Computer Communications 35 (2012) 1472–1483

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Overload control in SIP networks using no explicit feedback: A window based approach

Seyed Vahid Azhari *, Maryam Homayouni, Hani Nemati, Javad Enayatizadeh, Ahmad Akbari

School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 2 August 2011
Received in revised form 20 February 2012
Accepted 14 April 2012
Available online 23 April 2012

Keywords:
SIP
Overload control
Window based
Fairness

ABSTRACT

The Session Initiation Protocol (SIP) has gained momentum and is being widely used both in the Internet and Next Generation Telecommunications networks as the core signaling protocol. SIP operation relies on SIP servers which are responsible for routing SIP messages. It has been shown that the performance of SIP servers is largely degraded during overload periods due to the built in message re-transmission mechanism of SIP. In this paper we propose a distributed and end-to-end adaptive window based overload control algorithm, which does not use explicit feedback from the downstream server. Upstream servers use call establishment delay as a measure of the amount of load on the downstream server. Therefore, the proposed algorithm imposes no additional complexity or processing on the downstream server which is overloaded, making it a very robust approach. Using simulations we show that our proposed method achieves higher throughput than a commonly used overload control algorithm and is also fair among different upstream servers under different network latencies. To the best of our knowledge, fairness under different network latencies has not been previously addressed in the context of SIP overload control. In addition, compared with approaches using explicit feedback, our scheme is less sensitive to network latency. The proposed overload control algorithm is also implemented in the OpenSIPS open source SIP proxy and shown to perform as expected under various conditions.

1. Introduction

Session Initiation Protocol (SIP) [1] is an application layer signalling protocol standardized by IETF and used for initiating, modifying and tearing down multimedia sessions. It is believed that SIP can facilitate creating services over Next Generation Networks (NGN). In fact SIP has been recently adopted by the 3rd Generation Partnership Project (3GPP) as the main signaling protocol of the IP Multimedia Subsystem (IMS).

Providing carrier grade service over SIP-enabled NGN, necessitates providing mechanisms for handling SIP traffic surges that overload SIP servers. A list of causes for overload is provided in RFC5390 [2] including, poor capacity planning, component failures, avalanche restart, flash crowds and denial of service attacks. The overload condition is exacerbated when SIP uses UDP and provides application layer reliability via re-transmitting all non-served requests increasing server load in a regenerative way [3–5]. Although using TCP, to some extent, improves server performance in overload, it poses scalability problems as pointed in [6,7]. By far, UDP is the most common choice of transport for SIP and we will assume that SIP is running on UDP throughout this paper.

In this paper we propose a novel distributed SIP overload control algorithm, which does not require explicit feedback. Our approach uses call setup delay measured by the upstream server as an indication of overload at some downstream server. Using this information, the upstream server controls the signaling load imposed on the downstream server using an adaptive window based approach, which is shown to achieve very high throughput and be fair. Our approach has virtually no impact on the overloaded server, in particular, the overloaded server does not have to deal with the burden of generating feedback and possibly even rejecting calls. Also, being window based it enjoys a self-clocking feature which has been shown to improve stability of Internet congestion control algorithms [8]. We simulate our proposed scheme under different SIP network topologies and conditions, including steady and dynamic load, non-zero network latency and packet loss. The proposed overload control algorithm is also implemented in the OpenSIPS open source SIP proxy [9] and shown to perform as expected.

The rest of this paper is organized as follows: Section 2 includes an overview of the SIP protocol, existing overload control methods and our contribution. Section 3 provides the details of the proposed overload control algorithm. In Section 4 we present our simulation

* Corresponding author.
E-mail addresses: azharivs@iust.ac.ir (S.V. Azhari), mhomayouni@comp.iust.ac.ir (M. Homayouni), nemati@comp.iust.ac.ir (H. Nemati), j_enayati@comp.iust.ac.ir (J. Enayatizadeh), akbari@iust.ac.ir (A. Akbari).

