Reliable Data Delivery for Publish/Subscribe Networks

Markku Antikainen
Aalto University School of Science and Technology
mantikai@cc.hut.fi

Abstract

Network architectures built completely on publish and subscribe primitives instead of send and receive, have recently attracted many researchers. Pub/sub paradigm changes the networking completely: end-to-end principle breaks, communication channels become unidirectional, and multicast becomes a norm. Although pub/sub has attracted many, the research has been focusing on the network layer – functionalities that lie on the higher layers have not been studied so thoroughly.

This paper focuses on how reliability, which is normally addressed above networking layer, can be achieved in pub/sub networks. Reliability and guaranteed message delivery have traditionally been solved at the endpoints of the communication but the approach does not work anymore in pub/sub paradigm. The focus of this paper is on understanding the concept of reliability in publish/subscribe networks, how reliability has been solved in other (e.g. overlay) pub/sub architectures, and what are the fundamental requirements for reliable data transfer. In addition, recently proposed Cache-and-Forward architecture is used as an example when considering ways to achieve reliability in pub/sub networks.

KEYWORDS: publish/subscribe network, reliability, transport

1 Introduction

Internet applications have evolved enormously during recent decades and changed our concept of the Internet. These new applications require many new services, such as mobility, from the underlying network. Regardless of the changed needs and requirements of the applications, the Internet architecture remains very much based on protocols, such as TCP and IP, that were designed over 30 years ago. Despite the apparent success of TCP and IP, these protocols have several widely recognized shortcomings such as poor performance in highly dynamic mobile network topologies [9, 5, 13]. The fact that current protocols do not satisfy the needs of modern networking has led to development of new "clean-slate" networking architectures.

Cache-and-forward network architecture [4] (CNF) is a proposed data centric architecture that aims to solve many of the transport layer problems by using a reliable hop-by-hop transfer instead of a TCP-like end-to-end transport. The CNF is based on a concept of store-and-forward routers, which store the data packages before sending them onwards. This allows an opportunistic content delivery to occasionally disconnected machines that cannot be achieved with current end-to-end transport protocols. Efficient caching and content matching is enabled utilizing content-centric approach: the network names content instead of endpoints like in current Internet[1]. It has been shown that this kind of hop-by-hop transport has many advantages over traditional end-to-end transport protocols (such as TCP) in networks that are not as reliable as traditional wired networks [9, 13].

Although the CNF-like hop-by-hop transport service appears to be promising, it has not been widely studied in publish/subscribe networking context. The publish/subscribe networking can be implemented as an overlay on top of the existing architecture, but there are also architecture proposals where the whole networking is based on pub/sub [1, 15]. These pub/sub architectures aim to solve many of the existing network problems by giving the control of the information to the receiver, which contrasts with traditional network architectures where the sender has all control (i.e. the sender may choose what to send and to whom) [1]. This is achieved by changing the bottommost networking primitives *send* and *receive* into *publish* and *subscribe*: when a publisher publishes content, the content is delivered to all subscribers.

This chance has many consequences: it makes the network data-centric (since users publish and subscribe to data objects) and decouples the sender of the information from the receiver completely thus breaking the famous end-to-end principle. In addition, the communication channel becomes unidirectional multicast structure whereas currently almost all applications utilize bidirectional one-to-one communication. Among many other things, pub/sub networking solves the problem of unsolicited network traffic, such as SPAM and denial of service (DoS) attacks, since the receiver can choose what data to receive by subscribing.

While decoupling senders from the receivers solves many problems, it also introduces new challenges for the transport services. Current transport layer protocols (e.g. TCP and SCTP) are built on the fact that the communication channel is bidirectional. In practice this means that in protocols such as TCP the receiver can for example acknowledge packets sent by the sender. However in publish/subscribe networks, where bidirectional end-to-end communication is only an exception, this kind of approach is not possible.

This paper examines how reliability and reliable message passing could be implemented in pure publish/subscribe networks and especially how suitable cache-and-forward transport is for this kinds of networks. The structure of this pa-

[1] To be exact, CNF uses both endpoint addressing similar to IP addresses as well as content addressing.

per is as follows: section 2 explains basic concepts related to pub/sub networking and the CNF architecture. Section 3 briefly goes through some related works, what they solve and what do not. After this, section 4 discusses in more detail the challenges of implementing transport services into publish/subscribe networks and how these challenges could be overcome. Finally section 5 summarizes the problem field and concludes this paper.

