbf71e da3ba502 fb248c20 eb819ad5 5b27e7e9 68c72b04 a966f6d5 b7c93f81 9fb54c6e eac0e339 71
e05f12 0f1c01dd 07ecd28a 62e7eb12 a4414cc2 93dd5c8e 7ab2f229 54bbbc60 8ea24295 b12cff54 8f4918d3 6968545b 83e2c450 02ec0ac1 77d8bb
13 baf807b6 ccf47c2f 1dbfa512 f72354fe 245b08f1 50c98fa3 dc5bddf1 3ab5639d 788dfafc 9aca46ba 80a29d18 9dd89aa2 dfd1a9d2 854c4683 6
c2e4d8a 83378e06 5399c330 742eab04 288d915e 25a2f912 0334b9e5 73746488 925e9a66 66f297c6 68a97b93 90559696 1f7515f1 83ea98b2 cf867
2e6 12a2ee03 949f3edc 0f1cfc77 c9f5386f e3ecbb76 c3e04473 d9b56db8 96d9c0f6 102a53af 1cd7bae8 11be2983 0749804a 91947a6f 2bf6ef2a
0252dbad f90fe818 efc5f0f3 10db4fd7 da1af532 fed374e0 370f48f9 4bc7ee33 007c9e94 de81e7a8 56400511 f4515cf6 7bd87a6a 8b33ab90 201c
e4bf 576badc6 7e8b1fad f45353ce 73f50b1e 3bd25ee5 b946409b 13bce169 0a614e0e 825e4035 5c3c324b 2f12ad5c 42698fde 4e2ecc8a 4155758d
 da6ef604 18361132 d51ec2c4 3970aff4 94ec97aa f95423f0 b8a7a9a0 9d225cf0 2b486984 f51d6d38 e5f55659 1743aa0f 459d0bed d7eb8c3f 3d2
fa2d0 64726458 5b821ffe 0d84d198 1e7bb4f5 97255b99 7950af2a 261c523b 11beac74 ebe7146f af56b1b9 5c7dc5d3 0ae9e59d 44330582 eb2209d
e 1c280e2c 5323463e 7696d6f1 6f61850d aa4d257f e0b908c8 b6c1447a 12fc6064 443b58be e68fcbb9 e9968efd 0f28eed6 3b377069 f54a070f 4d
de5e09 8944a236 ddf814b2 c6784e0b 4b178752 f2eefbde c4b971c2 847e1c6a fb23c874 401a4bda 2dec66e7 8a9874aa ab483046 022100be a7da40
c4 caade90d 5a88b0a0 1ed2efe9 253fb7a6 c6cc530b 8e150663 960ba302 2100a8ab d40454f4 ae3f4a2a f9cb3bc9 e6d0b5da cd7dc7aa a290342d 4
0183711 bad7>;
    U = <3dbf [redacted] 441>;
    c = 232;
    e = 1562140602210643773;
    eX = 0;
    sP = "mailto:[redacted]";
    t = <258[redacted]efc9f>;
    tP = "mailto:[redacted]";
    v = 3;
}

(lldb) po [0x7fb2ebdbb6f0 topic]
com.apple.private.alloy.facetime.video
```



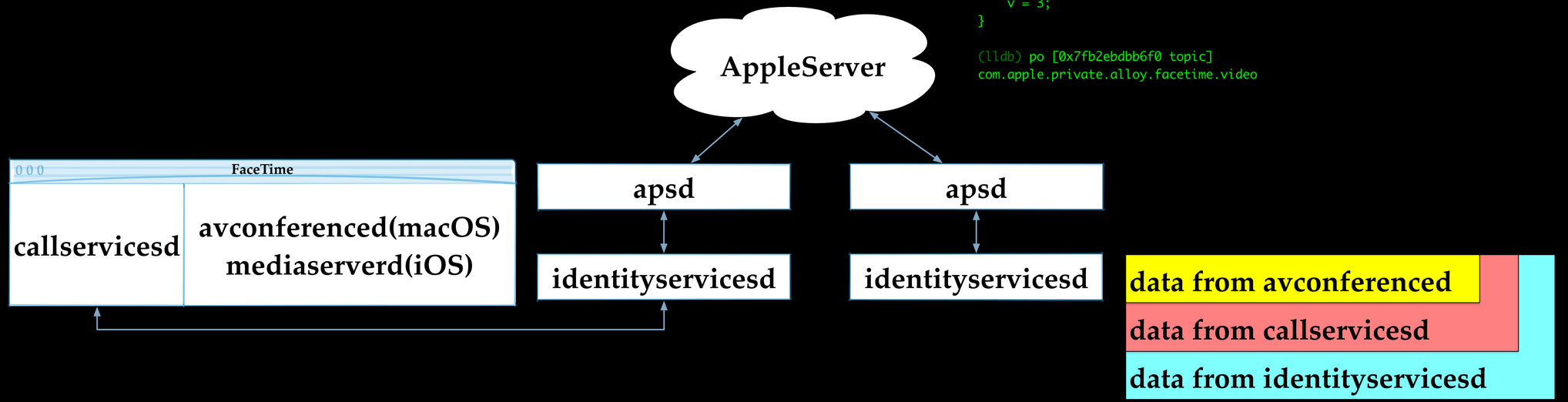
data from avconferenced

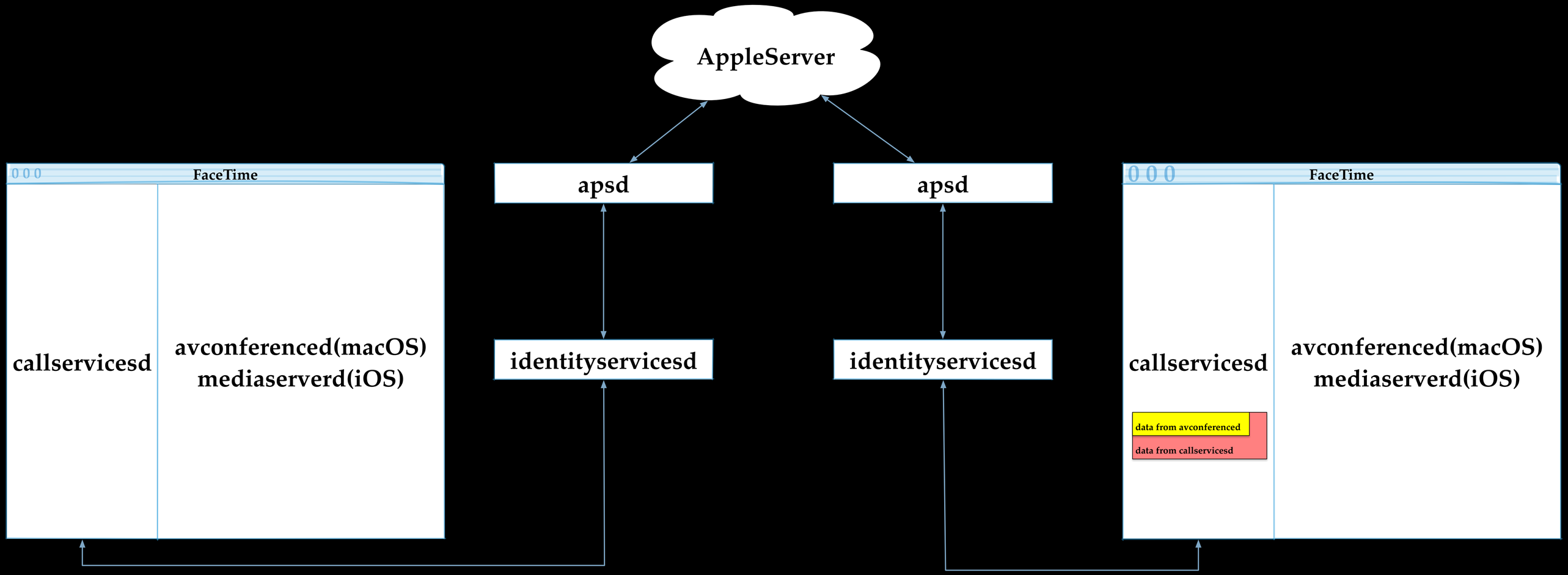
data from callservicesd

data from identityservicesd

- c=232 -> processIncomingInvitationWithPayload
- c=233 -> processIncomingSessionAcceptMessage
- c=234 -> processIncomingSessionDeclineMessage
- c=235 -> processIncomingSessionCancelMessage
- c=236 -> processIncomingSessionMessage
- c=237 -> processIncomingSessionEndMessage
- …

```
(lldb) po [0x7fb2ebdbb6f0 userInfo]
{
    E = pair;
    P = <02055a7b 3c4adccc 4495ae7f 7f140d29 13f8f24d 9fc48c62 c88fd217 e2b2f6f8 43524a05 d5712358 471931a2 11bc302b eb3d5486 eada
53c1 cc475e8a 71ec8ac8 6fb56c76 7284678a df8df42c e9d8f23a c3dc7dae 1d7af80c faa868b4 c8ba0800 126b3d3a 0b914fc9 c9dfaa99 0b889dda
 4cab19fb 0cade846 cbfb3130 c3f847d0 4c756343 e5fce26a fa6e5d73 0000066c 7b8039a4 29acd55c d76c078f 141b9c3a 1a0d67bd 6238c8f7 630
e4485 5b79c98a 2e289da5 7e62d0e9 bfc26ae7 27efa387 46c5a06a ca9ebb47 eda35dfa e7d7df2d 977f0a8b 5d395ada 2b896a50 7f9eceee 2f2831c
f dd957a87 27ca52cd fc455512 5c5f4d73 ca4eaf52 4a04491a da50a249 03adbac9 b26eba59 f6a27887 36ad5954 5d36ae9b 086cb4b1 848f835c fc
ed4736 936427c6 bf758079 2001c0a2 c311f8ed ea7a67e6 b5d61e55 a423cd4b 0443bcf1 92c79f47 1505c31a ec86415e 2d1a93f6 b5fbd80c 774b48
af d4533c7d fdf20fcd f3bacce9 62b24e9c a40963b5 ffe3050a ac04ca8d 7abb657c 1085b4ae b48a1190 d9eeb4dc f430eda6 da502566 da6c5ae5 4
2d8d362 bfe91488 58071bfa 2c8e9edd 31224718 e4e18760 cbc4a4b2 eb968094 db6f74eb ffb28a27 69067923 cb3adbe7 1c4c9822 17ab2dec 5173f
85a 4273df8c 2dbd2dcf e169588f ff54799d ea09fa4c 2e79a249 091dd613 2d04add1 910cb405 533684d7 bafa84ef 85552073 9e3c1aec 33aa7a4a
9508fbe6 a6b8133d 40098616 73267310 f467bb4f 09b879b0 1f84e02e 4fb762d2 85852480 a44b62ba 288dbf24 0532bc17 29798243 ca1fbe12 d149
97e7 58a86fe8 05def00e 353fa58c cb7ebd1f 0f239de2 b196abae 9c89e486 a74d30b2 11eee732 0a048e62 88c51097 32e95562 59e689f8 7c66181d
 c9b43831 e5841164 fcddd4bb 88ce7227 cd0f1019 cb2dfd3f 43c008a4 1f7bf93c 65e7ead6 431b49a1 6f5bec10 82de2fbf 2e21adc5 767e8633 f44
3989e 9cb5d10a 4697ca37 7494feb5 8dd3b9b5 8f758d0f f2ae8d60 5ebc7255 6e9d93a4 07232f91 2add387b 026eef2b 83b8e68e 73dfdf94 98cc485
b 28dde4e5 8601b559 f1088584 de5730ba d2cbf71e da3ba502 fb248c20 eb819ad5 5b27e7e9 68c72b04 a966f6d5 b7c93f81 9fb54c6e eac0e339 71
e05f12 0f1c01dd 07ecd28a 62e7eb12 a4414cc2 93dd5c8e 7ab2f229 54bbbc60 8ea24295 b12cff54 8f4918d3 6968545b 83e2c450 02ec0ac1 77d8bb
13 baf807b6 ccf47c2f 1dbfa512 f72354fe 245b08f1 50c98fa3 dc5bddf1 3ab5639d 788dfafc 9aca46ba 80a29d18 9dd89aa2 dfd1a9d2 854c4683 6
c2e4d8a 83378e06 5399c330 742eab04 288d915e 25a2f912 0334b9e5 73746488 925e9a66 66f297c6 68a97b93 90559696 1f7515f1 83ea98b2 cf867
2e6 12a2ee03 949f3edc 0f1cfc77 c9f5386f e3ecbb76 c3e04473 d9b56db8 96d9c0f6 102a53af 1cd7bae8 11be2983 0749804a 91947a6f 2bf6ef2a
0252dbad f90fe818 efc5f0f3 10db4fd7 da1af532 fed374e0 370f48f9 4bc7ee33 007c9e94 de81e7a8 56400511 f4515cf6 7bd87a6a 8b33ab90 201c
e4bf 576badc6 7e8b1fad f45353ce 73f50b1e 3bd25ee5 b946409b 13bce169 0a614e0e 825e4035 5c3c324b 2f12ad5c 42698fde 4e2ecc8a 4155758d
 da6ef604 18361132 d51ec2c4 3970aff4 94ec97aa f95423f0 b8a7a9a0 9d225cf0 2b486984 f51d6d38 e5f55659 1743aa0f 459d0bed d7eb8c3f 3d2
fa2d0 64726458 5b821ffe 0d84d198 1e7bb4f5 97255b99 7950af2a 261c523b 11beac74 ebe7146f af56b1b9 5c7dc5d3 0ae9e59d 44330582 eb2209d
e 1c280e2c 5323463e 7696d6f1 6f61850d aa4d257f e0b908c8 b6c1447a 12fc6064 443b58be e68fcbb9 e9968efd 0f28eed6 3b377069 f54a070f 4d
4e5e09 8944a236 ddf814b2 c6784e0b 4b178752 f2eefbde c4b971c2 847e1c6a fb23c874 401a4bda 2dec66e7 8a9874aa ab483046 022100be a7da40
c4 caade90d 5a88b0a0 1ed2efe9 253fb7a6 c6cc530b 8e150663 960ba302 2100a8ab d40454f4 ae3f4a2a f9cb3bc9 e6d0b5da cd7dc7aa a290342d 4
0 83711 bad7>;
    U = <3db[redacted]441>;
    c = 232;
    e = 1562140602210643773;
    eX = 0;
    sP = "mailto:[redacted]";
    t = <258[redacted]efc9f>;
    tP = "mailto:[redacted]";
    v = 3;
}

(lldb) po [0x7fb2ebdbb6f0 topic]
com.apple.private.alloy.facetime.video
```





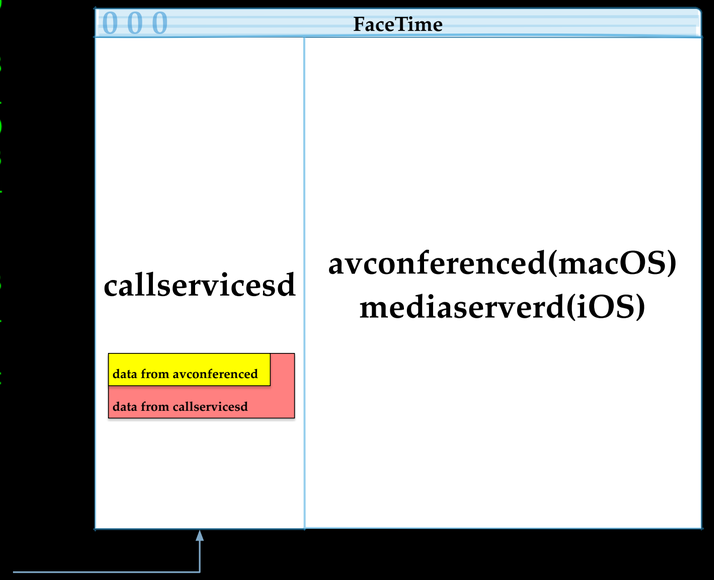
- identityservicesd continues to forward data to callservicesd

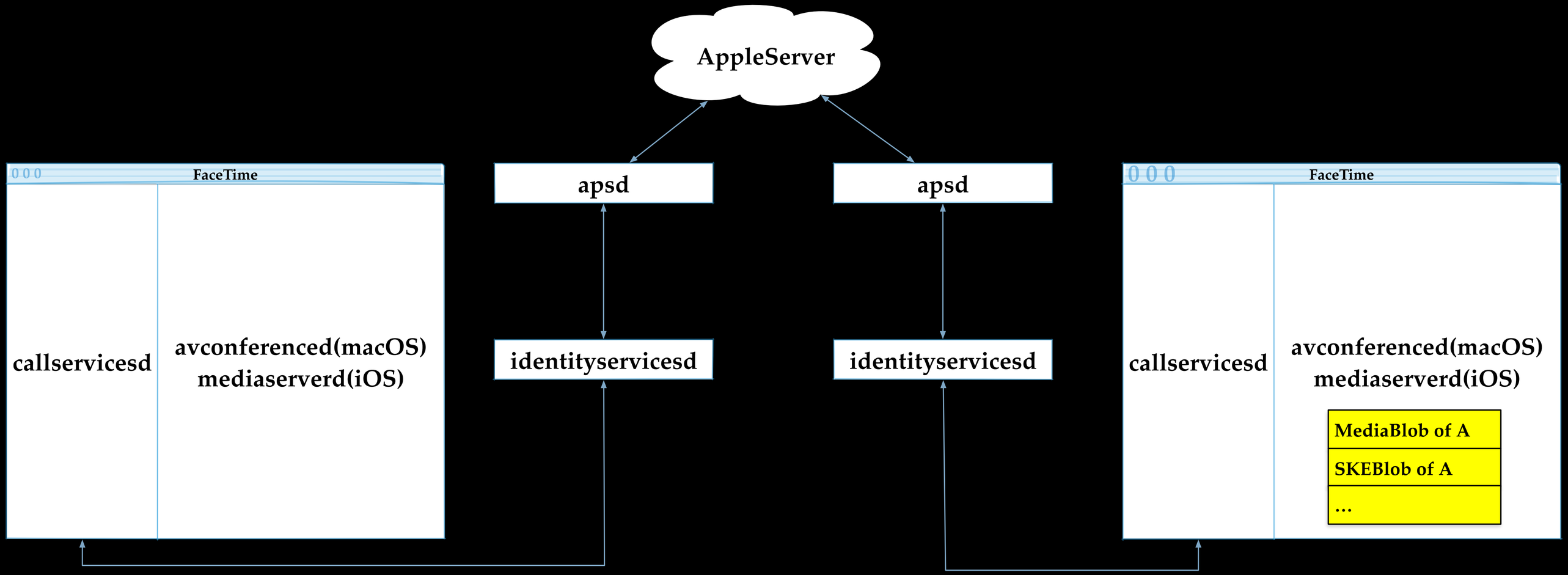
callservicesd parses binary payload through

-[CSDAbstractIDSProviderDelegate

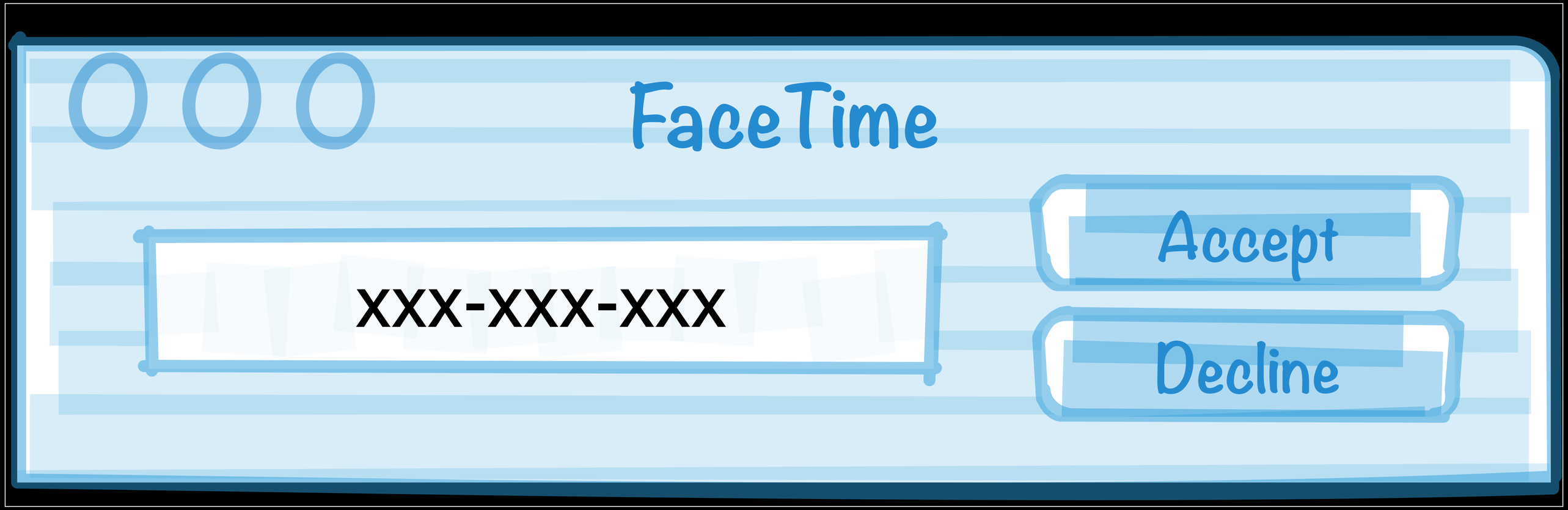
service:account:inviteReceivedForSession:fromID:withContext:]

