Abstract

Circuit networks, especially optical circuit networks, are characterized by high bandwidths and latencies. This thesis shows how circuit caching can improve connection latencies for some traffic patterns – particularly ones that exhibit locality in either the source or destination node sets.

A circuit is cached by allowing all, or part, of an established circuit to persist, to be potentially reused, after a typical circuit-switched network would have torn it down. A static function giving the lowest latency path for a message does not exist because switches are not assumed to be reset after every message, so message paths must be dynamically calculated. A packet network is run in parallel with the circuit-cached network to further improve latencies and caching performance. Simulations are used to study the topology, caching, and traffic patterns in these circuit-cached hybrid networks.

TABLE OF CONTENTS

GLOSSARY

Note: a superscript uppercase ‘P’ or ‘C’ denotes that a symbol belongs to the packet or circuit sub-network respectively. The ‘P’ and ‘C’ modifiers only appear as superscripts. Likewise, a superscript uppercase ‘I’, ‘A’, or ‘E’ denotes that a symbol belongs to the idle, active, or expired state respectively on the circuit sub-network. The ‘I’, ‘A’, and ‘E’ modifiers also only appear as superscripts. Superscript uppercase ‘H’ and ‘T’ denote that a symbol belongs to the tunnel head and tail sets respectively. ‘H’ and ‘T’ appear only has superscripts. Words ‘TUNNEL’ and ‘MIDDLE’ also appear as superscripts and denote the ‘TUNNEL’ and tunnel ‘MIDDLE’ set identifiers.

Note: single lowercase letter subscripts are used as array indexes to denote different individual members of a set. Subscripted words or word fragments indicate properties of the item being modified by the subscript.

Note: Except where noted, normal uppercase letters denote sets, and normal lowercase letters denote members of the set of the corresponding uppercase letter.

Symbols that do not follow the uppercase-lowercase / set-set member convention:
D – diameter of the network
  d – number of links traversed
w – number of wavelengths, colors, or channels on a link

Symbols that follow the uppercase-lowercase / set-set member convention:
M – the set of messages
  m – an individual message
    m_{length} – the length of a message

network parameters
α – setup delay
  α^P – setup delay for packet network
  α^C – setup delay for circuit network
δ – switching or routing delay
  δ^P – packet routing delay
  δ^C – circuit switching delay
τ – inverse of the data rate (in seconds per bit)
  τ^P – data rate for packet network

τ^C – data rate for circuit network
t_{link_prop} – link propagation delay

network physical hardware
Λ – the set of wavelengths or colors or channels (on the links)
λ – a wavelength on a link on the circuit network
Λ^I – set of idle wavelengths ($\Lambda^I \subseteq \Lambda$)
Λ^A – set of active wavelengths ($\Lambda^A \subseteq \Lambda$)
Λ^E – set of expired wavelengths ($\Lambda^E \subseteq \Lambda$)
L – the set of links connection routers
L^P – subset of links connecting packet routers ($L^P \subseteq L$)
ℓ^P – link connecting packet routers
L^C – subset of links connecting circuit routers ($L^C \subseteq L$)
ℓ^C – link connecting circuit routers
N – the set of networks
n^P – packet sub-network
n^C – circuit sub-network
R – the set of routers
R^P – subset of packet routers ($R^P \subseteq R$)
r^P – packet router
R^C – subset of circuit routers ($R^C \subseteq R$)
r^C – circuit router

Network policies
Φ – set of routing policies
Φ^P – subset of routing strategies on the packet network ($\Phi^P \subseteq \Phi$)
ϕ^P – a routing strategy on the packet network
Φ^C – subset of routing strategies on the circuit network ($\Phi^C \subseteq \Phi$)
ϕ^C – a routing strategy on the circuit network
b – a distribution strategy used by a node s

Circuit-cached-network-specific symbols
Θ – set of circuits in n^C
θ – a circuit in Θ
Θ^A – subset of active circuits ($\Theta^A \subseteq \Theta$)
Θ^E – subset of expired or cached circuits in n^C ($\Theta^E \subseteq \Theta$, $\Theta^E \cap \Theta^A = \emptyset$)
Θ^{TUNNEL} – subset of expired routes selected to be tunnel candidates ($\Theta^{TUNNEL} \subseteq \Theta^E$)
S – set of nodes on the network
s – a node on the network
s_{source} – a source node
s_{sink} – a destination node
S^H – subset of tunnel heads ($S^H \subseteq S$)
s^H – a tunnel head

S^T – subset of tunnel tails ($S^T \subseteq S$)
s^T – a tunnel tail

CHAPTER 1: INTRODUCTION AND MODEL

1.1 Introduction:

As network link capacity increases, the relationship between network transmission and switch speeds, and router and node processing speeds changes. There was a set of design tradeoffs in the past when network transmission speeds were very slow compared to processor technology. For example, it was possible for routers to examine every packet that passed through them. It is still possible to route packets at high speed today, but the packet routers necessary to do this are expensive.

