A Client-Server Approach to Enhance Interactive Virtual Environments on Mobile Devices over Wireless Ad Hoc Networks *

Azzedine Boukerche, Richard Werner Nelem Pazzi and Tingxue Huang

SITE - University of Ottawa, Canada
PARADISE Research Laboratory
{boukerch, rwerner}@site.uottawa.ca

Abstract. Interactive 3D environments have been studied for years and represent an important application area of computer graphics. However, high quality virtual environment interaction requires powerful computers equipped with expensive graphics accelerator cards. The high 3D data volume and the dynamic nature of bandwidth pose significant challenges when providing a smooth virtual navigation on thin mobile devices over wireless ad hoc networks. In this paper, we show that it is possible to provide a virtual environment walkthrough on mobile devices through a client-server approach. Although mobile devices have low processing power and memory, they can still render images with relative ease. Based on this fact, instead of using traditional geometry-rendering techniques and locally rendering complex scenes, we employ an image-based mechanism on the client that uses images, which are provided by a remote server through an interactive streaming transport protocol. In this paper, we propose a bandwidth feedback algorithm together with a rate control and virtual user path prediction to better adapt the system to the changing bandwidth. We also discuss our ideas and show an extensive set of simulations in order to evaluate the performance of our solutions.

1 Introduction

The fast developments of computer graphics technologies such as fast geometry rendering algorithms and hardware implementation of graphics primitives, and the advances in communication networks and protocols have enabled the creation of a vast number of interesting applications related to navigating in a remote virtual environment, e.g. games, virtual tours, training, virtual shopping, etc. However, complex 3D models have been created for powerful computers, not for thin mobile devices such as cell phones and PDAs. In addition, downloading complex virtual environments requires both high bandwidth and storage capability. The high volume of 3D data and the dynamic nature of bandwidth pose significant challenges in terms of providing smooth virtual navigation on thin

* This work is partially sponsored by Grants from NSERC Canada Research Chair Program

mobile devices over wireless ad hoc networks: a mobile device can only hold a fraction of the entire virtual environment; the 3D rendering engine is not able to process complex scenes in real-time, the bandwidth is always changing and because the wireless communication channel is highly susceptible to error. A possible approach to these problems is rendering the complex 3D geometry on a graphics workstation server and transmitting only images to the remote client, depending on the user's position within the virtual environment. Instead of employing the traditional 3D geometry rendering mechanism on the client side, the approach can make use of an inexpensive image-based rendering (IBR) technique. IBR uses images as input and the rendering cost does not depend on the scene complexity, but on the final image resolution. As mobile devices usually provide small displays, IBR methods perfectly fit on these types of devices as the image size and therefore the bandwidth required to transmit the images will be small. Lately, there has been a great deal of interest in IBR algorithms lately. For instance, the view morphing [1] technique, which requires low processing as it renders novel views based on a collection of sample images. IBR algorithms are based on the plenoptic function [2]. View morphing is the simplest IBR algorithm as it relies on a certain amount of geometric information about the scene, whereas lumigraph [3] and lightfield [4] use implicit geometry or no geometry at all; this requires more processing. Basically, as the user walks through the 3D environment, the client device sends its position and orientation to the server, which will update the virtual camera position, render a reference image, and send this image back to the client. The client can use certain reference images to render novel views through the IBR while it is waiting to receive new reference images from the server.

Providing a less expensive rendering technique to the client device is not sufficient to solve all of the problems related to remote interaction on mobile devices. In order to cope with the dynamic bandwidth, efficient transport protocols, which take into consideration the user's behavior in a virtual environment, must be developed. In this paper, we propose a client/server architecture to enhance the user experience in a remote virtual environment through a hybrid rendering system, which uses traditional 3D geometry rendering on the server and an IBR method, such as view morphing [1], on the client. In order to improve the frame rate, or the number of images that the device can display per second, a virtual user path prediction algorithm is proposed, allowing the server to pre-fetch certain images to the client when enough bandwidth is available. The bandwidth feedback mechanism and rate control are designed to optimize the pre-fetching scheme, as its goal is to avoid starvation of images at the client side. The interactive streaming protocol is designed over the Real-Time Protocol (RTP) [5] and the Real-Time Streaming Protocol (RTSP) [6], which provide end-to-end delivery services for data with real-time constraints. For instance, audio and video.