```
<CSDMessagingCallMessage: 0x7f866e52d020> {
    inviteData =     {
        SKEBlob = <308202ca 02010102 01003082 02c00c09 696e6974 6961746f 720c0972 6573706f 6e646572 a08202a6 048202a2 3082029e 308
20207 a0030201 02020a03 35b9c2d6 a22b8429 a8300d06 092a8648 86f70d01 01050500 305a310b 30090603 55040613 02555331 13301106 0355040
a 130a4170 706c6520 496e632e 31153013 06035504 0b130c41 70706c65 20695068 6f6e6531 1f301d06 03550403 13164170 706c6520 6950686f 6e
652044 65766963 65204341 301e170d 31393032 31333037 33333539 5a170d32 30303231 33303733 3835395a 302f312d 302b0603 55040316 244336
43 43324231 432d3238 37312d34 4639392d 38383531 2d314332 31394431 36464645 3130819f 300d0609 2a864886 f70d0101 01050003 818d0030 8
1890281 8100c322 6588884c 250c8893 a9294335 0e79d876 2c6493c0 ba0db82c 1b326b55 f90682cb dddbda1d 102cb99a 3f65ef2c 81970c26 e7580
821 21e1e04a 5d5f31b3 8248984a ace32698 a9174cb2 8f97c1d3 a779fe1c 28f5fa0a 4aa1e080 bdcdc5f7 b6e6da26 d66898b4 ba83af9d ded76689
53eea4b8 4a559ba8 7a86b80b 887840fa fbfb0203 010001a3 81953081 92301f06 03551d23 04183016 8014b2fe 21234486 956a79d5 81268e73 10d8
a74c 8e74301d 0603551d 0e041604 14acfd6d 375c754c 696ee8a5 77153ca7 9b7b9c81 0c300c06 03551d13 0101ff04 02300030 0e060355 1d0f0101
 ff040403 0205a030 20060355 1d250101 ff041630 1406082b 06010505 07030106 082b0601 05050703 02301006 0a2a8648 86f76364 060a0404 020
50030 0d06092a 864886f7 0d010105 05000381 8100adbf 56fcd79d 4c6b0ef5 8e0d3298 886548e5 afba4798 85bdec3d 599d76ef 1c6512d3 f9c44b3
7 275084b3 d223d2c1 a82ff6a8 383d630f 57cbbc51 58f7749f c590bb90 b8df91c6 e3477b7d 79eb5cf2 1a85b41a 679e54a3 165cd5c8 c5515fcf 34
2ecd03 24a163fa a7549cc0 20770690 2b3dd756 3411e01e 717bbc4c a8d01bbb 4142>;
        callInfoBlob = <08809eef 9c031001 1a0e4d61 63426f6f 6b50726f 31312c35 22083134 37302e34 2e322a06 31384631 3332>;
        mediaBlob = <78dae360 146094e2 e7787e75 6ab804a3 c27f198d 4f22068c 4a7b9838 1eb67d58 c72dc020 359d91a3 4e888b03 a84c62ff 3
c1e0506 309b4962 ff4c109b 87830928 dedac0a8 d0d0c008 e631c178 52b16e3e c1d66141 8e5606d6 be102a08 48195a19 82692320 ed186c65 64ed1
96c a56b68ed e3062643 82ac9d1d 9d1c9dad 9d83ac8c ad03803c 0f432b13 6b37472b 536bc720 2b0b7d53 1d237d63 eb88206b 054690cb aa07a5cb
945a19e1 c1570d75 2393c4f9 767eb01b 412e39da c4aac040 737718f1 866526a7 16e5572a 18ea99eb 19383078 b172b008 5ce00752 02022d0a 5e2c
4077 b90339af e4051892 98385832 180a1801 0c15781e>;
    };
    protoProtocolVersion = 3;
    protoWantsVideo = 1;
    type = Invite;
}
```



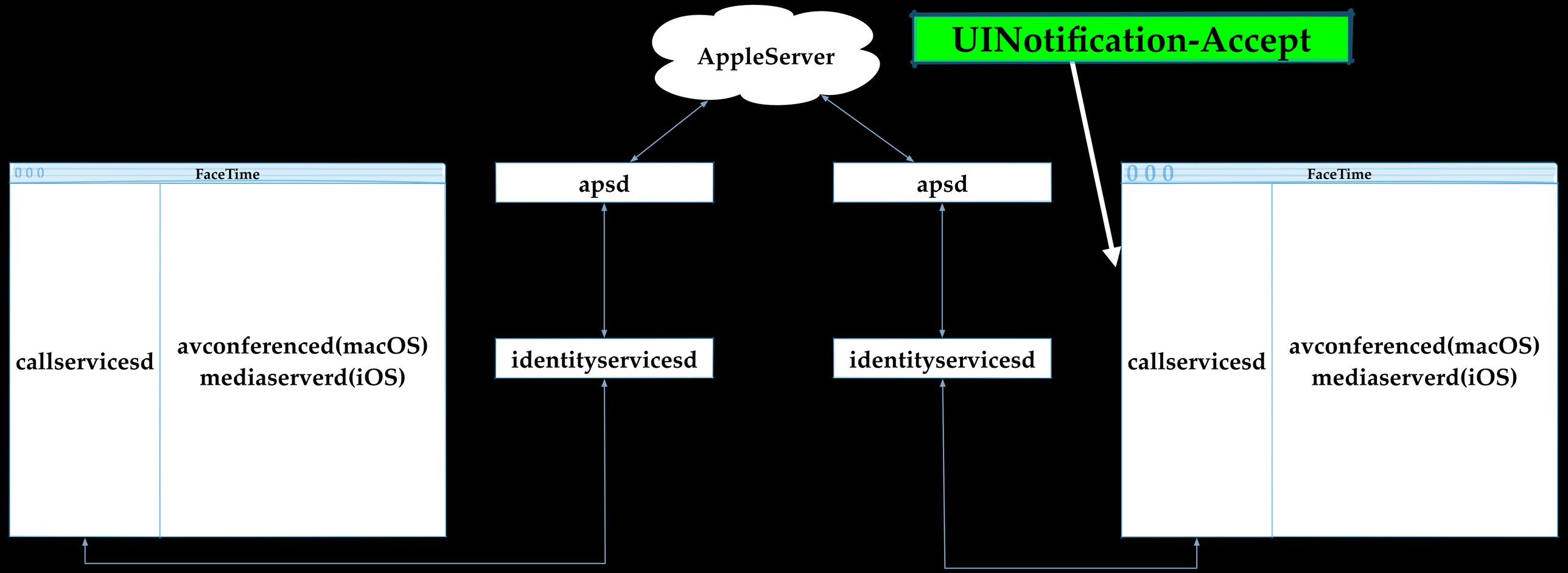


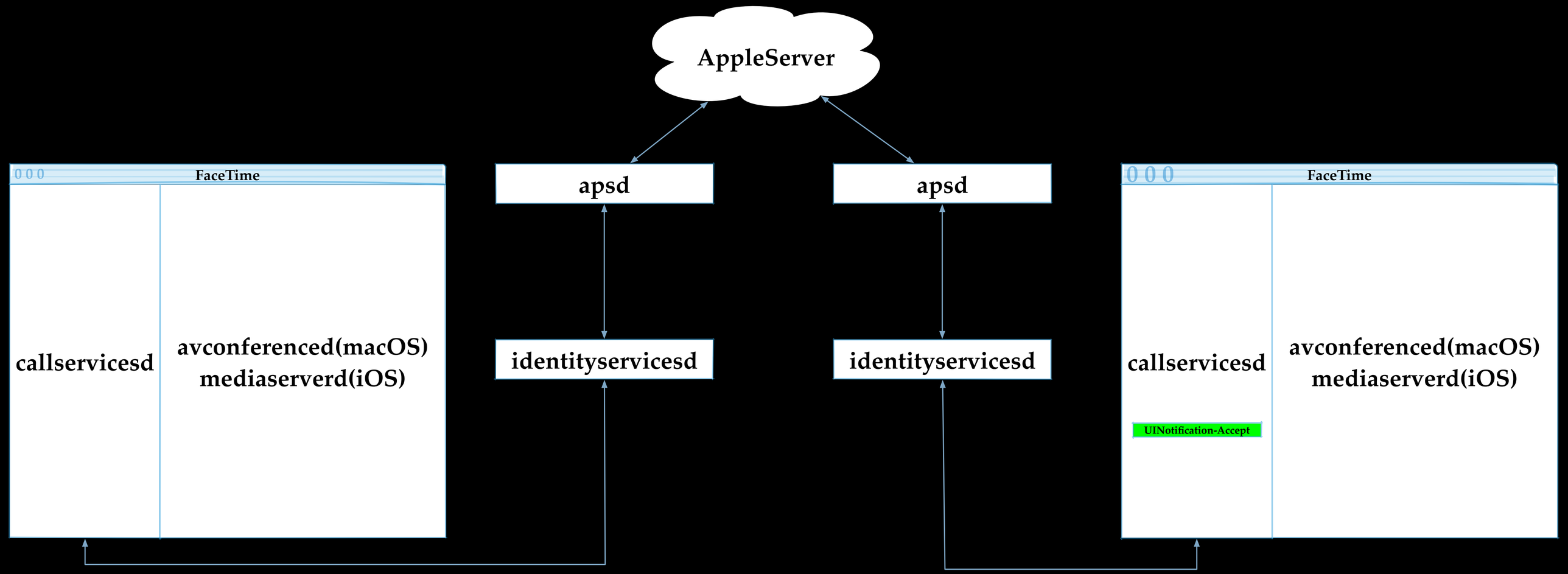
- callservicesd continues to forward blobs to avconferenced / mediaserverd

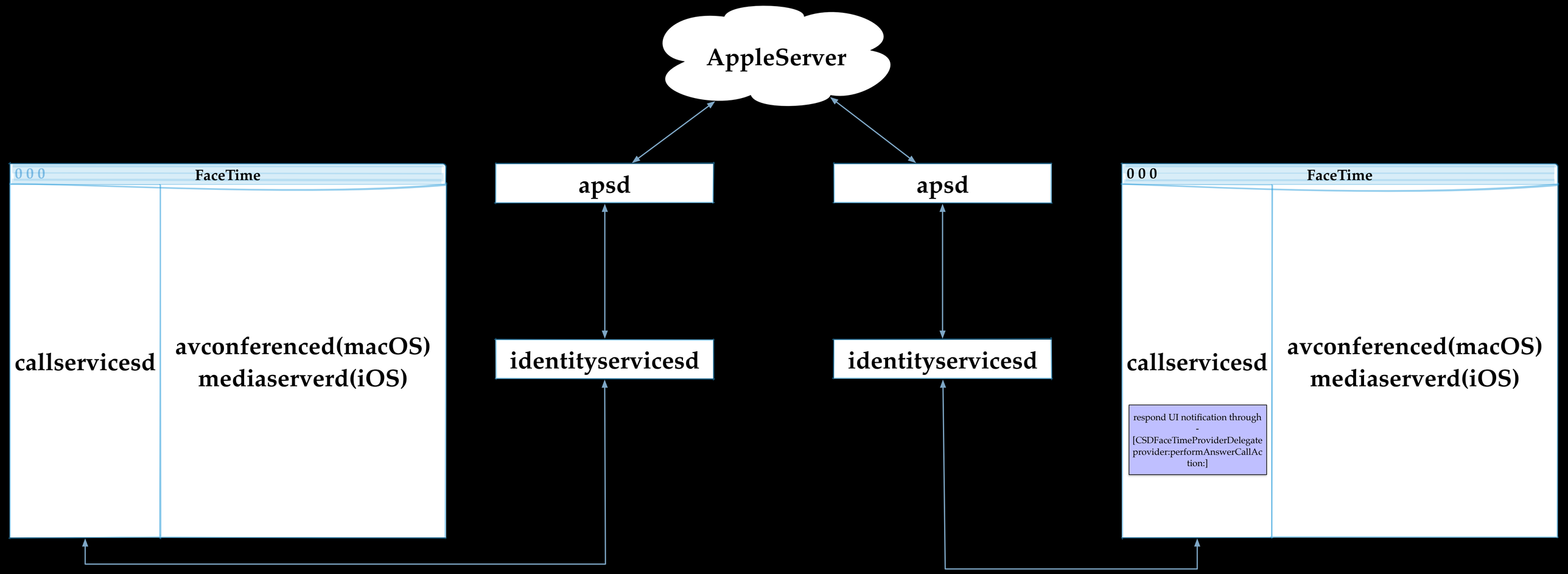

FaceTime
xxx-xxx-xxx
Accept
Decline

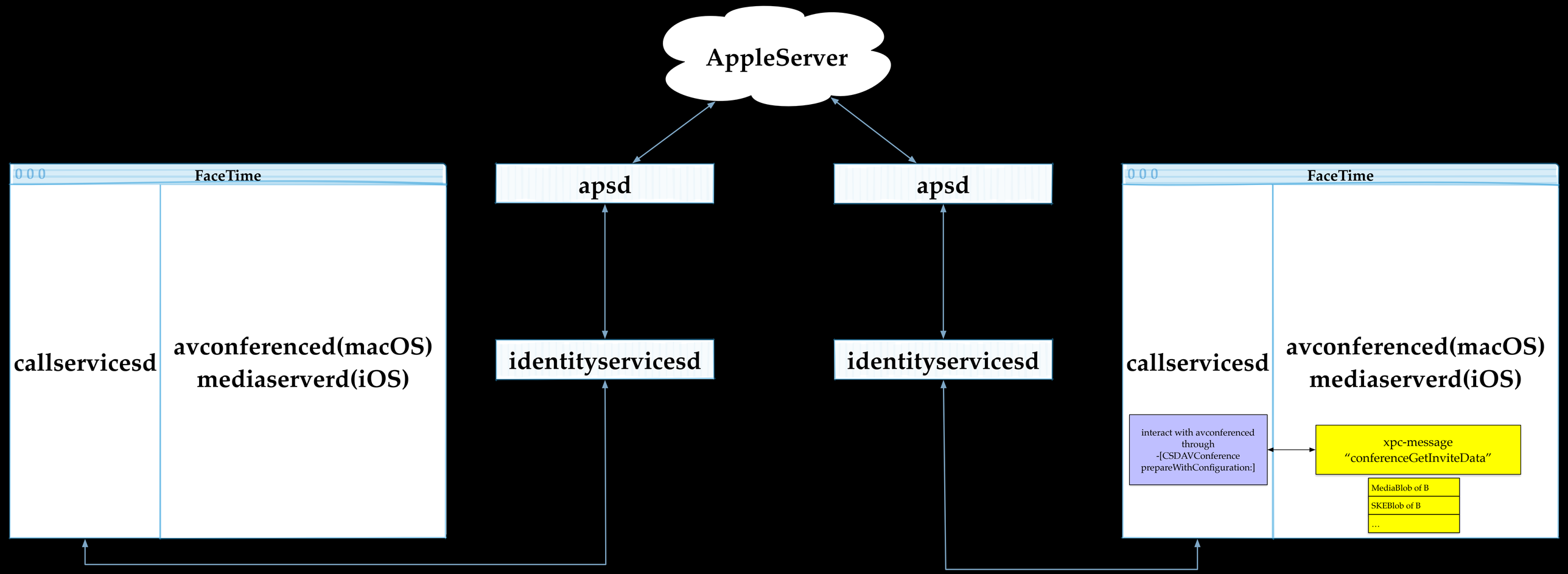

FaceTime
xxx-xxx-xxx
Accept
Decline

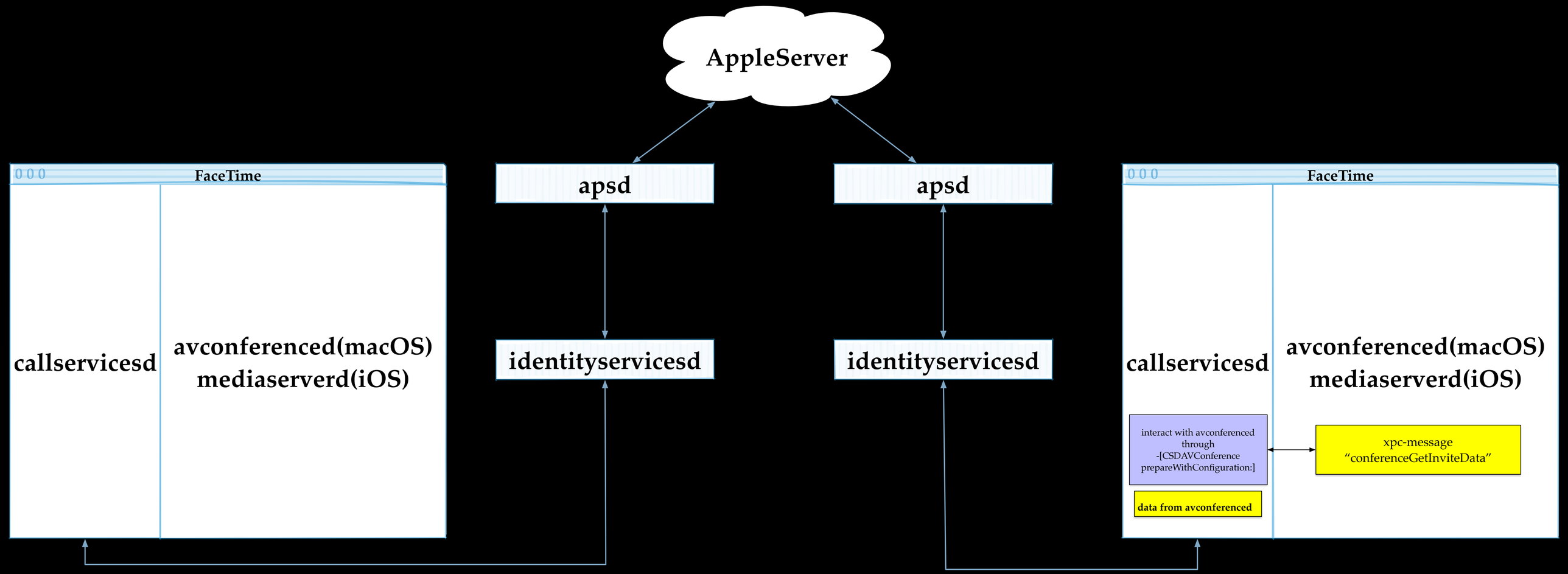

UINotification-Accept
FaceTime
xxx-xxx-xxx
Accept
Decline

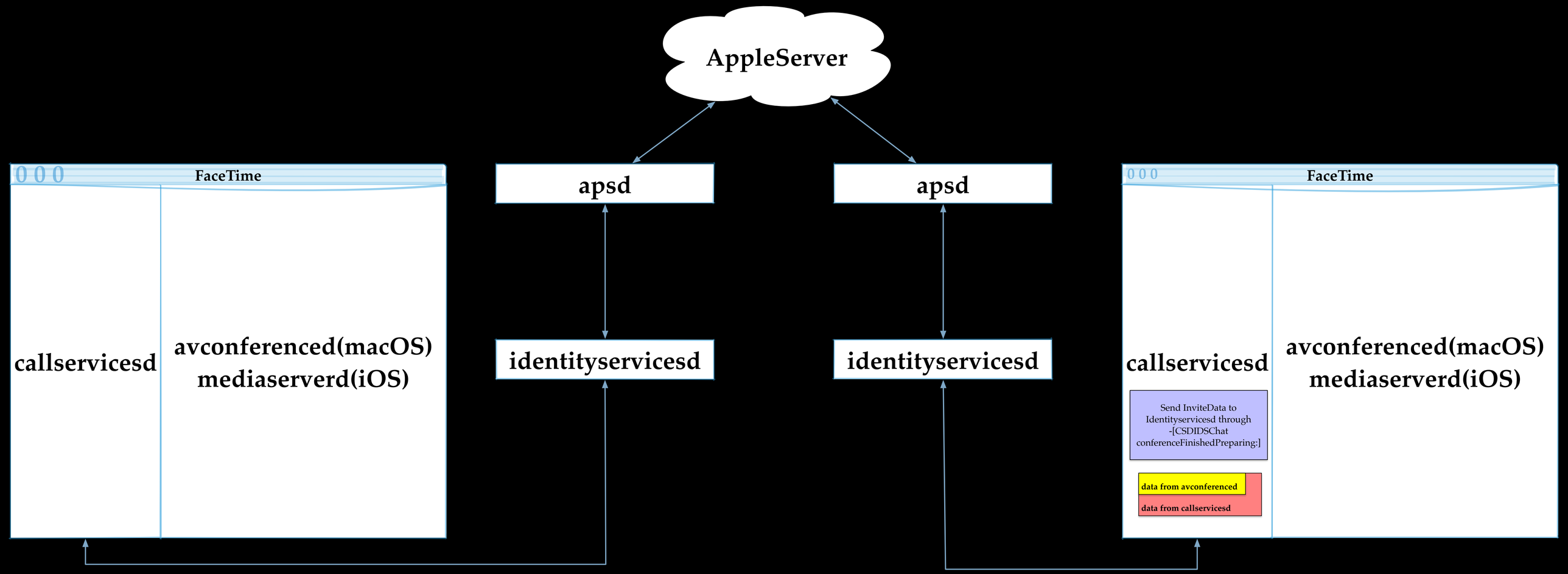

AppleServer
UINotification-Accept
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
apsd
apsd
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

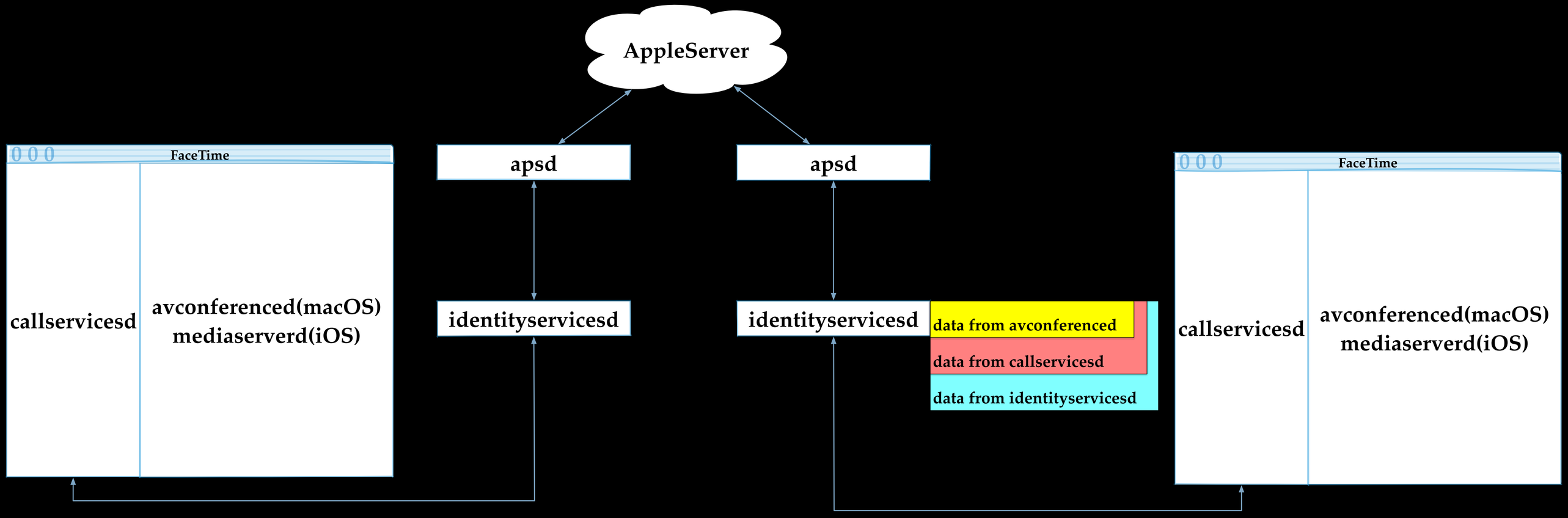

AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
FaceTime