One way to decrease the cost of the network while still running a high speeds is to reduce the processing performed at each router. This can be done with circuit routers. More specifically, optical circuit networks allow very high throughputs, but they are more difficult to utilize efficiently. Cheaper and simpler optical networks with no wavelength conversion capabilities are especially difficult to utilize efficiently given an unknown and dynamic traffic load.

Older network designs are often not able to fully exploit the capabilities of current equipment because the assumptions that they are based on have changed (one such change is that network link capacity continues to increase relative to processor technology). Therefore, either the original assumptions have to reassert themselves (possible if low cost wavelength conversion or optical buffering and processing become available), or different designs must be used. Connection caching as presented in this thesis is a method of exploiting the tradeoff between circuit setup latencies and utilization efficiency so that overall performance can improve when network link capacities greatly exceed the processing capacities of network nodes.

Decreasing circuit set-up latencies can be beneficial <<reference>>. In the case where circuit setup time approaches total transmission time, the effect of decreasing circuit setup time becomes noticeable <<reference>>. This is especially true in, for example, a shared memory multiprocessor system where network requests are relatively short, but occur frequently. It should be noted that decreasing circuit set-up latencies by itself does not justify changes in network design because these latencies can sometimes be hidden by appropriate buffering on the connected nodes <<reference>>. However, the buffering must often be tuned to the software, or the software does not require the nodes to interact in a fine-grained manner anyway.

Circuit-type and packet-type networks represent different design tradeoffs and each is suited to different classes of traffic patterns. Attempts to incorporate the strengths of both circuit-type and packet-type networks in a unified fashion have resulted in designs such as wormhole routing, virtual-cut-through routing, and pipelined circuit switching <<reference>>. The strengths of both circuit-type and packet-type networks can also be realized by the hybrid networks that this thesis studies where there is a circuit switching network working in parallel with a packet switching network. The costs of such a hybrid network are expected to be greater than any single approach or optimization just mentioned; neither will the hybrid network behave the same way, but it is a good way to isolate different factors that determine network performance.

These hybrid networks need not be excessively more expensive than their unified counterparts with appropriate selection of implementation tradeoffs. They also allow improvements to each component sub-network to contribute to global performance improvements. In fact, a hybrid network may be constructed out of a combination of existing networks such as wormhole, virtual-cut-through, and pipelined circuit. However, the extremes of the circuit-type and packet-type networks have been chosen as components in this thesis for simplicity, and to better illustrate the method of caching preexisting circuits until the time when they may be used again.

This thesis will first describe the model introduced above: both the hybrid aspect, and the connection caching aspect. Assumptions, justifications, and parameters will be stated. Next, the experimental design and results will be presented. Following will be a discussion and extrapolation of the model according to experimental results where the policies presented in this thesis are extended to such topics as clustering and the pattern of the temporary paths formed as a by-product of the caching. Finally, future work that may be done to extend the techniques presented in this thesis is stated.

1.2 Model

The idea of virtual circuit caching has been previously proposed and studied in <<references>> where long-duration virtual circuits (or paths) were created and destroyed under software control. Those virtual circuits were ‘cached’ by keeping them open in anticipation of future traffic. In addition, packet-circuit hybrid networks have been previously proposed and studied in <<references>>.

The networks in this thesis are a combination of circuit-type and packet-type component networks. However, our networks are parallel with each other and exist to complement the caching aspect of the circuit component network. This is reflected in the proposed implementation.

The circuit caching previously proposed in <<reference>> is extended by allowing partial-path caching on the circuit network instead of restricting the path reuse to a complete pre-existing path (<<reference>> introduces virtual circuit caching, but it is less developed than <<reference>>). This thesis also combines the circuit-cached network with a packet network instead of a pre-allocated set of channels that are used to form short-duration circuits <<reference>>.

