UNITED STATES PATENT APPLICATION

For

NETWORK ACCESS VIRTUAL EXPERIENCE DEVICE ACCELERATION (NAVEDA)

INVENTOR:

JASON W. RUPE

Description

Access virtual device acceleration - for gamers or anyone running applications that do not perform well on their edge devices, enable an edge application to know the device, assess its capabilities toward a use case (based on past experience from this and potentially other users, or based on tests or manufacturers (hw, sw) recommendations or service level) and determine when it is lacking to enable cloud or neighbor compute capabilities to step in and improve the service.

Like when a multiplayer game assesses the edge users' capabilities including bandwidth and latency, add the ability to boost bandwidth and use a third party or access provider additional compute to improve the service beyond what is done in gaming today. Further, apply the same concept to other application uses such as higher definition video, digital signal processing of audio, or most any software application (like software as a service) which requires hardware beyond the means of edge equipment. Service as a service. The uniqueness here is an assessment of the needs and available resources for each potential use case associated with a user and their location and means of connection and hardware capability, versus options to enhance the experience at their disposal.

Proactively define settings.

1) Define the use cases possible, the hardware and software possible to use, the location of hardware and software able to be used, and likely locations and access network choices where they are likely to use the use case.

2) Sort the combinations of location, hardware, software, network, and use case from most likely to least until they are no longer likely enough to plan for (this can be set by policy such as total probability covered, probability of the combination, etc.).

3) Select the most likely combination of location and network and devices and software and use case from the list. Look up the performance requirements including but not limited to latency, jitter, loss, throughput, and reliability. This can be user specific per SLA as well, or just a baseline for the combination and service level.

4) Search the combinations of hardware, software, access network settings that will meet the requirements at that location for the use case. Select the solution from the list based on service provider policy (for example, lowest cost that still provides expected service success is most likely).

5) Identify recovery scenarios for each resources failure or unavailability, and set monitoring and automatic switch over for protection of the service to meet overall reliability goals. For example, the SLA is for reliable completion of the 2 hour session with this use case is 0.9999, but the selected solution only meets 0.9995 alone, so a backup solution is identified, and is made ready to take over quickly in the failure of the current solution.

6) Log solutions in a database that can be reliably referenced when needed.

7) Go to the next use case and repeat until stopping conditions met.

Reactively set service when proactive planning done (of at least a default solution).

1) Identify the requested use case, the hardware and software available to use, the location of hardware and software able to be used, and location and access network choices where they are.

2) Find the combination from the precompute list that matches.

3) If no exact match found, but one that is close and known to be worse than is currently possible, adjust the settings and set up the solution.

4) If no exact match is found but there is one known to be better than what is possible in the current situation, but informs the current choice, set up service to not make the user wait. Meanwhile search for protection solutions and redundancy options or added capabilities that enhance the experience closer to the solution. Start the next process to search for a better solution in case one is found, and switch the user over if possible.

5) Log the new knowledge (new use case conditions, new solutions, into known solution database. Update the probabilities too.

Reactively find a solution when no proactive planning done.

1) Identify the requested use case, the hardware and software available to use, the location of hardware and software able to be used, and location and access network choices where they are.

2) Conduct a search technique (say start with lowest performing options, highest performing options, etc., and trade toward the other direction, or run a multi-objective optimization routine such as knapsack) to find the best initial solution. Start the service with that solution.

3) Look at tradeoffs with backup capacity. Set up protection resources as needed to meet the requirements.

4) Search for solutions that may become available before end of use case. Schedule routines to watch for resources changes during the use case that may lower cost, improve use case experience, etc., according to the service provider policy and SLAs offered and needs of use case. Define the switchover routines and trigger to watch for switch to new settings during the use case mission time.

Log the new knowledge (new use case conditions, new solutions, into known solution database. Update the probabilities too.

Background

Users expect a great experience no matter where they are, no matter what device they use, no matter the conditions or situation. But their use conditions are not always ideal. A service can be defined to optimize the experience for users based on their situation, SLA, location, etc.

Abstract

access virtual device acceleration - for gamers or anyone running applications that do not perform well on their edge devices, enable an edge application to know the device, assess its capabilities toward a use case (based on past experience from this and potentially other users, or based on tests or manufacturers (hw, sw) recommendations or service level) and

determine when it is lacking to enable cloud or neighbor compute capabilities to step in and improve the service.