callservicesd
UINotification-Accept
avconferenced(macOS)
mediaserverd(iOS)

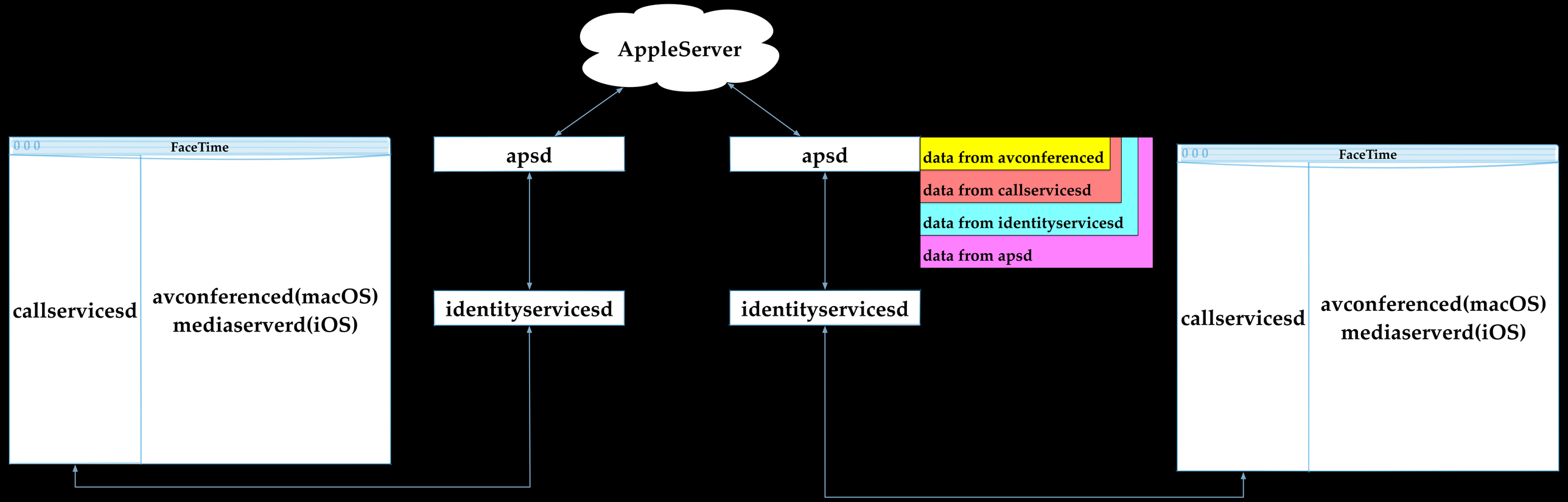

AppleServer
apsd
apsd
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
respond UI notification through
-
[CSDFaceTimeProviderDelegate
provider:performAnswerCallAc
tion:]

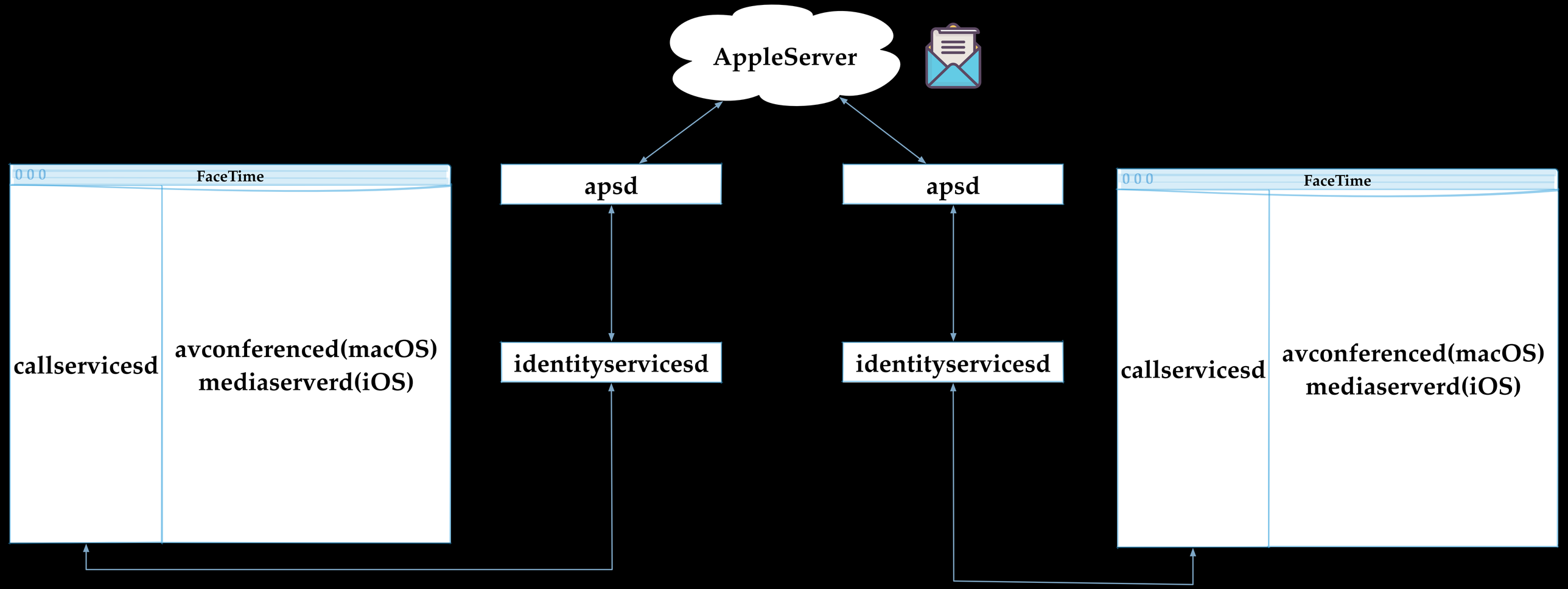

AppleServer
apsd
apsd
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
interact with avconferenced through -[CSDAVConference prepareWithConfiguration:]
xpc-message “conferenceGetInviteData”
MediaBlob of B
SKEBlob of B
…

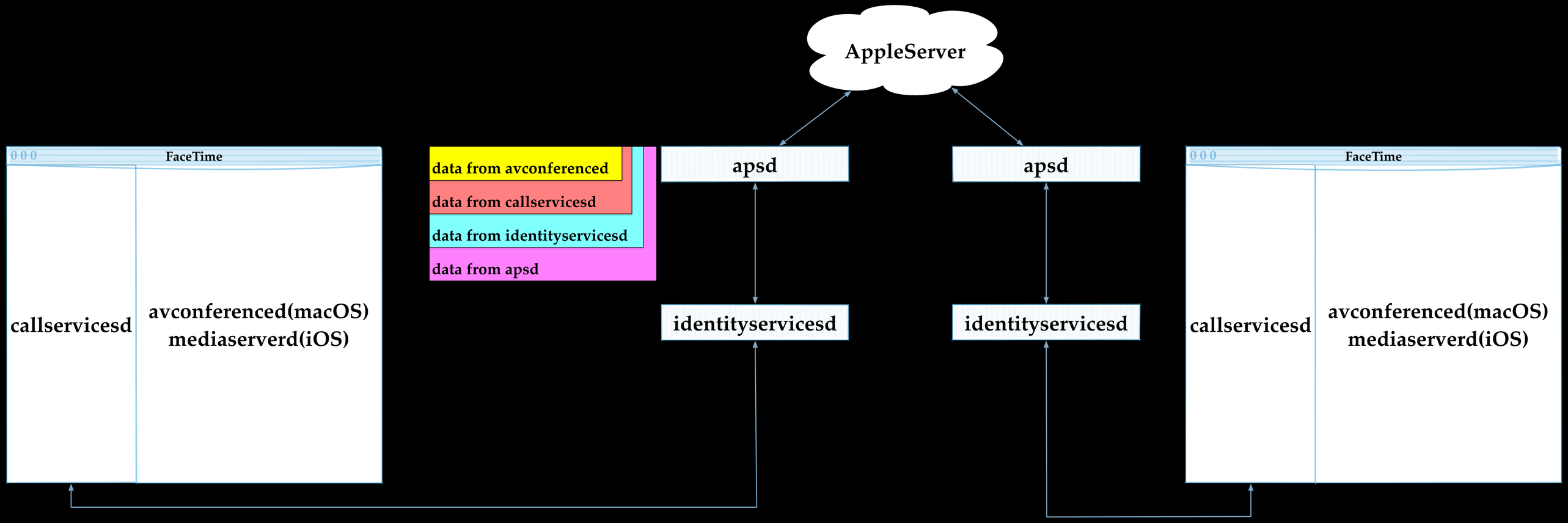

AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
interact with avconferenced through -[CSDAVConference prepareWithConfiguration:]
xpc-message "conferenceGetInviteData"
data from avconferenced


AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
Send InviteData to Identityservicesd through -[CSDIDSChat conferenceFinishedPreparing:]
data from avconferenced
data from callservicesd

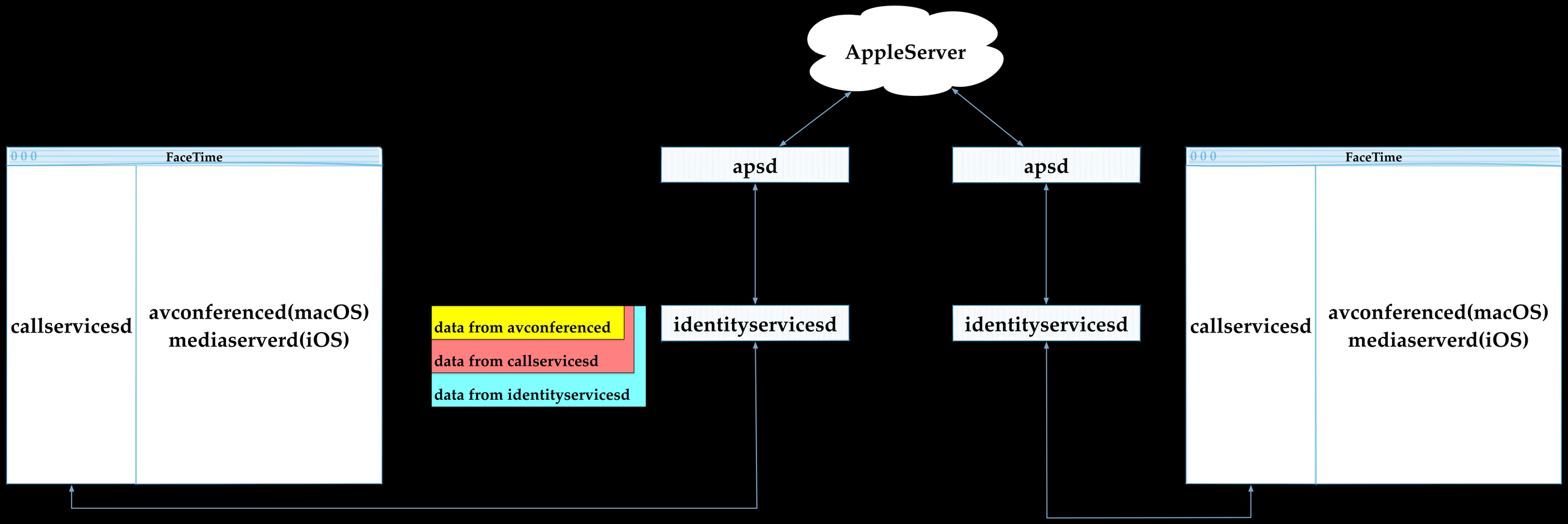

AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
data from avconferenced
data from callservicesd
data from identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

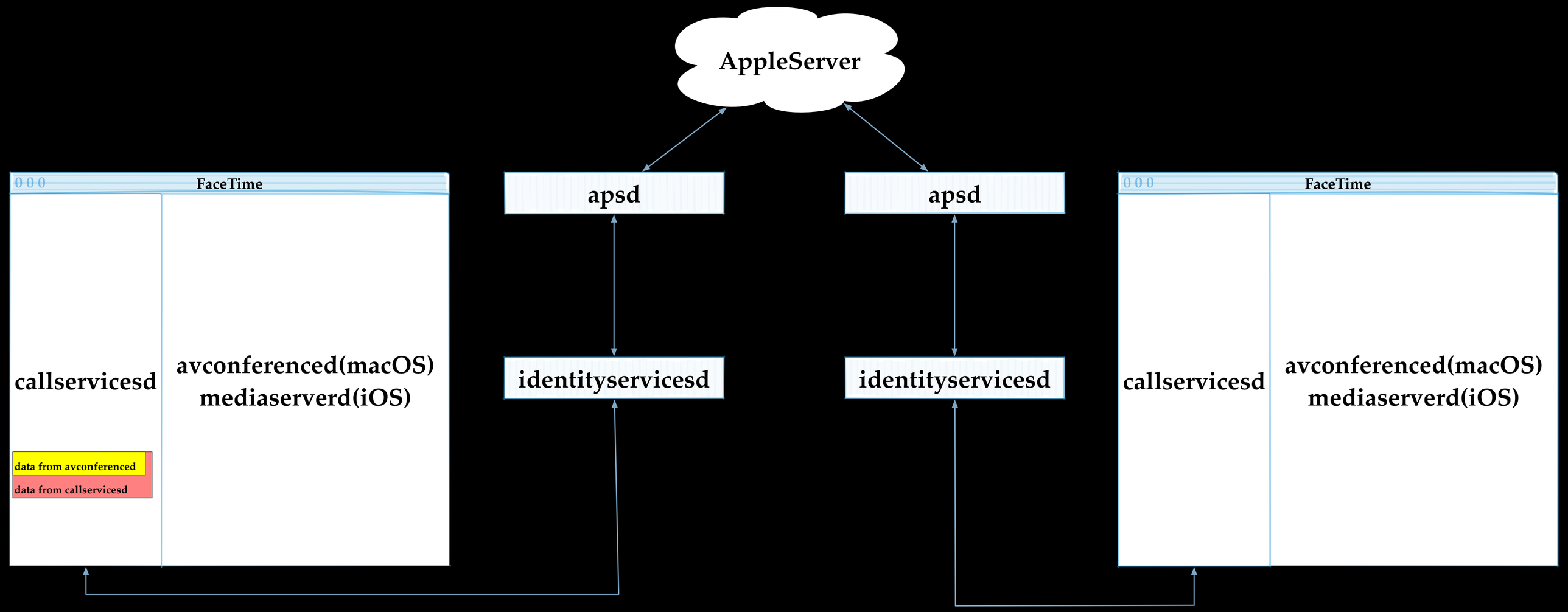

AppleServer
apsd
apsd
data from avconferenced
data from callservicesd
data from identityservicesd
data from apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

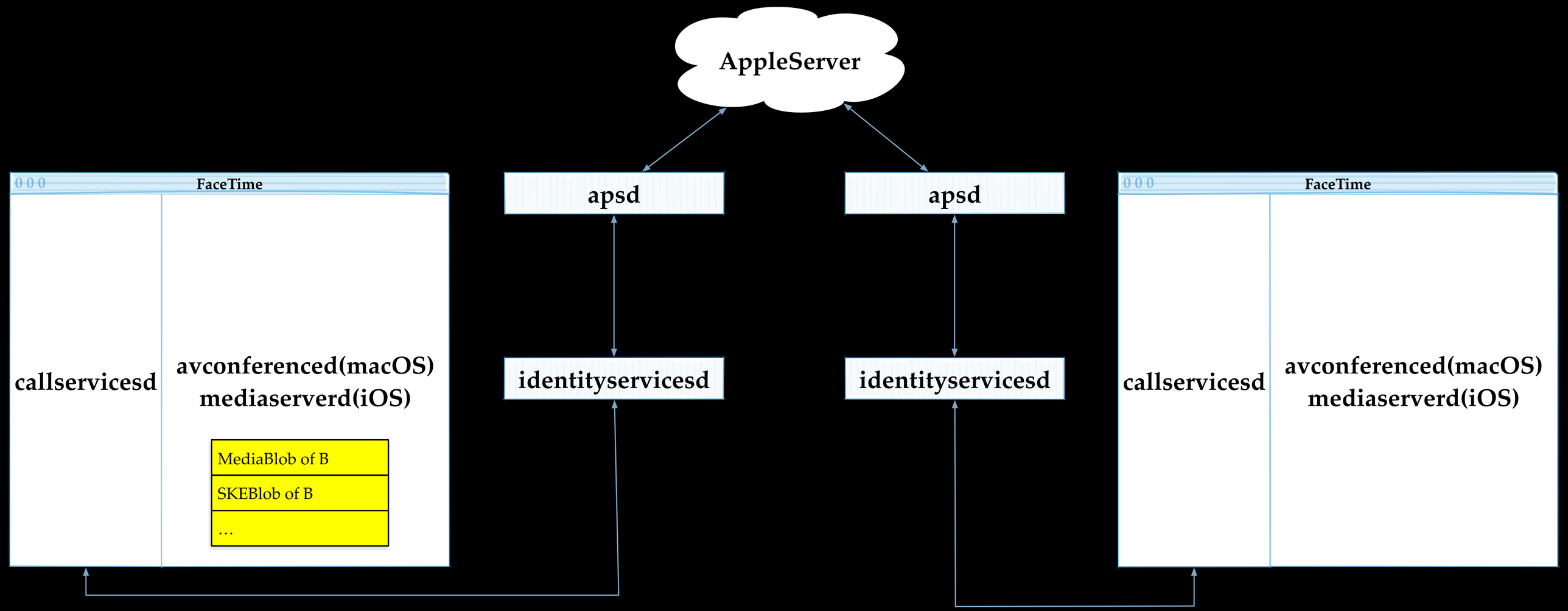

AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

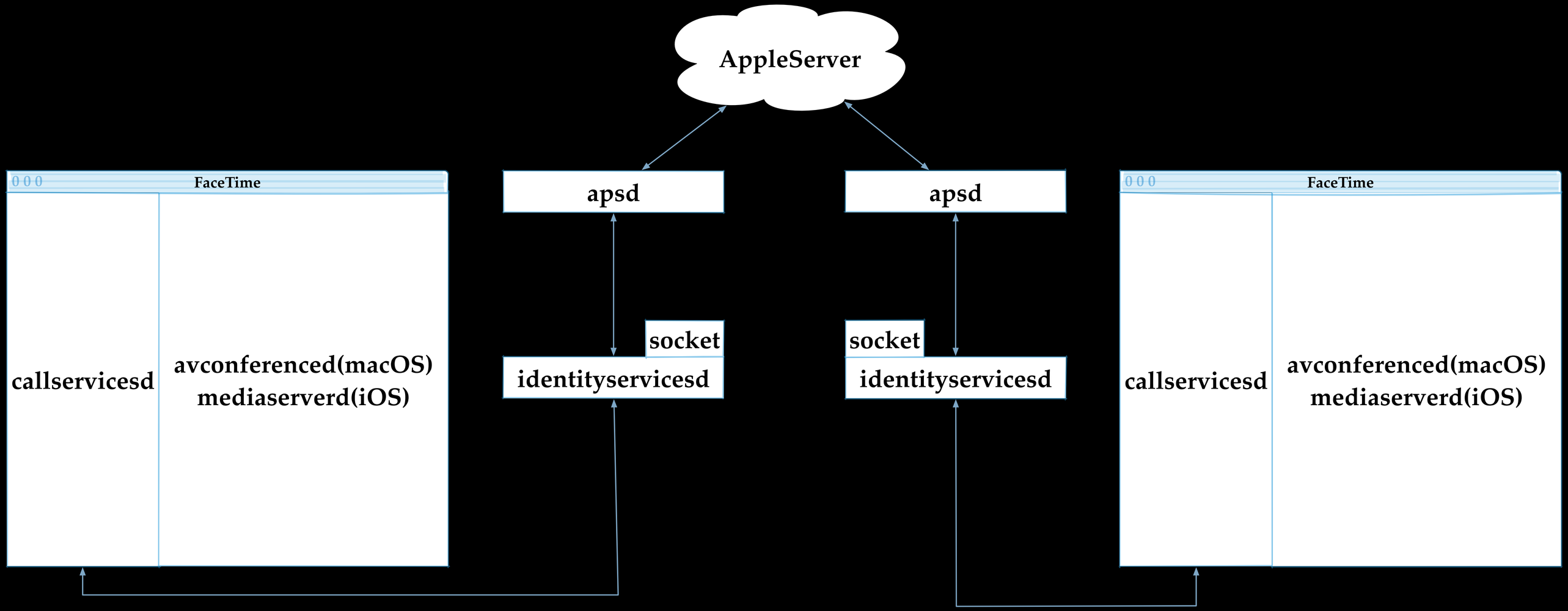

AppleServer
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
data from avconferenced
data from callservicesd
data from identityservicesd
data from apsd
apsd
apsd
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

- c=233 -> processIncomingSessionAcceptMessage

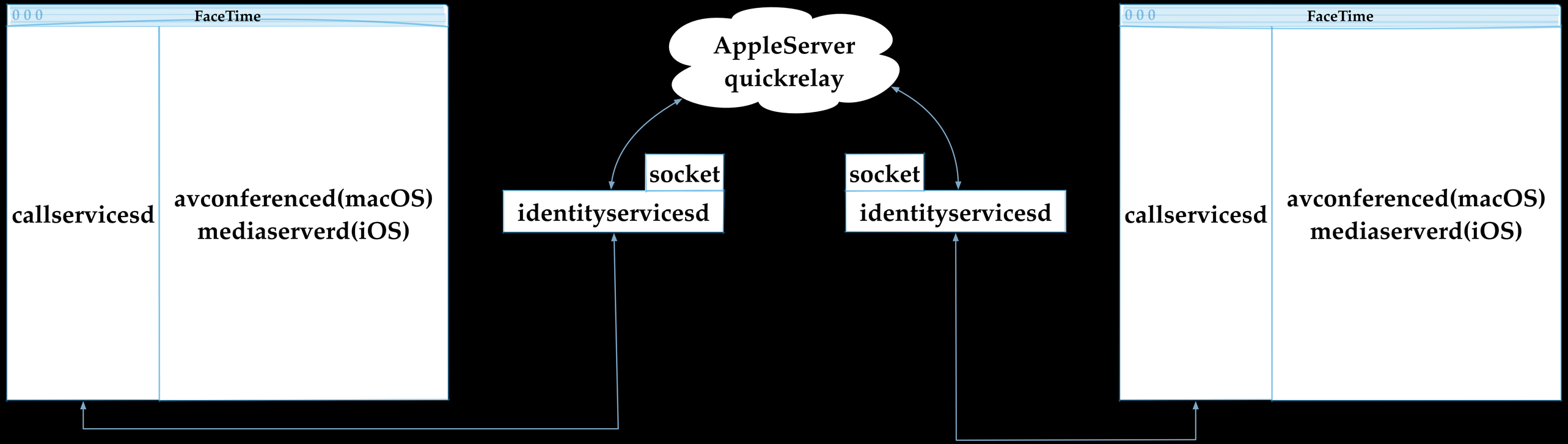



AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
data from avconferenced
data from callservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

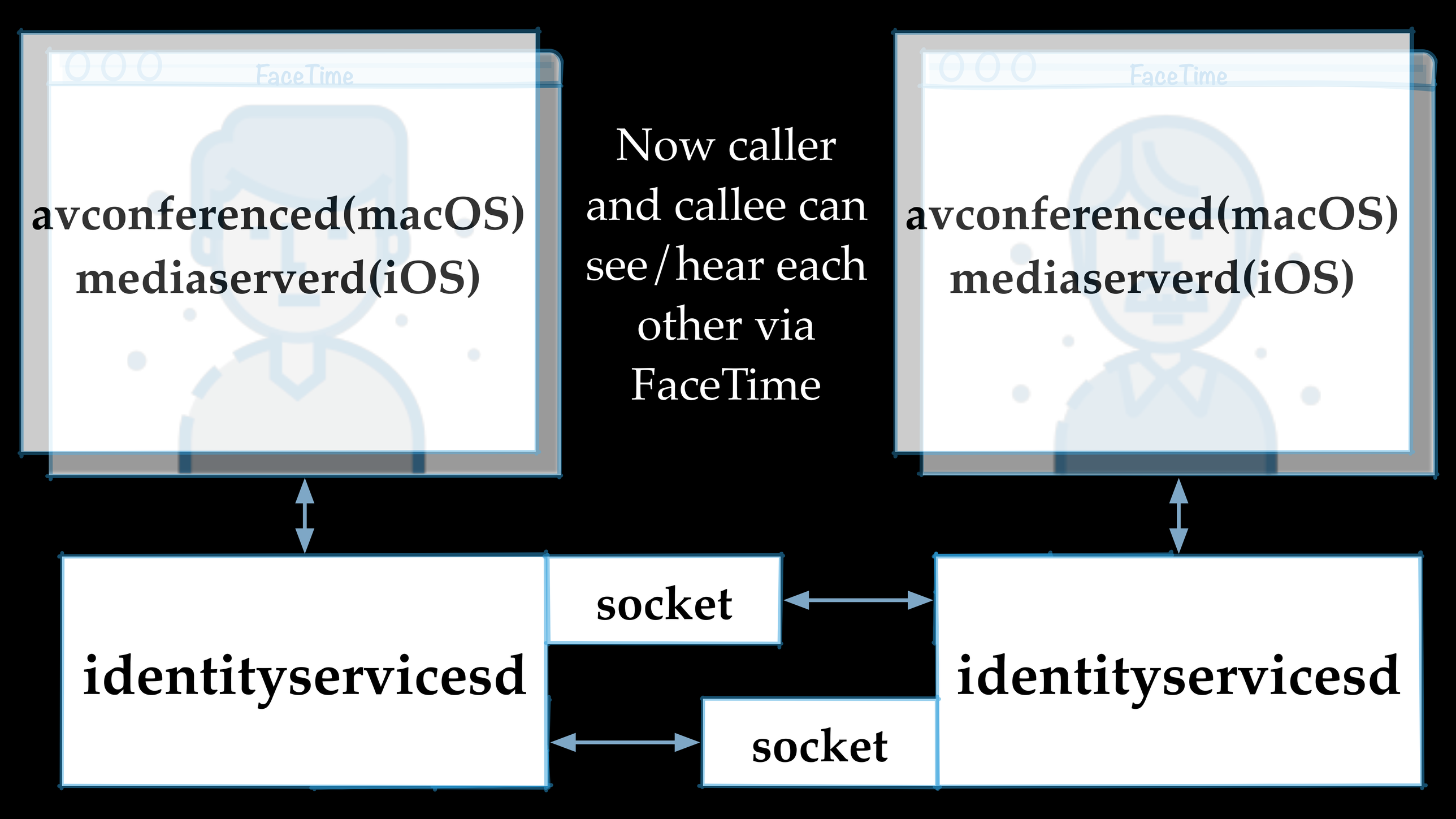

AppleServer
apsd
apsd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
MediaBlob of B
SKEBlob of B
…
identityservicesd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)



Meanwhile, identityservicesd creates sockets for session based connections



Apple’s QuickRelay server coordinates caller and callee build a direct connection

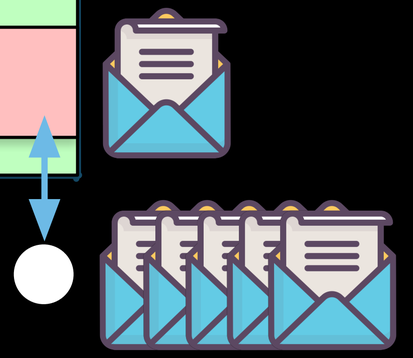

FaceTime
avconferenced(macOS)
mediaserverd(iOS)
Now caller and callee can see/hear each other via FaceTime
FaceTime
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
socket
socket
identityservicesd

identityservicesd is responsible
for first layer packet handling
and packet re-dispatching

identityservicesd

-[IDSUDPLink _processIncomingPacket]

recvmsg

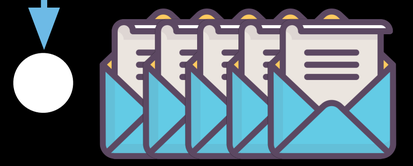


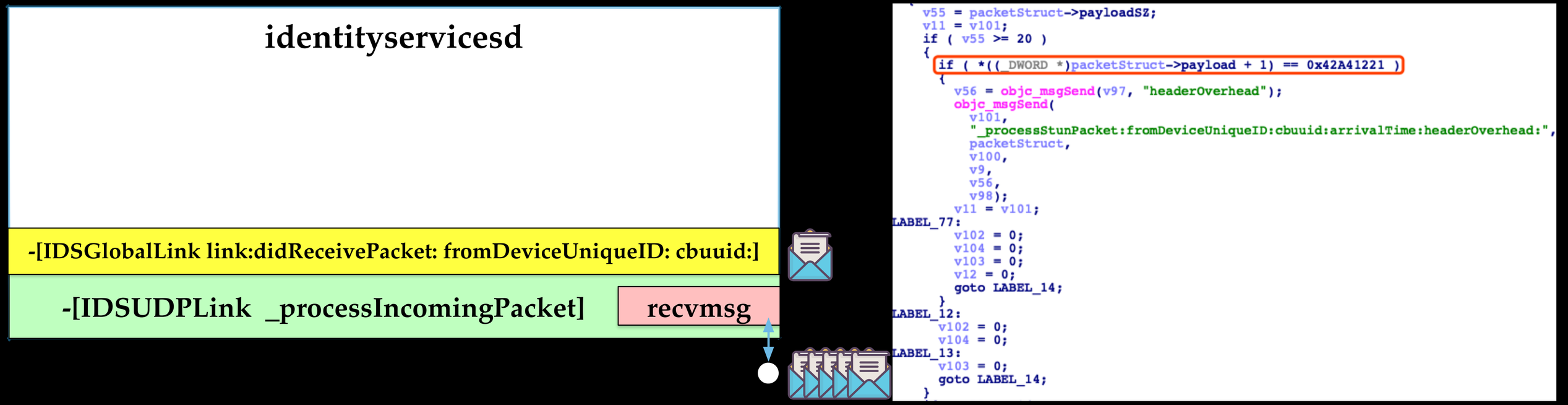
identityservicesd

-[IDSGlobalLink link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

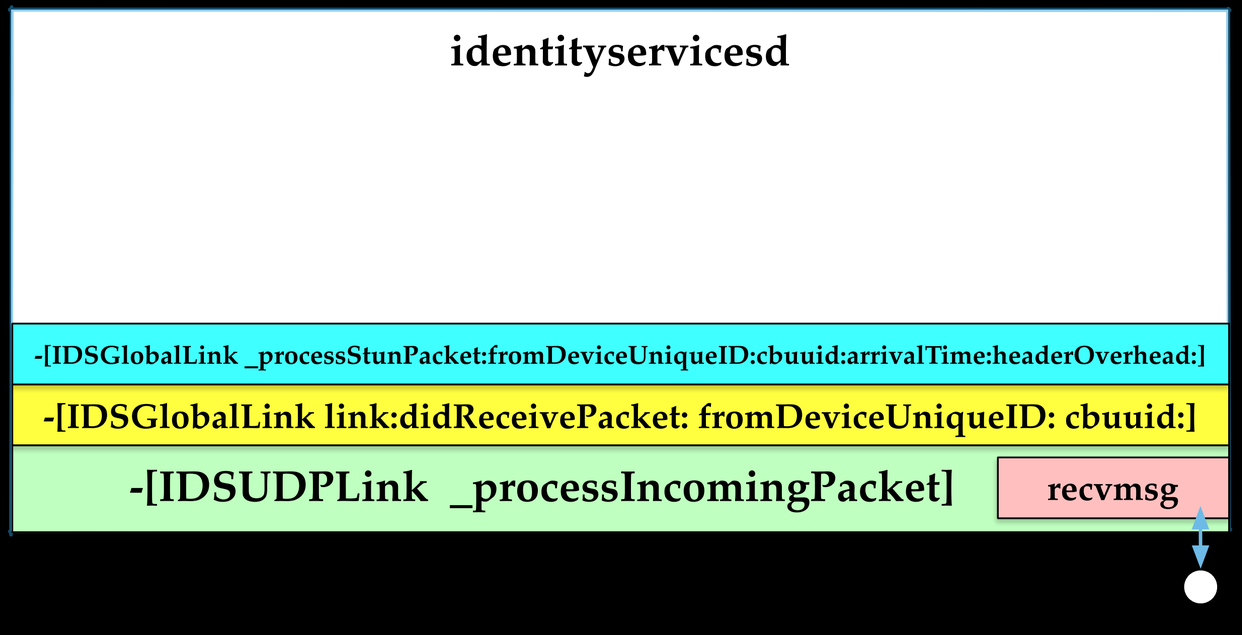
-[IDSUDPLink _processIncomingPacket]

recvmsg





In this function, identityservicesd identifies STUN messages according to magic number matching



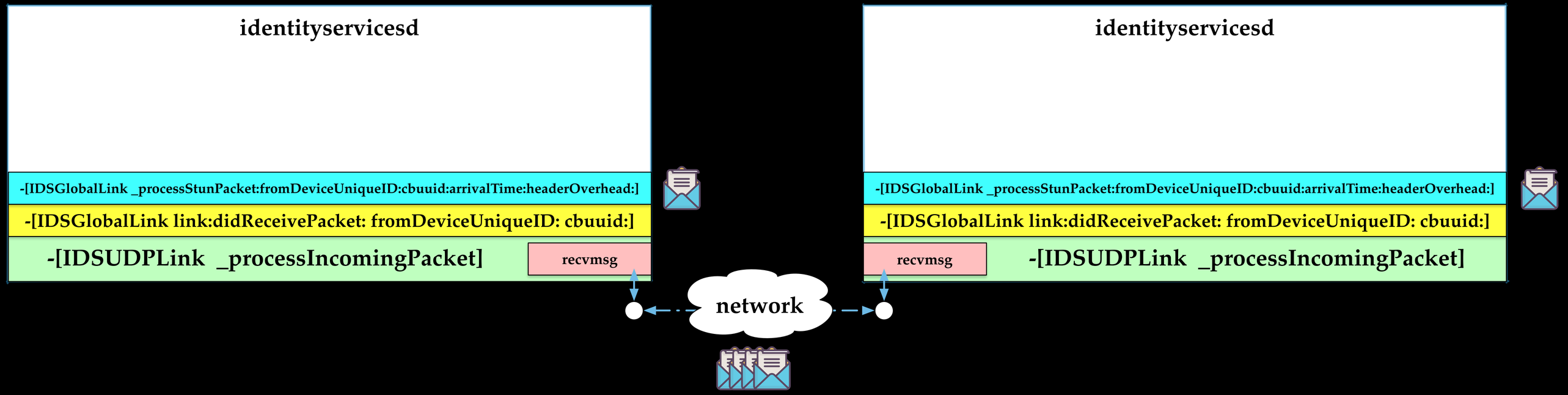