Our hybrid network has different properties from the virtual circuit caching presented in <<reference>>, but retains the goal of lowering communication latencies. Our version of circuit caching is implemented on a (simulated) physical network using different wavelengths to carry different channels instead of relying on the use of virtual circuits (although there is no reason why it will not work on a virtual network). This is a more difficult problem to solve than the previously proposed virtual circuit caching because it has to obey the constraints of the simple optical network (end-to-end light-paths, and no wavelength conversion), and not rely on the use of virtual channels that don't actually use physical resources when they are idle.

The cached virtual circuit network in <<reference>> relies on application software to allocate and tear down long duration paths. It was, in effect, a higher level application problem more analogous to virtual memory page replacement algorithms than chip-level hardware caches. Our network has automated caching behavior at the hardware level instead of relying on application software to determine when, between which nodes, and for how long, long-duration paths are open.

The next subsections present different aspects of the model. The different symbols used can also be found in the glossary.

1.2.1 Cost Model

In a packet routed network, the time cost of routing a message (assuming 1-bit-wide links) can be expressed as $d * \{t_{routing} + [m_{length} * (t_{switch} + t_{wire})]\}$ where d is the number of hops from source to destination, $t_{routing}$ is the routing decision latency (in seconds), t_{switch} is the time necessary to pass a bit of the message across the router (in seconds per bit), t_{wire} is the time necessary to pass a bit of the message across the link (in seconds per bit), and m_{length} is the length of the message (in bits) <<reference>>. In a circuit-switched network, the time cost can be expressed as $\alpha^C + (d * \delta^C) + (m_{length} * \tau^C)$ where α^C is the start-up time to set up the communications (in seconds), δ^C is the switching delay for the switch (in seconds), and τ^C is the time necessary to pass a bit of the message across the channel (in seconds per bit) <<reference>>. If, for the packet network, we assume that $(t_{switch} + t_{wire})$ always occurs together, we can define $\tau^P = (t_{switch} + t_{wire})$. Adding in a packet setup time α^P (in seconds), and assuming that the inter-router links on both the packet and circuit sub-networks are the same length, and so have the same propagation delay, t_{link_prop} (in seconds); and redefining $t_{routing}$ as δ^P, we can rewrite the above expressions using a more unified symbol set.

The packet network routing time becomes $\alpha^P + (d * (\delta^P + t_{link\_prop} + (m_{length} * \tau^P)))$. Circuit network routing time becomes $\alpha^C + (d * (\delta^C + t_{link\_prop})) + (m_{length} * \tau^C)$.

For most current circuit-switched systems, α^C, δ^C, and $(m_{length} * \tau^C)$ are relatively closely matched so that there is not an overwhelming advantage to optimizing any single constant. This thesis explores the situation where $(d * \delta^C) >> (m_{length} * \tau^C)$. Although most current production systems do not behave in this manner, future fiber optic circuit-switched systems will satisfy this relationship if they continue along their current technology trends.

Currently, 32 to 64 channel WDM circuits are available at 10 to 20 GBps <<reference>>. With the advent of VCSEL or other similar integrated signaling technology, it is likely that total optical fiber transfer rates will move upwards toward the estimated 100 TBps theoretical link capacity <<reference>>. At the same time, optical circuit switching speeds δ^c, either electrical or micro-mechanical cannot benefit from the channel and link parallelism that allows data rates to scale up so rapidly. I.e. it takes at least the same amount of time to switch one wavelength on a fiber optic cable as it takes to switch many wavelengths or wavebands.

1.2.2 Assumptions and Constraints

The $(d * \delta^C) >> (m_{length} * \tau^C)$ relationship strongly suggests that implementation will be on an optical network (because optical networks are capable of the highest data rates, but suffer from slow switching speeds). While optical buffers and frequency shifters may become widely available one day, they are currently expensive <<reference>>. If they do become widely available, it is likely that current network designs can be scaled up because current electrically-signaled circuit-switched network designs will be directly applicable. We will assume for this thesis that frequency shifting is not available, and that circuit switching is done using MEMs-type technology or the equivalent <<reference>>. This implies the following constraint:

There shall be one color or wavelength λ running at a speed defined by τ^C per established circuit θ_i in the circuit network n^C, and a newly formed circuit θ_j using the same wavelength λ may not overlap an existing circuit $\{\Theta - \theta_j\}$. I.e. for $n^C = (R^C, L^C)$ where R^C is the set of circuit routers and L^C is the set of links between the routers, if $(\ell^C_k, \lambda) \in \theta_i$ then $(\ell^C_k, \lambda) \notin \theta_j$ (where ℓ^C_k is a link in L^C, and

(ℓ^C_k, λ) is a link-wavelength pair participating either in circuit θ_i or circuit θ_j). For the purposes of the circuit sub-network in this thesis, there shall be exactly one color or wavelength per channel (but many wavelengths per link), so a channel is equivalent to a color or wavelength.