This paper is organized as follows: Section 2 gives an overview of related work. Section 3 presents the proposed system architecture. The algorithms are described in Section 4. Simulation experiment results are shown and discussed

in Section 5. Finally, in Section 6 the reader can find our conclusions and future work.

2 Related Work

In this Section we discuss some of the existing 3D rendering mechanisms on mobile devices. We also give the reader a brief review of image-based rendering and provide a discussion about interesting similar solutions to remote virtual environments.

OpenGL ES API [7] is used by several 3D applications on mobile and embedded devices. However, the rendering quality is still poor, or has a very low level of detail. The Mobile 3D Graphics API (M3G), defined in Java Specification Request (JSR 184) [8], is another industry effort to create a standard 3D API for Java-enabled thin devices. A solution to visualize more complex 3D scenes on mobile devices is made possible through Image-Based Rendering (IBR). IBR methods are categorized based on the geometry information they require to render novel views. Some image-based rendering techniques do not require geometric information. For instance, Lightfield [4] renders a new view by interpolating a set of samples without any geometric information such as a depth map. The problem with IBR methods that do not rely on geometric information is the huge storage capacity required to hold all of the pre-acquired image samples.

Our remote interactive system is based on View Morphing [1], which is able to render any novel image by morphing two or more reference images. The basic principle is depicted in Figure 1. Morphing parallel views is the simplest image morphing algorithm. As depicted in Figure 1, images I_0 and I_1 are acquired at points C_0 and C_1 respectively, with focal lengths f_0 and f_1. Novel image I_n, with focal length f_n, at point C_n, is rendered by the interpolation of images I_0 and I_1. There is also an image cache on the client in order to reuse a previously received image, significantly improving system performance and reducing network traffic.

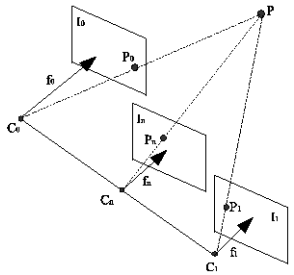


Fig. 1. View morphing with parallel views

QuickTime VR [9] is the most popular image-based rendering system. However, it is limited to panoramic scenarios, and the client device must download

the entire environment in order to start the navigation. Our solution offers a higher level of freedom, as a user can walk through the environment and there is no need to download the entire environment as it is being rendered while the user moves through different areas.

A client-server approach to image-based rendering on mobile terminals is presented in [10]. The objective of this solution is to make it possible to render complex scenes on mobile devices through an IBR method and a client/server architecture. However, its main contribution concerns how to place the cameras in order to avoid problems such as exposure and occlusion when using IBR. Thus, the camera placement solution works only for urban scenes, thereby limiting the applications of this solution.

The work presented in [11] aims at the protection of copyrighted 3D models when manipulated by remote users. The server owns the entire 3D environment and sends certain images to the client on demand. The client renders a low-polygon model of the scene as the user manipulates it. When the user stops, the client sends a request to the server, which will send back a high-resolution image of the scene. This approach is different from ours as the client is capable of 3D geometry rendering. This feature is not well suited to low-capacity devices such as PDAs. The reader can refer to [12–14] for detailed information on other solutions to remote virtual environments using IBR methods.

Unlike the presented solutions, we propose a system that can make better use of the available bandwidth, which is crucial to applications involving wireless communication and thin devices. To the best of our knowledge, the solutions found in the literature do not address the problem of dynamic bandwidth. Our approach uses a virtual user's path prediction together with bandwidth monitoring and rate control algorithms to adapt the protocol and pre-fetch images to the client when bandwidth permits.

3 The Proposed Streaming System

Our proposed system is organized in the following modules: a modified JPEG codestream, new RTP payload format for view morphing, streaming protocol, bandwidth feedback mechanism, rate control scheme, and path prediction and pre-fetching algorithms.

3.1 The Packetization and Streaming Schemes

We specified a new JPEG codestream to cope with wireless channel errors. Figure 2 shows this new JPEG codestream. A packetization scheme was developed to avoid the errors that a corrupted packet propagate to other packets; it keeps the packetization items independent from one another. A packetization item is an atomic component such as the main header, the layer header, or a pixel block. The image codestream is split into packetization items and is encapsulated in RTP packets, and is then sent to the client.