```
if ( *((_DWORD *)packetStruct->payload + 1) == 0x42A41221 )
{
  v56 = objc_msgSend(v97, "headerOverhead");
  objc_msgSend(
    v101,
    "_processStunPacket:fromDeviceUniqueID:cbuuid:arrivalTime:headerOverhead:",
    packetStruct,
    v100,
    v9,
    v56,
    v98);
  v11 = v101;
LABEL_77:
  v102 = 0;
  v104 = 0;
  v103 = 0;
  v12 = 0;
  goto LABEL_14;
}
LABEL_12:
  v102 = 0;
  v104 = 0;
LABEL_13:
  v103 = 0;
  goto LABEL_14;
}
```

identityservicesd passes STUN messages to a handler


```
switch ( (unsigned __int16)((_WORD)stunMessageType - 0xFE0) )
{
  case 0u:
  case 1u:
    objc_msgSend(
      v196,
      "_processAllocbindResponse:fromDevice:localIfIndex:localAddress:remmoteAddress:candidatePairToken:arrivalTime:",
      v11,
      v9,
      v8->var11,
      &v8->var12,
      *(double *)&v198,
      &v8->var13,
      v195);
    goto LABEL_152;
  case 2u:
  case 0x12u:
    objc_msgSend(
      v196,
      "_processUnallocbindResponse:fromDevice:localIfIndex:localAddress:remmoteAddress:candidatePairToken:arrivalTime:",
      v11,
      v9,
      v8->var11,
      &v8->var12,
      *(double *)&v198,
      &v8->var13,
      v195);
    goto LABEL_152;
  case 3u:
    objc_msgSend(v200, "processStatsResponse:arrivalTime:", v11, *(double *)&v198);
    goto LABEL_171;
  case 4u:
    objc_msgSend(v200, "processInfoResponse:packetBuffer:headerOverhead:", v11, v8, v192);
    goto LABEL_171;
  case 5u:
    objc_msgSend(v200, "processSessionInfoResponse:packetBuffer:headerOverhead:", v11, v8, v192);
    goto LABEL_171;
```

-[IDSGlobalLink _processStunPacket:fromDeviceUniqueID:cbuuid:arrivalTime:headerOverhead:] will further call different handlers according to different STUN message types.

- Besides STUN messages, many other types of packets are also handled by identityservicesd itself
- The rest of packets will be distributed to different processes such as avconferenced

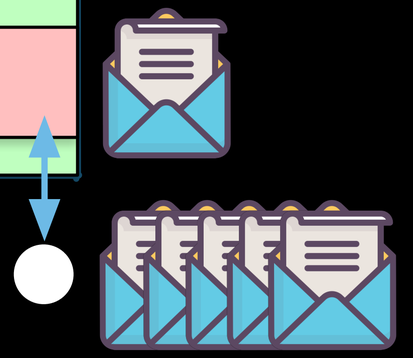


RTP packets handler

identityservicesd

-[IDSUDPLink _processIncomingPacket]

recvmsg

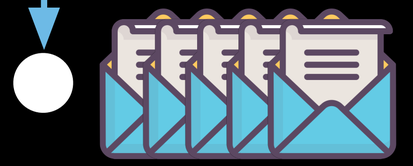


identityservicesd

-[IDSGlobalLink link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

-[IDSUDPLink _processIncomingPacket]

recvmsg



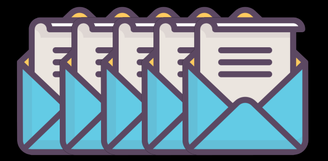
identityservicesd

-[IDSLinkManager link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

-[IDSGlobalLink link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

-[IDSUDPLink _processIncomingPacket]

recvmsg



identityservicesd

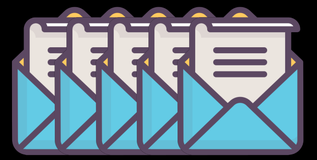
-[IDSDSession link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

-[IDSLinkManager link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

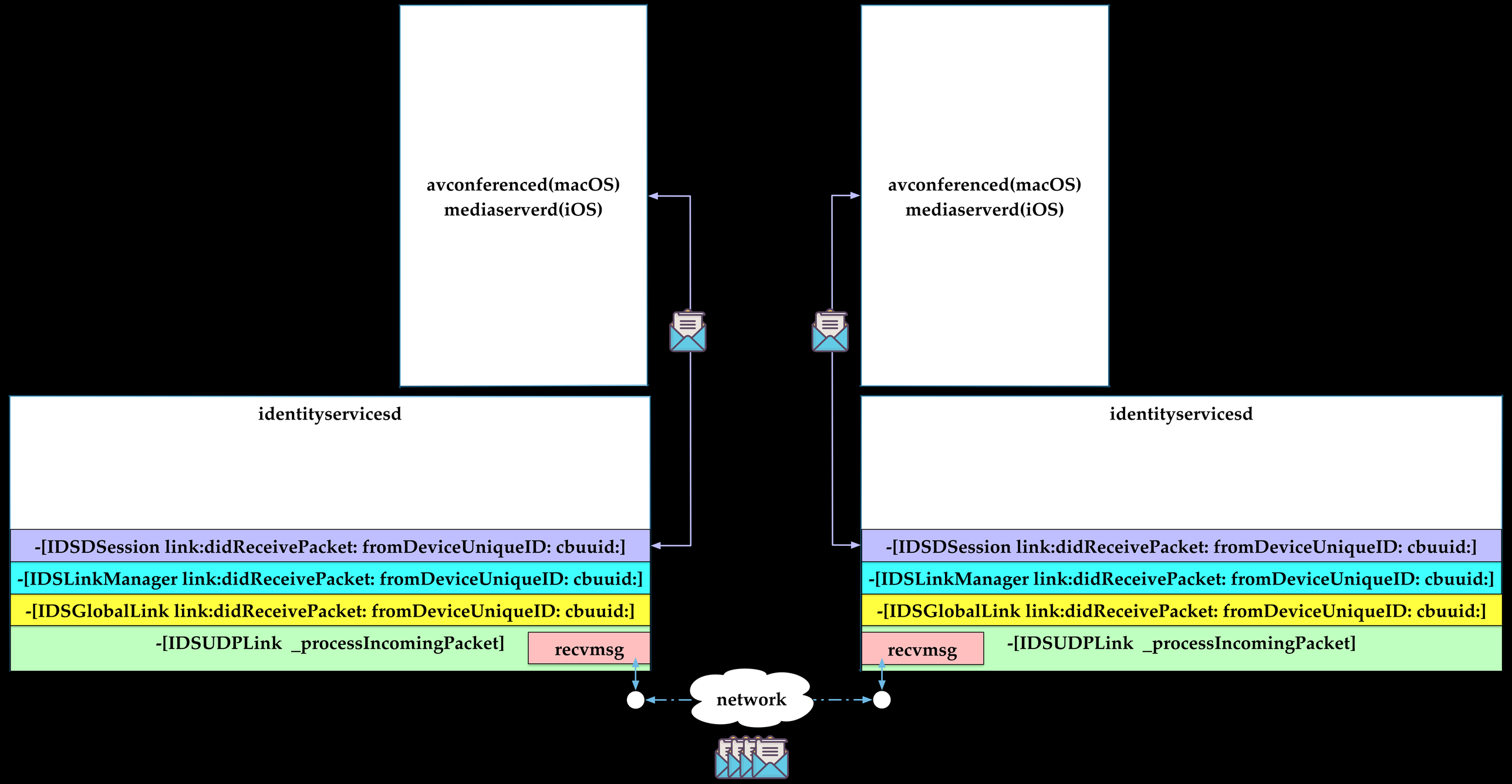
-[IDSGlobalLink link:didReceivePacket: fromDeviceUniqueID: cbuuid:]

-[IDSUDPLink _processIncomingPacket]

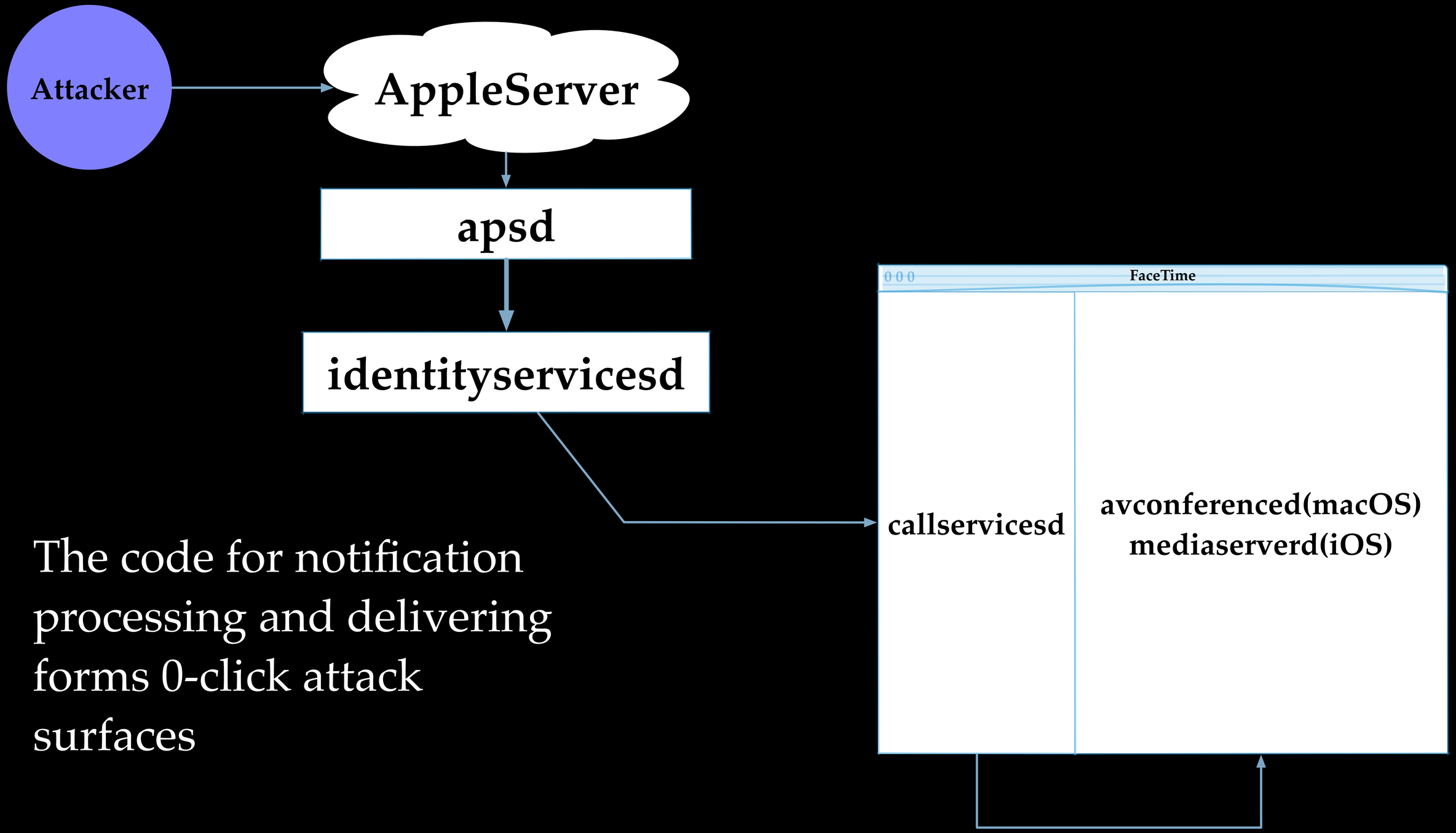
recvmsg



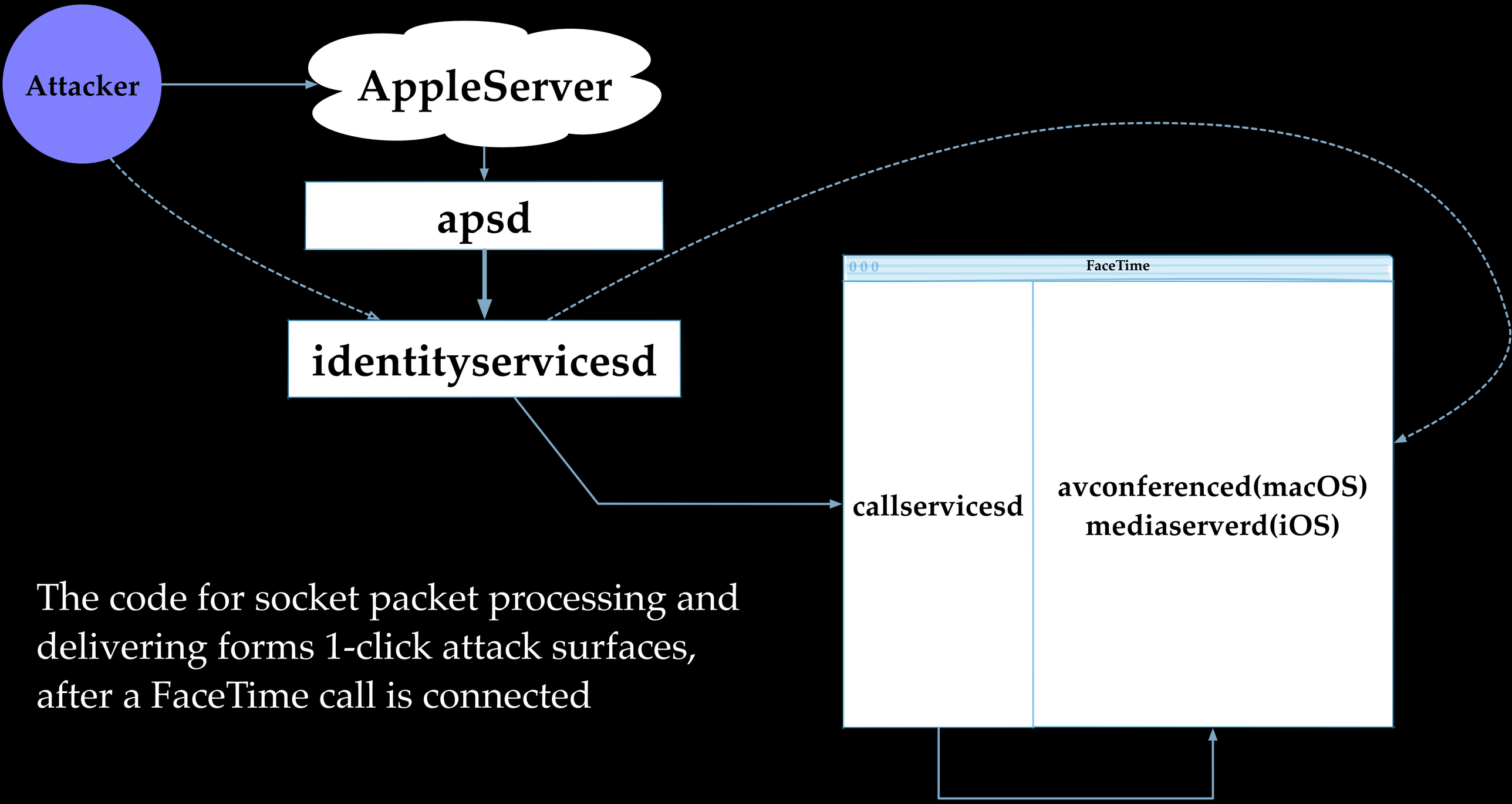
Transfer data to avconferenced through a series of undocumented syscalls (os_channel_* os_nexus_*)


avconferenced(macOS)
mediaserverd(iOS)
avconferenced(macOS)
mediaserverd(iOS)
identityservicesd
identityservicesd
-[IDSDSession link:didReceivePacket: fromDeviceUniqueID: cbuuid:]
-[IDSLinkManager link:didReceivePacket: fromDeviceUniqueID: cbuuid:]
-[IDSGlobalLink link:didReceivePacket: fromDeviceUniqueID: cbuuid:]
-[IDSUDPLink _processIncomingPacket]
recvmsg
-[IDSDSession link:didReceivePacket: fromDeviceUniqueID: cbuuid:]
-[IDSLinkManager link:didReceivePacket: fromDeviceUniqueID: cbuuid:]
-[IDSGlobalLink link:didReceivePacket: fromDeviceUniqueID: cbuuid:]
recvmsg
-[IDSUDPLink _processIncomingPacket]
network

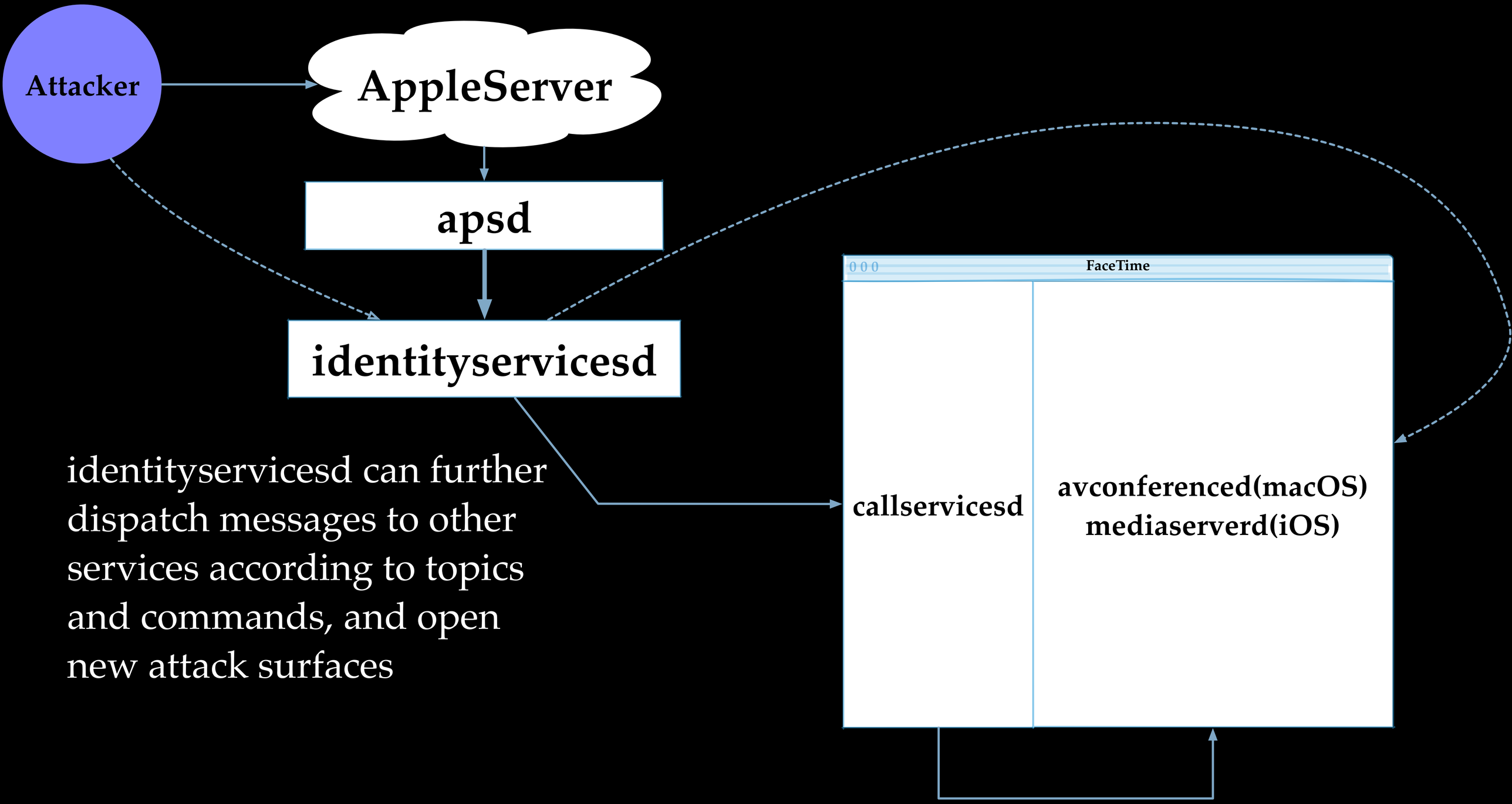
Attack Surfaces



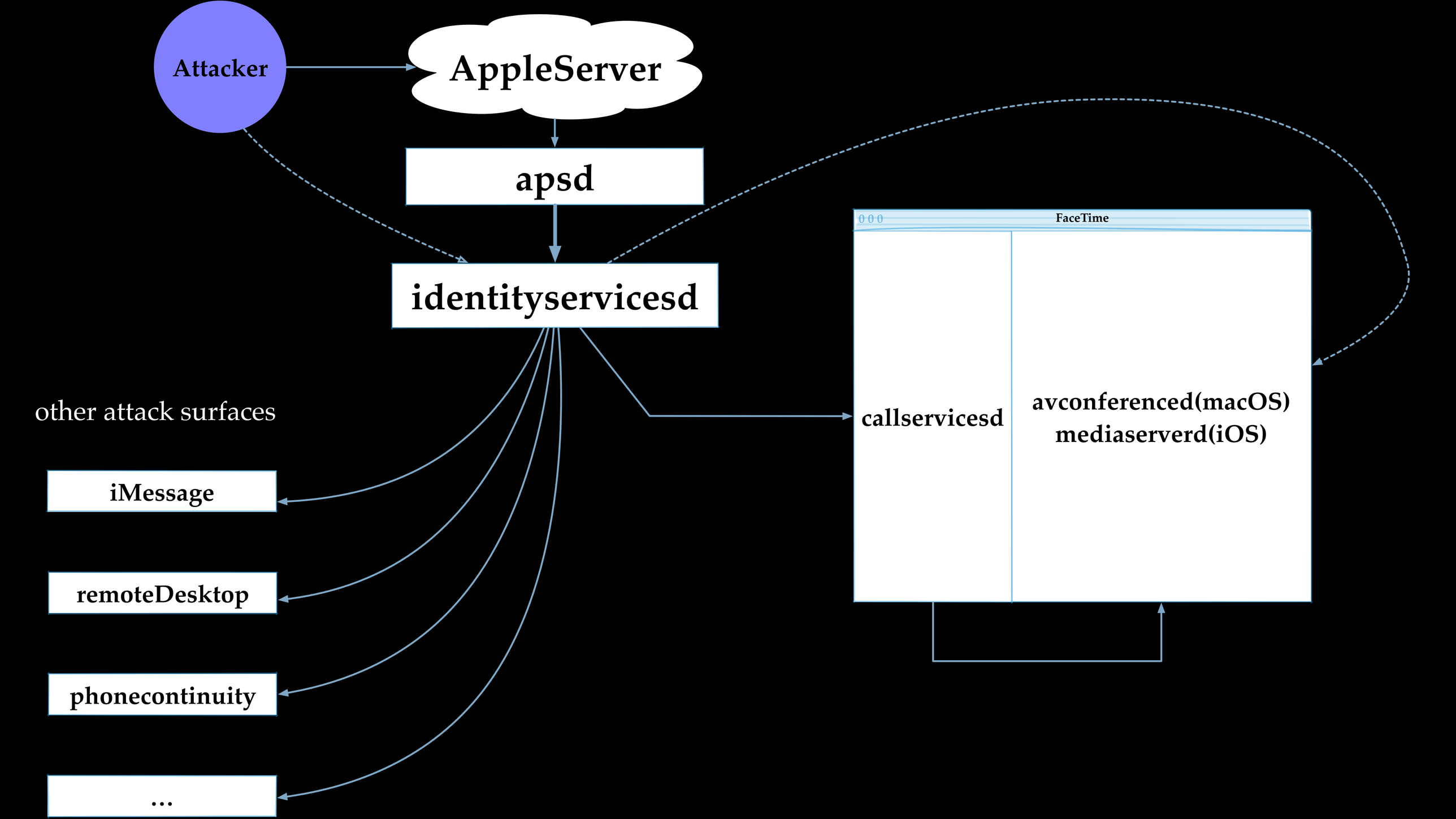
The code for notification processing and delivering forms 0-click attack surfaces



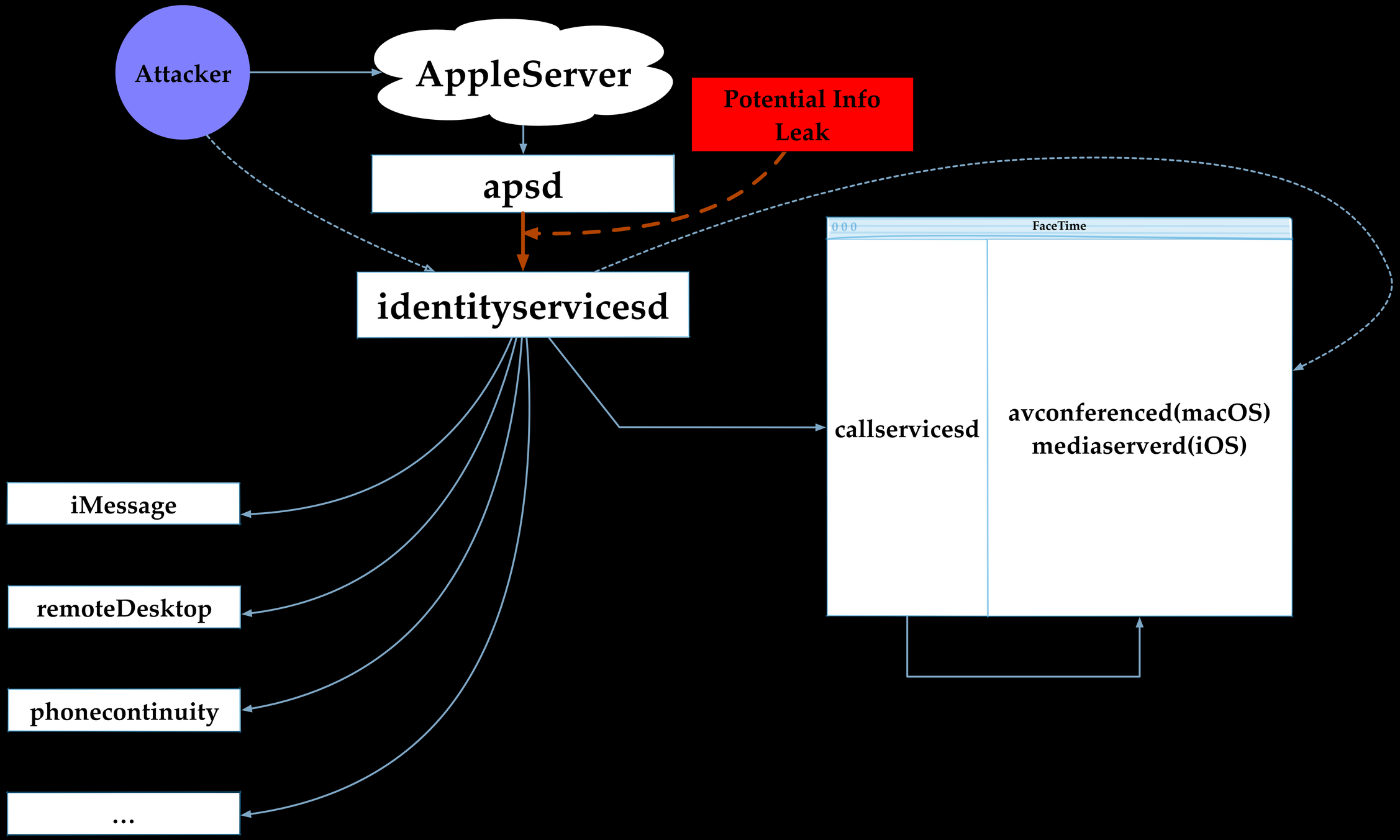
The code for socket packet processing and delivering forms 1-click attack surfaces, after a FaceTime call is connected

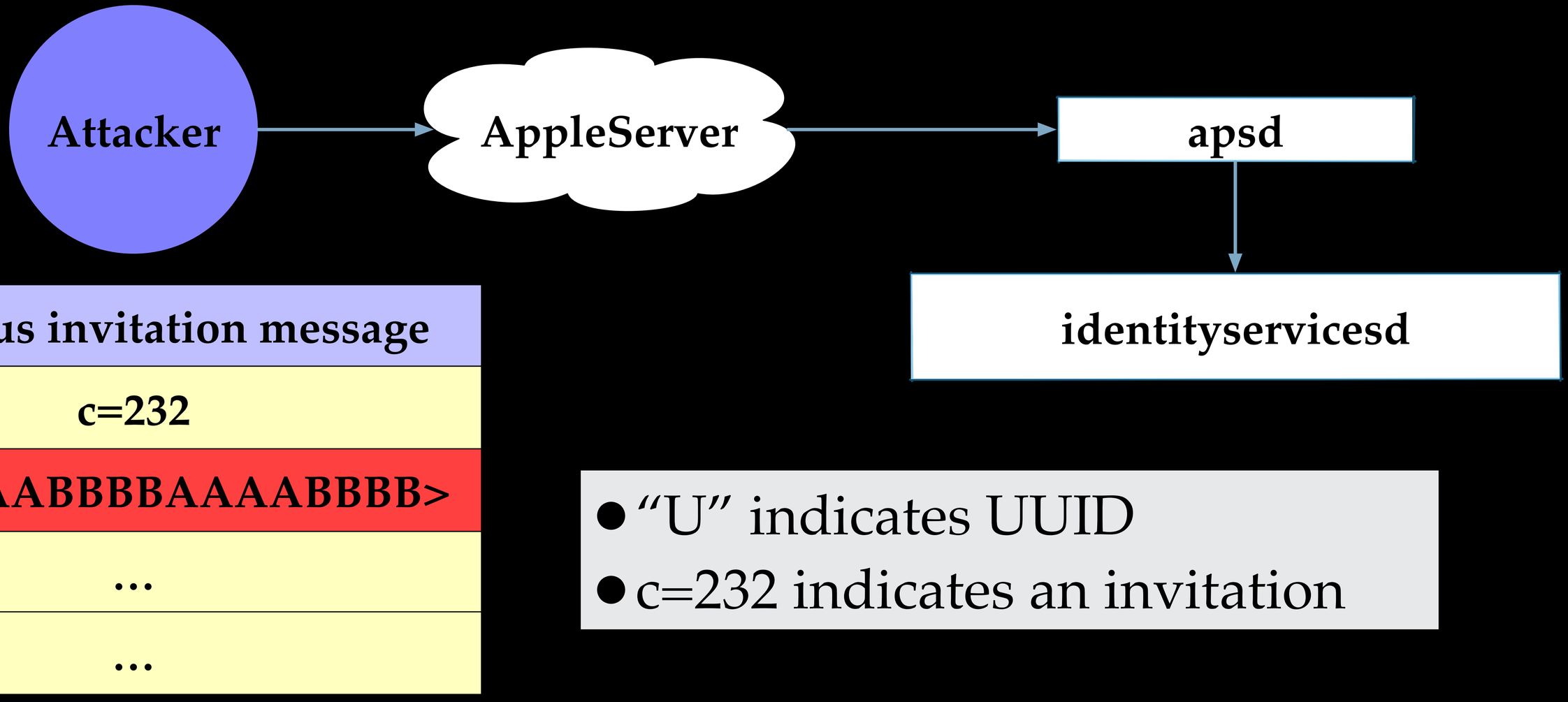


identityservicesd can further dispatch messages to other services according to topics and commands, and open new attack surfaces

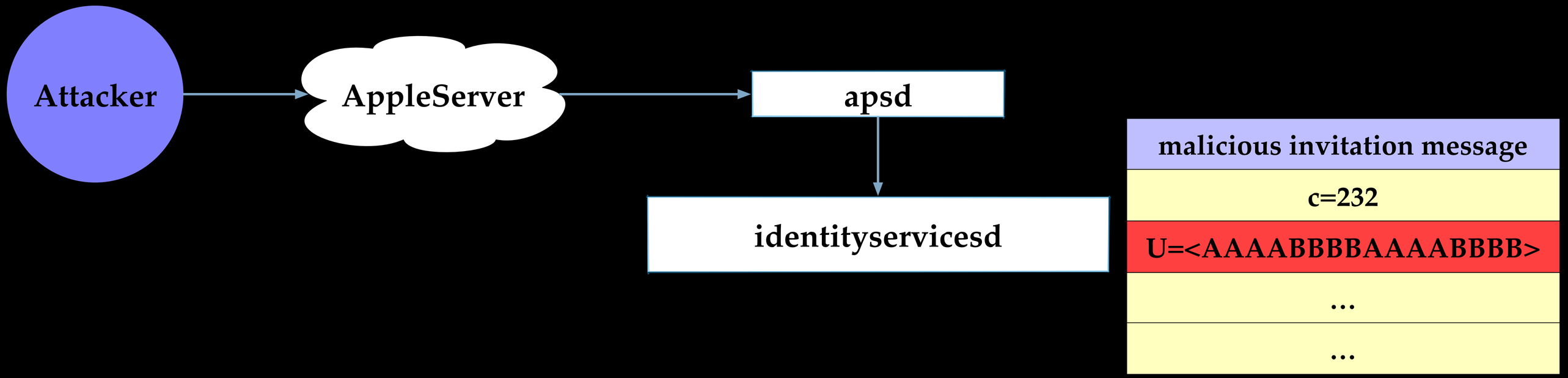

Attacker
AppleServer
apsd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
other attack surfaces
iMessage
remoteDesktop
phonecontinuity
…

Vulnerability Examples

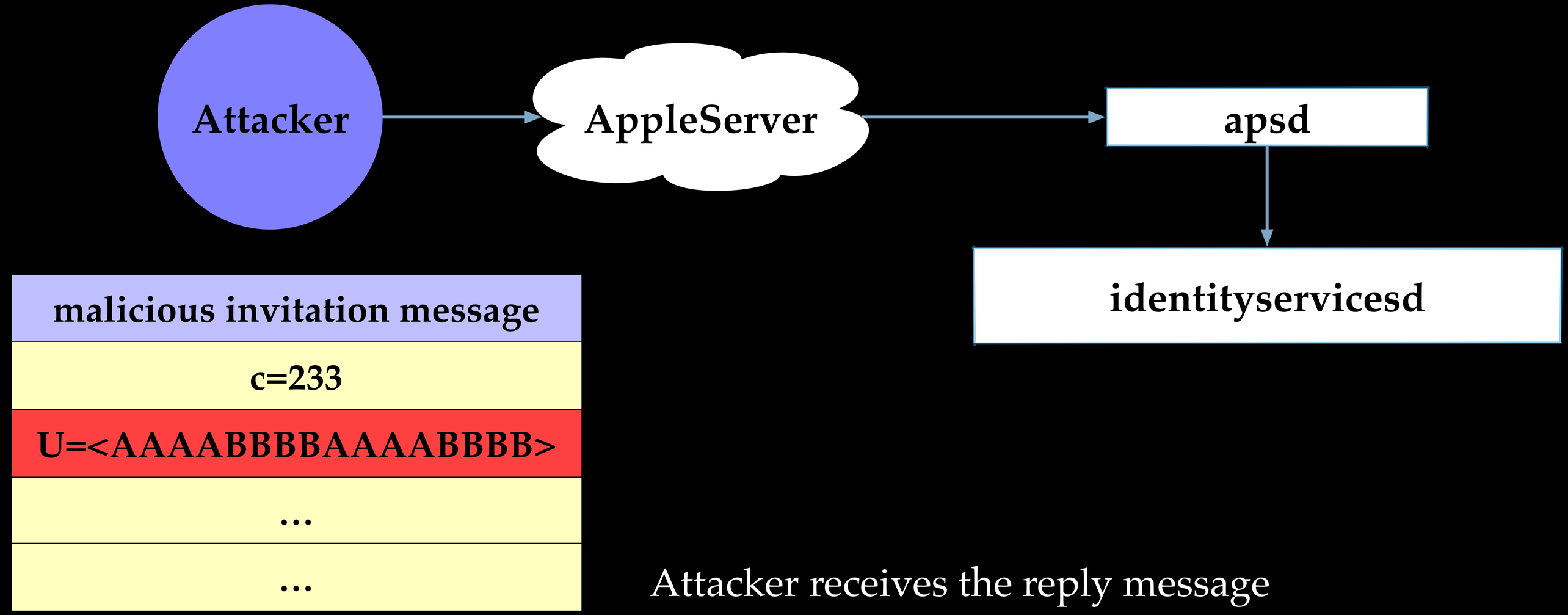

Attacker
AppleServer
Potential Info Leak
apsd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
iMessage
remoteDesktop
phonecontinuity
...



Invitation message contains a field indicating a UUID



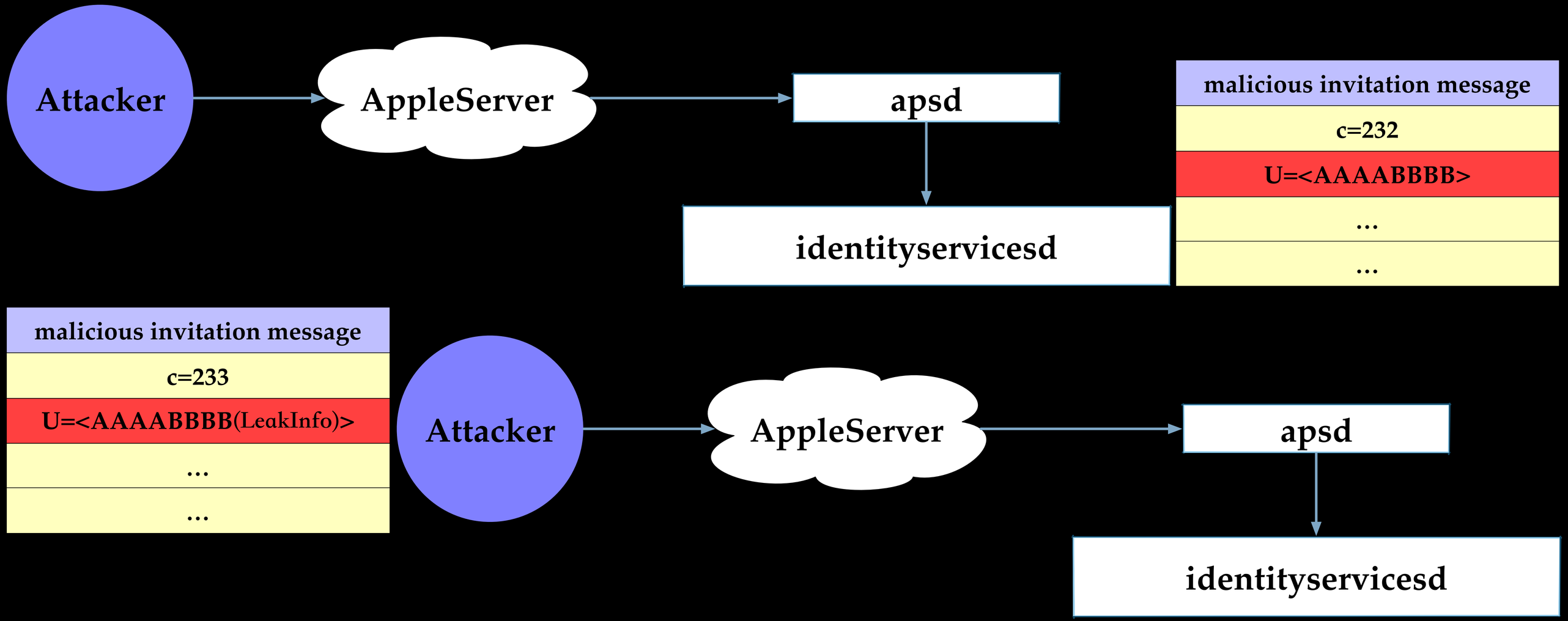
The victim’s identityservicesd will reply a message, using the same UUID


Attacker
AppleServer
apsd
identityservicesd
malicious invitation message
c=233
U=<AAAABBBBAAAABBBB>
…
…
Attacker receives the reply message