2 Basic concepts

This section describes the basic concepts of publish/subscribe networking (section 2.1) as well as the Cache-and-forward architecture (section 2.2).

2.1 Publish/subscribe networking

In end-to-end network, such as in current Internet, anyone can send a packet to anyone and the network does its best effort to deliver that packet. This has led to a problem of unsolicited traffic such as denial of service attacks: the receiver has no way of telling the network not to deliver packets to it. The underlying problem is imbalance of powers between the sender and the receiver of information [8].

As already explained in the introduction section of this paper, pub/sub paradigm aims to solve this imbalance of powers by changing the bottommost networking primitives and thus giving the power to the receiver: the network does not deliver any data to the receiver unless it has explicitly asked for it. This kind of major change however has many consequences [6]:

- Because there may be several subscribers per each publication, multicast becomes a norm. However, current multicast solutions, such as IP multicast, are not scalable enough for Internet wide content dissemination.
- Because publisher and receiver of the information are completely decoupled in space, there must be a third entity in the network that provides rendezvous service for the different parties to find each other.
- Sending and receiving data is also decoupled in time meaning that messaging is asynchronous. Thus extensive usage of caching (even opportunistic caching) is needed to make the system scalable.

Two different types of publish/subscribe networking can be identified: *topic based* and *content based* [6]. Topic-based pub/sub networking uses the notion of topics under which events are published: when an event is published under a certain topic, the content is delivered to all subscribers subscribed to the corresponding topic. Thus these topics can also be seen as communication groups [6].

While in topic-based pub/sub networks the subscription is made to some topic[2], in content-based pub/sub the subscription scheme is based on the actual content. In other words events are not categorized according to static topics but according to their content. The content-based pub/sub is thus much more expressive than the topic-based pub/sub.

Although many of the reliability issues are identical regardless of the underlying pub/sub type, this paper focuses mainly on topic-based pub/sub systems.

[2] In some cases the subscription can also be made to a group of topics utilizing for example wildcards and hierarchical addressing.

LIPSIN

LIPSIN (Line-speed publish/subscribe internetworking [11]) is a forwarding fabric proposed for publish/subscribe networks. It allows line-speed multicast (and multipath) data delivery using probabilistic data structures. It is completely stateless in a sense that no per delivery-tree state is required. Thus it is very feasible for pub/sub networks.

In LIPSIN, router interfaces are addressed instead of endpoints. These addresses are called link-identifiers (LIT). The forwarding is done in a strict source-routing manner: the sender sends a packet to a *forwarding identifier* that contains all link-identifiers in the path. LIPSIN utilizes in-packet Bloom filters (called zFilters) in order to make this feasible: the forwarding identifier is a 256bit long Bloom filter, which contains all link identifiers added together using bitwise-OR. The actual forwarding decision is then done independently for each router interface by calculating bitwise-AND between the zFilter and LIT and testing if the result matches LIT (i.e. `zFilter & LIT == LIT`). Because of the nature of Bloom filters, this may cause some false positives but never any false negatives. However, if LITs are chosen carefully, the probability of false positives is relatively low [11].

Because the routing is done in source routing fashion, a third party is needed to construct the forwarding identifier. In LIPSIN this third party is called *topology manager* that is a logically centralized entity responsible for choosing and building forwarding paths and identifiers. In other words, LIPSIN provides only the bottommost forwarding fabric for publish/subscribe networks – it does not solve challenges related for example to route discovery and reliable message passing.

