# Creating Butterflies in the Core – A Network Coding Extension for MPLS/RSVP-TE*
## (Work in Progress)

Thorsten Biermann, Arne Schwabe, and Holger Karl

University of Paderborn, Research Group Computer Networks,
Pohlweg 47-49, 33098 Paderborn, Germany
{thorsten.biermann,arne.schwabe,holger.karl}@upb.de

**Abstract.** Network Coding exploits network resources more efficiently than plain routing. Specifically, it reduces packet delays and packet loss rates. Such advantages are especially useful in core networks where many different users can benefit from this at once. Hence, network coding should be transparently integrated in technologies suitable for core networks. We present an extension for MPLS and RSVP-TE which can instantiate network-coded paths in core networks. The feasibility of this approach is demonstrated in a proof-of-concept simulation.

**Keywords:** Network coding, Signaling, Core, MPLS, RSVP-TE.

## 1 Introduction

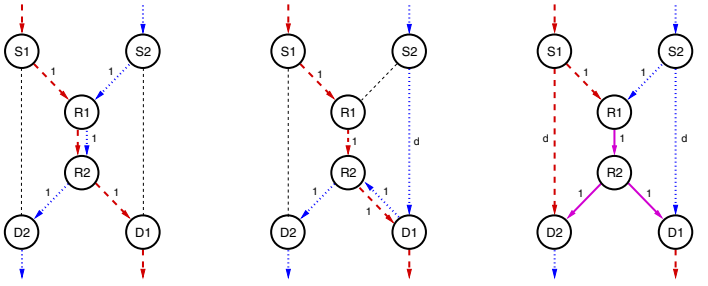
The technique of Network Coding (NC), as first introduced by Ahlswede et al., has become a vivid area in current networking research. Instead of only forwarding packets, NC permits nodes on the packets' route to process the data, i.e., to modify and mix up their content. These transformations are inverted at a decoder again to deliver the original content to the destination. Depending on where which transformations happen, various benefits can be achieved.

There is a variety of applications where users can benefit from NC. In this paper, we concentrate on NC to use available network resources more efficiently than with plain routing. In particular, we look at core networks where multiple flows share physical links. In such scenarios, links are usually redundantly deployed to deal with link failures or temporary overload situations. Affected flows are switched from the broken or overloaded link to alternative links. An example for such a scenario is depicted in Fig. 1(a) and 1(b) where the link R1→R2 gets congested due to a failure or a load peak.

As soon as the congestion is detected, Flow 2 (blue/dotted) is redirected to the alternative route on the right side (S2→D1→R2). Although all links are

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216041.

L. Fratta et al. (Eds.): NETWORKING 2009, LNCS 5550, pp. 883–894, 2009.




(a) Enough capacity is available. Packets of both flows are transported using classic forwarding. The overall latency is 3.

(b) Overload on R1→R2 forces the blue/dotted flow to be redirected via an alternative route. Its latency raises to $d+2$.

(c) Alternatively, NC is applied: Encoding is done at node R1; decoding at D1 and D2. Latency remains at $\max(d, 3)$.

**Fig. 1.** Comparison of classic transmission via packet forwarding and NC-enabled transmission of two data flows. The numbers next to links denote their latencies. In contrast to routing, NC permits load shifting from link R1→R2 to lateral links while retaining low latency.

now able to deal with the incoming load, depending on the alternative links' properties, the rerouted flow might suffer from noticeably higher delay.

To overcome this drawback, linear NC can be applied. This technique regards packets as vectors and allows a node to apply a linear transformation to them before they are forwarded. Linear NC achieves optimal throughput for multicast transmission, i.e., the maximum flow from the source to each receiving node. A special case of such linear transformations is calculating the XOR value of two packets. In our scenario, instead of completely redirecting one of the flows, both of them are jointly encoded at R1 by calculating the XOR of each packet pair of the flows. This reduces load on the bottleneck link and maintains low delay for both flows. Decoding is done at the destinations, as shown in Fig. 1(c).