```
v20 = IDSUUIDKey;
v21 = objc_msgSend(incomingMessage, "objectForKey:", IDSUUIDKey);
v22 = objc_retainAutoreleasedReturnValue(v21);
v23 = objc_msgSend(&OBJC_CLASS___NSString, "class");
v117 = v22;
v24 = (unsigned __int8)objc_msgSend(v22, "isKindOfClass:", v23);
v25 = objc_msgSend(incomingMessage, "objectForKey:", v20);
v26 = objc_retainAutoreleasedReturnValue(v25);
v27 = v26;
if ( v24 )
{
  v28 = objc_msgSend(v26, "_FTDataFromBase64String");
  v29 = objc_retainAutoreleasedReturnValue(v28);
  v30 = 1;
  v31 = 0;
}
else
{
  v30 = 0;
  v31 = 1;
  v29 = v26;
}
v32 = (void *)JWUUIDPushObjectToString((__int64)v29);
```

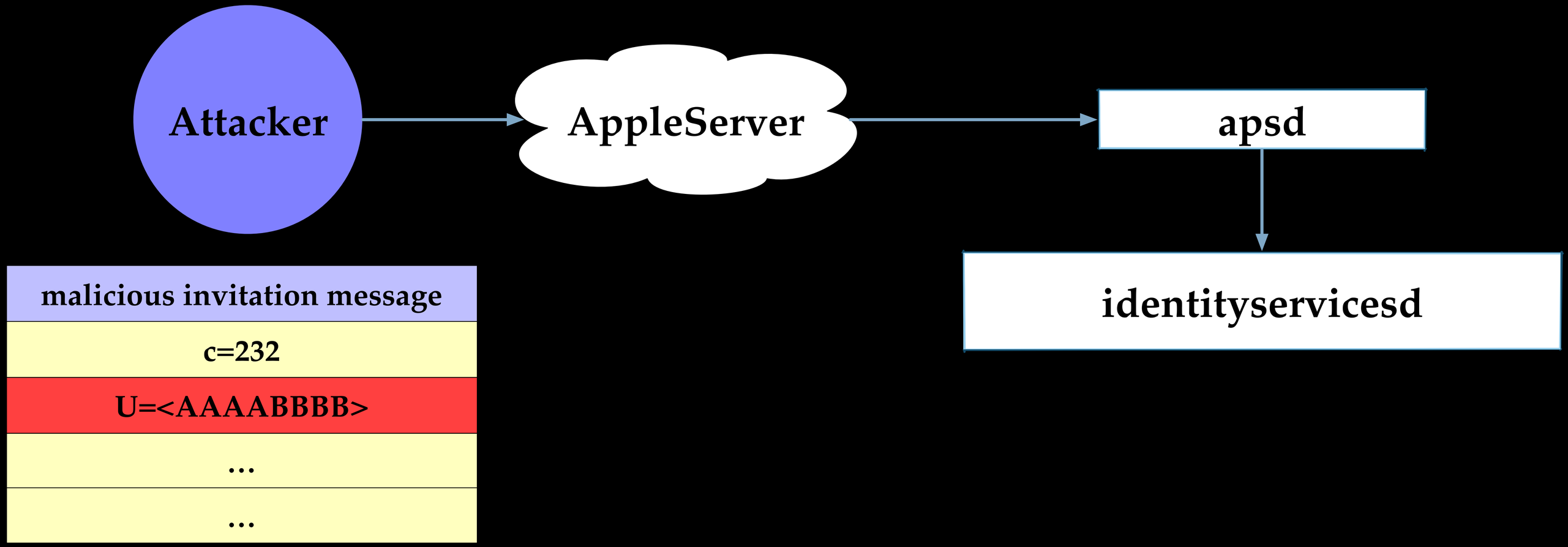
```
id __fastcall JWUUIDPushObjectToString(id a1)
{
  void *v1; // rbx
  id v2; // r14
  unsigned __int8 uu; // [rsp+0h] [rbp-70h]
  uuid_string_t out; // [rsp+10h] [rbp-60h]