Each circuit router $r^C \in R^C$ can transmit on any wavelength λ available to it, and any r^C can use any λ in Λ. This keeps all the circuit routers uniform. For simplicity, we assume that r^C has the ability to simultaneously receive every λ on every incoming ℓ^C. All r^C have identical switching times, and all ℓ^C have identical speeds and latencies. I.e. R^C and L^C are uniform.

For practical purposes (to avoid running extra fiber optic cable), the packet network n^P can be implemented on one (or a few) of the wavelengths available to the links L^C in the circuit network n^C, although the physical routers r^P for n^P are separate from the physical routers r^C in n^C. There is no constraint on the number of wavelengths made available to n^P, so the design can be varied such that n^P can use anywhere from a fraction of the fiber bandwidth to almost the entire fiber bandwidth (but this fraction is a static value). However, as already mentioned, high speed packet routing hardware is expensive, and so this thesis will assume that only a small fraction of the fiber bandwidth is used. This means that the maximum packet transmission speed is limited by the speed of a small multiple of τ^C. Because circuit control messages for the circuit sub-network n^C will likely be implemented using a variant of the packet subsystem (common or similar implementation technology), they will operate with comparable parameters to the packet sub-network n^P. Even so, the δ^{P} for n^P will likely to be the dominant factor even if it isn't as dominating as the δ^{C} for n^C.

An effect of limiting circuit control message speeds to be similar to the packet subsystem is that it makes comparing the effects of the circuit network easier because it can not 'cheat' by assuming that it has inherently faster control message processing. Another advantage to this approach is that we can tie the limits of packet transmission speed to the limits of processing speed: the circuit network control links cannot operate faster than the controllers controlling the switching.

A circuit router r^C can switch an incoming wavelength λ on an incoming link $\ell^C{}_i$ on the circuit network n^{C} onto any one outgoing $\ell^C{}_j$ provided that that the incoming λ isn't already used on that outgoing $\ell^C{}_j$. Note that r^C may not switch out an existing λ on the incoming $\ell^C{}_i$.

1.2.3 Hybrid networks

As mentioned earlier, hybrid networks composed of circuit-type and packet-type sub-networks have been studied before <<reference>>. One justification for these networks is that an existing packet-based electrical network and an optical backbone can be combined. This thesis does not examine the case were the sub-networks do not implement the same topology; instead it assumes that the sub-networks are implemented on the same physical infrastructure which gives rise to the same topology for both sub-networks. Circuit-caching, in particular, benefits from this hybrid architecture. This can be shown as follows:

Consider a circuit network with a given α^C, δ^C and τ^C. If a message sent through the network were short such that $(d * \delta^C) >> (m_{length} * \tau^C)$, the $(d * \delta^C)$ parameters dominate the time cost of the message. If the same message were sent over a packet network with the same α and τ ($\alpha^P = \alpha^C$ and $\tau^P = \tau^C$), but δ^P is such so that $\delta^P < \delta^C$, and $\delta^P >> (m_{length} * \tau^P)$, then $\alpha^P + (d * (\delta^P + t_{link\_prop} + (m_{length} * \tau^P))) < \alpha^C + (d * (\delta^C + t_{link\_prop})) + (m_{length} * \tau^C))$. δ^P is the reason why the message cost decreased, so it would be advantageous for a packet network to route short messages while a circuit network is reserved for longer messages. In fact, this bi-modal nature is illustrated by the IBM SP/2 switch implementation <<reference>>.

A circuit network with circuit-caching capability has a certain ‘hit’ rate: the probability that an already-established circuit would be reused (this probability is discussed in the results section). Every time a cached circuit is reused, the cost of the new circuit is reduced to $\alpha^C + (d' * (\delta^C + t_{link\_prop})) +$ (length_of_reused_path $* t_{link\_prop}) + (m_{length} * \tau^C)$ from $\alpha^C + (d * (\delta^C + t_{link\_prop})) + (m_{length} * \tau^C)$ where d' is the new number of links traversed and $d' > d$ (because if $d' \leq d$ then caching wouldn’t have been advantageous in the first place), and the length_of_reused_path function denotes the number of hop or routers spanned by the reused circuit. The number of links traversed by d' is the number of hops between the source router to the tunnel head router plus the number of hops between the tunnel tail router and the destination router.