The concept of reliability in pub/sub networks

In host-centric networks, such as current Internet, the transport protocols are sender driven and utilize bidirectional communication: the sender chooses to retransmit packets based on the feedback (acknowledgments) sent by the receiver. Based on the acknowledgments both communicating endpoints know what packages have been transferred successfully. [14, 7]

In publish/subscribe networking, the concept of reliability changes: the communication becomes unidirectional and endpoints separated thus disallowing all kinds of direct communication between the endpoints. [6] Additionally, because the networking is data centric, the subscribers are not even interested in the publishers. However, the subscribers may want to be certain that they receive all data objects that they are subscribed to. In this paper, we consider the publish/subscribe system to be reliable if *the subscriber can be certain that it will receive all publications that it has subscribed to*. Because it is the network that manages the decou-

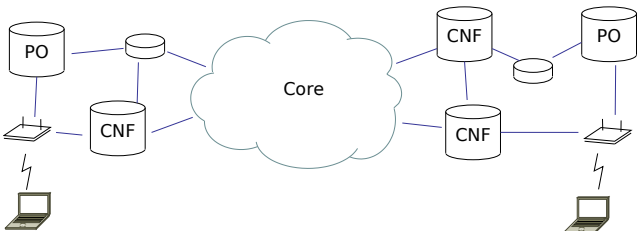

Figure 1: Simplified CNF architecture [4]

pling of the sender and the receiver, this could also be said with other words: pub/sub system is reliable if the network can deliver all publications reliably to subscribers who have subscribed to it. This means also that in publish/subscribe paradigm, the network has a role in providing the reliability, which is not the case in current Internet.

2.2 Cache-and-forward networking

While the motivation for publish/subscribe networking is the imbalance of powers in current Internet, the main motivation behind Cache-and-forward architecture (CNF [4]) is the inefficiency of the current transport protocols. Although traditional transport layer protocols work relatively well in wired and static network topologies, they are not optimal in wireless environments, where error rates are high [13, 5]. In this kind of networks, where link quality is poor and endpoints are occasionally disconnected, *hop-by-hop transport* is more efficient [12]. Hop-by-hop transport means that some transport services are provided between intermediate nodes instead of providing all transport services in the endpoints of communication like in TCP. Cache-and-forward architecture is a proposal that provides reliability between each intermediate network element (i.e. in hop-by-hop manner) as well as between the endpoints, and is thus capable of working also in environments where connections are often lost.

The architecture of CNF can be seen in figure 1. In the core network there are high-speed IP routers. Outside the core network there are *CNF router*s that work in hop-by-hop store-and-forward manner, where hop is a CNF-router hop instead of an IP hop. What this means is that each data packet is delivered completely to the next CNF router, stored there, and sent forward only after that. Thus the CNF routers contain large tightly integrated caches, which can also be used in opportunistic manner. The network may also contain traditional routers without CNF capabilities.

At the very edge of the network, there are mobile endpoints that are connected to the network using a wireless link[3]. Because mobile endpoints may be occasionally disconnected, the mobile endpoint may request that the data is delivered and stored to its a local *post office* (PO) which works as a rendezvous point for senders and receivers. When the receiver becomes online again, it may request the data chunk from the PO. In other words, the PO acts as a local cache.

[3]Naturally, there may also be nodes with wired connections, but these are not emphasized in CNF.

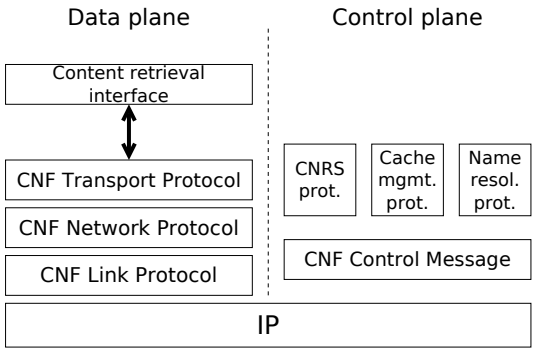

Figure 2: CNF protocol stack CNF is used on top of IP [4]. The left side contains data-plane protocols while the right side contains control plane protocols.

CNF protocol stack

CNF can be built either over an IP network as an overlay or as a clean-slate architecture. Figure 2 illustrates CNF protocol stack when it is used as an overlay. Because the focus of this paper is on reliability, only the data plane protocols are explained. Information on the CNF control plane protocols, as well as more complete descriptions on data-plane protocols, can be found in [4].