  if ( !objc_msgSend(a1, "length") )
    return 0LL;
  objc_msgSend(a1, "getBytes:length:", &uu, 16LL);
  uuid_unparse_upper(&uu, out);
  v1 = (void *)CFStringCreateWithCString(0LL, out, 0x8000100LL);
  v2 = objc_msgSend(v1, "uppercaseString");
  if ( v1 )
    CFRelease(v1);
  return v2;
}
```

- After fetching “U” in the invitation dictionary, identityservicesd presumes it is 16 bytes long
- Convert an NSData (presumed 16 bytes long) to UUID
- Stack variable uu is uninitialized
  - Uninitialized memory leak

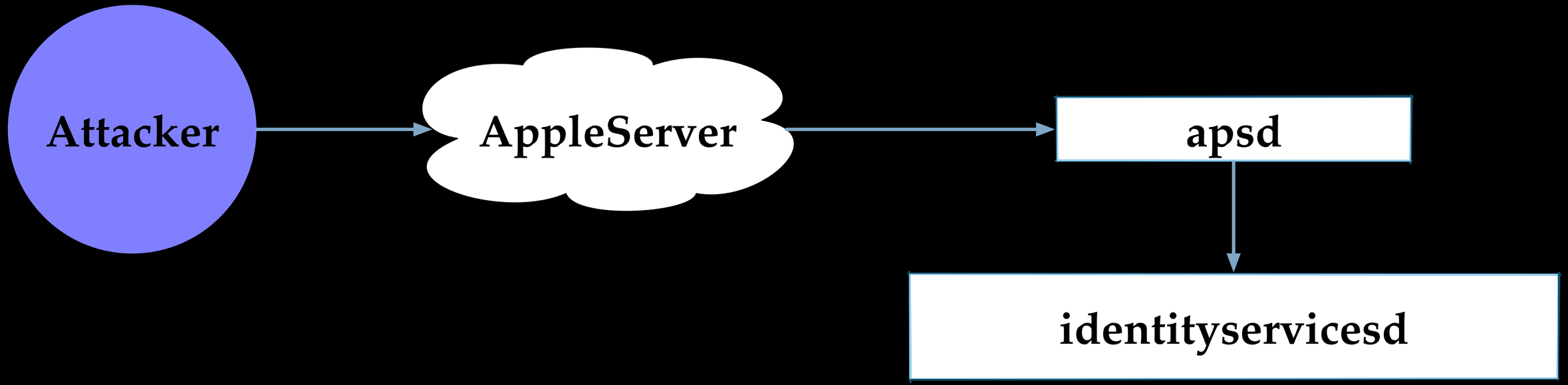


We hope to get an info leak by sending a short UUID in an invitation message

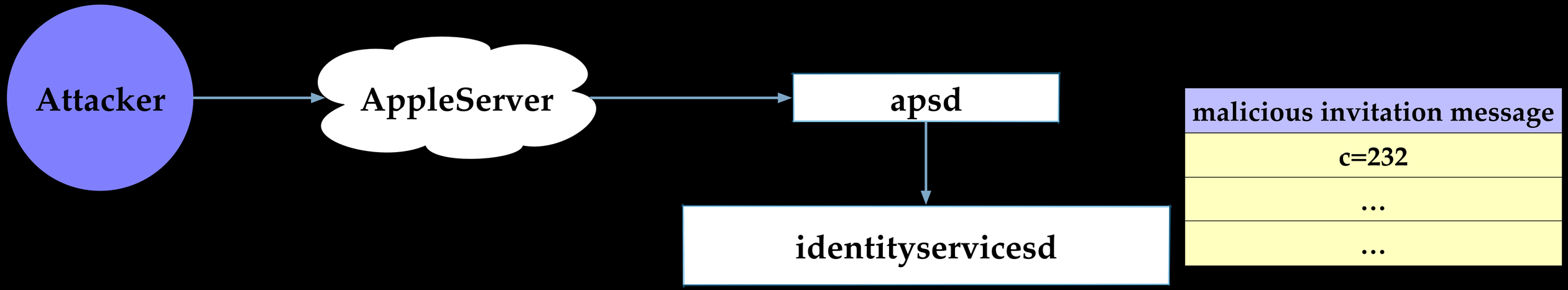
But what real happens



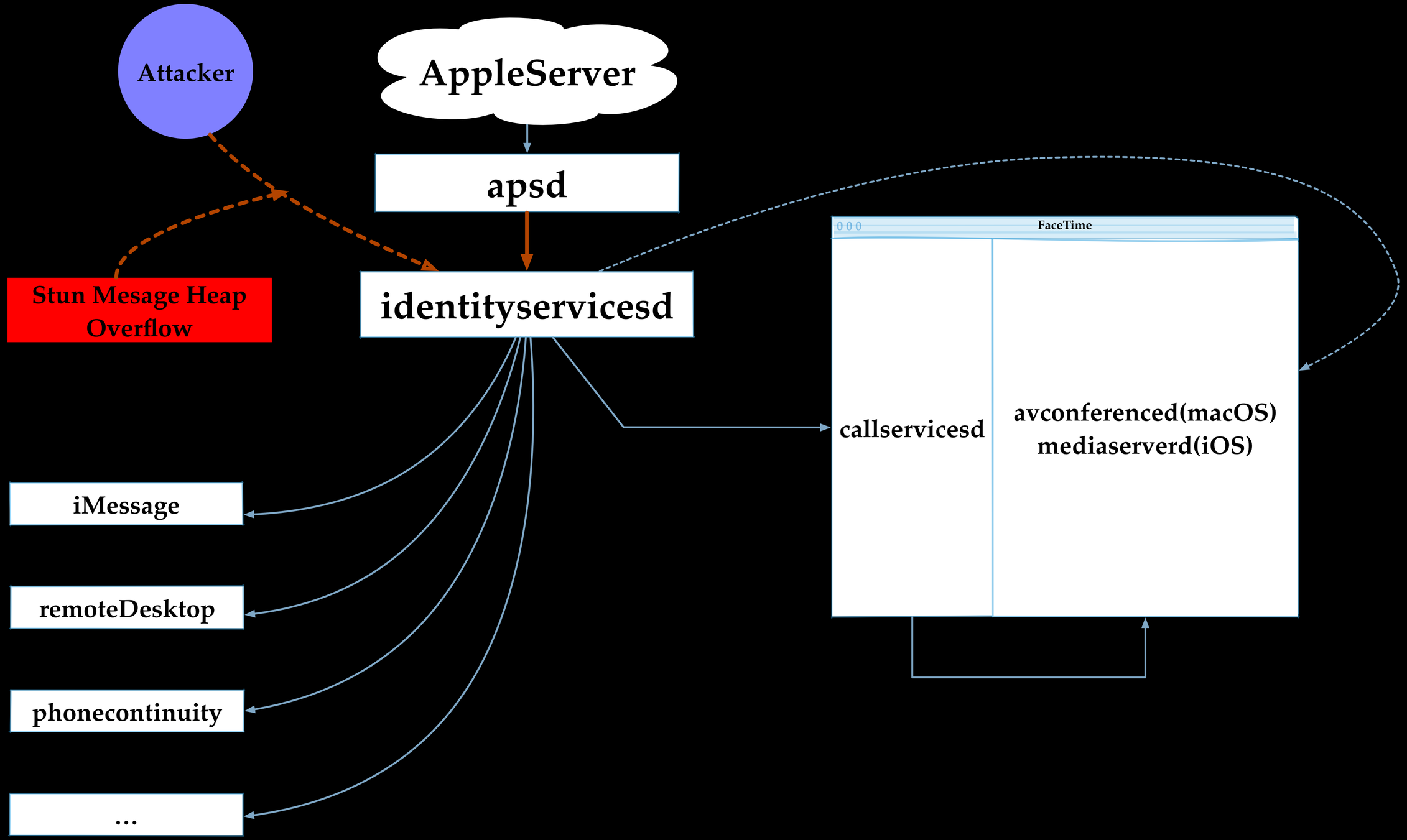
When we send U less then 16 bytes…

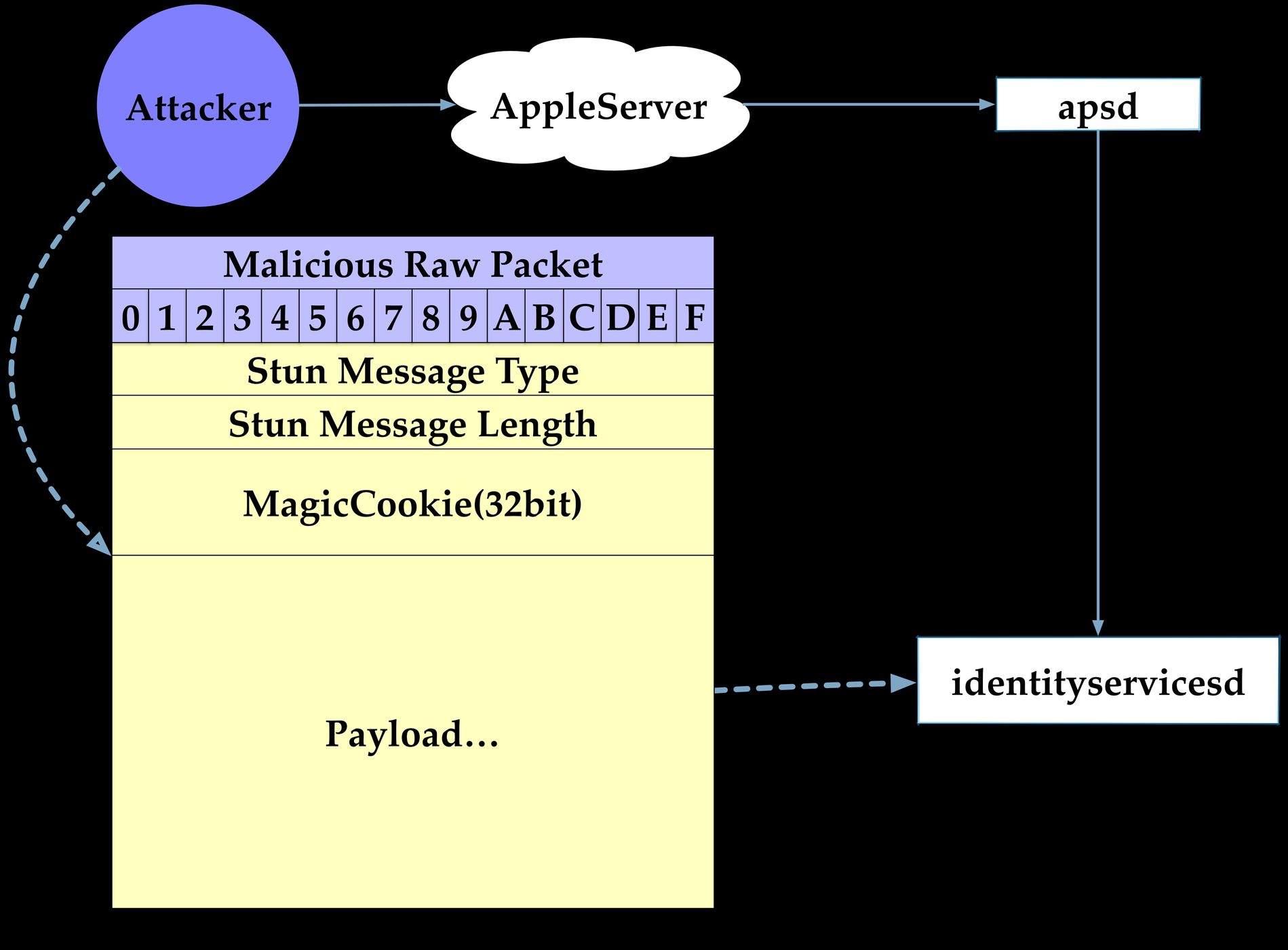


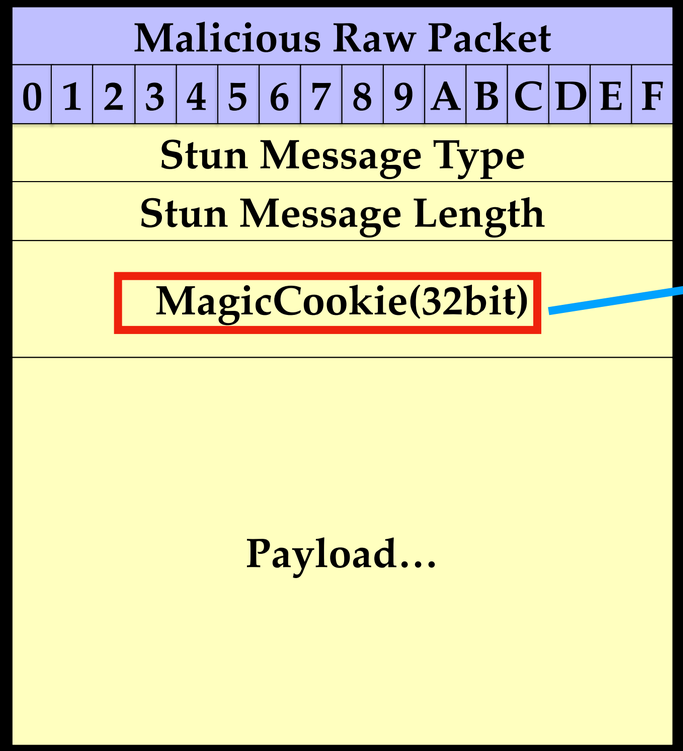
Apple Server seems to have validations on UUIDS



- The received message does not contain the “U” field
- From the code point of view, it’s definitely an info leak.
- We don’t know when Apple Server started to filter out the “short” UUIDs

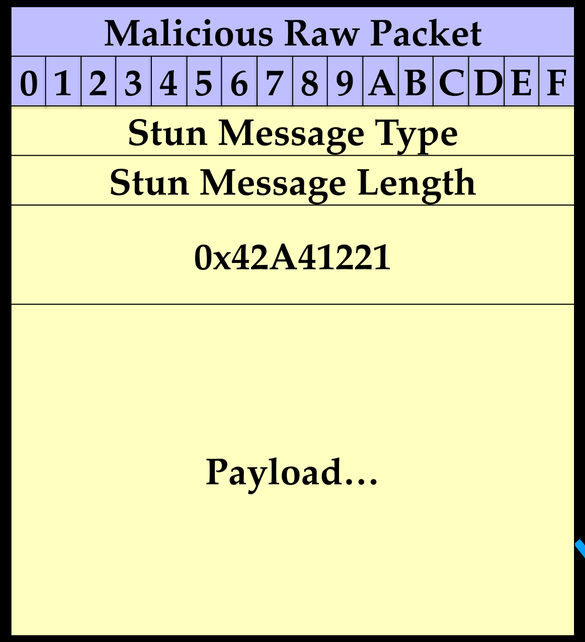

Attacker
AppleServer
apsd
Stun Mesage Heap Overflow
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
iMessage
remoteDesktop
phonecontinuity
...

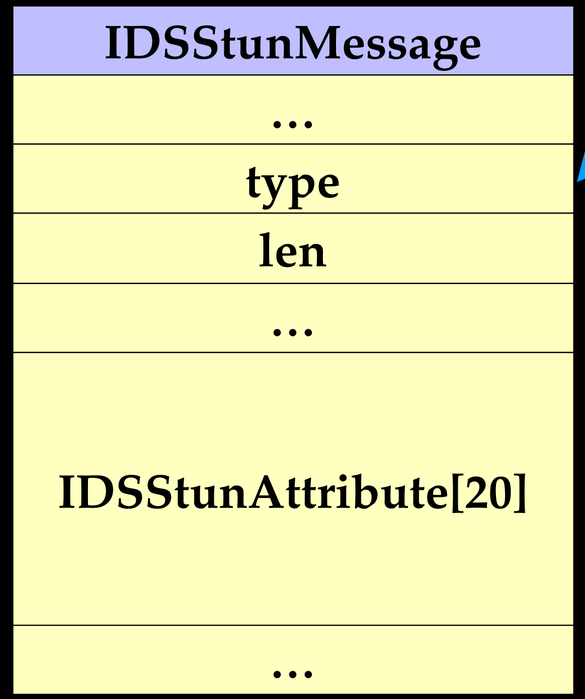

Attacker
AppleServer
apsd
Malicious Raw Packet
0 1 2 3 4 5 6 7 8 9 A B C D E F
Stun Message Type
Stun Message Length
MagicCookie(32bit)
Payload…
identityservicesd




```
if ( *(_DWORD *)(*(_QWORD *)packetStruct + 4LL) == 0x42A41221 )
{
  v56 = objc_msgSend(v97, "headerOverhead");
  objc_msgSend(
    v101,
    "_processStunPacket:fromDeviceUniqueID:cbuuid:arrivalTime:headerOverhead:",
    packetStruct,
    v100,
    v9,
    v56,
    v98);
  v11 = v101;
LABEL_77:
  v102 = 0;
  v104 = 0;
  v103 = 0;
  v12 = 0;
  goto LABEL_14;
}
```

- Session Traversal Utilities for NAT (STUN) is a standardized set of methods, including a network protocol, for traversal of network address translator (NAT) gateways in applications of real-time voice, video, messaging, and other interactive communications.






```
    if ( *(_DWORD *)(*(_QWORD *)packetStruct + 4LL) == 0x42A41221 )
    {
      v56 = objc_msgSend(v97, "headerOverhead");
      objc_msgSend(
        v101,
        "_processStunPacket:fromDeviceUniqueID:cbuuid:arrivalTime:headerOverhead:",
        packetStruct,
        v100,
        v9,
        v56,
        v98);
      v11 = v101;
LABEL_77:
      v102 = 0;
      v104 = 0;
      v103 = 0;
      v12 = 0;
      goto LABEL_14;
    }
```

```
if ( (unsigned __int8)-[IDSStunMessage read:inputLength:internal:](
                        v11,
                        "read:inputLength:internal:",
                        *(_QWORD *)StunPacketStruct,
                        *(int *)(StunPacketStruct + 16),
                        0LL) )
{
```

IDSStunMessage
…
type=0x17
len
…
IDSStunAttribute[0]
IDSStunAttribute[1]
…
IDSStunAttribute[n]
…

```
if ( (unsigned __int16)stunMessageType != 0x17 )
  goto LABEL_153;
objc_msgSend(
  v196,
  "_processDataIndication:fromDevice:localIfIndex:localAddress:remoteAddress:candidatePairToken:arrivalTime:",
  v11,
  v9,
  *(unsigned int *)(StunPacketStruct + 44),
  StunPacketStruct + 48,
  *(double *)&v198,
  StunPacketStruct + 176,
  v195);
}
```

```
if ( (unsigned __int8)objc_msgSend(v10, "getAttribute:attribute:", 19LL, attr) )
{
  v12 = (unsigned __int8)-[IDSGlobalLink _processIncomingIndicationData:length:candidatePairToken:arrivalTime:](
                           self,
                           "_processIncomingIndicationData:length:candidatePairToken:arrivalTime:",
                           &attr[12],
                           *(unsigned int *)&attr[8],
                           v11,
                           a9,
                           v20);
}
```

Certain type(0x17) of Stun message will go to

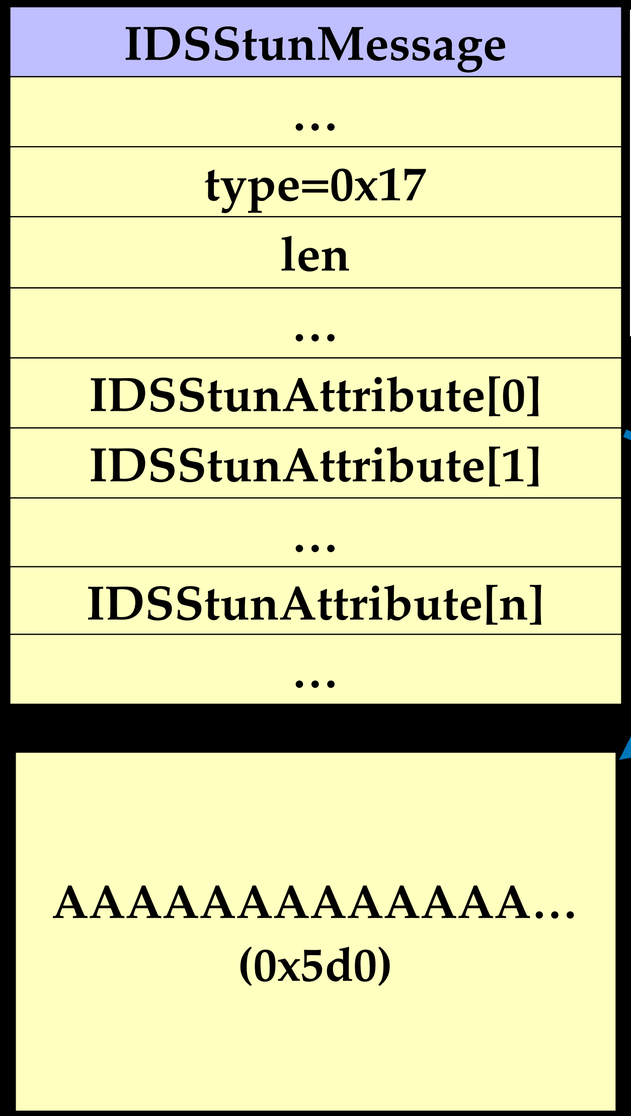
[IDSGlobalLink _processDataIndication:fromDevice:localIfIndex:localAddress:remoteAddress:candidatePairToken:arrivalTime:]

IDSStunMessage
…
type=0x17
len
…
IDSStunAttribute[0]
IDSStunAttribute[1]
…
IDSStunAttribute[n]
…

```
if ( (unsigned __int8)objc_msgSend(v10, "getAttribute:attribute:", 19LL, attr) )
{
  v12 = (unsigned __int8)-[IDSGlobalLink _processIncomingIndicationData:length:candidatePairToken:arrivalTime:](
                           self,
                           "_processIncomingIndicationData:length:candidatePairToken:arrivalTime:",
                           &attr[12],
                           *(unsigned int *)&attr[8],
                           v11,
                           a9,
                           v20);
}
```

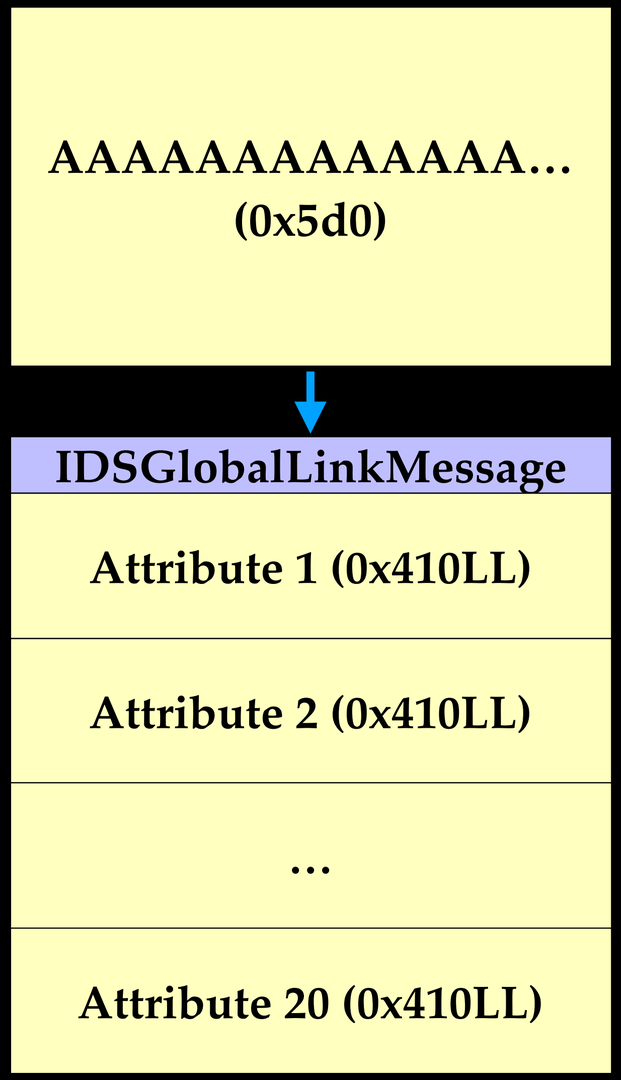
[IDSStunMessage getAttribute:attribute:] copies raw packet payload to a local buffer

```
char __cdecl -[IDSStunMessage getAttribute:attribute:]
{
  __int64 v4; // r8
  __int64 v5; // rax
  __int64 v6; // rdi

  v4 = self->_numAttribute;
  if ( v4 <= 0 )
    return 0;
  v5 = (__int64)self->_attributes;
  v6 = 0LL;
  while ( *(unsigned __int16 *)v5 != a3 )
  {
    ++v6;
    v5 += 0x5D0LL;
    if ( v6 >= v4 )
      return 0;
  }
  memcpy(a4, (const void *)v5, 0x5D0uLL);
  return 1;
}
```




```
if ( (unsigned __int8)objc_msgSend(v10, "getAttribute:attribute:", 19LL, attr) )
{
  v12 = (unsigned __int8)-[IDSGlobalLink _processIncomingIndicationData:length:candidatePairToken:arrivalTime:](
                           self,
                           "_processIncomingIndicationData:length:candidatePairToken:arrivalTime:",
                           &attr[12],
                           *(unsigned int *)&attr[8],
                           v11,
                           a9,
                           v20);
}
```

```
char __cdecl -[IDSStunMessage getAttribute:attribute:]
{
  __int64 v4; // r8
  __int64 v5; // rax
  __int64 v6; // rdi

  v4 = self->_numAttribute;
  if ( v4 <= 0 )
    return 0;
  v5 = (__int64)self->_attributes;
  v6 = 0LL;
  while ( *(unsigned __int16 *)v5 != a3 )
  {
    ++v6;
    v5 += 0x5D0LL;
    if ( v6 >= v4 )
      return 0;
  }
  memcpy(a4, (const void *)v5, 0x5D0uLL);
  return 1;
}
```




```
char __cdecl -[IDSGlobalLinkMessage read:inputLength:](IDSGlobalLinkMessage *self, SEL a2, __int64 pData, int length)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( length <= 19 )
  {
    v5 = (void *)OSLogHandleForTransportCategory("GL");
    v6 = objc_retainAutoreleasedReturnValue(v5);
    v7 = 0LL;
    if...
    objc_release(v6);
    if...
    return 0;
  }
  self->_command = (unsigned __int16)__ROL2__(*(_WORD *)pData, 8);
  v13 = (unsigned __int16)__ROL2__(*(_WORD *)(pData + 2), 8) + 20;
  if ( v13 <= length )
  {
    LODWORD(v21) = 0;
    if ( length >= 21 )
    {
      pEnd = pData + length;
      pCurNode = pData + 20;
      _attr = (__int64)self->_attributes;
      v21 = 0LL;
      do
      {
        nodeType = __ROL2__(*(_WORD *)pCurNode, 8);
        *(_WORD *)_attr = nodeType;
        nodeSize = __ROL2__(*(_WORD *)(pCurNode + 2), 8);
        *(_WORD *)(_attr + 2) = nodeSize;
        _nodeSZ = nodeSize;
        pCurNodeData = (_WORD *)(pCurNode + 4);
```

This local buffer is used to deserialize an IDSGlobalLink message

CurNode	
+0	Type
+2	Length
+4	Payload

```
nodeType = __ROL2__(*(_WORD *)pCurNode, 8);
*(_WORD *)_attr = nodeType;
nodeSize = __ROL2__(*(_WORD *)(pCurNode + 2), 8);
*(_WORD *)(_attr + 2) = nodeSize;
_nodeSZ = nodeSize;
pCurNodeData = (_WORD *)(pCurNode + 4);
switch ( (unsigned __int16)(nodeType - 1) )
{
  case 0u:
  case 1u:
  case 4u:
  case 5u:
  case 0xBu:
    readIDSGLAttrU16(_attr, pCurNodeData, &_nodeSZ);
    break;
  case 2u:
  case 3u:
  case 8u:
  case 9u:
  case 0xAu:
  case 0xCu:
  case 0xDu:
  case 0xEu:
  case 0xFu:
    readIDSGLAttrBinaryData(_attr, pCurNodeData, &_nodeSZ);
```

CurNode	
+0	Type
+2	Length
+4	Payload

CurNode	
+0	0xF
+2	0X4141
+4	0x4141414141414141…

```
nodeType = __ROL2__(*(_WORD *)pCurNode, 8);
*(_WORD *)_attr = nodeType;
nodeSize = __ROL2__(*(_WORD *)(pCurNode + 2), 8);
*(_WORD *)(_attr + 2) = nodeSize;
_nodeSZ = nodeSize;
pCurNodeData = (_WORD *)(pCurNode + 4);
switch ( (unsigned __int16)(nodeType - 1) )
{
  case 0u:
  case 1u:
  case 4u:
  case 5u:
  case 0xBu:
    readIDSGLAttrU16(_attr, pCurNodeData, &_nodeSZ);
    break;
  case 2u:
  case 3u:
  case 8u:
  case 9u:
  case 0xAu:
  case 0xCu:
  case 0xDu:
  case 0xEu:
  case 0xFu:
    readIDSGLAttrBinaryData(_attr, pCurNodeData, &_nodeSZ);
```

CurNode	
+0	0xF
+2	0X4141
+4	0x41414141414141414141…

```
nodeType = __ROL2__(*(_WORD *)pCurNode, 8);
*(_WORD *)_attr = nodeType;
nodeSize = __ROL2__(*(_WORD *)(pCurNode + 2), 8);
*(_WORD *)(_attr + 2) = nodeSize;
_nodeSZ = nodeSize;
pCurNodeData = (_WORD *)(pCurNode + 4);
switch ( (unsigned __int16)(nodeType - 1) )
{
  case 0u:
  case 1u:
  case 4u:
  case 5u:
  case 0xBu:
    readIDSGLAttrU16(_attr, pCurNodeData, &_nodeSZ);
    break;
  case 2u:
  case 3u:
  case 8u:
  case 9u:
  case 0xAu:
  case 0xCu:
  case 0xDu:
  case 0xEu:
  case 0xFu:
    readIDSGLAttrBinaryData(_attr, pCurNodeData, &_nodeSZ);
```

CurNode	
+0	0xF
+2	0X4141
+4	0x41414141414141414141…

T=0xF	L=0x4141	V=0x414141414141….