Like when a multiplayer game assesses the edge users' capabilities including bandwidth and latency, add the ability to boost bandwidth and use a third party or access provider additional compute to improve the service beyond what is done in gaming today. Further, apply the same concept to other application uses such as higher definition video, digital signal processing of audio, or most any software application (like software as a service) which requires hardware beyond the means of edge equipment. Service as a service. The uniqueness here is an assessment of the needs and available resources for each potential use case associated with a user and their location and means of connection and hardware capability, versus options to enhance the experience at their disposal.

Figure 1. Proactively define settings.

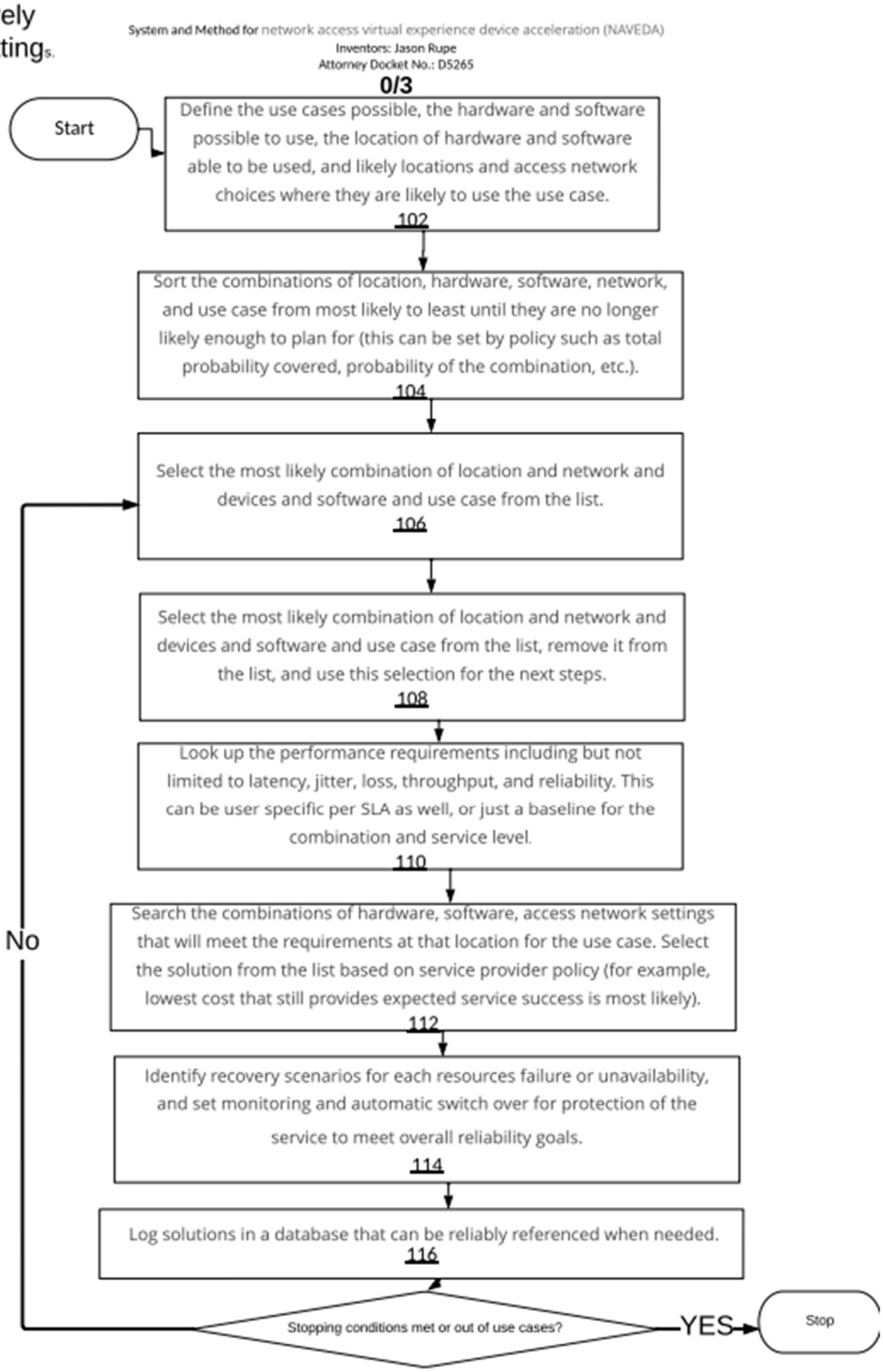

Figure 2. Reactively set service when proactive planning done (of at least a default solution).

