Multicast Source Authentication for Limited Devices

JANNE LUNDBERG, CATHARINA CANDOLIN, HANNU KARI
Laboratory for Theoretical Computer Science
Helsinki University of Technology
P.O. Box 5400, 02015 HUT
FINLAND
{jlu, candolin, hhk}@tcs.hut.fi

Abstract: - Multicast has potential to make delivery of large files and streams over the Internet very efficient. However, strong source authentication for multicast is needed to make delivery of important data possible. Information such as stock quotes is valuable only if the data can be trusted. In this paper, we present an architecture which allows wireless mobile devices with limited computational abilities to receive source authenticated multicast over an unreliable wireless access network.

Key-Words: - multicast, wireless, caching, source authentication, optimization, content delivery

1 Introduction

The problem in delivering source authenticated multicast data to wireless mobile devices is twofold. First, as the connectivity to the fixed network can be frequently lost in a sparce network, the multicast data cannot be always received immediately when it is sent. This problem can be alleviated using multicast caches which temporarily store packets. The multicast cache can then be requested to retransmit the lost packets locally. The second problem is in the computational capabilities of many mobile devices. If the source authentication algorithm requires heavy computations, such as digital signature verifications, the capabilities of small devices may be exceeded. Some optimization which allows mobile devices to easily verify digital signatures is needed.

In this paper, we introduce an architecture which allows a wireless device with limited capabilities to receive source authenticated multicast data over an unreliable wireless network. The optimization intriduced here is an addition to an existing architecture for delivering multicast data to a large group of receivers.

The rest of this paper is organized as follows. In Section 2 we present some relevant technologies and problems which are typical to multicast. Section 3 presents our solution, and Section 4 discusses the properties of our solution. Finally, Section 5 concludes the paper.

2 Background

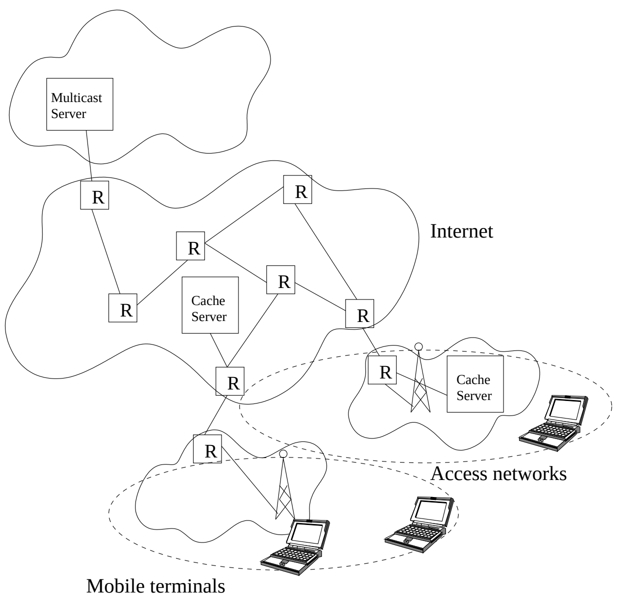

Figure 1: High-level view of the architecture

Figure 1 illustrates the environment where the architecture described in this paper is assumed to operate. The environment consists of one or more multicast servers which send content over the Internet to a number of mobile terminals which connect to the fixed network using wireless networks. In the figure, the boxed R-letters in the figure are routers which connect the network together. The towers with dashed ellipses are wireless access points with the range of their radio technology shown. The cache servers are points of temporary storage where content can be stored as long as the mobile terminals may request retransmissions of lost packets. The mobile terminals can freely move between the access networks. The mobile terminals are also expected to frequently move to an area without wireless network coverage. When a mobile terminal returns within network coverage, it can request the retransmission of the packet that were lost while the terminal was outside network coverage.

Next, we give an overview of the architecture at a high level. Further details of the architecture are omitted from this paper. The details can be found in [3], [1], and [2].