The differences come from the changes in number of routers traversed because of the ($d * \delta^C$) >> ($m_{length} * \tau^C$) relationship. It is advantageous to use a cached circuit if the (length_of_reused_path * t_{link_prop}) additional delay isn't more than the savings provided by (($d - d'$) * ($\delta^C + t_{link\_prop}$)), which is likely the case if δ^C is the dominating factor.

If many messages are sent over the circuit-cached network, eventually there will be a saturation point where most circuits being formed by new messages result in an existing cached circuit being torn down because the existing cached circuits do not contribute favorably to the setup time of the new circuits. This is analogous to elements being evicted from a full memory cache. Clearly, if all messages were short, the average lifetime of a cached circuit will be shorter (in absolute terms) than if all messages were long. This is true even for the ($d * \delta^C$) >> ($m_{length} * \tau^C$) case.

If we want to achieve better utilization of cached circuits, only long messages should be allowed to use the circuit network. Short messages should be diverted to the packet network as long as $\alpha^P + (d * (\delta^P + t_{link\_prop} + (m_{length} * \tau^P))) < \alpha^C + (d * (\delta^C + t_{link\_prop})) + (m_{length} * \tau^C)$, or in the case of circuit caching being used, $(\alpha^P + (d * (\delta^P + t_{link\_prop} + (m_{length} * \tau^P)))) < \alpha^C + (d' * (\delta^C + t_{link\_prop})) +$ (length_of_reused_path * t_{link_prop}) + ($m_{length} * \tau^C$). The main advantages of the circuit-cached sub-network over the packet sub-network in this case are the fewer switches that need to be set between the source and destination nodes (d' where $d' < d$), and a higher data transfer rate (better τ^C) compared to the packet sub-network's advantage of a lower ($d * \delta^P$) which gives a lower routing overhead for the same number of hops between source and destination nodes. Using the packet network for short messages keeps cached routes cached for a longer period of time to be used by long messages that can better exploit the advantages of a faster data transfer rate (smaller value of τ^C). Again, the packet sub-network could have been replaced by a system similar to the set of short-lived circuits as described in <<reference>>.

Finally, if τ decreases relative to δ for some network, i.e. the ratio of transfer times versus switching times decreases, the definition of what constitutes a short message changes. For example, in a circuit-switched network, a message that used to be 'long' when ($d * \delta^C$) > ($m_{length} * \tau^C$) may become 'short' when ($d * \delta^C$) >> ($m_{length} * \tau^C$). It is important to divert short messages to the packet network to avoid overloading the circuit-cached network with messages that gain little from the strengths of the circuit-cached network. However, if current message length distributions remain the same, a greater proportion of messages will be routed over the packet network (because there are more 'short' messages) saving the circuit-cached network for the long messages that it can handle best.

1.2.4 Parameters

Parameters that will be used in the experimental design section are described here (parameter values will be given in the experimental design section). These include α, δ, and τ as mentioned above; different routing strategies, network topologies, traffic patterns, and caching strategies. These parameters can all be changed because the requirements of the hybrid networks described above are not very restrictive; neither are the requirements for circuit-caching restrictive.

The parameters α, δ, and τ are constrained to roughly reflect values that have been reported in real or expected hardware. There is no logical restriction on their values except that some values of α, δ, and τ are not very realistic, nor are they very interesting. The α parameter will not be varied much and for the most part will be ignored. This is because the dominant costs in large diameter networks will be in δ and τ. Even though the simulations used were not capable of very large diameters, the relative effects of δ and τ can still be measured. The $(d * \delta^C) >> (m_{length} * \tau^C)$ relationship holds throughout most of the study.

Routing strategies determine the path taken by messages through each sub-network. Each sub-network may have its own routing strategy because each is independent of the other. Routing strategies are chosen such that they can be implemented without centralized control. The only other requirement on routing strategies is that they be deadlock-free (given the thesis goal of low latencies, the routing strategies should also ideally generate minimal routes). Each sub-network is independently routed and is only coupled to the other at the source and destination node endpoints so if each sub-network is deadlock-free, then the (combined) hybrid network is also deadlock-free. These are not strict requirements: a different implementation may have centralized routing control and non-deadlock-free routing, but this thesis applies these constraints because it more closely reflects the implementation ideals of a distributed network.

Different network topologies are used to determine the effects of degree and diameter on the network. This applies mostly to the caching aspects of the circuit network since both circuit and packet networks individually are well understood already.