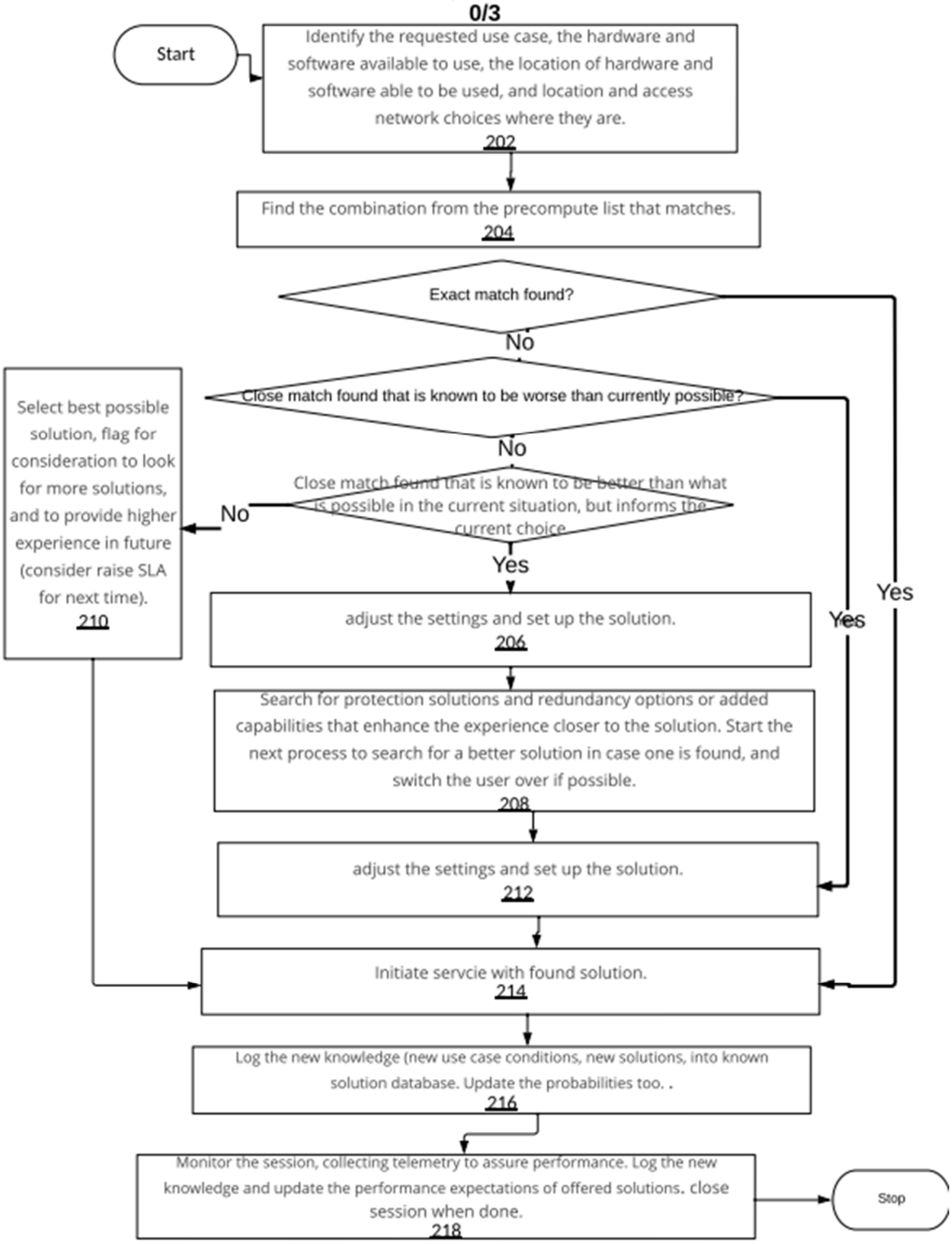

Figure 3. Reactively find a solution when no proactive planning done.