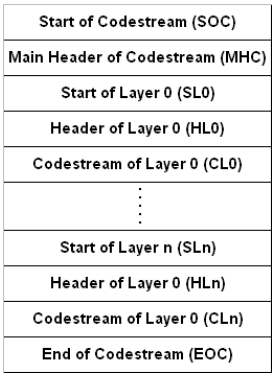
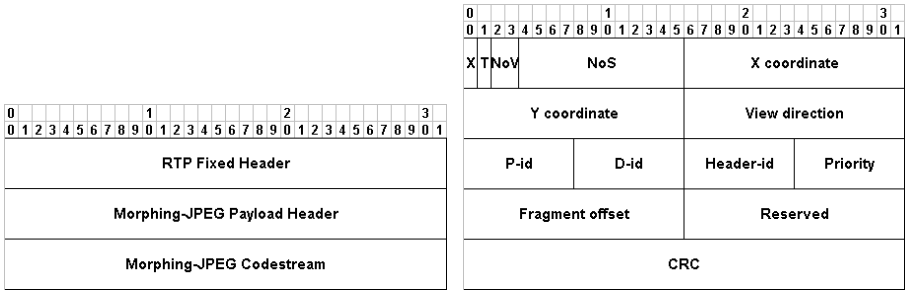


Fig. 2. Structure of the new JPEG codestream

Streaming multimedia compressed data over wired or wireless networks over RTP requires new payload formats such as H.26x over RTP [15, 16] and G.7xx over RTP [17, 18]. We introduced a new payload format for view morphing over RTP, as well as a new packet header for view morphing, which are depicted in Figures 3(a) and 3(b) respectively.



(a) (b)

Fig. 3. (a) Structure of a RTP packet for View Morphing. (b) View Morphing payload header.

The modified RTP fields are shown below.

Payload Type: according to the RTP standard, this field specifies the format of the RTP payload and determines its interpretation by the application [17]. Because our payload type is not specified by the RTP profile, the payload type for the View Morphing codestream is not assigned through RTP means; the upper layer defines the payload code;

Number of Video Unit **(NVU)**: If the current packet is the first fragment of a video unit or if it is the whole video unit, then check if this is the first packet

for that session. If it is the first packet, the field NVU and timestamp in the RTP header receive a random value; this will be the first value of a sequence number.

Timestamp: The View Morphing stream has no strict sampling instance. Unlike other media types, the timestamp for View Morphing does not indicate the sampling instance. Nevertheless, it is significant for calculating synchronization and jitter when other media streams are associated with View Morphing. All RTP packets for the same video unit will set the same timestamp. The packet header fields for view morphing over RTP are described below:

The payload header extension **(X)** is a bit flag that is activated when supplementary information follos the payload header. The twin images **(T)** bit field is set to inform the renderer to process two images; otherwise the rendering algorithm can render the image directly. The number of images in a video unit **(NoV)** field informs the renderer whether the packet contains zero, one, or two images. The number of an image in the streaming **(NoS)** field works as a sequence number to make sure that all image fragments were received by the client. The **(X,Y)** fields are the viewpoint coordinates and **(VD)** is the view direction angle. Viewpoint and view direction identification **(P-id and D-id)** help identify the viewpoint and view direction when data is corrupted or lost. The **(header-id)** field helps recover the main header when data is corrupted. The field **(Priority)** indicates the layer priority when sending different quality layers. The fragment **offset** field is used to reassemble the codestream. We also have the **reserved** field for future use. Finally, the **CRC** field detects whether or not the payload header is corrupted, which part is corrupted and tries to correct certain bits. Basically, the assembly of an RTP packet begins with the assignment of the payload type field for the Morphing-JPEG. If the current packet is the first fragment of a video unit or if it is the entire video unit, then check if this is the first packet for that session. If it is the first packet, the field NVU and timestamp in the RTP header receive a random value; this will be the first value of a sequence number. If it is not the first packet, NVU is incremented by one. All other fields are set according to the specification. For instance, field X is set to 1 if an optional payload follos the payload header; the field NoV is set according to the images the client will have to process; for instance, it will be set to 00 so as to instruct the client to process the first image on that video unit, 10 for the second, and 11 for the final one.