CNF Transport Protocol (CNF TP) is the only end-to-end protocol in CNF protocol stack. CNF TP is much lighter than TCP because most of the traditional transport services such as congestion control, error control, and flow control are included in link and network layers. CNF TP must also be very lightweight because transport layer connections may be much more long living than normal TCP connections because CNF TP allows occasional disconnections. The main service that CNF TP provides is end-to-end delivery acknowledgments.

CNF Network Protocol (CNF NP) is used to route content and content requests. This protocol is very much like current IP: packets contain both source and destination addresses and the packet is routed towards the receiver like in current Internet. However, because CNF is content-centric architecture, the CNF NP also contains a separate field for content identifier, which is needed for caching and content matching.

CNF Link Protocol (CNF LP) is the lowest-layer protocol in CNF architecture. CNF LP works between two CNF nodes and it provides reliable message passing between them. Because different CNF nodes are interconnected with different mediums, the CNF LP can negotiate parameters that correspond the underlying link.

Reliability in CNF

As explained, CNF provides reliability in two layers: link layer and transport layer. Although this approach may sound inefficient, this is not the case: Liu et. al. showed in [12] that CNF is competitive with TCP/IP in wired networks and provides much greater throughput in wireless networks.

Figure 3 illustrates the relationship between the two layers that provide reliability in CNF: reliable protocols complement each other. The transport protocol is needed for exam-

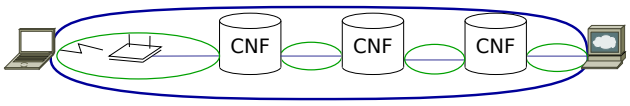

Figure 3: Reliability in CNF. The inner (green) ellipses illustrate link layer reliability while the blue one illustrates end-to-end reliability provided by the CNF TP.

ple to initiate the retransmission, while link protocol guarantees that although the underlying medium is unreliable (e.g. wireless), the message gets delivered to the next hop.

3 Related work

It is a well known fact that the loose coupling between the sender and the receiver of information in publish/subscribe networks makes reliable and guaranteed content delivery difficult [6]. This section explains some proposed solutions for this problem. Section 3.1 covers probabilistic gossip algorithms while section 3.2 focuses on more traditional ways of providing reliability in publish/subscribe networks.

3.1 Gossip algorithms

Costa et. al. proposed in [2] the use of gossip algorithms[4] in order to achieve probabilistic reliability for pub/sub networks. In gossip algorithms the nodes share (gossip) some of their information with some number of other nodes. This can be utilized in distributed computing: if all nodes reveal their partial state to some randomly chosen other nodes, all nodes can validate their own state with some accuracy.

The architecture proposed in [2] uses gossip algorithms in order to probabilistically guarantee that all event dispatchers (nodes that participate to the pub/sub routing and caching) have all data that they are interested in. This can be achieved in two ways: using proactive push-style messaging or reactive pull messaging. When the gossiping is done in push-style, all nodes periodically gossip some part of their internal state to randomly chosen counterparts. In pub/sub networking, the gossiped state contains identifiers of all recently received publications. When the dispatchers receive these gossip messages, they can then validate that both parties have received all publications that they are interested in. If this is not the case, the missing publications are sent to the one that does not have them.

When this algorithm is used in push style, the excess network utilization is heavy even when all dispatchers receive all publications that they are interested in. In order to tackle the excess network traffic, [2] also proposes pull-style gossip messaging for pub/sub networks. If all publications contain some kind of sequence identifier from which the dispatcher may check whether it has received all previous messages belonging to the same stream, it can initiate the gossiping only if it misses some piece of data. Although this approach reduces the gossip traffic, it has also shortcomings: first of all the message loss is detected only after the next publication belonging to the same stream is received. Secondly, although sequence identifiers work easily in topic-based pub/sub systems, in content-based pub/sub systems this is difficult to implement.

Although it has been shown in [3] that this kind of gossip mechanism is feasible for medium sized networks (where the number of dispatchers around 100), gossip algorithms have several drawbacks. The main two disadvantages are that although they work well with high probability, they are still probabilistic. In addition they gossip algorithms have the drawback of high network utilization (in the case of push-style gossiping) and late error discovery (in the case of pull-style gossiping).

[4] Gossip algorithms are sometimes referred to as *epidemic algorithms*.