```
nodeType = __ROL2__(*(_WORD *)pCurNode, 8);
*(_WORD *)_attr = nodeType;
nodeSize = __ROL2__(*(_WORD *)(pCurNode + 2), 8);
*(_WORD *)(_attr + 2) = nodeSize;
_nodeSZ = nodeSize;
pCurNodeData = (_WORD *)(pCurNode + 4);
switch ( (unsigned __int16)(nodeType - 1) )
{
  case 0u:
  case 1u:
  case 4u:
  case 5u:
  case 0xBu:
    readIDSGLAttrU16(_attr, pCurNodeData, &_nodeSZ);
    break;
  case 2u:
  case 3u:
  case 8u:
  case 9u:
  case 0xAu:
  case 0xCu:
  case 0xDu:
  case 0xEu:
  case 0xFu:
    readIDSGLAttrBinaryData(_attr, pCurNodeData, &_nodeSZ);
```

```
__int64 __fastcall readIDSGLAttrBinaryData(
{
  *(_DWORD *)(a1 + 8) = *a3;
  memcpy((void *)(a1 + 12), a2, *a3);
  return 1LL;
}
```

During the deserialization, memcpy's size argument is under attacker's control

```
__int64 __fastcall readIDSGLAttrBinaryData(
{
  *(_DWORD *)(a1 + 8) = *a3;
  memcpy((void *)(a1 + 12), a2, *a3);
  return 1LL;
}
```

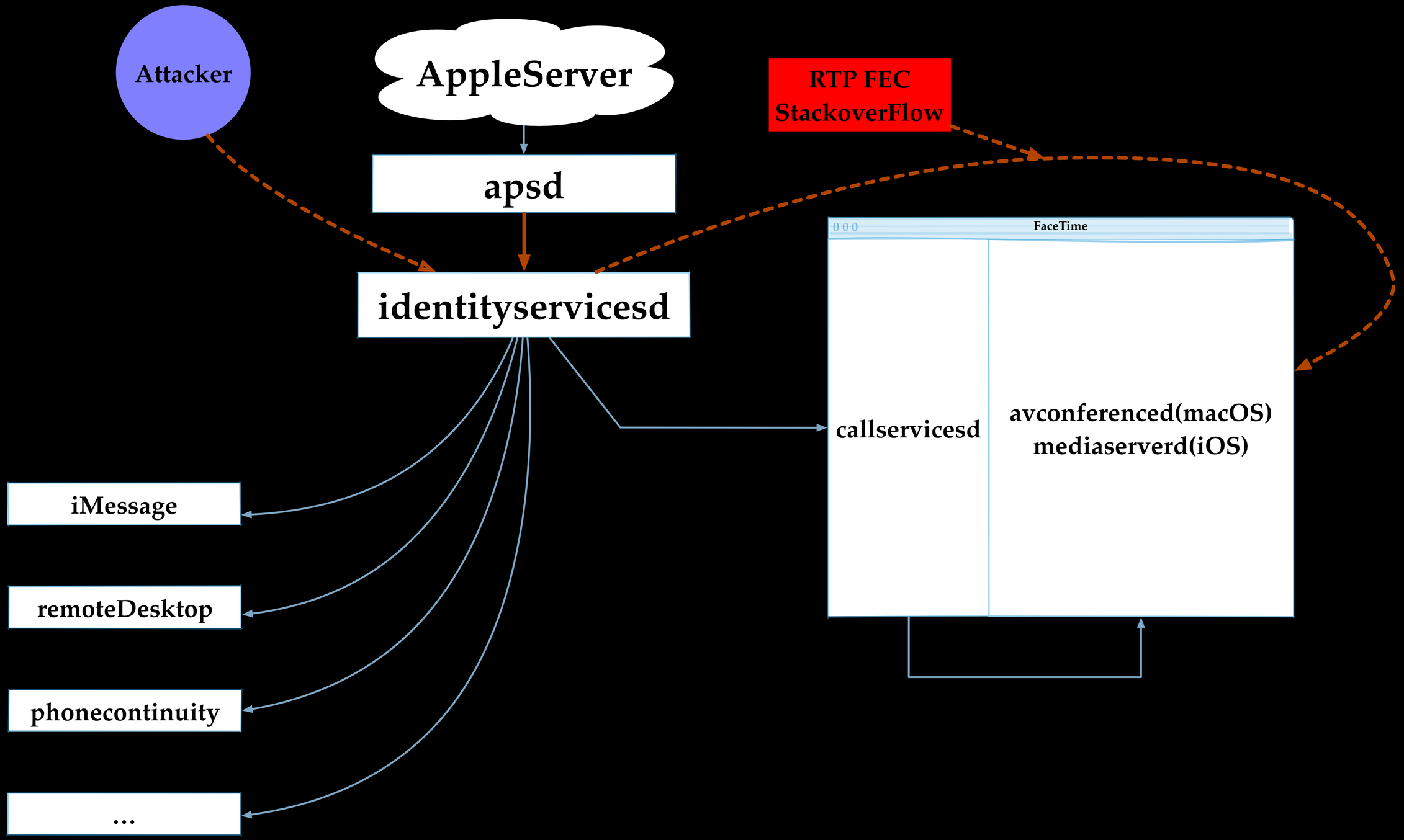
```
Thread 2 name:  TransportThread Primary
Thread 2 Crashed:
0   libsystem_platform.dylib        0x00000001b1235d94 0x1b122e000 + 32148
1   IDSFoundation                   0x00000001bc950e74 0x1bc88a000 + 814708 _readIDSGLAttrBinaryData
2   IDSFoundation                   0x00000001bc8e7938 0x1bc88a000 + 383288 -[IDSGlobalLinkMessage read:inputLength:]
3   IDSFoundation                   0x00000001bc8e62bc 0x1bc88a000 + 377532 +[IDSGlobalLinkMessage messageWithBuffer:length:]
4   IDSFoundation                   0x00000001bc8c13a8 0x1bc88a000 + 226216 -[IDSGlobalLink _processIncomingIndicationData:length:candidatePairToken:arrivalTime:]
5   IDSFoundation                   0x00000001bc8b7528 0x1bc88a000 + 185640 [IDSGlobalLink
_processDataIndication:fromDevice:localIfIndex:localAddress:remoteAddress:candidatePairToken:arrivalTime:]
6   IDSFoundation                   0x00000001bc8ba124 0x1bc88a000 + 196900 -[IDSGlobalLink _processStunPacket:fromDeviceUniqueID:cbuuid:arrivalTime:headerOverhead:]
7   IDSFoundation                   0x00000001bc8db040 0x1bc88a000 + 331840 -[IDSGlobalLink link:didReceivePacket:fromDeviceUniqueID:cbuuid:]
```

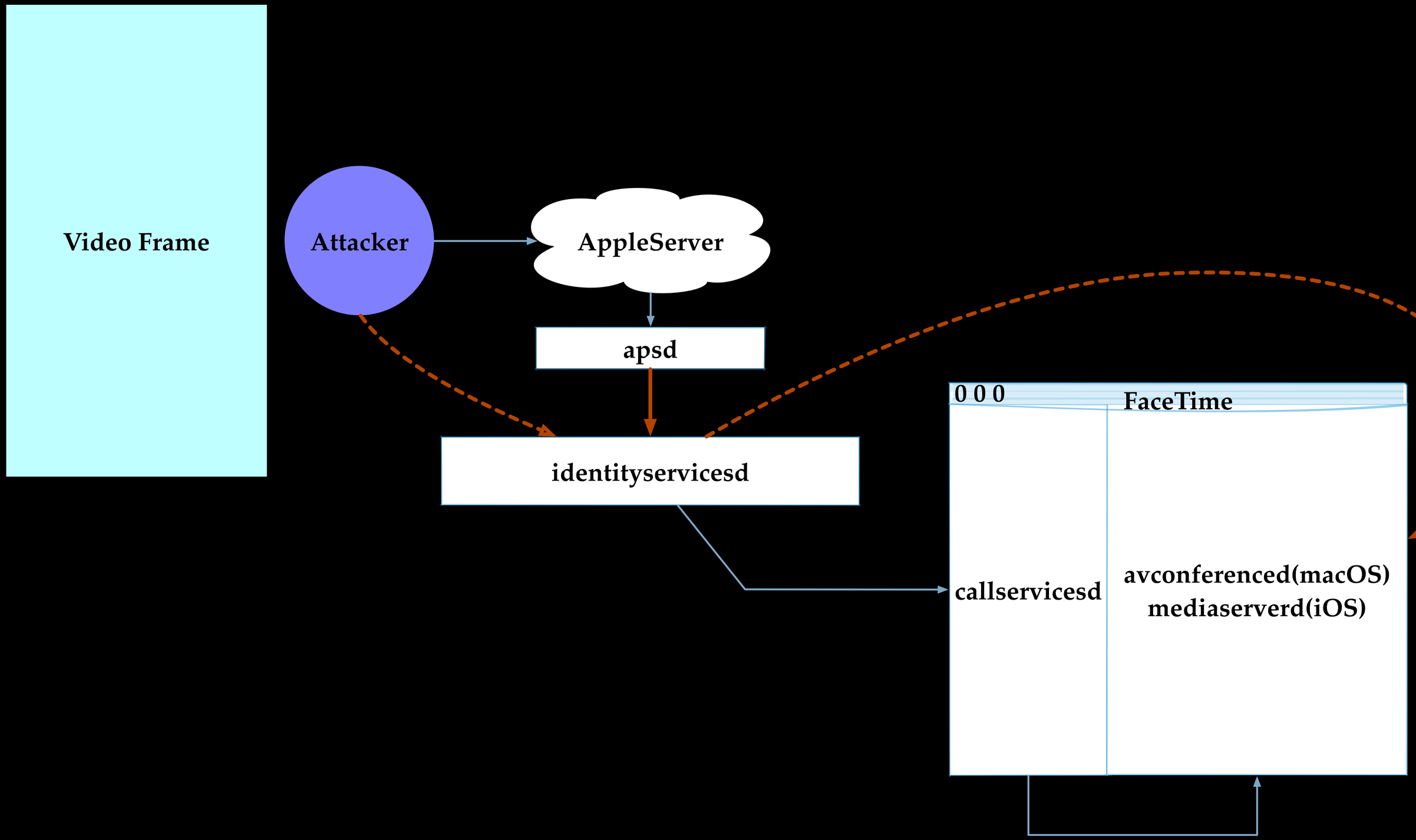
memcpy will easily trigger such a kind of memory corruptions

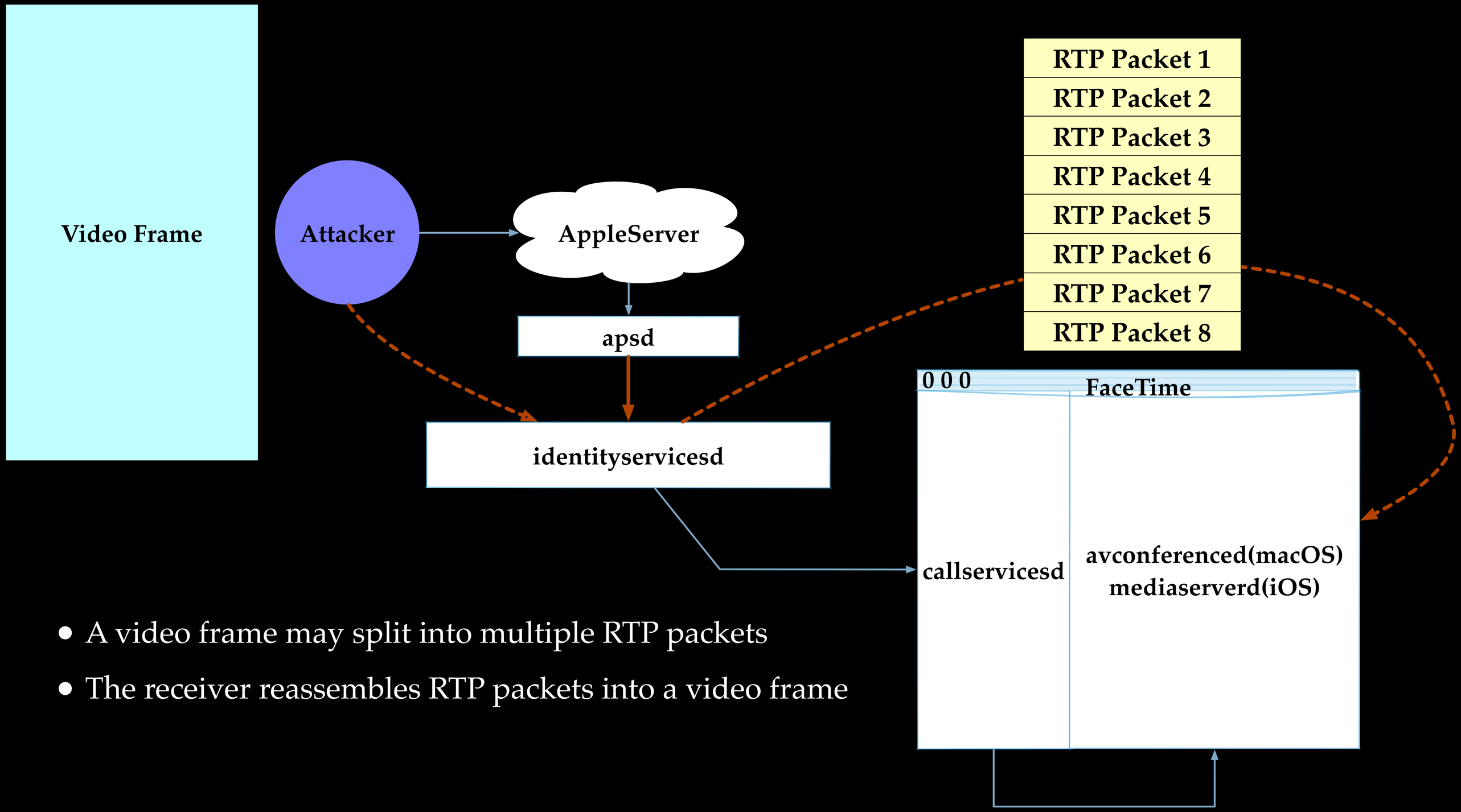
- The heap overflow can overwrite the ISA pointer of adjacent objects

```
Crashed Thread:        2  TransportThread Primary

Exception Type:        EXC_BAD_ACCESS (SIGSEGV)
Exception Codes:       KERN_INVALID_ADDRESS at 0x0000414141414160
Exception Note:        EXC_CORPSE_NOTIFY

Termination Signal:    Segmentation fault: 11
Termination Reason:    Namespace SIGNAL, Code 0xb
Terminating Process:   exc handler [12181]

VM Regions Near 0x414141414160:
    __LINKEDIT             000000011c1e9000-000000011c212000 [  164K] r--/rwx SM=COW  /usr/lib/dyld
-->
    STACK GUARD            0000700002508000-0000700002509000 [    4K] ---/rwx SM=NUL  stack guard for thread 6
```

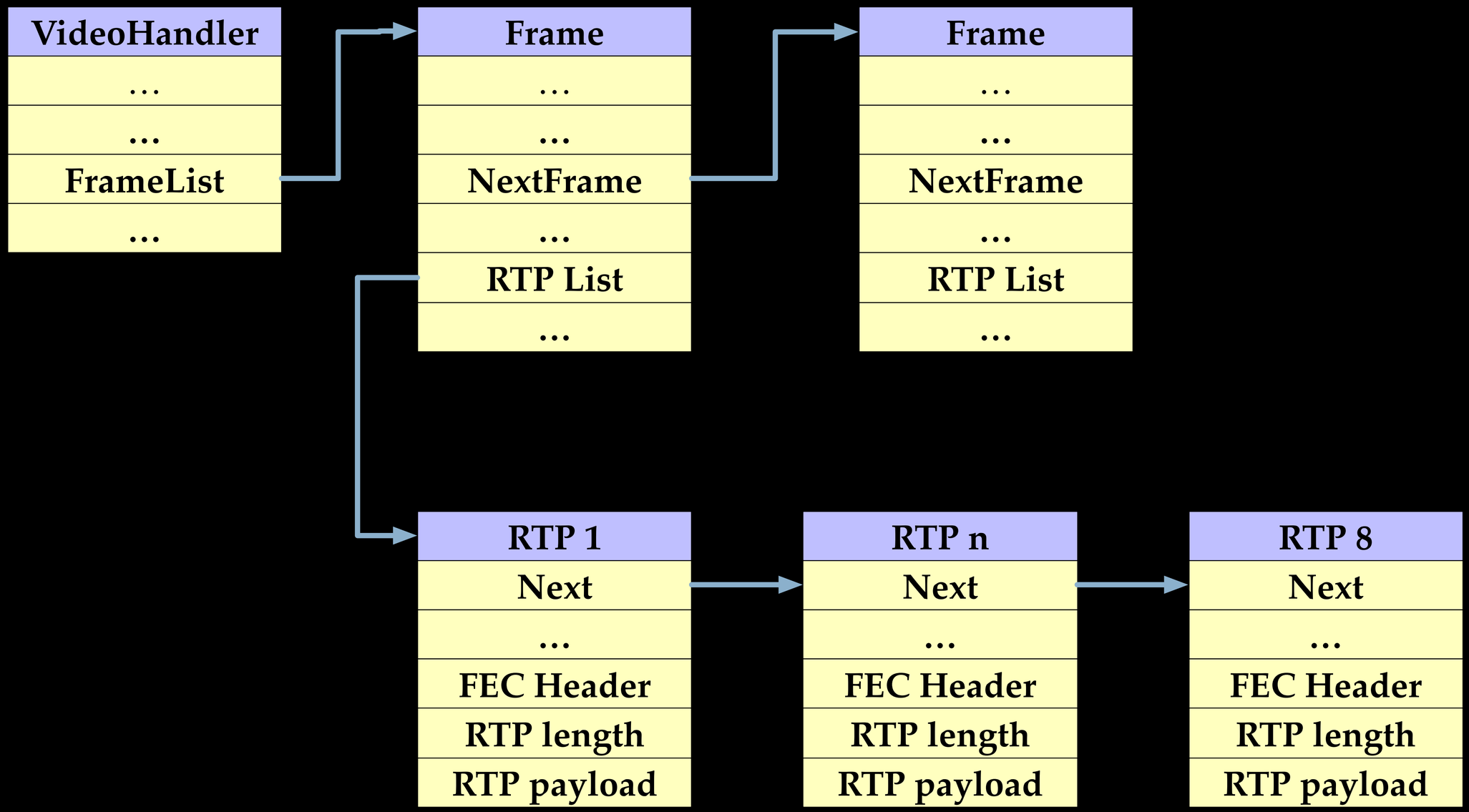
- Please refer to Neil Archibald, “Modern Objective-C Exploitation”, for exploitation techniques


Attacker
AppleServer
RTP FEC
StackoverFlow
apsd
identityservicesd
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)
iMessage
remoteDesktop
phonecontinuity
...

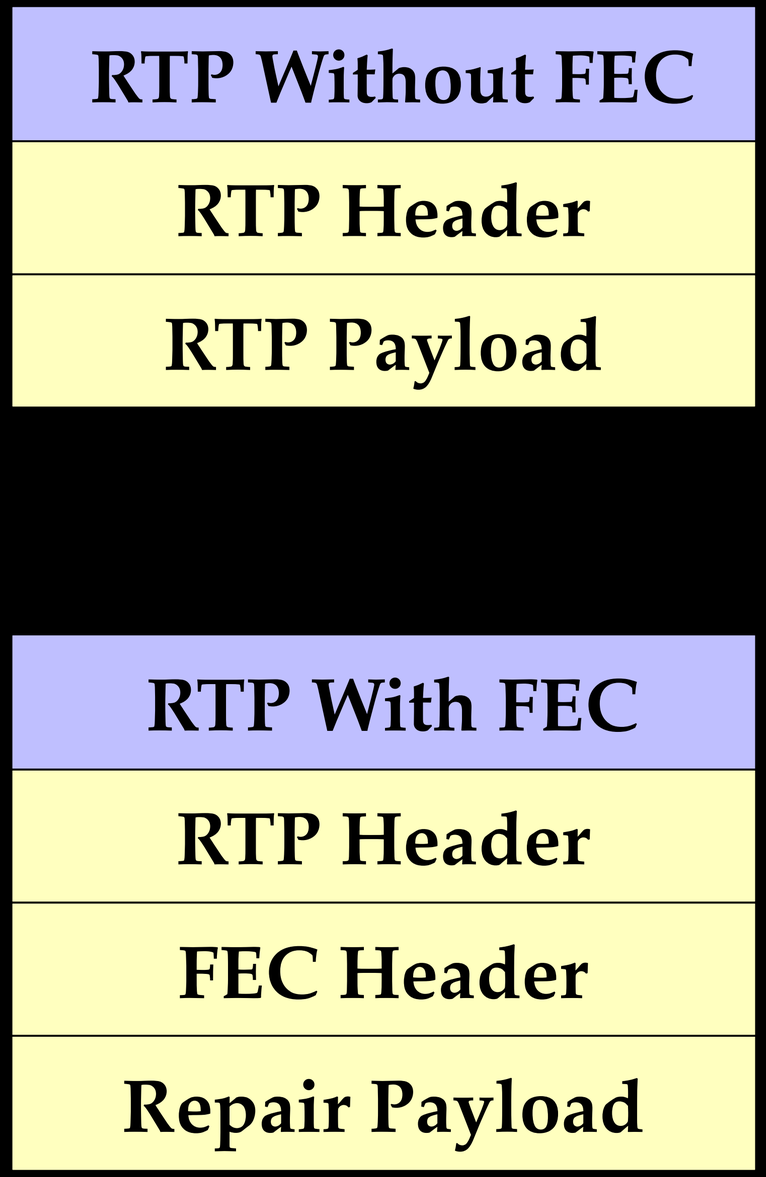

Video Frame
Attacker
AppleServer
apsd
identityservicesd
0 0 0
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)



- A video frame may split into multiple RTP packets
- The receiver reassembles RTP packets into a video frame



Length of each RTP packet is strictly checked at receiving point

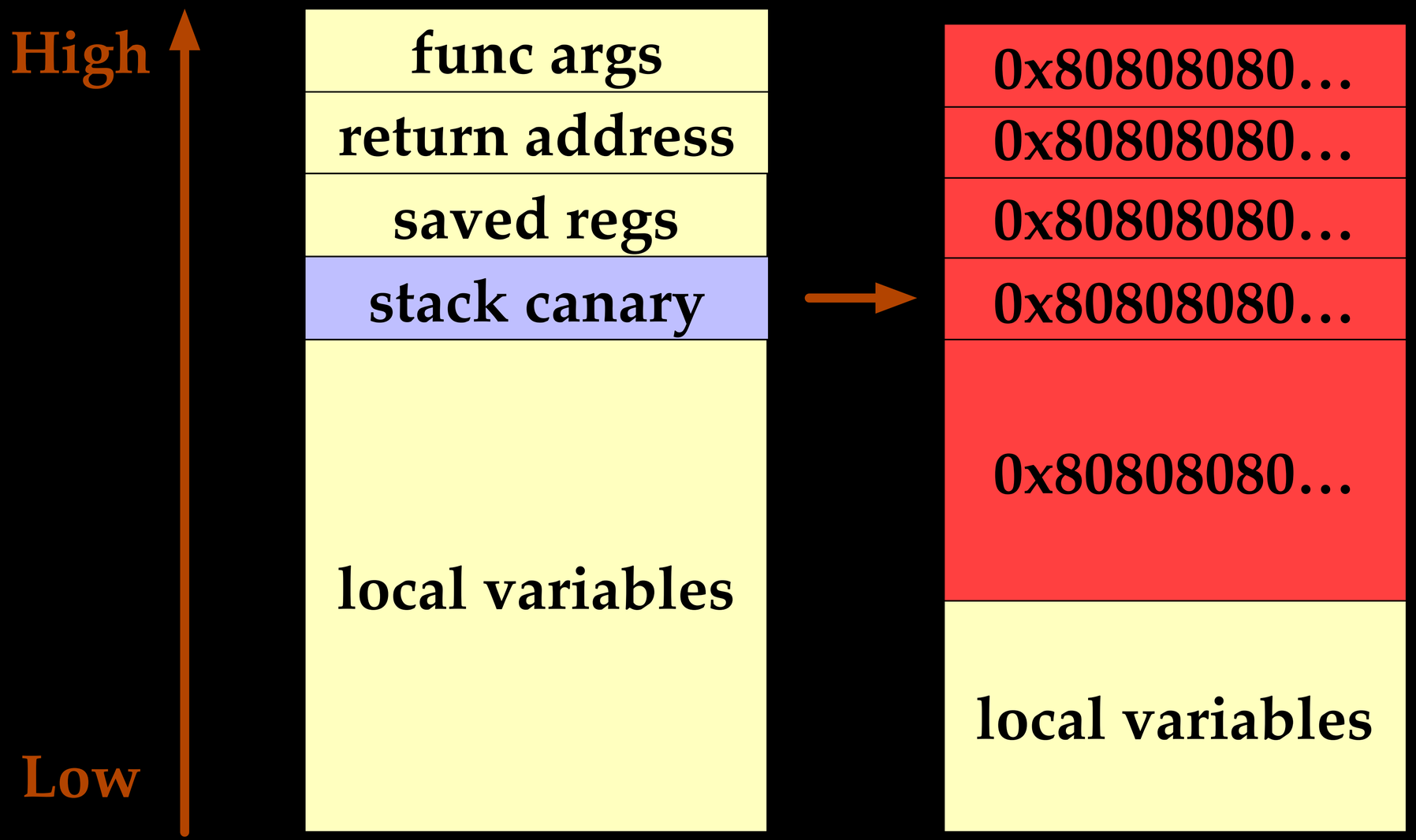


- FaceTime uses the Forward Error Correction (FEC) mechanism to minimize the influence of packet losses
- In case of packet loss, FaceTime will try to recover the lost packet, and build a repaired RTP packet
- However, the length of the repaired RTP packet is copied from FEC header, fully under attacker's control

- When the repaired frame is scheduled to process, avconferenced will execute the following functions:
  - AssembleFrame_H264
    - -> RTPProcessH264Packet(…)
      - -> memcpy(stack_buffer, heap_buffer, size)
  - Note that size is recovered from FEC header, under attacker’s control

Question: is stack overflow exploitable on iOS in 2019?

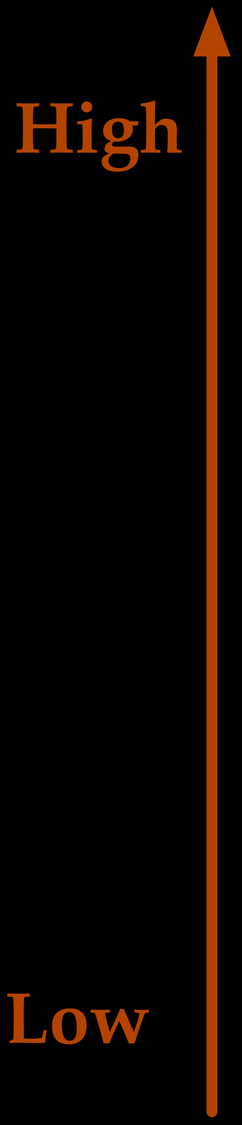
Stack Canary should effectively prevent stack overflows



```
_AssembleFrame_H264                     ; CODE XREF: __text:0000000198A03A78↑p

var_3AB0        = -0x3AB0
var_3AA8        = -0x3AA8
var_3A98        = -0x3A98
var_3A88        = -0x3A88
var_3A80        = -0x3A80
var_3A78        = -0x3A78
var_3A70        = -0x3A70
var_3A68        = -0x3A68
var_3A60        = -0x3A60
var_3A08        = -0x3A08
anonymous_2     = -0x39F2
anonymous_1     = -0x39F0
anonymous_0     = -0x39E8
anonymous_3     = -0x39D0
anonymous_4     = -0x39C8
var_50          = -0x50
var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10
var_s0          =  0