2.1 Architecture

Communication between the components of the system are illustrated in Figure 2. The system consists of three main components. The components which are located at the content provider network are responsible for encoding the data into a format that enables it to be delivered via multicast caches. The components in the access networks are responsible for storing the data while the data is considered to be interesting to a large enough number of receivers.

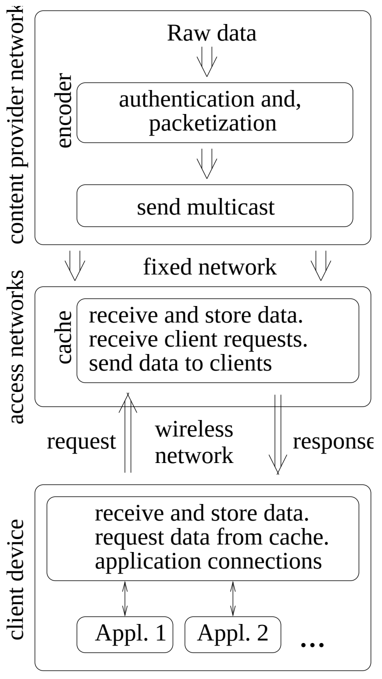

Figure 2: Communication between components.

The components in the client devices are responsible for requesting and storing data that has been received from the multicast caches.

2.1.1 Encoder

The encoder is located within the multicast server, and its task is to divide the information into packets that can be sent to the network. The encoder takes either a data file or a data stream as input. The encoder divides the data into packets that are small enough to be transmitted over the network without requiring fragmentation. Next, each packet is given a globally unique packet identity and possibly encrypted. The packet identity of each packet consists of a sequence number and the source and destination addresses of the multicast group to which the data will be sent. Finally, the encoder adds the source authentication data into to each packet using the technique described in [5].

After the data has been processed by the encoder, it is given to the multicast server in the form of packets that are ready to be sent to the network. The multicast server transmits the packets at a predefined rate, and the server can also use any one of the reliable multicast transport techniques described in Section **??** to guarantee that packet are received correctly by caches.

2.1.2 Multicast Cache

When a multicast cache receives a packet, it first verifies that the received packet belongs to a stream that the cache is configured to listen to. The stream is identified using the source and destination addresses from the packet. If the packet does not belong to a storable stream it is discarded. A packet that has passed the stream identification check is next authenticated to guarantee that the packet was sent by the multicast server that owns the address in the source address field of the received packet. A packet that passes the authentication, is then stored to the memory of the multicast cache. It is important to note, that the packet must be stored on disk in the exact format in which it was received in the network, including the header that contains its sequence number and authentication information. In particular, the cache does not need to decrypt the data even if the payload in the packet is encrypted. That is, caches are not required to have decryption keys to the data that they store.

The task of the multicast cache is to respond to request messages sent by other multicast caches and local host caches in client devices. A request message contains the the identities of the requested packets, that is, the source and destination addresses of the packets, and the sequence numbers of the packets that are needed. When a cache receives a request it sends the requested packets to the requester.

The replacement policy that is used by a cache can be configured individually for each data stream, and depends on the task the cache is performing to the given stream. If the cache is acting only as a device that is used to repair damaged multicast packet streams, it can start to delete packets that are no more than a few minutes older, or it can store data for periods of arbitrary length.

The mode of operation where data is stored for only a short time is useful, for example, when a server is sending a multicast stream containing a newspaper in PDF format and the client device is in a likely situation of experiencing brief network outages during the transmission. If the PDF newspaper is not received in its entirety, the rest of the data also is useless. If a client device experiences a network outage during the transmission, it can request the lost packets from the closest cache when it regains connection to the network.

The mode of operation where data is stored for longer periods of time can be used to cache, for example, popular movies or TV-shows, that are requested often and would require large resources from the fixed network to download from the originating server.