3.2 Other approaches for reliability

Eugster et. al. identify in [6] three methods that have been used in pub/sub networks for guaranteeing message delivery:

- Centralized pub/sub systems often store all messages in a centralized storage. Because the message transfer from/to the centralized data storage can be guaranteed, this provides reliability. However if the centralized data storage becomes unavailable, the message delivery is delayed.
- Pub/sub systems that are based on an overlay of event dispatchers often use a reliable protocol (e.g. reliable application-layer multicast) between the dispatchers. Publishers and subscribers communicate with their nearest dispatcher using reliable protocols at the lower layer. The reliable lower-layer protocol could be for example TCP.
- There are also pub/sub architectures, such as TIBCO Rendezvous [16], where the publisher communicates directly with the subscribers using a reliable multicast protocol. The time decoupling is then implemented completely separately.

Although all of these meet the requirement of reliable data delivery, none is optimal. The first one relies on a centralized database, which may cause scalability issues. Also the whole network becomes inoperable if the centralized blackboard becomes unavailable[5]. The second and third approaches do not decouple endpoints completely and thus do not make the most of pub/sub paradigm. For example direct denial of service attacks against endpoints are still possible in these architectures. What is notable, are the similarities of the first and second approaches. Both of these utilize reliable transport on under the pub/sub networking layer.

4 Reliability for publish/subscribe network

When discussing about reliability in pub/sub networks, the key question is how to provide end-to-end reliability in a

[5] Although it can be argued that also the current Internet is completely dependent on a centralized system, namely DNS, these cases are not comparable because DNS is not required for *networking* whereas if the centralized blackboard becomes unavailable, the message delivery will not work at all.

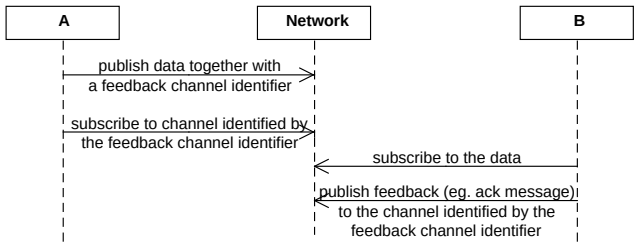

Figure 4: Building bidirectional communication channel using publish and subscribe primitives.

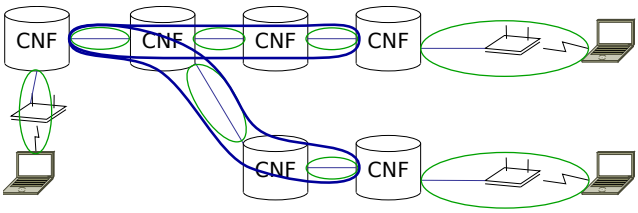

Figure 5: How guaranteed message delivery could be achieved in pub/sub. The inner (green) ellipses represent link layer reliability while the outer ones (blue) represent reliability achieved on a higher layer.

network where endpoint are decoupled and end-to-end communication is only an exception. Section 3 already touched on the mechanisms that can be used to provide reliability for pub/sub. However, as it was seen, none of the presented approaches was perfect: some provided only probabilistic reliability while others were designed for overlays.

This section tries to sketch ways that could be used to guarantee message delivery in non-overlay pub/sub network. At first, section 4.1 provides a rather naïve solution: since the challenge is to provide end-to-end reliability without end-to-end networking primitives, one possibility would be to build end-to-end connection using just the pub/sub primitives. After this, section 4.2 tries to solve the problem utilizing hop-by-hop transport mechanism derived from the CNF architecture.

4.1 Naïve solution

Because the problem is *building end-to-end reliability on a network that does not support by default end-to-end communication*, a naïve solution would be used to provide end-to-end communication paradigm over publish and subscribe primitives. When the endpoints are able to communicate in end-to-end fashion, reliability can be solved at the endpoints similarly as the current transport protocols. It is clear, that this kind of approach loses almost all the benefits that pub/sub paradigm offers since the endpoints become tightly coupled. For example, time and location decoupling is lost and multicast becomes inefficient because the publisher would be responsible for providing reliability for each individual receiver. However, the idea here is to show that this kind of approach is not impossible, even if it is infeasible.