The techniques and benefits of network coding have been widely explored [1], but the control problem – when and where to turn it on and how to signal this – has only been addressed in wireless contexts [2,3] or on the application layer [4]. Specifically, we are not aware of approaches trying to integrate NC into wired systems on the link or network layer. Especially, controlling NC for traffic flows in core networks, where many users benefit at once, is not possible so far. To achieve this, we present a signaling protocol in Sec. 3 and 4 that triggers and controls the instantiation of network-coded paths in networks using Multiprotocol Label Switching (MPLS), as a typical example of a core network technology. Our protocol is independent of the algorithm used for detecting NC possibilities, i.e., finding suitable subtopologies [5], as well as of the algorithm that decides when to actually activate coding. Both decisions are made on top of our signaling protocol. The feasibility is demonstrated in a proof-of-concept simulation in Sec. 5. Sec. 2 briefly summarizes relevant MPLS background.

## 2 Background

Data transport in core networks is often realized with *label switching*. Instead of evaluating hierarchical IP addresses, simple flat labels are prepended to packets. A switch processes incoming packets solely based on their label, i.e., all packets with the same label are treated equally.

Label switching requires two components: a data transport service which actually transports the data packets based on their labels and a label distribution protocol which sets up the Label Switched Paths (LSPs), i.e., distributes labels to be used among Label Switch Routers (LSRs). This determines the actual forwarding behavior. As MPLS [6] is a practically highly relevant transport service for label switching, we will focus on extending it by NC; in principle, the technique should also be applicable to other label switching transport services.

Resource Reservation Protocol – Traffic Engineering (RSVP-TE) [7] is one of several label distribution protocols that can be used with MPLS. It has been extended multiple times to support emerging networking technologies and consolidates many aspects of traffic engineering in a single protocol. Besides the wide adoption, we have chosen this protocol as base for our extension because there is already a Point-to-Multipoint (P2MP) extension available [8] which is necessary for applying XOR NC in a butterfly topology.

## 3 Extending MPLS to Support NC

To perform XOR NC in a butterfly, three issues have to be solved: packets must be encoded (at node R1 in Fig. 1(c)), the uncoded packets must be sent to decoding points (from S1/S2 to D2/D1), and the right encoded and uncoded packets must be matched for the decoding operation (at D1/D2). When implementing these additional functions, the following requirements have to be met:

- Using NC must not disrupt normal label switching operation.
- Modifying nodes not directly involved in NC operations, i.e., which only forward network-coded packets, is not acceptable.

To achieve this in MPLS, two modifications are required. First, LSRs which are responsible for en-/decoding need to be extended to support these operations and, second, one LSR passing the flow to be encoded must be able to add packet sequence numbers as packets must be uniquely identifiable. Hence, in contrast to traditional label switching where all LSRs are equal and perform the same operations (packet forwarding based on labels), NC requires to distinguish between different roles that LSRs may adopt:

- *Forwarding LSRs* perform traditional store-and-forward operations required for ordinary packet forwarding. This also encompasses cloning of flows for multicasting. The payload of forwarded packets is unchanged.
- *Numbering LSRs* add packet sequence numbers to MPLS flows. The packets' payload is not altered either.

- *Encoding LSRs* apply XOR coding to packets of two incoming flows. The result is one encoded output flow.
- *Decoding LSRs* receive one encoded and a corresponding plain flow. After decoding, the resulting decoded packets are forwarded.

Required modifications for implementing the different LSRs are described in the following two subsections. Setting up paths is mostly a problem for the signaling protocol and described separately in Sec. 4.

### 3.1 Encoding and Decoding

In MPLS, an LSR determines how to treat an incoming packet solely based on its label; the concrete action is determined by a Next Hop Label Forwarding Entry (NHLFE). NHLFEs are basically table entries which map an incoming label/interface combination to corresponding next-hop information (outgoing interface and next-hop address) and label manipulation instructions like label push, pop, or swap operations.