2.2 Mobile Terminal

The mobile terminal is the device which users use to consume content created by the content providers. The basic network connectivity requirements for mobile terminals are very small. The only connectivity requirement for a mobile terminal is the ability to receive data over one wireless network interface, which may even be unidirectional. The abilities of such a device is, however, very limited. A terminal with only unidirectional connectivity cannot be used to request data from the network. The terminal can only be used to receive data that is already being sent in an area covered by the wireless link technology used by the device. Such a device is very similar to a traditional radio or television. The user can choose which transmitted data is received but the user cannot request data to be transmitted.

A more advanced mobile terminal also has the ability to send data over a wireless channel. The channel can be used for requesting data, which might not otherwise be sent, from the network. The mobile node can request retransmission of data that was lost when it was transmitted earlier. It can also request content which would otherwise not be transmitted in the area where the mobile node is currently located.

Another advanced feature in a mobile terminal is multiple radio technologies. A mobile terminal which can utilize more than one radio technology has the additional option of selecting the best possible service for different situations. Switching between technologies can help the mobile terminal optimize between variables such as cost and capacity.

2.3 Multicast Source Authentication

Source authentication algorithms which are used in unicast communication cannot automatically be used for source authenticating multicast communication. Such algorithms are usually based on the assumption that only the two communicating parties have access to a shared secret. When one of the parties uses the shared secret to authenticate data, the other party uses the same shared secret to verify the authentication. If nodes A and B authenticate their communication using key k, A can verify that the data actually came from B by verifying a Message Authentication Code (MAC) in the data, using the same shared secret k. Now, if the communication system is expanded to include a third node C, which also has the key k, the system breaks down. Node A can no longer assume that data authenticated with the key k originates from node B, because it may as well have been created by node C. The problem becomes even worse as the number of nodes in the communication increases. This problem makes IPsec unsuitable for multicast authentication.

Another problem for multicast authentication is that each packet needs to be authenticatable immediately when it is received. That is, each packet needs to contain all information that is needed to authenticate it. The alternative of receiving several packets and authenticating all of them together, for example, through one public key signature over several packets cannot be used. A system relying on authenticating several packets in one operation is vulnerable to a simple denial of service attack. Even one forged packet among the ones being authenticated is enough to make the authentication fail. An attacker sending packets with forged signatures to a multicast group could effectively force receivers to drop even packets which have been sent by a legitimate sender.

A better method for source authentication of multicast streams, known as TESLA, is discussed in [4]. The system is based on loose time synchronization between the sender and the receiver. In this system, the sender authenticates packets using a keyed hash algorithm. The system uses late release of authentication keys as protection against forged messages. Since the data is received before the server reveals the key which was used for authenticating the data, a receiver cannot forge messages even if symmetric algorithms are used for authentication. However, this results in a system where the data cannot be even temporarily stored in the network, because data that is received after the authentication keys have been released, cannot be properly authenticated. Thus, the authentication mechanism is suitable for authenticating the data to a storage host which is guaranteed to receive the data as soon as it is transmitted into the network. However, a terminal which later receives the data from a storage server, cannot anymore source authenticate the data.

The property which is really needed from the source authentication algorithm is nonrepudiation. Nonrepudiation is an additional property to a source authentication algorithm. As in any source authentication algorithm, the algorithm needs the ability to prove the sender of the data to the receiver. The additional property, which is also needed, is the ability to prove the identity of the original sender to any terminal that receives the data from the storage server. The simplest method of providing nonrepudiation is through signing the data using a digital signature algorithm.

The most simplistic method of signing multicast flows is to sign each packet separately from all other packets in the flow. However, such a system requires performing one public key signature verification every time a multicast packet is received. In this case, terminals with low computational capacity cannot be used for receiving multicast data.

In [5], a method for signing and verifying the sender of multiple packets using only one public key operation per a set of packets is introduced. The scheme is based on amortizing the signature on several packets using a tree based chaining technique. Even though multiple packets can be authenticated by only one signature, the system allows the receiver to authenticate the data immediately as it is received, even if other packets using the same signature are lost. The system also allows the receiver to verify the identity of the sender, even if the packet has been stored in a cache for any length of time.