Figure 4 illustrates how bidirectional communication can be achieved in publish/subscribe paradigm: separate communication channels are used for each directions. In practice, this can be done so that the communicating endpoints subscribe to their own feedback channel, through which other nodes can send acknowledgments to them. With this trick, the communication channel becomes bidirectional: parties can send data to each other as with traditional send/receive primitives and thus also the reliability can be solved at the endpoints of the communication (like for example in TCP).

As already said, this approach is clearly very infeasible: all the benefits of pub/sub are lost in this kind of communication. In addition, it is the original publisher, who decides whether reliability is needed. This does not fit to receiver driven paradigm, where receiver's control over the data is emphasized.

4.2 Cache-and-forward in pub/sub network

Although cache-and-forward and pub/sub networking architectures are based on completely different paradigms, they have similarities: both are data centric and provide at least some degree of time-decoupling. This raises a question whether it could be possible to combine some of the parts of CNF to pub/sub networking in order to create reliable data delivery for the pub/sub network.

Building reliable pub/sub on top of CNF

We start sketching the reliable pub/sub system by having a network that contains CNF routers. A reliable link protocol like CNF LP is used between them so that the message delivery between two intermediate CNF routers is guaranteed. LIPSIN can be used in the network layer to enable efficient multicast. This however means that some benefits of LIPSIN are lost: because LIPSIN runs now over a reliable link protocol and CNF routers cache some of the packets, there is a risk that some efficiency of LIPSIN is lost.

If the network topology is static, reliable link layer is pretty much all needed for guaranteed data delivery: the reliable lower layer protocol guarantees that the packet always makes to the next hop. However if the network topology changes, for example a link breaks, this system does not work anymore. In CNF, situations like this are solved using a light-weight end-to-end transport protocol as well as dynamic rerouting if the link protocol fails [4]. Dynamic rerouting does not solve this completely since there are cases where it is not possible to route around the failure. Additionally LIPSIN can do this kind of rerouting[6] only in limitedly [11].

Because message delivery cannot be guaranteed only with reliable link layer, some mechanism is needed which can provide retransmission in cases of broken links and lost packets. As said, CNF uses a light-weight protocol between the endpoints (CNF TP) for this [4]. In pub/sub environment, bidirectional end-to-end communication is naturally out of the question. However the reliable higher-layer connection does not need to be *between the endpoints* – having reliable connection for example between two edge CNF

[6] In LIPSIN terminology this is called fast-recovery.

routers could be enough. Figure 5 visualizes this approach: the publisher sends the data to its nearest CNF-router, which then would be responsible for delivering it reliably to the subscribers.

Although the approach of creating light-weight connections between the edges of the network sounds feasible, there may be problems in cases where a publication has a large delivery tree. This is because the edge CNF-router at the publisher's end needs to store state on every connection, which may cause a bottle neck to the network. Because of this, some mechanism is needed that reduces the work of the edge CNF-router if some publication becomes very popular.

One consequence of creating reliable connection between two edge CNF-routers is that it introduces sender driven paradigm inside pub/sub paradigm: the message delivery between the edge CNF-routers is guaranteed by the fact that the publisher's side CNF-router can resend packets that are not acknowledged by the receiving CNF-router. Although this may seem inelegant, this is not the case because on some lower level the networking is always sender driven.

Efficiency of the approach

Although the motivation here is to decouple the endpoints, providing some of the transport services in hop-by-hop manner may also result in better efficiency compared to the situation where all transport services are provided by the endpoints of the communication. Simulations made by Heimlicher et. al. [10] support this hypothesis – for example, it is clearly inefficient to provide congestion control at the endpoints of the communication because there may be only one congested link at the path.

Although there is evidence that hop-by-hop cache-and-forward data delivery is in many cases more efficient than traditional TCP-like protocols, these result cannot be generalized to publish/subscribe networks. Paradigm shift may change the network substantially and there are no guarantees that the traffic flows in publish/subscribe network are similar to those that were simulated in [10, 12]. Thus, although it may justified to say that cache-and-forward style transporting is promising, validating this hypothesis requires more solid evidence.