0140-3664/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2012.04.013

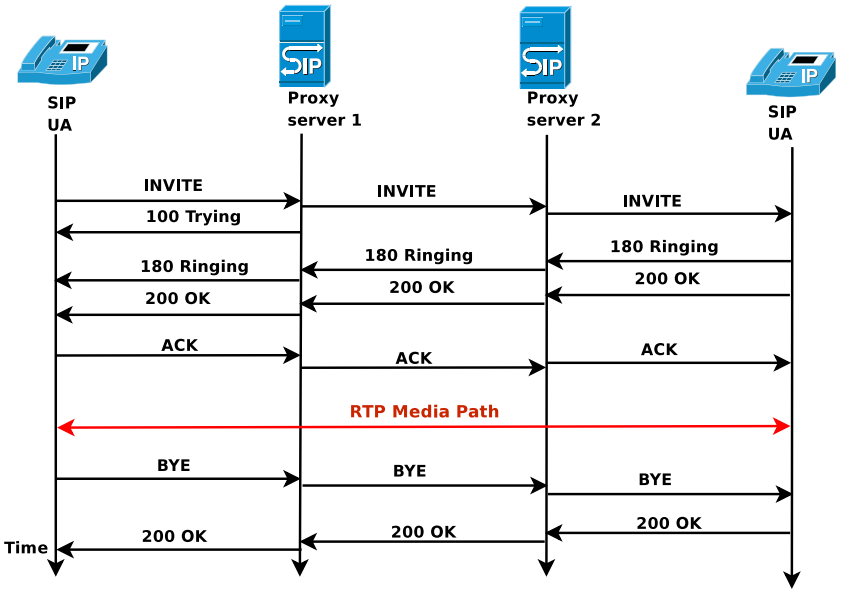

Fig. 1. Basic SIP call flow.

model and network topologies. Section 5 includes performance evaluation results obtained by simulation compared with two other well-known schemes. This section also considers the effect of network latency on throughput as well as fairness of the proposed scheme. Section 6 discusses some practical issues of employing our solution and includes experimental results obtained from our testbed. Finally Section 7 concludes the paper and outlines future work.

2. Background and contribution

2.1. SIP overview

Fig. 1 shows the typical SIP trapezoid topology and the standard SIP voice call signaling consisting of the INVITE-BYE message sequence. Setting up a session starts when the caller (User Agent Client: UAC) sends an "INVITE" request to the callee (User Agent Server: UAS), which is routed through SIP proxies in the path between them. The reception of this request in each proxy is confirmed by returning a "100 Trying" response to the previous hop in the path. Once the UAS receives the INVITE request, it sends back a "180 Ringing" response to the caller. It later also sends back a "200 OK" response when the call is accepted by the application in charge of taking the call. Finally to acknowledge receipt of "200 OK", an "ACK" request is sent to the callee. After this three way handshake, the media session is independently established between the two parties. The session is then terminated when one party sends a "BYE" request and the other responds with a "200 OK".

2.2. SIP overload problem

SIP uses its own reliability mechanism specially when used on top of an unreliable transport protocol, such as UDP, using a large set of re-transmission timers. For example, timer A is responsible for scheduling INVITE re-transmissions and starts with an initial value of typically $T_1 = 500$ ms and doubles whenever expires. SIP will stop re-transmitting and declare call failure after waiting $64 * T_1 = 32$ s. This mechanism is useful in case of having unreliable links, but is a major cause of performance degradation in overload conditions.

During overload, messages arriving at the overloaded server either get dropped or incur large delay. As a result, the UACs (and also possibly the upstream proxy) start re-transmitting unacknowledged messages. Furthermore, incoming responses from the UAS also experience loss or extensive delay before being processed by the server. This causes the server itself to re-transmit part of the requests it has already forwarded to the UAS. Therefore, the actual server load increases in a regenerative way leading to call failures. The dramatic decrease in server throughput is shown in Fig. 2 by the curve labeled "No-Ctrl". Here the SIP server can handle a maximum load of 200 calls per second (cps). When the load increases beyond this value the server becomes overloaded and congestion collapse occurs. Similar behavior have been reported in [3,4] and many others. We have also evaluated SIP server performance in a real test-bed and obtained similar results [5].

2.3. Related work on SIP overload control

The goal of SIP overload control (OC) is to keep server throughput close to its capacity in the presence of overload. The curve labeled "Theoretical" in Fig. 2 shows how an ideal OC scheme would work when server throughput is 200 cps. A system model for SIP overload control is introduced in [10] determining where the control loop components reside and the degree of cooperation between the SIP servers during overload condition. In general, there are two ways to control overload; local and distributed. In local overload control, the control loop is implemented internally on the overloaded server [10], therefore, a SIP server starts rejecting additional requests when getting close to its capacity limit. This is done by sending a 503 Service Unavailable message in response to an INVITE [1]. A number of local overload control methods differing mainly in the rejection policy and the overload detection criteria are proposed and evaluated in the literature.

Queue length based algorithms have been proposed in [3,11,4,12]. Occupancy based algorithms, namely OCC, using CPU utilization as a trigger for rejecting calls have also been proposed in [3,11] (Fig. 2 "Local-OCC"). In addition, the effect of priority queuing and transport protocol on SIP signaling performance is