To support XOR NC, NHLFEs must be extended to not only support simple forwarding of packets but to also indicate to the LSR that two incoming flows shall be combined to one single outgoing flow. This is done by feeding packets of an incoming flow into an encoder. The partner flow for encoding is selected by a second NHLFE which feeds another flow into the same encoder instance. This encoder contains all functionality required for the actual encoding operation like buffering, XOR calculation, etc. This modification is sufficient to implement an XOR *Encoding LSR*, as well as the corresponding *Decoding LSR*.

### 3.2 Packet Sequence Numbers

To be able to successfully decode packets at destination LSRs, these nodes need to know which packets from which flow were used for encoding (e.g., trying to decode ($a$ XOR $b$) from flows A and B with a packet $b'$ from flow B obviously gives wrong results). Hence, packets forming an MPLS flow which is involved in any coding operation must carry information that allows their unique identification. The simplest way to achieve this is by adding sequence numbers to packets.

As MPLS flows usually transport an aggregate of many different flows, it is impossible to reuse sequence numbers of upper layers for this; they have to be explicitly generated for a given MPLS flow. This is done by *Numbering LSRs* with a special NHLFE. Such coding-related sequence numbers are not required over the *whole* packet lifetime. It is sufficient to add them before the flow is multicasted. E.g., in the scenario depicted in Fig. 1(c), this is done by the LSRs S1 and S2 at the latest. At the Encoding LSR (R1), both flows are combined and, hence, each packet of the encoded flow must carry two sequence numbers.

Extending the MPLS header to transport the additional packet sequence number is not feasible as this breaks compatibility to standard MPLS implementations. Normal LSRs would *not* be able to forward packets already numbered for NC later on, but this is an obvious deployment requirement. Therefore, we use

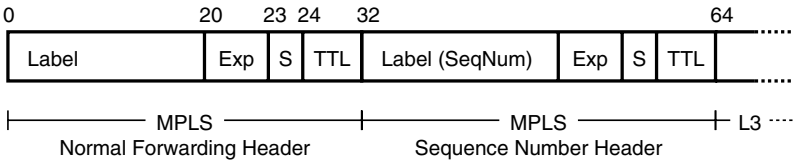



**Fig. 2.** MPLS label stacking is exploited to transport packet sequence numbers. The label field of the inner header contains the packet's sequence number.

the label stacking feature of MPLS to push an additional sequence number label on the stack, transporting the sequence numbers within the 20 bit label field. Fig. 2 shows a sample stack transporting a single packet sequence number.

Sequence numbers also allow us to treat boundary cases. For example, it could happen that at the Encoding LSR, a packet arrives and, even after waiting some time, no packet from the partner flow is available for joint coding. One option would be to stall such a packet until a partner packet for encoding arrives; an alternative is to send this packet uncoded (to avoid blocking buffers) and to indicate from which flow a packet is missing by an empty sequence number.

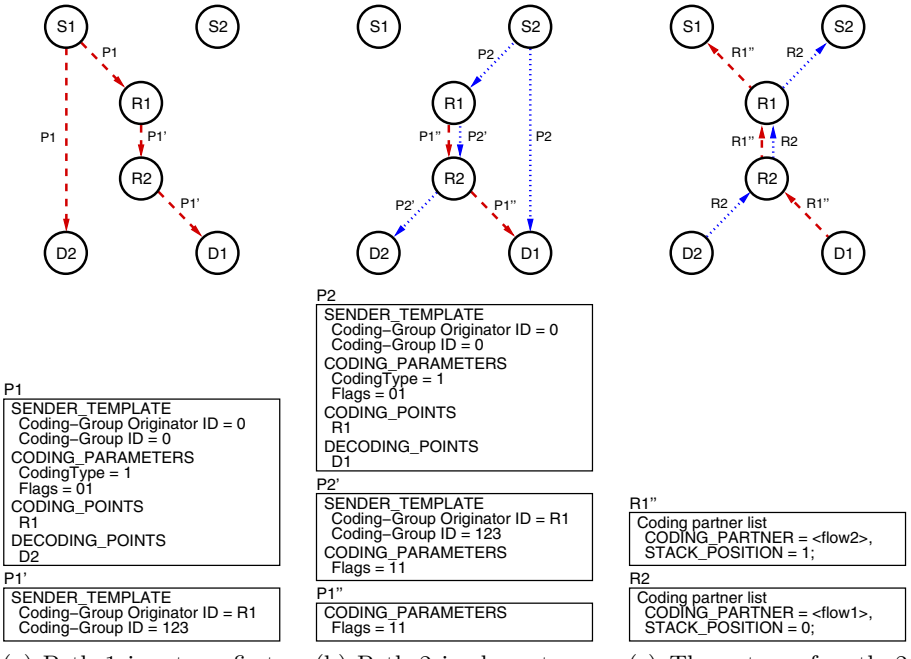
## 4 Extending RSVP-TE to Signal NC-Enabled Paths