On the client side, the algorithm is a simple parser for the server codestream. First it checks if the payload type field is set to Morphing-JPEG, then checks the NVU field to see if it is different from the last one. If it is different, it instructs the application layer to render the image at that moment. The algorithm then gets the timestamp to calculate the synchronization and jitter. If field X is set to 1, the algorithm will locate the optional payload header, parse the codestream, and send it to the application. For the CRC scheme, the algorithm verifies if the viewpoint and direction are correct. If they are correct, the algorithm proceeds by checking the priority and the remaining fields. If the viewpoint and direction are not correct, algorithm approximate values from P-id and D-id, and runs the

CRC on them. The final step is to check the offset field. If the offset is equal to the last offset plus the length of the packetization data, then it merely appends the packetization data to the previous one. If this is not the case, the algorithm waits for a short timeout period. If the delayed packet does not arrive during this period, the algorithm informs the server that an RTP packet was lost. This will adjust the parameters according to the bandwidth feedback mechanism.

3.2 The Pre-fetching Mechanism

Based on the path prediction mechanism, the server will pre-fetch certain images to the client. The pre-fetching algorithm takes advantage of available bandwidth to send images in advance to the client, saving some requests and network traffic, and improving the image quality and perhaps the frame rate on the client because the reference images will be available in the client's local cache. The periodic transmission of control packets conducted by RTCP is enough to control the adaptive encodings and the speed of data distribution in wireless network scenarios. Our proposed bandwidth feedback mechanism involves sending ACKs for every received RTP packet. The server can establish the network status by keeping track of these ACKs. An ACK packet is composed of its type, a sequence number for ordering, and a timestamp. We use the timestamp to calculate the round trip time (RTT) of RTP packets. Missing acknowledgments are interpreted as dropped RTP packets. When the server receives an ACK, it will parse the RTP packet and extract the sequence number and timestamp. Then, the server adds the sequence number to a list of successfully transmitted packets. The server then calculates the RTT and adds it to the list of recently transmitted packets. For each recently transmitted packet, if the sequence number does not exist in the list of successfully transmitted packets, and if the RTT of the previous packet minus the RTT of the next packet is greater than the acceptable variation value, the number of packets lost is incremented by one. If the number of packets lost is greater than 0, the network status is set to congested. Otherwise, if the RTT of the last received packet minus the RTT of the first received packet is greater than the threshold, the status is congested. If the RTT of the first received packet minus the RTT of the last received packet is greater than a threshold, the status is unloaded. Otherwise network status is constant.

The path prediction mechanism is based on the virtual user's previous and recent movement within the environment. There are two navigation modes: linear and rotational. For instance, if the user is moving along a straight line, the path prediction can determine that based on his/her previous movements, the user will continue along the same path. In the case of rotation within the virtual environment, the path prediction will obtain the nearby positions based on a threshold angle.

Our rate control mechanism is based on the reports from the bandwidth feedback mechanism. If the network status is congested, the rate is decreased. If the status is unloaded, the rate is increased; and if the status is constant, the rate is left unchanged. The increment value is crucial for the performance of our protocol. Upon receiving a request, the server will determine the network

status. If the network is unloaded or congested, the rate is increased or decreased respectively by an amount corresponding to one image.

4 Simulation Experiment Results

We have implemented and simulated our approach on the NS-2 [19] network simulator. We performed a set of simulations in different ad hoc network scenarios. They consist of one server and 15 to 60 mobile nodes moving at 5m/s to 20m/s in an area of 500x500 m^2. An 802.11b MAC layer was utilized during the simulations. We used the following metrics to evaluate our proposed interactive streaming system: system throughput, end-to-end delay, burst length and burst duration. The average number of images in the cache versus the number of requests was used to evaluate the performance of the pre-fetching algorithm.

As depicted in Figure 4, when only one client is connected to the server, the amount of images in the clients cache is 315 for only 150 requests. The rate control was aware of the bandwidth status, and the server could utilize the available bandwidth to pre-fetch additional reference images. With the pre-fetching feature turned off, the number of images in the clients cache would not exceed 150 (1 image per request). As we increased the number of clients, the bandwidth dropped. The rate control mechanism was able to adjust to the new scenario and reduce the streaming rate. For instance, the server sent 181 images for 150 requests.