Different traffic patterns are used to the same effect. The traffic patterns chosen are random, real traces, and idealized traces. These patterns are all pre-generated before being used as input data to the simulator. Random and real traces are used to show the expected performance of the network, and idealized traces are used to show what the network is capable of.

Caching strategies only apply to the circuit sub-network. These include cached route replacement policies and cached route flushing policies. Caching strategies modify the behavior of the circuit sub-network and the effects are observed while the packet sub-network remains fixed. In fact, the packet sub-network serves primarily in a traffic shaping role to filter out short messages from the circuit sub-network to allow observation of long messages only.

1.3 Protocol

1.3.1 Messages

<see diagram 1, 2>

A message *m* is created by a source node and knows its destination when it is created (multicast is not considered.)

A message is passed into a source node. The source node moves the message into one of the available networks (packet and circuit in this case). The message traverses the network to the destination node and moves from the network to the destination node and is consumed.

More precisely, a message $m \in M$ is generated at a node $s \in S$, traverses either a packet network n^P, or a cached circuit network n^C. Packet network n^P is formed by links L^P and packet routers R^P, and circuit network n^C is formed by links L^C and circuit routers R^C. L^P and L^C will have different characteristics: the packet network will be low bandwidth, low latency, and the circuit network will be high bandwidth, high latency. L^P contains w^P channels and L^C contains w^C channels, but since the packet sub-network is not studied closely, this thesis will arbitrarily choose L^P such that it only contains 1 channel.

Each node may have a different policy $b \in B$ on which network to pass a message onto. In our case, *B* is uniformly a constant break-over: messages smaller than size *k* are routed over n^P, and larger than size *k* routed over n^C.

All networks in N do not need to have the same topology, nor does each $n \in N$ need to contain all S. However, things are more interesting when each node participates in all networks. A pair (r^P, ℓ^P) may not intersect (r^C, ℓ^C): each network is completely independent. Each router in a network has a routing policy ϕ (a packet router has routing policy $\phi^P \in \Phi^P$, the packet router routing policies, and a circuit router has routing policy $\phi^C \in \Phi^C$, the circuit router routing policies) which when given a source and sink on that network, can find the shortest path between the two routers. Φ^P generates no cycles for M in n^P, and Φ^C generates no cycles for M in n^C. If m goes through n^P (or n^C), its route is completely determined by the ϕ^P (or ϕ^C). Since ϕ^P (or ϕ^C) can always find a route for m through n^P *(or* n^C*)*, m will always route because of the routing policy.

Routers r^P_i and r^C_i are attached to node $s_i \in S$; no other router is attached to s_i, and s_i has exactly two routers. The two parallel networks n^P and n^C are entirely separated and independent with the same topology. They do not interact with each other, but only to their attached nodes.

1.3.2 Circuit cached network

A circuit-cached network is like a classic-circuit switched network except that routes blocked on setup are torn down after a set time and retried, and the network caches expired routes. This network maintains permanent control connections between each pair of connected routers (R^C_i, R^C_j) where R^C_i is connected to R^C_j by $L^C_{i,j}$.

Θ^A are active circuits in n^C, and Θ^E are expired (or cached) circuits in n^C. Θ^A and Θ^E are disjoint (because circuits θ are created in Θ^A and move to Θ^E when they expire).

<see diagram 3>

A router on this network can be in the following states:

1. idle
2. setup (two phases)
3. active
4. expired route broadcast

section to be reworked with new diagram {1. transitions to 2. on [message received]. 2. to 3. on [circuit set up]. 3. to 4. on [message end]. A circuit is active in states 2,3 and expired in state 4(,1). After expired route timeout, transition from 4. to 1.}

On route setup, a new circuit θ in Θ is created and made a member of the active circuits subset Θ^A. On the transition to [4,] θ moves from the active circuits subset Θ^A to the expired circuits subset Θ^E and expired route broadcast begins. Router r^C_i will broadcast to its neighbors the existence of the cached route $\theta \in \Theta^E$, along with the wavelength λ of the expired route, and the time when the expired route times out.

1.3.2.1 Optical modification

An optical circuit-cached network adds the following behavior: route setup has two phases: forward route reservation, and backward route establishment. This works in a similar manner to standard backtracking circuit setup as described in <<reference>>. We enforce the constraint that θ uses exactly one λ.

1.3.2.2 Forward route reservation

Forward route reservation may be done with either normal setup, or setup using a cached route.