The modified MPLS system can realize NC operations within the network. What is still missing is a mechanism to propagate forwarding and de-/coding configurations to involved nodes to actually turn on NC. To do so, the label distribution protocol has to be extended as well. For the design of the RSVP-TE NC protocol extension we must meet the following requirements:

- Using NC functionality must not disrupt normal RSVP-TE operation.
- Switching an LSP from normal to coded operation must be possible without disturbing the ongoing transmission.
- Assured QoS constraints must not be violated when using NC, i.e., RSVP-TE must still be able to manage reservations and guarantee them.
- Although the extension was designed with XOR NC in butterfly-like topologies in mind, it is kept as flexible as possible to easily permit other coding techniques and topologies as well.

Fulfilling these requirements guarantees a seamless integration of NC LSPs into a running network. The actual decision if and where NC is applied within the network and which pairs of flows should be coded together is orthogonal to the signaling problem addressed in this paper. Our signaling scheme supports any kind of decision algorithm on top. The decision can even be made independently from the nodes forming a coded path. The only limitation is that the decision must be communicated to a node on the path to start the signaling process.

Based on the presented assumptions, the next sections describe how a NC LSP is actually set up (Sec. 4.1) and teared down again (Sec. 4.2).




(a) Path 1 is set up first. `PATH` message P1 is modified at R1. The resulting message is called P1'.

(b) Path 2 is also set up. Besides P2 being modified to P2' at R1, path 1 is also refreshed using P1".

(c) The setup of path 2 and refreshing of path 1 is acknowledged to the sources via `RESV` messages.

**Fig. 3.** Sample for joint setup of two network-coded LSPs. S1/S2 are ingress LSRs, R1 is the Encoding LSR, and D1/D2 are the egress LSRs which do the decoding. Red/dashed arrows denote signaling messages belonging to path 1; blue/dotted arrows the signaling of path 2. The first `RESV` reply for path 1 is omitted. The boxes denote the signaling messages (only NC-relevant data which has changed is shown).

### 4.1 Setting Up Network-Coded Paths

RSVP-TE uses end-to-end semantics to set up paths by sending a `PATH` message from the ingress to the egress LSR. It is acknowledged by a `RESV` message. Hence, the use of NC has to be signaled by the ingress LSRs of both flows to be encoded.

There are two possible constellations how network-coded LSPs can be set up. In the first and simpler case, both ingress LSRs are aware that their LSPs will be used for NC. Hence, they are both initially set up with NC enabled. In the second case, one or both sources start without the knowledge that their LSP will be used for NC later on. Hence, the paths are set up without NC enabled. This means that the NC-unaware LSPs must be converted with RSVP-TE's rerouting mechanisms [7, Sec. 4.6.4] afterwards. This procedure is not discussed here as it is identical to the standard behavior of RSVP-TE.

**Simultaneous Setup of Network-Coded Paths.** In this scenario, both ingress LSRs S1 and S2 know at instantiation time that their LSPs will be encoded. This information can either be propagated automatically by the algorithm which decides when and where NC is applied, or can be given manually.

The setup process is clarified in Fig. 3, which illustrates important steps of our signaling scheme. The LSPs start at the nodes S1/S2 and end at D1/D2. Path 1 (starting at S1) is set up first with initially no NC active. It is refreshed after the creation of Path 2 to be jointly encoded. To transport the additionally required NC parameters, the `PATH` and `RESV` messages are extended. These modifications will be described in detail later on. Note that at points where the path is multicast, `PATH` messages are copied and sent along both subpaths, akin to the P2MP upon which we built.