Figure 3 illustrates the algorithm. The leaves correspond to hashes that are calculated over the packets to be signed. The value of each parent node is the hash of the concatenation of its children. For example, the value of node $P8$ is computed as $P8 = hash(P0|P1)$, where the vertical line denotes concatenation. When the tree has been generated, the sender signs the root of the tree. To enable the receiver to verify the signature, the sender must include into each packet, the hashes which are necessary to generate the path to the root. In the figure, the packet to be sent is $P4$. The sender must therefore include into the packet the hashes of nodes $P5$, $P11$, and $P12$. The sender also sends the signature of the root of the tree to the receiver.

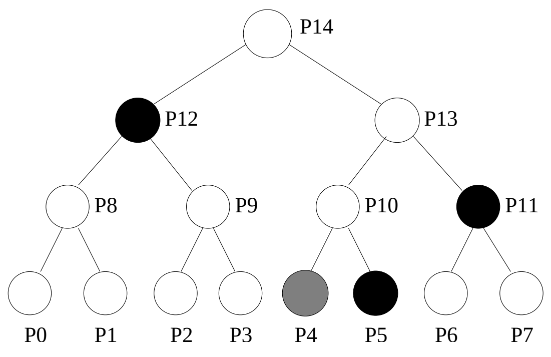

Figure 3: Hash tree

When the client receives a new packet, it must regenerate the path to the root of the tree. In the case of Figure 3, the client first computes the hash over the received packet, which corresponds to leaf $P4$ of the tree. It then generates the parent node by hashing the newly computed $P4$ and the $P5$ that was received in the packet, that is, $P10 = hash(P4|P5)$. The receiver continues this process until it hash created the hash of the root node. The receiver then checks the signature to verify the validity of the hash of the root. If the verification succeeds, the packet is valid. A more detailed description of the algorithm can be found in [5].

3 Solution

If the low capacity mobile terminal is unable to verify the digital signatures at a rate sufficient for receiving the multicast stream, it has several alternatives.

1. The mobile terminal may choose to ignore the digital signature entirely. However, this alternative makes the mobile terminal vulnerable to forged packets transmitted by hostile caches or nodes. If the integrity of the received data is important, this alternative cannot be used.

2. Another possibility is to only verify only some percentage of the packet signatures. If the signatures in the verified packets are correct, the mobile terminal assumed that the other signatures received in the unverified packets are also valid. This solution too makes the mobile terminal vulnerable to forged packets.

3. The mobile terminal may verify the packets later. The digital signatures in received packets are not immediately verified. The mobile terminal only stores the received packets in its memory. Once the mobile terminal has a sufficient amount processor capacity available for the verification, the packets can be processed. This solution, however, cannot be used for real time data streams where packets are used immediately when they are received. Examples of real time streams include, for example, radio shows that are delivered in real time to a multicast group.

4. The signature verification is delegated to another node. The node performing the verification is located somewhere in the network.

3.1 Delegated Verification

The alternative number 4 in the previous section delegates the verification of the signatures to another node in the network. This solution allows the mobile terminal to trust the received data while completely avoiding the computationally intensive digital signature verifications.

Figure 4 illustrates our solution. The mobile terminal is receiving multicast data from a multicast server in the Internet. The received data is source authenticated using the tree chaining algorithm. Although, the tree chaining algorithm reduces the number of digital signature verifications required to authenticate each packet of the multicast stream, the device may not be able to verify the signatures in real time. Instead, the mobile terminal delegates the verification to a trusted server in the Internet.