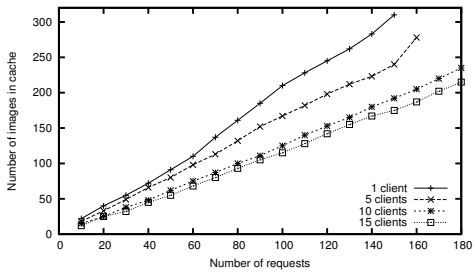


Fig. 4. Pre-fetching mechanism performance

The number of successfully sent packets per second is represented by the system throughput. As depicted in Figure 5(a), the system shows a reasonable scalability. Increasing the number of nodes leads the number of packets per second to increase from 25 to 45.

End-to-end delay is critical to our interactive system, because the system relies on quick response times. As can be seen in Figure 5(b), delay is less than 10ms when 35 nodes are employed and increases along with node density because the number of nodes in a path will most likely be higher.

Burst is a temporary connection lost that occurs when packets cannot reach the destination due to broken paths. Burst length is the number of packets that are dropped during a burst duration. As can be seen in Figures 5(c) and

5(d), burst length and duration depend on node density. With higher densities, routing paths are quickly restored. When a burst occurs, the bandwidth feedback mechanism is aware of the network status and immediately decreases the packet rate. Thus, the client renders novel views based on reference images stored on its cache, with a significant image quality depreciation, until it receives new reference images.

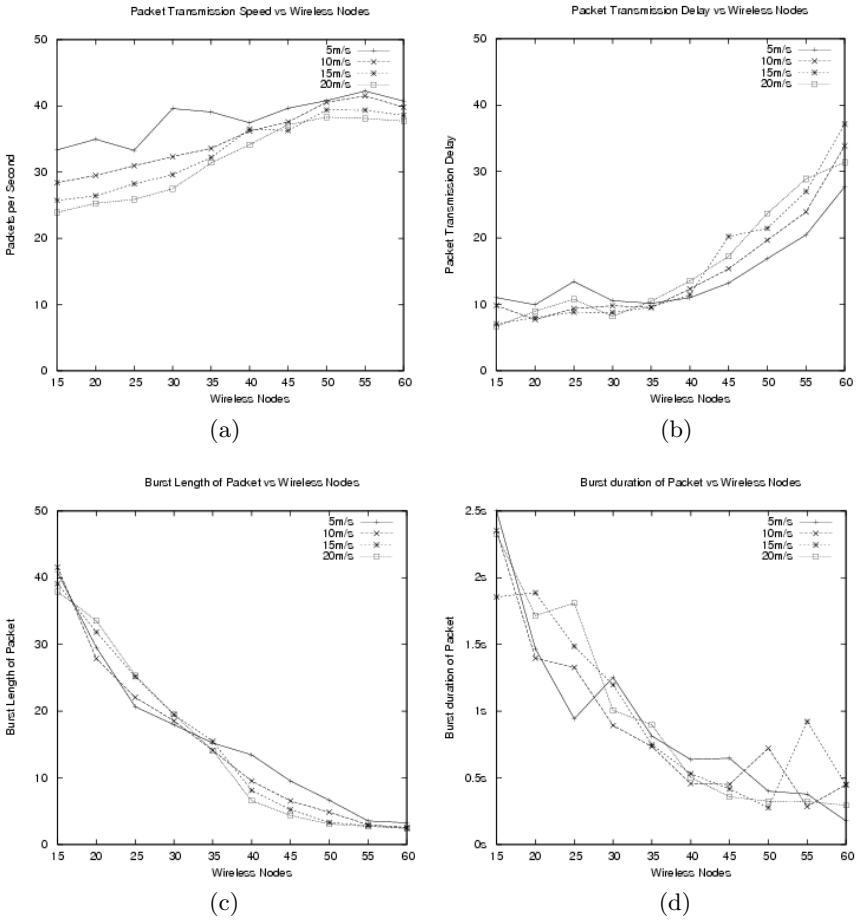


Fig. 5. (a) System throughput. (b) End-to-end delay. (c) Burst length. (d) Burst duration.