The path setup starts at S1 by sending the first `PATH` message P1 to R1 and D2. After R1 added itself as `Coding-Group Originator` and assigned a new `Coding-Group ID`, the message P1' is sent to R2 and finally to D1. As coding is not yet activated, D1 responds with a standard `RESV` message (not shown). Next, S2 also starts its path setup by sending P2. R1 receives P2 and detects the coding possibility with the first LSP based on the `DECODING_POINTS`. Hence, coding is enabled in P2' and path 1 is refreshed using a new `PATH` message P1".

Signaling of label stack positions for particular sequence number labels is only done by the Decoding LSRs when coding is actually enabled. Fig. 3(c) shows this signaling within the `RESV` message after encoding has been enabled by R1.

*Modified PATH message.* The intention of setting up an NC-enabled path is signaled by modifying its `SENDER_TEMPLATE` object and by adding four additional objects to the `PATH` message. These changes are discussed in the following.

`SENDER_TEMPLATE.` Two LSPs which shall be jointly encoded must be marked such that Numbering, Encoding, and Decoding LSRs know about this intention. Therefore, two additional fields (`Coding-Group Originator ID` and `Coding-Group ID`) are added to the `SENDER_TEMPLATE` object. Both are initialized with zeros by the source LSR and will be filled by the Encoding LSR. Similar to the P2MP extension, the `Originator ID` specifies the ID of the Encoding LSR (usually its IP address), whereas the `ID` is assigned by the Encoding LSR for this particular encoding operation. In combination, both IDs identify the group of LSPs which are jointly encoded on a global scope.

`CODING_PARTNERS.` There are two ways of establishing encoded LSPs: explicit and automatic partner flow selection. For the explicit partner selection, the desired partner flows are specified in the `CODING_PARTNERS` object by a list of subobject pairs. Each pair consists of the partner flow's `SESSION` and `SENDER_TEMPLATE` object. This uniquely identifies LSPs for joint encoding. In case there is no `CODING_PARTNERS.` object in the `PATH` message, an Encoding LSR may choose any LSPs for joint encoding. The LSR's decision is signaled by adding a `CODING_PARTNERS` object referencing the selected partner flow.

`CODING_PARAMETERS.` This field is used to define the coding operation that is actually applied at the Encoding LSR (`CodingType`) and transports several status flags concerning the setup procedure of the coded path (`Flags`).

The `CodingType` field carries a simple identifier of the coding technique that has to be applied at Encoding LSRs. The meaning of these identifiers has to be unique within a NC MPLS domain.

As mentioned, the flags are used to signal important parameters during the path setup. Their meaning usually differs for different coding techniques. For the butterfly NC we implemented the following flags:

- `0x01` *Sequence numbers active.* The Numbering LSR sets this flag. This will usually be the LSR where the path is multicasted. It is required to check whether encoding and decoding is possible and to prevent multiple Numbering LSRs.
- `0x02` *Coding active.* The Encoding LSR enables this flag when the encoding operation is active. It is required to prevent multiple Encoding LSRs and to signal active encoding to potential Decoding LSRs to enable buffering. Furthermore, intermediate LSRs, e.g., R2 in Fig. 3, react on two LSPs sharing the same `Coding-Group Originator ID` and `Coding-Group ID` which both have the active coding flag set. In this case both paths must be merged to multicast the encoded packets.

`CODING_POINTS.` This object contains a list of LSRs which are permitted to take over the role of the Encoding LSR. LSRs listed in this object watch for existing and future LSPs to enable the encoding operation whenever possible. The decision which flows are coded depends on the LSPs' `CODING_PARTNERS` objects. The format is identical to the `EXPLICIT_ROUTE` object in RSVP-TE.