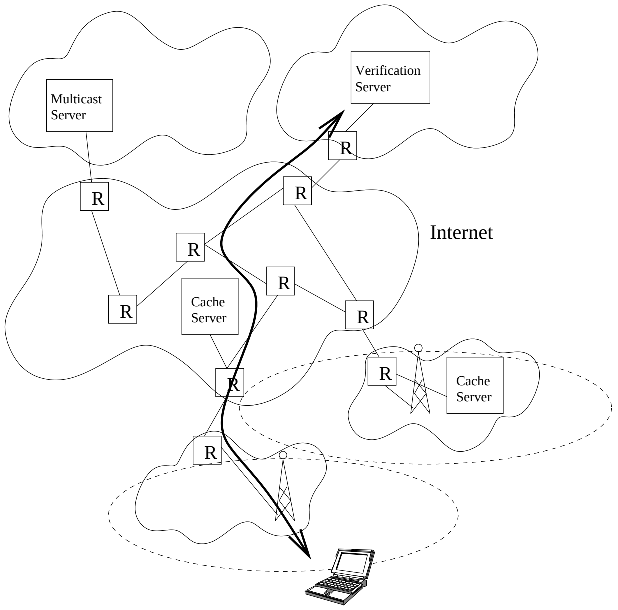

Figure 4: Solution

Verification requests, which are sent to the verification server, contains the following information.

1. The root hashes of the authentication trees which need to be verified.

2. The signatures received in the packets which need to be verified.

3. The verification key which can be used for authenticating the received packets. If this key has already been delivered to the verifications server, the key does not need to be sent again.

The verification request is sent to the verification server through an authenticated path, such as an IPsec tunnel. Any other security protocol which provides authentication for unicast communication can also be used instead of IPsec.

Once the server received the verification request, it verifies the signatures. The results of the verification is then sent back to the mobile terminal through the same autheneticated communication path which was used for sending the request. The mobile terminal then wither accepts or rejects the multicast packets based on the information received from the verification server.

4 Analysis

The most important benefit of using delegated verification for the signtures is the reduction in procesor capacity needed from the device receicing multicast. The system completely eliminates the need to perform digital signature verifications for receiving multicast in the mobile terminal. The signature verifications are replaced with a small number of symmetric cryptographic operations. Symmetric algorithms are orders of magnitude less expensive compared with digital signatures. Even very simple devices are able to easily perform these operations.

Delegated verification requires that a trusted verification server is always available to perform verifications when needed. Additionally, transferring the verification requests and responses between the mobile terminal and the verification server adds extra delay to the verification process.

The amount of network capacity required between the mobile terminal and the verification server is also small. The tree chaining algorithm in [5], allows a large number of packets to be authenticated using one hash tree. Therefore, only one signature needs to be transferred to the verification server to verify a large number of packets. Furthermore, if several verification requests are transferred in one request, the number of packets transferred between the mobile terminal and the verification server is reduced even further.

5 Conclusion

In this paper, we presented an architecture which allows mobile devices with very limited computational resources to receive source authenticated multicast data over an unreliable wireless network. Our method allows the mobile terminal to temporarily lose connection with the fixed infrastructure and request the lost packets to be retransmitted when connectivity with the fixed network is regained.

Earlier solutions based on digital signatures have been too computationally intesive. The previous alternatives to digital signatures, such as TESLA, cannot be used together with caching, since the source authentication is based on late release of keys. Our arhitecture solves both problems by using a trusted verification server in the fixed network for performing the digital signature computations.

Future work includes extending our existing implementation of the architecture with the optimization presented in this paper.

References

[1] Janne Lundberg and Catharina Candolin. Multicast caching: Efficient distribution of encrypted content to mobile clients. In *Proceedings of International Conference on Computer Networks (ICON'02)*, Rio de Janeiro, Brazil, October 2002.

[2] Janne Lundberg and Catharina Candolin. Hierarchical Multicast Caching. In *In Proceedings of The 9th IEEE Asia-Pacific Conference on Communication*, September 2003.

[3] Janne Lundberg and Catharina Candolin. Support for transparent multicast content distribution to mobile wireless clients. In *Proceedings of the 2003 International Conference on Wireless Networks (ICWN 2003)*, Las Vegas, Nevada, USA, June 2003.

[4] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[5] C. Wong and S. Lam. Digital signatures for flows and multicasts. In *IEEE ICNP '98*, 1998.