                STP             X28, X27, [SP,#-0x10+var_50]!
                STP             X26, X25, [SP,#0x50+var_40]
                STP             X24, X23, [SP,#0x50+var_30]
                STP             X22, X21, [SP,#0x50+var_20]
                STP             X20, X19, [SP,#0x50+var_10]
                STP             X29, X30, [SP,#0x50+var_s0]
                ADD             X29, SP, #0x50
                SUB             SP, SP, #3,LSL#12
                SUB             SP, SP, #0xA00
                MOV             X19, SP
                MOV             X21, X0
                ADD             X9, X19, #0x50 ; 'P'
                ADRP            X8, #stack_cookie@PAGE
                LDR             X8, [X8,#stack_cookie@PAGEOFF]
                LDR             X8, [X8]
                STR             X8, [X9]
                STR             X1, [X19,#0x3A50+var_3A08]
```

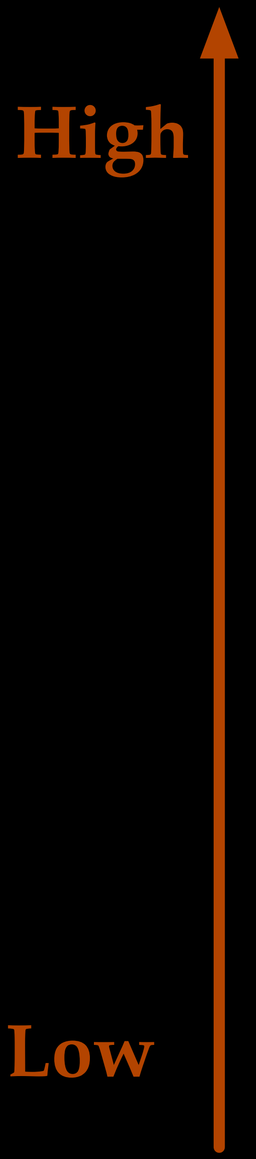
stack cookie is placed below local variables

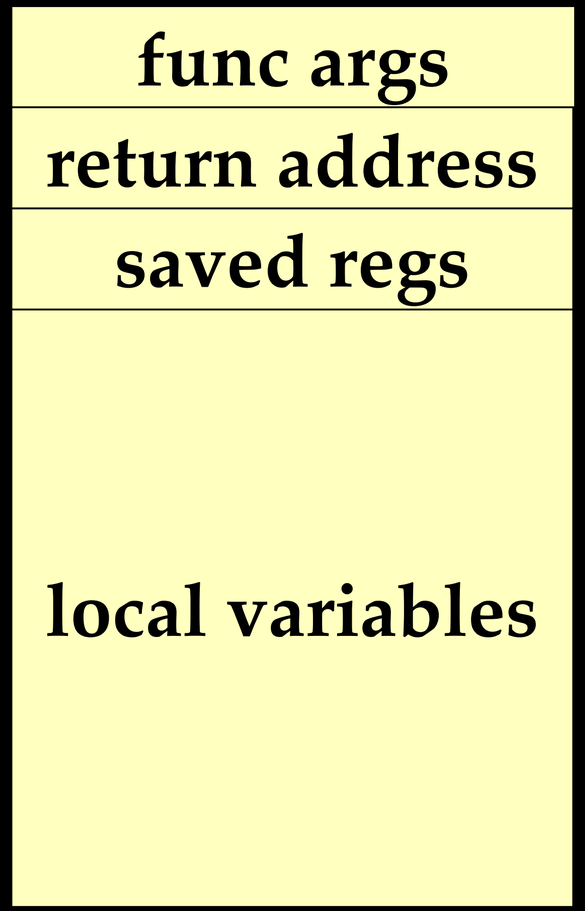
```
_AssembleFrame_H264                     ; CODE XREF: __text:0000000198A03A78↑p

var_3AB0        = -0x3AB0
var_3AA8        = -0x3AA8
var_3A98        = -0x3A98
var_3A88        = -0x3A88
var_3A80        = -0x3A80
var_3A78        = -0x3A78
var_3A70        = -0x3A70
var_3A68        = -0x3A68
var_3A60        = -0x3A60
var_3A08        = -0x3A08
anonymous_2     = -0x39F2
anonymous_1     = -0x39F0
anonymous_0     = -0x39E8
anonymous_3     = -0x39D0
anonymous_4     = -0x39C8
var_50          = -0x50
var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10
var_s0          =  0

                STP             X28, X27, [SP,#-0x10+var_50]!
                STP             X26, X25, [SP,#0x50+var_40]
                STP             X24, X23, [SP,#0x50+var_30]
                STP             X22, X21, [SP,#0x50+var_20]
                STP             X20, X19, [SP,#0x50+var_10]
                STP             X29, X30, [SP,#0x50+var_s0]
                ADD             X29, SP, #0x50
                SUB             SP, SP, #3,LSL#12
                SUB             SP, SP, #0xA00
                MOV             X19, SP
                MOV             X21, X0
                ADD             X9, X19, #0x50 ; 'P'
                ADRP            X8, #stack_cookie@PAGE
                LDR             X8, [X8,#stack_cookie@PAGEOFF]
                LDR             X8, [X8]
                STR             X8, [X9]
                STR             X1, [X19,#0x3A50+var_3A08]
```



func args
return address
saved regs

```
_AssembleFrame_H264                     ; CODE XREF: __text:0000000198A03A78↑p

var_3AB0        = -0x3AB0
var_3AA8        = -0x3AA8
var_3A98        = -0x3A98
var_3A88        = -0x3A88
var_3A80        = -0x3A80
var_3A78        = -0x3A78
var_3A70        = -0x3A70
var_3A68        = -0x3A68
var_3A60        = -0x3A60
var_3A08        = -0x3A08
anonymous_2     = -0x39F2
anonymous_1     = -0x39F0
anonymous_0     = -0x39E8
anonymous_3     = -0x39D0
anonymous_4     = -0x39C8
var_50          = -0x50
var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10
var_s0          =  0

                STP             X28, X27, [SP,#-0x10+var_50]!
                STP             X26, X25, [SP,#0x50+var_40]
                STP             X24, X23, [SP,#0x50+var_30]
                STP             X22, X21, [SP,#0x50+var_20]
                STP             X20, X19, [SP,#0x50+var_10]
                STP             X29, X30, [SP,#0x50+var_s0]
                ADD             X29, SP, #0x50
                SUB             SP, SP, #3,LSL#12
                SUB             SP, SP, #0xA00
                MOV             X19, SP
                MOV             X21, X0
                ADD             X9, X19, #0x50 ; 'P'
                ADRP            X8, #stack_cookie@PAGE
                LDR             X8, [X8,#stack_cookie@PAGEOFF]
                LDR             X8, [X8]
                STR             X8, [X9]
                STR             X1, [X19,#0x3A50+var_3A08]
```

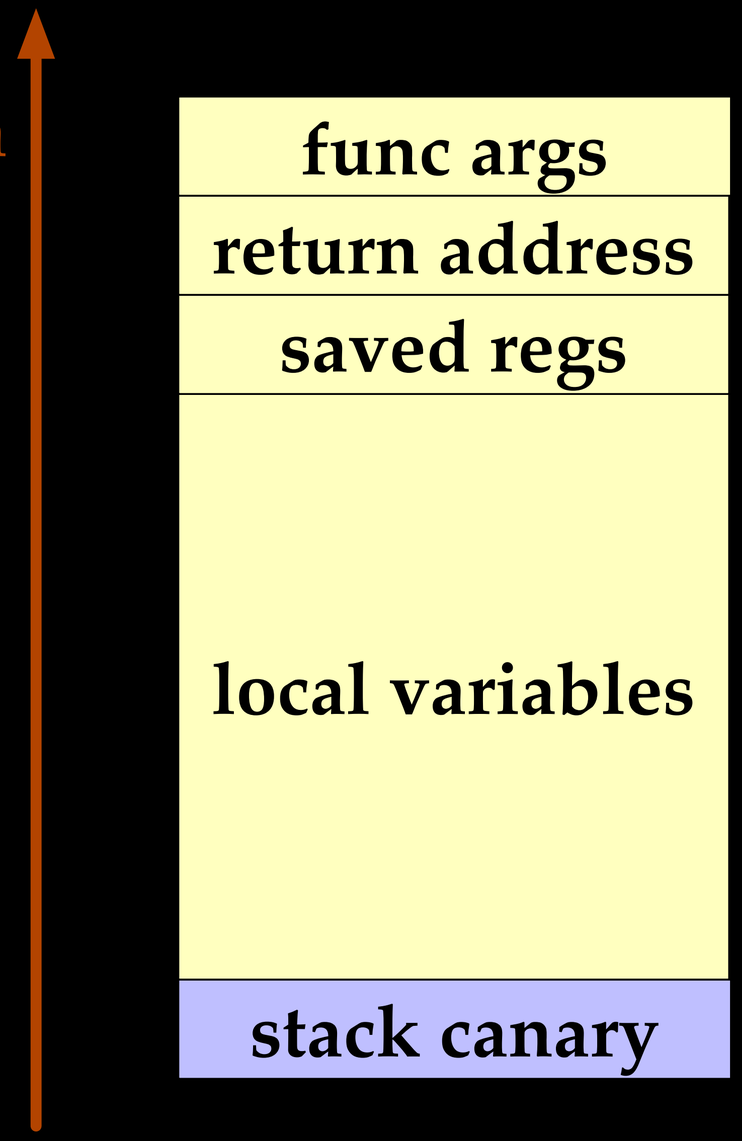




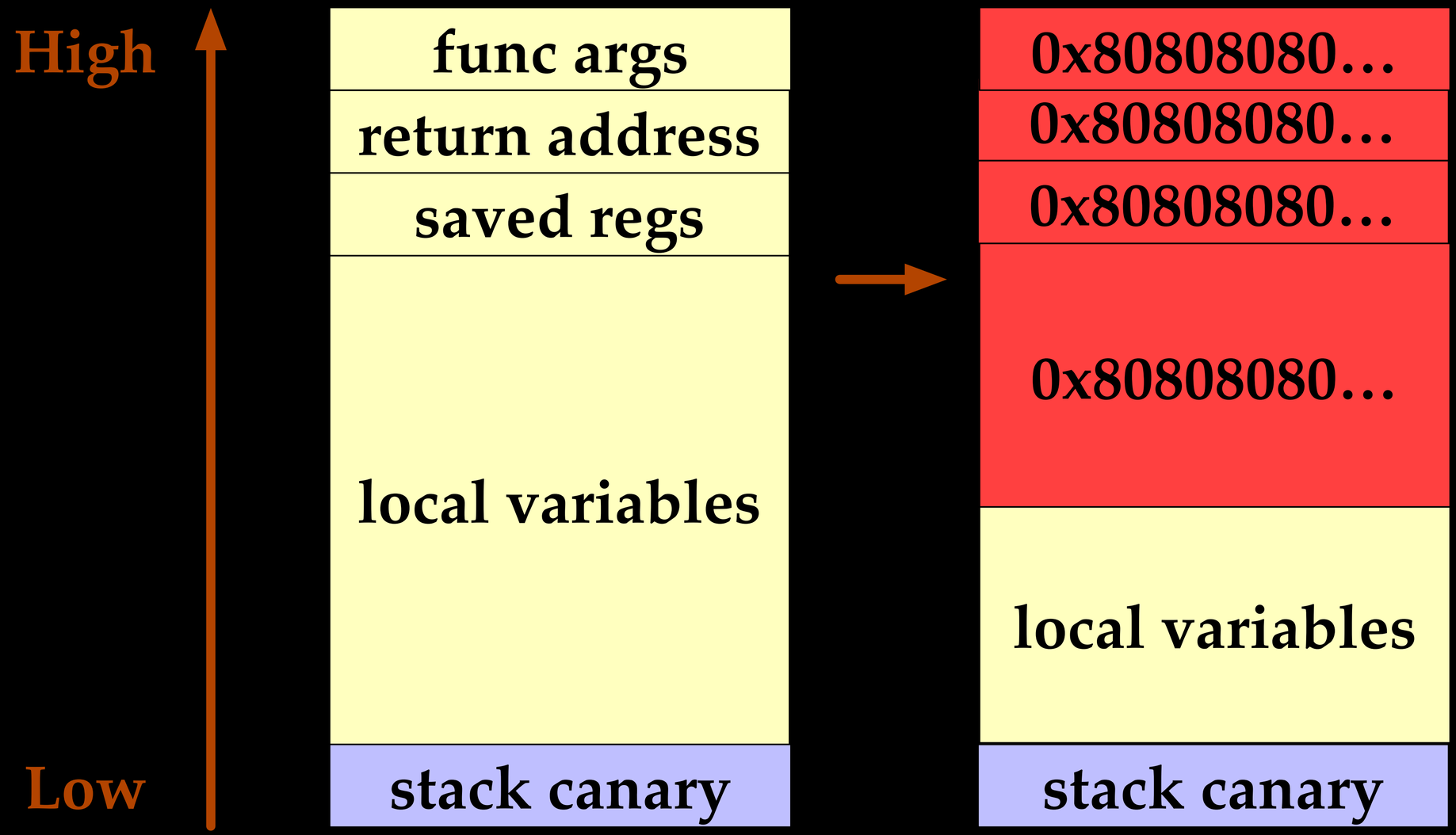
```
_AssembleFrame_H264                     ; CODE XREF: __text:0000000198A03A78↑p

var_3AB0        = -0x3AB0
var_3AA8        = -0x3AA8
var_3A98        = -0x3A98
var_3A88        = -0x3A88
var_3A80        = -0x3A80
var_3A78        = -0x3A78
var_3A70        = -0x3A70
var_3A68        = -0x3A68
var_3A60        = -0x3A60
var_3A08        = -0x3A08
anonymous_2     = -0x39F2
anonymous_1     = -0x39F0
anonymous_0     = -0x39E8
anonymous_3     = -0x39D0
anonymous_4     = -0x39C8
var_50          = -0x50
var_40          = -0x40
var_30          = -0x30
var_20          = -0x20
var_10          = -0x10
var_s0          =  0

                STP             X28, X27, [SP,#-0x10+var_50]!
                STP             X26, X25, [SP,#0x50+var_40]
                STP             X24, X23, [SP,#0x50+var_30]
                STP             X22, X21, [SP,#0x50+var_20]
                STP             X20, X19, [SP,#0x50+var_10]
                STP             X29, X30, [SP,#0x50+var_s0]
                ADD             X29, SP, #0x50
                SUB             SP, SP, #3,LSL#12
                SUB             SP, SP, #0xA00
                MOV             X19, SP
                MOV             X21, X0
                ADD             X9, X19, #0x50 ; 'P'
                ADRP            X8, #stack_cookie@PAGE
                LDR             X8, [X8,#stack_cookie@PAGEOFF]
                LDR             X8, [X8]
                STR             X8, [X9]
                STR             X1, [X19,#0x3A50+var_3A08]
```

High

Low



Stack overflow happened in _AssembleFrameH264



mediaserverd crash log on iOS

```
Thread 34 crashed with ARM Thread State (64-bit):
    x0: 0x0000000000000000   x1: 0x00000001702ea200   x2: 0xffffffffffffffff   x3: 0x0000000000000000
    x4: 0x000000000000000b   x5: 0x000000000000000b   x6: 0x0000000000000000   x7: 0x0000000000000000
    x8: 0x2d1b939d33fb0048   x9: 0x2d1b939d33fb0048  x10: 0x0000000000000000  x11: 0x000000000017a0c1
   x12: 0x00000001612c4000  x13: 0x0000000000003fff  x14: 0x000000000000f6c4  x15: 0x0000000000000001
   x16: 0x000000019956a230  x17: 0x0000000066855132  x18: 0x0000000000000000  x19: 0x8080808080808080
   x20: 0x8080808080808080  x21: 0x8080808080808080  x22: 0x8080808080808080  x23: 0x8080808080808080
   x24: 0x8080808080808080  x25: 0x8080808080808080  x26: 0x8080808080808080  x27: 0x8080808080808080
   x28: 0x8080808080808080   fp: 0x8080808080808080   lr: 0x8080808080808080
    sp: 0x00000001702edbf0   pc: 0x8080808080808080 cpsr: 0x60000000
```

```
Thread 34 Crashed:
0   ???                           0x8080808080808080 0 + -9187201950435737472
```

FaceTime

Available for: iPhone 5s and later, iPad Air and later, and iPod touch 6th generation and later

Impact: A remote attacker may be able to cause arbitrary code execution

Description: A memory corruption issue was addressed with improved input validation.

CVE-2019-8648: Tao Huang and Tielei Wang of Team Pangu

A compiler bug

- Many functions in different modules have incorrect stack cookies
- Apparently Apple’s compiler got a problem

LLVMs Arm stack protection feature can be rendered ineffective

Vulnerability Note VU#129209

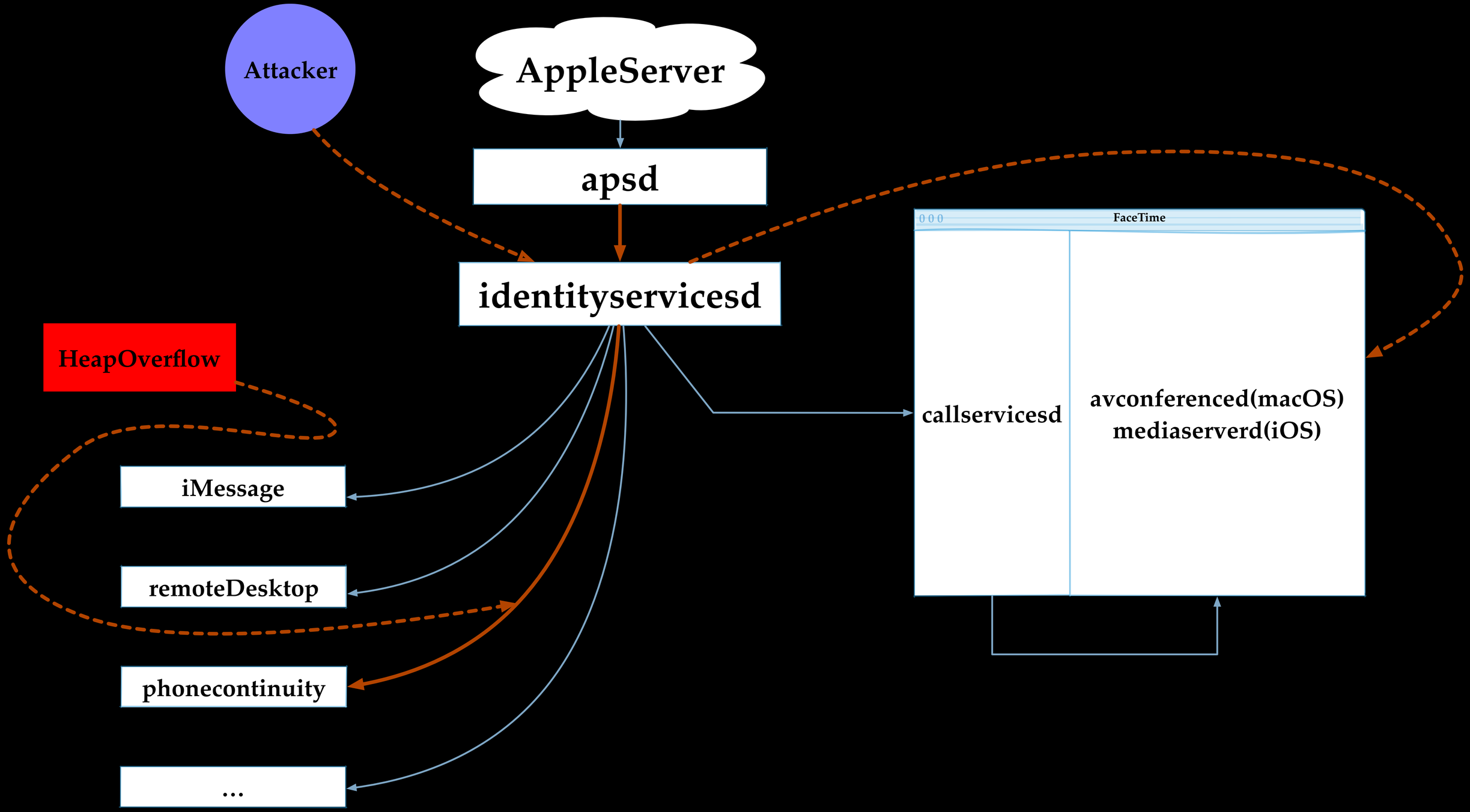
Original Release Date: 2019-07-15 | Last Revised: 2019-07-15

Overview

The stack protection feature in LLVM's Arm backend can be rendered ineffective when the stack protector slot is re-allocated so that is appears after the local variables that it is meant to protect, leaving the function potentially vulnerable to a stack-based buffer overflow.

Description

The stack protection feature provided in the LLVM Arm backend is an optional mitigating feature used to protect against buffer overflows. It works by adding a cookie value between local variables and the stack frame return address. The compiler stores this value in memory and checks the cookie with the `LocalStackSlotAllocation` function to ensure that it has not changed or been overwritten. If the value has changed, then the function will terminate. Since it currently pre-allocates the stack protector before the local variables in the stack, it's possible that a new stack protector can be allocated later in the process. If that happens, it leaves the stack protection ineffective as the new stack protector slot appears after the local variables that it is meant to protect. Additionally, it is also possible for the stack cookie pointer to spill to the stack and potentially be overwritten. This could happen in an area on the stack before the stack protector slot, rendering it ineffective.


Attacker
AppleServer
apsd
identityservicesd
HeapOverflow
iMessage
remoteDesktop
phonecontinuity
...
FaceTime
callservicesd
avconferenced(macOS)
mediaserverd(iOS)

Phone continuity

- Allows to make and receive calls from Mac, iPad, or iPod touch when those devices are on the same network as iPhone.
- Apparently this needs bidirectional network connections between Mac and iPhone
- Based on our manual inspection, we found the network packets are also first processed by identityservicesd

Following this execution path

[IDSLinkManager link:didReceivePacket:fromDeviceUniqueID:cbuuid:]

[IDSLinkManager _processLMCommandPacket:fromLink:deviceUniqueID:cbuuid:]

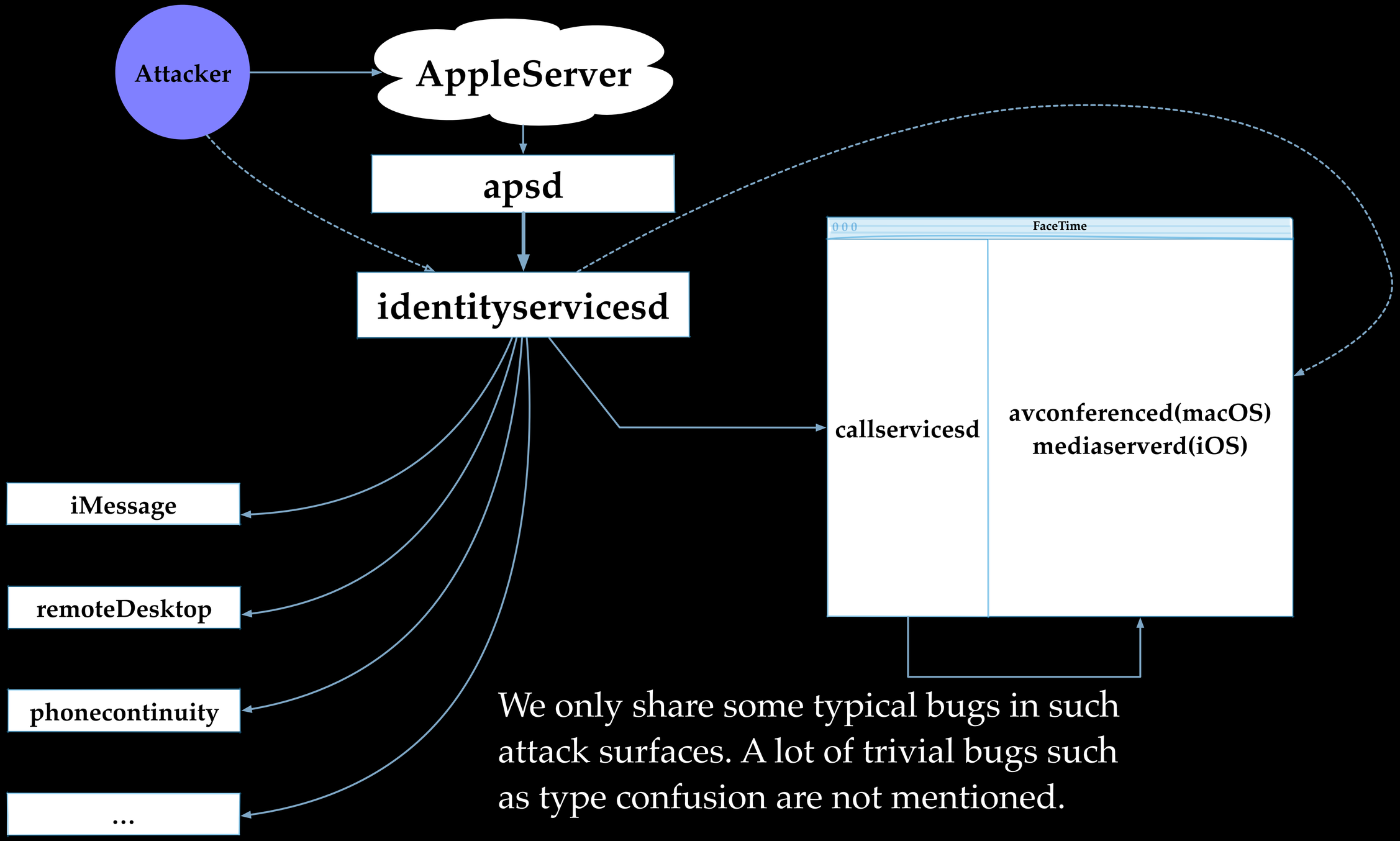
[IDSInterfaceAddress interfaceAddressWithTransmittedBytes:length:withLocalInterfaceName:]

[IDSInterfaceAddress initWithInterfaceAddress:bflags:bssid:bssidLength:]

[IDSSockAddrWrapper initWithSockAddr:]

```
IDSSockAddrWrapper *__cdecl -[IDSSockAddrWrapper initWithSockAddr:](IDSSockAddrWrapper *self, SEL a2, const sockaddr *a3)
{
  __int64 v3; // r14
  char *v4; // rbx
  char v5; // al
  IDSSockAddrWrapper *v6; // r14
  struct objc_super v8; // [rsp+0h] [rbp-20h]

  v3 = (__int64)a3;
  v8.receiver = self;
  v8.cls = &OBJC_CLASS___IDSSockAddrWrapper;
  v4 = (char *)objc_msgSendSuper2(&v8, "init");
  if ( v4 )
  {
    if ( !v3 || !*(_BYTE *)v3 || (v5 = *(_BYTE *)(v3 + 1), v5 != 0x1E) && v5 != 2 )
    {
      v6 = 0LL;
      goto LABEL_9;
    }
    memcpy(v4 + 8, (const void *)v3, *(unsigned __int8 *)v3);
  }
  v6 = objc_retain(v4);
LABEL_9:
  objc_release(v4);
  return v6;
}
```

- Very similar to Ian Beer's mp_socket kernel vulnerability(CVE-2018-4241)(https://bugs.chromium.org/p/project-zero/issues/detail?id=1558)
- sockaddr is under attacker's control, but the function has no checks on the length field of sockaddr (first byte)
- memcpy will lead to a heap overflow and overwrite ISA pointers of adjacent objects



We only share some typical bugs in such attack surfaces. A lot of trivial bugs such as type confusion are not mentioned.

Black Hat Sound Bytes

- FaceTime’s implementation needs significant improvements
- Attack surfaces exposed by messaging interfaces need more attentions
- Typical stack overflows may still affect iOS in 2019

Thank you!