`DECODING_POINTS.` This object has the same format as the `CODING_POINTS` object. It defines a list of LSRs which are able to decode data flows which have been jointly encoded with this flow, i.e., which will receive a plain copy of this flow. Each LSR which is contained in this list looks for other flows that are jointly encoded to take the role of the Decoding LSR.

`PATH_TIME.` Packets that are required for decoding traverse two different paths through the network. As these two paths will usually have different delays, the packets going over the faster path have to be buffered at the decoder until the corresponding packets for decoding arrive. This difference in delay and the available buffer size limit the maximum possible data rate.

As it is required to take this into account for resource reservations, this difference in delay has to be estimated on both branches of a coding path, e.g., on path S1→D2 and S2→R1→R2→D2 in the sample topology in Fig. 1(c). This is done with the `PATH_TIME` object. It contains two fields, a *minimum time* and a *maximum time*. Every LSR which forwards the `PATH` message adds the minimum and maximum observed latency for the link over which it received the message. Hence, at the decoder, the maximum delay difference of both flows is available to calculate whether the data rate requirements of the path can be fulfilled with the available buffer space or not. If not, the decoder must deny the path setup.

The required multicasting of a data flow is realized with the standard P2MP extension for RSVP-TE. LSRs where the actual copying of the flow is desired are configured in the `S2L sub-LSP descriptor list` [8, Section 4.5].

*Modified RESV message.* When a `PATH` message arrives at the egress LSR, RSVP-TE sends back a `RESV` message to the ingress `LSR` to actually set up the LSP and distribute labels between the routers. For the NC-enabled paths, i.e., where the `coding active` flag is set in the `PATH` message, labels carrying the packet sequence numbers will be added to the packets. As Encoding LSRs will receive two flows which have been augmented with sequence numbers, the encoded output flow contains both flows' sequence numbers (cf. Sec. 3.2). Hence, the Decoding LSRs must be aware of which sequence number label belongs to which flow. To signal this, the `RESV` message is extended with a `Coding partner list`.

`Coding partner list.` To retain the approach of RSVP-TE to signal labels from the destination to the source, a Decoding LSR signals the desired label ordering to the Encoding LSR. This is done by adding a `Coding partner list` to `RESV` messages. The list contains pairs of `CODING_PARTNER` and `STACK_POSITION` objects. The `CODING_PARTNER` objects are identical to the ones used in the `PATH` message and identify the partner flow which is used for encoding. The stack position is the expected index of the partner flow's sequence number labels within the label stack. For butterfly NC, the `Coding partner list` will only contain one entry (for the partner flow).

*Signaling costs.* The number of signaling messages required to set up two NC LSPs is analyzed in this paragraph. The setup procedure can be divided into three phases: setup of the first LSP, setup of the second LSP, and refreshing the first LSP to activate coding. Eq. 1 shows the set of `PATH` messages $P_1$ and Eq. 2 the set of `RESV` messages $R_1$, sent during the first phase (cf. Fig. 3(a)). A capital $P$ stands for a `PATH`, an $R$ for an `RESV` message. Indices denote the link over which the message is sent.

$$P_1 = \{P_{\mathrm{S1}\to\mathrm{D2}}, P_{\mathrm{S1}\to\mathrm{R1}}, P_{\mathrm{R1}\to\mathrm{R2}}, P_{\mathrm{R2}\to\mathrm{D1}}\} \tag{1}$$

$$R_1 = \{R_{\mathrm{D2}\to\mathrm{S1}}, R_{\mathrm{D1}\to\mathrm{R2}}, R_{\mathrm{R2}\to\mathrm{R1}}, R_{\mathrm{R1}\to\mathrm{S1}}\} \tag{2}$$

As the first two phases are symmetric, they require the same amount of signaling messages, i.e., $C(P_2) = C(P_1) = 4$ and $C(R_2) = C(R_1) = 4$, where $C(X)$ denotes the cardinality of the set $X$. This means that there are 16 signaling messages sent in total for setting up the two initial LSPs.