1.3.2.2.1 Normal setup

Define S^H as {tunnel heads} and S^T as {tunnel tails}. A tunnel is $\theta \in \Theta^E$. A tunnel head s^H_i is the node s_i that contains r^C_i on the expired circuit θ_x where the new circuit θ_y enters, and a tunnel tail s^T_j is the node s_j that contains r^C_j on θ_x where the new circuit θ_y exits. The 'H' and 'T' subset modifiers aren't necessary (because, for example, node s^H_i is just node s_i), but they are used to identify the tunnels that are used.

If r^C_i does not have an appropriate caching entry (from an expired route broadcast), normal setup applies. An entry is appropriate if length(s_{source}, s^H_i) + 2 + length(s^T_i, s_{sink}) < length(s_{source}, s_{sink}) where length(s_i, s_j) defines the distance between nodes s_i and s_j. The node s_{source} is the source node, and node s_{sink} is the destination node.

<see diagram 4>

Define Λ^I as {idle λ in ℓ^C}, Λ^A as {active λ in ℓ^C}, Λ^E as {expired λ in ℓ^C}.

Forward route reservation for r^C_i:

1. get output port according to p^C. p^C can also be used to generate a sequence of routers from r^C_{source} to r^C_{sink}

2. get channels in Λ^I_{i-1}, and channels in Λ^E_{i-1} from the previous router (on the circuit) r^C_{i-1} unless r^C_{i-1} is r^C_{source}. If r^C_{i-1} is r^C_{source}, then $\{\Lambda^I_{i-1}, \cup \Lambda^E_{i-1}\}$ contains all channels. If there is contention where channels in $\{\Lambda^I_i \cup \Lambda^I_{i-1}\}$ are already reserved by message m_j, the message with higher priority takes precedence ($m_i \mid i < j$), and the reservations requested by m_j (with lower priority) are dropped, and a teardown message is sent along all ports used by m_j. If m_i is such that $i > j$, then a teardown message is sent along all ports used by m_i.

3. mark $\{\Lambda^I_i \cup \Lambda^I_{i-1}\}$ channels as reserved

    a. put idle channels from r^C_i into idle channels Λ^I_i

    b. also put expired channel from r^C_i into expired channels Λ^E_i

4. pass $\{\Lambda^I_i \cup \Lambda^I_{i-1}\}$, $\{\Lambda^E_i \cup \Lambda^E_{i-1}\}$ to router r^C_{i+1}

    a. if $\{\Lambda^I \cup \Lambda^E\}$ is empty, then tear down the connection attempt by propagating teardown messages back to r^C_{source} by r^C_i sending a teardown message back to r^C_{i-1}. If r^C_{i-1} is r^C_{source}, r^C_{source} will retry the connection attempt.

5. at r^C_{i+1}, go to [1.]

1.3.2.2.2 Setup using cached route

1. Generate set of candidate expired routes from Θ^E such that length(s_{source}, s^H_i) + 2 + length(s^T_i, s_{sink}) < length(s_{source}, s_{sink}) and assign it to Θ^{TUNNEL}. Reduce the size of the set Θ^{TUNNEL} by elimination circuits that do not satisfy the condition length(s_{source}, s^H_i) + 2 + length(s^T_i, s_{sink}) $\leq$ a constant. (The value for this constant will be given later in the experiments and analysis sections.)

2. do normal setup from s_{source} to s^H_i except that the input channel set at r^C_{source} consists of only one channel, λ, the channel that θ_i uses.

3. traverse tunnel

    a. if tunnel traversal is interrupted, then tunnel middle s^{MIDDLE}_k sends a teardown message to s^H_i, which passes the message along r^C_{i-1} back to r^C_{source}

    b. while the tunnel is traversed, the routers r^C_i in the tunnel from s^H_i to s^T_j are locked. Locking occurs immediately after the message propagates past r^C_i.

4. do normal setup from s^T to s_{sink}: if a teardown message is sent, it is propagated by r^C_i, sending a teardown message back to r^C_{i-1} where the tunnel counts as one of the routers r^C

1.3.2.3 Backward route establishment

Once forward route reservation is finished by sending a setup message from r^C_{source} to r^C_{sink}, backward route establishment may execute.

All channels that have been marked reserved along the route by the forward route reservation phase are now marked as free except for the new active channel λ. λ is marked as active along the entire circuit.