5 Conclusions and Future Work

In this paper, we proposed a client/server approach, which consists of a new payload format for RTP, an interactive streaming algorithm, the packetization scheme, and the pre-fetching mechanism, for remote interaction in virtual 3D environments for mobile devices over ad hoc networks. We addressed some issues of remote interactive virtual environments on mobile devices and focused on optimizing bandwidth usage and maximizing user experience. To the best of our knowledge, the related work did not tackle the dynamic bandwidth issue of mobile ad hoc networks. Pre-fetching images when there is available bandwidth has proven to be beneficial to the rendering system because the client does not need to request and wait for the images, which implies long delays. The simulation experiments demonstrated satisfactory performance results. As future work, the virtual user path prediction will be improved using a probabilistic virtual path prediction so as to optimize the tradeoff between sending reference images and the real use of them at the client side. We are also working on system prototype in order to evaluate the proposed algorithm.

References

1. S. M. Seitz and C. M. Dyer. View morphing. In Computer Graphics Proceedings, Annual Conference Series, pages 2130, Proc. SIGGRAPH96 (New Orleans), August 1996. ACM SIGGRAPH.
2. E. H. Adelson and J. Bergen. The plenoptic function and the elements of early vision. In Computational Models of Visual Processing, pages 320. MIT Press, Cambridge, MA, 1991.
3. S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In Computer Graphics Proceedings, Annual Conference Series, pages 4354, Proc. SIGGRAPH96 (New Orleans), August 1996. ACM SIGGRAPH.
4. M. Levoy and P. Hanrahan. Light field rendering. In Computer Graphics Proceedings, Annual Conference Series, pages 3142, Proc. SIGGRAPH96 (New Orleans), August 1996. ACM SIGGRAPH.
5. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. Standards Track, Network Working Group, January 1996.
6. H. Schulzrinne, A. Rao, R. Lanphier, M. Westerlund, and A. Narasimhan. Real time streaming protocol (rtsp). Inernet Draft, Internet Engineering Task Force, February 2004.
7. OpenGL Embeded System. http://www.khronos.org/opengles/ 2006
8. Java Specification Request 184 (2005) - Mobile 3D Graphics API for J2ME http://www.jcp.org/en/jsr/detail?id=184
9. S. E. Chen. QuickTimeVR an image-based approach to virtual environment navigation. Computer Graphics (SIGGRAPH95), pages 2938, August 1995.
10. G. Thomas, G. Point, and K. Bouatouch. A client-server approach to image-based rendering on mobile terminals. Technical Report, ISSN 0249-6399, France, January 2005.
11. Koller, D., Turitzin, M., Levoy, M., Tarini, M., Croccia, G., Cignoni, P., and Scopigno, R. 2004. Protected interactive 3D graphics via remote rendering. ACM Trans. Graph. 23, 3 (Aug. 2004), 695-703.
12. Biermann, H., Hertzmann, A., Meyer, J., Perlin, K., Stateless Remote Environment Navigation with View Compression, NYU Technical Report 1999-784. April 22, 1999.
13. Chang, C. and Ger, S. 2002. Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering. In Proceedings of the Third IEEE Pacific Rim Conference on Multimedia: Advances in Multimedia information Processing (December 16 - 18, 2002). Y. Chen, L. Chang, and C. Hsu, Eds. Lecture Notes In Computer Science, vol. 2532. Springer-Verlag, London, 1105-1111.
14. J. Li, Y. Tong, Y. Wang, H.-Y. Shum, Y.-Q. Zhang, "Image-based Walkthrough over the Internet", International Workshop on Very Low Bitrate Video Coding (VLBV01), October 2001, Athens, Greece.
15. T. Turletti and C. Huitema. Rfc2032: Rtp payload format for h.261 video streams. Stardards Track, Network Working Group, Octorber 1996.
16. C. Zhu. Rfc2190: Rtp payload format for h.263 video streams. Standards Track, Network Working Group, September 1997.

17. H. Schulzrinne and S. Petrack. Rfc2833: Rtp payload for dtmf digits, telephony tones and telephony signals. Standards Track, Network Working Group, May 2000.
18. R. Zopf. Rfc3389: Real-time transport protocol (rtp) payload for comfort noise (cn). Standards Track, Network Working Group, September 2002.
19. The Network Simulator. http://www.isi.edu/nsnam/ns/