For refreshing the first LSP to also activate coding, additional signaling is necessary. The `PATH` and `RESV` messages that are used for this are listed in Eq. 3.

$$P_3 = \{P_{\mathrm{R1}\to\mathrm{R2}}, P_{\mathrm{R2}\to\mathrm{D1}}\} \qquad R_3 = \{R_{\mathrm{D1}\to\mathrm{R2}}, R_{\mathrm{R2}\to\mathrm{R1}}, R_{\mathrm{R1}\to\mathrm{S1}}\} \tag{3}$$

This results in a total of 21 signaling messages. Although this is more than the required 12 messages for the uncoded case, many of them are exchanged in parallel, such that the temporal overhead is very small. Furthermore, none of the two LSPs has to wait for the partner flow, i.e., data transfer can start immediately after path initiation.

### 4.2 Tearing Down Network-Coded Paths

Tearing down one of the jointly encoded paths is straightforward. The source LSR of the LSP to be torn down sends a `PathTear` message down the path. The Encoding LSR receives this message and stops encoding both flows from then on. As packets of the partner path still have to reach their destination, they are sent uncoded and the special sequence number 0 is used to signal this to the decoder (cf. Sec. 3.2).

Using this method guarantees a disruption-free operation of paths whose partners are torn down during NC operations. Depending on the application scenario, a conversion of the multicast LSP to a unicast LSP after such an event might be reasonable to save capacity. This is the case if the multicast capability has only been set up to do NC and is not further required. Such a conversion is done by the ingress LSR by additionally setting up a normal unicast LSP which uses the already allocated network ressources. After switching the traffic to the unicast LSP, the NC multicast LSP is torn down.

## 5 Evaluation

To demonstrate the feasibility of our protocol extensions, we implemented them in a simulator. The system model and the observed results are discussed next.

### 5.1 System Model

We implemented the topology already presented in Fig. 1(c) using the OMNeT++ discrete event simulator and the MPLS and RSVP-TE models from the INET framework. The data sources and destinations are outside the butterfly. All links are equal and have a capacity of 10 Mbit/s and a latency of 2 ms. The sources, connected to S1 and S2, periodically send packets of 1435 byte to the destinations, which are connected to D1 and D2, with inter-packet times of 13 ms. This is the minimum possible inter-packet time before congestion occurs at the chosen link configuration. RSVP-TE refreshes paths every 5 s.

To demonstrate NC, the following events occur at the simulated times $t$:

- $t = 0$ s: S1 creates a normal LSP without coding functionality (LSP1-1). The source at S1 starts sending packets which are transported through LSP1-1.
- $t = 4$ s: S1 creates an NC-enabled LSP (LSP1-2) in parallel to LSP1-1.
- $t = 4.5$ s: The data packets from S1 are switched from LSP1-1 to LSP1-2. LSP1-1 is torn down. The encoding of LSP1-2 is still inactive.
- $t = 5$ s: S2 creates an NC-enabled LSP (LSP2-1). This triggers the joint coding for LSP1-2 and LSP2-1 at R1.
- $t = 6$ s: Source at S2 starts sending packets.
- $t = 15$ s: Source at S1 stops sending packets. LSP1-2 is torn down.

During the simulation, the end-to-end delay $d$ is measured for each packet to see changes in the packet forwarding behavior. Furthermore, packet loss is monitored to detect possible problems during path switching. Nevertheless, the point of this simulation is *not* a stochastic performance evaluation but rather serves to validate the protocol. Therefore, a variation of parameters is not necessary.

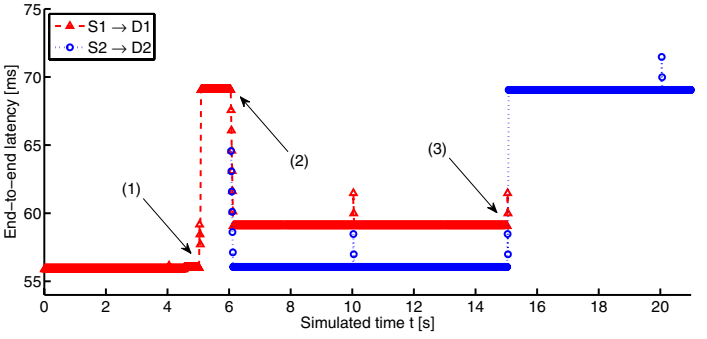



**Fig. 4.** End-to-end delay $d$ over simulated time $t$

### 5.2 Results

The end-to-end delays of both flows, measured during an example simulation run, are plotted in Fig. 4. No packet was lost during the simulated time.