5 Conclusion

Implementing reliability for pub/sub network so that the endpoints will not get tightly coupled is not straightforward. As section 3 explained, there have been quite many different approaches. However many of these solutions are either infeasible or used in overlay pub/sub systems, which do not face the same problems as pure pub/sub networks.

Section 4 presented two different approaches on reliability in pub/sub network. While the naïve solution produces a bidirectional end-to-end communication channel between the publisher and subscribers, the latter solution (section 4.2) utilizes hop-by-hop approach. Although these are only coarse examples, they illustrate the differences of end-to-end style transport and Cache-and-forward style hop-by-hop transport. Understanding the difference of these is essential when designing transport services.

The main observation of this paper is that although the development of hop-by-hop transport has been mainly motivated by network efficiency, hop-by-hop transport also decouples the sender of the information from the receiver of the information. Thus Cache-and-forward style hop-by-hop transport is a very good candidate for publish/subscribe networks. However, how hop-by-hop transport should be implemented is another question, which requires much more research.

This paper did not examine very thoroughly how multicast affects the reliability in hop-by-hop environment. Additionally, this paper intentionally left out the consideration on unreliable data transfer. There are situations where errors in the packets are not prejudicial and communicating parties want rather low latencies than reliability. These requirements must not be forgotten when designing transport solutions for pub/sub network.

References

[1] Ain M. (edit). Architecture definition, component descriptions, and requirements. Deliverable D2.3. Technical report, Jan. 2008. PSIRP Project.

[2] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola. Introducing reliability in content-based publish-subscribe through epidemic algorithms. In *DEBS '03: Proceedings of the 2nd international workshop on Distributed event-based systems*, pages 1–8, New York, NY, USA, 2003. ACM.

[3] P. Costa, M. Migliavacca, G. P. Picco, and G. Cugola. Epidemic algorithms for reliable content-based publish-subscribe: An evaluation. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 552–561, Washington, DC, USA, 2004. IEEE Computer Society.

[4] L. Dong, H. Liu, Y. Zhang, S. Paul, and D. Raychaudhuri. On the cache-and-forward network architecture. In *ICC '09. IEEE International Conference on Communications, 2009*, pages 1–5, Dresden, June 2009.

[5] T. D. Dyer and R. V. Boppana. A comparison of tcp performance over three routing protocols for mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 56–66, New York, NY, USA, 2001. ACM.

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, 2003.

[7] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (Proposed Standard), Apr. 2004.

[8] N. Fotiou, G. C. Polyzos, and T. D. Illustrating a publish-subscribe internet architecture. Future Internet Architectures: New Trends in Service Architectures (2nd Euro-NF Workshop), June 2009.

[9] Z. Fu, X. Meng, and S. Lu. How bad tcp can perform in mobile ad hoc networks. In *ISCC '02: Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, page 298, Washington, DC, USA, 2002. IEEE Computer Society.

[10] S. Heimlicher, R. Baumann, M. May, and B. Plattner. The transport layer revisited. In *COMSWARE 2007: Communication Systems Software and Middleware*. IEEE, 2007.

[11] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: line speed publish/subscribe inter-networking. In *SIGCOMM '09: Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, pages 195–206, New York, NY, USA, 2009. ACM.

[12] H. Liu, Y. Zhang, and D. Raychaudhuri. Performance evaluation of the "cache-and-forward (CNF)" network for mobile content delivery services. In *Communications Workshops, 2009. ICC Workshops 2009*, pages 1–5, Dresden, June 2009.

[13] S. Paul, R. Yates, D. Raychaudhuri, and J. Kurose. The cache-and-forward network architecture for efficient mobile content delivery services in the future internet. In *Innovations in NGN: Future Network and Services, 2008. K-INGN 2008. First ITU-T Kaleidoscope Academic Conference*, pages 367–374, Geneva, May 2008.

[14] J. Postel. Transmission Control Protocol. RFC 793 (Standard), Sept. 1981. Updated by RFCs 1122, 3168.

[15] M. Säärelä, T. Rintaho, and S. Tarkoma. RTFM: Publish/subscribe internetworking architecture. In *ICT-MobileSummit 2008 Conference Proceedings*, 2008.

[16] TIBCO. Tibco rendezvous. White paper, 1999.