A route reply message is sent from r^C_{sink} to r^C_{source} by propagating the message from r^C_i back to r^C_{i-1}. The route reply message contains the fact that λ is the new active channel. When a router receives a route reply message, if channel λ on the router is

a) not already part of a cached route, λ is marked active, and all channels marked reserved by the corresponding setup message are marked free. The route reply message is relayed to r^C_{i-1}.

b) Already part of a cached route, λ is marked active, the portion of the cached route that is not going to be part of the new circuit is flushed and marked free. This should happen only at s^T. The route reply message is relayed directly to the router r^C_i attached to s^H_i which relays it to r^C_{i-1}, and

r^C_i also sends a teardown message to r^C_j that is not on θ_i. Expired circuit θ^E_i that is subsumed by θ_i, and the remaining parts of θ^E_i are moved from Θ^E to Θ^A.

Once r^C_{source} receives the route reply message, all λ in r^C_i on θ_j are marked active and the circuit is established. r^C_{source} sends the message to r^C_{sink}.

When it is done, θ_j will be moved from Θ^A to Θ^E, and all r^C_i in θ_j will broadcast the expired route information to its neighbors as mentioned earlier.

1.3.2.4 On receipt of expired route broadcast

When r^C_i receives an expired route broadcast message for circuit θ_j, it will a) check to see if the expiry time (of the expired route broadcast message) is after the current time, b) check to see if its distance from the originator of the broadcast r^C_k is outside bounds (the bounds are derived in the analysis section), and c) check to see if it already has an entry for a θ_j that was broadcast from r^C_m where length(r^C_i, r^C_m) < length(r^C_i, r^C_k).

If any condition is true, r^C_i does not process the broadcast message further. Else, an entry for θ_j is associated with r^C_k, and the message is re-broadcast to all neighbors of r^C_i. In this way, routers will get an optimal route to any part of θ_j.

1.3.2.5 On receipt of teardown message

If r^C_i receives a teardown message, it will propagate the message along all ports with an associated setup message. This means that both r^C_{i+1} and r^C_{i-1} may receive the propagated teardown message. If a teardown message reaches r^C_{source}, r^C_{source} will retry the message. Teardown messages cause r^C_i to release any resources locked or reserved by the associated setup message.

1.3.2.5.1 Backward propagation of dropped channels

As forward reservation in n^C is performed, channels may be dropped if they are blocked by an intermediary router. If channel λ^i is dropped from router r^C_n, λ^i will also be dropped in all of routers r^C_{n-1} to r^C_0 propagated using the same control channels as the forward reservation used. Backward propagation of dropped channels is an enhancement that improves forward propagation of other circuits being set up.

A cached circuit may be viewed as a special case of backward propagation because only one channel is left in the forward reservation process. Attempted use of a cached circuit on channel λ^i implies that only channel λ^i will be used in the rest of the circuit. This allows the circuit to be set up with minimal interference to any other circuit being set up. As cached circuit reuse becomes common in the steady state of the network, the efficiency of channel use increases <insert stats> because arbitration based on message priority does not have to be performed as often.

1.3.2.6 Message Queuing

Messages are queued after they are produced so that local blocking doesn't affect the network too much. Queuing is done by both the distribution policy and the routers themselves. One disadvantage to queuing is that the time from message creation to message delivery is increased by the length of the queue at steady state. Queuing is mostly a technical solution to the problem of not knowing what the next produced message will be: if the attached r^P has just become free, and the new message m_i of length greater than the limit is created, it will either block r^P, or have to be queued onto r^C in the hopes that another message m_j will have length less than the limit.

1.3.2.7 Distribution strategy

Each node s in the network has a distribution strategy b that it uses to determine which network a message will traverse. b is based on the message size. In this network, a fixed distribution strategy is used where m traverses n^C if size(m) is greater than a preset limit; otherwise, m traverses n^P. b may be made adaptive in the future, or b_i for s_i may be different from b_j for s_j. The fixed distribution strategy allows us to do basic traffic shaping by assigning messages to the network where they are most likely suitable. Note that with the addition of the distribution strategy, messages no longer have to be accepted by both networks even though they may be accepted by either.

1.3.2.8 Switch long messages onto packet network

On the off chance that a long message is made available after the queue for r^C at s_{source} is full, the message will queue onto r^P instead. This will cause r^P to fill up so that it will not signal that it will accept messages until the queue length is less than the limit. For simplicity, we will allow r^P to continue accepting messages even though r^C is full because we are more interested in the behavior of the circuit cached network. This is the only place in the system where r^P may route a message intended for r^C: after the message leaves s_{source}, r^P and r^C do not interact. The condition where both r^P and r^C are full does not happen very often, but the distribution strategy takes that into account for completeness. Although r^P is not well suited to handle long messages intended for r^C, it is allowed to route these overflow messages in the interest of continued message delivery.