When S1 sends its packets via the uncoded LSP1-1, an end-to-end delay of $d = 56\,\text{ms}$ can be observed. This corresponds to the sum of five link delays plus the corresponding packet transmission times. At event (1), enabling NC for LSP1-2 causes $d$ to raise to 69 ms. The difference is exactly the maximum buffering time of each packet at the encoder before it is sent uncoded.

At (2), data is sent via LSP2-1. LSP1-2's latency reduces again as now packets can be encoded before the maximum encoder buffering time elapses. Whereas LSP2-1's delay reduces to the minimum possible 56 ms, the delay of LSP1 stays at approximately 59 ms. This offset compared to LSP2-1 results from S2's packets arriving slightly later at the Encoding LSR than those of S1. Hence, as both sources send with identical rates, the packets of LSP1-2 have to wait until a partner packet of LSP2-1 arrives. Obviously, this changes in other simulation runs if the starting offsets of S1 and S2 are different.

The teardown of LSP1-2 happens at event (3). As from then on, packets of LSP2-1 have to wait for the maximum encoder buffering time until they are sent out uncoded, the delay of LSP2-1 raises to the already previously seen 69 ms. As LSP2-1 is not converted to disable the NC, the delay remains at this level.

Another phenomenon which can be seen every 5 s (at 5, 10, 15, and 20 s in Fig. 4) is a short increase of the end-to-end delay. It is caused by the path refresh messages (`PATH` and `RESV`) which are periodically sent and have to be queued in addition to the data packets. This temporarily increases the delay.

The absence of any packet loss shows that our protocol was able to switch between different operation modes without disrupting ongoing transfers. The observed packet delay corresponds to the expected behavior and demonstrates the operability of our approach; it behaves accordingly for different initial conditions.

## 6 Conclusion

We presented an extension to the MPLS and RSVP-TE protocols to support creating network-coded paths. The decision whether to use coding or not does not have to be made during the paths' initiation; seamless switching between non-coded and coded operation is possible. Another advantage of the protocol extension is that only those LSRs have to be modified which are actually involved in the coding operation. All others, e.g., intermediate routers which forward coded packets, can remain untouched. Compared to traditional packet forwarding, our extension enables new load distribution schemes which are especially useful for the operation of core networks.

## References

1. Fragouli, C., Boudec, J.Y.L., Widmer, J.: Network coding: An instant primer. SIGCOMM Comput. Commun. Rev. 36(1), 63–68 (2006)
2. Katti, S., Rahul, H., Hu, W., Katabi, D., Médard, M., Crowcroft, J.: XORs in the air: Practical wireless network coding. SIGCOMM Comput. Commun. Rev. 36(4), 243–254 (2006)
3. Chaporkar, P., Proutiere, A.: Adaptive network coding and scheduling for maximizing throughput in wireless networks. In: MobiCom, pp. 135–146. ACM, New York (2007)
4. Gkantsidis, C., Rodriguez, P.R.: Network coding for large scale content distribution. In: INFOCOM 2005, vol. 4, pp. 2235–2245 (2005)
5. Wang, C.C., Shroff, N.B.: Beyond the butterfly – a graph-theoretic characterization of the feasibility of network coding with two simple unicast sessions. In: Proc. IEEE Intl. Symp. on Information Theory, Nice, France (June 2007)
6. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard) (January 2001)
7. Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G.: RSVP-TE: Extensions to RSVP for LSP Tunnels. RFC 3209 (Proposed Standard), Updated by RFCs 3936, 4420, 4874, 5151 (December 2001)
8. Aggarwal, R., Papadimitriou, D., Yasukawa, S.: Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs). RFC 4875 (Proposed Standard) (May 2007)